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

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

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

```
0000     1          .TITLE  PUDRIVER
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;******************************************************************************
0000     5  ;*                                                                            *
0000     6  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                   *
0000     7  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
0000     8  ;*  ALL RIGHTS RESERVED.                                                      *
0000     9  ;*                                                                            *
0000    10  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED      *
0000    11  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE     *
0000    12  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER      *
0000    13  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY      *
0000    14  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY      *
0000    15  ;*  TRANSFERRED.                                                               *
0000    16  ;*                                                                            *
0000    17  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE      *
0000    18  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
0000    19  ;*  CORPORATION.                                                               *
0000    20  ;*                                                                            *
0000    21  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS      *
0000    22  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                    *
0000    23  ;*                                                                            *
0000    24  ;*                                                                            *
0000    25  ;******************************************************************************
0000    26  ;
0000    27  ;++
0000    28  ;
0000    29  ; FACILITY:
0000    30  ;
0000    31  ;       VAX/VMS EXECUTIVE, I/O DRIVERS
0000    32  ;
0000    33  ; ABSTRACT:  This module contains the UDA port driver.
0000    34  ;
0000    35  ; AUTHORS:  Richard I. Hustvedt,  July 1981
0000    36  ;           Robert Rappaport
0000    37  ;
0000    38  ; MODIFIED BY:
0000    39  ;
0000    40  ;       V03-159 RLRSTEP4        Robert L. Rappaport     16-Jul-1984
0000    41  ;               Expand CNTRLTYP field in Port Step 4 from 4 bits to 7 bits.
0000    42  ;               Also add in Scorpio-BUA support and hooks for BDA support.
0000    43  ;
0000    44  ;       V03-158 RLRQDA          Robert L. Rappaport     19-Jun-1984
0000    45  ;               Add recognition of QDA.
0000    46  ;
0000    47  ;       V03-157 RLRTRACE        Robert L. Rappaport     01-Jun-1984
0000    48  ;               Add support to be able to dynamically configure tracing.
0000    49  ;
0000    50  ;       V03-156 RLRMAYA         Robert L. Rappaport     22-May-1984
0000    51  ;               Add in MAYA tape support.  Also add in ability to just
0000    52  ;               trace "PT" ports.
0000    53  ;
0000    54  ;       V03-155 RLRMVER         Robert L. Rappaport     26-Apr-1984
0000    55  ;               Add code to send operator message to complain about ucode
0000    56  ;               out of date on RDRX controller.
0000    57  ;
```

```
0000     58 ;       V03-154 RLRPDTADP       Robert L. Rappaport      9-Apr-1984
0000     59 ;               1) Init PDT$L_ADP.  2) Initialize PDT$L_WAITQFL list head
0000     60 ;               in INIT_PU_PDT. 3) Clear connection active bit in
0000     61 ;               PDT$B_CONBITMAP on calls to disconnect.
0000     62 ;
0000     63 ;       V03-153 ROW0334         Ralph O. Weber           3-APR-1984
0000     64 ;               Add setup of PDT$L_MAXBCNT to BUILD_PDT.  Give PDT$L_MAXBCNT a
0000     65 ;               value of 127*512 (i.e. 127 blocks).
0000     66 ;
0000     67 ;       V03-152 RLRPHYPGS       Robert L. Rappaport      21-Mar-1984
0000     68 ;               Make use of new system routine to allocate physically
0000     69 ;               contiguous pages.  Also add null routine FPC$STOP_VCS.
0000     70 ;
0000     71 ;       V03-151 PRD0070         Paul R. DeStefano        25-Feb-1984
0000     72 ;               Clear SB$L_CSB (link to newest CSB) when system block
0000     73 ;               is created.
0000     74 ;
0000     75 ;       V03-150 RLRDELPQ        Robert L. Rappaport      25-Jan-1984
0000     76 ;               Eliminate separate conditionalized PQDRIVER and rather
0000     77 ;               use CPUDISP to accomodate differences.  In general,
0000     78 ;               differences are in initialization code or in main line code.
0000     79 ;               For initialization code merely use CPUDISP.  For main line
0000     80 ;               code (MAPIRP and UNMAP), have two sets of entry points,
0000     81 ;               one for all CPU's except uVAX I, and one set for uVAX I.
0000     82 ;               Then at initialization time, in BUILD_PDT, build PDT
0000     83 ;               dispatch table to dispatch to proper entry point for the
0000     84 ;               processor we are running on.
0000     85 ;
0000     86 ;               Also fix bug in uVAX I BUILD_PDT that manifested itself when
0000     87 ;               we have more than one port; namely we tried to map the
0000     88 ;               map registers in the 2nd PDT into the System Addresses
0000     89 ;               pointed at by @ADP$L_CSR+^x800 of the common ADP, where we
0000     90 ;               had already mapped the map registers from the first PDT.
0000     91 ;               Solution is to only go thru map code once.  A new flag,
0000     92 ;               MAP$V_MAPREGS, in PU$L_DRIVER_STS, if set, means that the code
0000     93 ;               has already been executed.
0000     94 ;
0000     95 ;       V03-149 ROW0268         Ralph O. Weber           28-DEC-1983
0000     96 ;               Change instructions used to set aside space for NULL_CDT so
0000     97 ;               that space allocated is always large enough to accomidate a
0000     98 ;               complete CDT.  (I.E. base the space allocation on CDT$K_LENGTH.)
0000     99 ;
0000    100 ;       V03-148 RLREXEALLOC     Robert L. Rappaport      22-Dec-1983
0000    101 ;               Correct subtle bug introduced in previous fix.  Namely
0000    102 ;               the call to EXE$ALONONPAGED was restored and the call to
0000    103 ;               EXE$ALLOCATE was removed.  To deal with the problem of
0000    104 ;               deallocating fragments that might be in LRP space,
0000    105 ;               the deallocation of the PDT fragment (in PQDRIVER BUILD_PDT)
0000    106 ;               was eliminated and it was decided to forget about that small
0000    107 ;               bit of space.
0000    108 ;
0000    109 ;       V03-147 RLRMCRED        Robert L. Rappaport      18-Nov-1983
0000    110 ;               Bring driver into conformance with latest UQPORT spec.
0000    111 ;                       1. Ignore credits field (in envelope) for Maintenance
0000    112 ;                          type messages.
0000    113 ;                       2. Wait for at least 100 uSecs after initialization
0000    114 ;                          interrupts before reading SA to look for step bit.
```

```
0000   115 ;                    3. Test for controller tolerance of odd addresses before
0000   116 ;                       segmenting a transfer and using the aligned buffer.
0000   117 ;
0000   118 ;       V03-146 RLRPQ02         Robert L. Rappaport     14-Nov-1983
0000   119 ;               Clear up problems in non-aligned transfers.  In
0000   120 ;               particular, make it so that we copy WRITE data to
0000   121 ;               the aligned page BEFORE the I/O and we copy READ
0000   122 ;               data from the aligned page AFTER the I/O operation.
0000   123 ;               We accomplish this by adding two subroutines,
0000   124 ;               SETUP_COPY_SEG1 and SETUP_COPY_SEG2, that perform
0000   125 ;               the necessary setup operations to prepare for a copy
0000   126 ;               in either direction.
0000   127 ;
0000   128 ;       V03-145 KDM0104         Kathleen D. Morse       20-Oct-1983
0000   129 ;               Fix reference to MMG$GL_SPTBASE to be PIC.
0000   130 ;
0000   131 ;       V03-144 KDM0103         Kathleen D. Morse       01-Oct-1983
0000   132 ;               Invalidate virtual address after changing the system
0000   133 ;               page table entry to do unaligned transfers.  Allocate
0000   134 ;               a system page table entry in the initialization routine
0000   135 ;               for the Qbus.
0000   136 ;
0000   137 ;       V03-143 NPK3050         N. Kronenberg           30-Sep-1983
0000   138 ;               For PQ allocate PDT so that PDT$L_PQ_MAP within
0000   139 ;               the PDT is page aligned.
0000   140 ;
0000   141 ;       V03-142 KDM0102         Kathleen D. Morse       29-Sep-1983
0000   142 ;               Change Qbus code to compute physical address of
0000   143 ;               ring buffer during initialization.
0000   144 ;
0000   145 ;       V03-141 KDM0101         Kathleen D. Morse       29-Sep-1983
0000   146 ;               Fix indexing through page table entries for Qbus.
0000   147 ;
0000   148 ;       V03-140 KDM0100         Kathleen D. Morse       29-Sep-1983
0000   149 ;               Change the way the ADP points to the map registers
0000   150 ;               for the MicroVAX I.  Fix virtual to physical translation.
0000   151 ;               Add MicroVAX I to CPUDISP macros.
0000   152 ;
0000   153 ;       V03-139 CWH3139         CW Hobbs                17-Sep-1983
0000   154 ;               Change DT$_RC25 symbol for Aztec port to DT$_LESI so
0000   155 ;               that DT$_RC25 can refer to the removable pack on the
0000   156 ;               Aztec.
0000   157 ;
0000   158 ;       V03-138 RLRNEWPB        Robert L. Rappaport     28-Jul-1983
0000   159 ;               Incorporate new PBDEF changes.
0000   160 ;
0000   161 ;       V03-137 RLRQIOCHNLa     Robert L. Rappaport     15-Jul-1983
0000   162 ;               Must refresh R4 => PDT after REQPCHAN.
0000   163 ;
0000   164 ;       V03-136 RLRQIOCHNL      Robert L. Rappaport     6-Jul-1983
0000   165 ;               Correct subtle error in QIO routine.  After REQCOM,
0000   166 ;               PU channel was released.  Next interrupt crashed
0000   167 ;               system.  Fix is to have STARTIO, BSBW to REQCOM, so
0000   168 ;               as to retain control after REQCOM.  In this way we
0000   169 ;               can REQPCHAN again.
0000   170 ;
0000   171 ;       V03-135 RLRSAPOLL       Robert L. Rappaport     5-Jul-1983
```

```
0000   172 ;               Implement periodic polling of the SA register.  Also
0000   173 ;               cleanup some miscellaneous bugs:
0000   174 ;                       1. In HARDWARE_INIT, insert BRB 190$ at the
0000   175 ;                          end of the logic following label TESTUDA_780.
0000   176 ;                          This corrects a bug that was permanently
0000   177 ;                          wasting a buffered datapath on 780's.
0000   178 ;                       2. Add a FUNCTAB +EXE$ZEROPARM for the
0000   179 ;                          STOP and INITIALIZE qio functions that we now
0000   180 ;                          support.
0000   181 ;                       3. In INIT_UDA_BUFFERS, save values placed into
0000   182 ;                          UCB$W_BOFF, UCB$W_BCNT, and UCB$L_SVAPTE in
0000   183 ;                          new UCB fields.  Then in HARDWARE_INIT, restore
0000   184 ;                          these values immediately prior to the LOADUBA
0000   185 ;                          invocation.  This corrects the problem that
0000   186 ;                          the QIO functions cause the these fields to
0000   187 ;                          be cleared.
0000   188 ;
0000   189 ;       V03-134 RLRSRVCN1        Robert L. Rappaport     3-Jun-1983
0000   190 ;               Corect two bugs introduced in previous edit.
0000   191 ;
0000   192 ;       V03-133 RLRSRVCON        Robert L. Rappaport     1-Jun-1983
0000   193 ;               1. Prevent logging redundant Initialization Log entries.
0000   194 ;               2. Correct infinte loop typo in Maintenance Type messages.
0000   195 ;               3. Connect changes:
0000   196 ;                       a) Add PDT$B_SERVERS, bit map that lists servers
0000   197 ;                          supported at this port.
0000   198 ;                       b) Have CONNECT return SS$_FAILRSP if caller is
0000   199 ;                          trying to Connect to a server not supported
0000   200 ;                          on this port.
0000   201 ;                       c) Have CONNECT return R2=>Connect data.
0000   202 ;
0000   203 ;       V03-132 RLRCPUDISP       Robert L. Rappaport     25-May-1983
0000   204 ;               Use new form of CPUDISP macro.
0000   205 ;
0000   206 ;       V03-131 RLRPCHAR         Robert L. Rappaport     20-May-1983
0000   207 ;               Set PDT$M_SNGLHOST bit in PDT$W_PORTCHAR field.
0000   208 ;
0000   209 ;       V03-130 RLRPUR780        Robert L. Rappaport     26-Apr-1983
0000   210 ;               Prevent losing context of which datapath to purge.
0000   211 ;               Do this by saving and then restoring the contents
0000   212 ;               of CRB$L_INTD+VEC$B_DATAPATH when doing a UDA
0000   213 ;               requested purge at device interrupt level.
0000   214 ;
0000   215 ;       V03-029 RLRUDAREV        Robert L. Rappaport     11-Apr-1983
0000   216 ;               Test for out of rev UDA50's and UDA50A's on 780 systems.
0000   217 ;
0000   218 ;       V03-028 RLRSAVCRED       Robert L. Rappaport     8-Apr-1983
0000   219 ;               On Disconnect, save available credits in PDT$W_PU_CREDx,
0000   220 ;               where x (ID) is index of disconnecting connection.
0000   221 ;
0000   222 ;       V03-027 RLRPQ01a         Robert L. Rappaport     31-Mar-1983
0000   223 ;               Fix branch out of range brought about by previous addition
0000   224 ;               which also included setting UCB$B_DEVTYPE according to
0000   225 ;               UQPORT type.
0000   226 ;
0000   227 ;       V03-026 RLRPQ01          Robert L. Rappaport     17-Mar-1983
0000   228 ;               Add conditionally assembled support for Q-BUS port.
```

```
0000   229 ;
0000   230 ;       V03-025 TCM0001         Trudy C. Matthews       28-Feb-1983
0000   231 ;               Update occurrences of CPUDISP macro so that the function
0000   232 ;               correctly on an 11/790.  In both cases, we just take the
0000   233 ;               same code path as the 11/780.
0000   234 ;
0000   235 ;       V03-024 RLRPBSB         Robert L. Rappaport     11-Feb-1983
0000   236 ;               Cleanup minor phasing problem in Path Block and System
0000   237 ;               Block initialization by adding new subroutine, UPDATE_PB_SB.
0000   238 ;
0000   239 ;       V03-023 RLRMSGTYP       Robert L. Rappaport     3-Feb-1983
0000   240 ;               Add ability to handle simple credit type messages and
0000   241 ;               maintenance messages.
0000   242 ;
0000   243 ;       V03-022 RLRDUP1         Robert L. Rappaport     31-Jan-1983
0000   244 ;               Fix typo in original RLRDUP fix that used R5 instead of R0.
0000   245 ;
0000   246 ;       V03-021 RLRUSECNT       Robert L. Rappaport     25-Jan-1983
0000   247 ;               Modify logic that permanently allocates Buffered Data
0000   248 ;               Path on VAX-11/750.  From now on we will only permanently
0000   249 ;               allocate a BDP if the controller is a UDA.  Other UQPORT
0000   250 ;               controllers "semi_permanently" allocate a BDP for the
0000   251 ;               duration of a burst.  This is implemented via adding
0000   252 ;               a new cell, PDT$B_BDPUSECNT, which counts the number of
0000   253 ;               transfers in a burst that are currently using the
0000   254 ;               "semi-permanent" BDP.  When this count goes to zero, the
0000   255 ;               BDP is deallocated.  For the UDA, the usecount is biased
0000   256 ;               by one, so that it never goes to zero and therefore
0000   257 ;               never get deallocated.
0000   258 ;
0000   259 ;       V03-020 RLRPPFORK       Robert L. Rappaport     7-Jan-1983
0000   260 ;               Eliminate Bugcheck in POST_POWER_FORK, that was
0000   261 ;               activated when it found the UCB fork block busy.
0000   262 ;               We do this by defining a new UCB$W_DEVSTS flag,
0000   263 ;               UCB$M_PU_MRESET, that if set causes PUDRIVER to
0000   264 ;               reset itself upon awakening from the busy UCB.
0000   265 ;
0000   266 ;       V03-019 RLRDUP          Robert L. Rappaport     5-Jan-1983
0000   267 ;               Add support for two QIO functions that (1) effectively
0000   268 ;               shut off Class Drivers from the port and only let
0000   269 ;               DUP connections thru, and then (2) reopen the port.
0000   270 ;               The Shutoff QIO function is IO$_STOP and the reopen
0000   271 ;               function is IO$_INITIALIZE.
0000   272 ;
0000   273 ;       V03-018 RLRCREDITa      Robert L. Rappaport     20-Dec-1982
0000   274 ;               Additional correction to take care of bug introduced
0000   275 ;               by original fix.  Must add space to NULL CDT to
0000   276 ;               accomodate CDT$L_CRWAITQFL and CDT$L_CRWAITQBL, and
0000   277 ;               also code to initialize this header.
0000   278 ;
0000   279 ;       V03-017 RLRCREDIT       Robert L. Rappaport     3-Dec-1982
0000   280 ;               Corrected bug brought out in TU81 testing where the
0000   281 ;               credit that was received on the End Packet of a
0000   282 ;               GET COMMAND STATUS (i.e. the "immediate" credit)
0000   283 ;               was assigned to a waiter on the credit resource
0000   284 ;               wait queue.  Then the time out mechanism had no
0000   285 ;               credits to allocate.  The fix is to add in the
```

```
0000   286 ;               received credits upon receipt of the message,
0000   287 ;               then call the Input Dispatcher and only then
0000   288 ;               after return try to resume waiters for credits.
0000   289 ;
0000   290 ;       V03-016 RLRTRBUG        Robert L. Rappaport     8-Oct-1982
0000   291 ;               Correct in trace when we have more than 1 UDA.
0000   292 ;
0000   293 ;       V03-015 RLRPATHB        Robert L. Rappaport     6-Oct-1982
0000   294 ;               Put proper port name into path block. Also fill in
0000   295 ;               SB$Q_SWINCARN with value in EXE$GQ_SYSTIME.
0000   296 ;
0000   297 ;       V03-014 RLRUDASA        Robert L. Rappaport     12-Aug-1982
0000   298 ;               Read UDASA after IOFORK in HARDWARE_INIT so that TU81
0000   299 ;               has enough time to update UDASA after interrupting.
0000   300 ;
0000   301 ;       V03-013 KDM0002         Kathleen D. Morse       28-Jun-1982
0000   302 ;               Added $DYNDEF, $PRDEF, $SSDEF, and $VADEF.
0000   303 ;
0000   304 ;       V03-012 RLREC003        Robert L. Rappaport     4-June-1982
0000   305 ;               Add separate REQDATAP_750 and REQDATAP_730 to replace
0000   306 ;               REQDATAP_750730.  New routine for 750 will only use
0000   307 ;               pre-allocated buffered datapath if the transfer is
0000   308 ;               longword aligned.  Otherwise it will use direct datapath.
0000   309 ;
0000   310 ;       V03-011 RLREC002        Robert L. Rappaport     7-May-1982
0000   311 ;               INSQUE Path Block onto System Block.
0000   312 ;
0000   313 ;       V03-010 RLREC001        Robert L. Rappaport     7-May-1982
0000   314 ;               Add purging of UNIBUS Buffered Data Path in UNMAP.
0000   315 ;               This tracks patch made to V3.0 before release.
0000   316 ;
0000   317 ;
0000   318 ;--
```

```
0000    320          .SBTTL  DEFINITIONS
0000    321 ;
0000    322 ; System definitions (LIB.MLB):
0000    323 ;
0000    324
0000    325          $ADPDEF                 ;Adapter Control Block offsets
0000    326          $CDRPDEF                ;Class Driver Request Packet offsets
0000    327          $CDTDEF                 ;Connection Descriptor offsets
0000    328          $CRBDEF                 ;Channel Request Block offsets
0000    329          $DCDEF                  ;Device type codes
0000    330          $DDBDEF                 ;Device Data Block offsets
0000    331          $DPTDEF                 ;Driver Prologue Table offsets
0000    332          $DYNDEF                 ;Dynamic data structure types
0000    333          $EMBDEF                 ;Error log definitions
0000    334          $EMBLTDEF               ;Log message type codes
0000    335          $IDBDEF                 ;IDB offsets
0000    336          $IODEF                  ;IO function definitions
0000    337          $IPLDEF                 ;IPL symbolic definitions
0000    338          $IRPDEF                 ;IRP offsets
0000    339          $MSGDEF                 ;Opcom message code definitions
0000    340          $PBDEF                  ;Path Block offsets
0000    341          $PDTDEF                 ;Port Descriptor Table offsets
0000    342          $PRDEF                  ;Processor register numbers
0000    343          $PTEDEF                 ;Page Table Entry definitions
0000    344          $RDDEF                  ;Response Descriptor offsets
0000    345          $SBDEF                  ;System Block offsets
0000    346          $SCSCMGDEF              ;SCS Connection Management format
0000    347          $SDIRDEF                ;SCS Directory Entry offsets
0000    348          $SSDEF                  ;System status codes
0000    349          $UBADEF                 ;Unibus Adapter registers
0000    350          $UBMDDEF                ;Unibus Mapping Descriptor format
0000    351          $UCBDEF                 ;Unit Control Block offsets
0000    352          $VADEF                  ;Virtual address fields
0000    353          $VECDEF                 ;CRB transfer vector blk offsets
0000    354
```

```
0000    356          .SBTTL  +       UDA Command Packet Layout
0000    357
0000    358 ;+
0000    359 ;       Each UDA command packet has the following structure where the
0000    360 ;               individual fields are described below the diagram.
0000    361 ;
0000    362 ;       31                                      0
0000    363 ;
0000    364 ;       ,-------------------------------.
0000    365 ;       |       UDAB$L_FLINK            |  Forward link longword
0000    366 ;       |-------------------------------|
0000    367 ;       |       UDAB$L_BLINK            |  Backward link longword
0000    368 ;       |-------------------------------|
0000    369 ;       |\\\\\\\| Buffer | Ring | Ring |
0000    370 ;       |\\\\\\\| Number | Number | Index |
0000    371 ;       |-------|--------|--------|-------|
0000    372 ;       |       UDAB$L_DESCRIP          |  UNIBUS virtual address of this buffer
0000    373 ;       |-------------------------------|
0000    374 ;       | CONID |Msg|Cre|    Message    |
0000    375 ;       |       |typ|dit|     length    |  Controller message envelope
0000    376 ;       |-------|---|---|---------------|
0000    377 ;       |                               |  UDAB$T_TEXT
0000    378 ;       |  Contents of an MSCP Packet   |
0000    379 ;       |                               |
0000    380 ;       |                               |
0000    381 ;       |                               |
0000    382 ;       |                               |
0000    383 ;       |                               |
0000    384 ;       |                               |
0000    385 ;       |                               |
0000    386 ;       |-------------------------------|
0000    387 ;
0000    388 ;
0000    389 ; $UDABDEF -- Define UDA packet buffer structure.  Structure includes
0000    390 ;       port driver header, controller header and text body.  The port
0000    391 ;       driver header contains:
0000    392 ;
0000    393 ;               1. A FLINK and a BLINK for queueing the buffer on the
0000    394 ;                       free queue and also on the SEND Q.
0000    395 ;
0000    396 ;               2. $B_RINGINX which contains the index into a ring on
0000    397 ;                       which this buffer has been placed (valid only
0000    398 ;                       if the buffer is NOT on the free queue).
0000    399 ;
0000    400 ;               3. $B_RINGNO which contains the number (0 => command ring
0000    401 ;                       and 1 => response ring) of the ring on which the
0000    402 ;                       buffer is currently residing.
0000    403 ;
0000    404 ;               4. $B_BUFFNO which contains the number of this buffer.
0000    405 ;                       This value serves as an index into the
0000    406 ;                       PDT$L_BDTABLE, whose elements point to these
0000    407 ;                       buffers.
0000    408 ;
0000    409 ;               5. $L_DESCRIP which contains the UNIBUS virtual address
0000    410 ;                       of the text portion of this buffer in the low order
0000    411 ;                       30 bytes of this longword, and which also has the
0000    412 ;                       two high order bits (ownership and full bits) set.
```

```
                     0000   413 ;                       This is the precise value that must be placed in
                     0000   414 ;                       a ring longword so as to present the buffer to the
                     0000   415 ;                       controller.
                     0000   416 ;
                     0000   417 ;                   6. The controller header which contains a word of length
                     0000   418 ;                       (of the following text portion only), a byte
                     0000   419 ;                       containing two four bit fields encoding the credit
                     0000   420 ;                       field and the message type field, and a byte of
                     0000   421 ;                       Connection ID.
                     0000   422 ;
                     0000   423 ;                   7. The message text portion.
                     0000   424 ;-
                     0000   425
                     0000   426
                     0000   427          $DEFINI UDAB
                     0000   428
                     0000   429 $DEF     UDAB$L_FLINK            .BLKL   1       ; Queue FLINK.
                     0004   430
                     0004   431 $DEF     UDAB$L_BLINK            .BLKL   1       ; Queue BLINK.
                     0008   432
                     0008   433 $DEF     UDAB$B_RINGINX          .BLKB   1       ; Index into ring.
                     0009   434
                     0009   435 $DEF     UDAB$B_RINGNO           .BLKB   1       ; Ring number.
                     000A   436
                     000A   437 $DEF     UDAB$B_BUFFNO           .BLKB   1       ; Index of this buffer.
0000000C             000B   438                                 .BLKB   1       ; Reserved.
                     000C   439
                     000C   440 $DEF     UDAB$L_DESCRIP          .BLKL   1       ; UNIBUS virtual address of buff.
                     0010   441
                     0010   442 $DEF     UDAB$L_CTRLHDR                          ; Alternate name for following
                     0010   443                                                 ;  longword of fields.
                     0010   444 $DEF     UDAB$W_MSG_LEN          .BLKW   1       ; Length of text portion.
                     0012   445
                     0012   446 $DEF     UDAB$B_CREDTYPE         .BLKB   1       ; Encoded CREDIT and MESSAGE TYPE.
                     0013   447
                     0013   448          $VIELD  UDAB,0,<-
                     0013   449                          <CREDITS,4>,-          ; Credit bit field.
                     0013   450                          <MSGTYPE,4>,-          ; Message type bit field.
                     0013   451                          >
                     0013   452
                     0013   453 $DEF     UDAB$B_CONID            .BLKB   1       ; Connection ID.
                     0014   454
                     0014   455 $DEF     UDAB$T_TEXT             .BLKB   60      ; Space for minimal maximum.
                     0050   456
00000050             0050   457 UDAB$C_LENGTH = .
                     0050   458
                     0050   459          $DEFEND UDAB
```

```
                     0000   461
                     0000   462 ; Define Device I/O Page Registers
                     0000   463
                     0000   464         $DEFINI UDA
                     0000   465 $DEF     UDAIP   .BLKW   1       ; Initialization and Polling Register
                     0002   466 $DEF     UDASA   .BLKW   1       ; Status, Address, & VAX Purge ACK Register
                     0004   467         $DEFEND UDA
                     0000   468
                     0000   469
                     0000   470 ; ++
                     0000   471 ; Local symbol definitions
                     0000   472 ;--
                     0000   473
00000FAB             0000   474 LOOP_LIMIT      = ^X<FAB>       ; Step 1 maximum wait time for response
000000B8             0000   475 INTR_VEC        = ^O<270>       ; Primary Interupt vector
                     0000   476
00000000             0000   477 UDA$K_SEQMSGTYP = 0                     ; Sequential Message Type
00000001             0000   478 UDA$K_DGTYPE = 1                        ; Datagram type
00000002             0000   479 UDA$K_CREDTYPE = 2                      ; Credit type
0000000F             0000   480 UDA$K_MAINTTYPE = 15                    ; Maintenance type
00000004             0000   481 UDA$K_RINGEXP = 4                       ; Log base 2 of desired ring size
00000010             0000   482 UDA$K_RINGSIZE = 1@UDA$K_RINGEXP        ; Number of Ring & Packet entries
                     0000   483
                     0000   484 ; Command and Message Ring Control Flags
                     0000   485
0000001E             0000   486 UDA_V_FLAG = 30                         ; Buffer control bit number
40000000             0000   487 UDA_M_FLAG = 1 @ UDA_V_FLAG             ; Buffer control flag mask
0000001F             0000   488 UDA_V_OWN  = 31                         ; Own flag bit number
80000000             0000   489 UDA_M_OWN  = 1 @ UDA_V_OWN             ; Own flag mask
                     0000   490
80000000             0000   491 UQPORT_M_MAPPED=1@31                   ; Transfer Mapped by port.
                     0000   492
0000000A             0000   493 STEP1_LIMIT=10                         ; Number of seconds for STEP 1 timeout
0000000A             0000   494 STEP2_LIMIT=10                         ; Number of seconds for STEP 2 timeout
0000000A             0000   495 STEP3_LIMIT=10                         ; Number of seconds for STEP 3 timeout
                     0000   496
00000080             0000   497 NUMUBAVEC=128                          ; Number of slots in UBA vector
                     0000   498
00000005             0000   499 NO_CONSEC_INITS=5                      ; Number of consecutive times to retry
                     0000   500                                       ;  hardware init without waiting for awhile.
0000000A             0000   501 INIT_DELTA=10                          ; Number of seconds to wait when we wait
                     0000   502                                       ;  for a while.
                     0000   503
00000001             0000   504 ALLOC_DELTA=1                          ; Number of seconds to wait before
                     0000   505                                       ;  retrying allocation request.
                     0000   506
0000001C             0000   507 REGSAVE=2+2+4+4+16                     ; Size of data saved in PU_REGDUMP.
                     0000   508                                       ;  Includes space for ATTNCODE(2),
                     0000   509                                       ;   NUMBINITS(2), MAPREG(4),
                     0000   510                                       ;   UDASA(4-MOVZWL), HOST-PORT(16).
00000001             0000   511 INIT_ATTNCODE=1                        ; Attention code for INIT record.
00000002             0000   512 FAIL_ATTNCODE=2                        ; Attention code for failing INIT record.
00000003             0000   513 UDASA_ATTNCODE=3                       ; Attention code for record after error.
00000004             0000   514 PURGE_ATTNCODE=4                       ; Attention code for Purge error record.
00000005             0000   515 UCODE_ATTNCODE=5                       ; Attention code for uCODE out of rev.
                     0000   516
00000000             0000   517 UDA50_CNTRLTYP=0                       ; UDA50 controller type code.
```

```
                        ; UDA Command Packet Layout

00000001  0000    518 LESI_CNTRLTYP=1                         ; LESI (RC25) controller type code.
00000005  0000    519 TU81_CNTRLTYP=5                         ; TU81 controller type code.
00000006  0000    520 UDA50A_CNTRLTYP=6                       ; UDA50A controller type code.
00000007  0000    521 RDRX_CNTRLTYP=7                         ; RD/RX
00000003  0000    522 MAYA_CNTRLTYP=3                         ; MAYA
0000000D  0000    523 QDA50_CNTRLTYP=13                       ; QDA50
          0000    524
00000000  0000    525 DISK_CONID=0                            ; Disk Server Connection ID
00000001  0000    526 TAPE_CONID=1                            ; Tape Server Connection ID
00000002  0000    527 DUP_CONID=2                             ; DUP  Server Connection ID
          0000    528
0000000F  0000    529 SA_POLL_INTVAL=15                       ; SA register polling interval.
00000001  0000    530 NO_PHYCONTIGPGS=1                       ; # of physically contiguous pages
          0000    531                                         ;  allocate for non-aligned transfers
          0000    532                                         ;  on QBUS.
00000200  0000    533 NO_PHYCONTIGBYT=NO_PHYCONTIGPGS*512     ; # of physically contiguous bytes.
```

```
                     0000   535          .SBTTL  +       Define PU Port specific PDT extension
                     0000   536
                     0000   537 ;+
                     0000   538 ; PUPDT -- Define PU port specific extension to the PDT
                     0000   539 ;         (Must appear after the definition of UDA$K_RINGEXP and
                     0000   540 ;          UDA$K_RINGSIZE)
                     0000   541 ;-
                     0000   542
                     0000   543          $DEFINI PUPDT
                     0000   544
000000E4             0000   545 .=.+PDT$C_LENGTH                        ; Position to end of port-
                     00E4   546                                         ;  independent portion of PDT
                     00E4   547 $DEF     PDT$L_PU_CDTARY                ; Array of pointers to CDT's.
                     00E4   548 $DEF     PDT$L_PU_VC0     .BLKL   1     ;  Connection block address for
                     00E8   549                                         ;  virtual circuit 0 Disk MSCP
                     00E8   550
                     00E8   551 $DEF     PDT$L_PU_VC1     .BLKL   1     ; Connection block address for
                     00EC   552                                         ;  virtual circuit 1 Tape MSCP
                     00EC   553
                     00EC   554 $DEF     PDT$L_PU_VC2     .BLKL   1     ; Connection block address for
                     00F0   555                                         ;  virtual circuit 2 DUP
                     00F0   556
                     00F0   557 $DEF     PDT$L_PU_VC255   .BLKL   1     ; Connection block address for
                     00F4   558                                         ;  virtual circuit 255 (-1)
                     00F4   559                                         ;  Maintenance protocol
                     00F4   560
                     00F4   561 $DEF     PDT$W_PU_CRDARY                ; Array of initial credits.
                     00F4   562 $DEF     PDT$W_PU_CRED0   .BLKW   1     ; Credits to assign on next
                     00F6   563                                         ;  CONNECT to VC0.
                     00F6   564 $DEF     PDT$W_PU_CRED1   .BLKW   1     ; Credits to assign on next
                     00F8   565                                         ;  CONNECT to VC1.
                     00F8   566 $DEF     PDT$W_PU_CRED2   .BLKW   1     ; Credits to assign on next
                     00FA   567                                         ;  CONNECT to VC2.
                     00FA   568 $DEF     PDT$W_PU_CRD255  .BLKW   1     ; Credits to assign on next
                     00FC   569                                         ;  CONNECT to VC255.
                     00FC   570
                     00FC   571 $DEF     PDT$L_PU_SB      .BLKL   1     ; Address of our System Block.
                     0100   572
                     0100   573 $DEF     PDT$L_PU_CSR     .BLKL   1     ; Pointer to controller CSR.
                     0104   574
                     0104   575 $DEF     PDT$L_PU_FQPTR   .BLKL   1     ; Pointer to Free Q. Used to
                     0108   576                                         ;  test for emptiness of Free Q.
                     0108   577 $DEF     PDT$L_PU_FQFL    .BLKL   1     ; Free Q forward pointer.
                     010C   578 $DEF     PDT$L_PU_FQBL    .BLKL   1     ; Free Q backward pointer.
                     0110   579
                     0110   580 $DEF     PDT$L_PU_SNDQFL  .BLKL   1     ; Send Q forward  pointer.
                     0114   581 $DEF     PDT$L_PU_SNDQBL  .BLKL   1     ; Send Q backward pointer.
                     0118   582
                     0118   583 $DEF     PDT$L_PU_BUFQFL  .BLKL   1     ; Buffer wait Q forward pointer.
                     011C   584 $DEF     PDT$L_PU_BUFQBL  .BLKL   1     ; Buffer wait Q backward pointer.
                     0120   585
                     0120   586 $DEF     PDT$B_CRINGINX   .BLKB   1     ; Index of next available slot
                     0121   587                                         ;  in command ring.
                     0121   588 $DEF     PDT$B_CPOLLINX   .BLKB   1     ; Index of next slot to be polled
                     0122   589                                         ;  by host, to see if released.
                     0122   590 $DEF     PDT$B_CRINGCNT   .BLKB   1     ; Count of number of command
                     0123   591                                         ;  ring slots in use. Also
```

```
                     0123   592                                             ;  absolute difference between
                     0123   593                                             ;  previous two fields.
                     0123   594
                     0123   595 $DEF     PDT$B_RRINGINX  .BLKB   1        ; Index of next available slot
                     0124   596                                             ;  in response ring.
                     0124   597 $DEF     PDT$B_RPOLLINX  .BLKB   1        ; Index of next slot to poll by
                     0125   598                                             ;  host, to see if filled in.
                     0125   599 $DEF     PDT$B_RRINGCNT  .BLKB   1        ; Count of not yet filled in
                     0126   600                                             ;  buffers on response ring.
                     0126   601                                             ;  Absolute difference between
                     0126   602                                             ;  previous two fields.
                     0126   603
                     0126   604 $DEF     PDT$B_NOCURCON  .BLKB   1        ; # of current connections.
                     0127   605 $DEF     PDT$B_CONBITMAP .BLKB   1        ; Current Connection bit map. Corresponding
                     0128   606                                             ;  bit set for each open connection.
                     0128   607 $DEF     PDT$B_SERVERS   .BLKB   1        ; Servers Supported at this Port. Bit
                     0129   608                                             ;  set for each supported Server.
                     0129   609
                     0129   610 $DEF     PDT$B_DATAPATH  .BLKB   1        ; Permanently (or semi-permanently)
                     012A   611                                             ;  allocated datapath. For VAX11-750
                     012A   612                                             ;  only. Else zero.
                     012A   613 $DEF     PDT$B_BDPUSECNT .BLKB   1        ; Count of # commands using above BDP.
            0000012C 012B   614                             .BLKB   1        ; Reserved
                     012C   615
                     012C   616 $DEF     PDT$B_CRCONTENT .BLKB   UDA$K_RINGSIZE  ; Array of bytes (one per each
                     013C   617                                                 ;  command ring slot) which
                     013C   618                                                 ;  maintain index of buffer
                     013C   619                                                 ;  currently filling this slot.
                     013C   620
                     013C   621 $DEF     PDT$B_RRCONTENT .BLKB   UDA$K_RINGSIZE  ; Array of bytes (one per each
                     014C   622                                                 ;  response ring slot) which
                     014C   623                                                 ;  maintain index of buffer
                     014C   624                                                 ;  currently filling this slot.
                     014C   625
                     014C   626 $DEF     PDT$L_BDTABLE   .BLKL   2*UDA$K_RINGSIZE; Array of longwords (one per
                     01CC   627                                                 ;  buffer) which point to the
                     01CC   628                                                 ;  respective buffers.
                     01CC   629
                     01CC   630 ; For uVAX I the communications area must not straddle a 64KB boundary.
                     01CC   631 ; So the following .ALIGN starts the communications area on a new page
                     01CC   632 ; and since it is less than 512 bytes long, it cannot straddle pages.
                     01CC   633 ; Note, for uVAX I the PDT is allocated in contiguous memory on a page
                     01CC   634 ; boundary and therefore the .ALIGN really is relevant to a page.
                     01CC   635
            00000200 01CC   636 .=<.+511>&<^c511>       ; Equivalent of .ALIGN  PAGE
                     0200   637
                     0200   638 $DEF     PDT$L_COMAREA                       ; UDA communication area base
            00000202 0200   639                             .BLKW   1           ; Reserved word
            00000203 0202   640                             .BLKB   1           ; Reserved byte
                     0203   641 $DEF     PDT$B_PURGEDP   .BLKB   1           ; Purge data path number
                     0204   642
                     0204   643 $DEF     PDT$W_CMDINT    .BLKW   1           ; Command ring transition flag
                     0206   644
                     0206   645 $DEF     PDT$W_RSPINT    .BLKW   1           ; Response ring transition flag
                     0208   646
                     0208   647 $DEF     PDT$L_RINGBASE
                     0208   648 $DEF     PDT$L_RSPRING   .BLKL   UDA$K_RINGSIZE  ; Response ring
```

```
                     ;+ Define PU Port specific PDT extension

                0248   649
                0248   650 $DEF    PDT$L_CMDRING   .BLKL   UDA$K_RINGSIZE  ; Command ring
00000088        0288   651 PDT$C_COMAREA[N = .-PDT$L_COMAREA                ; Length of Comarea
                0288   652
                0288   653 $DEF    PDT$L_PU_BUFARY                         ; Buffer array.
                0288   654
                0288   655         .REPT   2*UDA$K_RINGSIZE                ; Allocate packets for response
                0288   656                                                 ; and command rings
                0288   657
                0288   658                         .BLKB   UDAB$C_LENGTH   ; Length of a buffer.
                0288   659
                0288   660         .ENDR
00000A88        0C88   661 PDT$C_RINGLEN = .-PDT$L_COMAREA                  ; Length of area to map
                0C88   662
                0C88   663 $DEF    PDT$L_TRTABLE   .BLKL   1       ; Pointer to base of usable (beyond
                0C8C   664                                         ;  header) portion of trace table
                0C8C   665 $DEF    PDT$L_TRTBLPTR  .BLKL   1       ; Pointer to next area in trace table
                0C90   666                                         ;  to use.
                0C90   667 $DEF    PDT$L_TRTBLEND  .BLKL   1       ; Pointer beyond end of trace table.
                0C94   668
                0C94   669
00000C94        0C94   670 PDT$C_PULENGTH = .                           ; Total size of a
                0C94   671                                              ;  PDT for the UDA port
                0C94   672
                0C94   673 ; The following PDT extension is only needed when running on uVAX I.
                0C94   674
                0C94   675 $DEF    PDT$L_PQ_MAP    .BLKL   1       ; Vitrual pointer to 496 pseudo map
                0C98   676                                         ;  "registers".
                0C98   677
                0C98   678 $DEF    PDT$L_PQ_PGQFL  .BLKL   1       ; Queue Header for CDRP's waiting to
                0C9C   679 $DEF    PDT$L_PQ_PGQBL  .BLKL   1       ;  allocate the aligned Page that follows.
                0CA0   680 $DEF    PDT$L_PQ_POWNER .BLKL   1       ; CDRP that currently owns the Page.
                0CA4   681 $DEF    PDT$L_PQ_PGPHAD .BLKL   1       ; Physical address of the Page.
                0CA8   682 $DEF    PDT$L_PQ_UBFSVA .BLKL   1       ; Virtual address of the page mapped by
                0CAC   683                                         ;  UCB$L_SVPN.  This page is used to map
                0CAC   684                                         ;  one page of the user's buffer in order
                0CAC   685                                         ;  to be able to copy it to the aligned
                0CAC   686                                         ;  Page.
