C 8

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

FILEID**PACONFIG

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

P
V

```
0000          .TITLE  PACONFIG
0000          .IDENT  'V04-001'
0000
0000  ;*************************************************************************
0000  ;*                                                                       *
0000  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
0000  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.               *
0000  ;*  ALL RIGHTS RESERVED.                                                 *
0000  ;*                                                                       *
0000  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
0000  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE *
0000  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER *
0000  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
0000  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY *
0000  ;*  TRANSFERRED.                                                         *
0000  ;*                                                                       *
0000  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE *
0000  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT *
0000  ;*  CORPORATION.                                                         *
0000  ;*                                                                       *
0000  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS *
0000  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.              *
0000  ;*                                                                       *
0000  ;*                                                                       *
0000  ;*************************************************************************
0000
0000  ;++
0000  ;
0000  ; FACILITY:
0000  ;
0000  ;       VAX/VMS EXECUTIVE, I/O DRIVERS
0000  ;
0000  ; ABSTRACT:  CI CLUSTER CONFIGURATION DATABASE MAINTENANCE
0000  ;
0000  ; AUTHOR:  N. KRONENBERG,  MAY 1981
0000  ;
0000  ;MODIFIED BY:
0000  ;
0000  ;       V04-001 NPK3066         N. Kronenberg           7-Sep-1984
0000  ;               If the port microcode rev check fails, clear the
0000  ;               flag, INI$PORT_REV to indicate that, if a bugcheck
0000  ;               is taken as a result of crashing this port, it should
0000  ;               be the UCODEREV bugcheck, rather than the usual CIPORT
0000  ;               bugcheck.
0000  ;
0000  ;       V03-39  NPK3063         N. Kronenberg           20-Aug-1984
0000  ;               Fix SET_CIRCUIT to operate at high priority.  Fixes
0000  ;               the lost connect request message problem.
0000  ;               Add check to REFRESH_SB to return conflicting SCS
0000  ;               node name/ID if the SB being refreshed is the local
0000  ;               SB and the incarnation number being refreshed is
0000  ;               different from the incarnation currently there.
0000  ;
0000  ;       V03-38  NPK3060         N. Kronenberg           1-Aug-1984
0000  ;               Fix CNF$LBREC to attribute the loopback dg to the
0000  ;               correct path in the case where PANUMPORT .LE.
0000  ;               PAMAXPORT.
```

