```
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      AAAAAA      CCCCCCCC    000000    NN      NN  FFFFFFFFFF     IIIIII      GGGGGGGG
PPPPPPPP      AAAAAA      CCCCCCCC    000000    NN      NN  FFFFFFFFFF     IIIIII      GGGGGGGG
PP      PP   AA    AA   CC          00    00   NN      NN  FF               II      GG
PP      PP   AA    AA   CC          00    00   NN      NN  FF               II      GG
PP      PP   AA    AA   CC          00    00   NNNN    NN  FF               II      GG
PP      PP   AA    AA   CC          00    00   NNNN    NN  FF               II      GG
PPPPPPPP     AA    AA   CC          00    00   NN  NN  NN  FFFFFFFF         II      GG
PPPPPPPP     AA    AA   CC          00    00   NN  NN  NN  FFFFFFFF         II      GG
PP           AAAAAAAAAA   CC          00    00   NN    NNNN  FF               II      GG  GGGGGG
PP           AAAAAAAAAA   CC          00    00   NN    NNNN  FF               II      GG  GGGGGG
PP           AA    AA   CC          00    00   NN      NN  FF               II      GG      GG      ....
PP           AA    AA   CC          00    00   NN      NN  FF               II      GG      GG      ....
PP           AA    AA     CCCCCCCC    000000    NN      NN  FF             IIIIII      GGGGGG        ....
PP           AA    AA     CCCCCCCC    000000    NN      NN  FF             IIIIII      GGGGGG        ....

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

Table of contents

```
0000     1          .TITLE  PACONFIG
0000     2          .IDENT  'V04-001'
0000     3
0000     4  ;*****************************************************************************
0000     5  ;*                                                                           *
0000     6  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8  ;*  ALL RIGHTS RESERVED.                                                     *
0000     9  ;*                                                                           *
0000    10  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    11  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    12  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    13  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    14  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    15  ;*  TRANSFERRED.                                                             *
0000    16  ;*                                                                           *
0000    17  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    18  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    19  ;*  CORPORATION.                                                             *
0000    20  ;*                                                                           *
0000    21  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    22  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23  ;*                                                                           *
0000    24  ;*                                                                           *
0000    25  ;*****************************************************************************
0000    26  ;
0000    27  ;++
0000    28  ;
0000    29  ; FACILITY:
0000    30  ;
0000    31  ;       VAX/VMS EXECUTIVE, I/O DRIVERS
0000    32  ;
0000    33  ; ABSTRACT:  CI CLUSTER CONFIGURATION DATABASE MAINTENANCE
0000    34  ;
0000    35  ; AUTHOR:  N. KRONENBERG,  MAY 1981
0000    36  ;
0000    37  ;MODIFIED BY:
0000    38  ;
0000    39  ;       V04-001 NPK3066         N. Kronenberg           7-Sep-1984
0000    40  ;               If the port microcode rev check fails, clear the
0000    41  ;               flag, INI$PORT_REV to indicate that, if a bugcheck
0000    42  ;               is taken as a result of crashing this port, it should
0000    43  ;               be the UCODEREV bugcheck, rather than the usual CIPORT
0000    44  ;               bugcheck.
0000    45  ;
0000    46  ;       V03-39  NPK3063         N. Kronenberg           20-Aug-1984
0000    47  ;               Fix SET_CIRCUIT to operate at high priority.  Fixes
0000    48  ;               the lost connect request message problem.
0000    49  ;               Add check to REFRESH_SB to return conflicting SCS
0000    50  ;               node name/ID if the SB being refreshed is the local
0000    51  ;               SB and the incarnation number being refreshed is
0000    52  ;               different from the incarnation currently there.
0000    53  ;
0000    54  ;       V03-38  NPK3060         N. Kronenberg           1-Aug-1984
0000    55  ;               Fix CNF$LBREC to attribute the loopback dg to the
0000    56  ;               correct path in the case where PANUMPORT .LE.
0000    57  ;               PAMAXPORT.
```

```
0000     58 ;               Fix check for own port number which was erroneously
0000     59 ;               concluding we had an ID pkt from a port other than
0000     60 ;               self and could therefore disable loopback datagrams.
0000     61 ;
0000     62 ;       V03-37  NPK3057         N. Kronenberg           23-Jul-1984
0000     63 ;               On port ucode rev level check failure, zero port's
0000     64 ;               reinit retry remaining count to force port to
0000     65 ;               stay offline.
0000     66 ;
0000     67 ;       V03-36  NPK3055         N. Kronenberg           14-Jul-1984
0000     68 ;               Add tally to CNF$IDREC, NEW_PATH, to track number
0000     69 ;               of ports known and if that number equals, or exceeds
0000     70 ;               the number of free dg buffers queued to the port
0000     71 ;               for receiving IDREC pkts, then queue 2 more dg buffers
0000     72 ;               to the port, one for IDREC and one for HSC error log
0000     73 ;               datagrams.  (This will be somewhat excessive if the
0000     74 ;               number of ports polled per poll interval is fewer
0000     75 ;               than 16.)
0000     76 ;               Modify CNF$REMOVE_PB to decrement PDT$W_STDGUSED for
0000     77 ;               ports that disappear (but the free dg's queued for
0000     78 ;               IDRECs and HSC error log dgs concerning that port
0000     79 ;               are left queued for future use.)
0000     80 ;               Add the concept of legal port ucode rev's that require
0000     81 ;               a warning message and error log entry, but are still
0000     82 ;               supported.
0000     83 ;               Change behavior of illegal port ucode rev to set
0000     84 ;               the port offline permanently.
0000     85 ;               Change CNF$CALC_POLLSW to use number of free dgs
0000     86 ;               currently queued for IDREC's rather than SCS$GW_PAPPDDG,
0000     87 ;               then number sysgened.
0000     88 ;
0000     89 ;       V03-35  NPK3054         N. Kronenberg           24-Jun-1984
0000     90 ;               Add check for ci780/ci750 minimum microcode rev level.
0000     91 ;               Do this check only on own port when ID packet is
0000     92 ;               received and we are getting ready to open a vc to
0000     93 ;               own port.
0000     94 ;
0000     95 ;       V03-34  NPK3052         N. Kronenberg           19-Apr-1984
0000     96 ;               Correct computation of poll sweep time:  add PASTIMOUT
0000     97 ;               and account for limit in number of free datagram buffers
0000     98 ;               set aside for concurrent handshakes.
0000     99 ;
0000    100 ;       V03-33  WHM0001         Bill Matthews           14-Apr-1984
0000    101 ;               Remove reference to SCS$GB_NODENAMEH.
0000    102 ;
0000    103 ;       V03-32  NPK3048         N. Kronenberg           4-Apr-1984
0000    104 ;               Overhaul CNF$STOP_VCS to scan the path blocks for
0000    105 ;               circuits to send shutdowns over.  This allows us
0000    106 ;               to check the PPD protocol level of target systems
0000    107 ;               and to send shutdowns only to ports with protocol
0000    108 ;               level 1 or above.  With that protocol level PPD
0000    109 ;               implementations are required to tolerate PPD types
0000    110 ;               they don't act upon.
0000    111 ;               Modify BREAK_HOST, which is executed upon receipt
0000    112 ;               of a host shutdown dg, to save SS$_NOSUCHNODE in
0000    113 ;               PB$W_VCFAIL_RSN as the aux status to report to SYSAPs.
0000    114 ;               Modify PB creation to initialize PB$W_VCFAIL_RSN to
```

```
0000   115 ;               0, i.e., no host shutdown in progress.
0000   116 ;               Modify SB creation to save PPD protocol level in
0000   117 ;               formative PB.
0000   118 ;
0000   119 ;       V03-31  NPK3047         N. Kronenberg           15-Mar-1984
0000   120 ;               Add new routine CNF$STOP_VCS to send host shutdown dgs
0000   121 ;               to all ports to which we have vcs open or are in
0000   122 ;               the process of opening circuits.
0000   123 ;               Modify logic in ENTER_PB which excludes systems
0000   124 ;               with unique system ID's but the same node names.
0000   125 ;               Enforce the exclusion except for V3.x systems which
0000   126 ;               will all have the same node name.
0000   127 ;               Fix EDIV in CNF$CALC_POLLSW.
0000   128 ;
0000   129 ;       V03-30  NPK3046         N. Kronenberg           8-Mar-1984
0000   130 ;               On receipt of an error log datagram, call new routine
0000   131 ;               REC_ERROR_DG which returns the datagram to the free
0000   132 ;               queue and decrements the PA device error count.
0000   133 ;               Add to CNF$TIMER calculation of the number of
0000   134 ;               seconds to poll every port at least once and put
0000   135 ;               the result in PDT$L_POLL_SWEEP.
0000   136 ;               Fix local port name in PB to be PAc0, with the 0
0000   137 ;               in ASCII instead of binary.
0000   138 ;
0000   139 ;       V03-29  TMK0002         Todd M. Katz            14-Feb-1984
0000   140 ;               When ENTER_PB discovers that there is a conflict between a
0000   141 ;               known system in the local system-wide configuration database
0000   142 ;               and the information provided by a remote system to which
0000   143 ;               it is attempting to establish a virtual circuit, the routine
0000   144 ;               terminates with an error status indicating that such a virtual
0000   145 ;               circuit can not be allowed to be established. Add support for
0000   146 ;               the error logging of such events.
0000   147 ;
0000   148 ;               This error logging is done only for the first time ENTER_PB
0000   149 ;               discovers that it is unable to talk to a remote system. This is
0000   150 ;               accomplished through the use of the PDT bit mask, PDT$B_PLOGMAP.
0000   151 ;               Whenever ENTER_PB determines that the information provided by a
0000   152 ;               remote system conflicts with a known system it checks the bit
0000   153 ;               within this mask which corresponds to the remote port number.
0000   154 ;               If the bit is set this means that this particular conflict has
0000   155 ;               already been logged; however, if the bit is clear this means
0000   156 ;               that this particular conflict has not yet been logged, so the
0000   157 ;               bit is set and the conflict between the remote and known systems
0000   158 ;               is logged. The bit corresponding to the remote port number is
0000   159 ;               always un-conditionally cleared whenever ENTER_PB finds no
0000   160 ;               conflict and moves the formative path block into the system-wide
0000   161 ;               configuration data base before returning success.
0000   162 ;
0000   163 ;       V03-28  PRD0071         Paul R. DeStefano       25-Feb-1984
0000   164 ;               Clear SB$L_CSB (link to newest Cluster System Block)
0000   165 ;               when a system block is created.
0000   166 ;
0000   167 ;       V03-27  NPK3044         N. Kronenberg           06-Feb-1984
0000   168 ;               Juggle action table event codes (EV$C...) to add
0000   169 ;               EV$C_ELOG = 5 = PPD$C_ELOG, the new error log datagram.
0000   170 ;               Add error log datagram handling instructions to the
0000   171 ;               action table.
```

```
0000   172 ;               Change FMT_START_DATA to set protocol rev level to 1
0000   173 ;               so we can receive error log datagrams.
0000   174 ;
0000   175 ;       V03-26  TMK0001         Todd M. Katz            03-Feb-1984
0000   176 ;               Change the use of the SYSGEN parameter PAMAXPORT. The setting
0000   177 ;               of this parameter used to indicate not only whether the local
0000   178 ;               port(s) should poll remote ports, but also represented a
0000   179 ;               software setable value for the maximum port number to poll.
0000   180 ;               PAMAXPORT still retains this latter function, but the former,
0000   181 ;               whether any polling at all should be done, has been taken over
0000   182 ;               by the new SYSGEN parameter PANOPOLL.
0000   183 ;
0000   184 ;               I have also fixed two bugs within CNF$TIMER:
0000   185 ;
0000   186 ;               1. Correct how the check is made for expiration of START/STACK
0000   187 ;                  datagrams. Right now timeouts will always be signalled for
0000   188 ;                  those timer cells within formative PBs which have not
0000   189 ;                  expired while timeouts will never be signalled for those
0000   190 ;                  timer cells that within formative PBs that have expired.
0000   191 ;                  It should be the other way around.
0000   192 ;
0000   193 ;               2. The check made for an empty pool waiter queue is done
0000   194 ;                  incorrectly. The way it is currently done guarentees that
0000   195 ;                  the queue will never be found to be empty. It is left
0000   196 ;                  up to the subsequent REMQUE, which consquently must always
0000   197 ;                  be done, to discover that the queue is actually empty.
0000   198 ;
0000   199 ;       V03-25  NPK3041         N. Kronenberg           30-Jan-1984
0000   200 ;               Fix ENTER_PB to not talk to a formative system with
0000   201 ;               different system ID, but same node name as a system
0000   202 ;               already in the system list.
0000   203 ;
0000   204 ;       V03-24  NPK3040         N. Kronenberg           20-Jan-1984
0000   205 ;               Fix bug in extraction of port number in CNF$SCSMSG_REC.
0000   206 ;
0000   207 ;       V03-23  NPK3039         N. Kronenberg           11-Jan-1984
0000   208 ;               Modify the routine to transition a formative PB
0000   209 ;               to fully open upon receipt of a CONNECT_REQ.  If
0000   210 ;               there is no formative or fully open PB (because the
0000   211 ;               ENTER_PB and no pool was available to close the vc
0000   212 ;               that was opened in anticipation of a successful
0000   213 ;               ENTER_PB), then close the vc now and return.
0000   214 ;               Modify ENTER_PB to close the vc if the enter fails.
0000   215 ;
0000   216 ;       V03-022 NPK3031         N. Kronenberg           9-Aug-1983
0000   217 ;               Change UPDATE_SWINCARN to copy PPD$Q_SWINCARN instead
0000   218 ;               of PPD$Q_CURTIME.
0000   219 ;
0000   220 ;       V03-021 NPK3029         N. Kronenberg           18-Jul-1983
0000   221 ;               Enhancements for V4.0:
0000   222 ;               Remove temporary assembled in sysgen param for max
0000   223 ;               port number to poll.
0000   224 ;               Add routine CNF$SCSMSG_REC to complete transition of
0000   225 ;               formative path block to fully open state if a CONNECT_
0000   226 ;               REQ scs control msg is received before the start handshake
0000   227 ;               is complete or if the final ack is lost.
0000   228 ;               Add UPDATE_SWINCARN to use the latest sw incarnation from
```

```
0000    229 ;               c start handshake rather than the one received with the
0000    230 ;               1st START dg.
0000    231 ;               Clean up local symbols in ENTER_PB.
0000    232 ;               Drop PB$L_SB in favor of PB$L_SBLINK.
0000    233 ;               Change CNF$IDREC to reflect slightly reordered PB.
0000    234 ;               Prevent systems from being configured that have the
0000    235 ;               same system id and different node names or the same
0000    236 ;               node name and different id's.
0000    237 ;
0000    238 ;       V03-020 KTA3046         Kerbey T. Altmann       30-Mar-1983
0000    239 ;               Redo for SCS/PPD split.
0000    240 ;
0000    241 ;       V03-019 NPK3022         N. Kronenberg           28-Feb-1983
0000    242 ;               Get system software version from SYS$GQ_VERSION instead
0000    243 ;               of SYS$K_VERSION for the start handshake.
0000    244 ;
0000    245 ;       V03-018 NPK3020         N. Kronenberg           28-Feb-1983
0000    246 ;               Fix word arithmetic in action dispatcher that computes
0000    247 ;               next state/action to be longwd arithmetic.
0000    248 ;
0000    249 ;       V03-017 DWT0068         David W. Thiel          20-Jan-1983
0000    250 ;               Add call to SCS$NEW_SB when a system block is created
0000    251 ;               or reused.
0000    252 ;
0000    253 ;       V03-016 NPK3015         N. Kronenberg           28-Dec-1982
0000    254 ;               Fix bugs in LB_ENABLE which turns loopback dgs back on
0000    255 ;               when all remote vc's gone.
0000    256 ;               Fix disable of lb dg in CNF$IDREC to be BICW instead of
0000    257 ;               BISW.
0000    258 ;
0000    259 ;       V03-015 NPK3014         N. Kronenberg           16-Dec-1982
0000    260 ;               Fix to return IDREC dg to free queue in case virtual circuit
0000    261 ;               must be crashed due to remote being in neither the enabled
0000    262 ;               nor maint enabled states.
0000    263 ;               Get node name for start/stack from the sysgened node name.
0000    264 ;
0000    265 ;       V03-014 NPK3010         N. Kronenberg           11-Nov-1982
0000    266 ;               Implement probe of n ports per poll rather than 16
0000    267 ;               ports per poll.
0000    268 ;               Implement poll of sysgenable maximum number of ports
0000    269 ;               rather than all 16 (or 240).
0000    270 ;               Add loopback dg enabled flag which is updated when
0000    271 ;               VC's are broken or attempted rather than figuring out
0000    272 ;               if loopback dg's should be enabled each poller interval.
0000    273 ;               Allow SB's with no path blocks to stay in configuration
0000    274 ;               database and expand info held in SB.
0000    275 ;
0000    276 ;       V03-013 NPK3008         N. Kronenberg           6-Oct-1982
0000    277 ;               Change FMT_START_DATA to include new protocol, nodename,
0000    278 ;               current time, and shortened hardware version fields in
0000    279 ;               start/stack dgs.
0000    280 ;
0000    281 ;       V03-012 NPK3006         N. Kronenberg           9-Sep-1982
0000    282 ;               Fixed action table to show that SET_CIRCUIT can
0000    283 ;               return status.  Fixed action dispatcher to save event
0000    284 ;               code on stack and to discard received START/STACK dg
0000    285 ;               if any, in case of action routine error status.  Fixes
```

```
0000   286 ;               free dg disappearance problem.  Also fixed action
0000   287 ;               dispatcher to discard received dg on action table lookup
0000   288 ;               failure only if there is a dg in hand.  Changed
0000   289 ;               FMT_START_DATA to put correct CPU type in dg.
0000   290 ;
0000   291 ;       V03-011 NPK3005         N. Kronenberg           19-Aug-1982
0000   292 ;               In CNF$DGREC fix search of configuration database
0000   293 ;               to call CNF$LKP_PB_MSG instead of manually matching
0000   294 ;               on remote station addr (which is an incomplete check)
0000   295 ;
0000   296 ;       V03-010 ROW0114         Ralph O. Weber          30-JUN-1982
0000   297 ;               Add a check to CNF$LBREC which prevents it from logging a
0000   298 ;               successful loopback datagram received when the previous
0000   299 ;               loopback datagram for the path in question was also
0000   300 ;               successfully received.
0000   301 ;               This change will be in a new driver image shipped in V3.1.
0000   302 ;
0000   303 ;       V03-009 NPK3001         N. Kronenberg           28-Jun-1982
0000   304 ;               Modify ENTER_PB to save SB link permanently in PB$L_SBLINK.
0000   305 ;               Fix CNF$REMOVE_PB to patch the SB link to the next path to
0000   306 ;               use for a connection.
0000   307 ;
0000   308 ;       V03-008 ROW0112         Ralph O. Weber          27-JUN-1982
0000   309 ;               Change loopback datagram logging to use ELOG$CABLES instead of
0000   310 ;               ELOG$PACKET so that the error log type field gets set
0000   311 ;               correctly.  Remove crossed loopback path logic which isn't
0000   312 ;               supported by the hardware anyway.  Fix loopback status to
0000   313 ;               always be successful when no loopback datagram is sent because
0000   314 ;               there is another known node.
0000   315 ;               This change will be in a new driver image shipped in V3.1.
0000   316 ;
0000   317 ;       V03-007 ROW0109         Ralph O. Weber          24-JUN-1982
0000   318 ;               Modify CNF$POLL to send loopback datagrams if and only if no
0000   319 ;               bits are set in the PDT port bit map, or the only bit set in
0000   320 ;               the map is the one for the port on which the loopback datagram
0000   321 ;               would be sent.
0000   322 ;               This change will be in a new driver image shipped in V3.1.
0000   323 ;
0000   324 ;       V03-006 ROW0106         Ralph O. Weber          23-JUN-1982
0000   325 ;               Add error logging for loopback datagrams to CNF$POLL and
0000   326 ;               CNF$LBREC.  Enhance this error logging to aid in the detection
0000   327 ;               of a single pair of crossed wires between a port and the star
0000   328 ;               coupler.  (N.B. the hardware currently does not support these
0000   329 ;               crossed wires checks.)
0000   330 ;               This change will be in a new driver image shipped in V3.1.
0000   331 ;
0000   332 ;       V03-005 ROW0097         Ralph O. Weber           7-JUN-1982
0000   333 ;               Added calls to error logging routines in CNF$IDREC at
0000   334 ;               UPDATE_CBL_STS and UPDATE_PTH_STS.  Modified comments in
0000   335 ;               CNF$POLL is show that loop-back datagrams are not currently
0000   336 ;               supported and thus their results need not be logged.  Also
0000   337 ;               added necessary reference to the $PAERDEF macro.
0000   338 ;               This change will be in a new driver image shipped in V3.1.
0000   339 ;
0000   340 ;       V03-004 NPK2020         N. Kronenberg           23-Apr-1982
0000   341 ;               Modified ENTER_PB to discard formative PB for system
0000   342 ;               that is already in the database but with a different
```

```
0000   343 ;               incarnation number.  Prevents configuration of two
0000   344 ;               different systems that have the same system ID.
0000   345 ;
0000   346 ;       V03-003 NPK2019         N. Kronenberg           9-Apr-1982
0000   347 ;               Fixed PB allocation failure bug.
0000   348 ;               Made PB lookup failure in CNFSDGREC recoverable.
0000   349 ;
0000   350 ;       V03-002 NPK2018         N. Kronenberg           25-Mar-1982
0000   351 ;               Fixed to use short datagrams instead of LRP's for
0000   352 ;               REQID and SETCKT's.
0000   353 ;               Fixed to not do start handshake with remote port
0000   354 ;               in other than an enabled state.  If IDREC arrives
0000   355 ;               from port to which VC is open and remote port is
0000   356 ;               in other than an enabled state, crash the VC.
0000   357 ;               Updated format of start/stack dg.
0000   358 ;               Modify to allocate and attach a dg pkt to each
0000   359 ;               PB for use during VC crash.
0000   360 ;
0000   361 ;       V03-001 NPK2016         N. Kronenberg           18-Mar-1982
0000   362 ;               Fixed .TITLE
0000   363 ;
0000   364 ;--
```

```
0000    366 ;++
0000    367 ; This module of the CI port driver is responsible for polling the
0000    368 ; nodes in the cluster for new arrivals and for conducting the
0000    369 ; START handshake protocol necessary to opening port-port virtual
0000    370 ; circuits to new nodes.
0000    371 ;
0000    372 ; The system wide configuration database consists of:
0000    373 ;
0000    374 ;
0000    375 ;       SCS$GQ_CONFIG
0000    376 ;            !
0000    377 ;            v
0000    378 ;       System Block ----> Path Block ----> Path Block ---->...
0000    379 ;            !
0000    380 ;            v
0000    381 ;       System Block ----> Path Block ---->...
0000    382 ;            !
0000    383 ;            v
0000    384 ;           ...
0000    385 ;
0000    386 ; Both systems and paths with open port-port VC's and systems
0000    387 ; with no open paths are kept on the above list.
0000    388 ;
0000    389 ; When an IDREC datagram is received for a node which is currently
0000    390 ; unknown, a PB is created for it and linked to the formative PB
0000    391 ; list for this port.  When a START/STACK datagram is received from
0000    392 ; that port as part of the START handshake, a formative SB is
0000    393 ; created and linked to the PB.  The formative datastructure looks
0000    394 ; like:
0000    395 ;
0000    396 ;       PDT
0000    397 ;        !
0000    398 ;        v
0000    399 ;       Path Block ----> (System Block)
0000    400 ;        !
0000    401 ;        v
0000    402 ;       Path Block ----> (System Block)
0000    403 ;        !
0000    404 ;        v
0000    405 ;       ...
0000    406 ;
0000    407 ; When the START handshake is complete, a matching SB is sought in
0000    408 ; the system configuration database.  If one is found, then the
0000    409 ; formative SB is discarded and the formative PB linked to the
0000    410 ; existing SB.  If no matching SB is found, then the formative SB
0000    411 ; is moved to the system configuration database and, with it, its
0000    412 ; formative PB.
0000    413 ;
0000    414 ; The configuration poller is awakened periodically for each local
0000    415 ; port by the timer scan module.  Each time it wakes up, it allocates
0000    416 ; n (SCS$GB_PANPOLL) datagrams from pool and uses these datagrams
0000    417 ; to send REQID's to the next n ports.
0000    418 ;
0000    419 ; Datagram management is as follows:  Upon port initialization
0000    420 ; SGN$GB_PPDDG datagrams are preallocated and linked to the
0000    421 ; datagram free queue for receipt of IDREC's.  When any start
0000    422 ; handshake datagram is received (including IDREC) which is turned
```

```
0000    423 ; around to send the next protocol message, it is sent with
0000    424 ; RETFLAG=FALSE so that the datagram is returned to the free
0000    425 ; queue.  A received datagram which does not result in a new
0000    426 ; datagram being sent is simply returned to the free queue.
0000    427 ; Datagrams that must be allocated from pool because there is no
0000    428 ; received datagram to turn around (e.g., START/STACK retries)
0000    429 ; are sent out with RETFLAG=TRUE to return them on the response
0000    430 ; queue.  Datagram buffers returned via the response queue are
0000    431 ; deallocated to pool again.
0000    432 ;
0000    433 ; The major routines in this module (in order of appearance) are:
0000    434 ;
0000    435 ; CNF$POLL              -The configuration poller which wakes up
0000    436 ;                       periodically and sends REQID's.
0000    437 ;
0000    438 ; CNF$IDREC            -Called by the interrupt service module when
0000    439 ;                       an unsolicited (XCT_ID=0) IDREC arrives.
0000    440 ;                       If the sending port (station) currently has
0000    441 ;                       no PB in either the system wide database or
0000    442 ;                       in the PDT formative PB list, then a PB is
0000    443 ;                       created and START handshake initiated.  Else
0000    444 ;                       the IDREC is discarded.
0000    445 ;
0000    446 ; CNF$DGREC            -Called by the interrupt service module when
0000    447 ;                       a START, STACK, or ACK dg is received.  The
0000    448 ;                       action dispatcher, ACTION_DISP is called.
0000    449 ;
0000    450 ; ACTION_DISP          -Based on the path's current state and the
0000    451 ;                       event that just occurred, a sequence of
0000    452 ;                       action routines is called.  These correspond
0000    453 ;                       to the handshake steps described in the
0000    454 ;                       SCA spec.  The actions are table driven.
0000    455 ;
0000    456 ; Assorted action      -E.g., send a START dg, set a timer on the
0000    457 ; routines              path, build a system block...
0000    458 ;
0000    459 ;--
```