                0CAC   687 $DEF    PDT$L_PQ_SVPPTE .BLKL   1       ; Pointer to the PTE of the UCB$L_SVPN.
                0CB0   688 $DEF    PDT$L_PQ_USRPTE .BLKL   1       ; Pointer into user page table that
                0CB4   689                                         ;  points to current user SVAPTE.
                0CB4   690
                0CB4   691 $DEF    PDT$L_PQ_ALGNPG .BLKB   NO_PHYCONTIGBYT ; A word aligned string of
                0EB4   692                                                 ;  physically contiguous pages.
                0EB4   693
00000CB4        0EB4   694 PDT$C_CONTIGLEN = .-PDT$L_COMAREA            ; Length of area needed to be Physically
                0EB4   695                                              ;  contiguous.
00000EB4        0EB4   696 PDT$C_UV1LENGTH = .                          ; Length of PDT needed for uVAX I.
                0EB4   697         $DEFEND PUPDT
```

```
                     0000   699          .SBTTL  +       Define PU specific UCB extension
                     0000   700
                     0000   701 ;+
                     0000   702 ; $PUUCBDEF -- Defined UDA extension to UCB.
                     0000   703 ;       (Must appear after the definition of UDA$K_RINGEXP and
                     0000   704 ;        UDA$K_RINGSIZE)
                     0000   705 ;-
                     0000   706
                     0000   707          $DEFINI UCB
                     0000   708
000000A0             0000   709 .=UCB$L_DPC+4                                    ; Position to end of
                     00A0   710                                                  ;  standard UCB for
                     00A0   711                                                  ;  error logging devices
                     00A0   712
                     00A0   713 $DEF     UCB$L_PU_ALLOC  .BLKL   1       ; Space to save size of alloc. request.
                     00A4   714
                     00A4   715 $DEF     UCB$B_UDAFLAGS  .BLKB   1       ; Internal control flags
                     00A5   716          $VIELD  UDA,0,<-                ; Internal flag definitions
                     00A5   717                  <ONLINE,,V>,-           ; UDA is On Line
                     00A5   718                  <STOPPED,,M>,-          ; Port only open for DUP CONNECTIONS.
                     00A5   719           >
                     00A5   720
                     00A5   721 $DEF     UCB$B_INITCNT   .BLKB   1       ; Count of # of times left to retry
                     00A6   722                                          ;  hardware init consecutively.
                     00A6   723 $DEF     UCB$W_NUMBINITS .BLKW   1       ; Number of times UDA50 has been Inited.
                     00A8   724
                     00A8   725 $DEF     UCB$W_ATTNCODE  .BLKW   1       ; Attention code for PU_REGDUMP.
                     00AA   726
                     00AA   727 $DEF     UCB$W_UDASA     .BLKW   1       ; Contents of SA saved at last interrupt
                     00AC   728
                     00AC   729 $DEF     UCB$W_PORTSTEP1 .BLKW   1       ; Contents of SA at start of STEP1.
                     00AE   730          _VIELD  PS1,0,<-                ; Fields in STEP1
                     00AE   731                  <,6>,-                  ;  Reserved
                     00AE   732                  <MP,,M>,-               ;  If set port supports address mapping
                     00AE   733                  <OD,,M>,-               ;  If set port supports odd addresses
                     00AE   734                  <DI,,M>,-               ;  If set port implements enhanced diagnosti
                     00AE   735                  <QB,,M>,-               ;  If set port supports 22-bit bus addr
                     00AE   736                  <NV,,M>,-               ;  If set no host settable vector addr
                     00AE   737                  <S1,,M>,-               ;  Must be set in step 1
                     00AE   738                  <S2,,M>,-               ;  Must be clear in step 1
                     00AE   739                  <S3,,M>,-               ;  Must be clear in step 1
                     00AE   740                  <S4,,M>,-               ;  Must be clear in step 1
                     00AE   741                  <ER,,M>,-               ;  If set then we had an error
                     00AE   742           >
                     00AE   743
                     00AE   744 $DEF     UCB$W_HOSTSTEP1 .BLKW   1       ; What host writes to SA in STEP1.
                     00B0   745          _VIELD  HS1,0,<-                ; Fields in Host STEP1
                     00B0   746                  <INTVEC,7,M>,-          ;  Interrupt vector address/4
                     00B0   747                  <IE,,M>,-               ;  Interrupt enable during STEPS 1-3
                     00B0   748                  <RRNGLEN,3,M>,-         ;  Response ring length (exponent)
                     00B0   749                  <CRNGLEN,3,M>,-         ;  Command ring length (exponent)
                     00B0   750                  <WR,,M>,-               ;  If set port enters diagnostic wrap mode
                     00B0   751                  <BIT15,,M>,-            ;  High bit always set
                     00B0   752           >
                     00B0   753
                     00B0   754 $DEF     UCB$W_PORTSTEP2 .BLKW   1       ; What port responds at start of STEP2.
                     00B2   755          _VIELD  PS2,0,<-                ; Fields in Port STEP2
```

```
00B2    756                 <RRNGLEN,3,M>,-         ;  Echoed response ring length (exponent)
00B2    757                 <CRNGLEN,3,M>,-         ;  Echoed command ring length (exponent)
00B2    758                 <WR,,M>,-               ;  Echoed diagnostic wrap bit
00B2    759                 <,1>,-                  ;  Echoed BIT15 always set
00B2    760                 <PORTTYPE,3,M>,-        ;  Type of this port
00B2    761                 <S1,,M>,-               ;  Must be clear in STEP2
00B2    762                 <S2,,M>,-               ;  Must be set in STEP2
00B2    763                 <S3,,M>,-               ;  Must be clear in STEP2
00B2    764                 <S4,,M>,-               ;  Must be clear in STEP2
00B2    765                 <ER,,M>,-               ;  If set then error in STEP1
00B2    766         >
00B2    767
00B2    768 $DEF    UCB$W_HOSTSTEP2 .BLKW   1       ; What host writes to SA during STEP2.
00B4    769         _VIELD  HS2,0,<-                ; Fields in Host STEP2
00B4    770                 <PI,,M>,-               ;  Host requests adapter purge interrupts
00B4    771                 <RINGBASEL,15,M>,-      ;  Lo order of address of communication
00B4    772         >                               ;   area (in UNIBUS virtual space)
00B4    773
00B4    774 $DEF    UCB$W_PORTSTEP3 .BLKW   1       ; What port responds at start of STEP3.
00B6    775         _VIELD  PS3,0,<-                ; Fields in Port STEP3
00B6    776                 <INTVEC,7,M>,-          ;  Echoed interrupt address/4
00B6    777                 <IE,,M>,-               ;  Echoed interrupt enable
00B6    778                 <,3>,-                  ;  Reserved
00B6    779                 <S1,,M>,-               ;  Must be clear in STEP3
00B6    780                 <S2,,M>,-               ;  Must be clear in STEP3
00B6    781                 <S3,,M>,-               ;  Must be set in STEP3
00B6    782                 <S4,,M>,-               ;  Must be clear in STEP3
00B6    783                 <ER,,M>,-               ;  If set we had error in STEP2
00B6    784         >
00B6    785
00B6    786 $DEF    UCB$W_HOSTSTEP3 .BLKW   1       ; What host writes to SA during STEP3.
00B8    787         _VIELD  HS3,0,<-                ; Fields in Host STEP3
00B8    788                 <RINGBASEH,15,M>,-      ;  Hi order of address of comm. area
00B8    789                 <PP,,M>,-               ;  If set host requests execution of
00B8    790         >                               ;   purge and poll tests
00B8    791
00B8    792 $DEF    UCB$W_PORTSTEP4 .BLKW   1       ; What port responds at start of STEP4.
00BA    793         _VIELD  PS4,0,<-                ; Fields in Port STEP4
00BA    794                 <UCODEVER,4,M>,-        ;  Microcode version number
00BA    795                 <CNTRLTYP,7,M>,-        ;  Controller type
00BA    796                 <S1,,M>,-               ;  Must be clear in STEP4
00BA    797                 <S2,,M>,-               ;  Must be clear in STEP4
00BA    798                 <S3,,M>,-               ;  Must be clear in STEP4
00BA    799                 <S4,,M>,-               ;  Must be set in STEP4
00BA    800                 <ER,,M>,-               ;  If set we had an error in STEP3
00BA    801         >
00BA    802
00BA    803 $DEF    UCB$W_HOSTSTEP4 .BLKW   1       ; What host writes to SA during STEP4.
00BC    804         _VIELD  HS4,0,<-                ; Fields in Host STEP4
00BC    805                 <GO,,M>,-               ;  If set controller begins immediately
00BC    806                 <LF,,M>,-               ;  If set host wants Last Fail response
00BC    807                 <BURST,6,M>,-           ;  Maximum lonwords(-1) / NPR transfer
00BC    808                 <,8>,-                  ;  Reserved
00BC    809         >
00BC    810
00BC    811
00BC    812 $DEF    UCB$L_PU_SVAPTE .BLKL   1       ; Place to save UCB$L_SVAPTE.
```

```
                     ;+ Define PU specific UCB extension

00C0   813 $DEF     UCB$W_PU_BOFF   .BLKW   1       ; Place to save UCB$W_BOFF.
00C2   814 $DEF     UCB$W_PU_BCNT   .BLKW   1       ; Place to save UCB$W_BCNT.
00C4   815
00C4   816          $VIELD  UCB,0,<-                        ; Define bits for UCB$W_DEVSTS
00C4   817          <,1>,-                                  ;  Unused
00C4   818          <PU_FRKBSY,,M>,-                        ;  Fork block interlock bit
00C4   819          <PU_HRDINI,,M>,-                        ;  Hardware init in progress
00C4   820          <PU_PWINIT,,M>,-                        ;  POWER init in progress
00C4   821          <PU_MRESET,,M>,-                        ;  Goto POST_POWER_FORK after fork.
00C4   822          <PU_BDPATH,,M>,-                        ;  BDP already allocated (or not).
00C4   823          <PU_INILOG,,M>,-                        ;  Init Log in progress
00C4   824          >                                       ;
00C4   825
000000C4 00C4   826 UCB$C_PUSIZE = .
00C4   827
00C4   828          $DEFEND UCB
```

```
                        .SBTTL  DRIVER STRUCTURES
                        .SBTTL  +       Driver Prologue Table
                        ;
                        ; DPTAB - Driver Prologue Table
                        ;
                        DPTAB   END=PU$END,-            ;End of driver label
                                ADAPTER=UBA,-           ;Adapter type
                                FLAGS=DPT$M_SCS!DPT$M_SVP!DPT$M_NOUNLOAD,-
                                                        ;Driver requires SCS to be loaded
                                                        ; uVAX I needs System Virtual Page
                                UCBSIZE=UCB$C_PUSIZE,-  ;UCB size
                                NAME=PUDRIVER           ;Driver name
                        ;
                        ; Initialize fork IPL, device characteristics, and device interrupt IPL
                        ;
                        DPT_STORE INIT
                        ;
                        DPT_STORE       UCB,UCB$B_FIPL,B,8      ;Fork IPL
                        ;
                        DPT_STORE       UCB,UCB$L_DEVCHAR,L,<-  ;Device characteristics:
                                        DEV$M_SHR!-             ; Sharable
                                        DEV$M_AVL!-             ; Available
                                        DEV$M_ELG!-             ; Error logging device
                                        DEV$M_IDV!-             ; Input device
                                        DEV$M_ODV>              ; Output device
                        ;
                        DPT_STORE       UCB,UCB$B_DIPL,B,21     ;Device interrupt IPL
                        ;
                        ; Reinitialize DDT address, interrupt routine address, controller init address,
                        ; and unit init address
                        ;
                        DPT_STORE REINIT
                        ;
                        DPT_STORE       DDB,DDB$L_DDT,D,PU$DDT  ;DDT address
                        DPT_STORE       CRB,CRB$L_INTD+4,-      ;Interrupt routine addr
                                        D,PU$INT                ;
                        DPT_STORE       CRB,CRB$L_INTD+VEC$L_INITIAL,-
                                        D,PU$CTLINIT            ;Controller init addr
                        DPT_STORE       CRB,CRB$L_INTD+VEC$L_UNITINIT,-
                                        D,PU$UNITINIT           ;Unit init addr
                        DPT_STORE       END                     ;
```

```
                     0000   865          .SBTTL  +       Driver Dispatch Table
                     0000   866
                     0000   867          DDTAB   DEVNAM=PU,-                     ;
                     0000   868                  START=PU$STARTIO,-              ;QIO's are illegal temporarily
                     0000   869                  FUNCTB=PU$FUNCTABLE,-           ;Function decision table
                     0000   870                  UNITINIT=PU$UNITINIT,-          ;Unit init routine addr
                     0000   871                  ERLGBF=REGSAVE+4+EMB$L_DV_REGSAV,-
                     0000   872                  REGDMP=PU_REGDUMP
                     0038   873
                     0038   874 ;
                     0038   875 ; We only support two functions; IO$_STOP and IO$_INITIALIZE.  The first
                     0038   876 ;       effectively closes the Port to all connections except those to
                     0038   877 ;       the DUP SERVER.  The second function reopens the Port and initializes
                     0038   878 ;       it.
                     0038   879 ;
                     0038   880 ; Inputs:
                     0038   881 ;       R3 => IRP
                     0038   882 ;       R5 => UCB of Port
                     0038   883 ;
                     0038   884
                     0038   885 PU$STARTIO:
                     0038   886
          00   EF   0038   887          EXTZV   #IRP$V_FCODE,-          ; Extract function code.
          06        003A   888                  #IRP$S_FCODE,-
    50 20 A3        003B   889                  IRP$W_FUNC(R3),R0
       50 03   91   003E   890          CMPB    #IO$_STOP,R0            ; See if IO$_STOP request.
          20   13   0041   891          BEQL    START_STOP              ; EQL means yes it was.
       50 04   91   0043   892          CMPB    #IO$_INITIALIZE,R0      ; See if IO$_INITIALIZE requested.
          38   12   0046   893          BNEQ    ILLIOFUNC               ; NEQ means not one of supported functions.
                    0048   894
          55   DD   0048   895          PUSHL   R5                      ; Save R5 => UCB.
          2F   10   004A   896          BSBB    SUCCESS                 ; Complete QIO request but get control
                    004C   897                                          ;  after REQCOM so as to continue.
          55 8ED0   004C   898          POPL    R5                      ; Restore R5 => UCB.
                    004F   899          REQPCHAN                        ; Re-allocate channel after REQCOM.
54  0084 C5   DO   0055   900          MOVL    UCB$L_PDT(R5),R4        ; R4 => PDT.
      0CA2   30   005A   901          BSBW    FPC$MRESET              ; Do initialize.
          02   8A   005D   902          BICB    #UDA$M_STOPPED,-        ; Open up port.
     00A4 C5        005F   903                  UCB$B_ODAFLAGS(R5)
                    0062   904          RSB                             ; Kill this thread.
                    0063   905 START_STOP:
          55   DD   0063   906          PUSHL   R5                      ; Save R5 => UCB.
          14   10   0065   907          BSBB    SUCCESS                 ; Complete QIO request but get control
                    0067   908                                          ;  after REQCOM so as to continue.
          55 8ED0   0067   909          POPL    R5                      ; Restore R5 => UCB.
                    006A   910          REQPCHAN                        ; Re-allocate channel after REQCOM.
54  0084 C5   DO   0070   911          MOVL    UCB$L_PDT(R5),R4        ; R4 => PDT.
          02   88   0075   912          BISB    #UDA$M_STOPPED,-        ; Close port.
     00A4 C5        0077   913                  UCB$B_ODAFLAGS(R5)
                    007A   914          RSB                             ; Kill this thread.
                    007B   915 SUCCESS:
     50   01   DO   007B   916          MOVL    #SS$_NORMAL,R0          ; Return success code.
          05   11   007E   917          BRB     COMPLETE_IO             ; And branch around.
                    0080   918 ILLIOFUNC:
50  00F4 8F   3C   0080   919          MOVZWL  #SS$_ILLIOFUNC,R0               ; We do not support other QIO's
                    0085   920 COMPLETE_IO:
          51   D4   0085   921          CLRL    R1
```

```
;+ Driver Dispatch Table
;
0087    922             REQCOM
```

```
                     008D   924          .SBTTL  +       "Register" Dump routine
                     008D   925
                     008D   926 ;+ PU_REGDUMP
                     008D   927 ;
                     008D   928 ; Inputs:
                     008D   929 ;       R0 => Buffer to fill
                     008D   930 ;       R5 => UCB
                     008D   931 ;
                     008D   932 ; Outputs:
                     008D   933 ;       R1 modified
                     008D   934 ;
                     008D   935
                     008D   936 PU_REGDUMP:
                     008D   937
     80    1C   D0   008D   938          MOVL    #REGSAVE,(R0)+          ; Save number of bytes following.
80   00A8 C5   B0   0090   939          MOVW    UCB$W_ATTNCODE(R5),(R0)+; Save Attention code (record type).
80   00A6 C5   B0   0095   940          MOVW    UCB$W_NUMBINITS(R5),(R0)+; Number times port init'ed.
  51   24 A5   D0   009A   941          MOVL    UCB$L_CRB(R5),R1        ; R1 => CRB.
                     009E   942
                     009E   943          ASSUME  VEC$B_NUMREG    EQ      VEC$W_MAPREG+2
                     009E   944          ASSUME  VEC$B_DATAPATH  EQ      VEC$B_NUMREG+1
  80   34 A1   D0   009E   945          MOVL    CRB$L_INTD+VEC$W_MAPREG(R1),(R0)+       ; Save dedicated map registe
                     00A2   946
80   00AA C5   3C   00A2   947          MOVZWL  UCB$W_UDASA(R5),(R0)+   ; Save contents of SA register.
                     00A7   948
                     00A7   949          ASSUME  UCB$W_HOSTSTEP1 EQ      UCB$W_PORTSTEP1+2
                     00A7   950          ASSUME  UCB$W_PORTSTEP2 EQ      UCB$W_HOSTSTEP1+2
                     00A7   951          ASSUME  UCB$W_HOSTSTEP2 EQ      UCB$W_PORTSTEP2+2
80   00AC C5   7D   00A7   952          MOVQ    UCB$W_PORTSTEP1(R5),(R0)+       ; Save Host-Port dialogue.
                     00AC   953
                     00AC   954          ASSUME  UCB$W_HOSTSTEP3 EQ      UCB$W_PORTSTEP3+2
                     00AC   955          ASSUME  UCB$W_PORTSTEP4 EQ      UCB$W_HOSTSTEP3+2
                     00AC   956          ASSUME  UCB$W_HOSTSTEP4 EQ      UCB$W_PORTSTEP4+2
80   00B4 C5   7D   00AC   957          MOVQ    UCB$W_PORTSTEP3(R5),(R0)+       ; Save Host-Port dialogue.
                     00B1   958
                05   00B1   959          RSB                             ; Return to caller.
```

```
                        .SBTTL  ♦       Function Decision Table

                PU$FUNCTABLE:

                        FUNCTAB ,-                      ;Valid functions:
                                <STOP,-                 ; Stop port.
                                 INITIALIZE>            ; Reopen Port and Initialize.

                        FUNCTAB ,-                      ;Buffered functions:
                                <STOP,-                 ; Stop port.
                                 INITIALIZE>            ; Reopen Port and Initialize.
                        FUNCTAB ♦EXE$ZEROPARM,-         ;ZERO PARAMETER FUNCTIONS
                                <STOP,-                 ; Stop port.
                                 INITIALIZE>            ; Reopen Port and Initialize.
```

```
                        00CE    976          .SBTTL  +       Static Storage
                        00CE    977
                        00CE    978 ; Static driver storage
                        00CE    979
                 00'00' 00CE    980          .ALIGN  LONG,0
00000000                00D0    981 PU$L_DRIVER_STS:        .LONG   0       ; Status flag longword.
00000000                00D4    982 MAP$V_MAPREGS=0                         ; Bit meaning that the pseudo map
                        00D4    983                                         ;  registers have been mapped into
                        00D4    984                                         ;  system space at ^x800 in the Q-bus
                        00D4    985                                         ;  "CSR" area pointed at by ADP.
00000001                00D4    986 MAP$M_MAPREGS=1                         ; Mask of above bit.
00000000                00D4    987 PU$L_MPHYAD:            .LONG   0       ; Physical address of "map registers".
00000000                00D8    988 PU$L_TRACE_VARIABLE:    .LONG   0       ; Variable describing what kinds (if
                        00DC    989                                         ;  any) of trace have been enabled.
                        00DC    990                                         ; 0 implies no port tracing.
                        00DC    991                                         ; 1 implies we try to allocate resources
                        00DC    992                                         ;  to trace for ports that support Tapes
                        00DC    993                                         ; 2 implies that we try to allocate
                        00DC    994                                         ;  resources to trace all ports
```

```
                     00DC   996          .SBTTL  +       NULL MESSAGE AND DATAGRAM INPUT ROUTINES
                     00DC   997          .SBTTL  +       NULL CDT
                     00DC   998
                     00DC   999 ;+
                     00DC  1000 ; NULL CDT and NULL message input and datagram input routines.
                     00DC  1001 ;
                     00DC  1002 ; Inputs:                (for routines)
                     00DC  1003 ;
                     00DC  1004 ;        R1                                  -Length of message
                     00DC  1005 ;        R2                                  -Addr of message
                     00DC  1006 ;        R4                                  -Addr of PDT
                     00DC  1007 ;-
                     00DC  1008
                     00DC  1009          .enabl  lsb
                     00DC  1010
                     00DC  1011 NULL_MSG_INPUT:                              ; Sequential messages for null connections
                     00DC  1012                                              ;  are to be ignored.
           53   DD  00DC  1013          PUSHL   R3                          ; Save R3.
           10   11  00DE  1014          BRB     10$                         ; Branch around to deallocate buffer.
                     00E0  1015
                     00E0  1016 NULL_DG_INPUT:                               ; Datagrams for null connections are to
                     00E0  1017                                              ;  be logged.
           53   DD  00E0  1018          PUSHL   R3                          ; Save R3.
     53  00DC C4   D0  00E2  1019          MOVL    PDT$L_UCB0(R4),R3           ; R3 => UCB. (input to ERL$LOGMESSAGE)
      50    04   D0  00E7  1020          MOVL    #EMB$C_UM,R0                ; Log message type.
 00000000'GF    16  00EA  1021          JSB     G^ERL$LOGMESSAGE            ; Call to log message.
                     00F0  1022 10$:
      52   14   C2  00F0  1023          SUBL    #UDAB$T_TEXT,R2             ; R2 => buffer header.
         0DA7   30  00F3  1024          BSBW    Q_DEALLOC_BUF               ; Call internal entry to deallocate buffer.
           53 8ED0  00F6  1025          POPL    R3                          ; Restore R3.
                     00F9  1026 NULL_ROUTINE:                                ; Label of an RSB instruction.
                     00F9  1027 NULL_ERR_ROUT:
           05        00F9  1028          RSB                                 ; Return to our caller.
                     00FA  1029
        00'00'       00FA  1030          .ALIGN  LONG,0
                     00FC  1031 NULL_CDT:
     0000019C        00FC  1032          .BLKB   CDT$K_LENGTH                ; Allocate space for the NULL CDT.
                     019C  1033          .dsabl  lsb
```

```
                     019C  1035          .SBTTL  Request and Release DATAPATH transfer vectors
                     019C  1036
                     019C  1037 ;+
                     019C  1038
                     019C  1039          .ALIGN  LONG,0
                     019C  1040
                     019C  1041 REQDATAPATH_TV:                          ; Transfer vector to routine to request
                     019C  1042                                          ;  a UNIBUS datapath.
00000000'            019C  1043          .LONG   IOC$REQDATAPUDA         ; Routine to call (for VAX-11/780).
                     01A0  1044                                          ;  For other processors, this pointer
                     01A0  1045                                          ;  will be overlaid with the address of
                     01A0  1046                                          ;  REQDATAP_750 or REQDATAP_730, below.
                     01A0  1047
                     01A0  1048 RELDATAPATH_TV:                          ; Transfer vector to routine to release
                     01A0  1049                                          ;  a UNIBUS datapath.
00000000'            01A0  1050          .LONG   IOC$RELDATAPUDA         ; Routine to call when running on
                     01A4  1051                                          ;  VAX-11/780.  For other processors,
                     01A4  1052                                          ;  this address is overlaid with the
                     01A4  1053                                          ;  RELDATAP_750 or RELDATAP_730, below.
                     01A4  1054
                     01A4  1055 ;
                     01A4  1056 ; REQDATAP_750 assigns the "semi-permanently" allocated datapath, currently
                     01A4  1057 ;       in use by the current burst, bumping the useage count by one,
                     01A4  1058 ;       or if no burst is in progress, allocates a buffered datapath and
                     01A4  1059 ;       then increments the useage count by one.
                     01A4  1060 ;
                     01A4  1061 ; The following paragraph applies only until the Buffered Datapath bug is
                     01A4  1062 ; fixed on VAX-11/750's.
                     01A4  1063 ;       If the current transfer is NOT longword aligned or if it is NOT
                     01A4  1064 ;       an integral number of longwords in length, we do NOT use the Buffered
                     01A4  1065 ;       Datapath but instead use the direct datapath.  We implicitly assign
                     01A4  1066 ;       direct datapath (i.e. datapath zero) by simply
                     01A4  1067 ;       RSB'ing to our caller, depending on the fact that our caller cleared
                     01A4  1068 ;       CDRP$L_UBARSRCE before calling us.  Note that in this case we do NOT
                     01A4  1069 ;       increment any useage count.
                     01A4  1070 ;
                     01A4  1071 ; REQDATAP_730 implicitly assigns the direct datapath to the current transfer
                     01A4  1072 ;       since the 730 has no buffered datapaths (i.e. datapath zero).  This is
                     01A4  1073 ;       accomplished by simply RSB'ing to our caller, depending on the fact
                     01A4  1074 ;       that our caller cleared CDRP$L_UBARSRCE before calling us.
                     01A4  1075 ;
                     01A4  1076 ;
                     01A4  1077 ; Inputs:
                     01A4  1078 ;       R4                                  -Addr of PDT
                     01A4  1079 ;       R5                                  -Addr of CDRP
                     01A4  1080 ;
                     01A4  1081 ; Outputs:(for REQDATAP_750 and REQDATAP_730)
                     01A4  1082 ;       CDRP$L_UBARSRCE+UBMD$B_DATAPATH set from PDT$B_DATAPATH.
                     01A4  1083 ;
                     01A4  1084
                     01A4  1085 REQDATAP_750:
                     01A4  1086
      D2 A5    89    01A4  1087          BISB3   CDRP$L_BCNT(R5),-              ; R0 has "OR" of low
50    D0 A5          01A7  1088                  CDRP$W_BOFF(R5),R0              ;  bits of BOFF and BCNT,
   50    03    93    01AA  1089          BITB    #3,R0                           ; Test for longword alignment
                     01AD  1090                                                  ;  and integral # longwords.
         34    12    01AD  1091          BNEQ    REQDATAP_730                    ; NEQ means not aligned or
```

```
                     01AF  1092                                    ;  possibly odd # of words
                     01AF  1093                                    ;  being transferred.
                     01AF  1094                                    ; Else fall thru to use pre-
                     01AF  1095                                    ;  allocated datapath.
                     01AF  1096 REQDATAP_8SS:                      ; UQport on a BUA.
    012A C4    95    01AF  1097          TSTB    PDT$B_BDPUSECNT(R4)       ; Do we have semi-perm BDP?
         0C    12    01B3  1098          BNEQ    10$                       ; NEQ implies yes. Goto use it.
00000000'GF    16    01B5  1099          JSB     G^IOC$REQDATAPUDA         ; Call to allocate a BDP.
      3F A5    90    01BB  1100          MOVB    CDRP$L_UBARSRCE+UBMD$B_DATAPATH(R5),-  ; Record new semi-perm
    0129 C4          01BE  1101                  PDT$B_DATAPATH(R4)        ;  BDP in PDT.
                     01C1  1102 10$:
    0129 C4    90    01C1  1103          MOVB    PDT$B_DATAPATH(R4),-      ; Assign semi-perm data-
      3F A5          01C5  1104                  CDRP$L_UBARSRCE+UBMD$B_DATAPATH(R5)  ;  path to transfer.
    012A C4    96    01C7  1105          INCB    PDT$B_BDPUSECNT(R4)       ; Inc useage count.
               05    01CB  1106          RSB                               ; Return to caller.
                     01CC  1107
                     01CC  1108 RELDATAP_750:
                     01CC  1109
      D2 A5    89    01CC  1110          BISB3   CDRP$L_BCNT(R5),-         ; R0 has "OR" of low
   50 D0 A5          01CF  1111                  CDRP$W_BOFF(R5),R0        ;  bits of BOFF and BCNT,
   50    03    93    01D2  1112          BITB    #3,R0                     ; Test for longword alignment
                     01D5  1113                                            ;  and integral # longwords.
         0C    12    01D5  1114          BNEQ    REQDATAP_730              ; NEQ means not aligned or
                     01D7  1115                                            ;  possibly odd # of words
                     01D7  1116                                            ;  being transferred.
                     01D7  1117                                            ; Else fall thru to use pre-
                     01D7  1118                                            ;  allocated datapath.
                     01D7  1119 RELDATAP_8SS:                      ; UQport on a BUA.
    012A C4    97    01D7  1120          DECB    PDT$B_BDPUSECNT(R4)       ; DECR useage count.
         06    12    01DB  1121          BNEQ    REQDATAP_730              ; NEQ means not end of burst.
00000000'GF    16    01DD  1122          JSB     G^IOC$RELDATAPUDA         ; Release BDP after burst.
                     01E3  1123
                     01E3  1124 REQDATAP_730:                      ; Return to caller implicitly assigning
                     01E3  1125                                    ;  direct datapath (i.e. datapath zero)
                     01E3  1126 RELDATAP_730:                      ; Return to caller implicitly de-assigning
                     01E3  1127                                    ;  direct datapath (i.e. datapath zero)
                     01E3  1128
               05    01E3  1129          RSB                               ; Return to caller.
```

```
01E4  1131          .SBTTL  INITIALIZATION
01E4  1132 ;+
01E4  1133 ; The following table gives word offsets for fork process SCS calls.
01E4  1134 ; Offsets are relative to the address of the controller initialization
01E4  1135 ; routine, PU$CTLINIT.
01E4  1136 ;-
01E4  1137
01E4  1138 ;
01E4  1139 ; Macro to generate the table and ASSUME statements about PDT format:
01E4  1140 ;
01E4  1141
01E4  1142          .MACRO  SCS_OFFSET_TAB  ENTRY_LIST
01E4  1143
01E4  1144           $$$ENTRYNUM=0                              ; No entries in table yet
01E4  1145           .IRP    ENTRY,ENTRY_LIST                   ; For each entry in the list...
01E4  1146           .WORD   <FPC$'ENTRY'-PU$CTLINIT>           ;  insert offset from ctl init,
01E4  1147           .IF     NE $$$ENTRYNUM                     ;  and for entries after the 1st
01E4  1148           ASSUME $$$PREV+4 EQ PDT$L_'ENTRY'          ;  specify assumed PDT adjacency
01E4  1149           .ENDC                                      ;
01E4  1150           $$$PREV=PDT$L_'ENTRY'                      ; Set previous entry as this entry
01E4  1151           $$$ENTRYNUM=$$$ENTRYNUM+1                  ; Step entry count
01E4  1152           .ENDR                                      ;
01E4  1153
01E4  1154           ASSUME $$$PREV+4 EQ PDT$C_SCSEND           ; Final PDT assumption
01E4  1155           .WORD  0                                   ; Offset table terminator
01E4  1156
01E4  1157          .ENDM   SCS_OFFSET_TAB                      ;
01E4  1158
01E4  1159 ;
01E4  1160 ; Table itself:
01E4  1161 ;
01E4  1162
01E4  1163 PU$SCSOFFSET::
01E4  1164
01E4  1165          SCS_OFFSET_TAB  <-                          ; Invoke macro to define offsets
01E4  1166                  ACCEPT,-
01E4  1167                  ALLOCDG,-
01E4  1168                  ALLOCMSG,-
01E4  1169                  CONNECT,-
01E4  1170                  DEALLOCDG,-
01E4  1171                  DEALLOMSG,-
01E4  1172                  DEALRGMSG,-
01E4  1173                  DCONNECT,-
01E4  1174                  MAP,-
01E4  1175                  MAPBYPASS,-
01E4  1176                  MAPIRP,-
01E4  1177                  MAPIRPBYP,-
01E4  1178                  QUEUEDG,-
01E4  1179                  QUEUEMDGS,-
01E4  1180                  RCHMSGBUF,-
01E4  1181                  RCLMSGBUF,-
01E4  1182                  REJECT,-
01E4  1183                  REQDATA,-
01E4  1184                  SENDDATA,-
01E4  1185                  SENDDG,-
01E4  1186                  SENDMSG,-
01E4  1187                  SNDCNTMSG,-
```

```
                01E4  1188                UNMAP,-
                01E4  1189                READCOUNT,-
                01E4  1190                RLSCOUNT,-
                01E4  1191                MRESET,-
                01E4  1192                MSTART,-
                01E4  1193                MAINTFCN,-
                01E4  1194                SENDRGDG,-
                01E4  1195                STOP_VCS,-
                01E4  1196                >
                0222  1197
                0222  1198          .SBTTL  +       UNIT_INIT
                0222  1199
                0222  1200 ;+
                0222  1201 ; TBS
                0222  1202 ;
                0222  1203 ; Inputs:
                0222  1204 ;
                0222  1205 ;       R3                      -Address of PA configuration reg
                0222  1206 ;       R4                      -Same as R3
                0222  1207 ;       R5                      -Addr of UCB
                0222  1208 ;
                0222  1209 ; Outputs:
                0222  1210 ;
                0222  1211 ;       R0                              -Status code
                0222  1212 ;                                       (SS$_NORMAL,SS$_INSFMEM, or 0 if no SVPN)
                0222  1213 ;       R1-R4                           -Destroyed
                0222  1214 ;       Other registers         -Preserved
                0222  1215 ;-
                0222  1216
                0222  1217          .ENABL  LSB
                0222  1218
                0222  1219 PU$UNITINIT::
                0222  1220
                0222  1221 ;        MOVB    #DT$_UDA50,UCB$B_DEVTYPE  ; Set the device type to UDA50
          54 A5 B5 0222  1222        TSTW    UCB$W_UNIT(R5)            ; Is this unit 0?
             06 13 0225  1223        BEQL    INIT_CTLR                 ; If YES, go to init controller first.
                0227  1224
             10 A8 0227  1225        BISW    #UCB$M_ONLINE,-           ; Else merely set unit on line
          64 A5    0229  1226                UCB$W_STS(R5)             ;  and
                05 022B  1227        RSB                               ; Return
                022C  1228          .DSABL  LSB
```

```
                     1230          .SBTTL  ✦       CONTROLLER INIT
                     1231
                     1232 ;✦
                     1233 ; The controller initialization entry as seen by the system, PU$CTLINIT,
                     1234 ; is a noop since initialization can't begin without the unit 0 UCB.
                     1235 ; Actual controller init is called from unit 0 unit initialization with
                     1236 ; the same inputs as unit init.
                     1237 ;
                     1238 ;
                     1239 ; Inputs:
                     1240 ;
                     1241 ;       R3                      -Addr of PU CSR
                     1242 ;       R4                      -Same as R3
                     1243 ;       R5                      -Addr of UCB for unit 0
                     1244 ;
                     1245 ; Outputs:
                     1246 ;
                     1247 ;       R0-R3                   -Destroyed
                     1248 ;       Other registers         -Preserved
                     1249 ;-
                     1250
                     1251 PU$CTLINIT::                             ; Controller init called by system
                     1252
  05                 1253          RSB                             ; Return
```

```
                     022D  1255 ;
                     022D  1256 ; Controller initialization called from unit 0 init.
                     022D  1257 ;
                     022D  1258 ; Inputs:
                     022D  1259 ;       R5                              -UCB address
                     022D  1260 ;
                     022D  1261 ; Outputs:
                     022D  1262 ;       R0                              -Status code
                     022D  1263 ;                                       (SS$_NORMAL,SS$_INSFMEM, or 0 if no SVPN)
                     022D  1264
           00000031  022D  1265 BRW_OPCODE=^x31                          ; Opcode of BRW instruction, used in
                     022D  1266                                          ;  concocting PATCH below.
                     022D  1267
                     022D  1268         .enabl  lsb
                     022D  1269
                     022D  1270 INIT_CTLR::
                     022D  1271
                06   11  022D  1272         BRB     0$                  ; Branch around breakpoint.
      00000000'GF    16  022F  1273         JSB     G^INI$BRK           ; Breakpoint for debugging.
                     0235  1274 0$:
           0084 C5   D5  0235  1275         TSTL    UCB$L_PDT(R5)       ; See if first time thru here.
                7A   12  0239  1276         BNEQ    POWER_INIT          ; NEQ implies this is POWER FAILURE.
           FE99 CF   D5  023B  1277         TSTL    PU$L_TRACE_VARIABLE ; See if any tracing requested.
                12   13  023F  1278         BEQL    5$                  ; EQL implies NO tracing.
                     0241  1279
                     0241  1280 ; Here if tracing has been enabled.  There are two places in the inline code
                     0241  1281 ; where we would like to trace; once in Send Sequence message, and once when
                     0241  1282 ; receiving a message.  In order to accomplish this, we here dynamically
                     0241  1283 ; patch a BRW instruction into the two appropriate locations.
                     0241  1284
                     0241  1285         MOVL    #BRW_OPCODE!-               ; If tracing enabled, patch code
                     0242  1286                 <ENABLE_COMMAND_OFFSET@8>,- ;  into Send Message Logic to trace
      0FB0'CF    00001E31'8F  0242  1287                 W^ENABLE_COMMAND_START      ;  outgoing message.
                     024A  1288         MOVL    #BRW_OPCODE!-               ; Here patch code into Receive Logic
                     024B  1289                 <ENABLE_RESPONSE_OFFSET@8>,-; to trace received message.
      12BD'CF    00009031'8F  024B  1290                 W^ENABLE_RESPONSE_START
                     0253  1291 5$:
                   02   A8  0253  1292         BISW    #UCB$M_PU_FRKBSY,-  ; Indicate the UCB fork block is in
                 68 A5       0255  1293                 UCB$W_DEVSTS(R5)    ;  use. Setting this bit defers processing
                             0257  1294                                     ;  powerfailure recovery until after
                             0257  1295                                     ;  initialization completes.
                             0257  1296         IOFORK                      ; Lower IPL.
                             025D  1297
                             025D  1298         REQPCHAN                    ; Permanently allocate channel.
                             0263  1299
                 80 8F   90  0263  1300         MOVB    #DC$_BUS,-          ; Initialize constant UCB fields.
                 40 A5       0266  1301                 UCB$B_DEVCLASS(R5)
                    03   90  0268  1302         MOVB    #DT$_UQPORT,-       ; Type of UQPORT will be determined
                 41 A5       026A  1303                 UCB$B_DEVTYPE(R5)   ;  later.
                             026C  1304
                             026C  1305 ; Initialize CRB wakeup mechanism.
                             026C  1306
              53   24 A5   D0  026C  1307         MOVL    UCB$L_CRB(R5),R3    ; R3 => CRB.
              10 A3   55   D0  0270  1308         MOVL    R5,CRB$L_AUXSTRUC(R3) ; CRB => UCB as auxillary structure.
              18 A3   01   CE  0274  1309         MNEGL   #1,CRB$L_DUETIME(R3) ; Set infinite time.
                 FE7D CF   9E  0278  1310         MOVAB   NULL_ROUTINE,-
                    1C A3      027C  1311                 CRB$L_TOUTROUT(R3)  ; Inocuous routine for now.
```

```
                        ;  CONTROLLER INIT

00000000'GF   16  027E  1312          JSB     G^IOC$THREADCRB         ; Put CRB on wakeup list.
                  0284  1313
         007C   30  0284  1314          BSBW    BUILD_PDT               ; Allocate and fill in PDT.
       2A 50    E9  0287  1315          BLBC    R0,15$                  ; Return error status to caller.
         0258   30  028A  1316          BSBW    INIT_UDA_BUFFERS        ; Map area in PDT containing buffers.
       24 50    E9  028D  1317          BLBC    R0,15$                  ; Return error status to caller.
         046E   30  0290  1318          BSBW    INIT_INIT_STEPS         ; Initialize values in UCB to use
                  0293  1319                                          ;  in UDA hardware initialization.
       1E 50    E9  0293  1320          BLBC    R0,15$                  ; Return error status to caller.
         0926   30  0296  1321          BSBW    BUILD_PB_SB             ; Create and fill in system block and
                  0299  1322                                          ;  path block.
       18 50    E9  0299  1323          BLBC    R0,15$                  ; Return error status to caller.
                  029C  1324 10$:
            05    90  029C  1325          MOVB    #NO_CONSEC_INITS,-      ; Initialize UCB field that counts
         00A5 C5      029E  1326                  UCB$B_INITCNT(R5)       ;  hardware init retries.
                  02A1  1327
         056D   30  02A1  1328          BSBW    HARDWARE_INIT           ; Init UDA hardware.
         0D 50    E9  02A4  1329          BLBC    R0,15$                  ; LBC implies couldn't Init hardware.