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

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

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

```
0000  ;              a start handshake rather than the one received with the
0000  ;              1st START dg.
0000  ;              Clean up local symbols in ENTER_PB.
0000  ;              Drop PB$L_SB in favor of PB$L_SBLINK.
0000  ;              Change CNF$IDREC to reflect slightly reordered PB.
0000  ;              Prevent systems from being configured that have the
0000  ;              same system id and different node names or the same
0000  ;              node name and different id's.
0000  ;
0000  ;      V03-020 KTA3046         Kerbey T. Altmann       30-Mar-1983
0000  ;              Redo for SCS/PPD split.
0000  ;
0000  ;      V03-019 NPK3022         N. Kronenberg           28-Feb-1983
0000  ;              Get system software version from SYS$GQ_VERSION instead
0000  ;              of SYS$K_VERSION for the start handshake.
0000  ;
0000  ;      V03-018 NPK3020         N. Kronenberg           28-Feb-1983
0000  ;              Fix word arithmetic in action dispatcher that computes
0000  ;              next state/action to be longwd arithmetic.
0000  ;
0000  ;      V03-017 DWT0068         David W. Thiel          20-Jan-1983
0000  ;              Add call to SCS$NEW_SB when a system block is created
0000  ;              or reused.
0000  ;
0000  ;      V03-016 NPK3015         N. Kronenberg           28-Dec-1982
0000  ;              Fix bugs in LB_ENABLE which turns loopback dgs back on
0000  ;              when all remote vc's gone.
0000  ;              Fix disable of lb dg in CNF$IDREC to be BICW instead of
0000  ;              BISW.
0000  ;
0000  ;      V03-015 NPK3014         N. Kronenberg           16-Dec-1982
0000  ;              Fix to return IDREC dg to free queue in case virtual circuit
0000  ;              must be crashed due to remote being in neither the enabled
0000  ;              nor maint enabled states.
0000  ;              Get node name for start/stack from the sysgened node name.
0000  ;
0000  ;      V03-014 NPK3010         N. Kronenberg           11-Nov-1982
0000  ;              Implement probe of n ports per poll rather than 16
0000  ;              ports per poll.
0000  ;              Implement poll of sysgenable maximum number of ports
0000  ;              rather than all 16 (or 240).
0000  ;              Add loopback dg enabled flag which is updated when
0000  ;              VC's are broken or attempted rather than figuring out
0000  ;              if loopback dg's should be enabled each poller interval.
0000  ;              Allow SB's with no path blocks to stay in configuration
0000  ;              database and expand info held in SB.
0000  ;
0000  ;      V03-013 NPK3008         N. Kronenberg           6-Oct-1982
0000  ;              Change FMT_START_DATA to include new protocol, nodename,
0000  ;              current time, and shortened hardware version fields in
0000  ;              start/stack dgs.
0000  ;
0000  ;      V03-012 NPK3006         N. Kronenberg           9-Sep-1982
0000  ;              Fixed action table to show that SET_CIRCUIT can
0000  ;              return status.  Fixed action dispatcher to save event
0000  ;              code on stack and to discard received START/STACK dg
0000  ;              if any, in case of action routine error status.  Fixes
```

```
0000    ;                 free dg disappearance problem.  Also fixed action
0000    ;                 dispatcher to discard received dg on action table lookup
0000    ;                 failure only if there is a dg in hand.  Changed
0000    ;                 FMT_START_DATA to put correct CPU type in dg.
0000    ;
0000    ;       V03-011 NPK3005          N. Kronenberg              19-Aug-1982
0000    ;                 In CNF$DGREC fix search of configuration database
0000    ;                 to call CNF$LKP_PB_MSG instead of manually matching
0000    ;                 on remote station addr (which is an incomplete check)
0000    ;
0000    ;       V03-010 ROW0114          Ralph O. Weber             30-JUN-1982
0000    ;                 Add a check to CNF$LBREC which prevents it from logging a
0000    ;                 successful loopback datagram received when the previous
0000    ;                 loopback datagram for the path in question was also
0000    ;                 successfully received.
0000    ;                 This change will be in a new driver image shipped in V3.1.
0000    ;
0000    ;       V03-009 NPK3001          N. Kronenberg              28-Jun-1982
0000    ;                 Modify ENTER_PB to save SB link permanently in PB$L_SBLINK.
0000    ;                 Fix CNF$REMOVE_PB to patch the SB link to the next path to
0000    ;                 use for a connection.
0000    ;
0000    ;       V03-008 ROW0112          Ralph O. Weber             27-JUN-1982
0000    ;                 Change loopback datagram logging to use ELOG$CABLES instead of
0000    ;                 ELOG$PACKET so that the error log type field gets set
0000    ;                 correctly.  Remove crossed loopback path logic which isn't
0000    ;                 supported by the hardware anyway.  Fix loopback status to
0000    ;                 always be successful when no loopback datagram is sent because
0000    ;                 there is another known node.
0000    ;                 This change will be in a new driver image shipped in V3.1.