```
                        .SBTTL  DEFINITIONS
                ;
                ; Set PSECT to driver code:
                ;

00000000        .PSECT  $$$115_DRIVER,LONG

                ;
                ; System definitions (LIB.MLB):
                ;

                        .nocross
                        $CRBDEF                         ; Channel Request Block offsets
                        $DDBDEF                         ; Device Datablock offsets
                        $DYNDEF                         ; Structure type codes
                        $IPLDEF                         ; IPL definitions
                        $PBDEF                          ; Path Block offsets
                        $PDTDEF                         ; Port Descriptor Table offsets
                        $PRDEF                          ; Internal Processor Registers
                        $SBDEF                          ; System Block offsets
                        $SSDEF                          ; System service definitions
                        $SYSAPDEF                       ; DG disposal flags
                        $UCBDEF                         ; Unit Control Block offsets

                ;
                ; PADRIVER definitions (PALIB.MLB):
                ;

                        $PAERDEF                        ; Port driver error code values
                        $PAPBDEF                        ; CI extension to PB
                        $PAPDTDEF                       ; CI extension to PDT
                        $PAUCBDEF                       ; CI extension to UCB
                        $PPDDEF                         ; PPD layer of message/dg header
                        .cross
```

```
                        .SBTTL  CNF$POLL, PERIODICALLY SEND REQID TO PORTS

;+
; CNF$POLL is awakened periodically by CNF$TIMER. If remote port polling is
; enabled (SCS$GB_PANOPOLL is set to 0), it allocates as many datagram buffers
; as there are ports to poll per interval (up to the maximum legal port #
; specified by SCS$GB_PAMXPORT or the maximum legal hardware port # specified by
; PDT$B_MAX_PORT - which is ever is the smallest), and sends a REQID to each
; port. The sent buffers are reclaimed on the response queue and returned to
; pool.
;
; If datagram receipt is currently inhibited from this remote port,
; then datagrams are first reenabled via a SETCKT command.
;
; If the sweep does not complete due to lack of pool, CNF$POLL returns
; without error.
;
; Later receipt of the IDREC's will cause the START
; handshake to begin for the remote systems not currently known.
;
; The poller also initiates various diagnostic activities to
; check for physical connection problems or other errors in the
; cluster:
;
;       -Before polling begins, a loopback datagram is sent out if
;        loopback dg's are enabled.  LB dg's are enabled when no
;        remote port is known; otherwise, they are disabled.
;        Later, successful receipt of the LB dg is recorded in routine
;        CNF$LBREC.  Successful receipt of the last LB dg sent on this
;        path is checked here in LB_CHECK, before sending a new LB dg.
;
;       -REQID's are sent to all ports even if we have already
;        succeeded in a START handshake.  REQID's are sent with
;        explicit path select thus forcing the port to try the path
;        even if it thinks it is bad.  Later receipt of an IDREC on this
;        path forces the port to bring it back if it was previously
;        marked bad.  It also lets us log the transition of a path
;        from bad to good.
;
; Inputs:
;
;       R4                              -Addr of PDT
;
; Outputs:
;
;       R0-R2                           -Destroyed
;       other registers                 -Preserved
;
;-

        .ENABL  LSB

CNF$POLL::

00E8 8F    BB   0000    PUSHR   #^M<R3,R5,R6,R7>        ; Save some registers
00000000'GF    95   0004    TSTB    G^SCS$GB_PANOPOLL       ; Is remote polling enabled?
          03   13   000A    BEQL    5$                      ; Continue if it is
```

```
                     00E7   31  000C   554          BRW     CONFIG_EXIT             ; Else exit poller
                            000F   555
                56   017E C4   9A  000F   556 5$:     MOVZBL  PDT$B_NXT_PORT(R4),R6   ; Get starting port # to poll
             55   00000000'GF   9A  0014   557          MOVZBL  G^SCS$GB_PAMXPORT,R5    ; Get maximum port #
                50   017C C4   9A  001B   558          MOVZBL  PDT$B_MAX_PORT(R4),R0   ; Get max port supported by CI
                     50   55   D1  0020   559          CMPL    R5,R0                   ; SYSGENed max greater than hardware?
                          03   15  0023   560          BLEQ    7$                      ; Branch if not
                     55   50   D0  0025   561          MOVL    R0,R5                   ; Else hardware max prevails
                            0028   562
                57   017F C4   9A  0028   563 7$:     MOVZBL  PDT$B_REQIDPS(R4),R7    ; Get value of path to select
                            002D   564
                            002D   565 LB_CHECK:
                            002D   566
              50   017F C447   90  002D   567          MOVB    PDT$B_PO_LBSTS-1(R4)[R7],R0  ; Get LB status byte for
                            0033   568                                                  ;  current path.
         51   50   FFFFFFFE 8F   CB  0033   569          BICL3   #^C<PDT$M_CUR_LBS>,R0,R1     ; Isolate current status in R1
                          0E   12  003B   570          BNEQ    10$                     ; Branch if current status is good.
                     50   02   93  003D   571          BITB    #PDT$M_PRV_LBS, R0      ; Was previous status bad?
                          09   13  0040   572          BEQL    10$                     ; Branch if it was bad.
                          52   D4  0042   573          CLRL    R2                      ; Indicate no packet present.
                            0044   574          ASSUME  PAER$K_ES_L1GB EQ <PAER$K_ES_L0GB + 1>
                50   57   05   C1  0044   575          ADDL3   #<PAER$K_ES_L0GB-1>, R7, R0 ; Form error subtype code.
                        FFB5'   30  0048   576          BSBW    ELOG$CABLES             ; Log error via general cables state
                            004B   577                                                  ; change logger.
                            004B   578
                53   51   51   C1  004B   579 10$:     ADDL3   R1,R1,R3                ; Position current status as
                            004F   580                                                  ;  previous and save
                          02   E0  004F   581          BBS     #PDT$V_LBDG,-           ; Branch if loopback dg's currently
                     0110 C4        0051   582                  PDT$W_PORT_STS(R4),-    ;  enabled
                          09        0054   583                  SEND_CB                 ;
           017F C447   53   01   89  0055   584          BISB3   #PDT$M_CUR_LBS, R3, -   ; Otherwise, loopback datagrams are
                            005C   585                  PDT$B_PO_LBSTS-1(R4)[R7];  not needed; pretend they were
                          24   11  005C   586          BRB     START_REQID             ;  successful and go do request id's.
                            005E   587
                            005E   588 SEND_LB:
                            005E   589
                        FF9F'   30  005E   590          BSBW    INT$ALLOC_DG1           ; Get a dg buffer for the
                            0061   591                                                  ;  loopback dg
                     31 50   E9  0061   592          BLBC    R0,20$                  ; Branch if no pool -- skip
                            0064   593                                                  ;  poller altogether
           017F C447   53   90  0064   594          MOVB    R3,PDT$B_PO_LBSTS-1(R4)[R7]  ; Else update LB status
                            006A   595                                                  ;  with current and set
                            006A   596                                                  ;  current to pending
                          3C   BB  006A   597          PUSHR   #^M<R2,R3,R4,R5>        ; Save registers
              50   0184 C4   D0  006C   598          MOVL    PDT$L_LBDG(R4),R0       ; Get addr of LB dg template
                          3A   28  0071   599          MOVC3   #<PPD$C_LB_LENGTH-PPD$B_PORT>,-
                     0C A0        0073   600                  PPD$B_PORT(R0),-        ; Copy LB dg from tmplate
                     0C A2        0075   601                  PPD$B_PORT(R2)          ;  to actual dg buffer
                          3C   BA  0077   602          POPR    #^M<R2,R3,R4,R5>        ; Restore registers
                   01   57   F0  0079   603          INSV    R7,#PPD$V_PS,-          ; Insert current path
                0F A2   02        007C   604                  #PPD$S_PS,PPD$B_FLAGS(R2)   ;  select in LB dg
                        FF7E'   30  007F   605          BSBW    INT$INS_COMQL           ; Send loopback dg on its way
                            0082   606
                            0082   607 START_REQID:
                            0082   608
         53   00000000'GF   9A  0082   609          MOVZBL  G^SCS$GB_PANPOLL,R3     ; Init count of # ports to poll this
                            0089   610                                                  ;  cycle
```

```
                                0089   611
                                0089   612 NEXT_REQID:
                                0089   613
            24 0154 C4    56    E1  0089   614          BBC     R6,PDT$B_DQIMAP(R4),40$ ; Branch if dg rec'v enabled on
                                008F   615                                              ;  this port
                      FF6E'     30  008F   616          BSBW    INT$ALLOC_PPDDG         ; Else get a dg for SETCKT
                     03 50      E8  0092   617          BLBS    R0,30$                  ; Branch if got it.
                                0095   618
                      005E      31  0095   619 20$:     BRW     CONFIG_EXIT             ; Else skip polling altogether
                                0098   620
                                C9  0098   621 30$:     BISL3   #<PPD$M_RSP@24>!-       ; Else command port to
                                0099   622                       <PPD$C_SETCKT@16>,-   ;  enable dg reception
        0C A2    56  01190000 8F    0099   623                       R6,PPD$B_PORT(R2)     ;  from specified remote port
                        14 A2  D4  00A1   624          CLRL    PPD$W_M_VAL(R2)         ; SETCKT
            10 A2    1000 8F   3C  00A4   625          MOVZWL  #PPD$M_DQI,PPD$W_MASK(R2)
            00 0154 C4    56   E5  00AA   626          BBCC    R6,PDT$B_DQIMAP(R4),35$ ; Clear DG inhibit
                      FF4D'     30  00B0   627 35$:     BSBW    INT$INS_COMQL           ; Send it on its way
                      FF4A'     30  00B3   628 40$:     BSBW    INT$ALLOC_PPDDG         ; Allocate a buffer from pool
                     3D 50      E9  00B6   629          BLBC    R0,CONFIG_EXIT          ; Branch if none available
               50    57   19    78  00B9   630          ASHL    #<PPD$V_PS+24>,R7,R0    ; Use current path
           50    01050000 8F    C8  00BD   631          BISL    #<PPD$M_RSP@24>!-       ; Send REQID to next port
                                00C4   632                       <PPD$C_REQID@16>,R0   ; REQID
            0C A2    56    50   C9  00C4   633          BISL3   R0,R6,PPD$B_PORT(R2)    ;
                        10 A2  7C  00C9   634          CLRQ    PPD$Q_XCT_ID(R2)        ; Set transaction id = 0
                      FF31'     30  00CC   635          BSBW    INT$INS_COMQL           ; Send it on its way
                         56     D6  00CF   636          INCL    R6                      ; Step to next port
                      55 56     D1  00D1   637          CMPL    R6,R5                   ; Past max legal port #?
                         0D     1A  00D4   638          BGTRU   60$                     ; Branch if so
                     07 53      F5  00D6   639          SOBGTR  R3,50$                  ; Branch if more ports to poll
                017E C4    56    90  00D9   640          MOVB    R6,PDT$B_NXT_PORT(R4)   ; Else save # of next port to
                                00DE   641                                              ;  probe on next poll interval and
                         16     11  00DE   642          BRB     CONFIG_EXIT             ;  return.
                                00E0   643
                      FFA6      31  00E0   644 50$:     BRW     NEXT_REQID              ; Go poll next port
                                00E3   645
                017E C4    94    00E3   646 60$:     CLRB    PDT$B_NXT_PORT(R4)      ; Zero # of next port to probe
                                00E7   647                                              ;  next poll interval
                         57     D6  00E7   648          INCL    R7                      ; Step to next path to use
                      02 57     D1  00E9   649          CMPL    R7,#PPD$C_PSP1          ; More than max legal?
                         03     15  00EC   650          BLEQ    70$                     ; Branch if not
                      57 01     90  00EE   651          MOVB    #PPD$C_PSP0,R7          ; Else reset to path A
                                00F1   652
            017F C4    57    90  00F1   653 70$:     MOVB    R7,PDT$B_REQIDPS(R4)    ; Put next path to use in PDT
                                00F6   654
                                00F6   655 CONFIG_EXIT:
                                00F6   656
                00E8 8F    BA  00F6   657          POPR    #^M<R3,R5,R6,R7>        ; Restore registers
                         05     00FA   658          RSB                             ; Return
                                00FB   659
                                00FB   660          .DSABL  LSB
```

```
                        .SBTTL  CNF$IDREC, PROCESS UNSOLICITED IDREC

                ;+
                ; CNF$IDREC is called from IDREC for IDREC's with transaction
                ; ID = 0.  CNF$IDREC checks the port bitmap to see if the IDREC
                ; is from a path already established or with START handshake in
                ; progress.  If not, and if the remote port is enabled, then
                ;  a formative path block is set up and a START handshake initiated.
                ;
                ; If the PB does exist, then go to UPDATE_CBL_STS.  UPDATE_CBL_STS
                ; checks if the path is fully open.  If not, no cable or path status
                ; information is maintained, and the IDREC is simply discarded.  If
                ; the path is open, and the remote port is in a state other than enabled,
                ; then the virtual circuit is crashed.  If the remote port is enabled,
                ; then cabling status is recorded in the path block as follows:
                ;
                ;       current cable status = 1 (OK) if the send path =
                ;                                 receive path in IDREC;
                ;
                ;                            = 0 (bad) otherwise.
                ;
                ; If the new current status differs from the previous, then a cable status
                ; transition is logged.
                ;
                ; The arrival of the IDREC says that the receive path of the ID must
                ; be good.  Therefore, the path status in the PB is also updated as follows:
                ;
                ;       current path status  = 1 (OK).
                ;
                ; If the current path status differs from the previous, then a path status
                ; transition is logged.
                ;
                ; Inputs:
                ;
                ;       R2                          -Addr of IDREC datagram
                ;       R4                          -Addr of PDT
                ;
                ; Outputs:
                ;
                ;       R0-R2                       -Destroyed
                ;       other registers             -Preserved
                ;-

                ;
                ; Assumptions about PB format:
                ;
00FB   709 ASSUME  PB$W_SIZE+2     EQ  PB$B_TYPE
00FB   710 ASSUME  PB$B_TYPE+1     EQ  PB$B_SUBTYP
00FB   711 ASSUME  PB$B_SUBTYP+1   EQ  PB$B_RSTATION
00FB   712 ASSUME  PB$B_RSTATION+6 EQ  PB$W_STATE
00FB   713 ASSUME  PB$W_STATE+2    EQ  PB$L_RPORT_TYP
00FB   714 ASSUME  PB$L_RPORT_TYP+4   EQ  PB$L_RPORT_REV
00FB   715 ASSUME  PB$L_RPORT_REV+4   EQ  PB$L_RPORT_FCN
00FB   716 ASSUME  PB$L_RPORT_FCN+4   EQ  PB$B_RST_PORT
00FB   717 ASSUME  PB$B_RST_PORT+1 EQ  PB$B_RSTATE
00FB   718 ASSUME  PB$B_RSTATE+1   EQ  PB$W_RETRY
```

```
                     00FB   719 ASSUME  PB$W_RETRY+2     EQ  PB$T_LPORT_NAME
                     00FB   720 ASSUME  PB$T_LPORT_NAME+4  EQ  PB$B_CBL_STS
                     00FB   721 ASSUME  PB$B_CBL_STS+1   EQ  PB$B_P0_STS
                     00FB   722 ASSUME  PB$B_P0_STS+1    EQ  PB$B_P1_STS
                     00FB   723 ASSUME  PB$B_P1_STS+2    EQ  PB$L_PDT
                     00FB   724 ASSUME  PB$L_PDT+4       EQ  PB$L_SBLINK
                     00FB   725 ASSUME  PB$L_SBLINK+4    EQ  PB$L_CDTLST
                     00FB   726 ASSUME  PB$L_CDTLST+4    EQ  PB$L_WAITQFL
                     00FB   727 ASSUME  PB$L_WAITQFL+4   EQ  PB$L_WAITQBL
                     00FB   728 ASSUME  PB$L_WAITQBL     EQ  PB$L_DUETIME
                     00FB   729 ASSUME  PB$L_DUETIME+4   EQ  PB$L_SCSMSG
                     00FB   730 ASSUME  PB$L_SCSMSG+4    EQ  PB$W_STS
                     00FB   731 ASSUME  PB$W_STS+2       EQ  PB$W_VCFAIL_RSN
                     00FB   732
                     00FB   733          .ENABL  LSB
                     00FB   734
                     00FB   735 CNF$IDREC::
                     00FB   736
   51    0C A2   9A  00FB   737          MOVZBL  PPD$B_PORT(R2),R1       ; Get sender port #
0114 C4    51   E1  00FF   738          BBC     R1,PDT$B_PORTMAP(R4),-  ; Branch if this path is
             03        0104   739                  NEW_PATH                ;  currently unknown
          00C3   31  0105   740          BRW     UPDATE_CBL_STS          ; Go update cabling status info
                     0108   741
                     0108   742 NEW_PATH:
                     0108   743
017D C4    51   91  0108   744          CMPB    R1,PDT$B_PORT_NUM(R4)   ; Is this ID from self
             03   12  010D   745          BNEQ    5$                      ; Branch if not
          084C   30  010F   746          BSBW    CHECK_PORT_REV          ; Else got check port rev level
                     0112   747
             01   EF  0112   748 5$:     EXTZV   #PPD$V_STATE,-          ; Get state of remote
             02        0114   749                  #PPD$S_STATE,-          ;  port from ID
       50    25 A2   0115   750                  PPD$B_RSTATE(R2),R0     ;
          02   50   91  0118   751          CMPB    R0,#PPD$C_ENAB          ; Is remote enabled or enab maint?
             03   13  011B   752          BEQL    10$                     ; Branch if yes
          00A8   31  011D   753          BRW     NEW_PATH_ERR            ; Else dont try for start handshake
                     0120   754
             52   DD  0120   755 10$:    PUSHL   R2                      ; Save copy of IDREC dg addr
51    00000060 8F   D0  0122   756          MOVL    #PB$C_PALENGTH,R1       ; Get size of a pathblock
      00000000'GF   16  0129   757          JSB     G^EXE$ALONONPAGED       ; Allocate one from pool
             06   50  E8  012F   758          BLBS    R0,15$                  ; Branch if got pool
             52 8ED0  0132   759          POPL    R2                      ; Else restore saved register
          0090   31  0135   760          BRW     NEW_PATH_ERR            ;  and clean up before exit
                     0138   761
          53   52   D0  0138   762 15$:    MOVL    R2,R3                   ; Set PB addr in standard register
             52 8ED0  013B   763          POPL    R2                      ; Retreive IDREC dg addr
       50    08 A3   DE  013E   764          MOVAL   PB$W_SIZE(R3),R0        ; Get addr within PB of struct size
          80   51   B0  0142   765          MOVW    R1,(R0)+                ; Set structure size
    80    0460 8F   B0  0145   766          MOVW    #DYN$C_SCS+<DYN$C_SCS_PB@8>,(R0)+  ; Set struct type, subtype
       51    0C A2   9A  014A   767          MOVZBL  PPD$B_PORT(R2),R1       ; Get remote port #
00 0114 C4    51   E3  014E   768          BBCS    R1,PDT$B_PORTMAP(R4),20$  ; Mark port has PB in map
                     0154   769
          019A C4   B6  0154   770 20$:    INCW    PDT$W_STDGUSED(R4)      ; Step # dgs needed for IDRECs
          019A C4   B1  0158   771          CMPW    PDT$W_STDGUSED(R4),-    ; Compare # dgs needed with # queued now
          0198 C4        015C   772                  PDT$W_STDGDYN(R4)       ;
                11   1F  015F   773          BLSSU   22$                     ; Branch if enough for now
                07   BB  0161   774          PUSHR   #^M<R0,R1,R2>           ; Else save our registers and
          50    02   9A  0163   775          MOVZBL  #2,R0                   ;  queue 1 dg for IDRECs + 1 dg
```

```
                FE97'   30  0166   776          BSBW    SCS$ALL_FRDGS           ;  for HSC error logging
              04 50     E9  0169   777          BLBC    R0,21$                  ; Branch if didn't get buffers
          0198 C4       B6  016C   778          INCW    PDT$W_STDGDYN(R4)       ; Show 1 more dg available for IDRECs
                                0170   779
                07      BA  0170   780 21$:     POPR    #^M<R0,R1,R2>           ; Restore registers
                                0172   781
      017D C4     51      91  0172   782 22$:     CMPB    R1,PDT$B_PORT_NUM(R4)   ; ID from self?
                  05      13  0177   783          BEQL    25$                     ; Branch if so
                  04      AA  0179   784          BICW    #PDT$M_LBDG,-           ; Else disable LB dg's because
          0110 C4         017B   785                  PDT$W_PORT_STS(R4)      ;  we can contact somebody else
                                017E   786
        80    0C A2     9A  017E   787 25$:     MOVZBL  PPD$B_PORT(R2),(R0)+    ; Set PB parameters:  remote station,
                  80      B4  0182   788          CLRW    (R0)+                   ;
              80  00      B0  0184   789          MOVW    #PB$C_CLOSED,(R0)+      ;  state = closed,
        80    18 A2     7D  0187   790          MOVQ    PPD$L_RPORT_TYP(R2),(R0)+ ;  port type, dual path bit,
                                018B   791                                          ;  and ucode revision,
        80    20 A2     D0  018B   792          MOVL    PPD$L_RPORT_FCN(R2),(R0)+ ;  port function mask,
        80    24 A2     3C  018F   793          MOVZWL  PPD$B_RST_PORT(R2),(R0)+ ;  reset port (owning port),
                                0193   794                                          ;  and remote port state,
                                0193   795                                          ;  zero retry count,
    51  00DC C4     D0  0193   796          MOVL    PDT$L_UCB0(R4),R1       ; Trace back through
          51  28 A1     D0  0198   797          MOVL    UCB$L_DDB(R1),R1        ;  the UCB and DDB to device
          80  15 A1     D0  019C   798          MOVL    DDB$T_NAME+1(R1),(R0)+  ;  name, assumed to be format 'PAc0'
          FF A0   30      90  01A0   799          MOVB    #^A/0/,-1(R0)           ; Fix unit to be ascii 0 instead of binary
              80  01      90  01A4   800          MOVB    #PB$M_CUR_CBL,(R0)+     ; Set current cable status ok --
                                01A7   801                                          ;  will update later when PB is
                                01A7   802                                          ;  fully open
              80  01      90  01A7   803          MOVB    #PB$M_CUR_PS,(R0)+      ; Set current path status good,
              80  01      9B  01AA   804          MOVZBW  #PB$M_CUR_PS,(R0)+      ;  both paths
              80  54      D0  01AD   805          MOVL    R4,(R0)+                ; Fill in addr of PDT
                  80      7C  01B0   806          CLRQ    (R0)+                   ; Zero SB link and CDT list pointer
                  80      7C  01B2   807          CLRQ    (R0)+                   ; Clear formative SB link
                                01B4   808                                          ;  and due time
                  80      D4  01B4   809          CLRL    (R0)+                   ; Clear SCS msg addr
                  80      D4  01B6   810          CLRL    (R0)+                   ; Zero handshake status and VC
                                01B8   811                                          ;  fail reason
            54 A3     D4  01B8   812          CLRL    PB$L_CLSCKT_DG(R3)      ; Zero addr of emergency SETCKT dg
    0178 D4   63      0E  01BB   813          INSQUE  (R3),@PDT$Q_FORMPB+4(R4); Link PB to formative PB list
    51   8002'8F      3C  01C0   814          MOVZWL  #EV$C_SEND_START,R1     ; Set event=send a start
            02FB    31  01C5   815          BRW     ACTION_DISP             ; Init START handshake
                                01C8   816
                                01C8   817 GOT_PATH:
                                01C8   818 NEW_PATH_ERR:
                                01C8   819
                FE35'   31  01C8   820          BRW     INT$INS_DFREEQ1         ; Return dg to free queue and return
                                01CB   821
                                01CB   822 UPDATE_CBL_STS:
                                01CB   823
                  0654    30  01CB   824          BSBW    CNF$LKP_PB_MSG          ; Look up path block
                F7 50     E9  01CE   825          BLBC    R0,GOT_PATH             ; Branch if only formative
              53  51      D0  01D1   826          MOVL    R1,R3                   ; Copy PB addr to standard register
                  01      EF  01D4   827          EXTZV   #PPD$V_STATE,-          ; Get remote port state
                  02          01D6   828                  #PPD$S_STATE,-          ;  from ID
        50    25 A2         01D7   829                  PPD$B_RSTATE(R2),R0     ;
          02    50      91  01DA   830          CMPB    R0,#PPD$C_ENAB          ; Is remote enabled or maint enab?
                05      13  01DD   831          BEQL    30$                     ; Branch if so
                FE1E'   30  01DF   832          BSBW    ERR$CRASHVC             ; Else go crash VC
```

```
                E4   11   01E2   833          BRB     GOT_PATH                ; Go return dg to free queue
                                  01E4   834
                51   D4   01E4   835 30$:     CLRL    R1                      ; Set assumed new path status = bad
          02    01   EF   01E6   836          EXTZV   #PPD$V_RP,#PPD$S_RP,-   ; Isolate rec'v path in R0
     50    0F   A2        01E9   837                  PPD$B_FLAGS(R2),R0      ;
                50   DD   01EC   838          PUSHL   R0                      ; Save rec'v path for later
          02    04   ED   01EE   839          CMPZV   #PPD$V_SP,#PPD$S_SP,-   ; Send path =
     50    0F   A2        01F1   840                  PPD$B_FLAGS(R2),R0      ;  receive path?
                02   12   01F4   841          BNEQ    40$                     ; Branch if not -- paths are crossed
                51   D6   01F6   842          INCL    R1                      ; Else set new cable status ok
                                  01F8   843
          01    00   ED   01F8   844 40$:     CMPZV   #PB$V_CUR_CBL,#1,-      ; Previous status
     51    28   A3        01FB   845                  PB$B_CBL_STS(R3),R1     ;  = new status?
                03   13   01FE   846          BEQL    50$                     ; Branch if so
              FDFD'  30   0200   847          BSBW    ELOG$CBL_X_CHG          ; Else, log change in cables crossed -
                                  0203   848                                  ; uncrossed status.
                                  0203   849
01        00    51   F0   0203   850 50$:     INSV    R1,#PB$V_CUR_CBL,#1,-   ; Record new status
          28    A3        0207   851                  PB$B_CBL_STS(R3)        ;  as the current status
                50 8ED0   0209   852          POPL    R0                      ; Retreive receive path number
                BA   13   020C   853          BEQL    GOT_PATH                ; Branch if internal loopback
50        28 A340   9E   020E   854          MOVAB   PB$B_PO_STS-1(R3)[R0],R0 ; Get addr of path status byte
          06 60     E8   0213   855          BLBS    (R0),60$                ; Branch if previous status ok
          51   53   D0   0216   856          MOVL    R3, R1                  ; Else, copy PB addr. to required place
              FDE4'  30   0219   857          BSBW    ELOG$PTH_ST_CHG         ; and log presence of new good path.
                                  021C   858
          60    01   88   021C   859 60$:     BISB    #PB$M_CUR_PS,(R0)       ; Set current status good
                A7   11   021F   860          BRB     GOT_PATH                ; Clean up IDREC dg and return
                                  0221   861
                                  0221   862          .DSABL  LSB
```