         09C1   30  02A7  1330          BSBW    UPDATE_PB_SB            ; Update fields in the Path and System block
            10    A8  02AA  1331          BISW    #UCB$M_ONLINE,-         ; After hardware init, we are online.
       64 A5        02AC  1332                  UCB$W_STS(R5)
         0FAC   30  02AE  1333          BSBW    POLL_RSPRING            ; This call has the effect of clearing
                  02B1  1334                                          ;  the UCB$M_PU_FRKBSY bit in
                  02B1  1335                                          ;  UCB$W_DEVSTS and also of finding any
                  02B1  1336                                          ;  responses that may have backed up
                  02B1  1337                                          ;  due to the bit's having been set.
    50    01    D0  02B1  1338          MOVL    #SS$_NORMAL,R0          ; Set return code.
                  02B4  1339 15$:
                  05  02B4  1340          RSB
                  02B5  1341
                  02B5  1342 POWER_INIT:
                  02B5  1343
            01    E3  02B5  1344          BBCS    #UCB$V_PU_FRKBSY,-      ; Appropriate UCB fork block if free,
         01 68 A5     02B7  1345                  UCB$W_DEVSTS(R5),20$    ;  else continue and RSB.
                  05  02BA  1346          RSB                             ; If fork block busy, relie on
                  02BB  1347                                          ;  Interrupt Service Routine to GOTO
                  02BB  1348                                          ;  POST_POWER_FORK.
                  02BB  1349 20$:
                  02BB  1350          IOFORK                          ; Lower IPL so as to continue with
                  02C1  1351                                          ;  signaling SYSAPS that the VC fell.
            18    11  02C1  1352          BRB     30$                     ; Branch around Fork block appropriation.
                  02C3  1353 POST_POWER_FORK:
                  02C3  1354
                  02C3  1355 ; Here we turn off CRB wakeup mechanism.
                  02C3  1356
    50    24 A5   D0  02C3  1357          MOVL    UCB$L_CRB(R5),R0        ; R0 => CRB.
 18 A0    01      CE  02C7  1358          MNEGL   #1,CRB$L_DUETIME(R0)    ; Set infinite time.
       FE2A CF    9E  02CB  1359          MOVAB   NULL_ROUTINE,-
            1C A0        02CF  1360                  CRB$L_TOUTROUT(R0)      ; Inocuous routine for now.
                  02D1  1361
            01    E3  02D1  1362          BBCS    #UCB$V_PU_FRKBSY,-      ; Appropriate the UCB fork block if NOT
         05 68 A5     02D3  1363                  UCB$W_DEVSTS(R5),30$    ;  busy and branch around.
            10    A8  02D6  1364          BISW    #UCB$M_PU_MRESET,-      ; Indicate that whoever has fork block
               68 A5     02D8  1365                  UCB$W_DEVSTS(R5)        ;  busy, should reset upon wakeup.
                  05  02DA  1366          RSB                             ; Return to caller to kill thread.
                  02DB  1367 30$:
            10    AA  02DB  1368          BICW    #UCB$M_PU_MRESET,-      ; Clear bit possibly set above.
```

```
                        ;  CONTROLLER INIT

                68 A5       02DD  1369                  UCB$W_DEVSTS(R5)
                      AA    02DF  1370          BICW    #UCB$M_ONLINE!-         ; If here we are not online,
                            02E0  1371                   UCB$M_POWER,-          ;  and the power failure is being handled
             64 A5   30     02E0  1372                  UCB$W_STS(R5)
                            02E3  1373
                            02E3  1374
                            02E3  1375 ; Here we call the CONNECTION error routine for each connection.
                            02E3  1376
                53   03   D0  02E3  1377          MOVL    #3,R3                   ; R3 = highest possible connection index.
                            02E6  1378 40$:
           54   0084 C5   D0  02E6  1379          MOVL    UCB$L_PDT(R5),R4                ; R4 => PDT.
                      53   DD  02EB  1380          PUSHL   R3                      ; Save connection index.
                      55   DD  02ED  1381          PUSHL   R5                      ; Save R5 => UCB.
        53   00E4 C443   D0  02EF  1382          MOVL    PDT$L_PU_CDTARY(R4)[R3],R3      ; R3 => CDT.
                   0C B3   16  02F5  1383          JSB     @CDT$L_ERRADDR(R3)              ; Call connection error routine.
                            02F8  1384                                              ; NOTE: that for closed connections,
                            02F8  1385                                              ;  the error routine is the NULL
                            02F8  1386                                              ;  one that is just an RSB.
                      55 8ED0   02F8  1387          POPL    R5                      ; Restore R5 => UCB.
                      53 8ED0   02FB  1388          POPL    R3                      ; Restore R3 = connection index.
                   E5 53   F4   02FE  1389          SOBGEQ  R3,40$                  ; Loop thru all possible indices.
                      99   11   0301  1390          BRB     10$                     ; Branch back to re-init port.
                            0303  1391          .dsabl  lsb
```

```
                     0303  1393          .SBTTL  +       Build PDT
                     0303  1394
                     0303  1395 ;+
                     0303  1396 ; BUILD_PDT - allocate and fill in non variable data in the UDA port PDT.
                     0303  1397 ; This routine has processor dependent code to separate code streams for
                     0303  1398 ; uVAX I ports, and for all other systems.  The uVAX I needs special
                     0303  1399 ; treatment for various reasons.
                     0303  1400 ;
                     0303  1401 ;       1. First, for uVAX I, the communications area, the buffers, and the
                     0303  1402 ;          Pseudo Map Registers (these are not needed at all for other systems)
                     0303  1403 ;          must all be in physically contiguous memory pages.  To accomodate
                     0303  1404 ;          this, for uVAX I we allocate the entire PDT in physically
                     0303  1405 ;          contiguous memory.
                     0303  1406 ;
                     0303  1407 ;       2. Second, the base of the Pseudo Map Registers must be on a page
                     0303  1408 ;          boundary.  To accomodate this need, the first four pages of the
                     0303  1409 ;          contiguous region we allocate for the PDT is used for this purpose.
                     0303  1410 ;          A pointer to this region is left in PDT$L_PQ_MAP.
                     0303  1411 ;
                     0303  1412 ;       3. For uVAX I, the communications area must not straddle a 64KB
                     0303  1413 ;          boundary.  Currently, the communicatons area is much less
                     0303  1414 ;          than a page in length and if it can be shown that the communications
                     0303  1415 ;          area does not straddle a page boundary, this would be sufficient
                     0303  1416 ;          to show that it does not straddle a 64KB boundary.  Since the PDT
                     0303  1417 ;          is located on a page boundary, we determine the byte offset of the
                     0303  1418 ;          base of the communications area (symbol COMAREA_BOFF).
                     0303  1419
00000000             0303  1420             COMAREA_BOFF = PDT$L_COMAREA & 511
                     0303  1421
                     0303  1422 ;           This allows us to establish an ASSUME statement that proves
                     0303  1423 ;           that the communications area does not straddle a page boundary.
                     0303  1424
                     0303  1425             ASSUME      COMAREA_BOFF+PDT$C_COMAREALN    LE      512
                     0303  1426
                     0303  1427 ;           Future changes to the PDT might break this ASSUME, and a possible
                     0303  1428 ;           re-arrangement of the layout of the PDT would be called for at that
                     0303  1429 ;           time.
                     0303  1430 ;
                     0303  1431 ;       4. Finally, the routines that implement SCS functions 'MAP_IRP'' and
                     0303  1432 ;          ''UNMAP'' are different on uVAX I systems, so therefore the  PDT
                     0303  1433 ;          dispatch vectors for these functions are treated separately for
                     0303  1434 ;          uVAX I.
                     0303  1435 ;
                     0303  1436 ;
                     0303  1437 ; Inputs:
                     0303  1438 ;       R5                                  -Addr of UCB
                     0303  1439 ;
                     0303  1440 ; Outputs:
                     0303  1441 ;       R0                                  -Status code (SS$_NORMAL,SS$_INSFMEM)
                     0303  1442 ;
                     0303  1443
                     0303  1444 ;
                     0303  1445 ; PDT adjacency assumptions:
                     0303  1446 ;
                     0303  1447
                     0303  1448 ASSUME  PDT$W_SIZE      EQ 8
                     0303  1449 ASSUME  PDT$W_SIZE+2    EQ PDT$B_TYPE
```

```
                     0303  1450 ASSUME  PDT$B_TYPE+1    EQ PDT$B_SUBTYP
                     0303  1451 ASSUME  PDT$B_SUBTYP+1  EQ PDT$C_SCSBASE
                     0303  1452
                     0303  1453                 .enabl  lsb
                     0303  1454
                     0303  1455 BUILD_PDT:
                     0303  1456
                     0303  1457         CPUDISP <<780,ALLOC_PDT_NOTUV1>,-        ; Dispatch to allocate a CPU depende
                     0303  1458                  <750,ALLOC_PDT_NOTUV1>,-        ;  sized PDT.
                     0303  1459                  <730,ALLOC_PDT_NOTUV1>,-
                     0303  1460                  <790,ALLOC_PDT_NOTUV1>,-
                     0303  1461                  <UV1,ALLOC_PDT_UV1>,-
                     0303  1462                  <8SS,ALLOC_PDT_8SS>,-
                     0303  1463                 >
                     031D  1464
                     031D  1465
                     031D  1466 ALLOC_PDT_8SS:                               ; See if we are on a UNIBUS adapter
                     031D  1467                                              ;  or else on a BDA.
     50   24 A5   D0  031D  1468         MOVL    UCB$L_CRB(R5),R0               ; R0 => CRB.
     50   38 A0   D0  0321  1469         MOVL    CRB$L_INTD+VEC$L_ADP(R0),R0    ; R0 => ADP.
  0E A0    01   B1  0325  1470         CMPW    #AT$_UBA,ADP$W_ADPTYPE(R0)     ; See if on UNIBUS adapter.
           04   13  0329  1471         BEQL    ALLOC_PDT_NOTUV1               ; If yes, branch.
                     032B  1472 ;
                     032B  1473 ; Here we must be on a BDA. Appropriate code may one day be placed here.
                     032B  1474 ; Till then,
                     032B  1475         BUG_CHECK       UDAPORT,FATAL
                     032F  1476
                     032F  1477 ALLOC_PDT_NOTUV1:
                     032F  1478
 51   0C94 8F   3C  032F  1479         MOVZWL  #PDT$C_PULENGTH,R1             ; R1 contains PDT size needed for other
                     0334  1480                                              ;  CPU's.
       0955   30  0334  1481         BSBW    ALLOC_POOL                     ; Allocate R1 bytes from pool
      23 50   E8  0337  1482         BLBS    R0,20$                         ; Branch if success to common code.
       0138   31  033A  1483         BRW     100$                           ; Else goto return error
                     033D  1484
                     033D  1485 ALLOC_PDT_UV1:                               ; uVAX I needs physically contiguous
                     033D  1486                                              ;  memory long enough for the map
                     033D  1487                                              ;  registers (4 pages) and the PDT.
                     033D  1488
          00001000   033D  1489 UV1_PDT_LENGTH = <PDT$C_UV1LENGTH + 511> & <^c511>
          00000008   033D  1490 UV1_PDT_PAGES = UV1_PDT_LENGTH / 512      ; Pages needed for uVAX I PDT.
                     033D  1491
     51   08   D0  033D  1492         MOVL    #UV1_PDT_PAGES,R1              ; R1 contains pages needed for uVAX I
                     0340  1493                                              ;  PDT.
                     0340  1494
 00000000'GF   16  0340  1495         JSB     G^EXE$ALOPHYCNTG               ; Allocate physically contiguous memory.
      03 50   E8  0346  1496         BLBS    R0,10$                         ; LBS means successful allocation.
       0129   31  0349  1497         BRW     100$                           ; Else goto error return.
                     034C  1498 10$:
         3F   BB  034C  1499         PUSHR   #^M<R0,R1,R2,R3,R4,R5>         ; Save MOVC registers
00   6E   00   2C  034E  1500         MOVC5   #0,(SP),#0,-                   ; Zero initialize structure
     1000 8F       0352  1501                 #<UV1_PDT_PAGES>*512,-
           62      0355  1502                 (R2)
         3F   BA  0356  1503         POPR    #^M<R0,R1,R2,R3,R4,R5>         ; Restore MOVC registers
                     0358  1504
51   1000 8F   B0  0358  1505         MOVW    #UV1_PDT_LENGTH,R1             ; Set size of uVAX I PDT into R1.
                     035D  1506 20$:
```

```
      0084 C5    52   D0  035D           MOVL    R2,UCB$L_PDT(R5)            ; Save PDT addr
      00DC C2    55   D0  0362           MOVL    R5,PDT$L_UCB0(R2)           ; PDT => UCB.
       53   24 A5     D0  0367           MOVL    UCB$L_CRB(R5),R3            ; Get CRB addr
       10 A3     52   D0  036B           MOVL    R2,CRB$L_AUXSTRUC(R3)       ;  and save PDT addr in CRB
                 82   7C  036F           CLRQ    (R2)+                       ; Init PDT, unused longwds,
                 01   B0  0371           MOVW    #PDT$M_SNGLHOST,-           ; Indicate port to single host bus,(-8
              FC A2       0373                   PDT$W_PORTCHAR-8(R2)        ;  takes into account (R2)+ above).
                 02   90  0375           MOVB    #PDT$C_PU,-                 ; Indicate type of PDT created, (-8
              FF A2       0377                   PDT$B_PDT_TYPE-8(R2)        ;  takes into account (R2)+ above).
                          0379
                          0379           ASSUME  PDT$W_SIZE      EQ      8
           82    51   B0  0379           MOVW    R1,(R2)+                    ;  PDT size.
                          037C
                          037C           ASSUME  PDT$B_TYPE      EQ      10
                          037C           ASSUME  PDT$B_SUBTYP    EQ      11
     82    0560 8F    B0  037C           MOVW    #<DYN$C_SCS_PDT@8 + DYN$C_SCS>,(R2)+  ;  structure subtype and type
                          0381
      53    30 A3     D0  0381           MOVL    CRB$L_INTD+VEC$L_INITIAL(R3),R3  ; Get addr of controller
                          0385                                               ;  init routine
     51    FE5B CF    3E  0385           MOVAW   PU$SCSOFFSET,R1             ; Get addr of table of offsets
                          038A                                               ;  to SCS entries in PUDRIVER
                          038A
           50    81   32  038A 30$:      CVTWL   (R1)+,R0                    ; Get offset to next SCS routine
                 06   13  038D           BEQL    40$                         ; Branch if no more
     82    53    50   C1  038F           ADDL3   R0,R3,(R2)+                 ; Add offset from controller init
                          0393                                               ;  to addr of controller init
                          0393                                               ;  and store in PDT
                 F5   11  0393           BRB     30$                         ; Get next offset
                          0395
                          0395 40$:
     51    0084 C5    D0  0395           MOVL    UCB$L_PDT(R5),R1                    ; R1 => PDT.
      50    24 A5     D0  039A           MOVL    UCB$L_CRB(R5),R0                    ; R0 => CRB.
            38 A0     D0  039E           MOVL    CRB$L_INTD+VEC$L_ADP(R0),-          ; Save ADP address in PDT.
          00E0 C1         03A1                   PDT$L_ADP(R1)
                          03A4           ASSUME  IDB$L_CSR       EQ      0
            2C B0     D0  03A4           MOVL    @CRB$L_INTD+VEC$L_IDB(R0),-         ; Save CSR addr in PDT.
          0100 C1         03A7                   PDT$L_PU_CSR(R1)
 00BC C1   FE00 8F    3C  03AA           MOVZWL  #<127*512>, -               ; Define maximum byte count supported
                          03B1                   PDT$L_MAXBCNT(R1)           ; for block transfers as 127 blocks.
                          03B1
                          03B1
                          03B1           CPUDISP <<780,NO_OVERLAYMAP>,-  ; Dispatch around overlaying of MAP
                          03B1                    <750,NO_OVERLAYMAP>,-  ;  and UNMAP pointers uVAX I.
                          03B1                    <730,NO_OVERLAYMAP>,-
                          03B1                    <790,NO_OVERLAYMAP>,-
                          03B1                    <UV1,OVERLAYMAP>,-
                          03B1                    <8SS,OVERLAYMAP_8SS>,-
                          03B1                   >
                          03CB
                          03CB OVERLAYMAP_8SS:                               ; If on a UNIBUS adapter, goto
                          03CB                                               ;  NO_OVERLAYMAP.
     50    00E0 C1    D0  03CB           MOVL    PDT$L_ADP(R1),R0            ; R0 => ADP.
                 01   B1  03D0           CMPW    #AT$_UBA,-                  ; See if on UNIBUS adapter.
              0E A0       03D2                   ADP$W_ADPTYPE(R0)
                 1A   13  03D4           BEQL    NO_OVERLAYMAP               ; EQL implies UNIBUS adapter.
                          03D6
                          03D6 ; For the BDA port the MAPIRP and UNMAPIRP functions are distinct.
```

```
                     03D6  1564 ;  Here we overlay the PDT dispatch vectors for these functions.
                     03D6  1565
112F'CF   9E         03D6  1566          MOVAB   W^FPC$MAPIRP_BDA,-      ; Overlay PDT dispatch vector for MAPIRP
  34 A1              03DA  1567                  PDT$L_MAPIRP[R1)        ;  on BDA.
11BD'CF   9E         03DC  1568          MOVAB   W^FPC$UNMAP_BDA,-       ; Overlay PDT dispatch vector for UNMAP
  64 A1              03E0  1569                  PDT$L_UNMAP[R1)         ;  on BDA.
     0C    11        03E2  1570          BRB     NO_OVERLAYMAP           ; Branch around to join common code.
                     03E4  1571
                     03E4  1572 OVERLAYMAP:
                     03E4  1573
                     03E4  1574 ; For the uVAX I Q-BUS port the MAPIRP and UNMAPIRP functions are distinct.
                     03E4  1575 ;  Here we overlay the PDT dispatch vectors for these functions.
                     03E4  1576
1014'CF   9E         03E4  1577          MOVAB   W^FPC$MAPIRP_UV1,-      ; Overlay PDT dispatch vector for MAPIRP
  34 A1              03E8  1578                  PDT$L_MAPIRP[R1)        ;  on uVAX I.
1162'CF   9E         03EA  1579          MOVAB   W^FPC$UNMAP_UV1,-       ; Overlay PDT dispatch vector for UNMAP
  64 A1              03EE  1580                  PDT$L_UNMAP[R1)         ;  on uVAX I.
                     03F0  1581
                     03F0  1582 NO_OVERLAYMAP:
                     03F0  1583
                     03F0  1584 ; Init NULL CDT with addresses of message and datagram input routines.
                     03F0  1585
FCE8 CF   9E         03F0  1586          MOVAB   NULL_MSG_INPUT,-                ; Address of message input
FD05 CF              03F4  1587                  NULL_CDT+CDT$L_MSGINPUT         ;  routine.
FCE5 CF   9E         03F7  1588          MOVAB   NULL_DG_INPUT,-                 ; Address of datagram input
FD02 CF              03FB  1589                  NULL_CDT+CDT$L_DGINPUT          ;  routine.
FCF7 CF   9E         03FE  1590          MOVAB   NULL_ERR_ROUT,-                 ; Address of error routine.
FD03 CF              0402  1591                  NULL_CDT+CDT$L_ERRADDR
FD2B CF   9E         0405  1592          MOVAB   NULL_CDT+CDT$L_CRWAITQFL,-      ; Init dummy list header.
FD28 CF              0409  1593                  NULL_CDT+CDT$L_CRWAITQFL
FD24 CF   9E         040C  1594          MOVAB   NULL_CDT+CDT$L_CRWAITQFL,-
FD25 CF              0410  1595                  NULL_CDT+CDT$L_CRWAITQBL
                     0413  1596
                     0413  1597 ; Init CDT pointers to all point to NULL CDT.
                     0413  1598
50   FCE5 CF   9E    0413  1599          MOVAB   NULL_CDT,R0                     ; R0 => NULL CDT.
00E4 C1   50   D0    0418  1600          MOVL    R0,PDT$L_PU_VC0(R1)             ; VC0 CDT pointer => NULL CDT.
00E8 C1   50   D0    041D  1601          MOVL    R0,PDT$L_PU_VC1(R1)             ; VC1 CDT pointer => NULL CDT.
00EC C1   50   D0    0422  1602          MOVL    R0,PDT$L_PU_VC2(R1)             ; VC2 CDT pointer => NULL CDT.
00F0 C1   50   D0    0427  1603          MOVL    R0,PDT$L_PU_VC255(R1)           ; VC255 CDT pointer => NULL CDT.
                     042C  1604
01   FCA8 CF   D1    042C  1605          CMPL    PU$L_TRACE_VARIABLE,#1  ; Determine what kind (if any) tracing
                     0431  1606                                          ;  is desired.  0 implies no tracing,
                     0431  1607                                          ;  1 implies only try to allocate
                     0431  1608                                          ;  resources for "PT" tracing, and
                     0431  1609                                          ;  2 implies always try to allocate.
                     0431  1610
          3F   19    0431  1611          BLSS    90$                     ; LSS implies 0, so branch around since
                     0433  1612                                          ;  we want no tracing.
          0B   14    0433  1613          BGTR    80$                     ; GTR implies 2, so branch around to try
                     0435  1614                                          ;  to allocate trace tables.
                     0435  1615
                     0435  1616                                          ; If here we only want "PT" tracing,
                     0435  1617                                          ;  then test for a "PT" port and
                     0435  1618                                          ;  if not there, branch around.
                     0435  1619
  51   28 A5   D0    0435  1620          MOVL    UCB$L_DDB(R5),R1        ; R1 => DDB.  See if "PU" or "PT" device
```

```
                16 A1    54 8F    91  0439  1621              CMPB    #^A/T/,DDB$T_NAME+2(R1) ; Test for "T" in name.
                               32   12  043E  1622              BNEQ    90$                     ; Since we want to trace "PT" only,
                                                  0440  1623                                      ;  if NOT "PT" then branch around.
                                                  0440  1624  80$:
            51    0000BB90 8F    D0  0440  1625              MOVL    #500*96+16,R1           ; Allocate trace table for 500 entries.
                        0842   30  0447  1626              BSBW    ALLOC_POOL              ; Call to allocate.
                           25 50    E9  044A  1627              BLBC    R0,90$                  ; LBC implies failure.
                                                  044D  1628
                50    0084 C5    D0  044D  1629              MOVL    UCB$L_PDT(R5),R0        ; R0 => PDT.
                                                  0452  1630                                      ;  check.
                                  82   7C  0452  1631              CLRQ    (R2)+                   ; Initialize trace table header for SDA.
                              82    51   B0  0454  1632              MOVW    R1,(R2)+                ; Save size.
                82    0060 8F    B0  0457  1633              MOVW    #DYN$C_SCS,(R2)+        ; Type.
                                  82   D4  045C  1634              CLRL    (R2)+                   ; Round header upto 16 byte boundary.
                0C88 C0    52   D0  045E  1635              MOVL    R2,PDT$L_TRTABLE(R0)    ; Save pointer to base of trace table.
                0C8C C0    52   D0  0463  1636              MOVL    R2,PDT$L_TRTBLPTR(R0)   ; Pointer to next area to use.
    0C90 C0     52    0000BB80 8F    C1  0468  1637              ADDL3   #500*96,R2,-            ; Pointer to beyond end of trace
                                                  0472  1638                                      PDT$L_TRTBLEND(R0)      ;  table.
                                                  0472  1639  90$:
                              50    01   D0  0472  1640              MOVL    S^#SS$_NORMAL,R0        ; Return success.
                                                  0475  1641  100$:
                                  05   0475  1642              RSB
                                                  0476  1643              .dsabl  lsb
```

```
                                0476  1645          .SBTTL  +       TRACE_COMMAND and TRACE_RESPONSE
                                0476  1646
                                0476  1647 ; Routines to record command and response buffer contents in the trace table.
                                0476  1648 ;       Trace table entries are 96 bytes long so that they line up nicely in
                                0476  1649 ;       a dump.
                                0476  1650 ;
                                0476  1651
                                0476  1652 TRACE_COMMAND:                          ; INPUTS: R2 => Command buffer,
                                0476  1653                                        ;         R4 => PDT.
                                0476  1654
                     7E  50  7D 0476  1655          MOVQ    R0,-(SP)              ; Save R0 and R1.
                     50  52  D0 0479  1656          MOVL    R2,R0                 ; R0 => buffer to trace.
                         06  11 047C  1657          BRB     TRACE_COMMON          ; Branch around to common code.
                                047E  1658
                                047E  1659 TRACE_RESPONSE:                         ; INPUTS: R3 => Response buffer,
                                047E  1660                                        ;         R4 => PDT.
                                047E  1661
                     7E  50  7D 047E  1662          MOVQ    R0,-(SP)              ; Save R0 and R1.
                     50  53  D0 0481  1663          MOVL    R3,R0                 ; R0 => buffer to trace.
                                0484  1664
                                0484  1665 TRACE_COMMON:
                                0484  1666
                   0C88  C4  D5 0484  1667          TSTL    PDT$L_TRTABLE(R4)     ; Test for existence of trace table.
                         57  13 0488  1668          BEQL    30$                   ; EQL implies unable to allocate table.
                                048A  1669
                                048A  1670          DSBINT                        ; Prevent interrupts during allocation
                                0490  1671                                        ;  of trace table entry.
               51   0C8C  C4  D0 0490  1672          MOVL    PDT$L_TRTBLPTR(R4),R1 ; R1 => area in trace table to use.
               51   0C90  C4  D1 0495  1673          CMPL    PDT$L_TRTBLEND(R4),R1 ; See if we should circle back to start
                                049A  1674                                        ;  of trace table.
                         05  14 049A  1675          BGTR    20$                   ; GTR implies NO.
               51   0C88  C4  D0 049C  1676          MOVL    PDT$L_TRTABLE(R4),R1  ; R1 => base of trace table.
                                04A1  1677 20$:
  0C8C C4   51    00000060  8F  C1 04A1  1678          ADDL3   #96,R1,PDT$L_TRTBLPTR(R4); Point to next entry.
                                04AB  1679          ENBINT                        ; Undo DSBINT.
                     81  80  7D 04AE  1680          MOVQ    (R0)+,(R1)+           ; Twelve long words are 96 bytes.
                     81  80  7D 04B1  1681          MOVQ    (R0)+,(R1)+
                     81  80  7D 04B4  1682          MOVQ    (R0)+,(R1)+
                     81  80  7D 04B7  1683          MOVQ    (R0)+,(R1)+
                     81  80  7D 04BA  1684          MOVQ    (R0)+,(R1)+
                     81  80  7D 04BD  1685          MOVQ    (R0)+,(R1)+
                     81  80  7D 04C0  1686          MOVQ    (R0)+,(R1)+
                     81  80  7D 04C3  1687          MOVQ    (R0)+,(R1)+
                     81  80  7D 04C6  1688          MOVQ    (R0)+,(R1)+
                     81  80  7D 04C9  1689          MOVQ    (R0)+,(R1)+
                     81  6E  7D 04CC  1690          MOVQ    (SP),(R1)+            ; Trace saved R0, R1.
                   81    08  AE D0 04CF  1691          MOVL    8(SP),(R1)+           ; Also trace caller's return point.
                 50    0ODC  C4 D0 04D3  1692          MOVL    PDT$L_UCB0(R4),R0     ; R0 => UCB.
               FC A1   0088  C0 C2 04D8  1693          SUBL    UCB$L_DDT(R0),-4(R1)  ; Make traced return point relative.
                     81  01  CE 04DE  1694          MNEGL   #1,(R1)+              ; Flag marks end of trace entry.
                                04E1  1695 30$:
                     50  8E  7D 04E1  1696          MOVQ    (SP)+,R0              ; Restore R0 and R1.
                         05  05 04E4  1697          RSB
```

```
                        04E5  1699          .SBTTL  +       INIT_UDA_BUFFERS
                        04E5  1700
                        04E5  1701 ;+
                        04E5  1702 ; INIT_UDA_BUFFERS - accomplishes the following:
                        04E5  1703 ;
                        04E5  1704 ;       1. It fills in UCB$W_PU_BCNT, UCB$W_PU_BOFF and UCB$L_PU_SVAPTE and clears
                        04E5  1705 ;             CRB$L_INTD+VEC$Q_MAPREG. It then copies them to UCB$W_BCNT,
                        04E5  1706 ;             UCB$W_BOFF and UCB$L_SVAPTE so that the standard UNIBUS
                        04E5  1707 ;             map register allocation routine will allocate enough
                        04E5  1708 ;             of these map registers to map the area located at
                        04E5  1709 ;             PDT$L_COMAREA which is PDT$C_RINGLEN long.
                        04E5  1710 ;             It then calls IOC$ALOUBMAPRM to permanently allocate
                        04E5  1711 ;             the map registers.  The value left in UCB$L_PU_SVAPTE, along
                        04E5  1712 ;             with the values left in CRB$L_INTD+VEC$W_MAPREG by
                        04E5  1713 ;             IOC$ALOUBMAPRM, allow the later loading of the allocated
                        04E5  1714 ;             map registers. (Note the map registers are loaded at
                        04E5  1715 ;             intialization time and after every reset of the port.)
                        04E5  1716 ;
                        04E5  1717 ;       2. It calculates the UNIBUS virtual address of the text portion
                        04E5  1718 ;             of each of the 2*UDA$K_RINGSIZE buffers and stores this
                        04E5  1719 ;             virtual address in the low order 18 bits of the buffer
                        04E5  1720 ;             header, at offset UDAB$L_DESCRIP.  The UDA_M_OWN and
                        04E5  1721 ;             UDA_M_FLAG bits are also set on in UDAB$L_DESCRIP so
                        04E5  1722 ;             that this value can now be moved directly into a ring
                        04E5  1723 ;             slot (be it command or response ring) to effect the transfer
                        04E5  1724 ;             of this buffer to the port.
                        04E5  1725 ;
                        04E5  1726 ; Inputs:
                        04E5  1727 ;       R5                                  -Addr of UCB
                        04E5  1728 ;
                        04E5  1729 ; Outputs:
                        04E5  1730 ;       R0                                  -Status code (SS$_NORMAL or 0 if no SVPN)
                        04E5  1731 ;       R1-R4                               -Destroyed
                        04E5  1732 ;       R5                                  -Preserved
                        04E5  1733 ;-
                        04E5  1734
                        04E5  1735          .enabl  lsb
                        04E5  1736
                        04E5  1737 INIT_UDA_BUFFERS:
                        04E5  1738
54   0084 C5    D0      04E5  1739          MOVL    UCB$L_PDT(R5),R4            ; R4 => PDT.
                        04EA  1740
                        04EA  1741          CPUDISP <<780,COM_INIT_UDA_BUFS>,-  ; Dispatch to allow special
                        04EA  1742                   <750,COM_INIT_UDA_BUFS>,-  ;  casing of uVAX I.
                        04EA  1743                   <730,COM_INIT_UDA_BUFS>,-
                        04EA  1744                   <790,COM_INIT_UDA_BUFS>,-
                        04EA  1745                   <UV1,INIT_UDA_BUFS_UV1>,-
                        04EA  1746                   <8SS,INIT_UDA_BUFS_8SS>,-
                        04EA  1747                  >
                        0504  1748
                        0504  1749 INIT_UDA_BUFS_8SS:
50   00E0 C4    D0      0504  1750          MOVL    PDT$L_ADP(R4),R0            ; R0 => ADP.
        01    B1        0509  1751          CMPW    #AT$_UBA,-                  ; See if on UNIBUS adapter.
      0E A0             050B  1752                  ADP$W_ADPTYPE(R0)
        03    13        050D  1753          BEQL    COM_INIT_UDA_BUFS           ; EQL implies UNIBUS adapter.
      01BD    31        050F  1754          BRW     INIT_BDA_BUFS               ; Else branch to init BDA buffers.
                        0512  1755
```

```
                                0512  1756 COM_INIT_UDA_BUFS:
                                0512  1757
                                0512  1758 ; Initialize fields in UCB so as to allow standard IOSUBNPAG routines
                                0512  1759 ;       to allocate MAP registers.
                                0512  1760
                     0A88 8F    B0  0512  1761         MOVW    #PDT$C_RINGLEN,-        ; Length of area to map.
                     00C2 C5        0516  1762                 UCB$W_PU_BCNT(R5)
                52   0200 C4    9E  0519  1763         MOVAB   PDT$L_COMAREA(R4),R2    ; R2 => area to map.
          00C0 C5   52   FE00 8F    AB  051E  1764         BICW3   #^XFE00,R2,-
                                0526  1765                 UCB$W_PU_BOFF(R5)       ; Get offset of area to map.
                      09    EF  0526  1766         EXTZV   S^#VA$V_VPN,-
                52    52    15        0528  1767                 S^#VA$S_VPN,R2,R2       ; R2 = virtual page # of area.
             50   00000000'GF    D0  052B  1768         MOVL    G^MMG$GL_SPTBASE,R0     ; R0 => system page table.
               00BC C5    6042    DE  0532  1769         MOVAL   (R0)[R2],UCB$L_PU_SVAPTE(R5)
                                0538  1770                                         ;
                                0538  1771
                53    24 A5    D0  0538  1772         MOVL    UCB$L_CRB(R5),R3        ; R3 => CRB.
                      34 A3    D4  053C  1773         CLRL    CRB$L_INTD+VEC$W_MAPREG(R3)
                                053F  1774                                         ; Initialize to direct datapath (0).
                                053F  1775         CPUDISP <<780,DATAPATH_780>,-   ; Label of VAX-11/780 specific code.
                                053F  1776                  <750,DATAPATH_750>,-   ; Label of VAX-11/750 specific code.
                                053F  1777                  <730,DATAPATH_730>,-   ; Label of VAX-11/730 specific code.
                                053F  1778                  <790,DATAPATH_790>,-   ; label of VAX-11/790 specific code.
                                053F  1779                  <8SS,DATAPATH_8SS>,-   ; label of VAX-11/8SS specific code.
                                053F  1780                  >
                                0555  1781 DATAPATH_8SS:
                     FC56 CF    9E  0555  1782         MOVAB   REQDATAP_8SS,-          ; Overlay transfer vector when running
                     FC40 CF        0559  1783                 REQDATAPATH_TV          ;  on a VAX-11/8SS.
                     FC77 CF    9E  055C  1784         MOVAB   RELDATAP_8SS,-          ; Overlay transfer vector when running
                     FC3D CF        0560  1785                 RELDATAPATH_TV          ;  on a VAX-11/8SS.
                          1E    11  0563  1786         BRB     DATAPATH_780            ; And branch around.
                                0565  1787
                                0565  1788 DATAPATH_750:
                     FC3B CF    9E  0565  1789         MOVAB   REQDATAP_750,-          ; Overlay transfer vector when running
                     FC30 CF        0569  1790                 REQDATAPATH_TV          ;  on a VAX-11/750.
                     FC5C CF    9E  056C  1791         MOVAB   RELDATAP_750,-          ; Overlay transfer vector when running
                     FC2D CF        0570  1792                 RELDATAPATH_TV          ;  on a VAX-11/750.
                          0E    11  0573  1793         BRB     DATAPATH_780            ; And branch around.
                                0575  1794
                                0575  1795 DATAPATH_730:
                     FC6A CF    9E  0575  1796         MOVAB   REQDATAP_730,-          ; Overlay transfer vector when running
                     FC20 CF        0579  1797                 REQDATAPATH_TV          ;  on a VAX-11/730.
                     FC63 CF    9E  057C  1798         MOVAB   RELDATAP_730,-          ; Overlay transfer vector when running
                     FC1D CF        0580  1799                 RELDATAPATH_TV          ;  on a VAX-11/730. Then fall thru.
                                0583  1800
                                0583  1801 DATAPATH_780:
                                0583  1802 DATAPATH_790:                             ; 11/790 mimics the 11/780.
                                0583  1803         ASSUME  PDT$B_BDPUSECNT EQ      PDT$B_DATAPATH+1
                     0129 C4    B4  0583  1804         CLRW    PDT$B_DATAPATH(R4)      ; Clear # of semi-perm BDP and use count.
                                0587  1805
                       37 A3    94  0587  1806         CLRB    CRB$L_INTD+VEC$B_DATAPATH(R3)   ; Clear CRB field.
                                058A  1807
                                058A  1808         ASSUME  UCB$W_BOFF      EQ      UCB$L_SVAPTE+4
                                058A  1809         ASSUME  UCB$W_BCNT      EQ      UCB$W_BOFF+2
                                058A  1810         ASSUME  UCB$W_PU_BOFF   EQ      UCB$L_PU_SVAPTE+4
                                058A  1811         ASSUME  UCB$W_PU_BCNT   EQ      UCB$W_PU_BOFF+2
                                058A  1812
```

```
                00BC C5   7D  058A  1813          MOVQ    UCB$L_PU_SVAPTE(R5),-   ; Copy parameters to standard
                   78 A5      058E  1814                  UCB$L_SVAPTE(R5)        ;  locations.
            00000000'GF   16  0590  1815          JSB     G^IOC$ALOUBMAPRM        ; Permanently allocate map registers.
                          0596  1816
                          0596  1817 ; Here calculate the UNIBUS virtual address of each buffer.
                          0596  1818
            50    00C0 C5   3C  0596  1819          MOVZWL  UCB$W_PU_BOFF(R5),R0    ; Byte offset of mapped area.
              51    24 A5   D0  059B  1820          MOVL    UCB$L_CRB(R5),R1        ; R1 => CRB
                          059F  1821
      50  09  09    34 A1   F0  059F  1822          INSV    CRB$L_INTD+VEC$W_MAPREG(R1),#9,#9,R0    ; R0 contains UNIBUS
                          05A5  1823                                                  ;  VA of base of area.
                          05A5  1824
                       50 DD  05A5  1825          PUSHL   R0                      ; Save on stack.
                          05A7  1826
                       50 D4  05A7  1827          CLRL    R0                      ; Initialize loop counter.
                          05A9  1828
            51   029C C4   9E  05A9  1829          MOVAB   PDT$L_PU_BUFARY+UDAB$T_TEXT(R4),R1      ; R1 => text portion
                          05AE  1830                                                  ;  of first buffer.
                          05AE  1831 5$:
                          05AE  1832 ;        ASSUME  UDAB$B_BUFFNO+1         is      Reserved byte.
              F6 A1    50 9B  05AE  1833          MOVZBW  R0,UDAB$B_BUFFNO-UDAB$T_TEXT(R1); Store index of this
                          05B2  1834                                                  ;  buffer in its header.
                    EC A1   9E  05B2  1835          MOVAB   -UDAB$T_TEXT(R1),-      ; Point Buffer table array
                 014C C440      05B5  1836                  PDT$L_BDTABLE(R4)[R0]   ;  element to this buffer.
            52    0200 C4   9E  05B9  1837          MOVAB   PDT$L_COMAREA(R4),R2    ; R2 => base of mapped area.
            52    51   52   C3  05BE  1838          SUBL3   R2,R1,R2                ; R2 = offset of this buffer
                          05C2  1839                                                  ;  from base of mapped area.
         F8 A1    52   6E   C1  05C2  1840          ADDL3   (SP),R2,-               ; Calculate UNIBUS VA of this
                          05C7  1841                  UDAB$L_DESCRIP-UDAB$T_TEXT(R1)  ;  buffer.
                          05C7  1842
                       C8 05C7  1843          BISL    #UDA_M_OWN!-                ; Set port ownership and mark
                          05C8  1844                   UDA_M_FLAG,-               ;  so port should interrupt on
      F8 A1    C0000000 8F      05C8  1845                  UDAB$L_DESCRIP-UDAB$T_TEXT(R1)  ;  ring transitions.
                          05CF  1846
         51    00000050 8F   C0  05CF  1847          ADDL    #UDAB$C_LENGTH,R1       ; R1 => text portion of next buffer.
                D4 50    20   F2  05D6  1848          AOBLSS  #2*UDA$R_RINGSIZE,R0,5$ ; Loop thru all buffers.
                          05DA  1849
                          8E  D5  05DA  1850          TSTL    (SP)+                   ; Clear stack.
                   50    01   D0  05DC  1851          MOVL    #SS$_NORMAL,R0          ; Set status return.
                      00F1   31  05DF  1852          BRW     INIT_UDA_BUFS_RTN       ; Branch around to return.
                          05E2  1853
                          05E2  1854 INIT_UDA_BUFS_UV1:
                          05E2  1855
                53   24 A5   D0  05E2  1856          MOVL    UCB$L_CRB(R5),R3        ; R3 => CRB.
                   34 A3   D4  05E6  1857          CLRL    CRB$L_INTD+VEC$W_MAPREG(R3)
                          05E9  1858                                                  ; Initialize to direct datapath (0).
                          05E9  1859
                          05E9  1860 ; To use the common allocate/deallocate map register routines, the ADP$L_CSR
                          05E9  1861 ; field in the Qbus adapter control block must be set to point to a base address
                          05E9  1862 ; such that the pseudo map registers existing in the PDT appear at the UBA$L_MAP
                          05E9  1863 ; offset from this base address.  Currently the SVAPTEs allocated at the time
                          05E9  1864 ; the adapter is initialized (INIADP) point to non-existant memory addresses.
                          05E9  1865 ; These must instead point to the 4 pages of map entries allocated as part
                          05E9  1866 ; of the PDT.
                          05E9  1867
                    00    E2  05E9  1868          BBSS    #MAP$V_MAPREGS,-        ; Ensure that we use only one set of
                FAE2 CF       05EB  1869                  PU$L_DRIVER_STS,-       ;  "map registers", gets mapped into
```

```
                 53        05EE  1870                15$                     ;  the virtual range pointed at by
                           05EF  1871                                        ;  ADP$L_CSR.  This bit allows us to
                           05EF  1872                                        ;  pass thru here just once, for the
                           05EF  1873                                        ;  first UQPORT found on a UV1 system.
                           05EF  1874
               51  04     D0    05EF  1875          MOVL    #4,R1             ; R1 contains pages needed for uVAX I
                           05F2  1876                                        ;  pseudo map registers.