0000    ;
0000    ;       V03-007 ROW0109          Ralph O. Weber             24-JUN-1982
0000    ;                 Modify CNF$POLL to send loopback datagrams if and only if no
0000    ;                 bits are set in the PDT port bit map, or the only bit set in
0000    ;                 the map is the one for the port on which the loopback datagram
0000    ;                 would be sent.
0000    ;                 This change will be in a new driver image shipped in V3.1.
0000    ;
0000    ;       V03-006 ROW0106          Ralph O. Weber             23-JUN-1982
0000    ;                 Add error logging for loopback datagrams to CNF$POLL and
0000    ;                 CNF$LBREC.  Enhance this error logging to aid in the detection
0000    ;                 of a single pair of crossed wires between a port and the star
0000    ;                 coupler.  (N.B. the hardware currently does not support these
0000    ;                 crossed wires checks.)
0000    ;                 This change will be in a new driver image shipped in V3.1.
0000    ;
0000    ;       V03-005 ROW0097          Ralph O. Weber             7-JUN-1982
0000    ;                 Added calls to error logging routines in CNF$IDREC at
0000    ;                 UPDATE_CBL_STS and UPDATE_PTH_STS.  Modified comments in
0000    ;                 CNF$POLL is show that loop-back datagrams are not currently
0000    ;                 supported and thus their results need not be logged.  Also
0000    ;                 added necessary reference to the $PAERDEF macro.
0000    ;                 This change will be in a new driver image shipped in V3.1.
0000    ;
0000    ;       V03-004 NPK2020          N. Kronenberg              23-Apr-1982
0000    ;                 Modified ENTER_PB to discard formative PB for system
0000    ;                 that is already in the database but with a different
```

```
0000    ;               incarnation number.  Prevents configuration of two
0000    ;               different systems that have the same system ID.
0000    ;
0000    ;       V03-003 NPK2019         N. Kronenberg           9-Apr-1982
0000    ;               Fixed PB allocation failure bug.
0000    ;               Made PB lookup failure in CNF$DGREC recoverable.
0000    ;
0000    ;       V03-002 NPK2018         N. Kronenberg           25-Mar-1982
0000    ;               Fixed to use short datagrams instead of LRP's for
0000    ;               REQID and SETCKT's.
0000    ;               Fixed to not do start handshake with remote port
0000    ;               in other than an enabled state.  If IDREC arrives
0000    ;               from port to which VC is open and remote port is
0000    ;               in other than an enabled state, crash the VC.
0000    ;               Updated format of start/stack dg.
0000    ;               Modify to allocate and attach a dg pkt to each
0000    ;               PB for use during VC crash.
0000    ;
0000    ;       V03-001 NPK2016         N. Kronenberg           18-Mar-1982
0000    ;               Fixed .TITLE
0000    ;
0000    ;--
```

```
0000  ;++
0000  ; This module of the CI port driver is responsible for polling the
0000  ; nodes in the cluster for new arrivals and for conducting the
0000  ; START handshake protocol necessary to opening port-port virtual
0000  ; circuits to new nodes.
0000  ;
0000  ; The system wide configuration database consists of:
0000  ;
0000  ;
0000  ;       SCS$GQ_CONFIG
0000  ;             !
0000  ;             v
0000  ;       System Block ----> Path Block ----> Path Block ---->...
0000  ;             !
0000  ;             v
0000  ;       System Block ----> Path Block ---->...
0000  ;             !
0000  ;             v
0000  ;            ...
0000  ;
0000  ; Both systems and paths with open port-port VC's and systems
0000  ; with no open paths are kept on the above list.
0000  ;
0000  ; When an IDREC datagram is received for a node which is currently
0000  ; unknown, a PB is created for it and linked to the formative PB
0000  ; list for this port.  When a START/STACK datagram is received from
0000  ; that port as part of the START handshake, a formative SB is
0000  ; created and linked to the PB.  The formative datastructure looks
0000  ; like:
0000  ;
0000  ;       PDT
0000  ;         !
0000  ;         v
0000  ;       Path Block ----> (System Block)
0000  ;         !
0000  ;         v
0000  ;       Path Block ----> (System Block)
0000  ;         !
0000  ;         v
0000  ;        ...
0000  ;
0000  ; When the START handshake is complete, a matching SB is sought in
0000  ; the system configuration database.  If one is found, then the
0000  ; formative SB is discarded and the formative PB linked to the
0000  ; existing SB.  If no matching SB is found, then the formative SB
0000  ; is moved to the system configuration database and, with it, its
0000  ; formative PB.
0000  ;
0000  ; The configuration poller is awakened periodically for each local
0000  ; port by the timer scan module.  Each time it wakes up, it allocates
0000  ; n (SCS$GB_PANPOLL) datagrams from pool and uses these datagrams
0000  ; to send REQID's to the next n ports.
0000  ;
0000  ; Datagram management is as follows:  Upon port initialization
0000  ; SGN$GB_PPDDG datagrams are preallocated and linked to the
0000  ; datagram free queue for receipt of IDREC's.  When any start
0000  ; handshake datagram is received (including IDREC) which is turned
```