```
                     0221   864          .SBTTL  CNF$SCSMSG_REC, SCS MSG REC'D
                     0221   865
                     0221   866 ;+
                     0221   867 ; Since the final ACK or STACK may be lost in the start handshake,
                     0221   868 ; the arrival of an SCS CONNECT request message from the remote
                     0221   869 ; system should be treated as a satisfactory substitute for receiving
                     0221   870 ; a final ACK  or STACK.  CNF$SCSMSG_REC is called by PASCSCTL upon
                     0221   871 ; receipt of every connect request to handle the transition of a
                     0221   872 ; formative path block, if necessary, to the fully open state.
                     0221   873 ;
                     0221   874 ; Inputs:
                     0221   875 ;
                     0221   876 ;       R2                      -Addr of SCS message (start of application
                     0221   877 ;                                data)
                     0221   878 ;       R4                      -Addr of PDT
                     0221   879 ;
                     0221   880 ; Outputs:
                     0221   881 ;
                     0221   882 ;       R0,R1,R3                -Destroyed
                     0221   883 ;       Other registers         -Preserved
                     0221   884 ;-
                     0221   885
                     0221   886          .ENABL  LSB
                     0221   887
                     0221   888 CNF$SCSMSG_REC::
                     0221   889
            52   DD  0221   890          PUSHL   R2                              ; Save SCS msg addr
      53    0174 C4  7E  0223   891          MOVAQ   PDT$Q_FORMPB(R4),R3             ; Get list of formative PB's
51    52    00B4 C4  C3  0228   892          SUBL3   PDT$L_MSGHDRSZ(R4),R2,R1        ; Back up to start of pkt
        51    0C A1  9A  022E   893          MOVZBL  PPD$B_PORT(R1),R1               ; Get # of port that sent SCS msg
              05C9   30  0232   894          BSBW    SEARCH_PATHS                    ; See if this path is formative
             26 50   E8  0235   895          BLBS    R0,TRY_TRANSIT                  ; Branch if got formative PB
      0114 C4    51   E0  0238   896          BBS     R1,PDT$B_PORTMAP(R4),-          ; Branch if not formative, but is
                 28       023D   897                  10$                             ;  known (must be open)
                          023E   898                                                  ;  Else path is closed.  Since
                          023E   899                                                  ;  we got a sequenced msg, the
                          023E   900                                                  ;  port thinks the vc is open
                53   51   D0  023E   901          MOVL    R1,R3                   ; Save port number
                   FDBC'  30  0241   902          BSBW    INT$ALLOC_PPDDG         ; Allocate PPD dg
                  1F 50   E9  0244   903          BLBC    R0,10$                  ; Branch if no pool
                          C9  0247   904          BISL3   #<PPD$M_RSPA24>!-       ; Format PPD dg into a SETCKT
                              0248   905                  <PPD$C_SETCKTA16>,-     ;
0C A2   53   01190000 8F      0248   906                  R3,-                    ;  to port specified in R3
                              0250   907                  PPD$B_PORT(R2)          ;
                8000 8F   3C  0250   908          MOVZWL  #<PPD$M_CST>,-          ;
                  10 A2       0254   909                  PPD$W_MASK(R2)          ;
                  14 A2   D4  0256   910          CLRL    PPD$W_M_VAL(R2)         ;  and ask for vc state to be closed
                   FDA4'  30  0259   911          BSBW    INT$INS_COMQH           ; Do SETCKT at high priority
                     08   11  025C   912          BRB     10$                     ; Go to finish up
                          025E   913
                          025E   914 TRY_TRANSIT:
                          025E   915
            51  8000'8F   3C  025E   916          MOVZWL  #EV$C_SCSMSG,R1         ; Else set event code
                 025D     30  0263   917          BSBW    ACTION_DISP             ; Take action to move PB from
                          0266   918                                                  ;  formative to fully open
                          0266   919                                                  ;  If PB not in right state to
                          0266   920                                                  ;  transition to open or if
```

```
                     0266   921                             ;  there is insufficient pool,
                     0266   922                             ;  or if the system has bad
                     0266   923                             ;  system name or system ID,
                     0266   924                             ;  then formative PB and formative
                     0266   925                             ;  system block are cleaned up
                     0266   926                             ;  by action routines.
52 8ED0              0266   927 10$:    POPL    R2          ; Retreive SCS msg addr
      05             0269   928         RSB                 ; Return to PASCSCTL
                     026A   929
                     026A   930         .DSABL  LSB
```

```
                     026A   932          .SBTTL  CNF$LBREC,      VERIFY REC'D LOOPBACK DG
                     026A   933
                     026A   934 ;+
                     026A   935 ; CNF$LBREC checks the data in the received loopback datagram with
                     026A   936 ; the data stored in the template lb dg linked to the PDT.  If the
                     026A   937 ; data agrees, then the loopback status for the path on which the LB
                     026A   938 ; dg was received is updated to good.  (Transitions in the status are
                     026A   939 ; checked and logged in CNF$POLL.)
                     026A   940 ;
                     026A   941 ; Inputs:
                     026A   942 ;
                     026A   943 ;       R2                      -Addr of loopback datagram
                     026A   944 ;       R4                      -Addr of PDT
                     026A   945 ;       PDT$L_LBDG(R4)          -Addr of template LB dg
                     026A   946 ;
                     026A   947 ; Outputs:
                     026A   948 ;
                     026A   949 ;       R0-R2                   -Destroyed
                     026A   950 ;       Other registers         -Preserved
                     026A   951 ;-
                     026A   952
                     026A   953          .ENABL  LSB
                     026A   954
                     026A   955 CNF$LBREC::
                     026A   956
  51   0184 C4   D0   026A   957          MOVL    PDT$L_LBDG(R4),R1       ; Get addr of template
       7E   52   7D   026F   958          MOVQ    R2,-(SP)                ; Save registers
            32   29   0272   959          CMPC    #<PPD$L_LBCRC - PPD$W_LENGTH>,-
              10 A1        0274   960                  PPD$W_LENGTH(R1),-      ; Verify rec'd data against template
              10 A2        0276   961                  PPD$W_LENGTH(R2)        ;  including LB dg length
       52   8E   7D   0278   962          MOVQ    (SP)+,R2                ; Restore registers
            50   D5   027B   963          TSTL    R0                      ; Check results of comparison
            1B   12   027D   964          BNEQ    10$                     ; Branch if don't match
         02   01   EF   027F   965          EXTZV   #PPD$V_PS,#PPD$S_PS,-   ; Get path select, 1/2 for A/B
       50   0F A2        0282   966                  PPD$B_FLAGS(R2),R0      ;  in R0
0E 017F C440   00   E2   0285   967          BBSS    #PDT$V_CUR_LBS,-                ; Set loopback datagram received
                        028C   968                  PDT$B_PO_LBSTS-1(R4)[R0], 10$ ; & branch if already got one.
            02   93   028C   969          BITB    #PDT$M_PRV_LBS,-        ; Was the previous loopback datagram
        017F C440        028E   970                  PDT$B_PO_LBSTS-1(R4)[R0];  also successful?
            06   12   0292   971          BNEQ    10$                     ; Branch if last was successful too
                      0294   972          ASSUME  PAER$K_ES_L1BG_EQ <PAER$K_ES_L0BG +1>
          50   07   C0   0294   973          ADDL    #<PAER$K_ES_L0BG-1>,R0  ; Form LB dg succesful sybtype code
             FD66'   30   0297   974          BSBW    ELOG$CABLES             ; Log cables state change
                      029A   975
             FD63'   31   029A   976 10$:     BRW     INT$DEAL_DG1            ; Deallocate LB dg and return to
                      029D   977                                              ;  interrupt service from there
                      029D   978
                      029D   979          .DSABL  LSB
```

```
                     029D   981          .SBTTL  CNF$DGREC, DISPATCH A START/STACK/ACK DATAGRAM
                     029D   982
                     029D   983 ;+
                     029D   984 ; CNF$DGREC first checks the port bit map to see if a path
                     029D   985 ; block exists for the incoming datagram.  If not, the datagram
                     029D   986 ; is deallocated.  Otherwise, the formative path block list and
                     029D   987 ; system configuration data base are searched for the path block
                     029D   988 ; with matching station address.  When the path block is found,
                     029D   989 ; the ACTION_DISP routine is called to handle the datagram.
                     029D   990 ;
                     029D   991 ; Inputs:
                     029D   992 ;
                     029D   993 ;       R2                      -Addr of datagram
                     029D   994 ;       R4                      -Addr of PDT
                     029D   995 ;
                     029D   996 ; Outputs:
                     029D   997 ;
                     029D   998 ;       R0-R3                   -Destroyed
                     029D   999 ;       other registers         -Preserved
                     029D  1000 ;-
                     029D  1001
                     029D  1002          .ENABL  LSB
                     029D  1003
                     029D  1004 CNF$DGREC::
                     029D  1005
  51   0C A2   9A    029D  1006          MOVZBL  PPD$B_PORT(R2),R1       ; Get remote port #
0114 C4   51   E0    02A1  1007          BBS     R1,PDT$B_PORTMAP(R4),-  ; Look PB existence up in
        03           02A6  1008                  PB_EXISTS               ;  path map; branch if exists
      FD56'  31      02A7  1009          BRW     INT$INS_DFREEQ1         ; Discard datagram and return
                     02AA  1010                                          ;  from there to interrupt service
                     02AA  1011
                     02AA  1012 PB_EXISTS:
                     02AA  1013
53  0174 C4   DE      02AA  1014          MOVAL   PDT$Q_FORMPB(R4),R3     ; Get formative PB listhead
      054C   30      02AF  1015          BSBW    SEARCH_PATHS            ; Search path list for PB
    09 50    E8      02B2  1016          BLBS    R0,FOUND_PB             ; Branch if success
                     02B5  1017
                     02B5  1018 CONFIG_LIST:
                     02B5  1019
      056A   30      02B5  1020          BSBW    CNF$LKP_PB_MSG          ; Locate PB in open config database
    0A 50    E9      02B8  1021          BLBC    R0,CONFIG_ERR           ; Branch if couldn't find it
  53   51    D0      02BB  1022          MOVL    R1,R3                   ; Else copy PB addr to right reg
                     02BE  1023
                     02BE  1024 FOUND_PB:
                     02BE  1025
51   12 A2   3C      02BE  1026          MOVZWL  PPD$W_MTYPE(R2),R1      ; Set event = rec'd dg type
      01FE   31      02C2  1027          BRW     ACTION_DISP             ; Transfer to action dispatcher
                     02C5  1028                                          ;  and return from there
                     02C5  1029
                     02C5  1030 CONFIG_ERR:
                     02C5  1031
                     02C5  1032          BUGCHECK        CIPORT,NONFATAL ; Inconsistent database
                     02CC  1033
      FD31'  31      02CC  1034          BRW     INT$INS_DFREEQ1         ; If nonfatal, discard dg
                     02CF  1035                                          ;  and forget it happened
                     02CF  1036
                     02CF  1037          .DSABL  LSB
```

```
                        .SBTTL  CNF$STOP_VCS,   SEND STOPS TO ALL VCS

                ;+
                ; This routine is called during a bugcheck.  It is used to notify
                ; other systems to which we have circuits open, that this system is
                ; shutting down.  Notification is best try only, no guarantees of
                ; success.
                ;
                ; CNF$STOP_VCS first checks if the PDT is offline.  If so, return
                ; is taken since the port is not operating.  Otherwise, the port
                ; map is examined to determine each port which is known.  For each
                ; known port (except self), a shutdown datagram is sent.  After a hang
                ; of an adequate number of milliseconds, the port response queue is
                ; rummaged for the sent datagram.  If not found, the port is assumed
                ; to be not opreating and return is taken without further notifications.
                ; If the sent datagram is found, it is removed from the response queue
                ; for reuse in the next host shutdown datagram.
                ;
                ; Inputs:
                ;
                ;       R4                      -PDT address
                ;
                ; Outputs:
                ;
                ;       R0-R3                   -Destroyed
                ;
                ;       Other registers         -Preserved
                ;-

                ;
                ; Shutdown datagram is assembled into the PDT.  It must not be
                ; allocated from pool since that is too risky during a bugcheck:
                ;


                        .ENABL  LSB

                CNF$STOP_VCS::

50   00DC C4    D0          MOVL    PDT$L_UCB0(R4),R0       ; Get UCB address
          04    E1          BBC     #UCB$V_ONLINE,-         ; Branch if the port
       23 64    A0                  UCB$W_STS(R0),20$       ;  is offline
       019C C4  DE          MOVAL   PDT$Q_TEMP_RSPQ(R4),-   ; Init the temorary response
       019C C4                      PDT$Q_TEMP_RSPQ(R4)     ;  queue to empty
       019C C4  DE          MOVAL   PDT$Q_TEMP_RSPQ(R4),-   ;
       01A0 C4                      PDT$Q_TEMP_RSPQ+4(R4)   ;
52   01B0 C4    DE          MOVAL   PDT$B_HSHUT_DG(R4),R2   ; Get addr of host shutdown dg
          62    7C          CLRQ    (R2)                    ; Zero self relative links to
                                                            ;  show dg not queued anywhere
 003B0014 8F    D0          MOVL    #<PDT$C_HSHUT_SIZ + <DYN$C_CIDG@16>>,-
         08 A2                      PPD$W_SIZE(R2)          ; Set structure size and type just
                                                            ;  for completeness
            0569 30          BSBW    CNF$LKP_PB_PDT          ; Look up 1st/next PB on this PDT
          03 50  E8          BLBS    R0,FOUND_VC             ; Branch if PB found to start of
                                                            ;  coroutine processing. Coroutine
                                                            ;  called back from CNF$LKP_PB_PDT
```

```
                0080   31  02FC  1096 20$:    BRW     ALL_STOPPED             ; Else no PB found and we are done
                           02FF  1097
                           02FF  1098 FOUND_VC:
                           02FF  1099
              48 A3    91  02FF  1100         CMPB    PB$B_PROTOCOL(R3),-     ; Is remote end of vc speaking a
                 01            0302  1101                 #PPD$C_PRT_ELOG         ;  high enough rev level to receive
                               0303  1102                                         ;  a host shutdown even if he doesn't
                               0303  1103                                         ;  act upon it?
                    70   1F    0303  1104         BLSSU   40$                     ; Branch if not
                               0305  1105
                               0305  1106 STOP_NEXT:
                               0305  1107
                 0C A3    91  0305  1108         CMPB    PB$B_RSTATION(R3),-     ; Is the remote end our
               017D C4        0308  1109                 PDT$B_PORT_NUM(R4)      ;  own port number?
                    68   13    030B  1110         BEQL    40$                     ; Branch if so and bypass shutdown dg
        52    01B0 C4    DE  030D  1111         MOVAL   PDT$B_HSHUT_DG(R4),R2   ; Get addr of host shutdown dg buffer
                    62   D5    0312  1112         TSTL    (R2)                    ; Is dg still queued somewhere?
                    5F   12    0314  1113         BNEQ    40$                     ; Branch if so
                 0C A3    9B  0316  1114         MOVZBW  PB$B_RSTATION(R3),-     ; Set remote port # and
                 0C A2        0319  1115                 PPD$B_PORT(R2)          ;  zero status byte
               0101 8F    B0  031B  1116         MOVW    #<PPD$C_SNDDG+<PPD$M_RSP@8>>,-
                 0E A2        031F  1117                 PPD$B_OPC(R2)           ; Set opcode and response bit
         00060002 8F    D0  0321  1118         MOVL    #<PPD$C_HSHUT_LEN+<PPD$C_HOSTSHUT@16>>,-
                 10 A2        0327  1119                 PPD$W_LENGTH(R2)        ; Set PPD length and PPD type code
                  FCD4'   30  0329  1120         BSBW    INT$INS_COMQH           ; Send it out
                               032C  1121         TIMEWAIT #<2000>,#0,#0,B        ; Wait unconditionally for 20 msec
                               0353  1122
                               0353  1123 SEARCH_RSPQ:
                               0353  1124
                               0353  1125         $QRETRY REMQHI PDT$Q_RSPQ(R4),R0,ERROR=LOCK_UNAVAIL
                               0367  1126                                         ; Remove next response pkt from
                               0367  1127                                         ;  head of response queue
                    0C   1D    0367  1128         BVS     40$                     ; Branch if no more.
        52    01B0 C4    DE  0369  1129         MOVAL   PDT$B_HSHUT_DG(R4),R2   ; Retreive addr of our datagram
              52    50   D1  036E  1130         CMPL    R0,R2                   ; Is it our shutdown datagram?
                    03   12  0371  1131         BNEQ    60$                     ; Branch if not
                    62   7C  0373  1132         CLRQ    (R2)                    ; Else show dg buffer dequeued
                               0375  1133                                         ;  from port queue
                               0375  1134
                         05    0375  1135 40$:    RSB                             ; Return from coroutine call and
                               0376  1136                                         ;  go look for next port to send
                               0376  1137                                         ;  shutdown to
                               0376  1138
        01A0 D4    60   0E    0376  1139 60$:    INSQUE  (R0),@PDT$Q_TEMP_RSPQ+4(R4) ; Else save the response on
                               037B  1140                                         ;  private queue - may want to
                               037B  1141                                         ;  look at it in the dump
                    D6   11    037B  1142         BRB     SEARCH_RSPQ             ; Continue searching response queue
                               037D  1143
                               037D  1144 LOCK_UNAVAIL:
                               037D  1145
                    8E   D5    037D  1146         TSTL    (SP)+                   ; $QRETRY BSBWs here, so pop return
                               037F  1147
                               037F  1148 ALL_STOPPED:
                               037F  1149
                         05    037F  1150         RSB
                               0380  1151
                               0380  1152         .DSABL  LSB
```

```
0380  1154          .SBTTL  ACTION DISPATCHING
0380  1155          .SBTTL  -       ACTION TABLE FORMAT
0380  1156
0380  1157 ;+
0380  1158 ; The ACTION_TABLE is a list of action routines to execute for
0380  1159 ; each combination of port-port VC state and event.  The format
0380  1160 ; of the table is a list of VC state entries.  Each state entry
0380  1161 ; is followed by a list of events possible for that state.  Each
0380  1162 ; event entry is followed by a list of actions to be taken for
0380  1163 ; the event.  The table is arranged linearly.
0380  1164 ;
0380  1165 ; The various entries are generated by the macros STATE, EVENT,
0380  1166 ; ACTION, and ENDACTION defined in the next section.  Actions
0380  1167 ; may return status or not.  For actions which do return status,
0380  1168 ; the action dispatcher checks R0 for success/fail status.  In
0380  1169 ; case of failure the action disRatcher calls routine CLEANUP
0380  1170 ; and terminates action routine execution.
0380  1171 ;
0380  1172 ; The format of the various types of entry in the action table:
0380  1173 ;
0380  1174 ;       STATE:          +--------+--------+--------+--------+
0380  1175 ;                       |offset to nxt st |  state code     |
0380  1176 ;                       +--------+--------+--------+--------+
0380  1177 ;
0380  1178 ;       EVENT:          +--------+--------+--------+--------+
0380  1179 ;                       |offset to nxt evt|  event code     |
0380  1180 ;                       +--------+--------+--------+--------+
0380  1181 ;
0380  1182 ;       ACTION:         +--------+--------+--------+--------+
0380  1183 ;                       |offset to routine|  arg    |  code  |
0380  1184 ;                       +--------+--------+--------+--------+
0380  1185 ;
0380  1186 ; Standard inputs to action routines are:
0380  1187 ;
0380  1188 ;       R1              -Argument in action table entry
0380  1189 ;       R2              -Addr of IDREC/START/STACK/ACK dg, if any
0380  1190 ;       R3              -Addr of PB
0380  1191 ;       R4              -Addr of PDt
0380  1192 ;
0380  1193 ; The end action actin type is special:  it moves the argument
0380  1194 ; into the PB state word and terminates the list of actions.  End
0380  1195 ; action entries are a single word long.
0380  1196 ;-
```

```
0380  1198          .SBTTL  -       ACTION TABLE MACROS
0380  1199 ;
0380  1200 ; Macro to define a state entry:
0380  1201 ;
0380  1202
0380  1203          .MACRO  STATE   CODE
0380  1204           .NOSHOW
0380  1205           $$$=.                                  ; Save start of state entry
0380  1206           .WORD   CODE                           ; State code
0380  1207           .IF DF $$$LAST_STATE                   ; If there was a previous
0380  1208           .=$$$LAST_STATE+ST$W_NEXT              ;  state entry, go back and
0380  1209           .WORD   $$$-$$$LAST_STATE              ;  file in its fwd link
0380  1210           .=$$$+ST$W_NEXT                        ;  and reset pointer to this entry
0380  1211           .ENDC                                  ;
0380  1212           .WORD   0                              ; Allocate word for fwd link
0380  1213           $$$LAST_STATE=$$$                      ; Define start of this entry
0380  1214           $$$LAST_EVENT=0                        ; Reset addr of last event to
0380  1215                                                  ;  show start of new list of events
0380  1216           .SHOW
0380  1217          .ENDM   STATE
0380  1218
0380  1219 ;
0380  1220 ; Macro to define event entry:
0380  1221 ;
0380  1222
0380  1223          .MACRO  EVENT   CODE
0380  1224           .NOSHOW
0380  1225           $$$=.                                  ; Save start of entry
0380  1226           .WORD   CODE                           ; Event code
0380  1227           .IF NE $$$LAST_EVENT                   ; If there was a previous event,
0380  1228           .=$$$LAST_EVENT+EV$W_NEXT              ;  then go back to it and
0380  1229           .WORD   $$$-$$$LAST_EVENT              ;  fill in its fwd link
0380  1230           .=$$$+EV$W_NEXT                        ;  and return to current entry
0380  1231           .ENDC                                  ;
0380  1232           .WORD   0                              ; Allocate word for fwd link
0380  1233           $$$LAST_EVENT=$$$                      ; Define addr of this entry
0380  1234           .SHOW
0380  1235          .ENDM   EVENT
0380  1236
0380  1237 ;
0380  1238 ; Macro to define action entry:
0380  1239 ;
0380  1240
0380  1241          .MACRO  ACTION ROUTINE,FLAG=0,ARG=0,CODE=AC$C_CONTINUE
0380  1242           .NOSHOW
0380  1243           $$$=.                                  ; Save start of entry
0380  1244           .BYTE   CODE!FLAG                      ; Action type code
0380  1245           .BYTE   ARG                            ; Argument
0380  1246           .WORD   ROUTINE-$$$                    ; Offset to action routine
0380  1247           .SHOW
0380  1248          .ENDM   ACTION
0380  1249
0380  1250 ;
0380  1251 ; Macro to define an endaction entry:
0380  1252 ;
0380  1253
0380  1254          .MACRO  ENDACTION NEWSTATE
```

```
0380  1255          .NOSHOW
0380  1256          .BYTE   AC$C_END                ; Action type code
0380  1257          .WORD   NEWSTATE                ; Action arg =  new PB state
0380  1258          .SHOW
0380  1259        .ENDM   ENDACTION
```

```
                     0380  1261          .SBTTL  -       ACTION TABLE OFFSETS AND DEFINITIONS
                     0380  1262
                     0380  1263 ;
                     0380  1264 ; Offsets to state, event and action entries in the action
                     0380  1265 ; dispatch table:
                     0380  1266 ;
                     0380  1267
00000000             0380  1268 ST$W_CODE = 0                   ; State code (codes defined in $PBDEF)
00000002             0380  1269 ST$W_NEXT = 2                   ; Offset to next state entry
                     0380  1270
00000000             0380  1271 EV$W_CODE = 0                   ; Event code
00000002             0380  1272 EV$W_NEXT = 2                   ; Offset to next event entry
                     0380  1273
00000000             0380  1274 AC$B_CODE = 0                   ; Action code
00000001             0380  1275 AC$B_ARG = 1                    ; Action routine argument
00000001             0380  1276 AC$W_NEWST = 1                  ; New path blk state on end action
00000002             0380  1277 AC$W_ACTION = 2                 ; Offset to action routine
                     0380  1278
                     0380  1279 ;
                     0380  1280 ; Event code definitions:
                     0380  1281 ;
                     0380  1282
                     0380  1283                                 ; Following codes (sign bit clear) assumed equal
                     0380  1284                                 ;  to the corresponding PPD msg types:
00000000             0380  1285 EV$C_START = 0                  ;  START dg received
00000001             0380  1286 EV$C_STACK = 1                  ;  STACK dg reveived
00000002             0380  1287 EV$C_ACK = 2                    ;  ACK dg received
00000005             0380  1288 EV$C_ELOG = 5                   ;  Error log dg received
00000006             0380  1289 EV$C_HOSTSHUT = 6               ;  Host shutdown dg received
                     0380  1290                                 ; The following codes are assumed to have
                     0380  1291                                 ;  no definition as PPD types that we
                     0380  1292                                 ;  will ever receive (needs to be in
                     0380  1293                                 ;  architecture that sign bit set codes
                     0380  1294                                 ;  are reserved.)
00008000             0380  1295 EV$C_SCSMSG = ^X8000             ; SCS control msg received (connx
                     0380  1296                                 ;  management or credit)
00008001             0380  1297 EV$C_TIMEOUT = ^X8001           ; Path timer expired
00008002             0380  1298 EV$C_SEND_START = ^X8002        ; Send 1st START, initiate handshake
                     0380  1299
                     0380  1300 ;
                     0380  1301 ; Action code definitions:
                     0380  1302 ;
                     0380  1303
00000000             0380  1304 AC$C_END = 0                    ; No more action routines, update PB state
00000001             0380  1305 AC$C_CONTINUE = 1               ; More action routines.
00000080             0380  1306 STATUS = ^X80                   ; If set, action routine returns status
```