                           05F2  1877
     00000000'GF   16      05F2  1878          JSB     G^EXE$ALOPHYCNTG        ; Allocate physically contiguous memory.
           08 50   E8      05F8  1879          BLBS    R0,8$                   ; LBS means successful allocation.
              01   CA      05FB  1880          BICL    #MAP$M_MAPREGS,-        ; Undo setting of bit above since we
          FAD0 CF           05FD  1881                  PU$L_DRIVER_STS       ;  have failed to create map registers.
             00D0   31      0600  1882          BRW     INIT_UDA_BUFS_RTN       ; And goto error return.
                            0603  1883 8$:
        0C94 C4   52        0603  1884          MOVL    R2,PDT$L_PQ_MAP(R4)     ; Remember address of map registers.
             51   52        0608  1885          MOVL    R2,R1                   ; Setup for call to BSBW to convert
                            060B  1886                                        ;  virtual address of map registers
                            060B  1887                                        ;  to physical address.
                00C6   30   060B  1888          BSBW    VIRT_TO_PHYAD           ; Convert R1 to physical address and
                            060E  1889                                        ;  leave result on top of stack.
                            060E  1890          ASSUME  ADP$L_CSR       EQ      0
           50   38 B3       060E  1891          MOVL    @CRB$L_INTD+VEC$L_ADP(R3),R0 ; Get address of Qbus adapter CSR.
50   50   15   09  EF       0612  1892          EXTZV   #VA$V_VPN,#VA$S_VPN,R0,R0 ; Convert from VA to VPN.
 51   00000000'GF  D0       0617  1893          MOVL    G^MMG$GL_SPTBASE,R1     ; Get address of system page table.
          50   6140 DE       061E  1894          MOVAL   (R1)[R0],R0             ; Get SVAPTE of Qbus adapter CSR.
             50   10  C0     0622  1895          ADDL    #<4*4>,R0               ; Point to SVAPTEs for map registers.
  51   6E   F7  8F  78       0625  1896          ASHL    #-VA$V_VPN,(SP),R1      ; Get PFN of pseudo map registers.
             7E   04  9A     062A  1897          MOVZBL  #4,-(SP)                ; Count of pages of map registers.
60   15   00   51  F0       062D  1898 10$:     INSV    R1,#PTE$V_PFN,#PTE$S_PFN,(R0) ; Set SPTE to map pseudo map reg.
             50   04  C0     0632  1899          ADDL    #4,R0                   ; Get SVAPTE of next PTE.
                  51  D6     0635  1900          INCL    R1                      ; Point to next physical PFN.
               F3 6E  F5     0637  1901          SOBGTR  (SP),10$                ; Loop to map all four pages.
             5E   04  C0     063A  1902          ADDL    #4,SP                   ; Clean counter off stack.
                            063D  1903
             FA92 CF 8ED0   063D  1904          POPL    PU$L_MPHYAD             ; Pop physical address into static.
                            0642  1905 15$:
                            0642  1906          ASSUME  PDT$B_BDPUSECNT EQ      PDT$B_DATAPATH+1
             0129 C4   B4   0642  1907          CLRW    PDT$B_DATAPATH(R4)      ; Clear # of semi-perm BDP and use count.
                            0646  1908
                  50   D4   0646  1909          CLRL    R0                      ; Initialize loop counter.
                            0648  1910
          51   029C C4   9E  0648  1911          MOVAB   PDT$L_PU_BUFARY+UDAB$T_TEXT(R4),R1 ; R1 => text portion
                            064D  1912                                        ;  of first buffer.
                            064D  1913 20$:
                            064D  1914 ;        ASSUME  UDAB$B_BUFFNO+1         is      Reserved byte.
            F6 A1   50   9B  064D  1915          MOVZBW  R0,UDAB$B_BUFFNO-UDAB$T_TEXT(R1); Store index of this
                            0651  1916                                        ;  buffer in its header.
                EC A1   9E  0651  1917          MOVAB   -UDAB$T_TEXT(R1),-      ; Point Buffer table array
              014C C440     0654  1918                  PDT$L_BDTABLE(R4)[R0]   ;  element to this buffer.
                            0658  1919
                   0079   30  0658  1920          BSBW    VIRT_TO_PHYAD           ; Convert R1 to physical address and
                            065B  1921                                        ;  leave result on top of stack.
8E   C00C0000 8F   C9       065B  1922          BISL3   #UDA_M_OWN!-            ; Set port ownership and mark
                            0662  1923                   UDA_M_FLAG,(SP)+,-     ;  so port should interrupt on
                   F8 A1    0662  1924                  UDAB$L_DESCRIP-UDAB$T_TEXT(R1) ;  ring transitions.
                            0664  1925
51   00000050 8F   C0       0664  1926          ADDL    #UDAB$C_LENGTH,R1       ; R1 => text portion of next buffer.
```

```
                DE 50    20    F2  066B  1927          AOBLSS  #2*UDA$K_RINGSIZE,R0,20$; Loop thru all buffers.
                                   066F  1928
                                   066F  1929 ; Here initialize the Aligned Page and its accompanying items.  These are:
                                   066F  1930 ;
                                   066F  1931 ;       1. PDT$L_PQ_PGQFL and PDT$L_PQ_PGQBL - An allocation queue header
                                   066F  1932 ;              wherein we queue CDRP's waiting waiting for this resource.
                                   066F  1933 ;
                                   066F  1934 ;       2. PDT$L_PQ_POWNER - The CDRP of the current owner of the Aligned Page.
                                   066F  1935 ;
                                   066F  1936 ;       3. PDT$L_PQ_PGPHAD - The Physical address of PDT$L_PQ_ALGNPG.
                                   066F  1937 ;
                                   066F  1938
                    0C98 C4    9E  066F  1939          MOVAB   PDT$L_PQ_PGQFL(R4),-    ; Set up Queue Header.
                    0C98 C4        0673  1940                  PDT$L_PQ_PGQFL(R4)
                    0C98 C4    9E  0676  1941          MOVAB   PDT$L_PQ_PGQFL(R4),-
                    0C9C C4        067A  1942                  PDT$L_PQ_PGQBL(R4)
                                   067D  1943
                    0CA0 C4    D4  067D  1944          CLRL    PDT$L_PQ_POWNER(R4)     ; Initialize to NO owner.
                51  0CB4 C4    9E  0681  1945          MOVAB   PDT$L_PQ_ALGNPG(R4),R1  ; R1 => Aligned Page.
                       004B    30  0686  1946          BSBW    VIRT_TO_PHYAD           ; Convert R1 to physical address and
                                   0689  1947                                          ;  leave result on top of stack.
                    0CA4 C4 8ED0   0689  1948          POPL    PDT$L_PQ_PGPHAD(R4)     ; Save Aligned Page physical address.
                                   068E  1949
                                   068E  1950 ; Since the INIT logic only allocates a system virtual page for the class
                                   068E  1951 ; driver, we must allocate a page for the boot driver in its initialization
                                   068E  1952 ; routine.  We initialize the PDT fields associated with the System Virtual Page
                                   068E  1953 ; that we use to map the user's non-word aligned buffer so that we can copy
                                   068E  1954 ; it to the Aligned Page.  The fields are PDT$L_PQ_UBFSVA that we initialize
                                   068E  1955 ; to be the System Virtual Address of this page; and PDT$L_PQ_SVPPTE that
                                   068E  1956 ; we set to be the virtual address of the PTE that describes this page.
                                   068E  1957
                       51    01    9A  068E  1958          MOVZBL  #1,R1                   ; Request one SVPN.
                  00000000'GF    16  0691  1959          JSB     G^IOC$ALLOSPT           ; Allocate the system virtual page.
                        01 50    E8  0697  1960          BLBS    R0,30$                  ; Br if successful allocation.
                                   05  069A  1961          RSB                             ; Return error to caller.
                51  00000000'9F    D0  069B  1962 30$:     MOVL    @#MMG$GL_SPTBASE,R1     ; Get address of SPT (PIC)
                    901FFFFF 8F    D0  06A2  1963          MOVL    #<PTE$C_RW!PTE$M_VALID!PTE$M_PFN>,- ; Set SPTE to valid,
                            6142       06A8  1964                  (R1)[R2]                ;  writable, and non-existant PFN.
                      74 A5    52    D0  06AA  1965          MOVL    R2,UCB$L_SVPN(R5)       ; Get Virtual Page Number of page.
                   51    52    09    78  06AE  1966          ASHL    #9,R2,R1                ; Multipy by 512 to get relative address
0CA8 C4            51  80000000 8F    C9  06B2  1967          BISL3   #^a31,R1,-              ;  in system space.  Then set high
                                   06BC  1968                  PDT$L_PQ_UBFSVA(R4)     ;  order bit to make system address.
                53  00000000'GF    D0  06BC  1969          MOVL    G^MMG$GL_SPTBASE,R3     ; R3 => system page table.
                  0CAC C4    6342    DE  06C3  1970          MOVAL   (R3)[R2],-              ; Calcualte address of PTE slot that
                                   06C9  1971                  PDT$L_PQ_SVPPTE(R4)     ;  describes this page.
                          50    01    D0  06C9  1972          MOVL    #SS$_NORMAL,R0          ; Set return code.
                             0004    31  06CC  1973          BRW     INIT_UDA_BUFS_RTN       ; Branch around to common return.
                                   06CF  1974
                                   06CF  1975 INIT_BDA_BUFS:
                                   06CF  1976          BUG_CHECK       UDAPORT,FATAL   ; For now only a place holder.
                                   06D3  1977
                                   06D3  1978 INIT_UDA_BUFS_RTN:                       ; Common return label.
                                   06D3  1979
                                05  06D3  1980          RSB                             ; Return to caller
                                   06D4  1981          .dsabl  lsb
```

```
                                06D4  1983          .SBTTL  +       VIRT_TO_PHYAD
                                06D4  1984
                                06D4  1985 ;+
                                06D4  1986 ; VIRT_TO_PHYAD - routine to convert a System Virtual Address into a physical
                                06D4  1987 ;       address.
                                06D4  1988 ;
                                06D4  1989 ; Inputs:
                                06D4  1990 ;       R1 = System Virtual Address to convert
                                06D4  1991 ;
                                06D4  1992 ; Outputs:
                                06D4  1993 ;       Physical Address left on top of stack.
                                06D4  1994 ;       All registers preserved.
                                06D4  1995 ;
                                06D4  1996
                                06D4  1997 VIRT_TO_PHYAD:
                                06D4  1998
                       6E   DD  06D4  1999          PUSHL   (SP)                    ; Create slot at top of stack for return
                                06D6  2000                                          ;  value.
                       0F   BB  06D6  2001          PUSHR   #^M<R0,R1,R2,R3>        ; Save registers.
     50    51   FFFFFE00 8F   CB  06D8  2002          BICL3   #-512,R1,R0             ; R0 = Boff of address.
                       09   EF  06E0  2003          EXTZV   #VA$V_VPN,-             ; Extract VPN
          52    51    15       06E2  2004                  #VA$S_VPN,R1,R2         ;  and put it in R2.
          53   00000000'GF   D0  06E5  2005          MOVL    G^MMG$GL_SPTBASE,R3     ; R3 => system page table.
                  53   6342   D0  06EC  2006          MOVL    (R3)[R2],R3             ; R3 => PTE.
                                06F0  2007
                       00   EF  06F0  2008          EXTZV   #PTE$V_PFN,-            ; Get page frame number of buffer
          53    53    15       06F2  2009                  #PTE$S_PFN,R3,R3        ;  page into R3.
          53    53    09   78  06F5  2010          ASHL    #VA$V_VPN,R3,R3         ; Shift into place for physical address.
       14 AE    53    50   C9  06F9  2011          BISL3   R0,R3,20(SP)            ; Put result into stack slot.
                       0F   BA  06FE  2012          POPR    #^M<R0,R1,R2,R3>        ; Restore registers.
                            05  0700  2013          RSB                             ; Return to caller.
```

```
                     0701  2015          .SBTTL  +       INIT_INIT_STEPS
                     0701  2016
                     0701  2017 ;+
                     0701  2018 ; INIT_INIT_STEPS - initialize the host responses for UDA hardware
                     0701  2019 ;                  initialization.
                     0701  2020 ;
                     0701  2021 ; Inputs:
                     0701  2022 ;       R5                              -Addr of UCB
                     0701  2023 ;
                     0701  2024 ; Outputs:
                     0701  2025 ;       UCB$W_HOSTSTEPx fields all initialized
                     0701  2026 ;
                     0701  2027 ;       Registers R0-R2 destroyed
                     0701  2028 ;-
                     0701  2029
                     0701  2030          .enabl  lsb
                     0701  2031
                     0701  2032 INIT_INIT_STEPS:
                     0701  2033
                     0701  2034 ; First calculate the vector for this UDA.  This is done by scanning the
                     0701  2035 ;       UNIBUS adapter interrupt vector looking for a slot that points
                     0701  2036 ;       into this CRB$L_INTD.
                     0701  2037
50   24 A5    D0     0701  2038          MOVL    UCB$L_CRB(R5),R0                ; R0 => CRB.
     24 A0    DF     0705  2039          PUSHAL  CRB$L_INTD(R0)                  ; Save interrupt dispatcher
                     0708  2040                                                  ;  address on stack.
50   2C A0    D0     0708  2041          MOVL    CRB$L_INTD+VEC$L_IDB(R0),R0     ; R0 => IDB.
50   14 A0    D0     070C  2042          MOVL    IDB$L_ADP(R0),R0                ; R0 => ADP.
50   10 A0    D0     0710  2043          MOVL    ADP$L_VECTOR(R0),R0             ; R0 => UBA interrupt vectors.
                     0714  2044
        51    D4     0714  2045          CLRL    R1                          ; Initialize counter.
                     0716  2046 10$:
           52 80 D0  0716  2047          MOVL    (R0)+,R2                    ; R2 = contents of a vector.
           52 03 CA  0719  2048          BICL    #3,R2                       ; Clear low order 2 bits to remove
                     071C  2049                                              ;  processor (780 vs 750) differences.
           6E 52 D1  071C  2050          CMPL    R2,(SP)                     ; Is this our vector?
              0D 13  071F  2051          BEQL    20$                         ; EQL means yes.
ED 51 00000080 8F F2  0721  2052          AOBLSS  #NUMUBAVEC,R1,10$           ; Else loop back to try again.
              8E D5  0729  2053          TSTL    (SP)+                       ; It had better be there
              50 D4  072B  2054          CLRL    R0                          ;  or else we return with an error
                 05  072D  2055          RSB                                 ;  indication.
                     072E  2056 20$:
              8E D5  072E  2057          TSTL    (SP)+                       ; Remove unneeded stack value.
                     0730  2058
              51 A9  0730  2059          BISW3   R1,-                        ; Or in vector address/4
                     0732  2060                  #HS1_M_IE!-                 ;   (i.e. vector #) with
                     0732  2061                  <UDA$K_RINGEXP@HS1_V_RRNGLEN>!- ;  Interrupt Enable bit, and
                     0732  2062                  <UDA$K_RINGEXP@HS1_V_CRNGLEN>!- ;  the response and command ring
                     0732  2063                   HS1_M_BIT15,-              ;  lengths and the high bit on,
 00AE C5 A480 8F     0732  2064                  UCB$W_HOSTSTEP1(R5)         ;  and save it in the UCB.
                     0738  2065
                     0738  2066 ; Here we calculate the UNIBUS virtual address of the ring base so as to be
                     0738  2067 ;       able to communicate it to the controller.
                     0738  2068
                     0738  2069
                     0738  2070          CPUDISP <<780,RINGBASE_COM>,-
                     0738  2071                   <750,RINGBASE_COM>,-
```

```
                     0738  2072                    <730,RINGBASE_COM>,-
                     0738  2073                    <790,RINGBASE_COM>,-
                     0738  2074                    <UV1,RINGBASE_UV1>,-
                     0738  2075                    <8SS,RINGBASE_8SS>,-
                     0738  2076                   >
                     0752  2077
                     0752  2078 RINGBASE_8SS:
          50  00E0 C4    D0  0752  2079           MOVL    PDT$L_ADP(R4),R0        ; R0 => ADP.
                  01    B1   0757  2080           CMPW    #ATA_UBA,-              ; See if on UNIBUS adapter.
                0E A0        0759  2081                   ADP$W_ADPTYPE(R0)
                     04    13  075B  2082           BEQL    RINGBASE_COM            ; EQL implies UNIBUS adapter.
                           075D  2083           BUG_CHECK       UDAPORT,FATAL       ; For now, if on BDA.
                     0761  2084
                     0761  2085 RINGBASE_COM:
                     0761  2086
          50  00C0 C5    3C  0761  2087           MOVZWL  UCB$W_PU_BOFF(R5),R0    ; R0 contains byte offset of ringbase.
            51    24 A5    D0  0766  2088           MOVL    UCB$L_CRB(R5),R1        ; R1 => CRB.
                     076A  2089
50  09    09    34 A1    F0  076A  2090           INSV    CRB$L_INTD+VEC$W_MAPREG(R1),#9,#9,R0    ; R0 contains the UBA
                     0770  2091                                                                    ;  virtual address of
                     0770  2092                                                                    ;  the communications
                     0770  2093                                                                    ;  area.
               50    08    C0  0770  2094           ADDL    #PDT$L_RINGBASE-PDT$L_COMAREA,R0; Add in offset to ringbase.
                     12    11  0773  2095           BRB     AFTER_RINGBASE                  ; Rejoin common code.
                     0775  2096
                     0775  2097 RINGBASE_UV1:
                     0775  2098
          51  0084 C5    D0  0775  2099           MOVL    UCB$L_PDT(R5),R1        ; Get the PDT address.
        51  00000208 8F    C0  077A  2100           ADDL    #PDT$C_RINGBASE,R1      ; Point to the ring base.
                FF50    30  0781  2101           BSBW    VIRT_TO_PHYAD           ; Convert VA to physical address.
                  50 8ED0   0784  2102           POPL    R0                      ; Get physical address of ring base.\
                     0787  2103
                     0787  2104 AFTER_RINGBASE:
                     0787  2105
                     0787  2106           CPUDISP <<780,PURGE_780>,-
                     0787  2107                    <750,PURGE_750>,-
                     0787  2108                    <730,PURGE_730>,-
                     0787  2109                    <790,PURGE_790>,-
                     0787  2110                    <UV1,PURGE_UV1>,-
                     0787  2111                    <8SS,PURGE_8SS>,-
                     0787  2112                   >
                     07A1  2113
                     07A1  2114 PURGE_780:
                     07A1  2115 PURGE_790:                                       ; 11/790 acts same as 11/780.
          50  01    A8  07A1  2116           BISW    #HS2_M_PI,R0                ; Ask for purge interrupts when
                     07A4  2117                                                   ;  running on a VAX-11/780.
                     07A4  2118 PURGE_750:
                     07A4  2119 PURGE_730:
                     07A4  2120 PURGE_UV1:
                     07A4  2121 PURGE_8SS:
                     07A4  2122
00B2 C5    50    B0  07A4  2123           MOVW    R0,UCB$W_HOSTSTEP2(R5)      ; Save communication area address and
                     07A9  2124                                                   ;  purge interrupt request bit in UCB.
                     07A9  2125
                     07A9  2126           ASSUME  UCB$W_HOSTSTEP3 EQ      UCB$W_PORTSTEP3+2
00B4 C5    50    D0  07A9  2127           MOVL    R0,UCB$W_PORTSTEP3(R5)      ; Trash UCB$W_PORTSTEP3 as we store the
                     07AE  2128                                                   ;  high word of R0 in UCB$W_HOSTSTEP3.
```

```
                                07AE  2129
                                07AE  2130 ; Here we pickup the SYSGEN parameter UDABURSTRATE.
                                07AE  2131
     50   00000000'GF    9A  07AE  2132          MOVZBL  G^SCS$GB_UDABURST,R0    ; R0 contains the burst rate setting.
          50    50   02   78  07B5  2133          ASHL    #HS4_V_BURST,R0,R0      ; Shift burst rate into position.
                         A9  07B9  2134          BISW3   #HS4_M_GO!-             ; Set GO bit and last fail packet bits,
                                07BA  2135                   HS4_M_LF,-         ;  or in burst rate,
00BA C5  50    03            07BA  2136                   R0,-                   ;  and save as HOST response to STEP 4.
                                07BF  2137                   UCB$W_HOSTSTEP4(R5)
                                07BF  2138
               50    01   D0  07BF  2139          MOVL    S^#SS$_NORMAL,R0        ; Indicate success and
                         05  07C2  2140          RSB                             ;  return to caller.
                                07C3  2141          .dsabl  lsb
```

```
                        07C3  2143          .SBTTL  +       Hardware Initialization
                        07C3  2144
                        07C3  2145 ;+
                        07C3  2146 ; HARDWARE_INIT - internal subroutine
                        07C3  2147 ;
                        07C3  2148 ; Inputs:
                        07C3  2149 ;       R5                                  -Addr of UCB
                        07C3  2150 ;
                        07C3  2151 ; Outputs:
                        07C3  2152 ;
                        07C3  2153 ;-
                        07C3  2154
                        07C3  2155 CNTRLTYP_ARRAY:
                0000    07C3  2156          .WORD   UDA50_CNTRLTYP
                0001    07C5  2157          .WORD   LESI_CNTRLTYP
                0005    07C7  2158          .WORD   TU81_CNTRLTYP
                0006    07C9  2159          .WORD   UDA50A_CNTRLTYP
                0007    07CB  2160          .WORD   RDRX_CNTRLTYP
                0003    07CD  2161          .WORD   MAYA_CNTRLTYP
                000D    07CF  2162          .WORD   QDA50_CNTRLTYP
                FFFF    07D1  2163          .WORD   -1                          ; End of array fence.
    FFFF FFFF FFFF FFFF FFFF    07D3  2164          .WORD   -1,-1,-1,-1,-1              ; Space for expansion.
                        07DD  2165 DEVTYPE_ARRAY:
                  03    07DD  2166          .BYTE   DT$_UDA50
                  05    07DE  2167          .BYTE   DT$_LESI
                  06    07DF  2168          .BYTE   DT$_TU81P
                  04    07E0  2169          .BYTE   DT$_UDA50A
                  07    07E1  2170          .BYTE   DT$_RDRX
                  08    07E2  2171          .BYTE   DT$_TK50P
                  0B    07E3  2172          .BYTE   DT$_QDA50
    00 00 00 00 00 00   07E4  2173          .BYTE   0,0,0,0,0,0                 ; Space for expansion.
                        07EA  2174 SERVERS_ARRAY:
                  05    07EA  2175          .BYTE   <1@DISK_CONID>!<1@DUP_CONID>    ; UDA supports disks and DUP.
                  05    07EB  2176          .BYTE   <1@DISK_CONID>!<1@DUP_CONID>    ; Aztec supports disks and DUP.
                  06    07EC  2177          .BYTE   <1@TAPE_CONID>!<1@DUP_CONID>    ; TU81 supports tapes and DUP.
                  05    07ED  2178          .BYTE   <1@DISK_CONID>!<1@DUP_CONID>    ; UDA50A supports disks and DUP.
                  05    07EE  2179          .BYTE   <1@DISK_CONID>!<1@DUP_CONID>    ; RD/RX supports disks and DUP.
                  06    07EF  2180          .BYTE   <1@TAPE_CONID>!<1@DUP_CONID>    ; MAYA supports tapes and DUP.
                  05    07F0  2181          .BYTE   <1@DISK_CONID>!<1@DUP_CONID>    ; QDA50 supports disks and DUP.
    00 00 00 00 00 00   07F1  2182          .BYTE   0,0,0,0,0,0                 ; Space for expansion.
                        07F7  2183 UCODE_VER_ARRAY:                             ; Minimum ucode version levels
                  00    07F7  2184          .BYTE   0                           ; UDA50A (0 since special purpose
                        07F8  2185                                              ;  code already checks).
                  00    07F8  2186          .BYTE   0                           ; RC25 (0 for now)
                  00    07F9  2187          .BYTE   0                           ; TU81 (0 for now)
                  00    07FA  2188          .BYTE   0                           ; UDA50A (0 since special purpose
                        07FB  2189                                              ;  code already checks).
                  09    07FB  2190          .BYTE   9                           ; RDRX.
                  00    07FC  2191          .BYTE   0                           ; MAYA.
                  00    07FD  2192          .BYTE   0                           ; QDA.
    00 00 00 00 00 00   07FE  2193          .BYTE   0,0,0,0,0,0                 ; Space for expansion.
                        0804  2194 OPC_MSG_ARRAY:
                  57    0804  2195          .BYTE   MSG$_UDA50MVER              ; UDA50 Opcom message number
                  5D    0805  2196          .BYTE   MSG$_RC25MVER               ; RC25 Opcom message number
                  5F    0806  2197          .BYTE   MSG$_TU81MVER               ; TU81 Opcom message number
                  57    0807  2198          .BYTE   MSG$_UDA50MVER              ; UDA50A Opcom message number
                  5E    0808  2199          .BYTE   MSG$_RDRXMVER               ; RDRX Opcom message number
```

```
                 5F   0809  2200          .BYTE   MSG$_TU81MVER                   ; Use TU81 Opcom message number
                      080A  2201                                                  ;  for MAYA for now.
                 57   080A  2202          .BYTE   MSG$_UDA50MVER                  ; Use UDA50 Opcom message number
                      080B  2203                                                  ;  for QDA for now.
00 00 00 00 00 00     080B  2204          .BYTE   0,0,0,0,0,0                     ; Space for expansion.
                      0811  2205          .enabl  lsb
                      0811  2206 HARDWARE_INIT:
                      0811  2207
      009C C5 8ED0    0811  2208          POPL    UCB$L_DPC(R5)                   ; Remember return
      00A5 C5   97    0816  2209          DECB    UCB$B_INITCNT(R5)               ; Count # of consecutive times thru here.
            2A   12   081A  2210          BNEQ    8$                              ; NEQ means that we have not exhausted
                      081C  2211                                                  ;  consecutive retries.
   53   24 A5   D0    081C  2212          MOVL    UCB$L_CRB(R5),R3                ; R3 => CRB.
1C A3   2F'AF   9E    0820  2213          MOVAB   B^4$,CRB$L_TOUTROUT(R3)         ; Establish wakeup routine.
            0A   C1   0825  2214          ADDL3   #INIT_DELTA,-                   ; Establish a small delay.
 00000000'GF         0827  2215                  G^EXE$GL_ABSTIM,-
        18 A3        082C  2216                  CRB$L_DUETIME(R3)
                 05  082E  2217          RSB                                     ; Evaporate for awhile.
                      082F  2218 4$:
                      082F  2219          SETIPL  #IPL$_SCS                       ; Lower IPL after wakeup.
   54   10 A3   D0    0832  2220          MOVL    CRB$L_AUXSTRUC(R3),R4           ; R4 => PDT.
55    00DC C4   D0    0836  2221          MOVL    PDT$L_UCB0(R4),R5               ; R5 => UCB.
      F8BA CF   9E    083B  2222          MOVAB   NULL_ROUTINE,-                  ; Restabl.   null wakeup routine.
        1C A3         083F  2223                  CRB$L_TOUTROUT(R3)
            05   90   0841  2224          MOVB    #NO_CONSEC_INITS,-              ; Reinit count that went to zero.
      00A5 C5         0843  2225                  UCB$B_INITCNT(R5)
                      0846  2226 8$:
            04   A8   0846  2227          BISW    #UCB$M_PU_HRDINI,-              ; Set bit signalling
        68 A5         0848  2228                  UCB$W_DEVSTS(R5)                ;  that we initing the port.
54    0084 C5   D0    084A  2229          MOVL    UCB$L_PDT(R5),R4                ; R4 => PDT.
53    0100 C4   D0    084F  2230          MOVL    PDT$L_PU_CSR(R4),R3             ; R3 => UDA CSR.
      0203 C4   94    0854  2231          CLRB    PDT$B_PURGEDP(R4)               ; Prevent spurious purges.
                      0858  2232 10$:
                      0858  2233
            63   B4   0858  2234          CLRW    UDAIP(R3)                       ; Start hard init.
                      085A  2235
                      085A  2236 ; At least 100 micro seconds must pass between here
                      085A  2237
                      085A  2238          ASSUME  UCB$W_BOFF      EQ      UCB$L_SVAPTE+4
                      085A  2239          ASSUME  UCB$W_BCNT      EQ      UCB$W_BOFF+2
                      085A  2240          ASSUME  UCB$W_PU_BOFF   EQ      UCB$L_PU_SVAPTE+4
                      085A  2241          ASSUME  UCB$W_PU_BCNT   EQ      UCB$W_PU_BOFF+2
                      085A  2242
                      085A  2243
                      085A  2244          CPUDISP <<780,LOADUBA_COMMON>,-
                      085A  2245                   <750,LOADUBA_COMMON>,-
                      085A  2246                   <730,LOADUBA_COMMON>,-
                      085A  2247                   <790,LOADUBA_COMMON>,-
                      085A  2248                   <UV1,NOLOADUBA_UV1>,-
                      085A  2249                   <8SS,LOADUBA_8SS>,-
                      085A  2250                  >
                      0874  2251
                      0874  2252 LOADUBA_8SS:
50    00E0 C4   D0    0874  2253          MOVL    PDT$L_ADP(R4),R0                ; R0 => ADP.
            01   B1   0879  2254          CMPW    #AT$_UBA,-                      ; See if on UNIBUS adapter.
        0E A0         087B  2255                  ADP$W_ADPTYPE(R0)
            04   13   087D  2256          BEQL    LOADUBA_COMMON                  ; EQL implies UNIBUS adapter.
```

```
                     087F  2257          BUG_CHECK       UDAPORT,FATAL   ; For now, if on BDA.
                     0883  2258
                     0883  2259 LOADUBA_COMMON:
                     0883  2260
          00BC C5    7D  0883  2261          MOVQ    UCB$L_PU_SVAPTE(R5),-   ; Copy parameters that allow mapping
            78 A5        0887  2262                  UCB$L_SVAPTE(R5)        ;  of communication area and buffers.
                     0889  2263          LOADUBA                         ; Load UNIBUS map registers with values
                     088F  2264                                          ;  that map communication area and buffers.
                     088F  2265 NOLOADUBA_UV1:
                     088F  2266
                     088F  2267          DSBINT
                     0895  2268          WFIKPCH 15$,#2                  ; Cause timeout to waste time.
                     089F  2269 15$:
                     089F  2270          SETIPL  UCB$B_FIPL(R5)          ; Lower IPL after timeout.
              0282   30  08A3  2271          BSBW    INIT_PU_PDT             ; Initialize variable PDT data.
          00AC C5    B4  08A6  2272          CLRW    UCB$W_PORTSTEP1(R5)     ; Clear port responses so that
          00B0 C5    B4  08AA  2273          CLRW    UCB$W_PORTSTEP2(R5)     ;  the error log record we create
          00B4 C5    B4  08AE  2274          CLRW    UCB$W_PORTSTEP3(R5)     ;  will contain only the responses
          00B8 C5    B4  08B2  2275          CLRW    UCB$W_PORTSTEP4(R5)     ;  received in this INIT attempt.
          00A6 C5    B6  08B6  2276          INCW    UCB$W_NUMBINITS(R5)     ; Increment # of times we have attempted
                     08BA  2277                                          ;  to init port.
                     08BA  2278 ; and here
                     08BA  2279
            02 A3    B0  08BA  2280          MOVW    UDASA(R3),-             ; Read SA register.
          00AC C5        08BD  2281                  UCB$W_PORTSTEP1(R5)
               0F    E1  08C0  2282          BBC     #PS1_V_ER,-             ; Test for error condition
       03 00AC C5        08C2  2283                  UCB$W_PORTSTEP1(R5),30$ ;  and branch around if clear.
                     08C6  2284 20$:
              01E9   31  08C6  2285          BRW     HARD_RETRY              ; For now only.
                     08C9  2286 30$:
                 0B  E1  08C9  2287          BBC     #PS1_V_S1,-             ; Make sure we are in STEP1
            F7 00AC C5        08CB  2288                  UCB$W_PORTSTEP1(R5),20$ ;  or else go back.
                     08CF  2289
                     08CF  2290 ; In Step 1.
                     08CF  2291
                     08CF  2292          DSBINT                          ; Prepare to respond to STEP1.
                 05  E5  08D5  2293          BBCC    #UCB$V_POWER,-          ; Power failure negates what we have
            03 64 A5        08D7  2294                  UCB$W_STS(R5),35$       ;  done until now.
              01D2   31  08DA  2295          BRW     HARDPOWER               ; If power failure branch around.
                     08DD  2296 35$:
            00AE C5    B0  08DD  2297          MOVW    UCB$W_HOSTSTEP1(R5),-   ; Else move value to SA register.
              02 A3        08E1  2298                  UDASA(R3)
                     08E3  2299          WFIKPCH STEP1_TIMEOUT,#STEP1_LIMIT      ; Wait for interrupt.
                     08ED  2300
                     08ED  2301          IOFORK                          ; Lower IPL and continue.
              01FE   30  08F3  2302          BSBW    WAIT100US               ; Call to wait for at least 100 uSecs.
              02 A3    B0  08F6  2303          MOVW    UDASA(R3),-             ; For slow TU81, copy UDASA after fork-
          00AA C5        08F9  2304                  UCB$W_UDASA(R5)         ;  ing so as to give it enough time.
          00AA C5    B0  08FC  2305          MOVW    UCB$W_UDASA(R5),-               ; Save Port STEP2 start value
          00B0 C5        0900  2306                  UCB$W_PORTSTEP2(R5)
                     0903  2307
                     0903  2308 ; In Step 2.
                     0903  2309
    06 00B0 C5   0F  E0  0903  2310          BBS     #PS2_V_ER,UCB$W_PORTSTEP2(R5),37$       ; For now!!!!
                     0909  2311
                 0C  E0  0909  2312          BBS     #PS2_V_S2,-             ; Make sure we are in STEP2.
            03 00B0 C5        090B  2313                  UCB$W_PORTSTEP2(R5),40$
```

```
                     090F  2314 37$:
                01A0 31 090F  2315          BRW     HARD_RETRY                  ; Else branch to retry.
                     0912  2316 40$:
            00B0 C5  91 0912  2317          CMPB    UCB$W_PORTSTEP2(R5),-       ; Test whether controller echoed
            00AF C5     0916  2318                  UCB$W_HOSTSTEP1+1(R5)       ;  fields correctly.
                03  13 0919  2319          BEQL    50$                         ; EQL implies yes.
                0194 31 091B  2320          BRW     HARD_RETRY                  ; Else branch to retry.
                     091E  2321 50$:
                     091E  2322          DSBINT                              ; Prepare to respond to STEP2.
                05  E5 0924  2323          BBCC    #UCB$V_POWER,-              ; Power failure negates what we have
         03 64 A5       0926  2324                  UCB$W_STS(R5),55$           ;  done until now.
                0183 31 0929  2325          BRW     HARDPOWER                   ; If power failure branch around.
                     092C  2326 55$:
            00B2 C5  B0 092C  2327          MOVW    UCB$W_HOSTSTEP2(R5),-       ; Else move value to SA register.
              02 A3     0930  2328                  UDASA(R3)
                     0932  2329          WFIKPCH STEP2_TIMEOUT,#STEP2_LIMIT      ; Wait for interrupt.
                     093C  2330
                     093C  2331          IOFORK                              ; Lower IPL and continue.
                01AF 30 0942  2332          BSBW    WAIT100US                   ; Call to wait for at least 100 uSecs.
              02 A3  B0 0945  2333          MOVW    UDASA(R3),-                 ; For slow TU81, copy UDASA after fork-
            00AA C5     0948  2334                  UCB$W_UDASA(R5)             ;  ing so as to give it enough time.
            00AA C5  B0 094B  2335          MOVW    UCB$W_UDASA(R5),-           ; Save Port STEP3 start value
            00B4 C5     094F  2336                  UCB$W_PORTSTEP3(R5)
                     0952  2337
                     0952  2338 ; In Step 3.
                     0952  2339
     06 00B4 C5  0F  E0 0952  2340          BBS     #PS3_V_ER,UCB$W_PORTSTEP3(R5),57$   ; For now!!!!
                     0958  2341
                0D  E0 0958  2342          BBS     #PS3_V_S3,-                 ; Make sure we are in STEP3.
         03 00B4 C5     095A  2343                  UCB$W_PORTSTEP3(R5),60$
                     095E  2344 57$:
                0151 31 095E  2345          BRW     HARD_RETRY                  ; Else branch to retry.
                     0961  2346 60$:
            00B4 C5  91 0961  2347          CMPB    UCB$W_PORTSTEP3(R5),-       ; Test whether controller echoed
            00AE C5     0965  2348                  UCB$W_HOSTSTEP1(R5)         ;  fields correctly.
                03  13 0968  2349          BEQL    70$                         ; EQL implies yes.
                0145 31 096A  2350          BRW     HARD_RETRY                  ; Else branch to retry.
                     096D  2351 70$:
                     096D  2352          DSBINT                              ; Prepare to respond to STEP3.
                05  E5 0973  2353          BBCC    #UCB$V_POWER,-              ; Power failure negates what we have
         03 64 A5       0975  2354                  UCB$W_STS(R5),72$           ;  done until now. No failure, continue.
                0134 31 0978  2355          BRW     HARDPOWER                   ; So branch to HARDPOWER if failure.
                     097B  2356 72$:
            00B6 C5  B0 097B  2357          MOVW    UCB$W_HOSTSTEP3(R5),-       ; Else move value to SA register.
              02 A3     097F  2358                  UDASA(R3)
                     0981  2359          WFIKPCH STEP3_TIMEOUT,#STEP3_LIMIT      ; Wait for interrupt.
                     098B  2360
                     098B  2361          IOFORK                              ; Lower IPL and continue.
                0160 30 0991  2362          BSBW    WAIT100US                   ; Call to wait for at least 100 uSecs.
              02 A3  B0 0994  2363          MOVW    UDASA(R3),-                 ; For slow TU81, copy UDASA after fork-
            00AA C5     0997  2364                  UCB$W_UDASA(R5)             ;  ing so as to give it enough time.
            00AA C5  B0 099A  2365          MOVW    UCB$W_UDASA(R5),-           ; Save Port STEP4 start value
            00B8 C5     099E  2366                  UCB$W_PORTSTEP4(R5)
                     09A1  2367
                     09A1  2368 ; In Step 4.
                     09A1  2369
     06 00B8 C5  0F  E0 09A1  2370          BBS     #PS4_V_ER,UCB$W_PORTSTEP4(R5),77$   ; For now!!!!
```

```
                     09A7  2371
          0E   E0   09A7  2372          BBS     #PS4_V_S4,-                ; Make sure we are in STEP4.
03 00B8   C5        09A9  2373                  UCB$W_PORTSTEP4(R5),80$
                    09AD  2374 77$:
        0102   31   09AD  2375          BRW     HARD_RETRY                 ; Else branch around to retry.
                    09B0  2376 80$:
          04   ED   09B0  2377          CMPZV   #PS4_V_CNTRLTYP,-          ; Here we assure that the controller
          07        09B2  2378                  #PS4_S_CNTRLTYP,-          ;  microcode is upto rev level. First
     00B8 C5        09B3  2379                  UCB$W_PORTSTEP4(R5),-      ;  see if we have a UDA50.
          00        09B6  2380                  #UDA50_CNTRLTYP
          13   12   09B7  2381          BNEQ    90$                        ; If NOT, branch around.
          03   90   09B9  2382          MOVB    #DT$_UDA50,-               ; Fill in UCB devtype field with value
       41 A5        09BB  2383                  UCB$B_DEVTYPE(R5)          ;  for UDA50.
          00   ED   09BD  2384          CMPZV   #PS4_V_UCODEVER,-          ; If a UDA50, see if microcode
          04        09BF  2385                  #PS4_S_UCODEVER,-          ;  upto rev by comparing against out of
     00B8 C5        09C0  2386                  UCB$W_PORTSTEP4(R5),-      ;  date microcode version #.
          01        09C3  2387                  #1                         ; Out of date version number.
          06   12   09C4  2388          BNEQ    90$                        ; NEQ implies Ucode is OK.
        0116   31   09C6  2389          BRW     UDA_OUTOFREV               ; EQL implies inoperative Ucode.
                    09C9  2390
                    09C9  2391 GOTO_HARDP:
        00E3   31   09C9  2392          BRW     HARDPOWER                  ; Branch to faraway label.
                    09CC  2393 90$:
        01DB   30   09CC  2394          BSBW    STOCK_RSPRING              ; Stock response ring after controller
                    09CF  2395                                             ;  clears it.
                    09CF  2396
                    09CF  2397          DSBINT                             ; Prepare to respond to STEP4.
          05   E4   09D5  2398          BBSC    #UCB$V_POWER,-             ; Power failure negates what we have
    EF 64 A5        09D7  2399                  UCB$W_STS(R5),GOTO_HARDP;  ;  done until now.
     00BA C5   B0   09DA  2400          MOVW    UCB$W_HOSTSTEP4(R5),-      ; Else move value to SA register.
       02 A3        09DE  2401                  UDASA(R3)
                    09E0  2402          ENBINT                             ; Lower IPL and
          04   AA   09E3  2403          BICW    #UCB$M_PU_HRDINI,-         ; Reset bit signalling
       68 A5        09E5  2404                  UCB$W_DEVSTS(R5)           ;  that we initing the port.
          01   B0   09E7  2405          MOVW    #INIT_ATTNCODE,-           ; Indicate what kind of error log record
        00A8 C5     09E9  2406                  UCB$W_ATTNCODE(R5)         ;  we are about to create.
00000000'GF   16    09EC  2407          JSB     G^ERL$DEVICEATTN           ; Call to create error log record of INIT.