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

```
    0000          .SBTTL  DEFINITIONS
    0000
    0000   ;
    0000   ; Set PSECT to driver code:
    0000   ;
    0000
00000000          .PSECT  $$$115_DRIVER,LONG
    0000
    0000   ;
    0000   ; System definitions (LIB.MLB):
    0000   ;
    0000
    0000          .nocross
    0000          $CRBDEF                         ; Channel Request Block offsets
    0000          $DDBDEF                         ; Device Datablock offsets
    0000          $DYNDEF                         ; Structure type codes
    0000          $IPLDEF                         ; IPL definitions
    0000          $PBDEF                          ; Path Block offsets
    0000          $PDTDEF                         ; Port Descriptor Table offsets
    0000          $PRDEF                          ; Internal Processor Registers
    0000          $SBDEF                          ; System Block offsets
    0000          $SSDEF                          ; System service definitions
    0000          $SYSAPDEF                       ; DG disposal flags
    0000          $UCBDEF                         ; Unit Control Block offsets
    0000
    0000   ;
    0000   ; PADRIVER definitions (PALIB.MLB):
    0000   ;
    0000
    0000          $PAERDEF                        ; Port driver error code values
    0000          $PAPBDEF                        ; CI extension to PB
    0000          $PAPDTDEF                       ; CI extension to PDT
    0000          $PAUCBDEF                       ; CI extension to UCB
    0000          $PPDDEF                         ; PPD layer of message/dg header
    0000          .cross
```

```
                      0000        .SBTTL  CNF$POLL, PERIODICALLY SEND REQID TO PORTS
                      0000
                      0000  ;+
                      0000  ; CNF$POLL is awakened periodically by CNF$TIMER. If remote port polling is
                      0000  ; enabled (SCS$GB_PANOPOLL is set to 0), it allocates as many datagram buffers
                      0000  ; as there are ports to poll per interval (up to the maximum legal port #
                      0000  ; specified by SCS$GB_PAMXPORT or the maximum legal hardware port # specified by
                      0000  ; PDT$B_MAX_PORT - which is ever is the smallest), and sends a REQID to each
                      0000  ; port. The sent buffers are reclaimed on the response queue and returned to
                      0000  ; pool.
                      0000  ;
                      0000  ; If datagram receipt is currently inhibited from this remote port,
                      0000  ; then datagrams are first reenabled via a SETCKT command.
                      0000  ;
                      0000  ; If the sweep does not complete due to lack of pool, CNF$POLL returns
                      0000  ; without error.
                      0000  ;
                      0000  ; Later receipt of the IDREC's will cause the START
                      0000  ; handshake to begin for the remote systems not currently known.
                      0000  ;
                      0000  ; The poller also initiates various diagnostic activities to
                      0000  ; check for physical connection problems or other errors in the
                      0000  ; cluster:
                      0000  ;
                      0000  ;       -Before polling begins, a loopback datagram is sent out if
                      0000  ;        loopback dg's are enabled.  LB dg's are enabled when no
                      0000  ;        remote port is known; otherwise, they are disabled.
                      0000  ;        Later, successful receipt of the LB dg is recorded in routine
                      0000  ;        CNF$LBREC.  Successful receipt of the last LB dg sent on this
                      0000  ;        path is checked here in LB_CHECK, before sending a new LB dg.
                      0000  ;
                      0000  ;       -REQID's are sent to all ports even if we have already
                      0000  ;        succeeded in a START handshake.  REQID's are sent with
                      0000  ;        explicit path select thus forcing the port to try the path
                      0000  ;        even if it thinks it is bad.  Later receipt of an IDREC on this
                      0000  ;        path forces the port to bring it back if it was previously
                      0000  ;        marked bad.  It also lets us log the transition of a path
                      0000  ;        from bad to good.
                      0000  ;
                      0000  ; Inputs:
                      0000  ;
                      0000  ;       R4                              -Addr of PDT
                      0000  ;
                      0000  ; Outputs:
                      0000  ;
                      0000  ;       R0-R2                           -Destroyed
                      0000  ;       other registers                 -Preserved
                      0000  ;
                      0000  ;-
                      0000
                      0000        .ENABL  LSB
                      0000
                      0000  CNF$POLL::
                      0000
      00E8 8F   BB    0000        PUSHR   #^M<R3,R5,R6,R7>        ; Save some registers
  00000000'GF   95    0004        TSTB    G^SCS$GB_PANOPOLL       ; Is remote polling enabled?
           03   13    000A        BEQL    5$                      ; Continue if it is
```