```
                        .SBTTL  -       ACTION TABLE

                ACTION_TABLE::

                        STATE   PB$C_CLOSED                     ; New PB just created

                          EVENT EV$C_SEND_START                 ; Initiate START handshake
                            ACTION      SEND_1ST_START          ; Send 1st START dg
                            ACTION      START_TIMER             ; Enable timer
                            ENDACTION   PB$C_ST_SENT            ; State moves to start sent

                          EVENT EV$C_ELOG                       ; Error log dg received
                            ACTION      REC_ERROR_DG            ; Go log it
                            ENDACTION   PB$C_CLOSED             ; State unchanged

                        STATE   PB$C_ST_SENT                    ; State= start sent

                          EVENT         EV$C_STACK              ; Received STACK dg
                            ACTION      STOP_TIMER              ; Disable timer
                            ACTION      BUILD_SB,STATUS         ; Build a formative SB
                            ACTION      SET_CIRCUIT,STATUS      ; Tell port to open VC
                            ACTION      ENTER_PB,STATUS         ; Move PB to system database
                            ACTION      SEND_ACK                ; Send ACK
                            ENDACTION   PB$C_OPEN               ; Move PB state to open

                          EVENT         EV$C_START              ; Received START dg
                            ACTION      BUILD_SB,STATUS         ; Build formative SB
                            ACTION      SET_CIRCUIT,STATUS      ; Tell port to open VC
                            ACTION      SEND_1ST_STACK          ; Send STACK dg
                            ACTION      START_TIMER             ; Start a timer
                            ENDACTION   PB$C_ST_REC             ; Move PB state to start rec'd

                          EVENT         EV$C_TIMEOUT            ; Timer expired
                            ACTION      SEND_START,STATUS       ; Retry send of START dg
                            ACTION      START_TIMER             ; Restart timer
                            ENDACTION   PB$C_ST_SENT            ; PB state stays start sent

                          EVENT         EV$C_ELOG               ; Error log dg received
                            ACTION      REC_ERROR_DG            ; Go log it
                            ENDACTION   PB$C_ST_SENT            ; State unchanged

                        STATE           PB$C_ST_REC             ; State is start rec'd

                          EVENT         EV$C_ACK                ; Rec'd ACK dg
                            ACTION      IGNORE_DG               ; Return dg to DFREEQ
                            ACTION      STOP_TIMER              ; Disable timer
                            ACTION      ENTER_PB,STATUS         ; Move PB to system database
                            ENDACTION   PB$C_OPEN               ; Move PB state to open

                          EVENT         EV$C_SCSMSG             ; Rec'd SCS control msg
                            ACTION      STOP_TIMER              ; Stop timer
                            ACTION      ENTER_PB,STATUS         ; Move PB to system database
                            ENDACTION   PB$C_OPEN               ; Move PB state to open

                          EVENT         EV$C_STACK              ; Rec'd STACK dg
                            ACTION      STOP_TIMER              ; Disable timer
```

```
041C  1365          ACTION      UPDATE_SWINCARN         ; Copy new incarn # to SB
0420  1366          ACTION      ENTER_PB,STATUS         ; Move PB to system database
0424  1367          ACTION      SEND_ACK                ; Send ACK dg
0428  1368          ENDACTION   PB$C_OPEN               ; Move PB state to open
042B  1369
042B  1370        EVENT         EV$C_START              ; Rec'd START dg
042F  1371          ACTION      UPDATE_SWINCARN         ; Copy new incarn # to SB
0433  1372          ACTION      SEND_1ST_STACK          ; Send STACK dg
0437  1373          ACTION      START_TIMER             ; Start timer
043B  1374          ENDACTION   PB$C_ST_REC             ; PB state stays same
043E  1375
043E  1376        EVENT         EV$C_TIMEOUT            ; Timer expired
0442  1377          ACTION      SEND_STACK,STATUS       ; Try another STACK dg
0446  1378          ACTION      START_TIMER             ; Start up the timer again
044A  1379          ENDACTION   PB$C_ST_REC             ; PB state stays same
044D  1380
044D  1381        EVENT         EV$C_ELOG               ; Error log dg received
0451  1382          ACTION      REC_ERROR_DG            ; Go log it
0455  1383          ENDACTION   PB$C_ST_REC             ; State unchanged
0458  1384
0458  1385      STATE           PB$C_OPEN               ; Path state is open
045C  1386
045C  1387        EVENT         EV$C_STACK              ; Rec'd STACK dg
0460  1388          ACTION      SEND_ACK                ; Send ACK dg
0464  1389          ENDACTION   PB$C_OPEN               ; Leave PB state open
0467  1390
0467  1391
0467  1392        EVENT         EV$C_ACK                ; Rec'd ACK dg
046B  1393          ACTION      IGNORE_DG               ; Return dg to DFREEQ
046F  1394          ENDACTION   PB$C_OPEN               ; Leave PB state open
0472  1395
0472  1396        EVENT         EV$C_START              ; Rec'd START dg on open VC
0476  1397          ACTION      BREAK_PATH              ; Collapse path
047A  1398          ENDACTION   PB$C_VC_FAIL            ;  leaving PB state as set
047D  1399                                              ;  by BREAK_PATH
047D  1400
047D  1401        EVENT         EV$C_ELOG               ; Error log dg received
0481  1402          ACTION      REC_ERROR_DG            ; Go log it
0485  1403          ENDACTION   PB$C_OPEN               ; State unchanged
0488  1404
0488  1405        EVENT         EV$C_HOSTSHUT           ; Host shutdown received
048C  1406          ACTION      BREAK_HOST              ; Go close VC with special status
0490  1407          ENDACTION   PB$C_VC_FAIL            ; State is vc fail
0493  1408
0493  1409      STATE           PB$C_VC_FAIL            ; VC failure in progress
0497  1410
0497  1411        EVENT         EV$C_START              ; Rec'd START dg
049B  1412          ACTION      IGNORE_DG               ; Discard without action
049F  1413          ENDACTION   PB$C_VC_FAIL            ;
04A2  1414
04A2  1415        EVENT         EV$C_STACK              ; Rec'd STACK dg
04A6  1416          ACTION      IGNORE_DG               ; Discard without action
04AA  1417          ENDACTION   PB$C_VC_FAIL            ;
04AD  1418
04AD  1419        EVENT         EV$C_ACK                ; Rec'd ACK dg
04B1  1420          ACTION      IGNORE_DG               ; Discard without action
04B5  1421          ENDACTION   PB$C_VC_FAIL            ;
```

```
                04B8  1422
                04B8  1423      EVENT       EV$C_ELOG               ; Error log dg received
                04BC  1424        ACTION    REC_ERROR_DG            ; Go log it
                04C0  1425        ENDACTION PB$C_VC_FAIL            ; State unchanged
                04C3  1426
```

```
                                        .SBTTL  -       ACTION_DISP, ACTION DISPATCHER

                                ;+
                                ; The action dispatcher looks up in the action table the list of
                                ; action routines to execute for the current path block state and
                                ; the event that occurred.  If an action routine specifies that it
                                ; returns status, the R0 is checked upon return for success (LBS)
                                ; or failure (LBC).  On failure the cleanup routine, CLEANUP, is called
                                ; and ACTION_DISP exits.  Normally, action routines are executed
                                ; until an end action routine is encountered.  The end action automatically
                                ; sets the path block state to the value specified in the end action
                                ; argument.
                                ;
                                ; The following register conventions apply for action routines:
                                ;
                                ;       R2              -Addr of START/STACK/ACK/IDREC dg, if any
                                ;       R3              -Addr of formative PB
                                ;       R4              -Addr of PDT
                                ;       R5              -Addr of current action entry
                                ;
                                ; Actions may use R0 and R1, but must use R2 with care.  Action
                                ; routines must preserve all other registers.
                                ;
                                ; Inputs to ACTION_DISP:
                                ;
                                ;       R1                      -Event code
                                ;       R2-R4                   -As shown above
                                ;
                                ; Outputs:
                                ;
                                ;       R0-R2                   -Destroyed
                                ;       other registers         -Preserved
                                ;-

                                ASSUME  EV$C_START      EQ  0           ; Assume that events START and
                                ASSUME  EV$C_STACK      EQ  1           ;  STACK are .LE. 1
                                ASSUME  EV$C_ACK        EQ  2           ; Assume that events associated with
                                                                        ;  rec'd dgs are .LE. 2

                                ASSUME  PB$C_CLOSED     EQ  0           ; Assume that all the
                                ASSUME  PB$C_ST_SENT    EQ  1           ;  formative path block states
                                ASSUME  PB$C_ST_REC     EQ  2           ;  are .LE. 2

                                        .ENABL  LSB

                                ACTION_DISP:

                55      DD      04C3    PUSHL   R5                      ; Save a register
                51      DD      04C5    PUSHL   R1                      ; Save event code
          55    FEB5    CF      04C7    MOVAL   ACTION_TABLE,R5         ; Get addr of action table
                                04CC

                                NEXT_STATE:

                50      65      B0      MOVW    ST$W_CODE(R5),R0        ; Get next state code
            12  A3      50      B1      CMPW    R0,PB$W_STATE(R3)       ; State codes match?
                        0B      13      BEQL    LOOKUP_EVENT            ; Branch if so
            50      02  A5      32      CVTWL   ST$W_NEXT(R5),R0        ; Get offset to next state
```

```
                 4C   13   04D9  1485          BEQL    PB_STATE_ERR            ; Branch if no more states
              55 50   C0   04DB  1486          ADDL    R0,R5                   ; Else step to nxt state entry
                 EC   11   04DE  1487          BRB     NEXT_STATE              ;  and try it
                           04E0  1488
                           04E0  1489 LOOKUP_EVENT:
                           04E0  1490
                 85   D5   04E0  1491          TSTL    (R5)+                   ; Step to start of event list
                           04E2  1492
                           04E2  1493 NEXT_EVENT:
                           04E2  1494
              51 65   B1   04E2  1495          CMPW    EV$W_CODE(R5),R1        ; Event codes match?
                 0B   13   04E5  1496          BEQL    NEXT_ACTION             ; Branch if yes
           50 02 A5   32   04E7  1497          CVTWL   EV$W_NEXT(R5),R0        ; Get offset to next event
                 3A   13   04EB  1498          BEQL    PB_STATE_ERR            ; Branch if no more events
              55 50   C0   04ED  1499          ADDL    R0,R5                   ; Else step to next event entry
                 F0   11   04F0  1500          BRB     NEXT_EVENT              ;  and try it
                           04F2  1501
                           04F2  1502 NEXT_ACTION:
                           04F2  1503
                 85   D5   04F2  1504          TSTL    (R5)+                   ; Step to 1st/next action entry
                 65   95   04F4  1505          TSTB    (R5)                    ; end of action routines?
                 23   13   04F6  1506          BEQL    END_ACTION              ; Branch if so
           51 01 A5   9A   04F8  1507          MOVZBL  AC$B_ARG(R5),R1         ; Pick up argument
           50 02 A5   32   04FC  1508          CVTWL   AC$W_ACTION(R5),R0      ; Get offset to routine
                6540  16   0500  1509          JSB     (R5)[R0]                ; Call action routine
                 65   95   0503  1510          TSTB    (R5)                    ; Does routine return status?
                 EB   14   0505  1511          BGTR    NEXT_ACTION             ; Branch if not
              E8 50   E8   0507  1512          BLBS    R0,NEXT_ACTION          ; Branch if status good
                 51 8ED0   050A  1513          POPL    R1                      ; Retreive event code
              01 51   D1   050D  1514          CMPL    R1,#EV$C_STACK          ; Is it rec'd START or STACK dg?
                 03   14   0510  1515          BGTR    10$                     ; Branch if not
               FAEB'  30   0512  1516          BSBW    INT$INS_DFREEQ1         ; Else must return rec'd dg to
                           0515  1517                                          ;  free queue to prevent depletion
                           0515  1518
                 55 8ED0   0515  1519 10$:     POPL    R5                      ; Restore R5
               02C3   31   0518  1520          BRW     CLEANUP                 ; Else xfer to PB/SB cleanup and
                           051B  1521                                          ;  return from there
                           051B  1522
                           051B  1523 END_ACTION:
                           051B  1524
           01 A5   B0   051B  1525          MOVW    AC$W_NEWST(R5),-        ; Update state of path block
           12 A3          051E  1526          PB$W_STATE(R3)          ;
                 51 8ED0   0520  1527          POPL    R1                      ; Clear event type code from stack
                           0523  1528
                 55 8ED0   0523  1529 20$:     POPL    R5                      ; Restore R5
                    05     0526  1530          RSB                             ; Return
                           0527  1531
                           0527  1532 PB_STATE_ERR:
                           0527  1533
                 51 8ED0   0527  1534          POPL    R1                      ; Retreive event code
                 51   D5   052A  1535          TSTL    R1                      ; Indicate that dg is held?
                 03   19   052C  1536          BLSS    30$                     ; Branch if not
               FACF'  30   052E  1537          BSBW    INT$INS_DFREEQ1         ; Else return PPD handshake dg
                           0531  1538                                          ;  to free queue
                           0531  1539
           12 A3   B1   0531  1540 30$:     CMPW    PB$W_STATE(R3),-        ; Is path state in formative
                 02           0534  1541          #PB$C_ST_REC            ;  state?
```

```
DE   1B  0535  1542          BLEQU   10$                 ; Branch if so to delete PB and
         0537  1543                                      ;  abandon start attempt
EA   11  0537  1544          BRB     20$                 ; Else ignore, join common exit
         0539  1545
         0539  1546          .DSABL  LSB
```

```
                        .SBTTL  ACTION ROUTINES
                        .SBTTL  -       SEND_1ST_START, SEND 1ST START DG
                        .SBTTL  -       SEND_START, SEND A START DATAGRAM

                ;+
                ; SEND_START allocates a dtagram buffer from nonpaged pool,
                ; formats a START message in it and sends the datagram.  The data
                ; that goes into the START message is assembled into the message
                ; by routine FMT_START_DATA.
                ;
                ; SEND_START has two entries:  SEND_1ST_START which resets the START
                ; retry count and SEND_START which decrements and checks the retry
                ; count before sending the datagram.
                ;
                ; The retries must continue until the target remote port is polled
                ; again.  This time depends on the interval between poller wakeups,
                ; the number of ports being polled at each poller wakeup, the total
                ; number of ports to be polled, and the time between retries
                ; (SCS$GW_PASTMOUT) as follows:
                ;
                ;       # retries = (SCS$GB_PAMXPORT * SCS$GW_PAPOLINT) /
                ;                         (SCS$GB_PANPOLL * SCS$GW_PASTMOUT)
                ;
                ; The retry count is computed each time it's set since the dependent
                ; variables are dynamic SYSGEN parameters.
                ;
                ; SEND_START may fail for two reasons:  insufficient pool to
                ; allocate the datagram buffer, or retry count exceeded.
                ;
                ; Inputs:
                ;
                ;       R2                      -Addr of datagram to turn around (1ST_START)
                ;       R3                      -Addr of PB
                ;       R4                      -Addr of PDT
                ;
                ; Outputs:
                ;
                ;       R0                      -0/1 for fail/success (SEND_START only)
                ;       R1,R2                   -Destroyed
                ;       other registers         -Preserved
                ;-

                ;
                ; PPD message format assumption:
                ;

                ASSUME  PPD$W_LENGTH+2  EQ PPD$W_MTYPE

                        .ENABL  LSB

                SEND_1ST_START:

   3E   D0      MOVL    #<PPD$C_START@16 + PPD$C_START_LEN>,-
10 A2                   PPD$W_LENGTH(R2)        ; Set dg size and type
   24   11      BRB     COM_SEND_1              ; Go do it

                SEND_START:
```

```
                     053F  1605
 22 A3    B7         053F  1606          DECW    PB$W_RETRY(R3)          ; Decrement retry count
    14    13         0542  1607          BEQL    SEND_ERR                ; Branch if no retries left
  FAB9'   30         0544  1608          BSBW    INT$ALLOC_DG1           ; Allocate buffer from pool
0E 50     E9         0547  1609          BLBC    R0,SEND_ERR             ; Branch if no pool
                     054A  1610
  0242    30         054A  1611 10$:      BSBW    FMT_START_DATA          ; Set up start data
    3E    D0         054D  1612          MOVL    #<PPD$C_START@16 + PPD$C_START_LEN>,-
10 A2                054F  1613                  PPD$W_LENGTH(R2)        ; Set dg size and type
  03AB    30         0551  1614          BSBW    SNDDG_RET               ; Send dg with RETFLAG=TRUE
                     0554  1615                                          ;  to channel dg to response
                     0554  1616                                          ;  queue for return to pool
                     0554  1617
                     0554  1618 SEND_SUCCESS:
                     0554  1619
50   01    9A         0554  1620          MOVZBL  #SS$_NORMAL,R0          ; Status is success
              05     0557  1621          RSB                             ; Return
                     0558  1622
                     0558  1623 SEND_ERR:
                     0558  1624
     50    D4         0558  1625          CLRL    R0                      ; Set status = fail
              05     055A  1626          RSB                             ;
                     055B  1627
                     055B  1628          .DSABL  LSB
```

```
                                        .SBTTL  -       SEND_STACK, SEND A STACK DATAGRAM

                                ;+
                                ; This routine has two entries:
                                ;
                                ; SEND_1ST_STACK resets the retry count for sending STACK's and
                                ; recycles the received START datagram into a STACK message.
                                ; See SEND_1ST_START comments regarding calculation of the
                                ; retry count.  This entry always completes with success.
                                ;
                                ; SEND_STACK is called when the timer expires and a retry is
                                ; necessary.  It decrements and checks the retry count.  If more retries
                                ; remain, it allocates a datagram buffer from pool.  This entry can
                                ; fail due to expired retry count or insufficient pool.
                                ;
                                ; Both entries wind up by formatting and sending a STACK datagram.
                                ;
                                ; Inputs:
                                ;
                                ;       R2                      -Addr of rec'd datagram (if 1ST_STACK)
                                ;       R3                      -Addr of PB
                                ;       R4                      -Addr of PDT
                                ;
                                ; Outputs:
                                ;
                                ;       R0                      -0/1 for fail/success
                                ;       R1,R2                   -Destroyed
                                ;       other registers         -Preserved
                                ;-
                                ;
                                ; PPD message format assumption:
                                ;

                                        .ENABL  LSB

                                SEND_1ST_STACK:

0001003E 8F    D0                       MOVL    #<PPD$C_STACK@16 + PPD$C_STACK_LEN>,-
      10 A2                             PPD$W_LENGTH(R2)        ; Set dg size and type

                                COM_SEND_1:

50    00000000'GF    9A                 MOVZBL  G^SCS$GB_PAMXPORT,R0    ; Get maximum number of ports
50    00000000'GF    A4                 MULW2   G^SCS$GW_PAPOLINT,R0    ; Compute maximum port #
                                                                        ;  * poller interval
51    00000000'GF    9A                 MOVZBL  G^SCS$GB_PANPOLL,R1     ; Get # ports to poll per interval
51    00000000'GF    A4                 MULW2   G^SCS$GW_PASTMOUT,R1    ; Compute # ports to poll per
                                                                        ;  interval * start timeout
   50    50    51    C7                 DIVL3   R1,R0,R0                ; Divide, increment in case of
 22 A3    50    01    A1                 ADDW3   #1,R0,PB$W_RETRY(R3)    ;  remainder, and save retry count
            0204    30                 BSBW    FMT_START_DATA          ; Set up start data
            037A    30                 BSBW    SNDDG_NORET             ; Send dg with RETFLAG=FALSE
                                                                        ;  to channel dg buffer back to
                                                                        ;  free queue
              C4    11                 BRB     SEND_SUCCESS            ; Take success exit
```

```
                     0590  1687
                     0590  1688 SEND_STACK:
                     0590  1689
          22 A3   B7 0590  1690         DECW    PB$W_RETRY(R3)          ; Decrement retry counter
             C3   13 0593  1691         BEQL    SEND_ERR                ; Branch if no retries left
           FA68'  30 0595  1692         BSBW    INT$ALLOC_DG1           ; Allocate dg buffer
          BD 50   E9 0598  1693         BLBC    R0,SEND_ERR             ; Branch if no pool
           01F1   30 059B  1694         BSBW    FMT_START_DATA          ; Set up start data
0001003E 8F   D0 059E  1695         MOVL    #<PPD$C_STACK@16 + PPD$C_STACK_LEN>,-
          10 A2      05A4  1696                 PPD$W_LENGTH(R2)        ; Set dg size and type
           0356   30 05A6  1697         BSBW    SNDDG_RET               ; Send dg with RETFLAG=TRUE
                     05A9  1698                                         ;  to channel dg to response
                     05A9  1699                                         ;  queue when sent
           FFA8   31 05A9  1700         BRW     SEND_SUCCESS            ; Take success exit
                     05AC  1701
                     05AC  1702         .DSABL  LSB
```

```
                        05AC  1704          .SBTTL  -       SEND_ACK, SEND ACK DATAGRAM
                        05AC  1705
                        05AC  1706 ;+
                        05AC  1707 ; SEND_ACK turns a previously received STACK datagram into an
                        05AC  1708 ; ACK and sends the datagram.  No failures are possible.
                        05AC  1709 ;
                        05AC  1710 ; Inputs:
                        05AC  1711 ;
                        05AC  1712 ;       R2                      -Addr of dg being turned around
                        05AC  1713 ;       R3                      -Addr of PB
                        05AC  1714 ;       R4                      -Addr of PDT
                        05AC  1715 ;
                        05AC  1716 ; Outputs:
                        05AC  1717 ;
                        05AC  1718 ;       R0,R1                   -Destroyed
                        05AC  1719 ;       other registers         -Preserved
                        05AC  1720 ;-
                        05AC  1721
                        05AC  1722 ;
                        05AC  1723 ; PPD message format assumption:
                        05AC  1724 ;
                        05AC  1725
                        05AC  1726 ASSUME  PPD$W_LENGTH+2  EQ PPD$W_MTYPE
                        05AC  1727
                        05AC  1728          .ENABL  LSB
                        05AC  1729
                        05AC  1730 SEND_ACK:
                        05AC  1731
00020004 8F    D0       05AC  1732          MOVL    #<PPD$C_ACK@16 + PPD$C_ACK_LEN>,-
      10 A2             05B2  1733                  PPD$W_LENGTH(R2)        ; Set dg size and type
    0351    31          05B4  1734          BRW     SNDDG_NORET             ; Send dg with RETFLAG=FALSE
                        05B7  1735                                          ;  to channel dg buffer back
                        05B7  1736                                          ;  free queue.
                        05B7  1737
                        05B7  1738          .DSABL  LSB
```

```
                        .SBTTL  -       UPDATE_INCARN,  UPDATE SW INCARN FROM
                        .SBTTL  -                       2ND START/STACK
                ;+
                ; This routine exists primarily for the convenience of the HSC
                ; which wants to sent its incarnation to its startup time, but
                ; does not have a clock.  The HSC uses the first PPD$Q_CURTIME
                ; it sees in a START/STACK that is nonzero as its start time.
                ; Until it receives the time from some system in the cluster,
                ; it conducts start handshakes with a software incarnation number
                ; of zero.
                ;
                ; If VMS receives a START from the HSC before the HSC has set
                ; its start time, then the received START has an incarnation number
                ; of zero.  A subsequent START/STACK from the HSC will however have
                ; a proper incarnation number which is used by this routine to
                ; revise the formative SB.
                ;
                ; Inputs:
                ;
                ;       R2                      -Addr of START/STACK dg
                ;       R3                      -Addr of formative PB
                ;       R4                      -Addr of PDT
                ;
                ; Outputs:
                ;
                ;       R0                      -Destroyed
                ;       Other registers         -Preserved
                ;-

                        .ENABL  LSB

                UPDATE_SWINCARN:

50  30 A3   D0  MOVL    PB$L_SBLINK(R3),R0      ; Get formative SB
    28 A2   7D  MOVQ    PPD$Q_SWINCARN(R2),-    ; Update formative SB with
    2C A0       SB$Q_SWINCARN(R0)               ;  latest SW incarnation #
            05  RSB                             ; Return

                        .DSABL  LSB
```

```
05C1  1781          .SBTTL  -       ENTER_PB, MOVE PB (AND SB) FROM FORMATIVE
05C1  1782          .SBTTL  -                  LISTS TO SYSTEM WIDE DATABASE
05C1  1783
05C1  1784 ;+
05C1  1785 ; ENTER_PB moves a pathblock and, if necessary, its associated system
05C1  1786 ; block from the formative pathblock list to the system wide
05C1  1787 ; configuration database.  In the process, and SCS send message
05C1  1788 ; buffer and receive buffer, and SETCKT dg are allocated.  The send
05C1  1789 ; buffer address is stored in the PB and the receive buffer is queued to
05C1  1790 ; the port. If the allocation fails, the path block ad system block remain
05C1  1791 ; on the formative list and error exit is taken.
05C1  1792 ;
05C1  1793 ; What happens to the formative system block depends upon the current
05C1  1794 ; database:
05C1  1795 ;
05C1  1796 ;       -If a matching SB does not already exist,
05C1  1797 ;        then the formative SB is inserted in the database along
05C1  1798 ;        with its formative PB.
05C1  1799 ;
05C1  1800 ;       -If a matching system exists, then check if the
05C1  1801 ;        existing SB has any PB's linked to it.  If not, refresh the
05C1  1802 ;        old SB with information from the formative SB and link the
05C1  1803 ;        formative PB to the refreshed SB.
05C1  1804 ;
05C1  1805 ;       -If the existing matching SB has paths to it, check if the
05C1  1806 ;        existing SB and formative SB have the same software incarnation.
05C1  1807 ;        If not, then two different systems must be masquarading as the
05C1  1808 ;        same system ID and the formative SB and PB are thrown away
05C1  1809 ;        (we refuse to talk to the newcomer.)
05C1  1810 ;
05C1  1811 ;        If the incarnation numbers match, then we just add the formative
05C1  1812 ;        PB to the existing SB's list of paths and discard the formative
05C1  1813 ;        SB.
05C1  1814 ;
05C1  1815 ; A matching system means one that matches in both system ID and node
05C1  1816 ; name.  SB's that match in one, but not the other are rejected and no
05C1  1817 ; vc will be opened to such a system.
05C1  1818 ;
05C1  1819 ; Naturally, there is an exception to the rule excluding systems with
05C1  1820 ; the same node name.  Version 3.x systems with matching node names
05C1  1821 ; but unique system ID's will be permitted to enter the database.
05C1  1822 ; This is because 3.x systems all had the same node name (all blanks)
05C1  1823 ; and their presence will have no effect on the VAXcluster sysap
05C1  1824 ; in a 4.x system.
05C1  1825 ;
05C1  1826 ; Inputs:
05C1  1827 ;
05C1  1828 ;       R3                      -Addr of formative PB
05C1  1829 ;       R4                      -Addr of PDT
05C1  1830 ;
05C1  1831 ; Outputs:
05C1  1832 ;
05C1  1833 ;       R0                      -0/1 for fail/success
05C1  1834 ;       R1                      -Destroyed
05C1  1835 ;       other registers         -Preserved
05C1  1836 ;-
05C1  1837
```