                    09F2  2408
                    09F2  2409 ; Here we have initialized the port.  If it is a UDA, permanently allocate
                    09F2  2410 ; a Buffered Datapath and bias the useage count so that it is never released.
                    09F2  2411 ; This is done only on the first time thru here.
                    09F2  2412
          05   E3   09F2  2413          BBCS    #UCB$V_PU_BDPATH,-         ; Make sure first time thru.
    03 68 A5        09F4  2414                  UCB$W_DEVSTS(R5),140$
        0095   31   09F7  2415          BRW     190$                       ; If NOT first time, branch around.
                    09FA  2416 140$:
          04   EF   09FA  2417          EXTZV   #PS4_V_CNTRLTYP,-          ; Extract controller type and
          07        09FC  2418                  #PS4_S_CNTRLTYP,-          ;  put it in on top of stack.
  7E    00B8 C5     09FD  2419                  UCB$W_PORTSTEP4(R5),-(SP)
          50   D4   0A01  2420          CLRL    R0                         ; Clear loop counter.
                    0A03  2421 150$:
FDBA CF40   6E   B1 0A03  2422          CMPW    (SP),CNTRLTYP_ARRAY[R0]    ; Look for UQPORT type.
          0B   13   0A09  2423          BEQL    160$                       ; EQL implies found.
     FDB3 CF40   B5 0A0B  2424          TSTW    CNTRLTYP_ARRAY[R0]         ; See if at end of array.
          30   19   0A10  2425          BLSS    170$                       ; LSS implies NOT found.
          50   D6   0A12  2426          INCL    R0                         ; Increment loop counter.
          ED   11   0A14  2427          BRB     150$                       ; And loop back.
```

```
                                0A16  2428 160$:
          41 A5    FDC2 CF40    90  0A16  2429          MOVB    DEVTYPE_ARRAY[R0],-     ; Copy appropriate UQPORT type to
                                0A1D  2430                  UCB$B_DEVTYPE(R5)       ;  UCB.
               54    0084 C5    D0  0A1D  2431          MOVL    UCB$L_PDT(R5),R4        ; R4 => PDT.
        0128 C4    FDC3 CF40    90  0A22  2432          MOVB    SERVERS_ARRAY[R0],-     ; Copy list of servers supported
                                0A2A  2433                  PDT$B_SERVERS(R4)       ;  to PDT field.
             51    FDC8 CF40    9A  0A2A  2434          MOVZBL  UCODE_VER_ARRAY[R0],R1  ; Minimum acceptable ucode version
                          00     ED  0A30  2435          CMPZV   #PS4_V_UCODEVER,-       ; Compare our ucode version to
                          04         0A32  2436                  #PS4_S_UCODEVER,-       ;  minimum acceptable level.
               51    00B8 C5        0A33  2437                  UCB$W_PORTSTEP4(R5),R1
                          09     18  0A37  2438          BGEQ    170$                    ; GEQ implies it IS acceptable.
             51    FDC6 CF40    9A  0A39  2439          MOVZBL  OPC_MSG_ARRAY[R0],R1    ; Opcom message number to R1.
                        00D1     30  0A3F  2440          BSBW    SEND_OPC_MSG            ; Subroutine to send message.
                                0A42  2441 170$:
                          50 8ED0    0A42  2442          POPL    R0                      ; Controller type to R0.
                                0A45  2443
                       00   50    D1  0A45  2444          CMPL    R0,#UDA50_CNTRLTYP      ; Is it UDA50?
                            05    13  0A48  2445          BEQL    180$                    ; If YES, then go allocate.
                       06   50    D1  0A4A  2446          CMPL    R0,#UDA50A_CNTRLTYP     ; Is it UDA50A?
                            40    12  0A4D  2447          BNEQ    190$                    ; If neither of above, no allocate.
                                0A4F  2448
                                0A4F  2449 180$:
                                0A4F  2450
                                0A4F  2451 ; Older UDA's on VAX-11/780 and VAX-11/790 systems have a purge problem
                                0A4F  2452 ;  that corrupts data.
                                0A4F  2453
                                0A4F  2454          CPUDISP <<780,TESTUDA_780>,-
                                0A4F  2455                   <750,UDA750>,-
                                0A4F  2456                   <730,UDA730>,-
                                0A4F  2457                   <790,TESTUDA_790>,-
                                0A4F  2458                   <UV1,UDAUV1>,-
                                0A4F  2459                   <8SS,UDA8SS>,-
                                0A4F  2460                  >
                                0A69  2461 TESTUDA_780:
                                0A69  2462 TESTUDA_790:
                          00    ED  0A69  2463          CMPZV   #PS4_V_UCODEVER,-       ; If UDA on 780, see if microcode
                          04        0A6B  2464                  #PS4_S_UCODEVER,-       ;  upto rev by comparing against out of
                     00B8 C5        0A6C  2465                  UCB$W_PORTSTEP4(R5),-   ;  date microcode version #.
                          02        0A6F  2466                  #2                      ; Out of date version number.
                          6D    15  0A70  2467          BLEQ    UDA_OUTOFREV            ; LEQ implies inoperative Ucode.
                          1B    11  0A72  2468          BRB     190$                    ; Branch around allocation of perm. BDP.
                                0A74  2469 UDA750:
                                0A74  2470 UDA8SS:
        00000000'GF     16      0A74  2471          JSB     G^IOC$REQDATAP          ; Permanently allocate a datapath.
         50    24 A5    D0      0A7A  2472          MOVL    UCB$L_CRB(R5),R0        ; R0 => CRB.
         54    10 A0    D0      0A7E  2473          MOVL    CRB$L_AUXSTRUC(R0),R4   ; R4 => PDT.
                                0A82  2474
               37 A0    90      0A82  2475          MOVB    CRB$L_INTD+VEC$B_DATAPATH(R0),- ; Remember datapath permanently
            0129 C4             0A85  2476                  PDT$B_DATAPATH(R4)      ;  allocated in PDT.
               37 A0    94      0A88  2477          CLRB    CRB$L_INTD+VEC$B_DATAPATH(R0)   ; Clear CRB field.
                                0A8B  2478
            012A C4    96      0A8B  2479          INCB    PDT$B_BDPUSECNT(R4)     ; Bias useage count to prevent dealloc.
                                0A8F  2480 UDA730:
                                0A8F  2481 UDAUV1:
                                0A8F  2482 190$:
                                0A8F  2483
                                0A8F  2484 ; Here we setup the CRB wakeup mechanism to periodically poll the SA register
```

```
                        0A8F  2485 ; to determine if the U/QPORT has had an uncorrectable error.
                        0A8F  2486
          50   24 A5   D0   0A8F  2487          MOVL    UCB$L_CRB(R5),R0          ; R0 => CRB.
                 0F   C1   0A93  2488          ADDL3   #SA_POLL_INTVAL,-         ; Establish SA polling interval.
00000000'GF              0A95  2489                  G^EXE$GL_ABSTIM,-
               18 A0      0A9A  2490                  CRB$L_DUETIME(R0)
           135B'CF   9E   0A9C  2491          MOVAB   W^PU$SA_POLL,-            ; Establish wakeup routine.
             1C A0        0AA0  2492                  CRB$L_TOUTROUT(R0)
                        0AA2  2493
             5C   01   D0   0AA2  2494          MOVL    S^#SS$_NORMAL,R0          ; Return normal status.
                        0AA5  2495 200$:
             0040 8F   AA   0AA5  2496          BICW    #UCB$M_PU_INILOG,-        ; Indicate Init Error Log no longer
               68 A5        0AA9  2497                  UCB$W_DEVSTS(R5)          ;  in progress (if it was).
             009C D5   17   0AAB  2498          JMP     @UCB$L_DPC(R5)            ;  return to caller.
                        0AAF  2499
                        0AAF  2500 HARDPOWER:                                    ; We got a powerfailure while initing
                        0AAF  2501                                               ;  the UDA.  Simply start hardware init
                        0AAF  2502                                               ;  over.
                        0AAF  2503          ENBINT                               ; Undo DSBINT.
                        0AB2  2504 HARD_RETRY:
          54   0084 C5   D0   0AB2  2505          MOVL    UCB$L_PDT(R5),R4          ; R4 => PDT.
          50   0100 C4   D0   0AB7  2506          MOVL    PDT$L_PU_CSR(R4),R0       ; R0 => UDA CSR.
00AA C5     02 A0   B0   0ABC  2507          MOVW    UDASA(R0),UCB$W_UDASA(R5); Save error status in UCB for logging.
                 02   B0   0AC2  2508          MOVW    #FAIL_ATTNCODE,-          ; Indicate what kind of error log record
             00A8 C5        0AC4  2509                  UCB$W_ATTNCODE(R5)        ;  we are about to create.
                 06   E2   0AC7  2510          BBSS    #UCB$V_PU_INILOG,-        ; Indicate Init Error Log in progress,
             06 68 A5        0AC9  2511                  UCB$W_DEVSTS(R5),210$     ;  and if in progress already, branch.
          00000C00'GF   16   0ACC  2512          JSB     G^ERL$DEVICEATTN          ; Call to create error log record of INIT.
                        0AD2  2513 210$:
             009C C5   DD   0AD2  2514          PUSHL   UCB$L_DPC(R5)             ; Restore caller's return to stack.
               FD38    31   0AD6  2515          BRW     HARDWARE_INIT             ; And branch back to restart hardware
                        0AD9  2516                                               ;  init of UDA.
                        0AD9  2517
                        0AD9  2518 STEP1_TIMEOUT:
                        0AD9  2519 STEP2_TIMEOUT:
                        0AD9  2520 STEP3_TIMEOUT:
                        0AD9  2521          SETIPL  UCB$B_FIPL(R5)            ; Lower IPL after timeout.
                 D3   11   0ADD  2522          BRB     HARD_RETRY                ; Go to try again.
                        0ADF  2523
                        0ADF  2524 UDA_OUTOFREV:
                        0ADF  2525          .IF     DF      UDA50_BYPASS
                        0ADF  2526          BRB     90$                       ; Bypass rejection.
                        0ADF  2527          .ENDC
                        0ADF  2528
                 05   B0   0ADF  2529          MOVW    #UCODE_ATTNCODE,-         ; Indicate what kind of error log record
             00A8 C5        0AE1  2530                  UCB$W_ATTNCODE(R5)        ;  we are about to create.
00000000'GF   16   0AE4  2531          JSB     G^ERL$DEVICEATTN          ; Call to create error log record of INIT.
                        0AEA  2532
          51   57 8F   9A   0AEA  2533          MOVZBL  #MSG$_UDA50MVER,R1        ; Message number to R1 for subroutine.
                 23   10   0AEE  2534          BSBB    SEND_OPC_MSG              ; Send message to OPCOM.
                        0AF0  2535
                 50   D4   0AF0  2536          CLRL    R0                        ; Indicate unable to init hardware.
                 B1   11   0AF2  2537          BRB     200$                      ; Branch back to return.
                        0AF4  2538
                        0AF4  2539          .dsabl  lsb
                        0AF4  2540
                        0AF4  2541 WAIT100US:
```

```
                     0AF4  2542          TIMEDWAIT       TIME=#15        ; Wait for 150 uSecs.
                  05 0B12  2543          RSB
                     0B13  2544
                     0B13  2545 ; R1 = MSG$_ message code.
                     0B13  2546
                     0B13  2547 SEND_OPC_MSG:
                     0B13  2548
                  3F BB   0B13  2549          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers.
               54 51 D0   0B15  2550          MOVL    R1,R4                   ; Message number to R4.
         53 00000000'GF 9E   0B18  2551          MOVAB   G^SYS$GL_OPRMBX,R3      ; R3 => operator's mailbox.
            00000000'GF 16   0B1F  2552          JSB     G^EXE$SNDEVMSG          ; Call to send operator message.
                  3F BA   0B25  2553          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers.
                  05 0B27  2554          RSB                             ; Return to caller.
```

```
                     0B28  2556          .SBTTL  +       INIT_PU_PDT     Fill in variable
                     0B28  2557          .SBTTL  +                       PDT data
                     0B28  2558
                     0B28  2559 ;+
                     0B28  2560 ; INIT_PU_PDT - internal subroutine to initialize (or re-initialize)
                     0B28  2561 ; variable PDT data.  This routine assumes that BUILD_PDT has been called
                     0B28  2562 ; prior to the activation of this routine.
                     0B28  2563 ;
                     0B28  2564 ; INIT_PU_PDT is called from HARDWARE_INIT, and initializes the Q headers
                     0B28  2565 ; in the PDT and also initializes the Ring structures by filling the response
                     0B28  2566 ; ring with available buffers and then placing all the rest of the buffers on
                     0B28  2567 ; the free Q.
                     0B28  2568 ;
                     0B28  2569 ; Inputs:
                     0B28  2570 ;
                     0B28  2571 ;       R4                      -Addr of PDT
                     0B28  2572 ;
                     0B28  2573 ; Outputs:
                     0B28  2574 ;
                     0B28  2575 ;       R0-R2                   -Destroyed
                     0B28  2576 ;       Other registers         -Preserved
                     0B28  2577 ;
                     0B28  2578 ;       Response ring filled
                     0B28  2579 ;       Free Q filled
                     0B28  2580 ;       PDT$L_PU_SNDQFL initialized
                     0B28  2581 ;       PDT$L_PU_BUFQFL initialized
                     0B28  2582 ;-
                     0B28  2583
                     0B28  2584 INIT_PU_PDT:
                     0B28  2585
                     0B28  2586          ASSUME  PDT$W_CMDINT    EQ      PDT$L_COMAREA+4
                     0B28  2587          ASSUME  PDT$W_RSPINT    EQ      PDT$W_CMDINT+2
0200 C4    7C        0B28  2588          CLRQ    PDT$L_COMAREA(R4)
                     0B2C  2589
                     0B2C  2590          ASSUME  PDT$B_CPOLLINX  EQ      PDT$B_CRINGINX+1
                     0B2C  2591          ASSUME  PDT$B_CRINGCNT  EQ      PDT$B_CPOLLINX+1
                     0B2C  2592          ASSUME  PDT$B_RRINGINX  EQ      PDT$B_CRINGCNT+1
                     0B2C  2593          ASSUME  PDT$B_RPOLLINX  EQ      PDT$B_RRINGINX+1
                     0B2C  2594          ASSUME  PDT$B_RRINGCNT  EQ      PDT$B_RPOLLINX+1
                     0B2C  2595          ASSUME  PDT$B_NOCURCON  EQ      PDT$B_RRINGCNT+1
                     0B2C  2596          ASSUME  PDT$B_CONBITMAP EQ      PDT$B_NOCURCON+1
0120 C4    7C        0B2C  2597          CLRQ    PDT$B_CRINGINX(R4)
                     0B30  2598
     14    D0        0B30  2599          MOVL    #UDAB$T_TEXT,-          ; Save UDA port SCS datagram header
00B8 C4              0B32  2600                  PDT$L_DGOVRHD(R4)       ;  size in PDT.
     14    D0        0B35  2601          MOVL    #UDAB$T_TEXT,-          ; Save UDA port SCS message header
00B4 C4              0B37  2602                  PDT$L_MSGHDRSZ(R4)      ;  size in PDT.
                     0B3A  2603
                     0B3A  2604 ; Initialize PDT Q headers.
                     0B3A  2605
00AC C4    9E        0B3A  2606          MOVAB   PDT$L_WAITQFL(R4),-     ; Empty wait queue header
00AC C4              0B3E  2607                  PDT$L_WAITQFL(R4)       ;
00AC C4    9E        0B41  2608          MOVAB   PDT$L_WAITQFL(R4),-     ; Empty wait queue header
00B0 C4              0B45  2609                  PDT$L_WAITQBL(R4)       ;
                     0B48  2610
0108 C4    9E        0B48  2611          MOVAB   PDT$L_PU_FQFL(R4),-     ; Free Q header.
0108 C4              0B4C  2612                  PDT$L_PU_FQFL(R4)
```

```
                                ;+  PDT data

                0108 C4   9E  0B4F  2613          MOVAB   PDT$L_PU_FQFL(R4),-      ; Free Q header.
                010C C4       0B53  2614                  PDT$L_PU_FQBL(R4)
                0108 C4   9E  0B56  2615          MOVAB   PDT$L_PU_FQFL(R4),-      ; Point Free Q pointer to Free Q. This
                0104 C4       0B5A  2616                  PDT$L_PU_FQPTR(R4)       ;  pointer is used to test for
                              0B5D  2617                                           ;  emptiness of Free Q.
                              0B5D  2618
                0110 C4   9E  0B5D  2619          MOVAB   PDT$L_PU_SNDQFL(R4),-    ; Send Q header.
                0110 C4       0B61  2620                  PDT$L_PU_SNDQFL(R4)
                0110 C4   9E  0B64  2621          MOVAB   PDT$L_PU_SNDQFL(R4),-    ; Send Q header.
                0114 C4       0B68  2622                  PDT$L_PU_SNDQBL(R4)
                              0B6B  2623
                0118 C4   9E  0B6B  2624          MOVAB   PDT$L_PU_BUFQFL(R4),-    ; Buffer wait Q header.
                0118 C4       0B6F  2625                  PDT$L_PU_BUFQFL(R4)
                0118 C4   9E  0B72  2626          MOVAB   PDT$L_PU_BUFQFL(R4),-    ; Buffer wait Q header.
                011C C4       0B76  2627                  PDT$L_PU_BUFQBL(R4)
                              0B79  2628
                              0B79  2629 ; Initialize initial credits for all possible connections.
                              0B79  2630
          00F4 C4   01   B0   0B79  2631          MOVW    #1,PDT$W_PU_CRED0(R4)    ; Initial credit of 1.
          00F6 C4   01   B0   0B7E  2632          MOVW    #1,PDT$W_PU_CRED1(R4)    ; Initial credit of 1.
          00F8 C4   01   B0   0B83  2633          MOVW    #1,PDT$W_PU_CRED2(R4)    ; Initial credit of 1.
          00FA C4   01   B0   0B88  2634          MOVW    #1,PDT$W_PU_CRD255(R4)   ; Initial credit of 1.
                              0B8D  2635
                              0B8D  2636 ; Following loop is for arrays that are RINGSIZE long.
                              0B8D  2637
                50    0F    D0  0B8D  2638          MOVL    #UDA$K_RINGSIZE-1,R0     ; Initialize loop counter.
                              0B90  2639 10$:
      012C C440   01    8E  0B90  2640          MNEGB   #1,PDT$B_CRCONTENT(R4)[R0]       ; Nothing in command slot.
      013C C440   01    8E  0B96  2641          MNEGB   #1,PDT$B_RRCONTENT(R4)[R0]       ; Nothing in response slot.
           0248 C440    D4  0B9C  2642          CLRL    PDT$L_CMDRING(R4)[R0]            ; Clear command ring slot.
           0208 C440    D4  0BA1  2643          CLRL    PDT$L_RSPRING(R4)[R0]            ; Clear response ring slot.
                E7 50   F4  0BA6  2644          SOBGEQ  R0,10$                           ; Loop thru all array elements.
                        05  0BA9  2645          RSB                                  ; Return to caller. Rest of PDT init
                              0BAA  2646                                           ;  called explicitly at label below.
                              0BAA  2647
                              0BAA  2648 STOCK_RSPRING:
                              0BAA  2649
                              0BAA  2650 ; Loop thru all buffers (there are 2*RINGSIZE of them) and put them on the
                              0BAA  2651 ;       response ring or the free Q.
                              0BAA  2652
                50    D4    0BAA  2653          CLRL    R0                       ; Initialize loop variable.
                              0BAC  2654 10$:
     52   014C C440   D0    0BAC  2655          MOVL    PDT$L_BDTABLE(R4)[R0],R2; R2 => buffer[R0]
                50    DD    0BB2  2656          PUSHL   R0                       ; Remember loop variable before call.
                02E6  30    0BB4  2657          BSBW    Q_DEALLOC_BUF            ; Put buffer on response ring or free
                              0BB7  2658                                           ;  Q, whichever is appropriate.
                50 8ED0     0BB7  2659          POPL    R0                       ; Restore loop variable.
           EE 50    20  F2  0BBA  2660          AOBLSS  #2*UDA$K_RINGSIZE,R0,10$; Loop thru all buffers.
                              0BBE  2661
                        05  0BBE  2662          RSB                              ; And return to caller.
```

```
                        .SBTTL  +       BUILD_PB_SB Build System Block and Path Block
                        ;+
                        ; BUILD_PB_SB - Build and fill in the System Block and the Path Block.
                        ;
                        ; This portion of the UDA port driver is responsible for adding
                        ; the UDA to the system-wide configuration database.  It is invoked
                        ; as a one time initialization routine.
                        ;
                        ; The system wide configuration database consists of:
                        ;
                        ;       SCS$GQ_CONFIG
                        ;            |
                        ;            v
                        ;       System Block ----> Path Block ----> Path Block ---->...
                        ;            |
                        ;            v
                        ;       System Block ----> Path Block ---->...
                        ;            |
                        ;            v
                        ;           ...
                        ;
                        ; Only systems and paths with open port-port VC's are kept on the
                        ; above list.
                        ;
                        ; For each UDA (AZTEC or TU81) we build a Path Block and a System Block,
                        ; initialize them and link them into the systemwide configuration database.
                        ;
                        ; Inputs:
                        ;
                        ;       R4                              -Addr of PDT
                        ;       R5                              -Addr of UCB
                        ;
                        ;-
                        BUILD_PB_SB:

51  00000054 8F   D0    0BBF 2702           MOVL    #PB$K_LENGTH,R1                 ; Get size of a pathblock
         00C3   30    0BC6 2703           BSBW    ALLOC_POOL                      ; Allocate one from pool
       03 50    E8    0BC9 2704           BLBS    R0,10$                          ; Branch around if success.
        0099    31    0BCC 2705           BRW     PB_ALLOC_FAIL                   ; Branch if no pool
                      0BCF 2706 10$:
    53    52    D0    0BCF 2707           MOVL    R2,R3                           ; Set PB addr in stable register
    62    52    D0    0BD2 2708           MOVL    R2,PB$L_FLINK(R2)               ; This will be the only Path
                      0BD5 2709                                                   ;  Block on this list.
 04 A2    52    D0    0BD5 2710           MOVL    R2,PB$L_BLINK(R2)               ; FLINK and BLINK point here.
                      0BD9 2711
 08 A2    51    B0    0BD9 2712           MOVW    R1,PB$W_SIZE(R2)                ; Set structure size
                      0BDD 2713
                      0BDD 2714           ASSUME  PB$B_SUBTYP     EQ      PB$B_TYPE+1
    0460 8F    B0    0BDD 2715           MOVW    #DYN$C_SCS+<DYN$C_SCS_PB@8>,-
      0A A2          0BE1 2716                   PB$B_TYPE(R2)                   ; Set struct type, subtype
                      0BE3 2717
 12 A2    03    B0    0BE3 2718           MOVW    #PB$C_OPEN,PB$W_STATE(R2)       ; Always in OPEN state.
                      0BE7 2719
 51   28 A5    D0    0BE7 2720           MOVL    UCB$L_DDB(R5),R1                ; R1 => DDB for port device.
```

```
                15 A1    D0  0BEB  2721          MOVL    DDB$T_NAME+1(R1),-              ; Copy port name to port block.
                24 A2        0BEE  2722                  PB$T_LPORT_NAME(R2)
            27 A2    30  90  0BF0  2723          MOVB    #^A/0/,PB$T_LPORT_NAME+3(R2)    ; Unit is always 0.
                         0BF4  2724
            2C A2    54  D0  0BF4  2725          MOVL    R4,PB$L_PDT(R2)                 ; Save PDT pointer in Path Block.
                         0BF8  2726
                38 A2    9E  0BF8  2727          MOVAB   PB$L_WAITQFL(R2),-              ; Initialize empty Q header.
                38 A2        0BFB  2728                  PB$L_WAITQFL(R2)
                38 A2    9E  0BFD  2729          MOVAB   PB$L_WAITQFL(R2),-
                3C A2        0C00  2730                  PB$L_WAITQBL(R2)
                         0C02  2731
                         0C02  2732 ; Now build System Block.
                         0C02  2733
        51   00000060 8F    D0  0C02  2734          MOVL    #SB$K_LENGTH,R1         ; Get size of SB
                 0080    30  0C09  2735          BSBW    ALLOC_POOL              ; Allocate from nonpaged pool
                59 50    E9  0C0C  2736          BLBC    R0,SB_ALLOC_FAIL        ; Branch if no pool
            30 A3    52  D0  0C0F  2737          MOVL    R2,PB$L_SBLINK(R3)      ; Save System Block address in PB.
         00FC C4     52  D0  0C13  2738          MOVL    R2,PDT$L_PU_SB(R4)      ; Save System Block address in PDT.
                         0C18  2739
            08 A2    51  B0  0C18  2740          MOVW    R1,SB$W_SIZE(R2)        ; Set struct size
                0760 8F  B0  0C1C  2741          MOVW    #DYN$C_SCS+<DYN$C_SCS_SB@8>,-   ; Set structure type
                  0A A2      0C20  2742                  SB$B_TYPE(R2)           ;  and subtype
          0C A2    0C A2  9E  0C22  2743          MOVAB   SB$L_PBFL(R2),SB$L_PBFL(R2)     ; Establish Q head of path
          10 A2    0C A2  9E  0C27  2744          MOVAB   SB$L_PBFL(R2),SB$L_PBBL(R2)     ;  blocks. And Q this path
             0C B2    63  0E  0C2C  2745          INSQUE  (R3),@SB$L_PBFL(R2)             ;  block to head of that Q.
             14 A2    53  D0  0C30  2746          MOVL    R3,SB$L_PBCONNX(R2)             ; Let this path be Next path.
                         0C34  2747
                         0C34  2748 ; Here we concoct the SYSTEMID of this controller.  It is made up of the
                         0C34  2749 ;       following pieces:
                         0C34  2750 ;
                         0C34  2751 ;        4               3 3
                         0C34  2752 ;       .7______________2.1_____________________________0.
                         0C34  2753 ;       |               |                               |
                         0C34  2754 ;       |1|  TR number  |       UDA CSR address         |
                         0C34  2755 ;       |_|_____________|_______________________________|
                         0C34  2756
            50   24 A5    D0  0C34  2757          MOVL    UCB$L_CRB(R5),R0                ; R0 => CRB.
            50   2C A0    D0  0C38  2758          MOVL    CRB$L_INTD+VEC$L_IDB(R0),R0     ; R0 => IDB.
                    60    D0  0C3C  2759          MOVL    IDB$L_CSR(R0),-                 ; Move CSR to low longword of
                18 A2        0C3E  2760                  SB$B_SYSTEMID(R2)               ;  SYSTEMID.
            50   14 A0    D0  0C40  2761          MOVL    IDB$L_ADP(R0),R0                ; R0 => ADP.
                 0C A0    B0  0C44  2762          MOVW    ADP$W_TR(R0),-                  ; Move nexus number to high
                 1C A2        0C47  2763                  SB$B_SYSTEMID+4(R2)             ;  word of SYSTEMID.
         1D A2   80 8F    88  0C49  2764          BISB    #^X80,SB$B_SYSTEMID+5(R2)       ; Set high bit on in SYSTEMID.
                         0C4E  2765
          00000000'GF    7D  0C4E  2766          MOVQ    G^EXE$GQ_SYSTIME,-      ; Copy current time to boot time
                 2C A2        0C54  2767                  SB$Q_SWINCARN(R2)       ;  of this port.
                 5C A2    D4  0C56  2768          CLRL    SB$L_CSB(R2)            ; Clear link to newest CSB.
                         0C59  2769
                         0C59  2770 ; Here link System Block onto system list.
                         0C59  2771
    50   00000000'GF    DE  0C59  2772          MOVAL   G^SCS$GQ_CONFIG,R0      ; R0 => Systemwide list head of system
                         0C60  2773                                              ;  blocks.
             04 B0    62  0E  0C60  2774          INSQUE  (R2),@4(R0)             ; Queue to tail of list.
                50    01  D0  0C64  2775          MOVL    S^#SS$_NORMAL,R0        ; Indicate success and
                          05  0C67  2776          RSB                             ;  return to caller.
                         0C68  2777
```

```
                0C68  2778 PB_ALLOC_FAIL:
                0C68  2779 SB_ALLOC_FAIL:
50   D4         0C68  2780          CLRL    R0                  ; Indicate failure and
     05         0C6A  2781          RSB                         ;  return to caller.
```

```
                     0C6B  2783          .SBTTL  +               UPDATE_PB_SB
                     0C6B  2784
                     0C6B  2785 ;+
                     0C6B  2786 ; UPDATE_PB_SB - routine to update fields in the PB (and potentially the SB)
                     0C6B  2787 ;       after the data has been obtained.
                     0C6B  2788 ;
                     0C6B  2789 ; Inputs:
                     0C6B  2790 ;       R5 => UCB
                     0C6B  2791
                     0C6B  2792 UPDATE_PB_SB:
                     0C6B  2793
50   0084 C5    D0   0C6B  2794          MOVL    UCB$L_PDT(R5),R0        ; R0 => PDT.
50   00FC C0    D0   0C70  2795          MOVL    PDT$L_PU_SB(R0),R0      ; R0 => SB.
           14   13   0C75  2796          BEQL    10$                     ; EQL means we failed to alloc SB.
  50    14 A0   D0   0C77  2797          MOVL    SB$L_PBCONNX(R0),R0     ; R0 => PB.
                     0C7B  2798
           08   EF   0C7B  2799          EXTZV   #PS2_V_PORTTYPE,-       ; Extract Port Type returned by hardware
           03         0C7D  2800                  #PS2_S_PORTTYPE,-       ;  initialization of controller.
      00B0 C5         0C7E  2801                  UCB$Q_PORTSTEP2(R5),-
        14 A0         0C81  2802                  PB$L_RPORT_TYP(R0)
                     0C83  2803
           00   EF   0C83  2804          EXTZV   #PS4_V_UCODEVER,-       ; Extract ucode version returned by
                     0C85  2805                  #PS4_S_UCODEVER+-       ;  hardware init of controller.
                     0C85  2806                   PS4_S_CNTRLTYP,-
00B8 C5    0B         0C85  2807                  UCB$Q_PORTSTEP4(R5),-
        18 A0         0C89  2808                  PB$L_RPORT_REV(R0)
                     0C8B  2809 10$:
                05   0C8B  2810          RSB                             ; Return to caller.
```

```
                                0C8C  2812          .SBTTL  +       ALLOC_POOL
                                0C8C  2813
                                0C8C  2814 ;+
                                0C8C  2815 ; This subroutine allocates and zeroes nonpaged pool.
                                0C8C  2816 ;       It is assumed here that the call here is always made from a called
                                0C8C  2817 ;       internal subroutine, implying that two return points must be popped
                                0C8C  2818 ;       off the stack if the thread is suspended.
                                0C8C  2819 ;
                                0C8C  2820 ; Inputs:
                                0C8C  2821 ;
                                0C8C  2822 ;       R1                      -# bytes of pool to allocate
                                0C8C  2823 ;       R5                      -Addr of UCB
                                0C8C  2824 ;
                                0C8C  2825 ; Outputs:
                                0C8C  2826 ;
                                0C8C  2827 ;       R0                      -0/1 for fail/success
                                0C8C  2828 ;       R1                      -# bytes actually allocated
                                0C8C  2829 ;       R2                      -Addr of buffer allocated
                                0C8C  2830 ;-
                                0C8C  2831
                                0C8C  2832 ALLOC_POOL:                              ; Allocate and zero pool
                                0C8C  2833
          009C C5 8ED0          0C8C  2834          POPL    UCB$L_DPC(R5)           ; Remember return
                53   DD         0C91  2835          PUSHL   R3                      ; Save R3.
    00000000'GF    16           0C93  2836          JSB     G^EXE$ALONONPAGED       ; Allocate from nonpaged pool
              11   50  E9        0C99  2837          BLBC    R0,10$                  ; Skip clearing structure if failure
                37   BB         0C9C  2838          PUSHR   #^M<R0,R1,R2,R4,R5>     ; Save MOVC registers
    62  51  00  6E   00   2C     0C9E  2839          MOVC5   #0,(SP),#0,R1,(R2)      ; Zero initialize structure
                37   BA         0CA4  2840          POPR    #^M<R0,R1,R2,R4,R5>     ; Restore MOVC registers
                53 8ED0          0CA6  2841          POPL    R3                      ; Restore R3.
          009C D5   17           0CA9  2842          JMP     @UCB$L_DPC(R5)          ; Return to caller.
                                0CAD  2843 10$:                                     ; Here we suffered an allocation failure.
                                0CAD  2844                                          ;  Prepare to wait awhile before trying
                                0CAD  2845                                          ;  again.
    00A0 C5   51   D0           0CAD  2846          MOVL    R1,UCB$L_PU_ALLOC(R5)   ; Save size of block to allocate.
          10 A5 8ED0            0CB2  2847          POPL    UCB$L_FR3(R5)           ; Pop given R3 into save area.
       14 A5   54   D0          0CB6  2848          MOVL    R4,UCB$L_FR4(R5)        ; Save R4.
          0C A5 8ED0            0CBA  2849          POPL    UCB$L_FPC(R5)           ; Save caller's caller's return.
                                0CBE  2850
       53   24 A5   D0          0CBE  2851          MOVL    UCB$L_CRB(R5),R3        ; R3 => CRB.
    1C A3   D1'AF   9E          0CC2  2852          MOVAB   B^20$,CRB$L_TOUTROUT(R3); Establish wakeup routine.
                01   C1         0CC7  2853          ADDL3   #ALLOC_DELTA,-          ; Establish a small delay.
     00000000'GF               0CC9  2854                  G^EXE$GL_ABSTIM,-
             18 A3              0CCE  2855                  CRB$L_DUETIME(R3)
                05   05         0CD0  2856          RSB                             ; Evaporate for awhile.
                                0CD1  2857 20$:
                                0CD1  2858          SETIPL  #IPL$_SCS               ; Lower IPL after wakeup.
       55   10 A3   D0          0CD4  2859          MOVL    CRB$L_AUXSTRUC(R3),R5   ; R5 => UCB or PDT.
                10   91         0CD8  2860          CMPB    #DYN$C_UCB,-            ; Determine which, UCB or PDT.
             0A A5              0CDA  2861                  UCB$B_TYPE(R5)
                05   13         0CDC  2862          BEQL    30$                     ; EQL implies UCB.
       55   00DC C5   D0        0CDE  2863          MOVL    PDT$L_UCB0(R5),R5       ; If PDT, go one level deeper so that
                                0CE3  2864 30$:                                     ;  here R5 => UCB.
          F412 CF   9E          0CE3  2865          MOVAB   NULL_ROUTINE,-          ; Restablish null wakeup routine.
             1C A3              0CE7  2866                  CRB$L_TOUTROUT(R3)
       51   00A0 C5   D0        0CE9  2867          MOVL    UCB$L_PU_ALLOC(R5),R1   ; Restore R1 = size of block to alloc.
         53   10 A5   7D        0CEE  2868          MOVQ    UCB$L_FR3(R5),R3        ; Restore R3 and R4.
```

```
                        + ALLOC_POOL

                OC A5   DD  0CF2  2869        PUSHL   UCB$L_FPC(R5)           ; Re-establish caller's caller's return poin
            009C C5     DD  0CF5  2870        PUSHL   UCB$L_DPC(R5)           ; Re-establish caller's return point.
                91      11  0CF9  2871        BRB     ALLOC_POOL              ; Go try again.
```

```
                        .SBTTL  UNIMPLEMENTED FORK PROCESS CALLS
FPC$ACCEPT::
FPC$ALLOCDG::
FPC$DEALLOCDG::
FPC$MAP::
FPC$MAPBYPASS::
FPC$MAPIRPBYP::
FPC$QUEUEMDGS::
FPC$REJECT::
FPC$REQDATA::
FPC$SENDDATA::
FPC$SENDDG::
FPC$SENDMSG::

FPC$READCOUNT::
FPC$RLSCOUNT::
FPC$MAINTFCN::
FPC$SENDRGDG::
FPC$STOP_VCS::
                BUG_CHECK       UDAPORT,FATAL
```

```
                     0CFF  2894          .SBTTL  MRESET and MSTART
                     0CFF  2895
                     0CFF  2896 ;+
                     0CFF  2897 ; FPC$MRESET - causes connection ERROR ROUTINE to be called for all open
                     0CFF  2898 ;       connections and then does a HARDWARE INIT of the port.
                     0CFF  2899 ; FPC$MSTART is a NOP.
                     0CFF  2900 ;
                     0CFF  2901 ; Inputs:
                     0CFF  2902 ;       R4 => PDT
                     0CFF  2903 ;
                     0CFF  2904 ; Outputs:
                     0CFF  2905 ;       All registers preserved, however caller is returned to before the
                     0CFF  2906 ;             thread started by FPC$MRESET finishes.
                     0CFF  2907 ;-
                     0CFF  2908
                     0CFF  2909 FPC$MRESET::
                     0CFF  2910
           3F   BB   0CFF  2911          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save caller's registers.
55 00DC C4   D0   0D01  2912          MOVL    PDT$L_UCB0(R4),R5       ; R5 => UCB for port.
     F5BA   30   0D06  2913          BSBW    POST_POWER_FORK         ; Start thread to call error routines
                0D09  2914                                          ;  and to do hardware init.
           3F   BA   0D09  2915          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore caller's registers.
                0D0B  2916 FPC$MSTART::
                05   0D0B  2917          RSB                             ; Return to caller.
```

```
0D0C  2919          .SBTTL  CONNECTION MANAGEMENT CALLS
0D0C  2920          .SBTTL  +       FPC$CONNECT,    COMPLETE PROCESSING A CONNECT
0D0C  2921
0D0C  2922  ;+
0D0C  2923  ; This routine is JMP'ed to from SCS$CONNECT with a CDT allocated
0D0C  2924  ; (and in the closed state) and initialized with the SYSAP's
0D0C  2925  ; connect parameters or 0's for fields not yet used.  FPC$CONNECT
0D0C  2926  ; does port-specific processing.
0D0C  2927  ;
0D0C  2928  ; The UDA port driver only supports one connection at a time.  The first
0D0C  2929  ; thing that FPC$CONNECT checks for is that there are no current connections
0D0C  2930  ; on this port. Then if there are none we check that the target of the
0D0C  2931  ; connection is one of the ones supported.  If so, then the initial credits
0D0C  2932  ; allotted to this connection are granted, the connection ID is saved, the
0D0C  2933  ; CDT address is stored and the CDT state is set to open.
0D0C  2934  ;
0D0C  2935  ; Inputs:
0D0C  2936  ;
0D0C  2937  ;       R3                      -Addr of CDT
0D0C  2938  ;       R4                      -Addr of PDT
0D0C  2939  ;
0D0C  2940  ;       CDT initialized as follows:
0D0C  2941  ;
0D0C  2942  ;       CDT$L_LCONID            -Local conid
0D0C  2943  ;             MSGINPUT          -Addr to call in SYSAP for rec'd msgs
0D0C  2944  ;             DGINPUT           -Addr to call in SYSAP for rec'd dgs
0D0C  2945  ;             ERRADDR           -Addr to call in SYSAP for connection errors
0D0C  2946  ;             RSTATION          -Remote station addr
0D0C  2947  ;             PDT               -Addr of PDT
0D0C  2948  ;             MINSEND           -Minimum send credit req'd by SYSAP
0D0C  2949  ;             INITLREC          -Initial credit extended by SYSAP
0D0C  2950  ;             DGREC             -Initial # of dg's queued
0D0C  2951  ;             STATE             -CLOSED
0D0C  2952  ;             PB                -Addr of selected PB to remote system
0D0C  2953  ;             WAITQFL/BL        -Set to show no entries
0D0C  2954  ;             RPROCNAM          -Addr of dest process name
0D0C  2955  ;             LPROCNAM          -Addr of local process name
0D0C  2956  ;             CONDAT            -Addr of connect data
0D0C  2957  ;
0D0C  2958  ;       other CDT fields        -0
0D0C  2959  ;
0D0C  2960  ;
0D0C  2961  ;       (SP)                    -Return PC in SYSAP
0D0C  2962  ;
0D0C  2963  ; Outputs:
0D0C  2964  ;
0D0C  2965  ;       R0                      -Status:  SS$_NORMAL, SS$_FAILRSP,
0D0C  2966  ;                                         SS$_REJECT, SS$_INSFMEM
0D0C  2967  ;       R1                      -Reject reason or fail response reason
0D0C  2968  ;                                if R0 = REJECT or FAILRSP
0D0C  2969  ;       R2                      -Addr of ACCEPT_REQ if R0 = success
0D0C  2970  ;       other registers         -Preserved
0D0C  2971  ;
0D0C  2972  ;       CDT$L_RCONID - UDA virtual circuit number
0D0C  2973  ;-
0D0C  2974
0D0C  2975          .ENABL  LSB
```

```
                4B 53 49 44 24 50 43 53 4D  0D0C  2976 VCONAM: .ASCII  /MSCP$DISK/                ; Name for UDA VC 0
                                00000009    0D15  2977 VCONAMLEN = .-VCONAM                     ;  name length
                45 50 41 54 24 50 43 53 4D  0D15  2978 VC1NAM: .ASCII  /MSCP$TAPE/             ; Name for UDA VC 1
                                00000009    0D1E  2979 VC1NAMLEN = .-VC1NAM                    ;  name length
                                50 55 44    0D1E  2980 VC2NAM: .ASCII  /DUP/                  ; Name for UDA VC 2
                                00000003    0D21  2981 VC2NAMLEN = .-VC2NAM                   ;  name length
                                            0D21  2982
                                            0D21  2983 FPC$CONNECT::
                                            0D21  2984
                      50   00DC C4   D0    0D21  2985         MOVL    PDT$L_UCB0(R4),R0        ; R0 => UCB.