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

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

```
00FB            .SBTTL  CNF$IDREC, PROCESS UNSOLICITED IDREC
00FB
00FB     ;+
00FB     ; CNF$IDREC is called from IDREC for IDREC's with transaction
00FB     ; ID = 0.  CNF$IDREC checks the port bitmap to see if the IDREC
00FB     ; is from a path already established or with START handshake in
00FB     ; progress.  If not, and if the remote port is enabled, then
00FB     ;  a formative path block is set up and a START handshake initiated.
00FB     ;
00FB     ; If the PB does exist, then go to UPDATE_CBL_STS.  UPDATE_CBL_STS
00FB     ; checks if the path is fully open.  If not, no cable or path status
00FB     ; information is maintained, and the IDREC is simply discarded.  If
00FB     ; the path is open, and the remote port is in a state other than enabled,
00FB     ; then the virtual circuit is crashed.  If the remote port is enabled,
00FB     ; then cabling status is recorded in the path block as follows:
00FB     ;
00FB     ;       current cable status = 1 (OK) if the send path =
00FB     ;                                    receive path in IDREC;
00FB     ;
00FB     ;                            = 0 (bad) otherwise.
00FB     ;
00FB     ; If the new current status differs from the previous, then a cable status
00FB     ; transition is logged.
00FB     ;
00FB     ; The arrival of the IDREC says that the receive path of the ID must
00FB     ; be good.  Therefore, the path status in the PB is also updated as follows:
00FB     ;
00FB     ;       current path status  = 1 (OK).
00FB     ;
00FB     ; If the current path status differs from the previous, then a path status
00FB     ; transition is logged.
00FB     ;
00FB     ; Inputs:
00FB     ;
00FB     ;       R2                          -Addr of IDREC datagram
00FB     ;       R4                          -Addr of PDT
00FB     ;
00FB     ; Outputs:
00FB     ;
00FB     ;       R0-R2                       -Destroyed
00FB     ;       other registers             -Preserved
00FB     ;-
00FB
00FB     ;
00FB     ; Assumptions about PB format:
00FB     ;
00FB
00FB     ASSUME  PB$W_SIZE+2       EQ  PB$B_TYPE
00FB     ASSUME  PB$B_TYPE+1       EQ  PB$B_SUBTYP
00FB     ASSUME  PB$B_SUBTYP+1     EQ  PB$B_RSTATION
00FB     ASSUME  PB$B_RSTATION+6 EQ  PB$W_STATE
00FB     ASSUME  PB$W_STATE+2      EQ  PB$L_RPORT_TYP
00FB     ASSUME  PB$L_RPORT_TYP+4    EQ  PB$L_RPORT_REV
00FB     ASSUME  PB$L_RPORT_REV+4    EQ  PB$L_RPORT_FCN
00FB     ASSUME  PB$L_RPORT_FCN+4    EQ  PB$B_RST_PORT
00FB     ASSUME  PB$B_RST_PORT+1 EQ  PB$B_RSTATE
00FB     ASSUME  PB$B_RSTATE+1     EQ  PB$W_RETRY
```