```
                     05C1  1838 ;
                     05C1  1839 ; System Block adjacency assumptions:
                     05C1  1840 ;
                     05C1  1841
                     05C1  1842 ASSUME  SB$B_SYSTEMID+8 EQ  SB$W_MAXDG
                     05C1  1843 ASSUME  SB$W_MAXDG+2    EQ  SB$W_MAXMSG
                     05C1  1844 ASSUME  SB$W_MAXMSG+2   EQ  SB$T_SWTYPE
                     05C1  1845 ASSUME  SB$T_SWTYPE+4   EQ  SB$T_SWVERS
                     05C1  1846 ASSUME  SB$T_SWVERS+4   EQ  SB$Q_SWINCARN
                     05C1  1847 ASSUME  SB$Q_SWINCARN+8 EQ  SB$T_HWTYPE
                     05C1  1848 ASSUME  SB$T_HWTYPE+4   EQ  SB$B_HWVERS
                     05C1  1849 ASSUME  SB$B_HWVERS+12  EQ  SB$T_NODENAME
                     05C1  1850 ASSUME  SB$T_NODENAME+16 EQ SB$L_DDB
                     05C1  1851
           0000003C  05C1  1852 UPDATE_LEN = SB$L_DDB-SB$B_SYSTEMID
                     05C1  1853
                     05C1  1854          .ENABL  LSB
                     05C1  1855
                     05C1  1856 ENTER_PB:
                     05C1  1857
             52  DD  05C1  1858          PUSHL   R2                      ; Save R2
          FA3A'  30  05C3  1859          BSBW    INT$ALLOC_MSG           ; Allocate a msg buffer
        03 50  E8  05C6  1860          BLBS    R0,10$                  ; Branch if got it
           0114  31  05C9  1861          BRW     ENTER_ERR               ; Else go to error
                     05CC  1862
     40 A3   52  D0  05CC  1863 10$:     MOVL    R2,PB$L_SCSMSG(R3)      ; Assign buffer to PB for SCS
                     05D0  1864                                          ;  control messages sent
          FA2D'  30  05D0  1865          BSBW    INT$ALLOC_PPDDG         ; Allocate a PPD dg buffer
        03 50  E8  05D3  1866          BLBS    R0,30$                  ; Branch if got it
           00D8  31  05D6  1867          BRW     ENTER_ERR1              ; Else go clean up
                     05D9  1868
     54 A3   52  D0  05D9  1869 30$:     MOVL    R2,PB$L_CLSCKT_DG(R3)   ; Save addr of PPD dg
          FA20'  30  05DD  1870          BSBW    INT$ALLOC_MSG           ; Allocate a msg buffer for
                     05E0  1871                                          ;  SCS control msg receive
           03 50  E8  05E0  1872          BLBS    R0,40$                  ; Branch if got it
           00CB  31  05E3  1873          BRW     ENTER_ERR2              ; Else handle error
                     05E6  1874
          FA17'  30  05E6  1875 40$:     BSBW    INT$INS_MFREEQ          ; Queue buffer to port
        50   30 A3  D0  05E9  1876          MOVL    PB$L_SBLINK(R3),R0      ; Get addr of formative SB
52  00000000'GF  DE  05ED  1877          MOVAL   G^SCS$GQ_CONFIG,R2      ; Get SB listhead
           51   52  D0  05F4  1878          MOVL    R2,R1                   ; Hold starting point
                     05F7  1879
                     05F7  1880 CMP_EXIST_SBS:
                     05F7  1881
           52   62  D0  05F7  1882          MOVL    (R2),R2                 ; Get next SB in list
           51   52  D1  05FA  1883          CMPL    R2,R1                   ; Back where we started?
                75  13  05FD  1884          BEQL    MOVE_SB                 ; Branch if so, this system
                     05FF  1885                                          ;  isn't here
             18 A0  D1  05FF  1886          CMPL    SB$B_SYSTEMID(R0),-     ; Check for system ID match
             18 A2      0602  1887                  SB$B_SYSTEMID(R2)       ;  on low 4 bytes
                07  12  0604  1888          BNEQ    50$                     ; Branch if no match
             1C A0  B1  0606  1889          CMPW    SB$B_SYSTEMID+4(R0),-   ; Check for system ID match
             1C A2      0609  1890                  SB$B_SYSTEMID+4(R2)     ;
                16  13  060B  1891          BEQL    55$                     ; Branch if matches
                     060D  1892
             29 A0  B1  060D  1893 50$:     CMPW    SB$T_SWVERS+1(R0),-     ; Is the formative system block
          2E33 8F       0610  1894                  #^A/3./                 ;  for a V3.n system?
```

```
                E2   13  0613  1895          BEQL    CMP_EXIST_SBS           ; Branch if so and bypass node name
                                                                              ;  uniqueness test
                0F   BB  0615  1897          PUSHR   #^M<R0,R1,R2,R3>        ; Save registers destroyed in CMPC
          44 A0   10   29  0617  1898          CMPC3   #16,SB$T_NODENAME(R0),- ; Are node names the same?
              44 A2                                   SB$T_NODENAME(R2)       ;
                0E   13  061D  1900          BEQL    56$                     ; Branch if node names are same,
                                                                              ;  but SYSIDs are not -- can't
                                                                              ;  talk to this system because
                                                                              ;  there is a configuration error
                0F   BA  061F  1904          POPR    #^M<R0,R1,R2,R3>        ; Restore registers
                D4   11  0621  1905          BRB     CMP_EXIST_SBS           ; Continue searching existing SBs
                                                                              ;
                0F   BB  0623  1907 55$:     PUSHR   #^M<R0,R1,R2,R3>        ; Save reg destroyed by cmpc
          44 A0   10   29  0625  1908          CMPC3   #16,SB$T_NODENAME(R0),- ; Do the system's node names
              44 A2                                   SB$T_NODENAME(R2)       ;  match?
                03   13  062B  1910          BEQL    57$                     ; Continue if so
               00AC  31  062D  1911 56$:     BRW     ENTER_ERR4              ; Branch if not -- don't talk to
                                                                              ;  this system
                0F   BA  0630  1913 57$:     POPR    #^M<R0,R1,R2,R3>        ; Restore destroyed registers
          14 A2   D5   0632  1914          TSTL    SB$L_PBCONNX(R2)        ; Does existing SB have paths?
                27   12  0635  1915          BNEQ    CHK_INCARN_ERR          ; If so, go check for
                                                                              ;  inconsistent incarnations
                                                                              ;
                                                                              ;
                                          1918 REFRESH_SB:
                                                                              ;
00000000'8F     52   D1  0637  1920          CMPL    R2,#SCS$GA_LOCALSB      ; Is this the local SB?
                0E   12  063E  1921          BNEQ    DO_REFRESH              ; Branch if not
            2C A0   D1  0640  1922          CMPL    SB$Q_SWINCARN(R0),-     ; Else is the new incarnation the
            2C A2                                   SB$Q_SWINCARN(R2)       ;  same as the old?
                73   12  0645  1924          BNEQ    ENTER_ERR3              ; Branch if not -- this must be
            30 A0   D1  0647  1925          CMPL    SB$Q_SWINCARN+4(R0),-   ;  a different host masquerading
            30 A2                                   SB$Q_SWINCARN+4(R2)     ;  as us
                6C   12  064C  1927          BNEQ    ENTER_ERR3              ;
                                                                              ;
                                          1929 DO_REFRESH:
                                                                              ;
      14 A2     53   D0  064E  1931          MOVL    R3,SB$L_PBCONNX(R2)     ; Set formative PB as first path
                                                                              ;  to use for a connx in old SB
                3F   BB  0652  1933          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save regs destroyed by movc
                3C   28  0654  1934          MOVC3   #UPDATE_LEN,-           ; Update old SB with new
            18 A0                                   SB$B_SYSTEMID(R0),-     ;  SB info
            18 A2                                   SB$B_SYSTEMID(R2)       ;  from start handshake dg
                3F   BA  065A  1937          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers destroyed
                0E   11  065C  1938          BRB     DELETE_SB               ; Go delete new SB and complete
                                                                              ;  entering PB in database
                                                                              ;
                                          1941 CHK_INCARN_ERR:
                                                                              ;
            2C A0   D1  065E  1943          CMPL    SB$Q_SWINCARN(R0),-     ; Is this the same incarnation of
            2C A2                                   SB$Q_SWINCARN(R2)       ;  of the system we've already got?
                55   12  0663  1945          BNEQ    ENTER_ERR3              ; Branch if not because this means
            30 A0   D1  0665  1946          CMPL    SB$Q_SWINCARN+4(R0),-   ;  the system is really a different
            30 A2                                   SB$Q_SWINCARN+4(R2)     ;  system with the same system ID
                4E   12  066A  1948          BNEQ    ENTER_ERR3              ;
                                                                              ;
                                                                              ;
                                          1951 ; This system already has an SB in the database.  Delete formative
```

```
                                066C  1952 ; SB and insert formative path block only into the system wide
                                066C  1953 ; configuration database.  R0 has the address of the formative SB
                                066C  1954 ;
                                066C  1955
                                066C  1956 DELETE_SB:
                                066C  1957
00000000'GF   16                066C  1958         JSB     G^COM$DRVDEALMEM        ; Deallocate it to pool
      0B   11                   0672  1959         BRB     MOVE_PB                 ; Join common PB move
                                0674  1960
                                0674  1961 ;
                                0674  1962 ; This system is new.  Move the formative SB to the system wide
                                0674  1963 ; configuration database and link formative PB to it.  R0 has the
                                0674  1964 ; address of the formative SB.
                                0674  1965 ;
                                0674  1966
                                0674  1967 MOVE_SB:
                                0674  1968
   52   50   D0                  0674  1969         MOVL    R0,R2                   ; Copy addr of formative SB
04 B1   62   0E                  0677  1970         INSQUE  (R2),@4(R1)             ; Insert formative SB on tail of
                                067B  1971                                         ;  system configuration list
14 A2   53   D0                  067B  1972         MOVL    R3,SB$L_PBCONNX(R2)     ; Set formative PB as first
                                067F  1973                                         ;  path to use for a connection
                                067F  1974
                                067F  1975 MOVE_PB:
                                067F  1976
   53   63   0F                  067F  1977         REMQUE  (R3),R3                 ; Remove formative path block
10 B2   63   0E                  0682  1978         INSQUE  (R3),@SB$L_PBBL(R2)     ;  and link to system block
      06   12                   0686  1979         BNEQ    60$                     ; Branch if not block in list
                                0688  1980 ;
                                0688  1981 ; Give notification that the SB is new or reused
                                0688  1982 ;
                                0688  1983 ;       R2 -> SB
                                0688  1984 ;       R0,R1 need not be preserved
                                0688  1985 ;
00000000'GF   16                0688  1986         JSB     G^SCS$NEW_SB            ; Note the new SB
                                068E  1987 60$:
30 A3   52   D0                  068E  1988         MOVL    R2,PB$L_SBLINK(R3)      ; Save final SB addr in PB
      38 A3   DE                 0692  1989         MOVAL   PB$L_WAITQFL(R3),-      ; Set PB general wait queue
      38 A3                      0695  1990                 PB$L_WAITQFL(R3)        ;  to no entries
      38 A3   DE                 0697  1991         MOVAL   PB$L_WAITQFL(R3),-      ;
      3C A3                      069A  1992                 PB$L_WAITQBL(R3)        ;
   0112 C4   B6                  069C  1993         INCW    PDT$W_PBCOUNT(R4)       ; Step count of PB's on this PDT
50   0C A3   D0                  06A0  1994         MOVL    PB$B_RSTATION(R3),R0    ; Retrieve the remote port number
00 0134 C4   50   E5             06A4  1995         BBCC    R0,PDT$B_PLOGMAP(R4),65$; Clear bit in error logging mask
                                06AA  1996                                         ; corresponding to remote port number
   50   01   3C                  06AA  1997 65$:    MOVZWL  #SS$_NORMAL,R0          ; Set status = success
                                06AD  1998
                                06AD  1999 ENTER_DONE:
                                06AD  2000
      52 8ED0                   06AD  2001         POPL    R2                      ; Restore saved register
         05                     06B0  2002         RSB                             ; Return
                                06B1  2003
                                06B1  2004 ENTER_ERR1:
                                06B1  2005 ENTER_ERR2:
                                06B1  2006
52   40 A3   D0                  06B1  2007         MOVL    PB$L_SCSMSG(R3),R2      ; Get addr of SCS send buffer
      F948'  30                 06B5  2008         BSBW    INT$DEAL_MSG            ;  and return to pool
```

```
                    26   11  06B8  2009          BRB     ENTER_ERR               ; Join common error exit
                                06BA  2010
                                06BA  2011 ENTER_ERR3:
                                06BA  2012
          51    0C A3    D0  06BA  2013          MOVL    PB$B_RSTATION(R3),R1    ; Retrieve the remote port number
    13 0134 C4    51    E2  06BE  2014          BBSS    R1,PDT$B_PLOGMAP(R4),70$; Branch if remote port already logged
                  55    DD  06C4  2015          PUSHL   R5                      ; Otherwise save R5
                55  52    D0  06C6  2016          MOVL    R2,R5                   ; Move known system SB address into R5
                  52    D4  06C9  2017          CLRL    R2                      ; Indicate that there is no packet
                51  53    D0  06CB  2018          MOVL    R3,R1                   ; Move remote PB address address into R3
                50  08    9A  06CE  2019          MOVZBL  #PAER$K_ES_RSCKS,R0     ; Set the appropriate error subtype
                   F92C'  30  06D1  2020          BSBW    ELOG$PACKET             ; Go log conflict
                     55 8ED0  06D4  2021          POPL    R5                      ; Restore R5
                                06D7  2022
                   F926'  30  06D7  2023 70$:     BSBW    INT$MFQ2POOL            ; Remove queued SCS recv buffer
                     D5   11  06DA  2024          BRB     ENTER_ERR2              ; Join rest of error handling
                                06DC  2025
                                06DC  2026 ENTER_ERR4:
                                06DC  2027
                     0F   BA  06DC  2028          POPR    #^M<R0,R1,R2,R3>        ; Restore reg lost in node name
                                06DE  2029                                          ;  comparison
                     DA   11  06DE  2030          BRB     ENTER_ERR3              ; Join common cleanup
                                06E0  2031
                                06E0  2032 ENTER_ERR:
                                06E0  2033
          52    54 A3    D0  06E0  2034          MOVL    PB$L_CLSCKT_DG(R3),R2   ; Get the close circuit dg addr
                   06    12  06E4  2035          BNEQ    80$                     ; Branch if got one
                  F917'  30  06E6  2036          BSBW    INT$ALLOC_PPDDG         ; Else allocate a dg buffer
                C1 50    E9  06E9  2037          BLBC    R0,ENTER_DONE           ; Branch if no pool -- this vc will
                                06EC  2038                                          ;  dangle till somebody tries to use
                                06EC  2039                                          ;  it by sending a connect request.
                                06EC  2040                                          ;  At that time we have another chance
                                06EC  2041                                          ;  to set it closed.
                                06EC  2042
                             C9 06EC  2043 80$:     BISL3   #<PPD$M_RSP@24>!-       ; Format the dg into a SETCKT
                                06ED  2044                  <PPD$C_SETCKT@16>,-      ;
    0C A3    01190000 8F        06ED  2045                  PB$B_RSTATION(R3),-      ;
                   0C A2        06F4  2046                  PPD$B_PORT(R2)           ;
        10 A2    8000 8F    3C  06F6  2047          MOVZWL  #PPD$M_CST,PPD$W_MASK(R2)
                     14 A2    D4  06FC  2048          CLRL    PPD$W_M_VAL(R2)         ;  and ask for vc state to be closed
                      F8FE'   30  06FF  2049          BSBW    INT$INS_COMQH           ; Do it at high priority
                        50    D4  0702  2050          CLRL    R0                      ; Set status to failed
                        A7    11  0704  2051          BRB     ENTER_DONE              ; Go to exit routine
                                0706  2052
                                0706  2053          .DSABL  LSB
```

```
                     0706  2055          .SBTTL  -       BUILD_SB, BUILD A FORMATIVE SYSTEM BLOCK
                     0706  2056
                     0706  2057 ;+
                     0706  2058 ; BUILD_SB allocates a system block from nonpaged pool and sets
                     0706  2059 ; it up with information from the received START or STACK datagram.
                     0706  2060 ; If insufficient pool is available, then the routine returns failure.
                     0706  2061 ;
                     0706  2062 ; Inputs:
                     0706  2063 ;
                     0706  2064 ;       R2                      -Addr of START/STACK dg
                     0706  2065 ;       R3                      -Addr of formative PB
                     0706  2066 ;       R4                      -Addr of PDT
                     0706  2067 ;
                     0706  2068 ; Outputs:
                     0706  2069 ;
                     0706  2070 ;       R0                      -0/1 for fail/success
                     0706  2071 ;       R1                      -Destroyed
                     0706  2072 ;       other registers         -Preserved
                     0706  2073 ;-
                     0706  2074
                     0706  2075 ;
                     0706  2076 ; Data structure adjacency assumptions:
                     0706  2077 ;
                     0706  2078
                     0706  2079 ASSUME  SB$B_SYSTEMID+8 EQ SB$W_MAXDG
                     0706  2080 ASSUME  SB$W_MAXDG+2    EQ SB$W_MAXMSG
                     0706  2081 ASSUME  SB$W_MAXMSG+2   EQ SB$T_SWTYPE
                     0706  2082 ASSUME  SB$T_SWTYPE+4   EQ SB$T_SWVERS
                     0706  2083 ASSUME  SB$T_SWVERS+4   EQ SB$Q_SWINCARN
                     0706  2084 ASSUME  SB$Q_SWINCARN+8 EQ SB$T_HWTYPE
                     0706  2085 ASSUME  SB$T_HWTYPE+4   EQ SB$B_HWVERS
                     0706  2086 ASSUME  SB$T_NODENAME+16 EQ SB$L_DDB
                     0706  2087
                     0706  2088 ASSUME  PPD$B_SYSTEMID+8 EQ PPD$W_MAXDG
                     0706  2089 ASSUME  PPD$W_MAXDG+2   EQ PPD$W_MAXMSG
                     0706  2090 ASSUME  PPD$W_MAXMSG+2  EQ PPD$T_SWTYPE
                     0706  2091 ASSUME  PPD$T_SWTYPE+4  EQ PPD$T_SWVERS
                     0706  2092 ASSUME  PPD$T_SWVERS+4  EQ PPD$Q_SWINCARN
                     0706  2093 ASSUME  PPD$Q_SWINCARN+8 EQ PPD$T_HWTYPE
                     0706  2094 ASSUME  PPD$T_HWTYPE+4  EQ PPD$B_HWVERS
                     0706  2095 ASSUME  PPD$Q_NODENAME+8 EQ PPD$Q_CURTIME
                     0706  2096
           0000002C  0706  2097 DATA_LEN = <SB$B_HWVERS+12> - <SB$B_SYSTEMID>
                     0706  2098
                     0706  2099          .ENABL  LSB
                     0706  2100
                     0706  2101 BUILD_SB:
                     0706  2102
              3C BB  0706  2103          PUSHR   #^M<R2,R3,R4,R5>        ; Save a bunch of registers
51 00000060 8F D0    0708  2104          MOVL    #SB$K_LENGTH,R1         ; Get size of SB
     00000000'GF 16  070F  2105          JSB     G^EXE$ALONONPAGED       ; Allocate from nonpaged pool
           54 50 E9  0715  2106          BLBC    R0,SB_DONE              ; Branch if no pool
        08 A2 51 B0  0718  2107          MOVW    R1,SB$W_SIZE(R2)        ; Set struct size
         0760 8F B0  071C  2108          MOVW    #DYN$C_SCS+<DYN$C_SCS_SB@8>,-   ; Set structure type
            0A A2    0720  2109                  SB$B_TYPE(R2)                   ;  and subtype
            0C A2 DE 0722  2110          MOVAL   SB$L_PBFL(R2),-         ; Set path block list head
            0C A2    0725  2111                  SB$L_PBFL(R2)           ;  to empty
```

```
                0C A2   DE   0727  2112          MOVAL   SB$L_PBFL(R2),-          ;
                10 A2        072A  2113                  SB$L_PBBL(R2)           ;
             51   52   D0   072C  2114          MOVL    R2,R1                   ; Copy SB addr to R1
             52   6E   D0   072F  2115          MOVL    (SP),R2                 ; Retreive dg addr
          53   04 AE   D0   0732  2116          MOVL    4(SP),R3                ;  and PB addr
          30 A3   51   D0   0736  2117          MOVL    R1,PB$L_SBLINK(R3)      ; Link new SB to PB
             1A A2   90   073A  2118          MOVB    PPD$B_PROTOCOL(R2),-    ; Save PPD protocol level in
             48 A3        073D  2119                  PB$B_PROTOCOL(R3)       ;  formative PB
             7E   51   7D   073F  2120          MOVQ    R1,-(SP)                ; Save regs destroyed by movc
                2C   28   0742  2121          MOVC3   #DATA_LEN,-             ; Copy system ID, dg and msg
                14 A2        0744  2122                  PPD$B_SYSTEMID(R2),-    ;  sizes, sw type, version,
                18 A1        0746  2123                  SB$B_SYSTEMID(R1)       ;  incarnation, HW type and version
          52   04 AE   D0   0748  2124          MOVL    4(SP),R2                ; Retreive START/STACK dg addr
             08   20   3A   074C  2125          LOCC    #^A/ /,#8,-             ; Compute # characters prior
                40 A2        074F  2126                  PPD$Q_NODENAME(R2)      ;  to blank fill
       50   08   50   C3   0751  2127          SUBL3   R0,#8,R0                ;  in node name
             51   8E   7D   0755  2128          MOVQ    (SP)+,R1                ; Retreive saved registers
          44 A1   50   90   0758  2129          MOVB    R0,SB$T_NODENAME(R1)    ; Set count of characters
             5C A1   D4   075C  2130          CLRL    SB$L_CSB(R1)            ; Zero link to newest CSB.
          40 A2   50   2C   075F  2131          MOVC5   R0,PPD$Q_NODENAME(R2),- ; Copy ASCII characters into
    45 A1   0F   00        0763  2132                  #0,#15,SB$T_NODENAME+1(R1) ;  counted string node name in SB
                63   D4   0767  2133          CLRL    (R3)                    ; Zero link to DDB chain for new SB
             50   01   3C   0769  2134          MOVZWL  #SS$_NORMAL,R0          ; Set status = success
                           076C  2135
                           076C  2136 SB_DONE:
                           076C  2137
                3C   BA   076C  2138          POPR    #^M<R2,R3,R4,R5>        ; Restore registers
                05   05   076E  2139          RSB                             ; Return
                           076F  2140
                           076F  2141          .DSABL  LSB
```

```
                        076F  2143          .SBTTL  -       BREAK_PATH,     INITIATE CRASH
                        076F  2144          .SBTTL  -                       OF VIRTUAL CIRCUIT
                        076F  2145          .SBTTL  -       BREAK_HOST,     HOST SHUTDOWN REC'D
                        076F  2146
                        076F  2147 ;+
                        076F  2148 ; BREAK_PATH is the action routine called when a START is received
                        076F  2149 ; on a VC we think is open.  The START implies that the remote system
                        076F  2150 ; has crashed the VC and that we should do the same.  Therefore, the
                        076F  2151 ; start datagram is discarded and ERR$CRASHVC is called to start
                        076F  2152 ; the process of crashing the virtual circuit.
                        076F  2153 ;
                        076F  2154 ; BREAK_HOST is the action routine called when a host shutdown
                        076F  2155 ; dg is received.  It does the same as BREAK_PATH, but saves
                        076F  2156 ; a special reason code in the path block to be used later when
                        076F  2157 ; notifying SYSAP's of the circuit failure.
                        076F  2158 ;
                        076F  2159 ; Inputs:
                        076F  2160 ;
                        076F  2161 ;       R2                      -Addr of START/Host shutdown dg
                        076F  2162 ;       R3                      -Addr of PB
                        076F  2163 ;       R4                      -Addr of PDT
                        076F  2164 ;
                        076F  2165 ; Outputs:
                        076F  2166 ;
                        076F  2167 ;       R0-R2                   -Destroyed
                        076F  2168 ;       Other registers         -Preserved
                        076F  2169 ;-
                        076F  2170
                        076F  2171          .ENABL  LSB
                        076F  2172
                        076F  2173 BREAK_HOST:
                        076F  2174
028C 8F    B0           076F  2175          MOVW    #SS$_NOSUCHNODE,-       ; Save vc fail reason for
  46 A3                 0773  2176                  PB$W_VCFAIL_RSN(R3)     ;  later reporting to SYSAPs
                        0775  2177                                          ;  as the aux status
                        0775  2178 BREAK_PATH:
                        0775  2179
   F888'   30           0775  2180          BSBW    INT$INS_DFREEQ1         ; Return dg buffer to
                        0778  2181                                          ;  free queue
51   53    D0           0778  2182          MOVL    R3,R1                   ; Transfer PB address
   F882'   31           077B  2183          BRW     ERR$CRASHVC             ; Start crash of VC on its way
                        077E  2184
                        077E  2185          .DSABL  LSB
```