                                04   E1    0D26  2986         BBC     #UCB$V_ONLINE,-          ; Reject connect if offline.
                           38 64 A0         0D28  2987                 UCB$W_STS(R0),CON_NO_NODE
                                            0D2B  2988
                                            0D2B  2989         $CHK_CDTSTATE -                  ; Verify that CDT state
                                            0D2B  2990                 CLOSED,-                 ;  is closed;  if not,
                                            0D2B  2991                 ERROR=STATE_ERR          ;  caller made error
                                            0D34  2992 ;
                                            0D34  2993 ; If the desired process name is MSCP$DISK, MSCP$TAPE or DUP, then the
                                            0D34  2994 ; connection can be completed.
                                            0D34  2995 ;
                                53   DD     0D34  2996         PUSHL   R3                       ; Remember R3=>CDT.
                                01   E0     0D36  2997         BBS     #UDA$V_STOPPED,-         ; Only allow Connections to DUP if
                           00A4 C0          0D38  2998                 UCB$B_UDAFLAGS(R0),-     ;  Port is closed.
                                1A          0D3B  2999                 TSTVC2
                      20   CC AF   09   2D   0D3C  3000         CMPC5   #VCONAMLEN,VCONAM,#^A/ /,-
                           50 B3   10        0D41  3001                 #16,@CDT$L_RPROCNAM(R3) ; Is the connection to MSCP$DISK?
                                5F   13      0D44  3002         BEQL    CONVC0                   ; If yes make connection to VC0
                                            0D46  3003
                                53   6E     0D46  3004         MOVL    (SP),R3                  ; Refresh R3 => CDT.
                      20   C8 AF   09   2D   0D49  3005         CMPC5   #VC1NAMLEN,VC1NAM,#^A/ /,-
                           50 B3   10        0D4E  3006                 #16,@CDT$L_RPROCNAM(R3) ; Is the connection to MSCP$TAPE?
                                43   13      0D51  3007         BEQL    CONVC1                   ; If yes make connection to VC1
                                            0D53  3008
                                53   6E     0D53  3009         MOVL    (SP),R3                  ; Refresh R3 => CDT.
                                            0D56  3010 TSTVC2:
                      20   C4 AF   03   2D   0D56  3011         CMPC5   #VC2NAMLEN,VC2NAM,#^A/ /,-
                           50 B3   10        0D5B  3012                 #16,@CDT$L_RPROCNAM(R3) ; Is the connection to DUP?
                                27   13      0D5E  3013         BEQL    CONVC2                   ; If yes make connection to VC2
                                53 8ED0     0D60  3014         POPL    R3                       ; Remove R3 from stack.
                                            0D63  3015 CON_NO_NODE:
                 00000000'GF    16          0D63  3016         JSB     G^SCS$DEALL_CDT          ; Free R3 => CDT.
                      50   028C 8F   3C     0D69  3017         MOVZWL  #SS$_NOSUCHNODE,R0       ; Indicate No Such Node.
                                05          0D6E  3018         RSB                              ;
                                            0D6F  3019 CONREJ:
                 00000000'GF    16          0D6F  3020         JSB     G^SCS$DEALL_CDT          ; Free R3 => CDT.
                      50   0294 8F   3C     0D75  3021         MOVZWL  #SS$_REJECT,R0           ; Otherwise reject connection
                                05          0D7A  3022         RSB                              ;
                                            0D7B  3023 CON_NO_LISTEN:
                 00000000'GF    16          0D7B  3024         JSB     G^SCS$DEALL_CDT          ; Free R3 => CDT.
                      50   215C 8F   3C     0D81  3025         MOVZWL  #SS$_NOLISTENER,R0       ; Indicate No Such Listener.
                                05          0D86  3026         RSB                              ;
                                            0D87  3027 ;
                                            0D87  3028 ; Make a connection to MSCP$DISK which is virtual circuit 0 for a
                                            0D87  3029 ; UDA.
                                            0D87  3030 ;
                                            0D87  3031 CONVC2:
                      50   00EC C4   9E     0D87  3032         MOVAB   PDT$L_PU_VC2(R4),R0      ; Setup for common code.
```

```
                51   00F8 C4     9E   0D8C  3033          MOVAB   PDT$W_PU_CRED2(R4),R1    ; Initial credits to assign.
                52        02     D0   0D91  3034          MOVL    #2,R2                   ; Remote connection ID.
                          1B     11   0D94  3035          BRB     CON_COMMON              ; Branch around.
                                      0D96  3036
                                      0D96  3037 CONVC1:
                50   00E8 C4     9E   0D96  3038          MOVAB   PDT$L_PU_VC1(R4),R0     ; Setup for common code.
                51   00F6 C4     9E   0D9B  3039          MOVAB   PDT$W_PU_CRED1(R4),R1   ; Initial credits to assign.
                52        01     D0   0DA0  3040          MOVL    #1,R2                   ; Remote connection ID.
                          0C     11   0DA3  3041          BRB     CON_COMMON              ; Branch around.
                                      0DA5  3042
                                      0DA5  3043 CONVC0:
                50   00E4 C4     9E   0DA5  3044          MOVAB   PDT$L_PU_VC0(R4),R0     ; Setup for common code.
                51   00F4 C4     9E   0DAA  3045          MOVAB   PDT$W_PU_CRED0(R4),R1   ; Initial credits to assign.
                          52     D4   0DAF  3046          CLRL    R2                      ; Remote connection ID is zero.
                                      0DB1  3047 CON_COMMON:
                          53   8ED0   0DB1  3048          POPL    R3                      ; Refresh R3 => CDT.
          0128 C4       52     E1   0DB4  3049          BBC     R2,PDT$B_SERVERS(R4),-  ; See if this Server supported at this
                          C1         0DB9  3050                  CON_NO_LISTEN           ;  port. If not, branch.
          0127 C4       52     E2   0DBA  3051          BBSS    R2,PDT$B_CONBITMAP(R4),-; Mark corresponding bit in bit map
                          AF         0DBF  3052                  CONREJ                  ;  as OPEN.  If already set, reject.
                60        53     D0   0DC0  3053          MOVL    R3,(R0)                 ; Save address of CDT for VCx
                  14 A3     52     D0   0DC3  3054          MOVL    R2,CDT$L_RCONID(R3)     ; Set virtual circuit number into CDT
                     0126 C4    96   0DC7  3055          INCB    PDT$B_NOCURCON(R4)      ; Increment # of current connections.
                                      0DCB  3056
                40 A3     61     B0   0DCB  3057          MOVW    (R1),CDT$W_SEND(R3)     ; Put in initial send credits.
                                      0DCF  3058
                          02     B0   0DCF  3059          MOVW    #CDT$C_OPEN,-           ; Move CDT state to
                     28 A3           0DD1  3060                  CDT$W_STATE(R3)         ;  open
                                      0DD3  3061
                                      0DD3  3062 ; Here we make room on the stack to create a Connect Message to return
                                      0DD3  3063 ; to the caller.  In order to get control after the caller is finished
                                      0DD3  3064 ; with the Connect data, we call him back as a co-routine.
                                      0DD3  3065
                          51   8ED0   0DD3  3066          POPL    R1                      ; R1 has callers return point.
                5E     D0 AE     9E   0DD6  3067          MOVAB   -SCSCMG$S_SCSCMGDEF(SP),SP ; Create space on stack.
                     52   5E     D0   0DDA  3068          MOVL    SP,R2                   ; R2=>Connect data area.
                                      0DDD  3069
                          3E     BB   0DDD  3070          PUSHR   #^M<R1,R2,R3,R4,R5>     ; Save MOVC registers
          00        6E     00     2C   0DDF  3071          MOVC5   #0,(SP),#0,-            ; Zero initialize structure
                     62    30         0DE3  3072                  #SCSCMG$S_SCSCMGDEF,(R2)
                          3E     BA   0DE5  3073          POPR    #^M<R1,R2,R3,R4,R5>     ; Restore MOVC registers
                                      0DE7  3074
                                      0DE7  3075 ;        MOVB    SYSGEN_PARAMETER,-
                                      0DE7  3076 ;                SCSCMG$B_SNDATA+1(R2)   ; Copy Allocation Class.
                                      0DE7  3077
                     50   01     3C   0DE7  3078          MOVZWL  S^#SS$_NORMAL,R0        ; Set normal completion
                          61     16   0DEA  3079          JSB     (R1)                    ; Call back Caller as co-routine.
                5E     30 AE     9E   0DEC  3080          MOVAB   SCSCMG$S_SCSCMGDEF(SP),SP ; Free up stack space.
                          05     05   0DF0  3081          RSB                             ; and return
                                      0DF1  3082
                                      0DF1  3083          .DSABL  LSB
                                      0DF1  3084
```

```
                     0DF1  3086          .SBTTL  +       FPC$DCONNECT, PROCESS A DISCONNECT CALL
                     0DF1  3087
                     0DF1  3088 ;+
                     0DF1  3089 ; FPC$DCONNECT is called by the SYSAP.  It may be called only from the
                     0DF1  3090 ; open state.  The CDT is moved to the closed state.
                     0DF1  3091 ;
                     0DF1  3092 ; Inputs:
                     0DF1  3093 ;
                     0DF1  3094 ;       R0                      -Disconnect reason
                     0DF1  3095 ;       R3                      -Addr of CDT being disconnected
                     0DF1  3096 ;       R4                      -Addr of PDT
                     0DF1  3097 ;
                     0DF1  3098 ; Outputs:
                     0DF1  3099 ;
                     0DF1  3100 ;       R0                      -Status:  SS$_NORMAL, SS$_ILLCDTST
                     0DF1  3101 ;       R1,R2                   -Destroyed
                     0DF1  3102 ;       other registers         -Preserved
                     0DF1  3103 ;-
                     0DF1  3104
                     0DF1  3105
                     0DF1  3106 FPC$DCONNECT::
                     0DF1  3107
          26 A3  50  B0  0DF1  3108          MOVW    R0,CDT$W_REASON(R3)     ; Save disconnect reason
                     0DF5  3109          $CHK_CDTSTATE   -                   ; Assure that CONNECTION is open.
                     0DF5  3110                  OPEN,-
                     0DF5  3111                  ERROR=STATE_ERR
                     0DFE  3112
                00   B0  0DFE  3113          MOVW    #CDT$C_CLOSED,-         ; Move state to closed
              28 A3      0E00  3114                  CDT$W_STATE(R3)         ;
            0126 C4  97  0E02  3115          DECB    PDT$B_NOCURCON(R4)      ; Decrement # of current connections.
                     0E06  3116
     50  14 A3  02  00  EF  0E06  3117          EXTZV   #0,#2,CDT$L_RCONID(R3),R0; Get index of CDT pointer for connection.
                     0E0C  3118
     00 0127 C4  50  E4  0E0C  3119          BBSC    R0,PDT$B_CONBITMAP(R4),10$ ; Clear the connection active bit
                     0E12  3120 10$:
                     0E12  3121
                     0E12  3122 ; The following two instructions save the available credits on the connection
                     0E12  3123 ;  that we are disconnecting so that we can later re-connect.  This is only
                     0E12  3124 ;  useful if dis-connecting and later re-connecting do not have an FPC$MRESET
                     0E12  3125 ;  done in between.  FPC$MRESET re-inits the credits for all possible
                     0E12  3126 ;  connections (in INIT_PU_PDT, called from HARDWARE_INIT).
                     0E12  3127
      51   00F4 C440  3E  0E12  3128          MOVAW   PDT$W_PU_CREDO(R4)[R0],R1; R1=> Repository of credits for this
                     0E18  3129                                              ;  connection ID.
          61   40 A3  B0  0E18  3130          MOVW    CDT$W_SEND(R3),(R1)     ; Save current credits for later connect.
                     0E1C  3131 ; End of code that saves available credits.
                     0E1C  3132
            F2DC  CF  9E  0E1C  3133          MOVAB   NULL_CDT,-
           00E4 C440      0E20  3134                  PDT$[_PU_CDTARY(R4)[R0] ; Reset ptr.
        00000000'GF  16  0E24  3135          JSB     G^SCS$DEALL_CDT         ; Deallocate R3 => CDT.
             50  01  3C  0E2A  3136          MOVZWL  S^#SS$_NORMAL,R0        ; Normal return status.
                 05  0E2D  3137          RSB                             ;
```

```
                     0E2E  3139          .SBTTL  SEQUENCED MESSAGE CALLS
                     0E2E  3140          .SBTTL  +       FPC$ALLOCMSG,   ALLOCATE A MESSAGE BUFFER
                     0E2E  3141
                     0E2E  3142 ;+
                     0E2E  3143 ; FPC$ALLOCMSG is optimized for the case where all resources that are
                     0E2E  3144 ; allocated are available.  FPC$ALLOCMSG  first checks the state of the
                     0E2E  3145 ; CONNECTION to assure that it is OPEN.  Then it allocates a send credit
                     0E2E  3146 ; and a buffer.  Finally it points R2 at the application (MSCP) portion
                     0E2E  3147 ; of the buffer, stores R2 in CDRP$L_MSG_BUF and returns a success code.
                     0E2E  3148 ;
                     0E2E  3149 ; Exceptions to this flow are handled out of line.
                     0E2E  3150 ;
                     0E2E  3151 ; First, if the CONNECTION state is NOT open, we return the SS$_ILLCDTST
                     0E2E  3152 ; status.
                     0E2E  3153 ;
                     0E2E  3154 ; If no send credits are available, the thread is suspended on the
                     0E2E  3155 ; CDT$L_CRWAITQBL.
                     0E2E  3156 ;
                     0E2E  3157 ; Finally, if no buffers are available, we first try to scare some up by
                     0E2E  3158 ; calling internal subroutine POLL_CMDRING, which polls the command ring
                     0E2E  3159 ; to free up slots and buffers.  If upon return from this call, a buffer
                     0E2E  3160 ; is available, we simply rejoin the mainline code.  If however, buffers
                     0E2E  3161 ; are still NOT available, we return the previously allocated send credit
                     0E2E  3162 ; and suspend the thread on PDT$L_BUFQBL.
                     0E2E  3163 ;
                     0E2E  3164 ; Upon resumption of threads from either of these wait Q's, we simply
                     0E2E  3165 ; branch back to the start of the routine and try all over again.
                     0E2E  3166 ;
                     0E2E  3167 ; Inputs:
                     0E2E  3168 ;
                     0E2E  3169 ;       R4                      -Addr of PDT
                     0E2E  3170 ;       R5                      -Addr of CDRP
                     0E2E  3171 ;       CDRP$L_CDT              -Addr of CDT
                     0E2E  3172 ;
                     0E2E  3173 ; Outputs:
                     0E2E  3174 ;
                     0E2E  3175 ;       R0                      -Status:  SS$_NORMAL, SS$_ILLCDTST
                     0E2E  3176 ;       R1                      -Destroyed
                     0E2E  3177 ;       R2                      -Addr of message buffer, if status=success
                     0E2E  3178 ;       Other registers         -Preserved
                     0E2E  3179 ;
                     0E2E  3180 ;       CDRP$L_MSG_BUF          -Addr of message buffer, if status=success
                     0E2E  3181 ;-
                     0E2E  3182
                     0E2E  3183
                     0E2E  3184 FPC$ALLOCMSG::
                     0E2E  3185
 52   24 A5    D0     0E2E  3186          MOVL    CDRP$L_CDT(R5),R2       ; Get CDT addr
                     0E32  3187          $CHK_CDTSTATE   -               ; Verify connection state
                     0E32  3188                  OPEN,-                  ;  is open.
                     0E32  3189                  ERROR=STATE_ERR,-       ; Else report error to caller
                     0E32  3190                  CDT=R2                  ;
                     0E3B  3191
      40 A2    B7     0E3B  3192          DECW    CDT$W_SEND(R2)          ; Allocate the send credit.
         13    19     0E3E  3193          BLSS    20$                     ; LSS means no credits available.
50  0108 D4    0F     0E40  3194          REMQUE  @PDT$L_PU_FQFL(R4),R0   ; R0 => free buffer.
         12    1D     0E45  3195          BVS     30$                     ; VS implies NO buffers.
```

```
                     0E47  3196 10$:
      52   14 A0    9E  0E47  3197          MOVAB   UDAB$T_TEXT(R0),R2       ; R2 => MSCP portion of buffer.
      1C A5   52    D0  0E4B  3198          MOVL    R2,CDRP$L_MSG_BUF(R5)    ; Return to caller in CDRP as well.
         50   01    3C  0E4F  3199          MOVZWL  S^#SS$_NORMAL,R0         ; Success return.
                     05  0E52  3200          RSB                              ; And return.
                     0E53  3201
                     0E53  3202 20$:
      51   3C A2    D0  0E53  3203          MOVL    CDT$L_CRWAITQBL(R2),R1   ; R1 => where to INSQUE to await credits.
              13    11  0E57  3204          BRB     40$                      ; Branch around to common suspend code.
                     0E59  3205 30$:
            0362    30  0E59  3206          BSBW    POLL_CMDRING             ; Reclaim released buffers from
                     0E5C  3207                                               ;  command ring.
   50   0108 D4    0F  0E5C  3208          REMQUE  @PDT$L_PU_FQFL(R4),R0    ; Again try for R0 => free buffer.
              E4    1C  0E61  3209          BVC     10$                      ; VC implies buffers. Branch to mainline.
      52   24 A5    D0  0E63  3210          MOVL    CDRP$L_CDT(R5),R2        ; Refresh R2=>CDT after call to POLL_CMDRING
   51   011C C4    D0  0E67  3211          MOVL    PDT$L_PU_BUFQBL(R4),R1   ; R1 => where to INSQUE to await buffers.
                     0E6C  3212 40$:
            40 A2    B6  0E6C  3213          INCW    CDT$W_SEND(R2)           ; Return improper allocate.
            18 A5  8ED0  0E6F  3214          POPL    CDRP$L_SAVD_RTN(R5)      ; Save high level return.
                     0E73  3215          $SUSP_SCS       (R1)             ; Suspend on R1 => wait Q element.
            18 A5    DD  0E8B  3216          PUSHL   CDRP$L_SAVD_RTN(R5)      ; Restore high level return.
              9E    11  0E8E  3217          BRB     FPC$ALLOCMSG             ; And go back to check on credits.
```

```
                     0E90  3219          .SBTTL  +       FPC$DEALLOMSG,  DEALLOCATE A MESSAGE BUFFER
                     0E90  3220
                     0E90  3221  ;+
                     0E90  3222  ; FPC$DEALLOMSG resets the message address specified by the caller to
                     0E90  3223  ; the top of the message buffer and clears CDRP$L_MSG_BUF.  It then
                     0E90  3224  ; decides whether to insert the free buffer onto the RESPONSE RING
                     0E90  3225  ; or onto the free Q of buffers, with the RESPONSE RING having priority,
                     0E90  3226  ; and being selected whenever it is not entirely full.  If the RESPONSE
                     0E90  3227  ; RING is full then the buffer is INSQUEd onto PDT$L_PU_FQBL.
                     0E90  3228  ;
                     0E90  3229  ; If this INSQUE represents the first buffer on the free Q, then we
                     0E90  3230  ; attempt to resume any threads waiting for buffers (PDT$L_PU_BUFQFL).
                     0E90  3231  ; While the free Q remains non-empty and there exist threads waiting
                     0E90  3232  ; for buffers, we continue to resume the threads until either we run
                     0E90  3233  ; out of buffers or we run out of threads to resume.
                     0E90  3234  ;
                     0E90  3235  ; Internal entrypoint Q_DEALLOC_BUF is called from POLL_CMDRING after
                     0E90  3236  ; R2 has been pointed at the buffer header.  Also internal entrypoint
                     0E90  3237  ; INSERT_IN_RRING is called from INIT_PU_PDT at CONNECT time in order to
                     0E90  3238  ; prime the RESPONSE RING.
                     0E90  3239  ; Inputs:
                     0E90  3240  ;
                     0E90  3241  ;       R4                      -Addr of PDT
                     0E90  3242  ;       R5                      -Addr of CDRP
                     0E90  3243  ;       CDRP$L_MSG_BUF          -Addr of message buffer
                     0E90  3244  ;
                     0E90  3245  ; Outputs:
                     0E90  3246  ;
                     0E90  3247  ;       R0-R2                   -Destroyed
                     0E90  3248  ;       Other registers         -Preserved
                     0E90  3249  ;       CDRP$L_MSG_BUF          -Cleared
                     0E90  3250  ;-
                     0E90  3251
                     0E90  3252  FPC$DEALRGMSG::
          52    14   C2  0E90  3253          SUBL    #UDAB$T_TEXT,R2         ; R2 => buffer header.
                08   11  0E93  3254          BRB     Q_DEALLOC_BUF           ;  and branch around to common code.
                     0E95  3255  FPC$DEALLOMSG::
                     0E95  3256
                14   C3  0E95  3257          SUBL3   #UDAB$T_TEXT,-
          52    1C   A5  0E97  3258                  CDRP$L_MSG_BUF(R5),R2   ; R2 => buffer header.
                1C   A5  D4  0E9A  3259          CLRL    CDRP$L_MSG_BUF(R5)      ; Prevent spurious deallocates.
                     0E9D  3260
                     0E9D  3261  FPC$QUEUEDG::
                     0E9D  3262  Q_DEALLOC_BUF:                              ; Called here from POLL_CMDRING,
                     0E9D  3263                                              ;  with R2 => free buffer.
                10   91  0E9D  3264          CMPB    #UDA$K_RINGSIZE,-       ; See if response ring is full up.
          0125  C4       0E9F  3265                  PDT$B_RRINGCNT(R4)
                2A   15  0EA2  3266          BLEQ    INSERT_IN_FREEQ         ; LEQ implies full. Goto put on free Q.
                     0EA4  3267
                     0EA4  3268  INSERT_IN_RRING:                            ; Here R2 => buffer to insert.
                     0EA4  3269
          04    00   EF  0EA4  3270          EXTZV   #0,#UDA$K_RINGEXP,-     ; R0 = index of slot to use in response
     50   0123  C4       0EA7  3271                  PDT$B_RRINGINX(R4),R0   ;  ring.
                     0EAB  3272
             0A   A2  90  0EAB  3273          MOVB    UDAB$B_BUFFNO(R2),-     ; Label new contents of this slot.
        013C  C440       0EAE  3274                  PDT$B_RRCONTENT(R4)[R0]
                     0EB2  3275
```

```
08 A2   50   90  0EB2  3276          MOVB    R0,UDAB$B_RINGINX(R2)   ; Remember where this buffer is for
09 A2   01   90  0EB6  3277          MOVB    #1,UDAB$B_RINGNO(R2)    ;  debugging.
                0EBA  3278
                0EBA  3279          ASSUME  UDAB$B_CREDTYPE EQ      UDAB$W_MSG_LEN+2
                0EBA  3280          ASSUME  UDAB$B_CONID    EQ      UDAB$B_CREDTYPE+1
      3C   D0   0EBA  3281          MOVL    #UDAB$C_LENGTH-UDAB$T_TEXT,-    ; Initialize response buffer
   10 A2        0EBC  3282                  UDAB$W_MSG_LEN(R2)              ;  length and zero credits,
                0EBE  3283                                                  ;  type and conid.
                0EBE  3284
   0C A2   D0   0EBE  3285          MOVL    UDAB$L_DESCRIP(R2),-    ; Fill in command ring slot.
0208 C440       0EC1  3286                  PDT$L_RSPRING(R4)[R0]
  0123 C4   96  0EC5  3287          INCB    PDT$B_RRINGINX(R4)      ; Next time use next slot in ring.
  0125 C4   96  0EC9  3288          INCB    PDT$B_RRINGCNT(R4)      ; Increment # slots in use.
                0ECD  3289
                0ECD  3290 INSERT_RTN:
                0ECD  3291          RSB                             ; And return to caller or caller's caller.
                0ECE  3292
                0ECE  3293 INSERT_IN_FREEQ:
                0ECE  3294
      62   0E   0ECE  3295          INSQUE  UDAB$L_FLINK(R2),-      ; Insert free buffer onto free Q.
 010C D4         0ED0  3296                  @PDT$L_PU_FQBL(R4)
      F8   12   0ED3  3297          BNEQ    INSERT_RTN              ; NEQ implies not first buffer on Q.
                0ED5  3298 10$:
                0ED5  3299          $RESUME_FP      -               ; Resume first thread awaiting buffers.
                0ED5  3300                  @PDT$L_PU_BUFQFL(R4),-
                0ED5  3301                  QEMPTY=INSERT_RTN       ; Label to branch to if Q empty.
                0EE9  3302
  0104 C4   D1  0EE9  3303          CMPL    PDT$L_PU_FQPTR(R4),-    ; Test for emptiness of free Q.
  0108 C4       0EED  3304                  PDT$L_PU_FQFL(R4)       ;  If list points to itself, then empty.
      E3   12   0EF0  3305          BNEQ    10$                     ; NEQ means NOT empty, so try to resume
                05  0EF2  3306          RSB                             ; Return to caller if empty.
```

```
                     0EF3  3308          .SBTTL  +       FPC$RCHMSGBUF,  RECYCLE MESSAGE BUFFER
                     0EF3  3309          .SBTTL  +                       AT HIGH PRIORITY
                     0EF3  3310          .SBTTL  +       FPC$RCLMSGBUF,  RECYCLE MESSAGE BUFFER
                     0EF3  3311          .SBTTL  +                       AT LOW PRIORITY
                     0EF3  3312
                     0EF3  3313 ;+
                     0EF3  3314 ; FPC$RCxMSGBUF checks if there is at least one send credit.  If
                     0EF3  3315 ; not, the SYSAP is suspended until there is.  FPC$RCxMSGBUF then
                     0EF3  3316 ; decrements the send credit.  The wait, if required, places the
                     0EF3  3317 ; SYSAP CDRP at the end of the wait queue for low priority and at
                     0EF3  3318 ; the head of the queue for high priority.  The address of the
                     0EF3  3319 ; buffer being recycled is returned in R2.
                     0EF3  3320 ;
                     0EF3  3321 ; Inputs:
                     0EF3  3322 ;
                     0EF3  3323 ;       R4                      -Addr of PDT
                     0EF3  3324 ;       R5                      -Addr of CDRP
                     0EF3  3325 ;       CDRP$L_MSG_BUF          -Addr of message buffer
                     0EF3  3326 ;       CDRP$L_CDT              -Addr of CDT
                     0EF3  3327 ;
                     0EF3  3328 ; Outputs:
                     0EF3  3329 ;
                     0EF3  3330 ;       R0                      -Status:  SS$_NORMAL, SS$_ILLCDTST
                     0EF3  3331 ;       R1                      -Destroyed
                     0EF3  3332 ;       R2                      -Addr of message buffer
                     0EF3  3333 ;       Other registers         -Preserved
                     0EF3  3334 ;-
                     0EF3  3335
                     0EF3  3336
                     0EF3  3337 FPC$RCHMSGBUF::
                     0EF3  3338
51   24 A5   D0      0EF3  3339          MOVL    CDRP$L_CDT(R5),R1       ; R1 => CDT.
                     0EF7  3340          $CHK_CDTSTATE   -               ; Assure that connection open.
                     0EF7  3341                  OPEN,-
                     0EF7  3342                  ERROR=STATE_ERR,-
                     0EF7  3343                  CDT=R1
     40 A1   B7      0F00  3344          DECW    CDT$W_SEND(R1)          ; Test decrement a credit.
        08   19      0F03  3345          BLSS    10$                     ; LSS implies there were none to give.
52   1C A5   D0      0F05  3346          MOVL    CDRP$L_MSG_BUF(R5),R2   ; Return R2 => msg buffer.
   50   01   3C      0F09  3347          MOVZWL  S^#SS$_NORMAL,R0        ; Return status for caller
             05      0F0C  3348          RSB                             ; And return.
                     0F0D  3349 10$:                                     ; Here only if allocation failure.
     40 A1   B6      0F0D  3350          INCW    CDT$W_SEND(R1)          ; Restore from above test decrement.
     18 A5 8ED0      0F10  3351          POPL    CDRP$L_SAVD_RTN(R5)     ; Save first level return address.
                     0F14  3352          $SUSP_SCS       -               ; Suspend this thread at HEAD of
                     0F14  3353                  CDT$L_CRWAITQFL(R1)     ;  allocation wait list.
     18 A5   DD      0F2D  3354          PUSHL   CDRP$L_SAVD_RTN(R5)     ; Restore first level return to stack.
        C1   11      0F30  3355          BRB     FPC$RCHMSGBUF           ; Go back and try to allocate.
                     0F32  3356
                     0F32  3357 FPC$RCLMSGBUF::
                     0F32  3358
51   24 A5   D0      0F32  3359          MOVL    CDRP$L_CDT(R5),R1       ; R1 => CDT.
                     0F36  3360          $CHK_CDTSTATE   -               ; Assure that connection open.
                     0F36  3361                  OPEN,-
                     0F36  3362                  ERROR=STATE_ERR,-
                     0F36  3363                  CDT=R1
     40 A1   B7      0F3F  3364          DECW    CDT$W_SEND(R1)          ; Test decrement a credit.
```

```
                                ;+   AT LOW PRIORITY

                08   19  0F42  3365          BLSS    10$                             ; LSS implies there were none to give.
          52  1C A5   D0  0F44  3366          MOVL    CDRP$L_MSG_BUF(R5),R2           ; Return R2 => msg buffer.
           50   01   3C  0F48  3367          MOVZWL  S^#SS$_NORMAL,R0                ; Return status for caller
                     05  0F4B  3368          RSB                                     ; And return.
                         0F4C  3369 10$:                                             ; Here only if allocation failure.
              40 A1   B6  0F4C  3370          INCW    CDT$W_SEND(R1)                  ; Restore from above test decrement.
              18 A5 8ED0  0F4F  3371          POPL    CDRP$L_SAVD_RTN(R5)             ; Save first level return address.
                         0F53  3372          $SUSP_SCS -                             ; Suspend this thread at TAIL of
                         0F53  3373                  @CDT$L_CRWAITQBL(R1)            ;  allocation wait list.
              18 A5   DD  0F6C  3374          PUSHL   CDRP$L_SAVD_RTN(R5)             ; Restore first level return to stack.
                 C1   11  0F6F  3375          BRB     FPC$RC_MSGBUF                   ; Go back and try to allocate.
```

```
                     0F71  3377          .SBTTL  +       FPC$SNDCNTMSG,  SEND COUNTED SEQUENCED MESSAGE
                     0F71  3378 ; Inputs:
                     0F71  3379 ;
                     0F71  3380 ;       R1                      -# bytes of application data
                     0F71  3381 ;       R4                      -Addr of PDT
                     0F71  3382 ;       R5                      -Addr of CDRP
                     0F71  3383 ;       CDRP$L_CDT(R5)          -Addr of CDT
                     0F71  3384 ;       CDRP$L_MSG_BUF(R5)      -Addr of message (user portion)
                     0F71  3385 ;       CDRP$L_RSPID(R5)        -RSPID (to set RETFLG)
                     0F71  3386 ;
                     0F71  3387 ; Outputs:
                     0F71  3388 ;
                     0F71  3389 ;       R0                      -Status:  SS$_NORMAL, SS$_ILLCDTST
                     0F71  3390 ;       R1,R2                   -Destroyed
                     0F71  3391 ;       Other registers         -Preserved
                     0F71  3392 ;-
                     0F71  3393
                     0F71  3394 FPC$SNDCNTMSG::
                     0F71  3395
50   24 A5    D0     0F71  3396          MOVL    CDRP$L_CDT(R5),R0       ; R0 => CDT
                     0F75  3397          $CHK_CDTSTATE   -               ; Verify connection is
                     0F75  3398                  OPEN,-                  ;  open
                     0F75  3399                  ERROR=STATE_ERR,-       ; Else report error to SYSAP
                     0F75  3400                  CDT=R0
                     0F7E  3401
     14  C3          0F7E  3402          SUBL3   #UDAB$T_TEXT,-          ; Point to buffer header.
52   1C A5           0F80  3403                  CDRP$L_MSG_BUF(R5),R2   ; R2 => buffer header.
     1C A5    D4     0F83  3404          CLRL    CDRP$L_MSG_BUF(R5)      ; Prevent spurious deallocates.
                     0F86  3405
                     0F86  3406          ASSUME  UDAB$B_CREDTYPE         EQ      UDAB$W_MSG_LEN+2
                     0F86  3407          ASSUME  UDAB$B_CONID            EQ      UDAB$B_CREDTYPE+1
   10 A2    51     D0     0F86  3408          MOVL    R1,UDAB$W_MSG_LEN(R2)   ; Put message length in header.
      14 A0    90     0F8A  3409          MOVB    CDT$L_RCONID(R0),-      ; Put remote connection ID in message
      13 A2           0F8D  3410                  UDAB$B_CONID(R2)        ;  header.
                     0F8F  3411
   10 A5    53     7D     0F8F  3412          MOVQ    R3,CDRP$L_FR3(R5)
      0C A5 8ED0     0F93  3413          POPL    CDRP$L_FPC(R5)          ; Save context in CDRP.
                     0F97  3414
      10    91     0F97  3415          CMPB    #UDA$K_RINGSIZE,-       ; See if any slots available in command
    0122 C4           0F99  3416                  PDT$B_CRINGCNT(R4)      ;  ring.
      2A    15     0F9C  3417          BLEQ    CRING_FULL              ; LEQ implies ring full.
                     0F9E  3418
                     0F9E  3419 INSERT_IN_CRING:                        ; Called from POLL_CMDRING with
                     0F9E  3420                                         ;  R2 => buffer.
                     0F9E  3421
    04   00    EF     0F9E  3422          EXTZV   #0,#UDA$K_RINGEXP,-     ; R0 = index of slot to use in command
50  0120 C4           0FA1  3423                  PDT$B_CRINGINX(R4),R0   ;  ring.
                     0FA5  3424
                     0FA5  3425
                     0FA5  3432
     0A A2    90     0FA5  3433          MOVB    UDAB$B_BUFFNO(R2),-     ; Label new contents of this slot.
  012C C440           0FA8  3434                  PDT$B_CRCONTENT(R4)[R0]
                     0FAC  3435
                     0FAC  3436          ASSUME  UDAB$B_RINGNO   EQ      UDAB$B_RINGINX+1
08 A2    50    9B     0FAC  3437          MOVZBW  R0,UDAB$B_RINGINX(R2)   ; Remember where this buffer is for
                     0FB0  3438                                         ;  debugging.
                     0FB0  3439
```

```
                        0FB0  3440 ENABLE_COMMAND_START:
          0C A2   D0    0FB0  3441         MOVL    UDAB$L_DESCRIP(R2),-    ; Fill in command ring slot.
      0248 C440         0FB3  3442                 PDT$L_CMDRING(R4)[R0]   ; NOTE, this instruction copied below
                        0FB7  3443                                         ;  to allow for dynamic patching to
                        0FB7  3444                                         ;  enable tracing.
                        0FB7  3445 ENABLE_COMMAND_END:
   51   0100 C4   D0    0FB7  3446         MOVL    PDT$L_PU_CSR(R4),R1     ; R1 => UDA CSR.
        50   61   B0    0FBC  3447         MOVW    UDAIP(R1),R0            ; Jiggle controller register to force
                        0FBF  3448                                         ;  polling of command ring.
        0120 C4   96    0FBF  3449         INCB    PDT$B_CRINGINX(R4)      ; Next time use next slot in ring.
        0122 C4   96    0FC3  3450         INCB    PDT$B_CRINGCNT(R4)      ; Increment # slots in use.
                        0FC7  3451
                  05    0FC7  3452         RSB                             ; And return to caller or caller's caller.
                        0FC8  3453
                        0FC8  3454 CRING_FULL:
                        0FC8  3455
                        0FC8  3456 ; The following instructions (commented out) were useful in debugging only.
                        0FC8  3457 ;       BEQL    10$                     ; EQL means ring full not overflowed.
                        0FC8  3458 ;       BUG_CHECK       UDAPORT,FATAL   ; Here we have overflowed command ring.
                        0FC8  3459 ;10$:
             62   0E    0FC8  3460         INSQUE  UDAB$L_FLINK(R2),-      ; Insert onto tail of backed up buffers.
          0114 D4         0FCA  3461                 @PDT$L_PU_SNDQBL(R4)
            01EE   30    0FCD  3462         BSBW    POLL_CMDRING            ; Free up any possible slots in
                        0FD0  3463                                         ;  command ring and dequeue from SND Q.
                  05    0FD0  3464         RSB                             ; Return to caller's caller.
                        0FD1  3465
                        0FD1  3466 ENABLE_COMMAND_CODE:                    ; If we enable tracing, the driver will
                        0FD1  3467                                         ;  dynamically patch location
                        0FD1  3468                                         ;  ENABLE_COMMAND_START to BRW here.
            F4A2   30    0FD1  3469         BSBW    TRACE_COMMAND           ; Copy command buffer to trace table.
                        0FD4  3470
          0C A2   D0    0FD4  3471         MOVL    UDAB$L_DESCRIP(R2),-    ; Fill in command ring slot.
      0248 C440         0FD7  3472                 PDT$L_CMDRING(R4)[R0]
            FFD9   31    0FDB  3473         BRW     ENABLE_COMMAND_END      ; Branch back to normal stream.
                        0FDE  3474
         0000001E       0FDE  3475 ENABLE_COMMAND_OFFSET=ENABLE_COMMAND_CODE-ENABLE_COMMAND_START-3
```

```
                     0FDE  3477          .SBTTL  +       FPC$MAPIRP,     Map a buffer
                     0FDE  3478
                     0FDE  3479 ;+
                     0FDE  3480 ; FPC$MAPIRP - map a user buffer given IRP (CDRP) values.
                     0FDE  3481 ;
                     0FDE  3482 ; Inputs:
                     0FDE  3483 ;       R4 => PDT
                     0FDE  3484 ;       R5 => CDRP
                     0FDE  3485 ;       CDRP$L_LBUFH_AD => area to fill in with UNIBUS virtual address of
                     0FDE  3486 ;                          user buffer.
                     0FDE  3487 ;
                     0FDE  3488 ; Outputs:
                     0FDE  3489 ;       CDRP$L_UBARSRCE filled in with a) datapath assigned, b) # UBA map
                     0FDE  3490 ;                                      registers assigned, c) 1st map register
                     0FDE  3491 ;
                     0FDE  3492 ;       @CDRP$L_LBUFH_AD filled in with UNIBUS virtual address of user buffer
                     0FDE  3493 ;
                     0FDE  3494 ;       R0-R2   destroyed
                     0FDE  3495 ;       Other registers preserved.
                     0FDE  3496 ;-
                     0FDE  3497
                     0FDE  3498 FPC$MAPIRP::
          3C A5    D4  0FDE  3499          CLRL    CDRP$L_UBARSRCE(R5)     ; Initialize.
          18 A5  8ED0  0FE1  3500          POPL    CDRP$L_SAVD_RTN(R5)     ; Save return for two level process.
                     0FE5  3501
        F1B3 DF    16  0FE5  3502          JSB     @REQDATAPATH_TV         ; Allocate datapath, if any available.
                     0FE9  3503
00000000'GF    16  0FE9  3504          JSB     G^IOC$REQMAPUDA         ; Allocate map registers.
00000000'GF    16  0FEF  3505          JSB     G^IOC$LUBAUDAMAP        ; Load map registers for this transfer.
                     0FF5  3506
                     0FF5  3507 ; Here we fill in the local buffer handle.  A description of the UBA mapping
                     0FF5  3508 ;       resources assigned to this transfer are currently in CDRP$L_UBARSRCE.
                     0FF5  3509 ;       We calculate the UNIBUS virtual address of the transfer and put this
                     0FF5  3510 ;       value into the @CDRP$L_LBUFH_AD.
                     0FF5  3511
      50    D0 A5    3C  0FF5  3512          MOVZWL  CDRP$W_BOFF(R5),R0      ; Calculate UNIBUS virtual address of
                     0FF9  3513                                              ;  user buffer. First get byte offset.
                     0FF9  3514
            3F A5    F0  0FF9  3515          INSV    CDRP$L_UBARSRCE+UBMD$B_DATAPATH(R5),-
50    08    18         0FFC  3516                  #24,#8,R0               ; Place datapath number in high byte.
      50    01    8A  0FFF  3517          BICB    #1,R0                   ; Clear low bit in case of odd address.
                     1002  3518
            3C A5    F0  1002  3519          INSV    CDRP$L_UBARSRCE+UBMD$W_MAPREG(R5),-
50    09    09         1005  3520                  #9,#9,R0                ; High order of UNIBUS virtual
                     1008  3521                                              ;  address is map register #.
      51    2C A5    D0  1008  3522          MOVL    CDRP$L_LBUFH_AD(R5),R1  ; R1 => destination for buffer handle.
                     100C  3523
            81    50 D0  100C  3524          MOVL    R0,(R1)+                ; Write "UBA" buffer
                  61 7C  100F  3525          CLRQ    (R1)                    ;  handle and zero out rest.
                     1011  3526
            18 B5    17  1011  3527          JMP     @CDRP$L_SAVD_RTN(R5)    ; Return to top level caller.
```

```
                     1014  3529          .SBTTL  +       FPC$MAPIRP_UV1, Map a buffer for uVAX I
                     1014  3530
                     1014  3531 ;+
                     1014  3532 ; FPC$MAPIRP_UV1 - map a user buffer given IRP (CDRP) values.
                     1014  3533 ;
                     1014  3534 ; Inputs:
                     1014  3535 ;       R4 => PDT
                     1014  3536 ;       R5 => CDRP
                     1014  3537 ;       CDRP$L_LBUFH_AD => area to fill in with UNIBUS virtual address of
                     1014  3538 ;                          user buffer.