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

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

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

```
                                        0221            .SBTTL  CNF$SCSMSG_REC, SCS MSG REC'D
                                        0221
                                        0221    ;+
                                        0221    ; Since the final ACK or STACK may be lost in the start handshake,
                                        0221    ; the arrival of an SCS CONNECT request message from the remote
                                        0221    ; system should be treated as a satisfactory substitute for receiving
                                        0221    ; a final ACK  or STACK.  CNF$SCSMSG_REC is called by PASCSCTL upon
                                        0221    ; receipt of every connect request to handle the transition of a
                                        0221    ; formative path block, if necessary, to the fully open state.
                                        0221    ;
                                        0221    ; Inputs:
                                        0221    ;
                                        0221    ;       R2                              -Addr of SCS message (start of application
                                        0221    ;                                        data)
                                        0221    ;       R4                              -Addr of PDT
                                        0221    ;
                                        0221    ; Outputs:
                                        0221    ;
                                        0221    ;       R0,R1,R3                        -Destroyed
                                        0221    ;       Other registers                 -Preserved
                                        0221    ;-
                                        0221
                                        0221            .ENABL  LSB
                                        0221
                                        0221    CNF$SCSMSG_REC::
                                        0221
                           52     DD    0221            PUSHL   R2                      ; Save SCS msg addr
                 53   0174 C4     7E    0223            MOVAQ   PDT$Q_FORMPB(R4),R3     ; Get list of formative PB's
          51     52   00B4 C4     C3    0228            SUBL3   PDT$L_MSGHDRSZ(R4),R2,R1; Back up to start of pkt
                 51     0C A1     9A    022E            MOVZBL  PPD$B_PORT(R1),R1       ; Get # of port that sent SCS msg
                          05C9    30    0232            BSBW    SEARCH_PATHS            ; See if this path is formative
                        26 50     E8    0235            BLBS    R0,TRY_TRANSIT          ; Branch if got formative PB
              0114 C4     51      E0    0238            BBS     R1,PDT$B_PORTMAP(R4),-  ; Branch if not formative, but is
                          28            023D                    10$                     ;  known (must be open)
                                        023E                                            ;  Else path is closed.  Since
                                        023E                                            ;  we got a sequenced msg, the
                                        023E                                            ;  port thinks the vc is open
                     53   51      D0    023E            MOVL    R1,R3                   ; Save port number
                        FDBC'     30    0241            BSBW    INT$ALLOC_PPDDG         ; Allocate PPD dg
                        1F 50     E9    0244            BLBC    R0,10$                  ; Branch if no pool
                                  C9    0247            BISL3   #<PPD$M_RSP@24>!-       ; Format PPD dg into a SETCKT
                                        0248                    <PPD$C_SETCKT@16>,-     ;
 0C A2     53    01190000 8F            0248                    R3,-                    ;  to port specified in R3
                                        0250                    PPD$B_PORT(R2)          ;
                  8000 8F         3C    0250            MOVZWL  #<PPD$M_CST>,-          ;
                     10 A2              0254                    PPD$W_MASK(R2)          ;
                     14 A2        D4    0256            CLRL    PPD$W_M_VAL(R2)         ;  and ask for vc state to be closed
                     FDA4'        30    0259            BSBW    INT$INS_COMQH           ; Do SETCKT at high priority
                        08        11    025C            BRB     10$                     ; Go to finish up
                                        025E
                                        025E    TRY_TRANSIT:
                                        025E
                 51   8000'8F     3C    025E            MOVZWL  #EV$C_SCSMSG,R1         ; Else set event code
                         025D     30    0263            BSBW    ACTION_DISP             ; Take action to move PB from
                                        0266                                            ;  formative to fully open
                                        0266                                            ;  If PB not in right state to
                                        0266                                            ;  transition to open or if
                 0266   921                                      ;  there is insufficient pool,