```
                        077E  2187          .SBTTL  -       REC_ERROR_DG,           LOG ERROR DG
                        077E  2188
                        077E  2189 ;+
                        077E  2190 ; REC_ERROR_DG is the action routine called for an error log datagram
                        077E  2191 ; PPD type.  These are datagrams received from hosts that have minimal
                        077E  2192 ; error logging capability, do not have an scs connection over which
                        077E  2193 ; to send an application datagram containing error info, and choose to
                        077E  2194 ; send the info in one of these 'out of band' datagrams instead.
                        077E  2195 ;
                        077E  2196 ; Inputs:
                        077E  2197 ;
                        077E  2198 ;       R2                      -Address of start of dg
                        077E  2199 ;       R3                      -Address of PB
                        077E  2200 ;       R4                      -Address of PDT
                        077E  2201 ;
                        077E  2202 ; Outputs:
                        077E  2203 ;
                        077E  2204 ;       R0                      -Destroyed
                        077E  2205 ;       Other registers         -Preserved
                        077E  2206 ;-
                        077E  2207
                        077E  2208          .ENABL  LSB
                        077E  2209
                        077E  2210 REC_ERROR_DG:
                        077E  2211
      F87F'  30         077E  2212          BSBW    ELOG$ERROR_DG           ; Go log it
50  00DC  C4   D0        0781  2213          MOVL    PDT$L_UCB0(R4),R0       ; Get UCB address
    0082  C0   B7        0786  2214          DECW    UCB$W_ERRCNT(R0)        ; Decr error count incremented
                          078A  2215                                          ;  by error logger
        00   11          078A  2216          BRB     IGNORE_DG               ; Go recycle to dg free queue
                          078C  2217
                          078C  2218          .DSABL  LSB
```

```
                     078C  2220          .SBTTL  -       IGNORE_DG,               DISCARD DATAGRAM WITHOUT ACTION
                     078C  2221
                     078C  2222 ;+
                     078C  2223 ; IGNORE_DG is the action routine called for received start handshake datagrams
                     078C  2224 ; for a path block with VC failure in progress.  The datagram is returned to
                     078C  2225 ; the free queue and no further action taken.
                     078C  2226 ;
                     078C  2227 ; Inputs:
                     078C  2228 ;
                     078C  2229 ;       R2                      -Addr of handshake dg
                     078C  2230 ;
                     078C  2231 ; Outputs:
                     078C  2232 ;
                     078C  2233 ;       R0                      -Destroyed
                     078C  2234 ;       Other registers         -Preserved
                     078C  2235 ;-
                     078C  2236
                     078C  2237          .ENABL  LSB
                     078C  2238
                     078C  2239 IGNORE_DG:
                     078C  2240
F871'  31            078C  2241          BRW     INT$INS_DFREEQ1         ; Return dg to free queue
                     078F  2242
                     078F  2243          .DSABL  LSB
```

```
                        2245          .SBTTL  UTILITY ROUTINES
                        2246          .SBTTL  -       FMT_START_DATA, FORMAT START DATA IN A
                        2247          .SBTTL  -                       START/STACK DATAGRAM
                        2248
                        2249 ;+
                        2250 ; FMT_START_DATA fills in the start data in a STACK or START datagram.
                        2251 ; Data is drawn from sysgen paramters, SCS global locations, the
                        2252 ; system ID register, and constants.
                        2253 ;
                        2254 ; Inputs:
                        2255 ;
                        2256 ;       R2                      -Addr of datagram
                        2257 ;       R3                      -Addr of PB
                        2258 ;       R4                      -Addr of PDT
                        2259 ;
                        2260 ; Outputs:
                        2261 ;
                        2262 ;       R0,R1                   -Destroyed
                        2263 ;       other registers         -Preserved
                        2264 ;-
                        2265 ;
                        2266 ; Message format adjacency assumptions:
                        2267 ;
                        2268
                        2269          ASSUME  PPD$B_SYSTEMID+6 EQ PPD$B_PROTOCOL
                        2270          ASSUME  PPD$B_PROTOCOL+2 EQ PPD$W_MAXDG
                        2271          ASSUME  PPD$W_MAXDG+2    EQ PPD$W_MAXMSG
                        2272          ASSUME  PPD$W_MAXMSG+2   EQ PPD$T_SWTYPE
                        2273          ASSUME  PPD$T_SWTYPE+4   EQ PPD$T_SWVERS
                        2274          ASSUME  PPD$T_SWVERS+4   EQ PPD$Q_SWINCARN
                        2275          ASSUME  PPD$Q_SWINCARN+8 EQ PPD$T_HWTYPE
                        2276          ASSUME  PPD$T_HWTYPE+4   EQ PPD$B_HWVERS
                        2277          ASSUME  PPD$B_HWVERS+12 EQ PPD$Q_NODENAME
                        2278          ASSUME  PPD$Q_NODENAME+8 EQ PPD$Q_CURTIME
                        2279          ASSUME  PPD$Q_CURTIME+8 EQ PPD$C_MIN_DGSIZ
                        2280
                        2281          .ENABL  LSB
                        2282
                        2283 FMT_START_DATA:
                        2284
      50   14 A2   DE   2285          MOVAL   PPD$B_SYSTEMID(R2),R0   ; Get system ID field addr
80  00000000'GF   7D   2286          MOVQ    G^SCS$GB_SYSTEMID,(R0)+ ; Copy system ID
       FE A0    01   9B   2287          MOVZBW  #PPD$C_PRT_ELOG,-2(R0)  ; Set current protocol rev supported
80  00000000'GF   D0   2288          MOVL    G^SCS$GW_MAXDG,(R0)+    ; Specify max bytes of dg and
                        2289                                          ;  msg application data
80  20534D56 8F   D0   2290          MOVL    #^A/VMS /,(R0)+         ; Set operating system name
80  00000000'GF   D0   2291          MOVL    G^SYS$GQ_VERSION,(R0)+  ; Set operating system version
    0000002C'GF   7D   2292          MOVQ    G^SCS$GA_LOCALSB+ -
                  80   2293                  SB$Q_SWINCARN,(R0)+     ; Set system boot seq #
80  00000000'EF   D0   2294          MOVL    INI$T_HWTYPE,(R0)+      ; Set processor name
80  00000000'GF   7D   2295          MOVQ    G^EXE$GB_CPUDATA,(R0)+  ; Copy CPU data (hardware/ ucode
80  00000008'GF   D0   2296          MOVL    G^EXE$GB_CPUDATA+8,(R0)+ ;  rev levels)
80  00000000'GF   7D   2297          MOVQ    G^SCS$GB_NODENAME,(R0)+ ; Null node name, blank filled
80  00000000'GF   7D   2298          MOVQ    G^EXE$GQ_SYSTIME,(R0)+  ; Set current system time
                  05   2299          RSB                             ; Return
                       2300
                       2301          .DSABL  LSB
```

```
                        07DE  2303          .SBTTL  -       CLEANUP, REMOVE FORMATIVE PB AND SB
                        07DE  2304
                        07DE  2305 ;+
                        07DE  2306 ; CLEANUP is called by the ACTION_DISP routine when fail status
                        07DE  2307 ; has been returned by an action routine.  The action routine
                        07DE  2308 ; detecting the error is expected to perform all cleanup other
                        07DE  2309 ; than deleting the formative path block and system block.  CLEANUP
                        07DE  2310 ; deletes the formative system block (if any) and formative
                        07DE  2311 ; path block.  The start handshake is simply abandoned to be
                        07DE  2312 ; restarted by a new IDREC later.
                        07DE  2313 ;
                        07DE  2314 ; Inputs:
                        07DE  2315 ;
                        07DE  2316 ;       R3                      -Addr of formative PB
                        07DE  2317 ;       R4                      -Addr of PDT
                        07DE  2318 ;
                        07DE  2319 ; Outputs:
                        07DE  2320 ;
                        07DE  2321 ;       R0                      -Destroyed
                        07DE  2322 ;       other registers         -Preserved
                        07DE  2323 ;-
                        07DE  2324
                        07DE  2325          .ENABL  LSB
                        07DE  2326
                        07DE  2327 CLEANUP:
                        07DE  2328
     50    30 A3   D0   07DE  2329          MOVL    PB$L_SBLINK(R3),R0      ; Get addr of formative SB
           02      13   07E2  2330          BEQL    10$                     ; Branch if none
           11      10   07E4  2331          BSBB    CLEAN2                  ; Else deallocate SB
                        07E6  2332
        0C A3      E5   07E6  2333 10$:     BBCC    PB$B_RSTATION(R3),-     ; Mark no PB in path map
     00 0114 C4         07E9  2334                  PDT$B_PORTMAP(R4),20$   ;
        019A C4    B7   07ED  2335 20$:     DECW    PDT$W_STDGUSED(R4)      ; Decr count of # ports likely
                        07F1  2336                                          ;  to send IDREC's and need
                        07F1  2337                                          ;  start handshake
           011D    30   07F1  2338          BSBW    LB_ENABLE               ; Enable loopback dg's if necessary
        50    63    0F   07F4  2339          REMQUE  (R3),R0                 ; Remove PB from formative list
                        07F7  2340
00000000'GF         16   07F7  2341 CLEAN2:  JSB     G^COM$DRVDEALMEM        ; Deallocate PB
                    05   07FD  2342          RSB                             ; Return
                        07FE  2343
                        07FE  2344          .DSABL  LSB
```

```
                        07FE  2346          .SBTTL  -       SEARCH_PATHS, SEARCH FOR PB WITH STATION ADDR MATCH
                        07FE  2347
                        07FE  2348 ;+
                        07FE  2349 ; SEARCH_PATHS searches a doubly linked list of PB's for the first
                        07FE  2350 ; PB with station address matching a specified station address.  The
                        07FE  2351 ; match is done only on the low order 8 bits of station address since
                        07FE  2352 ; CI station addresses are known to fit in 8 bits.
                        07FE  2353 ;
                        07FE  2354 ; Inputs:
                        07FE  2355 ;
                        07FE  2356 ;       R1                      -Station address to match
                        07FE  2357 ;       R3                      -Addr of PB listhead
                        07FE  2358 ;
                        07FE  2359 ; Outputs:
                        07FE  2360 ;
                        07FE  2361 ;       R0                      -0/1 for fail/success on search
                        07FE  2362 ;       R3                      -PB address if success
                        07FE  2363 ;       other registers         -Preserved
                        07FE  2364 ;-
                        07FE  2365
                        07FE  2366          .ENABL  LSB
                        07FE  2367
                        07FE  2368 SEARCH_PATHS:
                        07FE  2369
 50   53   D0            07FE  2370          MOVL    R3,R0                   ; Hold start point
                        0801  2371
                        0801  2372 SEARCH_CONT:
                        0801  2373
 53   63   D0            0801  2374          MOVL    (R3),R3                 ; Get next PB
 50   53   D1            0804  2375          CMPL    R3,R0                   ; Back at start?
           0A   13        0807  2376          BEQL    20$                     ; Branch if so
51    0C  A3   91        0809  2377          CMPB    PB$B_RSTATION(R3),R1    ; Low byte matches?
           F2   12        080D  2378          BNEQ    SEARCH_CONT             ; Branch if not
 50   01   3C            080F  2379          MOVZWL  #SS$_NORMAL,R0          ; Else return success
                05        0812  2380          RSB                             ; Return
                        0813  2381
           50   D4        0813  2382 20$:     CLRL    R0                      ; Status - fail
                05        0815  2383          RSB                             ; Return
                        0816  2384
                         316  2385          .DSABL  LSB
```

```
                        0816  2387          .SBTTL  -       CNF$LKP_PB_MSG, LOOK UP THE PB CORRESPONDING
                        0816  2388          .SBTTL  -                       TO A PDT AND REMOTE STATION ADDR
                        0816  2389
                        0816  2390 ;+
                        0816  2391 ; CNF$LKP_PB_MSG extracts the remote station addr from a received message
                        0816  2392 ; and looks through the system wide configuration database for the
                        0816  2393 ; PB correxponding to the remote station and PDT.  Only the low order
                        0816  2394 ; 8 bits of the station address are matched since CI station addresses
                        0816  2395 ; always fit in 8 bits.
                        0816  2396 ;
                        0816  2397 ; Inputs:
                        0816  2398 ;
                        0816  2399 ;       R2                      -Addr of message
                        0816  2400 ;       R4                      -Addr of PDT
                        0816  2401 ;
                        0816  2402 ; Outputs:
                        0816  2403 ;
                        0816  2404 ;       R0                      -0/1 for fail/success on search
                        0816  2405 ;       R1                      -PB addr if success
                        0816  2406 ;       Other registers         -Preserved
                        0816  2407 ;-
                        0816  2408
                        0816  2409          .ENABL  LSB
                        0816  2410
                        0816  2411 CNF$LKP_PB_MSG2::
                        0816  2412
    51    52   00B4 C4   C3  0816  2413          SUBL3   PDT$L_MSGHDRSZ(R4),R2,R1; Back up to top of PPD layer
          51     0C A1   9A  081C  2414          MOVZBL  PPD$B_PORT(R1),R1       ; Get remote station addr
                    04   11  0820  2415          BRB     5$
                        0822  2416
                        0822  2417 CNF$LKP_PB_MSG::
                        0822  2418
          51     0C A2   9A  0822  2419          MOVZBL  PPD$B_PORT(R2),R1       ; Get remote station addr
                        0826  2420
                    55  DD  0826  2421 5$:       PUSHL   R5                      ; Save a couple of registers
                    53  DD  0828  2422           PUSHL   R3                      ;
    55    00000000'GF   DE  082A  2423           MOVAL   G^SCS$GQ_CONFIG,R5      ; Get addr of listhead for system
                        0831  2424                                           ;  configuration database
                        0831  2425
          55    65    D0  0831  2426 10$:       MOVL    (R5),R5                 ; Get next system block
00000000'8F     55    D1  0834  2427           CMPL    R5,#SCS$GQ_CONFIG       ; Back at start of list?
                    21  13  083B  2428           BEQL    PB_NOT_FOUND            ; Branch if so
          53    0C A5   DE  083D  2429           MOVAL   SB$L_PBFL(R5),R3        ; Get addr of PB listhead
                    BB  10  0841  2430           BSBB    SEARCH_PATHS            ; See if there is matching station
                        0843  2431
          EB    50  E9  0843  2432 20$:       BLBC    R0,10$                  ; Branch if no matching station
          54    2C A3   D1  0846  2433           CMPL    PB$L_PDT(R3),R4         ; Is this path block a path from
                        084A  2434                                           ;  the same PDT?
                    08  13  084A  2435           BEQL    PB_FOUND                ; Branch if yes
          50    0C A5   DE  084C  2436           MOVAL   SB$L_PBFL(R5),R0        ; Else set up PB listhead addr again
                    AF  10  0850  2437           BSBB    SEARCH_CONT             ; Continue PB search
                    EF  11  0852  2438           BRB     20$                     ;  and check results
                        0854  2439
                        0854  2440 PB_FOUND:
                        0854  2441
             51   53   D0  0854  2442           MOVL    R3,R1                   ; Move PB addr to R1
                        0857  2443
```

```
53 8ED0  0857  2444 30$:     POPL    R3                      ; Retreive caller's R3
55 8ED0  085A  2445           POPL    R5                      ;  and R5
     05  085D  2446           RSB                             ; Return
         085E  2447
         085E  2448 PB_NOT_FOUND:
         085E  2449
50   7C  085E  2450           CLRQ    R0                      ; Show failure status
F5   11  0860  2451           BRB     30$                     ; Join common exit
         0862  2452
         0862  2453           .DSABL  LSB
```

```
                        0862  2455          .SBTTL  -       CNF$LKP_PB_PDT, LOOK UP FIRST/NEXT
                        0862  2456          .SBTTL  -                       PB ASSOC WITH PDT
                        0862  2457
                        0862  2458 ;+
                        0862  2459 ; CNF$LKP_PB_PDT looks through the configuration database for PB's
                        0862  2460 ; associated with a specified PDT.  For each one found, the caller is
                        0862  2461 ; called back with the PB address in R3.  When the whole database has
                        0862  2462 ; been searched, return is taken to the caller with failure status in R0.
                        0862  2463 ;
                        0862  2464 ; This routine is called during power failure to cleanup PB's and SB's
                        0862  2465 ; associated with the local failing port.  Therefore, when a PB is
                        0862  2466 ; delivered to the caller, the PB and its SB may have been deleted
                        0862  2467 ; upon return from the coroutine.  The forward links to the next PB and
                        0862  2468 ; next SB in the configuration database will be destroyed in this case.
                        0862  2469 ; Whenever an SB is being processed, the link to the next SB is saved on
                        0862  2470 ; the stack.  When a PB is about to be delivered to the coroutine, the
                        0862  2471 ; link to the next PB is saved on the stack and, upon return, the saved
                        0862  2472 ; link used as the address of the next PB to look at.
                        0862  2473 ;
                        0862  2474 ; Inputs:
                        0862  2475 ;
                        0862  2476 ;       R4                      -PDT addr
                        0862  2477 ;
                        0862  2478 ; Outputs:
                        0862  2479 ;
                        0862  2480 ;       R0                      -Status:  LBS/C if PB found/not found
                        0862  2481 ;       R3                      -PB addr if success
                        0862  2482 ;       R1,R2                   -Destroyed
                        0862  2483 ;       Other registers         -Preserved
                        0862  2484 ;-
                        0862  2485
                        0862  2486 ASSUME  PB$L_FLINK       EQ 0
                        0862  2487 ASSUME  SB$L_FLINK       EQ 0
                        0862  2488
                        0862  2489          .ENABL  LSB
                        0862  2490
                        0862  2491 CNF$LKP_PB_PDT::
                        0862  2492
52   00000000'GF    DE  0862  2493          MOVAL   G^SCS$GQ_CONFIG,R2      ; Get configuration database ptr
        52    62    D0  0869  2494          MOVL    (R2),R2                 ; Get next system blk
                        086C  2495
00C00000'8F     52    D1  086C  2496 10$:     CMPL    R2,#SCS$GQ_CONFIG       ; Back at header?
                2D    13  0873  2497          BEQL    NOT_FOUND               ; Branch if so
                62    DD  0875  2498          PUSHL   (R2)                    ; Save link to next SB
      53    0C    A2    DE  0877  2499          MOVAL   SB$L_PBFL(R2),R3        ; Get PB list header
         51   53    D0  087B  2500          MOVL    R3,R1                   ; Save listhead addr
                        087E  2501
         53   63    D0  087E  2502 20$:     MOVL    (R3),R3                 ; Get next PB
                        0881  2503
         51   53    D1  0881  2504 30$:     CMPL    R3,R1                   ; Back at start of list?
                17    13  0884  2505          BEQL    NEXT_SB                 ; Branch if so -- move to next SB
      54    2C    A3    D1  0886  2506          CMPL    PB$L_PDT(R3),R4         ; Is PB on this PDT?
                F2    12  088A  2507          BNEQ    20$                     ; Branch if not
         50   01    3C  088C  2508          MOVZWL  #SS$_NORMAL,R0          ; Set success status for caller
                        088F  2509                                          ;  coroutine
                63    DD  088F  2510          PUSHL   (R3)                    ; Save link to next PB
                06    BB  0891  2511          PUSHR   #^M<R1,R2>              ; Save registers caller destroys
```

```
10 BE   16  0893  2512          JSB     @<4*4>(SP)              ; Call caller back to process PB
            0896  2513                                          ;  (There are 2 flinks and 2
            0896  2514                                          ;  registers saved on the stack)
   06   BA  0896  2515          POPR    #^M<R1,R2>              ; Restore registers
   53 8ED0  0898  2516          POPL    R3                      ; Retreive addr of next PB
   E4   11  089B  2517          BRB     30$                     ; Check next PB
            089D  2518
            089D  2519 NEXT_SB:
            089D  2520
   52 8ED0  089D  2521          POPL    R2                      ; Retreive addr of next SB
   CA   11  08A0  2522          BRB     10$                     ; Check next SB
            08A2  2523 NOT_FOUND:
            08A2  2524
   50   D4  08A2  2525          CLRL    R0                      ; Set fail status for caller coroutine
        05  08A4  2526          RSB                             ; Return to caller
            08A5  2527
            08A5  2528          .DSABL  LSB
```

```
                     08A5  2530          .SBTTL  -       CNF$REMOVE_PB,  REMOVE PB(SB) FROM
                     08A5  2531          .SBTTL  -                       CONFIG DATABASE
                     08A5  2532
                     08A5  2533 ;+
                     08A5  2534 ; CNF$REMOVE_PB is called by ERR$VCCLOSED_MSG/PB or ERR$VC_CACHECLR
                     08A5  2535 ; when all connections associated with a failing path block have
                     08A5  2536 ; been cleaned up.  CNF$REMOVE_PB marks the remote port as unknown in
                     08A5  2537 ; the port bitmap.  If this is a virutal circuit failure  due to reasons
                     08A5  2538 ; other than local port/system power failure, then the path block SCS
                     08A5  2539 ; receive buffer and, if available, the SCS send buffer, are reclaimed from
                     08A5  2540 ; the message free queue and returned to pool.  In the case of a power
                     08A5  2541 ; failure this step is omitted because all queue elements for all
                     08A5  2542 ; paths on the local port are collected together later.
                     08A5  2543 ;
                     08A5  2544 ; Finally, the path block is unlinked from the system block.  If this
                     08A5  2545 ; leaves the SB with no paths, then the SB link to the next PB to
                     08A5  2546 ; use in a connection is zeroed.  The PB is returned to pool and return taken.
                     08A5  2547 ;
                     08A5  2548 ; Inputs:
                     08A5  2549 ;
                     08A5  2550 ;       IPL                     -Fork IPL
                     08A5  2551 ;
                     08A5  2552 ;       R3                      -PB addr
                     08A5  2553 ;       R4                      -PDT addr
                     08A5  2554 ;
                     08A5  2555 ; Outputs:
                     08A5  2556 ;
                     08A5  2557 ;       R0-R2                   -Destroyed
                     08A5  2558 ;       Other registers         -Preserved
                     08A5  2559 ;-
                     08A5  2560
                     08A5  2561          .ENABL  LSB
                     08A5  2562
                     08A5  2563 CNF$REMOVE_PB::
                     08A5  2564
      34 A3    D5    08A5  2565          TSTL    PB$L_CDTLST(R3)         ; Verify no CDT's remain
         03    13    08A8  2566          BEQL    10$                     ; Branch if none do
       FA18    31    08AA  2567          BRW     CONFIG_ERR              ; Else inconsistent database
                     08AD  2568
      0C A3    E5    08AD  2569 10$:     BBCC    PB$B_RSTATION(R3),-     ; Mark the remote port unknown
00 0114 C4           08B0  2570                  PDT$B_PORTMAP(R4),20$   ;  to poller
                     08B4  2571
    019A C4    B7    08B4  2572 20$:     DECW    PDT$W_STDGUSED(R4)      ; Decr #ports that will likely
                     08B8  2573                                          ;  send us IDREC's for a while
        0056    30   08B8  2574          BSBW    LB_ENABLE               ; Enable loopback dg's if necessary
    0112 C4    B7    08BB  2575          DECW    PDT$W_PBCOUNT(R4)       ; Decr count of PB's on this PDT
      12 A3    B1    08BF  2576          CMPW    PB$W_STATE(R3),-        ; Is this a power fail recovery?
    4000 8F          08C2  2577                  #PB$C_PWR_FAIL          ;
         13    13    08C5  2578          BEQL    40$                     ; Branch if so
52    40 A3    D0    08C7  2579          MOVL    PB$L_SCSMSG(R3),R2      ; Else get SCS send buffer
         07    12    08CB  2580          BNEQ    30$                     ; Branch if available
       F730'   30    08CD  2581          BSBW    INT$MFQ2POOL            ; If unavailable, get it from
         08    1D    08D0  2582          BVS     40$                     ;  message free queue
         03    11    08D2  2583          BRB     35$
                     08D4  2584
       F729'   30    08D4  2585 30$:     BSBW    INT$DEAL_MSG            ; Deallocate to pool
       F726'   30    08D7  2586 35$:     BSBW    INT$MFQ2POOL            ; Get SCS receive buffer from free q
```

```
                     08DA  2587
52   54 A3   D0     08DA  2588 40$:    MOVL    PB$L_CLSCKT_DG(R3),R2   ; Get CLSCKT dg addr
       03   13     08DE  2589          BEQL    45$                     ; Branch if none
     F71D'  30     08E0  2590          BSBW    INT$DEAL_DG1            ; Else return to pool
                     08E3  2591
50   30 A3   D0     08E3  2592 45$:    MOVL    PB$L_SBLINK(R3),R0      ; Get addr of this path's SB
53   14 A0   D1     08E7  2593          CMPL    SB$L_PBCONNX(R0),R3     ; Is SB ptr to next PB to use for
                     08EB  2594                                          ;  a connection we are removing?
         04   12     08EB  2595          BNEQ    46$                     ; Branch if not
         63   D0     08ED  2596          MOVL    PB$L_FLINK(R3),-        ; Else patch SB to point to
       14 A0        08EF  2597                  SB$L_PBCONNX(R0)        ;  next path if any
                     08F1  2598
   53    63   0F     08F1  2599 46$:    REMQUE  (R3),R3                 ; Remove PB from PB list
         03   12     08F4  2600          BNEQ    50$                     ; Branch if not last PB
       14 A0   D4     08F6  2601          CLRL    SB$L_PBCONNX(R0)        ; Zero link to next connx to use
                     08F9  2602
50    53   D0     08F9  2603 50$:    MOVL    R3,R0                   ; Copy PB addr for deallocation
     FEF8  31     08FC  2604          BRW     CLEAN2                  ; Deallocate PB to pool
                     08FF  2605
                     08FF  2606          .DSABL  LSB
```

```
                     08FF  2608          .SBTTL  -       SNDDG_RET,      SEND DG, RETURN BUFFER
                     08FF  2609          .SBTTL  -                       TO RESPONSE QUEUE
                     08FF  2610          .SBTTL  -       SNDDG_NORET,    SEND DG, RETURN BUFFER
                     08FF  2611          .SBTTL  -                       TO FREE QUEUE
                     08FF  2612
                     08FF  2613 ;+
                     08FF  2614 ; The datagram is put on the low priority command queue with
                     08FF  2615 ; the response flag set/clear for the SEND_RET/NORET call.
                     08FF  2616 ;
                     08FF  2617 ; Inputs:
                     08FF  2618 ;
                     08FF  2619 ;       R2                      -Addr of dg buffer
                     08FF  2620 ;       R3                      -Addr of PB
                     08FF  2621 ;       R4                      -Addr of PDT
                     08FF  2622 ;
                     08FF  2623 ; Outputs:
                     08FF  2624 ;
                     08FF  2625 ;       R0                      -Destroyed
                     08FF  2626 ;       Other registers         -Preserved
                     08FF  2627 ;-
                     08FF  2628
                     08FF  2629          .ENABL  LSB
                     08FF  2630
                     08FF  2631 SNDDG_RET:
                     08FF  2632
 51    53    D0      08FF  2633          MOVL    R3,R1                   ; Transfer PB address
 50    02    D0      0902  2634          MOVL    #SYSAP$C_DISPPO,R0      ; RETFLAG=TRUE, DISP=POOL
     F6F8'   31      0905  2635          BRW     INT$SNDDG1              ; Send it
                     0908  2636
                     0908  2637 SNDDG_NORET:
                     0908  2638
 51    53    D0      0908  2639          MOVL    R3,R1                   ; Transfer PB address
 50    00    D0      090B  2640          MOVL    #SYSAP$C_DISPQ,R0       ; RETFLAG=FALSE
     F6EF'   31      090E  2641          BRW     INT$SNDDG1              ; Send it
                     0911  2642
                     0911  2643          .DSABL  LSB
```