                     1014  3539 ;
                     1014  3540 ; Outputs:
                     1014  3541 ;       CDRP$L_UBARSRCE filled in with a) datapath assigned, b) # UBA map
                     1014  3542 ;                                      registers assigned, c) 1st map register
                     1014  3543 ;
                     1014  3544 ;       @CDRP$L_LBUFH_AD filled in with UNIBUS virtual address of user buffer
                     1014  3545 ;
                     1014  3546 ;       R0-R2   destroyed
                     1014  3547 ;       Other registers preserved.
                     1014  3548 ;-
                     1014  3549
                     1014  3550 FPC$MAPIRP_UV1::
3C A5   D4           1014  3551          CLRL    CDRP$L_UBARSRCE(R5)     ; Initialize.
D0 A5   E8           1017  3552          BLBS    CDRP$W_BOFF(R5),-       ; Branch around if Not word aligned.
   30                101A  3553                  MAP_UNALIGN
                     101B  3554 MAP_ODD:                                 ; Label to allow branch back.
                     101B  3555
18 A5 8ED0           101B  3556          POPL    CDRP$L_SAVD_RTN(R5)     ; Save return for two level process.
                     101F  3557
00000000'GF   16     101F  3558          JSB     G^IOC$REQMAPUDA         ; Allocate map registers.
00000000'GF   16     1025  3559          JSB     G^IOC$LUBAUDAMAP        ; Load map registers for this transfer.
                     102B  3560
                     102B  3561 ; Here we fill in the local buffer handle.  A description of the UBA mapping
                     102B  3562 ;       resources assigned to this transfer are currently in CDRP$L_UBARSRCE.
                     102B  3563 ;       We calculate the UNIBUS virtual address of the transfer and put this
                     102B  3564 ;       value into the @CDRP$L_LBUFH_AD.
                     102B  3565
 50   D0 A5   3C     102B  3566          MOVZWL  CDRP$W_BOFF(R5),R0      ; Calculate UNIBUS virtual address of
                     102F  3567                                          ;  user buffer. First get byte offset.
                     102F  3568
    3C A5   F0       102F  3569          INSV    CDRP$L_UBARSRCE+UBMD$W_MAPREG(R5),-
 50   09   09         1032  3570                  #9,#9,R0                ; High order of UNIBUS virtual
                     1035  3571                                          ;  address is map register #.
 51   2C A5   D0     1035  3572          MOVL    CDRP$L_LBUFH_AD(R5),R1  ; R1 => destination for buffer handle.
                     1039  3573
81   50  80000000 8F   C9  1039  3574          BISL3   #UQPORT_M_MAPPED,R0,(R1)+; Write BUS virtual address of buffer
    81   F08F CF   D0     1041  3575          MOVL    PU$L_MPRVAD,(R1)+       ;  and physical address of the pseudo
            61     D4     1046  3576          CLRL    (R1)                    ;  map registers and zero out rest.
                     1048  3577
          18 B5     17     1048  3578          JMP     @CDRP$L_SAVD_RTN(R5)    ; Return to top level caller.
```

```
                     104B  3580          .SBTTL  +       MAP_UNALIGN     uVAX I Q-BUS Map Unaligned Buffer
                     104B  3581 ;+
                     104B  3582 ; MAP_UNALIGN - routine to allocate the Aligned Page (i.e. 512 physically
                     104B  3583 ;  contiguous bytes) and to copy upto 512 bytes of user data to this
                     104B  3584 ;  buffer. Things to keep in mind:
                     104B  3585 ;
                     104B  3586 ;  1. Allocation of the Aligned Page is governed by PDT$L_PQ_POWNER.  This
                     104B  3587 ;       longword, if zero, indicates the Aligned Page is available.  Non-zero
                     104B  3588 ;       implies it is allocated and the owner's CDRP is saved in the longword.
                     104B  3589 ;
                     104B  3590 ;  2. PDT$L_PQ_PGQFL, PDT$L_PQ_PGQBL are a queue header that lists CDRP's
                     104B  3591 ;       waiting for the Aligned Page.
                     104B  3592 ;
                     104B  3593 ;  3. PDT$L_PQ_SVPPTE points to a PTE slot in the System Page table that
                     104B  3594 ;       belongs to this Port.  PDT$L_PQ_UBFSVA contains the System Virtual
                     104B  3595 ;       Address associated with this PTE.
                     104B  3596 ;
                     104B  3597 ; "Mapping" an unaligned user buffer means copying all, or the first 512 bytes,
                     104B  3598 ;  whichever is less, of the user data to the Aligned Page.  This means that
                     104B  3599 ;  CDRP$L_BCNT (if originally >512) is reduced to 512.  The copying uses
                     104B  3600 ;  the PTE slot pointed at by PDT$L_PQ_SVPPTE, and the System Virtual Address
                     104B  3601 ;  associated with it.  Essentially, the PTE that points to the first segment
                     104B  3602 ;  (i.e. the user data in the first page of the user's buffer) is put into
                     104B  3603 ;  our reserved PTE slot.  Then the user data in this segment is copied.  Then
                     104B  3604 ;  the next user PTE is loaded into our reserved PTE slot and the second (if
                     104B  3605 ;  any) segment is copied.
                     104B  3606 ;
                     104B  3607 ; Inputs:
                     104B  3608 ;       R4 => PDT
                     104B  3609 ;       R5 => CDRP
                     104B  3610 ;
                     104B  3611 ; Outputs:
                     104B  3612 ;       Aligned Page allocated and user data copied.
                     104B  3613 ;       Registers R0-R2 modified.
                     104B  3614 ;-
                     104B  3615
                     104B  3616 MAP_UNALIGN:                                    ; We come here for unaligned transfers.
50   00DC C4   D0    104B  3617          MOVL    PDT$L_UCB0(R4),R0             ; R0 => port UCB.
          07   E0    1050  3618          BBS     #PS1_V_OD,-                   ; If controller supports odd addresses,
     00AC C0         1052  3619                  UCB$Q_PORTSTEP1(R0),-         ;  then branch back in line.
          C5         1055  3620                  MAP_ODD
     0CA0 C4   D5    1056  3621          TSTL    PDT$L_PQ_POWNER(R4)           ; Test if aligned page available.
          11   13    105A  3622          BEQL    10$                           ; EQL implies available.
  10 A5   53   7D    105C  3623          MOVQ    R3,CDRP$L_FR3(R5)             ; Else save context.
       28 B5   B6    1060  3624          INCW    @CDRP$L_RWCPTR(R5)            ; Bump UCB$L_RWAITCNT.
       0C A5 8ED0    1063  3625          POPL    CDRP$L_FPC(R5)                ; Save return point.
          65   0E    1067  3626          INSQUE  CDRP$L_FQFL(R5),-             ; Queue this CDRP to resource wait Q.
     0C9C D4         1069  3627                  @PDT$L_PQ_PGQBL(R4)
               05    106C  3628          RSB                                   ; Return to caller's caller.
                     106D  3629 10$:
0CA0 C4    55   D0    106D  3630          MOVL    R5,PDT$L_PQ_POWNER(R4)        ; Allocate Aligned Page.
 00000200 8F   D1    1072  3631          CMPL    #NO_PHYCONTIGBYT,-            ; Test for transfer shorter than length
       D2 A5         1078  3632                  CDRP$L_BCNT(R5)               ;  of aligned contiguous buffer.
          08   18    107A  3633          BGEQ    20$                           ; GEQ implies short transfer.
 00000200 8F   D0    107C  3634          MOVL    #NO_PHYCONTIGBYT,-            ; Reduce transfer to total bytes in
       D2 A5         1082  3635                  CDRP$L_BCNT(R5)               ;  aligned contiguous buffer.
                     1084  3636 20$:
```

```
                     1084  3637
                     1084  3638 ; Here after successfully allocating the aligned buffer, if the user operation
                     1084  3639 ; is a WRITE, we copy the user's data to the aligned buffer.  If the user
                     1084  3640 ; operation is a READ, we branch around the copy and merely fillin the
                     1084  3641 ; CDRP$L_LBUFH_AD field.  The data from a READ is copied to the user
                     1084  3642 ; buffer after the I/O is complete.
                     1084  3643
        01    E0     1084  3644          BBS     #IRP$V_FUNC,-           ; If a READ, branch around.
11 CA A5             1086  3645                  CDRP$W_STS(R5),40$
        1B    10     1089  3646          BSBB    SETUP_COPY_SEG1         ; Call to setup copy of user data.
                     108B  3647                                          ;  This is a complex routine that
                     108B  3648                                          ;  returns: R0 = # bytes in first
                     108B  3649                                          ;  segment, R1 => user data,
                     108B  3650                                          ;  R2 => aligned page, (SP) = # bytes
                     108B  3651                                          ;  in second segment (or zero if none),
                     108B  3652                                          ;  4(SP) (valid only if (SP) non-zero)
                     108B  3653                                          ;  zero.
                     108B  3654 30$:
  82    81    90     108B  3655          MOVB    (R1)+,(R2)+             ; Copy a byte.
    FA  50    F5     108E  3656          SOBGTR  R0,30$                  ; Loop thru entire segment.
        50  8ED0     1091  3657          POPL    R0                      ; Get length of next segment (if any).
        04    13     1094  3658          BEQL    40$                     ; EQL implies no more.
        77    10     1096  3659          BSBB    SETUP_COPY_SEG2         ; Call to setup copy of second segment.
        F1    11     1098  3660          BRB     30$                     ; Branch back to loop.
                     109A  3661 40$:
51      2C  A5   D0   109A  3662          MOVL    CDRP$L_LBUFH_AD(R5),R1  ; R1 => destination for buffer handle.
81    0CA4  C4   D0   109E  3663          MOVL    PDT$L_PQ_PGPRAD(R4),(R1)+; Copy physical address of Aligned Page
        61    7C   10A3  3664          CLRQ    (R1)                    ;  and zero out rest.
              05   10A5  3665          RSB                             ; And return to caller.
```

```
                        10A6  3667          .SBTTL  SETUP_COPY_SEG1 and SETUP_COPY_SEG2
                        10A6  3668
                        10A6  3669 ;+
                        10A6  3670 ; SETUP_COPY_SEG1 - a routine to setup a copy from (to) the aligned
                        10A6  3671 ;                   page to (from) the user's buffer.
                        10A6  3672 ;
                        10A6  3673 ; Inputs:
                        10A6  3674 ;       CDRP$L_SVAPTE   => System Virtual Address of the Page Table Entry that
                        10A6  3675 ;                          maps the first page of the user buffer into user
                        10A6  3676 ;                          space.
                        10A6  3677 ;       CDRP$W_BOFF     =  Offset in this page of user data.
                        10A6  3678 ;       PDT$L_PQ_SVPPTE => System Virtual Address of an available PTE in the
                        10A6  3679 ;                          System Page Table that we can use to map the user
                        10A6  3680 ;                          buffer into System space.
                        10A6  3681 ;       PDT$L_PQ_UBFSVA => System space address of the page associated with
                        10A6  3682 ;                          previous PTE.
                        10A6  3683 ;       PDT$L_PQ_ALGNPG => System space address of the aligned page.
                        10A6  3684 ;       CDRP$L_BCNT     =  Length of data to transfer. ( < or = 512)
                        10A6  3685 ;
                        10A6  3686 ; Outputs:
                        10A6  3687 ;       R0 = Number of bytes to copy in first segment.  The first segment is
                        10A6  3688 ;            defined by the data beginning at CDRP$W_BOFF in the first
                        10A6  3689 ;            page of the user buffer and continuing till the end of the page.
                        10A6  3690 ;       R1 => System space address of first byte in user buffer.
                        10A6  3691 ;       R2 => First byte of aligned page.
                        10A6  3692 ;       (SP)  = Either # bytes in the second segment or zero.
                        10A6  3693 ;       4(SP) = Valid only if (SP) is non-zero.  Then this is zero.
                        10A6  3694 ;
                        10A6  3695 ; Note:  CDRP$L_SAVD_RTN is modified.
                        10A6  3696 ;
                        10A6  3697
                        10A6  3698 SETUP_COPY_SEG1:
                        10A6  3699
          18 A5 8ED0    10A6  3700          POPL    CDRP$L_SAVD_RTN(R5)       ; Clean stack so that we can return
                        10AA  3701                                            ;  values to caller on it.
          CC A5   D0    10AA  3702          MOVL    CDRP$L_SVAPTE(R5),-       ; Save user SVAPTE in PDT.
        0CB0 C4         10AD  3703                  PDT$L_PQ_USRPTE(R4)
        15   00   EF     10B0  3704          EXTZV   #PTE$V_PFN,#PTE$S_PFN,-   ; Construct a PTE with the proper PFN,
          CC B5         10B3  3705                  @CDRP$L_SVAPTE(R5),-      ;  protection=KW, and own=K.  Here we
        0CAC D4         10B5  3706                  @PDT$L_PQ_SVPPTE(R4)      ;  move the PFN.
                  C8    10B8  3707          BISL    #PTE$M_VALID!-            ; And here we set VALID, the protection
                        10B9  3708                   PTE$C_KW!-               ;  and the ownership and thereby map
                        10B9  3709                   PTE$C_KOWN,-             ;  the first page of the user buffer
0CAC D4   90000000 8F   10B9  3710                  @PDT$L_PQ_SVPPTE(R4)      ;  into System Space.
     50   D0   A5   3C   10C1  3711          MOVZWL  CDRP$W_BOFF(R5),R0        ; R0 = BOFF.
51   50   0CA8 C4   C1   10C5  3712          ADDL3   PDT$L_PQ_UBFSVA(R4),R0,R1 ; R1 = SVA of 1st byte of user data.
                  7E   D4   10CB  3713          INVALID R1                        ; Invalidate virtual address.
                  7E   D4   10CE  3714          CLRL    -(SP)                     ; Place signal on top of stack.
52   D2 A5   50   C1   10D0  3715          ADDL3   R0,CDRP$L_BCNT(R5),R2     ; R2 = relative offset of end of buffer
52   00000200 8F   D1   10D5  3716          CMPL    #512,R2                   ; Does user buffer slop onto next pages
                  25   18   10DC  3717          BGEQ    20$                       ; GEQ implies no spill over.
50   00000200 8F   50   C3   10DE  3718          SUBL3   R0,#512,R0                ; R0 = length of 1ST segment to copy.
7E   52   FFFFFE00 8F   CB   10E6  3719          BICL3   #^C<^X1FF>,R2,-(SP)       ; Push length of spill into last page
                        10EE  3720                                            ;  of user buffer.
                  02   12   10EE  3721          BNEQ    5$                        ; NEQ implies there was such slop.
                  8E   D5   10F0  3722          TSTL    (SP)+                     ; If no such slop, clear top of stack.
                        10F2  3723 5$:
```

```
                52    52    F7 8F    78  10F2  3724          ASHL    #-9,R2,R2                       ; R2 now contains one more than the
                                       10F7  3725                                                  ;  number of full (512 byte) pages
                                       10F7  3726                                                  ;  in middle of user buffer that have
                                       10F7  3727                                                  ;  not yet been accounted for. Note
                                       10F7  3728                                                  ;  that the first page (possibly
                                       10F7  3729                                                  ;  partially filled - CDRP$W_BOFF)
                                       10F7  3730                                                  ;  is accounted for in R0, and any
                                       10F7  3731                                                  ;  spill over into the last page is
                                       10F7  3732                                                  ;  accounted for on the top of stack.
                                       10F7  3733 10$:
                      52    D7        10F7  3734          DECL    R2                              ; Reduce # full pages by one.
                      0C    15        10F9  3735          BLEQ    30$                             ; If zero, branch around.
                00000200    8F    DD  10FB  3736          PUSHL   #512                            ; Indicate full 512 byte segment.
                      F4    11        1101  3737          BRB     10$                             ; Branch back.
                                       1103  3738 20$:
                50    D2    A5    D0  1103  3739          MOVL    CDRP$L_BCNT(R5),R0              ; R0 = length of 1St and only segment.
                                       1107  3740 30$:
                                       1107  3741
                                       1107  3742 ; Here R0 has the length of the first segment to copy.  The lengths of the
                                       1107  3743 ; subsequent segments have all been pushed onto the stack in inverse order.
                                       1107  3744 ; The last item pushed onto the top of stack is a zero to indicate no more
                                       1107  3745 ; segments.  R1 is pointing to the first byte of user data.
                                       1107  3746
              52    0CB4    C4    9E  1107  3747          MOVAB   PDT$L_PQ_ALGNPG(R4),R2          ; R2 => Aligned page.
                    18    B5    17  110C  3748          JMP     @CDRP$L_SAVD_RTN(R5)            ; Return to caller leaving data on stack
                                       110F  3749
                                       110F  3750 ;+
                                       110F  3751 ; SETUP_COPY_SEG2 - Routine to setup copy of the second segment of user data
                                       110F  3752 ;                   from (to) the user buffer to (from) the aligned page.
                                       110F  3753 ;
                                       110F  3754 ; Inputs:
                                       110F  3755 ;       PDT$L_PQ_USRPTE = System address of PTE that describes previous page
                                       110F  3756 ;                         of user buffer
                                       110F  3757 ;       PDT$L_PQ_SVPPTE = Slot in System Page Table to map user buffer
                                       110F  3758 ;       PDT$L_PQ_UBFSVA = System Space address that corresponds to this slot
                                       110F  3759 ;
                                       110F  3760 ; Outputs:
                                       110F  3761 ;       PDT$L_PQ_USRPTE = PDT$L_PQ_USRPTE + 4
                                       110F  3762 ;       R1 => Byte zero of page that corresponds to slot defined by
                                       110F  3763 ;             PDT$L_PQ_SVPPTE.
                                       110F  3764 ;
                                       110F  3765 ; Note: R2 must be preserved.
                                       110F  3766 ;-
                                       110F  3767
                                       110F  3768 SETUP_COPY_SEG2:
                                       110F  3769
          0CB0    C4    04    C0  110F  3770          ADDL    #4,PDT$L_PQ_USRPTE(R4)          ; Point to next user PTE.
                15    00    EF  1114  3771          EXTZV   #PTE$V_PFN,#PTE$S_PFN,-         ; Construct a PTE with the proper PFN,
                    0CB0    D4  1117  3772                  @PDT$L_PQ_USRPTE(R4),-          ;  protection=KW, and own=K.  Here we
                    0CAC    D4  111A  3773                  @PDT$L_PQ_SVPPTE(R4)            ;  move the PFN.
                                C8  111D  3774          BISL    #PTE$M_VALID!-                  ; And here we set VALID, the protection
                                    111E  3775                   PTE$C_KW!-                     ;  and the ownership and thereby map
                                    111E  3776                   PTE$C_KOWN,-                   ;  the next page of the user buffer
OCAC    D4    90000000    8F        111E  3777                  @PDT$L_PQ_SVPPTE(R4)            ;  into System Space.
              51    0CA8    C4    D0  1126  3778          MOVL    PDT$L_PQ_UBFSVA(R4),R1          ; R1 => Rest of user data.
                                    112B  3779          INVALID R1                              ; Invalidate virtual address.
                                C5  112E  3780          RSB                                     ; Return to caller.
```

```
                112F  3782          .SBTTL  +       FPC$MAPIRP_BDA, Map a buffer for BDA
                112F  3783
                112F  3784 ;+
                112F  3785 ; Place holder.
                112F  3786 ;-
                112F  3787
                112F  3788 FPC$MAPIRP_BDA::
 05             112F  3789          RSB
```

```
                     1130  3791          .SBTTL  +       FPC$UNMAP,      Release mapping resources
                     1130  3792
                     1130  3793 ;+
                     1130  3794 ; FPC$UNMAP
                     1130  3795 ;
                     1130  3796 ; Inputs:
                     1130  3797 ;       R5 => CDRP
                     1130  3798 ;       CDRP$L_UBARSRCE contains accounting of allocated resources.
                     1130  3799 ;       CDRP$L_LBUFH_AD => local buffer handle.
                     1130  3800 ;
                     1130  3801 ; Outputs:
                     1130  3802 ;       Resources returned
                     1130  3803 ;       CDRP$L_LBUFH_AD cleared
                     1130  3804 ;       R0-R2 destroyed
                     1130  3805 ;       Other registers preserved
                     1130  3806 ;-
                     1130  3807
                     1130  3808 FPC$UNMAP::
                     1130  3809
          53   DD    1130  3810          PUSHL   R3                                       ; Save register.
          55   DD    1132  3811          PUSHL   R5                                       ; Save another.
    50    3F A5  90  1134  3812          MOVB    CDRP$L_UBARSRCE+UBMD$B_DATAPATH(R5),R0   ; R0 contains datapath
                     1138  3813                                                          ;  used in transfer.
55    00DC C4  D0  1138  3814          MOVL    PDT$L_UCB0(R4),R5               ; R5 => UCB.
  53    24 A5  D0  113D  3815          MOVL    UCB$L_CRB(R5),R3                ; R3 => CRB.
  37 A3    50  90  1141  3816          MOVB    R0,CRB$L_INTD+VEC$B_DATAPATH(R3); Set path number for purge.
00000000'GF   16  1145  3817          JSB     G^IOC$PURGDATAP                 ; Purge specified data path.
      37 A3   94  114B  3818          CLRB    CRB$L_INTD+VEC$B_DATAPATH(R3)   ; Clear datapath # from CRB.
          55 8ED0  114E  3819          POPL    R5                                      ; Restore register.
          53 8ED0  1151  3820          POPL    R3                                      ; Restore another.
                     1154  3821
00000000'GF   16  1154  3822          JSB     G^IOC$RELMAPUDA         ; Return allocated map registers.
                     115A  3823
    F042 DF   16  115A  3824          JSB     @RELDATAPATH_TV         ; And also the datapath.
                     115E  3825
      2C A5   D4  115E  3826          CLRL    CDRP$L_LBUFH_AD(R5)     ; Prevent spurious deallocates.
              05  1161  3827          RSB                             ; Return to caller.
                     1162  3828
```

```
                     1162  3830          .SBTTL  +       FPC$UNMAP_UV1,  Release mapping resources
                     1162  3831
                     1162  3832 ;+
                     1162  3833 ; FPC$UNMAP_UV1
                     1162  3834 ;
                     1162  3835 ; Inputs:
                     1162  3836 ;       R5 => CDRP
                     1162  3837 ;       CDRP$L_UBARSRCE contains accounting of allocated resources.
                     1162  3838 ;       CDRP$L_LBUFH_AD => local buffer handle.
                     1162  3839 ;
                     1162  3840 ; Outputs:
                     1162  3841 ;       Resources returned
                     1162  3842 ;       CDRP$L_LBUFH_AD cleared
                     1162  3843 ;       R0-R2 destroyed
                     1162  3844 ;       Other registers preserved
                     1162  3845 ;-
                     1162  3846
                     1162  3847 FPC$UNMAP_UV1::
                     1162  3848
          D0 A5  E8  1162  3849          BLBS    CDRP$W_BOFF(R5),-       ; Branch around if Not word aligned.
             0A      1165  3850                  UNMAP_UNALIGN
                     1166  3851 UNMAP_ODD:                               ; Label to allow branch back.
                     1166  3852
00000000'GF    16    1166  3853          JSB     G^IOC$RELMAPUDA         ; Return allocated map registers.
                     116C  3854
          2C A5  D4  116C  3855          CLRL    CDRP$L_LBUFH_AD(R5)     ; Prevent spurious deallocates.
                 05  116F  3856          RSB                             ; Return to caller.
                     1170  3857
                     1170  3858 UNMAP_UNALIGN:
                     1170  3859
                     1170  3860 ; Here after successfully using the aligned Page, if the user operation
                     1170  3861 ; is a READ, we copy the user's data from the aligned page.  If the user
                     1170  3862 ; operation is a WRITE, we branch around the copy and merely deallocate
                     1170  3863 ; the aligned Page.  The data from a WRITE is copied from the user
                     1170  3864 ; buffer to the aligned Page when the aligned page is allocated.
                     1170  3865
50   00DC C4    D0  1170  3866          MOVL    PDT$L_UCB0(R4),R0       ; R0 => port UCB.
          07    E0  1175  3867          BBS     #PS1_V_OD,-             ; If controller supports odd addresses,
     00AC C0        1177  3868                  UCB$W_PORTSTEP1(R0),-   ;  then branch back in line.
          EB        117A  3869                  UNMAP_ODD
          01    E1  117B  3870          BBC     #IRP$V_FUNC,-           ; If a WRITE, branch around.
       13 CA A5     117D  3871                  CDRP$W_STS(R5),20$
        FF23    30  1180  3872          BSBW    SETUP_COPY_SEG1         ; Call to setup copy of user data.
                    1183  3873                                          ;  This is a complex routine that
                    1183  3874                                          ;  returns: R0 = # bytes in first
                    1183  3875                                          ;  segment, R1 => user data,
                    1183  3876                                          ;  R2 => aligned page, (SP) = # bytes
                    1183  3877                                          ;  in second segment (or zero if none),
                    1183  3878                                          ;  4(SP) (valid only if (SP) non-zero)
                    1183  3879                                          ;  zero.
                    1183  3880 10$:
        81    82    90  1183  3881          MOVB    (R2)+,(R1)+             ; Copy a byte.
          FA 50     F5  1186  3882          SOBGTR  R0,10$                  ; Loop thru entire segment.
             50 8ED0  1189  3883          POPL    R0                      ; Get length of next segment (if any).
             05    13  118C  3884          BEQL    20$                     ; EQL implies no more.
          FF7E    30  118E  3885          BSBW    SETUP_COPY_SEG2         ; Call to setup copy of second segment.
             F0    11  1191  3886          BRB     10$                     ; Branch back to loop.
```

```
                        1193  3887 20$:
      0CA0 C4    D4     1193  3888          CLRL    PDT$L_PQ_POWNER(R4)      ; Release Aligned Page.
        2C A5    D4     1197  3889          CLRL    CDRP$L_LBUFH_AD(R5)      ; Prevent spurious deallocates.
50    0C98 D4    0F     119A  3890          REMQUE  @PDT$L_PQ_PGQFL(R4),R0   ; R0 => Waiting CDRP (if any).
           1B    1D     119F  3891          BVS     30$                      ; VS implies no waiters.
      7E   53    7D     11A1  3892          MOVQ    R3,-(SP)                 ; Save registers before resuming waiter.
           55    DD     11A4  3893          PUSHL   R5
                        11A6  3894
      55   50    D0     11A6  3895          MOVL    R0,R5                    ; R5 => Waiter's CDRP.
  53    10 A5    7D     11A9  3896          MOVQ    CDRP$L_FR3(R5),R3        ; Restore his registers.
         FE9B    30     11AD  3897          BSBW    MAP_UNALIGN              ; Call to allocate and copy data.
                        11B0  3898                                           ;  Note allocation MUST succeed.
00000000'GF      16     11B0  3899          JSB     G^SCS$RESUMEWAITR        ; Resume waiting thread and any backed
                        11B6  3900                                           ;  up IRP's.
                        11B6  3901
           55 8ED0      11B6  3902          POPL    R5                       ; Restore our original registers.
      53   8E   7D     11B9  3903          MOVQ    (SP)+,R3
                        11BC  3904 30$:
                  05    11BC  3905          RSB                              ; Return to caller.
```

```
                        11BD  3907          .SBTTL  +       FPC$UNMAP_BDA,  Release mapping resources BDA
                        11BD  3908
                        11BD  3909 ;+
                        11BD  3910 ; Place holder.
                        11BD  3911 ;-
                        11BD  3912
                        11BD  3913 FPC$UNMAP_BDA::
                  05    11BD  3914          RSB
```

```
                        .SBTTL  INTERNAL SUBROUTINES
                        .SBTTL  +       POLL_CMDRING

                        ;+
                        ; POLL_CMDRING is called to poll the command ring and reclaim any slots (and
                        ; the buffers pointed to by them) that have been released back to the host by
                        ; the port.  POLL_CMDRING makes use of some PDT fields:
                        ;
                        ;       PDT$B_CRINGCNT - which maintains the count of how many as yet
                        ;                        unreclaimed slots the host has sent to the port.
                        ;
                        ;       PDT$B_CPOLLINX - whose low order UDA$K_RINGEXP bits contain the index
                        ;                        of the command ring slot which we should poll next.
                        ;
                        ;       PDT$B_CRCONTENT - an array of UDA$K_RINGSIZE bytes, each of which
                        ;                        maintains the index of the buffer currently pointed
                        ;                        at by the corresponding ring slot.
                        ;
                        ;       PDT$L_BDTABLE - an array of UDA$K_RINGSIZE longwords, each of which
                        ;                        point to the buffer corresponding to the index of the
                        ;                        longword.
                        ;
                        ; POLL_CMDRING polls ring slots to see if the port has released them until
                        ; either of the following two conditions obtain: PDT$B_CRINGCNT goes to zero,
                        ; indicating that all command ring slots sent to the port have been reclaimed;
                        ; or upon polling a slot we come upon one that has NOT been released as yet.
                        ; Since slots are released in sequence, this means that we should cease
                        ; polling.
                        ;
                        ; POLL_CMDRING always polls the slot selected by PDT$B_CPOLLINX.  The low order
                        ; bits of this field are extracted and used as an index into the two arrays
                        ; mentioned above.
                        ;
                        ; A slot is polled by testing its high order bit. A zero bit indicates that
                        ; the slot has been released to the host.  Upon finding a released slot,
                        ; POLL_CMDRING reclaims it and the buffer pointed to it by:
                        ;
                        ;       1. The index of the buffer is obtained from the element of the
                        ;               PDT$B_CRCONTENT array corresponding to the current ring slot.
                        ;
                        ;       2. A pointer to the buffer is obtained from the PDT$L_BDTABLE array.
                        ;
                        ;       3. PDT$B_CPOLLINX is incremented so that the next poll will use the
                        ;               next slot in the command ring.
                        ;
                        ;       4. PDT$B_CRINGCNT is decremented to show one less unreclaimed slot.
                        ;
                        ;       5. If any buffers are queued waiting for available ring slots on
                        ;               PDT$L_PU_SNDQFL, the first one is removed from the Q and
                        ;               inserted into the command ring by calling internal subroutine
                        ;               INSERT_IN_CRING.
                        ;
                        ;       6. The now free buffer is put onto the response ring or the free Q,
                        ;               whichever is appropriate, by calling internal subroutine,
                        ;               Q_DEALLOC_BUF.
                        ;
                        ;       7. Finally we branch back to the beginning of POLL_CMDRING to poll
```

```
                     11BE  3973 ;                again.
                     11BE  3974 ;
                     11BE  3975 ; Inputs:
                     11BE  3976 ;
                     11BE  3977 ;       R4                      -Addr of PDT
                     11BE  3978 ;
                     11BE  3979 ; Outputs:
                     11BE  3980 ;
                     11BE  3981 ;       R0-R2                   -destroyed
                     11BE  3982 ;       Other registers         -preserved
                     11BE  3983 ;-
                     11BE  3984
                     11BE  3985 POLL_CMDRING:
                     11BE  3986
      0122 C4   95   11BE  3987         TSTB    PDT$B_CRINGCNT(R4)      ; See if any slots in use on command
            36   13   11C2  3988         BEQL    20$                     ;  ring. EQL implies NO.
                     11C4  3989
      04   00   EF   11C4  3990         EXTZV   #0,#UDA$K_RINGEXP,-     ; Extract ring index of slot of
52    0121 C4        11C7  3991                 PDT$B_CPOLLINX(R4),R2   ;  where to begin polling.
                     11CB  3992
    0248 C442   D5   11CB  3993         TSTL    PDT$L_CMDRING(R4)[R2]   ; Has controller released this slot?
            28   19   11D0  3994         BLSS    20$                     ; LSS implies NO.
                     11D2  3995
52  012C C442   9A   11D2  3996         MOVZBL  PDT$B_CRCONTENT(R4)[R2],R2      ; R2 = index of buffer pointed
                     11D8  3997                                                 ;  at by this slot.
52  014C C442   D0   11D8  3998         MOVL    PDT$L_BDTABLE(R4)[R2],R2        ; R2 => buffer header.
                     11DE  3999
          52   DD   11DE  4000         PUSHL   R2                              ; Remember R2 => free buffer.
                     11E0  4001
      0121 C4   96   11E0  4002         INCB    PDT$B_CPOLLINX(R4)      ; Bump polling index.
      0122 C4   97   11E4  4003         DECB    PDT$B_CRINGCNT(R4)      ; One less used command ring slot.
                     11E8  4004
52    0110 D4   0F   11E8  4005         REMQUE  @PDT$L_PU_SNDQFL(R4),R2 ; R2 => buffer (if any) to be inserted
            03   1D   11ED  4006         BVS     10$                     ;  in command ring. VS implies NONE.
          FDAC   30   11EF  4007         BSBW    INSERT_IN_CRING         ; Call to insert R2 => buffer in ring.
                     11F2  4008 10$:
          52 8ED0   11F2  4009         POPL    R2                      ; Restore R2 => free buffer.
          FCA5   30   11F5  4010         BSBW    Q_DEALLOC_BUF           ; Call to put free buffer onto response
                     11F8  4011                                         ;  ring or onto free Q, whichever is
                     11F8  4012                                         ;  appropriate.
            C4   11   11F8  4013         BRB     POLL_CMDRING            ; Branch back to reclaim more buffers.
                     11FA  4014 20$:
                 05  11FA  4015         RSB                             ; Return to caller.
```

```
                     11FB  4017          .SBTTL  -       STATE_ERR,      RETURN CDT STATE ERROR
                     11FB  4018          .SBTTL  -                       TO SYSAP
                     11FB  4019
                     11FB  4020 ;+
                     11FB  4021 ; Set error status code and return to caller.
                     11FB  4022 ;-
                     11FB  4023
                     11FB  4024 STATE_ERR:
                     11FB  4025
50   2154 8F   3C    11FB  4026          MOVZWL  #SS$_ILLCDTST,R0                ; Status = illegal CDT state
                 05  1200  4027          RSB                             ; Return to SYSAP
                     1201  4028
                     1201  4029
```

```
                        1201          .SBTTL  INTERRUPT SERVICING
                        1201          .SBTTL  +       PU$INT - Interrupt service routine
                        1201   ;
                        1201   ; Inputs:
                        1201   ;      00(SP) - Pointer to IDB
                        1201   ;      04(SP)  - SAVED R0
                        1201   ;      08(SP)  - SAVED R1
                        1201   ;      12(SP)  - SAVED R2
                        1201   ;      16(SP)  - SAVED R3
                        1201   ;      20(SP)  - SAVED R4
                        1201   ;      24(SP)  - SAVED R5
                        1201   ;      28(SP)  - PC AT THE TIME OF THE INTERRUPT
                        1201   ;      32(SP)  - PSL AT THE TIME OF THE INTERRUPT
                        1201   ;
                        1201
                        1201   PU$INT::                                                  ;
        53   9E    D0   1201          MOVL    @(SP)+,R3                  ; Get address of IDB
     55   04 A3    D0   1204          MOVL    IDB$L_OWNER(R3),R5         ; Get address of UCB
   54   0084 C5    D0   1208          MOVL    UCB$L_PDT(R5),R4           ; Get address of PDT
        53   63    D0   120D          MOVL    IDB$L_CSR(R3),R3           ; R3 => UDA CSR.
          02 A3    B0   1210          MOVW    UDASA(R3),-
        00AA C5         1213                  UCB$W_UDASA(R5)            ; Save error status in UCB.
                        1216
   50   0203 C4    90   1216          MOVB    PDT$B_PURGEDP(R4),R0             ; Fetch Purge data path number
             28    13   121B          BEQL    10$                              ; Br if none
        0203 C4    94   121D          CLRB    PDT$B_PURGEDP(R4)                ; Clear it
     51   24 A5    D0   1221          MOVL    UCB$L_CRB(R5),R1                 ; R1 => CRB to setup for purge
     52   37 A1    9A   1225          MOVZBL  CRB$L_INTD+VEC$B_DATAPATH(R1),R2; Save datapath #, in case we
                        1229                                                   ;  interrupted purge in UNMAP.
        7E   51    7D   1229          MOVQ    R1,-(SP)                         ; Push R1=>CRB, R2=datapath #.
     37 A1   50    90   122C          MOVB    R0,CRB$L_INTD+VEC$B_DATAPATH(R1); Set path number for purge
   00000000'GF     16   1230          JSB     G^IOC$PURGDATAP                  ; Purge specified data path
   53   0100 C4    D0   1236          MOVL    PDT$L_PU_CSR(R4),R3              ; Refresh R3=>CSR after purge
          02 A3    B4   123B          CLRW    UDASA(R3)                        ; Acknowledge purge interrupt
        51   8E    7D   123E          MOVQ    (SP)+,R1                         ; Pop R1=>CRB, R2=Datapath #.
     37 A1   52    90   1241          MOVB    R2,CRB$L_INTD+VEC$B_DATAPATH(R1); Restore old contents of
                        1245                                                   ;  datapath # to CRB.
                        1245   10$:
             01    E5   1245          BBCC    #UCB$V_INT,-               ; See if we are awaiting interrupt. If
       09 64 A5         1247                  UCB$W_STS(R5),20$          ;  so then we are in hardware init. If
                        124A                                             ;  not, branch around.
     53   10 A5    7D   124A          MOVQ    UCB$L_FR3(R5),R3           ; Restore initialization thread
          0C B5    16   124E          JSB     @UCB$L_FPC(R5)             ;  context.
             07    11   1251          BRB     30$                        ; And branch around to dismiss interrupt.
                        1253   20$:
                        1253
             01    E2   1253          BBSS    #UCB$V_PU_FRKBSY,-         ; Appropriate UCB fork block, or else
       02 68 A5         1255                  UCB$W_DEVSTS(R5),30$       ;  branch to dismiss interrupt
             03    10   1258          BSBB    POLL_RSPRING               ; If we were successful in appropriating
                        125A                                             ;  the fork block, BSBB to poll the
                        125A                                             ;  response ring.
                        125A   30$:
             3F    BA   125A          POPR    #^M<R0,R1,R2,R3,R4,R5>     ; Restore registers.
                   02   125C          REI                                ; And dismiss interrupt.
```

```
                        125D  4086          .SBTTL  +       POLL_RSPRING
                        125D  4087
                        125D  4088 ;+
                        125D  4089 ; POLL_RSPRING is called from the interrupt service routine at device IPL and
                        125D  4090 ; with the UCB fork allocated to this thread (i.e. UCB$M_PU_FRKBSY bit on in
                        125D  4091 ; UCB$W_DEVSTS).  POLL_RSPRING first IOFORK's on the UCB so that the interrupt
                        125D  4092 ; can be dismissed and then after resuming execution at fork IPL, it frees up
                        125D  4093 ; the UCB fork block (i.e. clears UCB$M_PU_FRKBSY) and tests whether a power
                        125D  4094 ; failure has occurred recently.  If so, then we merely branch out of the flow
                        125D  4095 ; here to begin a thread at POST_POWER_FORK, that causes all CONNECTION's to
                        125D  4096 ; resynchronize.
                        125D  4097 ;
                        125D  4098 ; If we remain here (normal case) we traverse the response ring looking for
                        125D  4099 ; buffers that have been released to the host (us), and upon finding one
                        125D  4100 ; we determine whether it is a SEQUENCED MESSAGE or a DATAGRAM and we call
                        125D  4101 ; the appropriate entrypoint in the SYSAP for the CONNECTION over which the
                        125D  4102 ; message was received.
                        125D  4103 ;
                        125D  4104 ; Inputs:
                        125D  4105 ;
                        125D  4106 ;       R4                              -Addr of PDT
                        125D  4107 ;       R5                              -Addr of UCB
                        125D  4108 ;
                        125D  4109 ;       UCB fork block allocated to this thread
                        125D  4110 ;
                        125D  4111 ; Outputs:
                        125D  4112 ;
                        125D  4113 ;       Response ring polled and SYSAPs called
                        125D  4114 ;-
                        125D  4115          .enabl  lsb
                        125D  4116 POLL_RSPRING:
                        125D  4117
                        125D  4118          IOFORK                              ; Lower IPL.
              02    AA  1263  4119          BICW    #UCB$M_PU_FRKBSY,-
                68  A5  1265  4120                  UCB$W_DEVSTS(R5)            ; Allow fork block to be re-used.
            00AA  C5  B5  1267  4121          TSTW    UCB$W_UDASA(R5)             ; See if we got a fatal error.
              0C  19  126B  4122          BLSS    5$                          ; LSS means fatal error.
              04  E0  126D  4123          BBS     #UCB$V_PU_MRESET,-          ; If set, then MRESET request received
           12 68  A5  126F  4124                  UCB$W_DEVSTS(R5),8$         ;  while we were forked, so go to do it.
              05  E1  1272  4125          BBC     #UCB$V_POWER,-              ; See if POWERFAIL occurred while
           10 64  A5  1274  4126                  UCB$W_STS(R5),10$           ;  UCB fork block was busy. Branch if not.
              0B  11  1277  4127          BRB     8$                          ; Branch around if YES powerfail.
                      1279  4128 5$:
              03  B0  1279  4129          MOVW    #UDASA_ATTNCODE,-           ; Indicate what kind of error log record
            00A8  C5  127B  4130                  UCB$W_ATTNCODE(R5)          ;  we are about to create.
      00000000'GF  16  127E  4131          JSB     G^ERL$DEVICEATTN            ; Call to create error log record of INIT.
                      1284  4132 8$:
             F03C  31  1284  4133          BRW     POST_POWER_FORK             ; If POWERFAIL, get out of here.
                      1287  4134 10$:
03 0204  C4   00  E5  1287  4135          BBCC    #0,PDT$W_CMDINT(R4),20$     ; Branch if command ring NOT UNfull.
             FF2E  30  128D  4136          BSBW    POLL_CMDRING                ; Reclaim free space in command ring.
                      1290  4137 20$:
            0206  C4  B4  1290  4138          CLRW    PDT$W_RSPINT(R4)            ; Always clear response interrupt
                      1294  4139                                              ;  indication since we always poll
                      1294  4140                                              ;  response ring on interrupt.