```
                        0911  2645          .SBTTL  -       LB_ENABLE,      ENABLE LB DG SENDS
                        0911  2646          .SBTTL  -                       IF NECESSARY
                        0911  2647
                        0911  2648 ;+
                        0911  2649 ; Called whenever a virtual circuit is lost to check and see if
                        0911  2650 ; there are now no remote ports known besides self.  (Known means
                        0911  2651 ; virtual circuits open or formative paths.) If there are no remote
                        0911  2652 ; ports known besides self, then the loopback dg test is enabled.
                        0911  2653 ; Otherwise, the loopback test flag is left alone.
                        0911  2654 ;
                        0911  2655 ; Inputs:
                        0911  2656 ;
                        0911  2657 ;       R4                          -PDT addr
                        0911  2658 ;       PDT$B_PORTMAP(R4)           -32 byte bit map of known ports
                        0911  2659 ;       PDT$B_PORT_NUM(R4)          -# of local port
                        0911  2660 ;
                        0911  2661 ; Outputs:
                        0911  2662 ;
                        0911  2663 ;       R0                          -Destroyed
                        0911  2664 ;       Other registers             -Preserved
                        0911  2665 ;       PDT$W_LPORT_STS             -PDT$M_LBDG set if no other
                        0911  2666 ;                                    ports known; else unchanged
                        0911  2667 ;-
                        0911  2668
                        0911  2669          .ENABL  LSB
                        0911  2670
                        0911  2671 LB_ENABLE:
                        0911  2672
7E      51      7D      0911  2673          MOVQ    R1,-(SP)                ; Save two registers for caller
        52      D4      0914  2674          CLRL    R2                      ; Zero count of # bytes in map
                        0916  2675
        51      01      CE    0916  2676 10$:     MNEGL   #1,R1                   ; Init prev known port #, modulo 32
                        0919  2677
                51      D6    0919  2678 20$:     INCL    R1                      ; Incr prev known port #, mod 32
            20  51      EA    091B  2679          FFS     R1,#32,-                ; Find next known port, mod 32
    51  0114 C442           091E  2680                  PDT$B_PORTMAP(R4)[R2],R1 ;  in this longwd of port map
                10      13    0923  2681          BEQL    40$                     ; Branch if none found
  50    52      08      78    0925  2682          ASHL    #32/4,R2,R0             ; Convert port # mod 32 to
        51      50      C0    0929  2683          ADDL    R0,R1                   ;  actual port number
017D    C4      51      91    092C  2684          CMPB    R1,PDT$B_PORT_NUM(R4)   ; Is known port = self?
                E6      13    0931  2685          BEQL    20$                     ; Branch if so to search more
                0D      11    0933  2686          BRB     50$                     ; Else return without doing anything
                        0935  2687
        52      04      C0    0935  2688 40$:     ADDL    #4,R2                   ; Step offset in port map to next longwd
        20      52      D1    0938  2689          CMPL    R2,#32                  ; Past last longwd in map?
                D9      1F    093B  2690          BLSSU   10$                     ; Branch if not
                04      A8    093D  2691          BISW    #PDT$M_LBDG,-           ; Else no port other than
            0110 C4           093F  2692                  PDT$W_LPORT_STS(R4)     ;  self known, so enable LB dgs
                        0942  2693
        51      8E      7D    0942  2694 50$:     MOVQ    (SP)+,R1                ; Restore caller's registers
                        05    0945  2695          RSB                             ; Return
                        0946  2696
                        0946  2697          .DSABL  LSB
```

```
                     0946  2699          .SBTTL  -       CHECK_PORT_REV,         CHECK PORT
                     0946  2700          .SBTTL  -                               UCODE REV LEVEL
                     0946  2701
                     0946  2702 ;+
                     0946  2703 ; Given and IDREC packet, check the port RAM and ROM rev levels
                     0946  2704 ; to make sure they are adequate.  If not, log an error, print a
                     0946  2705 ; message on OPA0, and (for now) continue.
                     0946  2706 ;
                     0946  2707 ; The algorithm for checking is to look up the ROM/RAM level read
                     0946  2708 ; from the ID in a table of legal ROM/REAM combinations.  If it isn't
                     0946  2709 ; in the table, then check to see if either the ROM or RAM level
                     0946  2710 ; exceeds the maximum the table knows about.  If either exceeds
                     0946  2711 ; the maximum in the table.  If either exceeds the max, do no
                     0946  2712 ; futher checking on the assumption that new ucode is being run
                     0946  2713 ; that VMS hasn't been taught to judge.  If neither exceeds the
                     0946  2714 ; max, then the ucode fails the test.
                     0946  2715 ;
                     0946  2716 ; If the rev level is found in the legal table, then check the
                     0946  2717 ; cautionary rev table to see if we should print a warning before
                     0946  2718 ; continuing.  A flag is set in the cautionary table for rev's
                     0946  2719 ; which are known to have problems, but which have not yet been
                     0946  2720 ; replaced by the fixed ucode in the field yet.  The cautionary
                     0946  2721 ; message on OPA0 alerts customers to ask field service to install
                     0946  2722 ; fixes.
                     0946  2723 ;
                     0946  2724 ; To add new legal rev combinations to the table, patch or extend
                     0946  2725 ; LEGAL_REV_TABLE with the new legal combination(s), and patch
                     0946  2726 ; MAX_RAM/ROM_REV.
                     0946  2727 ;
                     0946  2728 ; Inputs:
                     0946  2729 ;
                     0946  2730 ;       R2                      -Addr of IDREC packet
                     0946  2731 ;       R4                      -PDT addr
                     0946  2732 ;
                     0946  2733 ; Outputs:
                     0946  2734 ;
                     0946  2735 ;       R0                      -Destroyed
                     0946  2736 ;       Other registers         -Preserved
                     0946  2737 ;-
                     0946  2738
                     0946  2739 LEGAL_REV_TABLE:
                     0946  2740
                     0946  2741                                              ; .WORD n,n = RAM/ROM level
                     0946  2742
0002 0002            0946  2743          .WORD   2,2                         ; Current as of June, 1984
0003 0003            094A  2744          .WORD   3,3                         ; Next rev known to need fixes
                     094E  2745                                              ;  in both RAM and ROM
0000 0000            094E  2746          .WORD   0,0                         ; Patch space for future revs
0000 0000            0952  2747          .WORD   0,0
                     0956  2748
 00000004            0956  2749 REV_TABLE_SIZ = <.- LEGAL_REV_TABLE>/4
                     0956  2750
                     0956  2751 CAUTION_REV:
                     0956  2752
                     0956  2753                                              ; .BYTE nonzero/0 for caution/
                     0956  2754                                              ;  caution message needed
          00         0956  2755          .BYTE   0                           ; Rev 2,2 -- no caution
```

```
                     00  0957  2756          .BYTE   0                       ; Rev 3,3 -- no caution
                     00  0958  2757          .BYTE   0                       ; Future revs...
                     00  0959  2758          .BYTE   0
                         095A  2759
                         095A  2760 MAX_RAM_REV:
                         095A  2761
                   0003  095A  2762          .WORD   3                       ; Max RAM level in table
                         095C  2763
                         095C  2764 MAX_ROM_REV:
                         095C  2765
                   0003  095C  2766          .WORD   3                       ; Max ROM level in table
                         095E  2767
                         095E  2768          .ENABL  LSB
                         095E  2769
                         095E  2770 CHECK_PORT_REV:
                         095E  2771
              31   BB    095E  2772          PUSHR   #^M<R0,R4,R5>           ; Save caller's registers
55   00DC C4   D0    0960  2773          MOVL    PDT$L_UCB0(R4),R5       ; Get UCB in case error logging needed
  51   DE AF   DE    0965  2774          MOVAL   LEGAL_REV_TABLE,R1      ; Get addr of legal rev table
        50   D4    0969  2775          CLRL    R0                      ; Zero index into table
                         096B  2776
 1C A2   81   D1    096B  2777 10$:     CMPL    (R1)+,PPD$L_RPORT_REV(R2)  ; Is rev being checked in table?
        22   13    096F  2778          BEQL    CHECK_CAUTION           ; Branch if so
 F6 50   04   F2    0971  2779          AOBLSS  #REV_TABLE_SIZ,R0,10$   ; Branch if not, continue check
      1E A2   B1    0975  2780          CMPW    PPD$L_RPORT_REV+2(R2),- ; Is RAM level bigger than we know about?
      E0 AF         0978  2781                  MAX_RAM_REV
        23   1A    097A  2782          BGTRU   REV_OK                  ; Branch if so
      1C A2   B1    097C  2783          CMPW    PPD$L_RPORT_REV(R2),-   ; Is ROM level bigger than we know about?
      DB AF         097F  2784                  MAX_ROM_REV
        1C   1A    0981  2785          BGTRU   REV_OK                  ; Branch if so
     F67A'   30    0983  2786          BSBW    ELOG$UCODE_ERR          ; Log prcblem
00000000'EF   94    0986  2787          CLRB    INI$PORT_REV            ; Clear port rev okay flag to force
                         098C  2788                                  ;  more informative UCODEREV bugcheck
                         098C  2789                                  ;  if a bugcheck is done
    0080 C5   94    098C  2790          CLRB    UCB$B_ERTCNT(R5)        ; Take away all port's retries
     F66D'   30    0990  2791          BSBW    ERR$CRASH_PORT          ; Go crash port permanently
                         0993  2792
                         0993  2793 CHECK_CAUTION:
                         0993  2794
51   C0 AF   DE    0993  2795          MOVAL   CAUTION_REV,R1          ; Get addr of table of caution flags
      6140   95    0997  2796          TSTB    (R1)[R0]                ; Rev legal, check if caution msg needed
        03   13    099A  2797          BEQL    REV_OK                  ; Branch if completely okay
      F661'   30    099C  2798          BSBW    ELOG$UCODE_WARN         ; Log warning
                         099F  2799
                         099F  2800 REV_OK:
                         099F  2801
        32   BA    099F  2802          POPR    #^M<R1,R4,R5>           ; Restore caller's registers
             05    09A1  2803          RSB                             ; Return.
                         09A2  2804
                         09A2  2805          .DSABL  LSB
```

```
                        09A2  2807          .SBTTL  CNF$TIMER,      PERIODIC WAKEUP ROUTINE
                        09A2  2808          .SBTTL  CNF$CALCINTDUE, RESET WAKEUP DUE TIME
                        09A2  2809
                        09A2  2810 ;+
                        09A2  2811 ; CNF$TIMER is called from exec module TIMESCHDL once per n
                        09A2  2812 ; seconds, where n is the basic CI interval timeout.  Timer
                        09A2  2813 ; intervals are specifed in SYSGEN as follows:
                        09A2  2814 ;
                        09A2  2815 ;  Parameter name                Units                      Variable name
                        09A2  2816 ;
                        09A2  2817 ;       PASIMTOUT        seconds (2, 2^15-1)      SCS$GW_PASTMOUT
                        09A2  2818 ;       PAPOLLINTERVAL   seconds (2, 2^15-1)      SCS$GW_PAPOLINT
                        09A2  2819 ;       PAPOOL_INTERVAL  seconds (2, 2^15-1)      SCS$GW_PAPOOLIN
                        09A2  2820 ;
                        09A2  2821 ; Note that if the poller interval and pool checking interval are not
                        09A2  2822 ; exact multiples of the basic interval, then they will be effecitvely
                        09A2  2823 ; rounded up to the nearest multiple of the basic interval.  The basic
                        09A2  2824 ; interval is equal to the start handshake timeout interval.
                        09A2  2825 ;
                        09A2  2826 ; Inputs:
                        09A2  2827 ;
                        09A2  2828 ;       R3                       -Addr of CRB
                        09A2  2829 ;       IPL                      -IPL$_POWER
                        09A2  2830 ;
                        09A2  2831 ; Outputs:
                        09A2  2832 ;
                        09A2  2833 ;       IPL                      -IPL$_SCS
                        09A2  2834 ;       R0-R2,R4,R5              -Destroyed
                        09A2  2835 ;       Other registers          -Preserved
                        09A2  2836 ;
                        09A2  2837 ; Entry CNF$CALCINTDUE computes the due time for the next basic interval wakeup.
                        09A2  2838 ; It expects as inputs R3/CRB, R4/PDT and destroys R0.
                        09A2  2839 ;
                        09A2  2840 ;-
                        09A2  2841
                        09A2  2842          .ENABL  LSB
                        09A2  2843
                        09A2  2844 CNF$TIMER::
                        09A2  2845
  54    10 A3    D0     09A2  2846          MOVL    CRB$L_AUXSTRUC(R3),R4    ; Get PDT address
        01       12     09A6  2847          BNEQ    5$                       ; Branch if there is a PDT
                 05     09A8  2848          RSB                              ; Else port init aborted, can't
                        09A9  2849                                           ;  use port
                        09A9  2850
  55    00DC C4    D0   09A9  2851 5$:      MOVL    PDT$L_UCB0(R4),R5        ; Get UCB address
        04       E0     09AE  2852          BBS     #UCB$V_ONLINE,-          ; Branch if controller/unit is
     03 64 A5           09B0  2853                  UCB$W_STS(R5),CONT_POLL  ;  on line
        00A7     31     09B3  2854          BRW     CNF$CALCINTDUE           ; Else bypass poller and other activity
                        09B6  2855                                           ;  and compute next wakeup time
                        09B6  2856
                        09B6  2857 CONT_POLL:
                        09B6  2858
0104 D4    01    D0     09B6  2859          MOVL    #1,@PDT$L_MTC(R4)        ; Poke the maint timer in the
                        09BB  2860                                           ;  port to tell it we are alive
                        09BB  2861          SETIPL  #IPL$_SCS                ; Lower IPL for rest of polling, etc.
           53   DD      09BE  2862          PUSHL   R3                       ; Save CRB address
53   0174 C4    DE     09C0  2863          MOVAL   PDT$Q_FORMPB(R4),R3      ; Get formative PB listhead addr
```

```
                53   DD  09C5  2864          PUSHL   R3                      ;  and save a copy
           53   63   D0  09C7  2865          MOVL    (R3),R3                 ; Get addr of 1st entry in PB list
                         09CA  2866
                         09CA  2867 SCAN_FORMPB:
                         09CA  2868
           6E   53   D1  09CA  2869          CMPL    R3,(SP)                 ; Back at start of list?
                1F   13  09CD  2870          BEQL    FORM_PB_DONE            ; Branch if so
           55   63   D0  09CF  2871          MOVL    (R3),R5                 ; Save addr of next PB in
                         09D2  2872                                          ;  case this one gets deleted
                00   E1  09D2  2873          BBC     #PB$V_TIM,-             ; Branch if no timeout
              12 44 A3       09D4  2874                  PB$W_STS(R3),10$    ;  is in progress
                3C A3   D1  09D7  2875          CMPL    PB$L_DUETIME(R3),-      ; Passed this PB's duetime?
         00000000'GF         09DA  2876                  G^EXE$GL_ABSTIM         ;
                08   1A  09DF  2877          BGTRU   10$                     ; Branch if not
        51    8001 8F   3C  09E1  2878          MOVZWL  #EV$C_TIMEOUT,R1        ; Set event = timed out
                FADA   30  09E6  2879          BSBW    ACTION_DISP             ; Call action dispatcher for
                         09E9  2880                                          ;  this PB
                         09E9  2881
           53   55   D0  09E9  2882 10$:      MOVL    R5,R3                   ; Step to next formative  PB
                DC   11  09EC  2883          BRB     SCAN_FORMPB             ; Check next PB
                         09EE  2884
                         09EE  2885 FORM_PB_DONE:
                         09EE  2886
                8E   D5  09EE  2887          TSTL    (SP)+                   ; Clear PB listhd from stack
           0188 C4   D1  09F0  2888          CMPL    PDT$L_POOLDUE(R4),-     ; Passed pool chekcer's time?
         00000000'GF         09F4  2889                  G^EXE$GL_ABSTIM         ;
                3D   1A  09F9  2890          BGTRU   CHECK_POLLER            ; Branch if not
        55   00B0 C4   DE  09FB  2891          MOVAL   PDT$L_WAITQBL(R4),R5    ; Get pool waiter listhead addr
          FC A5   65   D1  0A00  2892          CMPL    (R5),-4(R5)             ; List empty?
                21   13  0A04  2893          BEQL    POOL_DONE               ; Branch if so
           55   65   D0  0A06  2894          MOVL    (R5),R5                 ; Else get addr of last waiter (if any)
                         0A09  2895
        53   00AC C4   D0  0A09  2896 20$:      MOVL    PDT$L_WAITQFL(R4),R3    ; Get addr of next CDRP we are
                         0A0E  2897                                          ;  going to try to wake
                         0A0E  2898          $RESUME_FP      -               ; Resume next waiter
                         0A0E  2899                  @PDT$L_WAITQFL(R4),-    ;
                         0A0E  2900                  QEMPTY=POOL_DONE        ; If none, go to POOL_DONE
           55   53   D1  0A22  2901          CMPL    R3,R5                   ; Was this waiter the last one when
                         0A25  2902                                          ;  we started scanning the list?
                         0A25  2903                                          ;  (More on the list now are
                         0A25  2904                                          ;  repeat failures )
                E2   12  0A25  2905          BNEQ    20$                     ; Branch if not
                         0A27  2906
                         0A27  2907 POOL_DONE:
                         0A27  2908
50    00000000'GF   3C  0A27  2909          MOVZWL  G^SCS$GW_PAPOOLIN,R0    ; Get pool check interval
00000000'GF   50   C1  0A2E  2910          ADDL3   R0,G^EXE$GL_ABSTIM,-    ; Add pool interval to current
           0188 C4         0A35  2911                  PDT$L_POOLDUE(R4)       ;  time and store as due time
                         0A38  2912
                         0A38  2913 CHECK_POLLER:
                         0A38  2914
                   53 8ED0  0A38  2915          POPL    R3                      ; Retreive CRB addr
              018C C4   D1  0A3B  2916          CMPL    PDT$L_POLLERDUE(R4),-   ; Passed poller's duetime?
         00000000'GF         0A3F  2917                  G^EXE$GL_ABSTIM         ;
                17   1A  0A44  2918          BGTRU   CNF$CALCINTDUE          ; Branch if not
                   F5B7   30  0A46  2919          BSBW    CNF$POLL                ; Call poller
                         0A49  2920
```

```
                50   00000000'GF   3C   0A49  2921          MOVZWL  G^SCS$GW_PAPOLINT,R0     ; Get poller interval
00000000'GF    50   C1   0A50  2922          ADDL3   R0,G^EXE$GL_ABSTIM,-    ; Add poll interval to current time and
                018C C4         0A57  2923                  PDT$L_POLLERDUE(R4)     ;  store as poller duetime
                   0011   30    0A5A  2924          BSBW    CNF$CALC_POLLSW         ; Compute current time it takes
                                0A5D  2925                                          ;  to do a complete poll sweep
                                0A5D  2926                                          ;  over both paths -- this has
                                0A5D  2927                                          ;  to be recomputed periodically because
                                0A5D  2928                                          ;  the parameters are dynamic
                                0A5D  2929
                                0A5D  2930 CNF$CALCINTDUE::
                                0A5D  2931
                50   00000000'GF   3C   0A5D  2932          MOVZWL  G^SCS$GW_PASTMOUT,R0    ; Get basic timer interval
00000000'GF    50   C1   0A64  2933          ADDL3   R0,G^EXE$GL_ABSTIM,-    ; Add it to current time and
                     18 A3      0A6B  2934                  CRB$L_DUETIME(R3)       ;  and save in CRB
                                0A6D  2935
                         05     0A6D  2936 30$:    RSB                             ; Return
                                0A6E  2937
                                0A6E  2938          .DSABL  LSB
```

```
                        0A6E  2940          .SBTTL  CNF$CALC_POLLSW, CALCULATE TIME TO POLL
                        0A6E  2941          .SBTTL  -                PORT AT LEAST ONCE
                        0A6E  2942
                        0A6E  2943 ;+
                        0A6E  2944 ; This routine computes the number of seconds it takes to poll
                        0A6E  2945 ; every possible port at least once, even if only one path is
                        0A6E  2946 ; working.  This value is used by the VAXcluster sysap.
                        0A6E  2947 ;
                        0A6E  2948 ; The formula is as follows:
                        0A6E  2949 ;
                        0A6E  2950 ;  ((maximum port # +1)/(# ports polled per interval)) * 2 paths * poll interval
                        0A6E  2951 ;   +maximum time to wake up poller
                        0A6E  2952 ;
                        0A6E  2953 ; If the number of ports polled per interval exceeds the number of free
                        0A6E  2954 ; datagrams available to conduct simultaneous start handshakes, then use
                        0A6E  2955 ; the number of free datagrams instead of the number of ports per interval
                        0A6E  2956 ; in the above formula.  The number of free datagrams available is not known
                        0A6E  2957 ; exactly since there is no accounting on the datagrams that can be tied
                        0A6E  2958 ; up doing start handshakes.  The number available is estimated as
                        0A6E  2959 ; PDT$W_STDGDYN(R4).
                        0A6E  2960 ;
                        0A6E  2961 ; Inputs:
                        0A6E  2962 ;
                        0A6E  2963 ;       R4                      -PDT address
                        0A6E  2964 ;
                        0A6E  2965 ;       SCS$GB_PAMXPORT         -SYSGEN'ed maximum port #
                        0A6E  2966 ;       SCS$GB_PANPOLL          -# ports to poll per interval
                        0A6E  2967 ;       SCS$GW_PAPOLINT         -# seconds between polls, poll interval
                        0A6E  2968 ;       SCS$GW_PASTIMOUT        -# seconds it might take to wake up poller
                        0A6E  2969 ;       PDT$B_MAX_PORT(R4)      -maximum port # supported by this CI
                        0A6E  2970 ;
                        0A6E  2971 ; Outputs:
                        0A6E  2972 ;
                        0A6E  2973 ;       R0,R1,R2                -Destroyed
                        0A6E  2974 ;       Other registers         -Preserved
                        0A6E  2975 ;
                        0A6E  2976 ;       PDT$L_POLLSWEEP(R4)     -# seconds to poll each port at least once
                        0A6E  2977 ;-
                        0A6E  2978
                        0A6E  2979          .ENABL  LSB
                        0A6E  2980
                        0A6E  2981 CNF$CALC_POLLSW::
                        0A6E  2982
51   00000000'GF    9A  0A6E  2983          MOVZBL  G^SCS$GB_PAMXPORT,R1    ; Get SYSGENed max port #
     50    017C C4   9A  0A75  2984          MOVZBL  PDT$B_MAX_PORT(R4),R0   ; Get hardware supported max port
           50   51   D1  0A7A  2985          CMPL    R1,R0                   ; SYSGENed .GT. hardware max?
                03   15  0A7D  2986          BLEQ    10$                     ; Branch if not
           51   50   D0  0A7F  2987          MOVL    R0,R1                   ; Else hardware value prevails
                     0A82  2988
           51   D6   0A82  2989 10$:       INCL    R1                      ; Convert port # to number of ports
50   00000000'GF    9A  0A84  2990          MOVZBL  G^SCS$GB_PANPOLL,R0     ; Get # ports polled per interval
     52    0198 C4   3C  0A8B  2991          MOVZWL  PDT$W_STDGDYN(R4),R2    ; Get # dgs available for start
                     0A90  2992                                          ;  start handshakes, max.
           52   50   D1  0A90  2993          CMPL    R0,R2                   ; # ports per interval .leq. free dg
                     0A93  2994                                          ;  limit?
                03   1B  0A93  2995          BLEQU   15$                     ; Branch if so
           50   52   D0  0A95  2996          MOVL    R2,R0                   ; Else use free dg limit instead
```

```
                          0A98  2997
                    52    0A98  2998 15$:    CLRL    R2                              ; Clear h.o. longwd of dividend
          50  51   51  50  0A9A  2999         EDIV    R0,R1,R1,R0                     ; Compute # ports/ # per interval polled
                    50  0A9F  3000         TSTL    R0                              ; If there was a remainder,
                    02  0AA1  3001         BEQL    20$                             ;
                    51  0AA3  3002         INCL    R1                              ;  then round quotient up
                          0AA5  3003
                 51 51  0AA5  3004 20$:    ADDL    R1,R1                           ; Multipy by 2 paths *
           50 00000000'GF  0AA8  3005         MOVZWL  G^SCS$GW_PAPOLINT,R0            ;  the number of seconds between
                 51 50  0AAF  3006         MULL    R0,R1                           ;  polls
           50 00000000'GF  0AB2  3007         MOVZWL  G^SCS$GW_PASTMOUT,R0            ; Get the timer before poller even
                          0AB9  3008                                                 ;  awakened,
      00D8 C4   51 50  0AB9  3009         ADDL3   R0,R1,PDT$L_POLLSWEEP(R4)       ;  add in and save total in PDT
                    05  0ABF  3010         RSB                                     ; Return
                          0AC0  3011
                          0AC0  3012         .DSABL  LSB
```

```
                     0AC0  3014          .SBTTL  START_TIMER, START A PATH BLOCK TIMER
                     0AC0  3015 ;+
                     0AC0  3016 ; START_TIMER computes the due time for PB timeout and sets the
                     0AC0  3017 ; timeout in porgress bit (PB$V_TIM in PB$W_STS) for the specified
                     0AC0  3018 ; pathblock.
                     0AC0  3019 ;
                     0AC0  3020 ; Inputs:
                     0AC0  3021 ;
                     0AC0  3022 ;       R3                          -Addr of PB
                     0AC0  3023 ;
                     0AC0  3024 ; Outputs:
                     0AC0  3025 ;
                     0AC0  3026 ;       R0                          -Destroyed
                     0AC0  3027 ;       Other registers             -Preserved
                     0AC0  3028 ;-
                     0AC0  3029
                     0AC0  3030          .ENABL  LSB
                     0AC0  3031
                     0AC0  3032 START_TIMER:
                     0AC0  3033
50   00000000'GF  3C 0AC0  3034          MOVZWL  G^SCS$GW_PASTMOUT,R0    ; Get basic timer interval
00000000'GF   50  C1 0AC7  3035          ADDL3   R0,G^EXE$GL_ABSTIM,-    ; Add it to the current time
            3C A3    0ACE  3036                  PB$L_DUETIME(R3)        ;  and save in PB due time
               00 E2 0AD0  3037          BBSS    #PB$V_TIM,-             ; Set timeout in progress
         00 44 A3    0AD2  3038                  PB$W_STS(R3),10$        ;  in pathblock
                  05 0AD5  3039 10$:     RSB                             ; Return
                     0AD6  3040
                     0AD6  3041          .DSABL  LSB
```

```
                     0AD6  3043          .SBTTL  STOP_TIMER, STOP PATH BLOCK TIMER
                     0AD6  3044
                     0AD6  3045 ;+
                     0AD6  3046 ; STOP_TIMER disables path block timeout by clearing the timeout
                     0AD6  3047 ; in progress bit in the pathblock.
                     0AD6  3048 ;
                     0AD6  3049 ; Inputs:
                     0AD6  3050 ;
                     0AD6  3051 ;       R3                      -Addr of PB
                     0AD6  3052 ;
                     0AD6  3053 ; Outputs:
                     0AD6  3054 ;
                     0AD6  3055 ;       All registers           -Preserved
                     0AD6  3056 ;-
                     0AD6  3057
                     0AD6  3058 STOP_TIMER:
                     0AD6  3059
          00    E5   0AD6  3060          BBCC    #PB$V_TIM,-             ; Clear the timeout in progress bit
00 44 A3               0AD8  3061                  PB$W_STS(R3),10$        ;  in specified pathblock
                 05    0ADB  3062 10$:     RSB                             ; Return
```

```
                        0ADC  3064          .SBTTL  SET_CIRCUIT, PORT OPENS A PORT-PORT VIRTUAL CIRCUIT
                        0ADC  3065
                        0ADC  3066 ;+
                        0ADC  3067 ; SET_CIRCUIT allocates a datagram buffer.  If none are available,
                        0ADC  3068 ; return with error status.  Otherwise, send the SETCKT datagram
                        0ADC  3069 ; to the port.
                        0ADC  3070 ;
                        0ADC  3071 ; Inputs:
                        0ADC  3072 ;
                        0ADC  3073 ;       R2                      -Addr of START/STACK dg
                        0ADC  3074 ;       R3                      -Addr of formative PB
                        0ADC  3075 ;       R4                      -Addr of PDT
                        0ADC  3076 ;
                        0ADC  3077 ; Outputs:
                        0ADC  3078 ;
                        0ADC  3079 ;       R0                      -0/1 for fail/success
                        0ADC  3080 ;       Other registers         -Preserved
                        0ADC  3081 ;-
                        0ADC  3082
                        0ADC  3083          .ENABL  LSB
                        0ADC  3084
                        0ADC  3085 SET_CIRCUIT:
                        0ADC  3086
              52   DD   0ADC  3087          PUSHL   R2                      ; Save dg addr
           F51F'   30   0ADE  3088          BSBW    INT$ALLOC_PPDDG         ; Allocate a dg buffer
         31 50     E9   0AE1  3089          BLBC    R0,SET_ERR              ; Branch if none
                   D0   0AE4  3090          MOVL    #<PPD$M_RSP@24>!-       ;
                        0AE5  3091                   <PPD$C_INVTC@16>,-      ;
 0C A2  01180000 8F      0AE5  3092                   PPD$B_PORT(R2)          ; Set opcode and ask for response
           F511'   30   0AEC  3093          BSBW    INT$INS_COMQH           ; Issue the invalidate command
           F50E'   30   0AEF  3094          BSBW    INT$ALLOC_DG1           ; Allocate a datagram buffer
                        0AF2  3095                                           ;  for the open circuit command
         20 50     E9   0AF2  3096          BLBC    R0,SET_ERR              ; Branch if insufficient pool
                   C9   0AF5  3097          BISL3   #<PPD$M_RSP@24>!-       ; Open VC, reset sequence #'s
                        0AF6  3098                   <PPD$C_SETCKT@16>,-     ; Get SETCKT back for pool
 0C A3  01190000 8F      0AF6  3099                   PB$B_RSTATION(R3),-     ;
           0C A2         0AFD  3100                   PPD$B_PORT(R2)          ;
                   3C   0AFF  3101          MOVZWL  #<PPD$M_CST!-           ;
                        0B00  3102                   PPD$M_NR!PPD$M_NS>,-    ;
   10 A2    E000 8F      0B00  3103                   PPD$W_MASK(R2)          ; Set mask
   14 A2    8000 8F   3C 0B05  3104          MOVZWL  #PPD$M_CST,PPD$W_M_VAL(R2)
         F4F2'   30   0B0B  3105          BSBW    INT$INS_COMQH           ; Send it on its way
      50   01   9A   0B0E  3106          MOVZBL  #SS$_NORMAL,R0          ; Set status to success
                        0B11  3107
           52 8ED0      0B11  3108 10$:     POPL    R2                      ; Retreive dg addr
                   05   0B14  3109          RSB                             ; Return
                        0B15  3110
                        0B15  3111 SET_ERR:
                        0B15  3112
                   50   D4   0B15  3113          CLRL    R0                      ; Set status to failure
                   F8   11   0B17  3114          BRB     10$                     ; Take common exit
                        0B19  3115
                        0B19  3116          .DSABL  LSB
                        0B19  3117
                        0B19  3118
                        0B19  3119
                        0B19  3120          .END
```

```
$$$                     = 000004BC R      01
$$$CURSIZ               = 000001C4
$$$LAST_EVENT           = 000004B8 R      01
$$$LAST_STATE           = 00000493 R      01
$$$NEWSIZ               = 000001D0
AC$B_ARG                = 00000001
AC$B_CODE               = 00000000
AC$C_CONTINUE           = 00000001
AC$C_END                = 00000000
AC$W_ACTION             = 00000002
AC$W_NEWST              = 00000001
ACTION_DISP               000004C3 R      01
ACTION_TABLE              00000380 RG     01
ALL_STOPPED               0000037F R      01
BREAK_HOST                0000076F R      01
BREAK_PATH                00000775 R      01
BUG$_CIPORT               ******** X      01
BUILD_SB                  00000706 R      01
CAUTION_REV               00000956 R      01
CHECK_CAUTION             00000993 R      01
CHECK_POLLER              00000A38 R      01
CHECK_PORT_REV            0000095E R      01
CHK_INCARN_ERR            0000065E R      01
CLEAN2                    000007F7 R      01
CLEANUP                   000007DE R      01
CMP_EXIST_SBS             000005F7 R      01
CNF$CALCINTDUE            00000A5D RG     01
CNF$CALC_POLLSW           00000A6E RG     01
CNF$DGREC                 0000029D RG     01
CNF$IDREC                 000000FB RG     01
CNF$LBREC                 0000026A RG     01
CNF$LKP_PB_MSG            00000822 RG     01
CNF$LKP_PB_MSG2           00000816 RG     01
CNF$LKP_PB_PDT            00000862 RG     01
CNF$POLL                  00000000 RG     01
CNF$REMOVE_PB             000008A5 RG     01
CNF$SCSMSG_REC            00000221 RG     01
CNF$STOP_VCS              000002CF RG     01
CNF$TIMER                 000009A2 RG     01
COM$DRVDEALMEM            ******** X      01
COM_SEND_1                00000563 R      01
CONFIG_ERR                000002C5 R      01
CONFIG_EXIT               000000F6 R      01
CONFIG_LIST               000002B5 R      01
CONT_POLL                 000009B6 R      01
CRB$C_AUXSTRUC          = 00000010
CRB$L_DUETIME           = 00000018
DATA_LEN                = 0000002C
DDB$T_NAME              = 00000014
DELETE_SB                 0000066C R      01
DO_REFRESH                0000064E R      01
DYN$C_CIDG              = 0000003B
DYN$C_SCS               = 00000060
DYN$C_SCS_PB            = 00000004
DYN$C_SCS_SB            = 00000007
ELOG$CABLES               ******** X    01
ELOG$CBL_X_CHG            ******** X    01
ELOG$ERROR_DG             ******** X    01
ELOG$PACKET               ******** X    01
ELOG$PTH_ST_CHG           ******** X    01
ELOG$UCODE_ERR            ******** X    01
ELOG$UCODE_WARN           ******** X    01
END_ACTION                0000051B R      01
ENTER_DONE                000006AD R      01
ENTER_ERR                 000006E0 R      01
ENTER_ERR1                000006B1 R      01
ENTER_ERR2                000006B1 R      01
ENTER_ERR3                000006BA R      01
ENTER_ERR4                000006DC R      01
ENTER_PB                  000005C1 R      01
ERR$BUGCHECKNF            ******** X      01
ERR$CRASHVC               ******** X      01
ERR$CRASH_PORT            ******** X      01
EV$C_ACK                = 00000002
EV$C_ELOG               = 00000005
EV$C_HOSTSHUT           = 00000006
EV$C_SCSMSG             = 00008000
EV$C_SEND_START         = 00008002
EV$C_STACK              = 00000001
EV$C_START              = 00000000
EV$C_TIMEOUT            = 00008001
EV$W_CODE               = 00000000
EV$W_NEXT               = 00000002
EXE$ALONONPAGED           ******** X      01
EXE$GB_CPUDATA            ******** X      01
EXE$GL_ABSTIM             ******** X      01
EXE$GL_LOCKRTRY           ******** X      01
EXE$GL_TENUSEC            ******** X      01
EXE$GL_UBDELAY            ******** X      01
EXE$GQ_SYSTIME            ******** X      01
FMT_START_DATA            0000078F R      01
FORM_PB_DONE              000009EE R      01
FOUND_PB                  000002BE R      01
FOUND_VC                  000002FF R      01
GOT_PATH                  000001C8 R      01
IGNORE_DG                 0000078C R      01
INI$PORT_REV              ******** X      01
INIST_HWTYPE              ******** X      01
INT$ALLOC_DG1             ******** X      01
INT$ALLOC_MSG             ******** X      01
INT$ALLOC_PPDDG           ******** X      01
INT$DEAL_DG1              ******** X      01
INT$DEAL_MSG              ******** X      01
INT$INS_COMQH             ******** X      01
INT$INS_COMQL             ******** X      01
INT$INS_DFREEQ1           ******** X      01
INT$INS_MFREEQ            ******** X      01
INT$MFQ2POOL              ******** X      01
INT$SNDDG1                ******** X      01
IPL$_SCS                = 00000008
LB_CHECK                  0000002D R      01
LB_ENABLE                 00000911 R      01
LEGAL_REV_TABLE           00000946 R      01
LOCK_ONAVAIL              0000037D R      01
```

```
LOOKUP_EVENT                    000004E0 R      01
MAX_RAM_REV                     0000095A R      01
MAX_ROM_REV                     0000095C R      01
MOVE_PB                         0000067F R      01
MOVE_SB                         00000674 R      01
NEW_PATH                        00000108 R      01
NEW_PATH_ERR                    000001C8 R      01
NEXT_ACTION                     000004F2 R      01
NEXT_EVENT                      000004E2 R      01
NEXT_REQID                      00000089 R      01
NEXT_SB                         0000089D R      01
NEXT_STATE                      000004CC R      01
NOT_FOUND                       000008A2 R      01
PAER$K_ES_LOBG                  =  00000008
PAER$K_ES_LOGB                  =  00000006
PAER$K_ES_L1BG                  =  00000009
PAER$K_ES_L1GB                  =  00000007
PAER$K_ES_LST0                  =  00000003
PAER$K_ES_LST1                  =  00000009
PAER$K_ES_LST2                  =  00000007
PAER$K_ES_LST3                  =  00000009
PAER$K_ES_LST4                  =  0000000C
PAER$K_ES_RSCKS                 =  00000008
PAER$K_ET_DALT                  =  00000003
PAER$K_ET_LMLT                  =  00000042
PB$B_CBL_STS                    =  00000028
PB$B_PO_STS                     =  00000029
PB$B_P1_STS                     =  0000002A
PB$B_PROTOCOL                   =  00000048
PB$B_RSTATE                     =  00000021
PB$B_RSTATION                   =  0000000C
PB$B_RST_PORT                   =  00000020
PB$B_SUBTYP                     =  0000000B
PB$B_TYPE                       =  0000000A
PB$C_CLOSED                     =  00000000
PB$C_LENGTH                     =  00000054
PB$C_OPEN                       =  00000003
PB$C_PALENGTH                      00000060
PB$C_PWR_FAIL                   =  00004000
PB$C_ST_REC                     =  00000002
PB$C_ST_SENT                    =  00000001
PB$C_VC_FAIL                    =  00008000
PB$L_CDTLST                     =  00000034
PB$L_CLSCKT_DG                     00000054
PB$L_DUETIME                    =  0000003C
PB$L_FLINK                      =  00000000
PB$L_PDT                        =  0000002C
PB$L_RPORT_FCN                  =  0000001C
PB$L_RPORT_REV                  =  00000018
PB$L_RPORT_TYP                  =  00000014
PB$L_SBLINK                     =  00000030
PB$L_SCSMSG                     =  00000040
PB$L_WAITQBL                    =  0000003C
PB$L_WAITQFL                    =  00000038
PB$M_CUR_CBL                    =  00000001
PB$M_CUR_PS                     =  00000001
PB$T_LPORT_NAME                 =  00000024

PB$V_CUR_CBL                    =  00000000
PB$V_TIM                        =  00000000
PB$W_RETRY                      =  00000022
PB$W_SIZE                       =  00000008
PB$W_STATE                      =  00000012
PB$W_STS                        =  00000044
PB$W_VCFAIL_RSN                 =  00000046
PB_EXISTS                          000002AA R      01
PB_FOUND                           00000854 R      01
PB_NOT_FOUND                       0000085E R      01
PB_STATE_ERR                       00000527 R      01
PDT$B_DQIMAP                       00000154
PDT$B_HSHUT_DG                     000001B0
PDT$B_MAX_PORT                     0000017C
PDT$B_NXT_PORT                     0000017E
PDT$B_PO_LBSTS                     00000180
PDT$B_P1_LBSTS                     00000181
PDT$B_PLOGMAP                      00000134
PDT$B_PORTMAP                      00000114
PDT$B_PORT_NUM                     0000017D
PDT$B_REQIDPS                      0000017F
PDT$C_HSHUT_SIZ                 =  00000014
PDT$C_LENGTH                    =  000000E4
PDT$C_PAREGBASE                    000000E4
PDT$C_PAREGEND                     00000110
PDT$C_PQB                       =  000001E0
PDT$L_CNF                          000000E4
PDT$L_CQ0                          000000F0
PDT$L_CQ1                          000000F4
PDT$L_DFQ                          000000FC
PDT$L_DFQHDR                       00000208
PDT$L_DGHDRSZ                      00000190
PDT$L_DGNETHD                      00000194
PDT$L_DQELOGOUT                    000002E0
PDT$L_GPTBASE                      0000022C
PDT$L_GPTLEN                       00000230
PDT$L_LBDG                         00000184
PDT$L_MFQ                          00000100
PDT$L_MFQHDR                       0000020C
PDT$L_MQELOGOUT                    00000320
PDT$L_MSGHDRSZ                  =  000000B4
PDT$L_MTC                          00000104
PDT$L_PFAR                         00000108
PDT$L_PMC                          000000E8
PDT$L_POLLERDUE                    0000018C
PDT$L_POLLSWEEP                 =  000000D8
PDT$L_POOLDUE                      00000188
PDT$L_PPR                          0000010C
PDT$L_PS                           000000EC
PDT$L_PSR                          000000F8
PDT$L_SPTBASE                      00000224
PDT$L_SPTLEN                       00000228
PDT$L_UCB0                      =  000000DC
PDT$L_VBDT                         0000021C
PDT$L_VPQB                         00000218
PDT$L_WAITQBL                   =  000000B0
PDT$L_WAITQFL                   =  000000AC
```

```
PDT$M_CUR_LBS           =  00000001
PDT$M_LBDG              =  00000004
PDT$M_PRV_LBS           =  00000002
PDT$Q_COMQ2                00001F0
PDT$Q_COMQ3                00001F8
PDT$Q_COMQBASE             00001E0
PDT$Q_COMQH                00001E8
PDT$Q_COMQL                00001E0
PDT$Q_DFREEQ               00001D0
PDT$Q_FORMPB               00000174
PDT$Q_MFREEQ               00001D8
PDT$Q_RSPQ                 00000200
PDT$Q_TEMP_RSPQ            0000019C
PDT$V_CUR_LBS           =  00000000
PDT$V_LBDG              =  00000002
PDT$W_BDTLEN               00000220
PDT$W_DQELEN               00000210
PDT$W_LPORT_STS            00000110
PDT$W_MQELEN               00000214
PDT$W_PBCOUNT              00000112
PDT$W_STDGDYN              00000198
PDT$W_STDGUSED             0000019A
POOL_DONE                  00000A27 R      01
PPD$B_DEF_ST               0000001C
PPD$B_FLAGS                0000000F
PPD$B_HWVERS               00000034
PPD$B_LBDATA               00000012
PPD$B_LCB_0                00000012
PPD$B_LCB_LPORT            00000010
PPD$B_LCB_NPORT            0000000F
PPD$B_LCB_OPC              00000011
PPD$B_LCB_PORT             0000000E
PPD$B_OPC                  0000000E
PPD$B_PORT                 0000000C
PPD$B_PROTOCOL             0000001A
PPD$B_RSTATE               00000025
PPD$B_RST_PORT             00000024
PPD$B_STATUS               0000000D
PPD$B_SWFLAG               0000000B
PPD$B_SYSTEMID             00000014
PPD$B_TYPE                 0000000A
PPD$C_ACK               =  00000002
PPD$C_ACK_LEN           =  00000004
PPD$C_ENAB              =  00000002
PPD$C_HOSTSHUT          =  00000006
PPD$C_HSHUT_LEN         =  00000002
PPD$C_INVTC             =  00000018
PPD$C_LB_LENGTH            00000046
PPD$C_LCB_DATA             00000013
PPD$C_LENGTH               00000012
PPD$C_MIN_DGSIZ            00000050
PPD$C_PRT_ELOG          =  00000001
PPD$C_PSP0              =  00000001
PPD$C_PSP1              =  00000002
PPD$C_REQID             =  00000005
PPD$C_SETCKT            =  00000019
PPD$C_SNDDG             =  00000001
PPD$C_STACK             =  00000001
PPD$C_STACK_LEN         =  0000003E
PPD$C_START             =  00000000
PPD$C_START_LEN         =  0000003E
PPD$K_LB_LENGTH            00000046
PPD$K_LENGTH               00000012
PPD$L_BLINK                00000004
PPD$L_DG_DISC              00000028
PPD$L_FLINK                00000000
PPD$L_IN_VCD               00000018
PPD$L_LBCRC                00000042
PPD$L_PO_ACK               00000010
PPD$L_PO_NAK               00000014
PPD$L_PO_NRSP              00000018
PPD$L_P1_ACK               0000001C
PPD$L_P1_NAK               00000020
PPD$L_P1_NRSP              00000024
PPD$L_REC_BOFF             00000028
PPD$L_REC_NAME             00000024
PPD$L_RPORT_FCN            00000020
PPD$L_RPORT_REV            0000001C
PPD$L_RPORT_TYP            00000018
PPD$L_SND_BOFF             00000020
PPD$L_SND_NAME             0000001C
PPD$L_ST_ADDR              00000018
PPD$L_XCT_LEN              00000018
PPD$M_CST               =  00008000
PPD$M_DQI               =  00001000
PPD$M_NR                =  00004000
PPD$M_NS                =  00002000
PPD$M_RSP               =  00000001
PPD$Q_CURTIME              00000048
PPD$Q_NODENAME             00000040
PPD$Q_SWINCARN             00000028
PPD$Q_XCT_ID               00000010
PPD$S_PS                =  00000002
PPD$S_RP                =  00000002
PPD$S_SP                =  00000002
PPD$S_STATE             =  00000002
PPD$T_HWTYPE               00000030
PPD$T_SWTYPE               00000020
PPD$T_SWVERS               00000024
PPD$V_PS                =  00000001
PPD$V_RP                =  00000001
PPD$V_SP                =  00000004
PPD$V_STATE             =  00000001
PPD$W_LCB_LEN7             0000000C
PPD$W_LENGTH               00000010
PPD$W_MASK                 00000010
PPD$W_MAXDG                0000001C
PPD$W_MAXMSG               0000001E
PPD$W_MTYPE                00000012
PPD$W_M_VAL                00000014
PPD$W_SIZE                 00000008
PR$_IPL                 =  00000012
REC_ERROR_DG               0000077E R      01
REFRESH_SB                 00000637 R      01
```

```
REV_OK                          0000099F R      01
REV_TABLE_SIZ                 = 00000004
SB$B_HWVERS                   = 00000038
SB$B_SYSTEMID                 = 00000018
SB$B_TYPE                     = 0000000A
SB$K_LENGTH                   = 00000060
SB$L_CSB                      = 0000005C
SB$L_DDB                      = 00000054
SB$L_FLINK                    = 00000000
SB$L_PBBL                     = 00000010
SB$L_PBLONNX                  = 00000014
SB$L_PBFL                     = 0000000C
SB$Q_SWINCARN                 = 0000002C
SB$T_HWTYPE                   = 00000034
SB$T_NODENAME                 = 00000044
SB$T_SWTYPE                   = 00000024
SB$T_SWVERS                   = 00000028
SB$W_MAXDG                    = 00000020
SB$W_MAXMSG                   = 00000022
SB$W_SIZE                     = 00000008
SB_DONE                         0000076C R      01
SCAN_FORMPB                     000009CA R      01
SCS$ALL_FRDGS                   ********   X    01
SCS$GA_LOCALSB                  ********   X    01
SCS$GB_NODENAME                 ********   X    01
SCS$GB_PAMXPORT                 ********   X    01
SCS$GB_PANOPOLL                 ********   X    01
SCS$GB_PANPOLL                  ********   X    01
SCS$GB_SYSTEMID                 ********   X    01
SCS$GQ_CONFIG                   ********   X    01
SCS$GW_MAXDG                    ********   X    01
SCS$GW_PAPOLINT                 ********   X    01
SCS$GW_PAPOOLIN                 ********   X    01
SCS$GW_PASTMOUT                 ********   X    01
SCS$NEQ_SB                      ********   X    01
SCS$RESUMEWAITR                 ********   X    01
SEARCH_CONT                     00000801 R      01
SEARCH_PATHS                    000007FE R      01
SEARCH_RSPQ                     00000353 R      01
SEND_1ST_STACK                  0000055B R      01
SEND_1ST_START                  00000539 R      01
SEND_ACK                        000005AC R      01
SEND_ERR                        00000558 R      01
SEND_LB                         0000005E R      01
SEND_STACK                      00000590 R      01
SEND_START                      0000053F R      01
SEND_SUCCESS                    00000554 R      01
SET_CIRCUIT                     00000ADC R      01
SET_ERR                         00000B15 R      01
SIZ...                        = 00000001
SNDDG_NORET                     00000908 R      01
SNDDG_RET                       000008FF R      01
SS$_NORMAL                    = 00000001
SS$_NOSUCHNODE                = 0000028C
ST$Q_CODE                     = 00000000
ST$W_NEXT                     = 00000002
START_REQID                     00000082 R      01
START_TIMER                     00000AC0 R      01
STATUS                        = 00000080
STOP_NEXT                       00000305 R      01
STOP_TIMER                      00000AD6 R      01
SYS$GQ_VERSION                  ********   X    01
SYSAP$C_DISPPO                = 00000002
SYSAP$C_DISPQ                 = 00000000
TRY_TRANSIT                     0000025E R      01
UCB$B_ERTCNT                  = 00000080
UCB$B_LMERTCNT                  000000D2
UCB$B_LMERTMAX                  000000D3
UCB$B_LMEST                     000000D0
UCB$B_LMET                      000000D1
UCB$K_ERRDGBYTS               = 000000B4
UCB$K_LMPKTBYTS               = 00000040
UCB$L_CICMD                     000000F0
UCB$L_DDB                     = 00000028
UCB$L_DPC                     = 0000009C
UCB$L_MSGFKBLK                  000000A0
UCB$N_LSADDR                    000000D8
UCB$N_LSID                      000000DE
UCB$N_RSADDR                    000000E4
UCB$N_RSID                      000000EA
UCB$T_MSGDATA                   000000F8
UCB$T_OPA0_TEMP                 000000B8
UCB$V_ONLINE                  = 00000004
UCB$W_ERRCNT                  = 00000082
UCB$W_LMERRCNT                  000000D4
UCB$W_MSGBYTCNT                 000000F4
UCB$W_MSGPPDTYP                 000000F6
UCB$W_STS                     = 00000064
UPDATE_CBL_STS                  000001CB R      01
UPDATE_LEN                    = 0000003C
UPDATE_SWINCARN                 000005B7 R      01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
$$$115_DRIVER	00000B19 (2841.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
ABS	00000360 (864.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.02	00:00:03.76
Command processing	135	00:00:00.46	00:00:04.40
Pass 1	547	00:00:16.16	00:00:58.07
Symbol table sort	0	00:00:01.82	00:00:06.89
Pass 2	501	00:00:05.25	00:00:18.02
Symbol table output	4	00:00:00.26	00:00:00.49
Psect synopsis output	2	00:00:00.01	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1226	00:00:23.98	00:01:31.66

The working set limit was 1950 pages.
140447 bytes (275 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1711 non-local and 80 local symbols.
3120 source lines were read in Pass 1, producing 25 object records in Pass 2.
40 pages of virtual memory were used to define 37 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[DRIVER.OBJ]PALIB.MLB;1	8
_255DUA28:[SYS.OBJ]LIB.MLB;1	12
_255DUA28:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	29

1956 GETS were required to define 29 macros.

There were no errors, warnings or information messages.

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

PAEND
LIS

PACONFIG
LIS

PAERROR
LIS