                      1294  4141 30$:
            0125  C4  95  1294  4142          TSTB    PDT$B_RRINGCNT(R4)          ; See if response ring has anything to
```

```
                01   12  1298       BNEQ    50$                         ;  poll.  NEQ implies YES.
                         129A  40$:
                     05  129A       RSB                                 ; If no more, kill this thread.
                         129B  50$:
         04     00   EF  129B       EXTZV   #0,#UDA$K_RINGEXP,-         ; Extract index mod sing size leaving
    50     0124 C4       129E               PDT$B_RPOLLINX(R4),R0       ;  R0 = index of slot to poll.
        0208 C440    D5  12A2       TSTL    PDT$L_RSPRING(R4)[R0]       ; See if slot released to host.
                F1   19  12A7       BLSS    40$                         ; LSS => slot still owned by controller.
                         12A9
           0124 C4   96  12A9       INCB    PDT$B_RPOLLINX(R4)          ; Bump response poll index.
           0125 C4   97  12AD       DECB    PDT$B_RRINGCNT(R4)          ; Decr # slots passed to controller.
  50    013C C440    9A  12B1       MOVZBL  PDT$B_RRCONTENT(R4)[R0],R0; R0 = index of buffer with response.
  53    014C C440    D0  12B7       MOVL    PDT$L_BDTABLE(R4)[R0],R3  ; R3 => buffer with response.
                         12BD
                         12BD  ENABLE_RESPONSE_START:
    52     0108 D4   0F  12BD       REMQUE  @PDT$L_PU_FQFL(R4),R2       ; R2 => free buffer. NOTE this instruction
                         12C2                                           ;  is copied below in the flow that gets
                         12C2                                           ;  executed if tracing is enabled.  Any
                         12C2                                           ;  edit to this instruction should also
                         12C2                                           ;  done to its copy.
                         12C2  ENABLE_RESPONSE_END:
                0E   1C  12C2       BVC     55$                         ; VC implies R2 => buffer.
              FEF7   30  12C4       BSBW    POLL_CMDRING                ; If no buffers, reclaim some.
                10   91  12C7       CMPB    #UDA$K_RINGSIZE,-           ; For debugging, assure that we indeed
           0125 C4       12C9               PDT$B_RRINGCNT(R4)          ;  have a full response ring.
                07   13  12CC       BEQL    57$                         ; EQL implies full response ring.
                         12CE       BUG_CHECK       UDAPORT,FATAL       ; Else bug check for now.
                         12D2  55$:
              FBCF   30  12D2       BSBW    INSERT_IN_RRING             ; Else insert free buffer in response
                         12D5                                           ;  ring.
                         12D5
           50     53  D0  12D5  57$:  MOVL    R3,R0                     ; R0 => buffer with response.
                         12D8
                         12D8  ;    MOVZBL  UDAB$B_CONID(R0),R3         ; R3 = connection ID of message.
                         12D8  ; Here we assume that VC0 is DISK MSCP, VC1 is TAPE MSCP, VC2 is DUP and
                         12D8  ;    VC255 (low order bits both on) so that VC255 maps to VC3
         02     00   EF  12D8       EXTZV   #0,#2,-                     ; Low order 2 bits select connection.
    53     13 A0         12DB               UDAB$B_CONID(R0),R3         ;  R3 = connection ID of message.
  53    00E4 C443    D0  12DE       MOVL    PDT$L_PU_CDTARY(R4)[R3],R3; R3 => CDT.
                         12E4
                00   EF  12E4       EXTZV   #UDAB$V_CREDITS,-           ; Extract credits returned by message.
                04       12E6               #UDAB$S_CREDITS,-
    51     12 A0         12E7               UDAB$B_CREDTYPE(R0),R1
    40 A3     51     A0  12EA       ADDW    R1,CDT$W_SEND(R3)           ; Add in new credits.
                         12EE
    52     14 A0     9E  12EE       MOVAB   UDAB$T_TEXT(R0),R2          ; R2 => application area of message.
    51     10 A0     3C  12F2       MOVZWL  UDAB$W_MSG_LEN(R0),R1       ; Pickup length of datagram or message.
       7E     53     7D  12F6       MOVQ    R3,-(SP)                    ; Save R3=>CDT and R4=>PDT before
                         12F9                                           ;  dispatching.
                04   EF  12F9       EXTZV   #UDAB$V_MSGTYPE,-           ; Extract type of message.
                04       12FB               #UDAB$S_MSGTYPE,-
    55     12 A0         12FC               UDAB$B_CREDTYPE(R0),R5
                         12FF       ASSUME  UDA$K_SEQMSGTYP EQ      0
                23   12  12FF       BNEQ    90$                         ; NEQ means NOT sequenced message.
             00 B3   16  1301       JSB     @CDT$L_MSGINPUT(R3)         ; Call sequenced message handler
                         1304                                           ;  passing R2 => message text.
                         1304  85$:
```

```
                     1304  4200                MOVQ    (SP)+,R3                ; Restore R3=>CDT and R4=>PDT after
                     1307  4201                                                 ;  dispatching.
                     1307  4202 87$:
      40 A3    B5    1307  4203                TSTW    CDT$W_SEND(R3)           ; See if we are positive here.
         15    15    130A  4204                BLEQ    89$                      ; LEQ means no more credits left.
                     130C  4205                $RESUME_FP      -                ; Resume anyone waiting for credits
                     130C  4206                        @CDT$L_CRWAITQFL(R3),-
                     130C  4207                        QEMPTY=89$               ; Where to go if no waiters.
            E6    11  131F  4208                BRB     87$                      ; Go back and try to resume more.
                     1321  4209 89$:
          FF70    31  1321  4210                BRW     30$                      ; Go back to test for more responses.
                     1324  4211 90$:
      55    01    D1  1324  4212                CMPL    #UDA$K_DGTYPE,R5         ; See if Datagram message.
            05    12  1327  4213                BNEQ    100$                     ; NEQ means test for something else.
        04 B3    16  1329  4214                JSB     @CDT$L_DGINPUT(R3)       ; Call datagram handler passing
                     132C  4215                                                 ;  R2 => datagram, with R1 = length.
            D6    11  132C  4216                BRB     85$                      ; Go back to test for more responses.
                     132E  4217 100$:
      55    02    D1  132E  4218                CMPL    #UDA$K_CREDTYPE,R5       ; See if CREDIT message.
            05    12  1331  4219                BNEQ    110$                     ; NEQ means test for something else.
         EDA6    30  1333  4220                BSBW    NULL_MSG_INPUT           ; Call Null Message handler to dispose
                     1336  4221                                                 ;  of and recycle buffer.
            CC    11  1336  4222                BRB     85$                      ; Go back to test for more responses.
                     1338  4223 110$:
      55    0F    D1  1338  4224                CMPL    #UDA$K_MAINTTYPE,R5      ; See if MAINTENANCE message.
            0F    12  133B  4225                BNEQ    120$                     ; NEQ means test for something else.
            00    EF  133D  4226                EXTZV   #UDAB$V_CREDITS,-        ; The Credit Field of Maintenance
            04        133F  4227                        #UDAB$S_CREDITS,-        ;  messages  is to be ignored. So we
   50    12 A0        1340  4228                        UDAB$B_CREDTYPE(R0),R0   ;  again extract credits field and
40 A3    50    A2  1343  4229                SUBW    R0,CDT$W_SEND(R3)        ; Subtract out credits added in above.
                     1347  4230
         ED96    30  1347  4231                BSBW    NULL_DG_INPUT            ; Call Null Datagram handler to log
                     134A  4232                                                 ;  message and then recycle buffer.
            B8    11  134A  4233                BRB     85$                      ; Go back to test for more responses.
                     134C  4234 120$:
                     134C  4235                BUG_CHECK       UDAPORT,FATAL
                     1350  4236
                     1350  4237 ENABLE_RESPONSE_CODE:                          ; If we enable tracing, the driver will
                     1350  4238                                                 ;  dynamically patch location
                     1350  4239                                                 ;  ENABLE_RESPONSE_START to BRW here.
         F12B    30  1350  4240                BSBW    TRACE_RESPONSE           ; Copy response buffer to trace table.
                     1353  4241
52    0108 D4    0F  1353  4242                REMQUE  @PDT$L_PU_FQFL(R4),R2    ; R2 => free buffer.
            FF67    31  1358  4243                BRW     ENABLE_RESPONSE_END      ; Branch back to normal stream.
                     135B  4244
                 00000090  135B  4245 ENABLE_RESPONSE_OFFSET=ENABLE_RESPONSE_CODE-ENABLE_RESPONSE_START-3
                     135B  4246                .dsabl  lsb
```

```
                     135B  4248          .SBTTL  +       PU$SA_POLL
                     135B  4249
                     135B  4250 ;+
                     135B  4251 ; Routine periodically called by CRB wakeup mechanism to see if the SA register
                     135B  4252 ; indicates that this port has had an uncorrectable error.  If so the port is
                     135B  4253 ;  re-initialized in order to bring it back.
                     135B  4254 ;
                     135B  4255 ; Inputs:
                     135B  4256 ;       R3 => CRB.
                     135B  4257 ;
                     135B  4258 ; Outputs:
                     135B  4259 ;       All registers, R0-R5 are modified.
                     135B  4260
                     135B  4261 PU$SA_POLL:
                     135B  4262
  54   10 A3   D0    135B  4263          MOVL    CRB$L_AUXSTRUC(R3),R4   ; R4 => PDT.
55   00DC C4   D0    135F  4264          MOVL    PDT$L_UCB0(R4),R5       ; R5 => UCB.
50   0100 C4   D0    1364  4265          MOVL    PDT$L_PU_CSR(R4),R0     ; R0 => Port CSR.
       02 A0   B0    1369  4266          MOVW    UDASA(R0),-             ; Retrieve SA register
     00AA C5         136C  4267                  UCB$W_UDASA(R5)         ;  and check for error.
          0F   19    136F  4268          BLSS    10$                     ; LSS means YES, error.
                     1371  4269
          0F   C1    1371  4270          ADDL3   #SA_POLL_INTVAL,-       ; Here we had no error, so we simply
 00000000'GF         1373  4271                  G^EXE$GL_ABSTIM,-       ;  re-establish SA polling interval.
       18 A3         1378  4272                  CRB$L_DUETIME(R3)
       DE AF   9E    137A  4273          MOVAB   PU$SA_POLL,-            ; And re-establish wakeup routine.
       1C A3         137D  4274                  CRB$L_TOUTROUT(R3)
                     137F  4275          RSB                             ; And simply return to caller.
                     1380  4276 10$:                                    ; Here we had an SA error.
                     1380  4277          SETIPL  #IPL$_SCS               ; Lower IPL for processing.
          03   B0    1383  4278          MOVW    #UDASA_ATTNCODE,-       ; Indicate what kind of errorlog
     00A8 C5         1385  4279                  UCB$W_ATTNCODE(R5)      ;  record we are about to create.
00000000'GF   16    1388  4280          JSB     G^ERL$DEVICEATTN        ; Call to create error log record of SA
                     138E  4281                                          ;  error.
     00AA C5   B4    138E  4282          CLRW    UCB$W_UDASA(R5)         ; Clear so that we do not report error
                     1392  4283                                          ;  redundantly.
        EF2E   31    1392  4284          BRW     POST_POWER_FORK         ; Branch to begin re-init of port.
                     1395  4285
                     1395  4286 PU$END:
                     1395  4287          .END
```

```
$$$                             =  00000020 R      02
$$$ENTRYNUM                     =  0000001E
$$$PREV                         =  00000080
$$BASE                          =  00000001
$$DISPL                         =  00000008
$$GENSW                         =  00000001
$$HIGH                          =  00000007
$$LIMIT                         =  00000006
$$LOW                           =  00000001
$$MNSW                          =  00000001
$$MXSW                          =  00000001
$$OP                            =  00000002
ADP$L_CSR                       =  00000000
ADP$L_VECTOR                    =  00000010
ADP$W_ADPTYPE                   =  0000000E
ADP$W_TR                        =  0000000C
AFTER_RINGBASE                     00000787 R      03
ALLOC_DELTA                     =  00000001
ALLOC_PDT_8SS                      0000031D R      03
ALLOC_PDT_NOTUV1                   0000032F R      03
ALLOC_PDT_UV1                      0000033D R      03
ALLOC_POOL                         00000C8C R      03
AT$_UBA                         =  00000001
BRW_OPCODE                      =  00000031
BUG$_UDAPORT                       ********   X    03
BUG$_UNSUPRTCPU                    ********   X    03
BUILD_PB_SB                        00000BBF R      03
BUILD_PDT                          00000303 R      03
CDRP$L_BCNT                     =  FFFFFFD2
CDRP$L_CDT                      =  00000024
CDRP$L_FPC                      =  0000000C
CDRP$L_FQFL                     =  00000000
CDRP$L_FR3                      =  00000010
CDRP$L_LBUFH_AD                 =  0000002C
CDRP$L_MSG_BUF                  =  0000001C
CDRP$L_RWCPTR                   =  00000028
CDRP$L_SAVD_RTN                 =  00000018
CDRP$L_SVAPTE                   =  FFFFFFCC
CDRP$L_UBARSRCE                 =  0000003C
CDRP$W_BOFF                     =  FFFFFFD0
CDRP$W_STS                      =  FFFFFFCA
CDT$C_CLOSED                    =  00000000
CDT$C_OPEN                      =  00000002
CDT$K_LENGTH                    =  000000A0
CDT$L_CRWAITQBL                 =  0000003C
CDT$L_CRWAITQFL                 =  00000038
CDT$L_DGINPUT                   =  00000004
CDT$L_ERRADDR                   =  0000000C
CDT$L_MSGINPUT                  =  00000000
CDT$L_RCONID                    =  00000014
CDT$L_RPROCNAM                  =  00000050
CDT$W_REASON                    =  00000026
CDT$W_SEND                      =  00000040
CDT$W_STATE                     =  00000028
CNTRLTYP_ARRAY                     000007C3 R      03
COMAREA_BOFF                    =  00000000
COMPLETE_IO                        00000085 R      03
COM_INIT_UDA_BUFS                  00000512 R      03
CONREJ                             00000D6F R      03
CONVC0                             00000DA5 R      03
CONVC1                             00000D96 R      03
CONVC2                             00000D87 R      03
CON_COMMON                         00000DB1 R      03
CON_NO_LISTEN                      00000D7B R      03
CON_NO_NODE                        00000D63 R      03
CRB$L_AUXSTRUC                  =  00000010
CRB$L_DUETIME                   =  00000018
CRB$L_INTD                      =  00000024
CRB$L_TOUTROUT                  =  0000001C
CRING_FULL                         00000FC8 R      03
DATAPATH_730                       00000575 R      03
DATAPATH_750                       00000565 R      03
DATAPATH_780                       00000583 R      03
DATAPATH_790                       00000583 R      03
DATAPATH_8SS                       00000555 R      03
DC$_BUS                         =  00000080
DDB$L_DDT                       =  0000000C
DDB$T_NAME                      =  00000014
DEV$M_AVL                          ********   X    02
DEV$M_ELG                          ********   X    02
DEV$M_IDV                          ********   X    02
DEV$M_ODV                          ********   X    02
DEV$M_SHR                          ********   X    02
DEVTYPE_ARRAY                      000007DD R      03
DISK_CONID                      =  00000000
DPT$C_LENGTH                    =  00000038
DPT$C_VERSION                   =  00000004
DPT$INITAB                         00000038 R      02
DPT$M_NOUNLOAD                  =  00000004
DPT$M_SCS                       =  00000008
DPT$M_SVP                       =  00000002
DPT$REINITAB                       00000047 R      02
DPT$TAB                            00000000 R      02
DT$_LESI                        =  00000005
DT$_QDA50                       =  0000000B
DT$_RDRX                        =  00000007
DT$_TK50P                       =  00000008
DT$_TU81P                       =  00000006
DT$_UDA50                       =  00000003
DT$_UDA50A                      =  00000004
DT$_UQPORT                      =  00000003
DUP_CONID                       =  00000002
DYN$C_CRB                       =  00000005
DYN$C_DDB                       =  00000006
DYN$C_DPT                       =  0000001E
DYN$C_SCS                       =  00000060
DYN$C_SCS_PB                    =  00000004
DYN$C_SCS_PDT                   =  00000005
DYN$C_SCS_SB                    =  00000007
DYN$C_UCB                       =  00000010
EMB$C_UM                        =  00000004
EMB$L_DV_REGSAV                 =  0000004E
ENABLE_COMMAND_CODE                00000FD1 R      03
ENABLE_COMMAND_END                 00000FB7 R      03
```

```
ENABLE_COMMAND_OFFSET           =  0000001E
ENABLE_COMMAND_START               00000FB0 R      03
ENABLE_RESPONSE_CODE               00001350 R      03
ENABLE_RESPONSE_END                000012C2 R      03
ENABLE_RESPONSE_OFFSET          =  00000090
ENABLE_RESPONSE_START              000012BD R      03
ERL$DEVICEATTN                     ********   X    03
ERL$LOGMESSAGE                     ********   X    03
EXE$ALONONPAGED                    ********   X    03
EXE$ALOPHYCNTG                     ********   X    03
EXE$GB_CPUTYPE                     ********   X    03
EXE$GL_ABSTIM                      ********   X    03
EXE$GL_TENUSEC                     ********   X    03
EXE$GL_UBDELAY                     ********   X    03
EXE$GQ_SYSTIME                     ********   X    03
EXE$IOFORK                         ********   X    03
EXE$SNDEVMSG                       ********   X    03
EXE$ZEROPARM                       ********   X    03
FAIL_ATTNCODE                   =  00000002
FPC$ACCEPT                         00000CFB RG     03
FPC$ALLOCDG                        00000CFB RG     03
FPC$ALLOCMSG                       00000E2E RG     03
FPC$CONNECT                        00000D21 RG     03
FPC$DCONNECT                       00000DF1 RG     03
FPC$DEALLOCDG                      00000CFB RG     03
FPC$DEALLOMSG                      00000E95 RG     03
FPC$DEALRGMSG                      00000E90 RG     03
FPC$MAINTFCN                       00000CFB RG     03
FPC$MAP                            00000CFB RG     03
FPC$MAPBYPASS                      00000CFB RG     03
FPC$MAPIRP                         00000FDE RG     03
FPC$MAPIRPBYP                      00000CFB RG     03
FPC$MAPIRP_BDA                     0000112F RG     03
FPC$MAPIRP_UV1                     00001014 RG     03
FPC$MRESET                         00000CFF RG     03
FPC$MSTART                         00000D0B RG     03
FPC$QUEUEDG                        00000E9D RG     03
FPC$QUEUEMDGS                      00000CFB RG     03
FPC$RCHMSGBUF                      00000EF3 RG     03
FPC$RCLMSGBUF                      00000F32 RG     03
FPC$READCOUNT                      00000CFB RG     03
FPC$REJECT                         00000CFB RG     03
FPC$REQDATA                        00000CFB RG     03
FPC$RLSCOUNT                       00000CFB RG     03
FPC$SENDDATA                       00000CFB RG     03
FPC$SENDDG                         00000CFB RG     03
FPC$SENDMSG                        00000CFB RG     03
FPC$SENDRGDG                       00000CFB RG     03
FPC$SNDCNTMSG                      00000F71 RG     03
FPC$STOP_VCS                       00000CFB RG     03
FPC$UNMAP                          00001130 RG     03
FPC$UNMAP_BDA                      000011BD RG     03
FPC$UNMAP_UV1                      00001162 RG     03
FUNCTAB_LEN                     =  0000001C
GOTO_HARDP                         000009C9 R      03
HARDPOWER                          00000AAF R      03
HARDWARE_INIT                      00000811 R      03
HARD_RETRY                         00000AB2 R      03
HS1_M_BIT15                     =  00008000
HS1_M_IE                        =  00000080
HS1_V_CRNGLEN                   =  0000000B
HS1_V_RRNGLEN                   =  00000008
HS2_M_PI                        =  00000001
HS4_M_GO                        =  00000001
HS4_M_LF                        =  00000002
HS4_V_BURST                     =  00000002
IDB$L_ADP                       =  00000014
IDB$L_CSR                       =  00000000
IDB$L_OWNER                     =  00000004
ILLIOFUNC                          00000080 R      03
INI$BRK                            ********   X    03
INIT_ATTNCODE                   =  00000001
INIT_BDA_BUFS                      000006CF R      03
INIT_CTLR                          0000022D RG     03
INIT_DELTA                      =  0000000A
INIT_INIT_STEPS                    00000701 R      03
INIT_PU_PDT                        00000B28 R      03
INIT_UDA_BUFFERS                   000004E5 R      03
INIT_UDA_BUFS_8SS                  00000504 R      03
INIT_UDA_BUFS_RTN                  000006D3 R      03
INIT_UDA_BUFS_UV1                  000005E2 R      03
INSERT_IN_CRING                    00000F9E R      03
INSERT_IN_FREEQ                    00000ECE R      03
INSERT_IN_RRING                    00000EA4 R      03
INSERT_RTN                         00000ECD R      03
INTR_VEC                        =  000000B8
IO$_INITIALIZE                  =  00000004
IO$_STOP                        =  00000003
IO$_VIRTUAL                     =  0000003F
IOC$ALLOSPT                        ********   X    03
IOC$ALOUBMAPRM                     ********   X    03
IOC$LOADUBAMAP                     ********   X    03
IOC$LUBAUDAMAP                     ********   X    03
IOC$MNTVER                         ********   X    03
IOC$PURGDATAP                      ********   X    03
IOC$RELDATAPUDA                    ********   X    03
IOC$RELMAPUDA                      ********   X    03
IOC$REQCOM                         ********   X    03
IOC$REQDATAP                       ********   X    03
IOC$REQDATAPUDA                    ********   X    03
IOC$REQMAPUDA                      ********   X    03
IOC$REQPCHANL                      ********   X    03
IOC$RETURN                         ********   X    03
IOC$THREADCRB                      ********   X    03
IOC$WFIKPCH                        ********   X    03
IPL$_SCS                        =  00000008
IRP$$_FCODE                     =  00000006
IRP$V_FCODE                     =  00000000
IRP$V_FUNC                      =  00000001
IRP$W_FUNC                      =  00000020
LESI_CNTRLTYP                   =  00000001
LOADUBA_8SS                        00000874 R      03
LOADUBA_COMMON                     00000883 R      03
LOOP_LIMIT                      =  00000FAB
```

```
MAP$M_MAPREGS                   = 00000001
MAP$V_MAPREGS                   = 00000000
MAP_ODD                           0000101B R      03
MAP_UNALIGN                       0000104B R      03
MASKH                           = 00000000
MASKL                           = 00000018
MAYA_CNTRLTYP                   = 00000003
MMG$GL_SPTBASE                    ********    X   03
MSG$_RC25MVER                   = 0000005D
MSG$_RDRXMVER                   = 0000005E
MSG$_TU81MVER                   = 0000005F
MSG$_UDA50MVER                  = 00000057
NOLOADUBA_UV1                     0000088F R      03
NO_CONSEC_INITS                 = 00000005
NO_OVERLAYMAP                     000003F0 R      03
NO_PHYCONTIGBYT                 = 00000200
NO_PHYCONTIGPGS                 = 00000001
NULL_CDT                          000000FC R      03
NULL_DG_INPUT                     000000E0 R      03
NULL_ERR_ROUT                     000000F9 R      03
NULL_MSG_INPUT                    000000DC R      03
NULL_ROUTINE                      000000F9 R      03
NUMUBAVEC                       = 00000080
OPC_MSG_ARRAY                     00000804 R      03
OVERLAYMAP                        000003E4 R      03
OVERLAYMAP_8SS                    000003CB R      03
PB$B_SUBTYP                     = 0000000B
PB$B_TYPE                       = 0000000A
PB$C_OPEN                       = 00000003
PB$K_LENGTH                     = 00000054
PB$L_BLINK                      = 00000004
PB$L_FLINK                      = 00000000
PB$L_PDT                        = 0000002C
PB$L_RPORT_REV                  = 00000018
PB$L_RPORT_TYP                  = 00000014
PB$L_SBLINK                     = 00000030
PB$L_WAITQBL                    = 0000003C
PB$L_WAITQFL                    = 00000038
PB$T_LPORT_NAME                 = 00000024
PB$W_SIZE                       = 00000008
PB$W_STATE                      = 00000012
PB_ALLOC_FAIL                     00000C68 R      03
PDT$B_BDPUSECNT                   0000012A
PDT$B_CONBITMAP                   00000127
PDT$B_CPOLLINX                    00000121
PDT$B_CRCONTENT                   0000012C
PDT$B_CRINGCNT                    00000122
PDT$B_CRINGINX                    00000120
PDT$B_DATAPATH                    00000129
PDT$B_NOCURCON                    00000126
PDT$B_PDT_TYPE                  = 00000007
PDT$B_PURGEDP                     00000203
PDT$B_RPOLLINX                    00000124
PDT$B_RRCONTENT                   0000013C
PDT$B_RRINGCNT                    00000125
PDT$B_RRINGINX                    00000123
PDT$B_SERVERS                     00000128
PDT$B_SUBTYP                    = 0000000B
PDT$B_TYPE                      = 0000000A
PDT$C_COMAREALN                 = 00000088
PDT$C_LENGTH                    = 000000E4
PDT$C_PU                        = 0000C002
PDT$C_PULENGTH                  = 00000C94
PDT$C_RINGLEN                   = 00000A88
PDT$C_SCSBASE                   = 0000000C
PDT$C_SCSEND                    = 00000084
PDT$C_UV1LENGTH                 = 00000EB4
PDT$L_ACCEPT                    = 0000000C
PDT$L_ADP                       = 000000E0
PDT$L_ALLOCDG                   = 00000010
PDT$L_ALLOCMSG                  = 00000014
PDT$L_BDTABLE                     0000014C
PDT$L_CMDRING                     00000248
PDT$L_COMAREA                     00000200
PDT$L_CONNECT                   = 00000018
PDT$L_DCONNECT                  = 00000028
PDT$L_DEALLOCDG                 = 0000001C
PDT$L_DEALLOMSG                 = 00000020
PDT$L_DEALRGMSG                 = 00000024
PDT$L_DGOVRHD                   = 000000B8
PDT$L_MAINTFCN                  = 00000078
PDT$L_MAP                       = 0000002C
PDT$L_MAPBYPASS                 = 00000030
PDT$L_MAPIRP                    = 00000034
PDT$L_MAPIRPBYP                 = 00000038
PDT$L_MAXBCNT                   = 000000BC
PDT$L_MRESET                    = 00000070
PDT$L_MSGHDRSZ                  = 000000B4
PDT$L_MSTART                    = 00000074
PDT$L_PQ_ALGNPG                   00000CB4
PDT$L_PQ_MAP                      00000C94
PDT$L_PQ_PGPHAD                   00000CA4
PDT$L_PQ_PGQBL                    00000C9C
PDT$L_PQ_PGQFL                    00000C98
PDT$L_PQ_POWNER                   00000CA0
PDT$L_PQ_SVPPTE                   00000CAC
PDT$L_PQ_UBFSVA                   00000CA8
PDT$L_PQ_USRPTE                   00000CB0
PDT$L_PU_BUFARY                   00000288
PDT$L_PU_BUFQBL                   0000011C
PDT$L_PU_BUFQFL                   00000118
PDT$L_PU_CDTARY                   000000E4
PDT$L_PU_CSR                      00000100
PDT$L_PU_FQBL                     0000010C
PDT$L_PU_FQFL                     00000108
PDT$L_PU_FQPTR                    00000104
PDT$L_PU_SB                       000000FC
PDT$L_PU_SNDQBL                   00000114
PDT$L_PU_SNDQFL                   00000110
PDT$L_PU_VC0                      000000E4
PDT$L_PU_VC1                      000000E8
PDT$L_PU_VC2                      000000EC
PDT$L_PU_VC255                    000000F0
PDT$L_QUEUEDG                   = 0000003C
```

```
PDT$L_QUEUEMDGS                 = 00000040
PDT$L_RCHMSGBUF                 = 00000044
PDT$L_RCLMSGBUF                 = 00000048
PDT$L_READCOUNT                 = 00000068
PDT$L_REJECT                    = 0000004C
PDT$L_REQDATA                   = 00000050
PDT$L_RINGBASE                    00000208
PDT$L_RLSCOUNT                  = 0000006C
PDT$L_RSPRING                     00000208
PDT$L_SENDDATA                  = 00000054
PDT$L_SENDDG                    = 00000058
PDT$L_SENDMSG                   = 0000005C
PDT$L_SENDRGDG                  = 0000007C
PDT$L_SNDCNTMSG                 = 00000060
PDT$L_STOP_VCS                  = 00000080
PDT$L_TRTABLE                     00000C88
PDT$L_TRTBLEND                    00000C90
PDT$L_TRTBLPTR                    00000C8C
PDT$L_UCBO                      = 000000DC
PDT$L_UNMAP                     = 00000064
PDT$L_WAITQBL                   = 000000B0
PDT$L_WAITQFL                   = 000000AC
PDT$M_SNGLHOST                  = 00000001
PDT$W_CMDINT                      00000204
PDT$W_PORTCHAR                  = 00000004
PDT$W_PU_CRD255                   000000FA
PDT$W_PU_CRDARY                   000000F4
PDT$W_PU_CREDO                    000000F4
PDT$W_PU_CRED1                    000000F6
PDT$W_PU_CRED2                    000000F8
PDT$W_RSPINT                      00000206
PDT$W_SIZE                      = 00000008
POLL_CMDRING                      000011BE R      03
POLL_RSPRING                      0000125D R      03
POST_POWER_FORK                   000002C3 R      03
POWER_INIT                        000002B5 R      03
PR$_IPL                         = 00000012
PR$_SID_TYP730                  = 00000003
PR$_SID_TYP750                  = 00000002
PR$_SID_TYP780                  = 00000001
PR$_SID_TYP790                  = 00000004
PR$_SID_TYP8SS                  = 00000005
PR$_SID_TYPUV1                  = 00000007
PR$_TBIS                        = 0000003A
PS1_V_ER                        = 0000000F
PS1_V_OD                        = 00000007
PS1_V_S1                        = 0000000B
PS2_S_PORTTYPE                  = 00000003
PS2_V_ER                        = 0000000F
PS2_V_PORTTYPE                  = 00000008
PS2_V_S2                        = 0000000C
PS3_V_ER                        = 0000000F
PS3_V_S3                        = 0000000D
PS4_S_CNTRLTYP                  = 00000007
PS4_S_UCODEVER                  = 00000004
PS4_V_CNTRLTYP                  = 00000004
PS4_V_ER                        = 0000000F
PS4_V_S4                        = 0000000E
PS4_V_UCODEVER                  = 00000000
PTE$C_KOWN                      = 00000000
PTE$C_KW                        = 10000000
PTE$M_PFN                       = 001FFFFF
PTE$M_VALID                     = 80000000
PTE$S_PFN                       = 00000015
PTE$V_PFN                       = 00000000
PU$CTLINIT                        0000022C RG     03
PU$DDT                            00000000 RG     03
PU$END                            00001395 R      03
PU$FUNCTABLE                      000000B2 R      03
PU$INT                            00001201 RG     03
PU$L_DRIVER_STS                   000000D0 R      03
PU$L_MPHYAD                       000000D4 R      03
PU$L_TRACE_VARIABLE               000000D8 R      03
PU$SA_POLL                        0000135B R      03
PU$SCSOFFSET                      000001E4 RG     03
PU$STARTIO                        00000038 R      03
PU$UNITINIT                       00000222 RG     03
PURGE_730                         000007A4 R      03
PURGE_750                         000007A4 R      03
PURGE_780                         000007A1 R      03
PURGE_790                         000007A1 R      03
PURGE_8SS                         000007A4 R      03
PURGE_ATTNCODE                  = 00000004
PURGE_UV1                         000007A4 R      03
PU_REGDUMP                        0000008D R      03
QDA50_CNTRLTYP                  = 0000000D
Q_DEALLOC_BUF                     00000E9D R      03
RDRX_CNTRLTYP                   = 00000007
REGSAVE                         = 0000001C
RELDATAPATH_TV                    000001A0 R      03
RELDATAP_730                      000001E3 R      03
RELDATAP_750                      000001CC R      03
RELDATAP_8SS                      000001D7 R      03
REQDATAPATH_TV                    0000019C R      03
REQDATAP_730                      000001E3 R      03
REQDATAP_750                      000001A4 R      03
REQDATAP_8SS                      000001AF R      03
RINGBASE_8SS                      00000752 R      03
RINGBASE_COM                      00000761 R      03
RINGBASE_UV1                      00000775 R      03
SA_POLL_INTVAL                  = 0000000F
SB$B_SYSTEMID                   = 00000018
SB$B_TYPE                       = 0000000A
SB$K_LENGTH                     = 00000060
SB$L_CSB                        = 0000005C
SB$L_PBBL                       = 00000010
SB$L_PBCONNX                    = 00000014
SB$L_PBFL                       = 0000000C
SB$Q_SWINCARN                   = 0000002C
SB$W_SIZE                       = 00000008
SB_ALLOC_FAIL                     00000C68 R      03
SCS$DEALL_CDT                     ******** X      03
SCS$GB_UDABURST                   ******** X      03
SCS$GQ_CONFIG                     ******** X      03
```

```
SCS$RESUMEWAITR                 ********    X   03
SCSCMG$S_SCSCMGDEF              = 00000030
SEND_OPC_MSG                      00000B13  R     03
SERVERS_ARRAY                     000007EA  R     03
SETUP_COPY_SEG1                   000010A6  R     03
SETUP_COPY_SEG2                   0000110F  R     03
SIZ...                          = 00000001
SS$_ILLCDTST                    = 00002154
SS$_ILLIOFUNC                   = 000000F4
SS$_NOLISTENER                  = 0000215C
SS$_NORMAL                      = 00000001
SS$_NOSUCHNODE                  = 0000028C
SS$_REJECT                      = 00000294
START_STOP                        00000063  R     03
STATE_ERR                         000011FB  R     03
STEP1_LIMIT                     = 0000000A
STEP1_TIMEOUT                     00000AD9  R     03
STEP2_LIMIT                     = 0000000A
STEP2_TIMEOUT                     00000AD9  R     03
STEP3_LIMIT                     = 0000000A
STEP3_TIMEOUT                     00000AD9  R     03
STOCK_RSPRING                     00000BAA  R     03
SUCCESS                           0000007B  R     03
SYS$GL_OPRMBX                   ********    X   03
TAPE_CONID                      = 00000001
TESTUDA_780                       00000A69  R     03
TESTUDA_790                       00000A69  R     03
TRACE_COMMAND                     00000476  R     03
TRACE_COMMON                      00000484  R     03
TRACE_RESPONSE                    0000047E  R     03
TSTVC2                            00000D56  R     03
TU81_CNTRLTYP                   = 00000005
UBMD$B_DATAPATH                 = 00000003
UBMD$W_MAPREG                   = 00000000
UCB$B_DEVCLASS                  = 00000040
UCB$B_DEVTYPE                   = 00000041
UCB$B_DIPL                      = 0000005E
UCB$B_FIPL                      = 0000000B
UCB$B_INITCNT                     000000A5
UCB$B_TYPE                      = 0000000A
UCB$B_UDAFLAGS                    000000A4
UCB$C_PUSIZE                    = 000000C4
UCB$L_CRB                       = 00000024
UCB$L_DDB                       = 00000028
UCB$L_DDT                       = 00000088
UCB$L_DEVCHAR                   = 00000038
UCB$L_DPC                       = 0000009C
UCB$L_FPC                       = 0000000C
UCB$L_FR3                       = 00000010
UCB$L_FR4                       = 00000014
UCB$L_PDT                       = 00000084
UCB$L_PU_ALLOC                    000000A0
UCB$L_PU_SVAPTE                   000000BC
UCB$L_SVAPTE                    = 00000078
UCB$L_SVPN                      = 00000074
UCB$M_ONLINE                    = 00000010
UCB$M_POWER                     = 00000020
UCB$M_PU_FRKBSY                 = 00000002
UCB$M_PU_HRDINI                 = 00000004
UCB$M_PU_INILOG                 = 00000040
UCB$M_PU_MRESET                 = 00000010
UCB$V_INT                       = 00000001
UCB$V_ONLINE                    = 00000004
UCB$V_POWER                     = 00000005
UCB$V_PU_BDPATH                 = 00000005
UCB$V_PU_FRKBSY                 = 00000001
UCB$V_PU_INILOG                 = 00000006
UCB$V_PU_MRESET                 = 00000004
UCB$W_ATTNCODE                    000000A8
UCB$W_BCNT                      = 0000007E
UCB$W_BOFF                      = 0000007C
UCB$W_DEVSTS                    = 00000068
UCB$W_HOSTSTEP1                   000000AE
UCB$W_HOSTSTEP2                   000000B2
UCB$W_HOSTSTEP3                   000000B6
UCB$W_HOSTSTEP4                   000000BA
UCB$W_NUMBINITS                   000000A6
UCB$W_PORTSTEP1                   000000AC
UCB$W_PORTSTEP2                   000000B0
UCB$W_PORTSTEP3                   000000B4
UCB$W_PORTSTEP4                   000000B8
UCB$W_PU_BCNT                     000000C2
UCB$W_PU_BOFF                     000000C0
UCB$W_STS                       = 00000064
UCB$W_UDASA                       000000AA
UCB$W_UNIT                      = 00000054
UCODE_ATTNCODE                  = 00000005
UCODE_VER_ARRAY                   000007F7  R     03
UDA$K_CREDTYPE                  = 00000002
UDA$K_DGTYPE                    = 00000001
UDA$K_MAINTTYPE                 = 0000000F
UDA$K_RINGEXP                   = 00000004
UDA$K_RINGSIZE                  = 00000010
UDA$K_SEQMSGTYP                 = 00000000
UDA$M_STOPPED                   = 00000002
UDA$V_STOPPED                   = 00000001
UDA50A_CNTRLTYP                 = 00000006
UDA50_CNTRLTYP                  = 00000000
UDA730                            00000A8F  R     03
UDA750                            00000A74  R     03
UDABSS                            00000A74  R     03
UDAB$B_BUFFNO                     0000000A
UDAB$B_CONID                      00000013
UDAB$B_CREDTYPE                   00000012
UDAB$B_RINGINX                    00000008
UDAB$B_RINGNO                     00000009
UDAB$C_LENGTH                   = 00000050
UDAB$L_BLINK                      00000004
UDAB$L_CTRLHDR                    00000010
UDAB$L_DESCRIP                    0000000C
UDAB$L_FLINK                      00000000
UDAB$S_CREDITS                  = 00000004
UDAB$S_MSGTYPE                  = 00000004
UDAB$T_TEXT                       00000014
```

```
UDAB$V_CREDITS                  = 00000000
UDAB$V_MSGTYPE                  = 00000004
UDAB$W_MSG_LEN                    00000010
UDAIP                             00000000
UDASA                             00000002
UDASA_ATTNCODE                  = 00000003
UDAUVT                            00000A8F R      03
UDA_M_FLAG                      = 40000000
UDA_M_OWN                       = 80000000
UDA_OUTOFREV                      00000ADF R      03
UDA_V_FLAG                      = 0000001E
UDA_V_OWN                       = 0000001F
UNMAP_ODD                         00001166 R      03
UNMAP_UNALIGN                     00001170 R      03
UPDATE_PB_SB                      00000C6B R      03
UQPORT_M_MAPPED                 = 80000000
UV1_PDT_LENGTH                  = 00001000
UV1_PDT_PAGES                   = 00000008
VA$S_VPN                        = 00000015
VA$V_VPN                        = 00000009
VCONAM                            00000D0C R      03
VCONAMLEN                       = 00000009
VC1NAM                            00000D15 R      03
VC1NAMLEN                       = 00000009
VC2NAM                            00000D1E R      03
VC2NAMLEN                       = 00000003
VEC$B_DATAPATH                  = 00000013
VEC$B_NUMREG                    = 00000012
VEC$L_ADP                       = 00000014
VEC$L_IDB                       = 00000008
VEC$L_INITIAL                   = 0000000C
VEC$L_UNITINIT                  = 00000018
VEC$W_MAPREG                    = 00000010
VIRT_TO_PHYAD                     000006D4 R      03
WAITTOOOS                         00000AF4 R      03
```

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	00000EB4 (3764.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$105_PROLOGUE	0000005C (92.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	00001395 (5013.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.03	00:00:02.71
Command processing	136	00:00:00.43	00:00:03.31
Pass 1	1207	00:00:33.14	00:01:59.58
Symbol table sort	0	00:00:03.55	00:00:12.38

Pass 2	467	00:00:08.49	00:00:30.27
Symbol table output	2	00:00:00.33	00:00:00.86
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1853	00:00:45.99	00:02:49.14

The working set limit was 2850 pages.
266333 bytes (521 pages) of virtual memory were used to buffer the intermediate code.
There were 170 pages of symbol table space allocated to hold 3129 non-local and 179 local symbols.
4287 source lines were read in Pass 1, producing 32 object records in Pass 2.
77 pages of virtual memory were used to define 71 macros.

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

Macro library name	Macros defined
_255DUA28:[DRIVER.OBJ]PALIB.MLB;1	3
_255DUA28:[SYS.OBJ]LIB.MLB;1	46
_255DUA28:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	60

3247 GETS were required to define 60 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:PUDRIVER/OBJ=OBJ$:PUDRIVER MSRC$:PUDRIVER/UPDATE=(ENH$:PUDRIVER)+EXECML$/LIB+LIB$:PALIB.MLB/LIB

PUDRIVER
LIS

RTTDRIVER
LIS

PATABLES
LIS

PASCSCTL
LIS