                 0266   922                                      ;  or if the system has bad
                 0266   923                                      ;  system name or system ID,
                 0266   924                                      ;  then formative PB and formative
                 0266   925                                      ;  system block are cleaned up
                 0266   926                                      ;  by action routines.
       52 8ED0   0266   927  10$:     POPL    R2                 ; Retreive SCS msg addr
               05   0269   928           RSB                        ; Return to PASCSCTL
                 026A   929
                 026A   930           .DSABL  LSB
```

```
                         026A          .SBTTL  CNF$LBREC,       VERIFY REC'D LOOPBACK DG
                         026A
                         026A    ;+
                         026A    ; CNF$LBREC checks the data in the received loopback datagram with
                         026A    ; the data stored in the template lb dg linked to the PDT.  If the
                         026A    ; data agrees, then the loopback status for the path on which the LB
                         026A    ; dg was received is updated to good.  (Transitions in the status are
                         026A    ; checked and logged in CNF$POLL.)
                         026A    ;
                         026A    ; Inputs:
                         026A    ;
                         026A    ;       R2                          -Addr of loopback datagram
                         026A    ;       R4                          -Addr of PDT
                         026A    ;       PDT$L_LBDG(R4)              -Addr of template LB dg
                         026A    ;
                         026A    ; Outputs:
                         026A    ;
                         026A    ;       R0-R2                       -Destroyed
                         026A    ;       Other registers             -Preserved
                         026A    ;-
                         026A
                         026A          .ENABL  LSB
                         026A
                         026A    CNF$LBREC::
                         026A
      51   0184 C4   D0  026A          MOVL    PDT$L_LBDG(R4),R1          ; Get addr of template
           7E   52   7D  026F          MOVQ    R2,-(SP)                   ; Save registers
                32   29  0272          CMPC    #<PPD$L_LBCRC - PPD$W_LENGTH>,-
              10 A1      0274                  PPD$W_LENGTH(R1),-         ; Verify rec'd data against template
              10 A2      0276                  PPD$W_LENGTH(R2)           ;  including LB dg length
           52   8E   7D  0278          MOVQ    (SP)+,R2                   ; Restore registers
                50   D5  027B          TSTL    R0                         ; Check results of comparison
                1B   12  027D          BNEQ    10$                        ; Branch if don't match
           02   01   EF  027F          EXTZV   #PPD$V_PS,#PPD$S_PS,-      ; Get path select, 1/2 for A/B
        50   0F A2       0282                  PPD$B_FLAGS(R2),R0         ;  in R0
  0E 017F C440   00  E2  0285          BBSS    #PDT$V_CUR_LBS, -                ; Set loopback datagram received
                         028C                  PDT$B_PO_LBSTS-1(R4)[R0], 10$ ; & branch if already got one.
                02   93  028C          BITB    #PDT$M_PRV_LBS,-           ; Was the previous loopback datagram
        017F C440        028E                  PDT$B_PO_LBSTS-1(R4)[R0];   also successful?
                06   12  0292          BNEQ    10$                        ; Branch if last was successful too
                         0294          ASSUME  PAER$K_ES_L1BG EQ <PAER$K_ES_L0BG +1>
           50   07   C0  0294          ADDL    #<PAER$K_ES_L0BG-1>,R0     ; Form LB dg succesful sybtype code
             FD66'   30  0297          BSBW    ELOG$CABLES                ; Log cables state change
                         029A
             FD63'   31  029A    10$:  BRW     INT$DEAL_DG1               ; Deallocate LB dg and return to
                         029D                                             ;  interrupt service from there
                         029D
                         029D          .DSABL  LSB
```

```
                          029D          .SBTTL  CNF$DGREC, DISPATCH A START/STACK/ACK DATAGRAM
                          029D
                          029D  ;+
                          029D  ; CNF$DGREC first checks the port bit map to see if a path
                          029D  ; block exists for the incoming datagram.  If not, the datagram
                          029D  ; is deallocated.  Otherwise, the formative path block list and
                          029D  ; system configuration data base are searched for the path block
                          029D  ; with matching station address.  When the path block is found,
                          029D  ; the ACTION_DISP routine is called to handle the datagram.
                          029D  ;
                          029D  ; Inputs:
                          029D  ;
                          029D  ;       R2                              -Addr of datagram
                          029D  ;       R4                              -Addr of PDT
                          029D  ;
                          029D  ; Outputs:
                          029D  ;
                          029D  ;       R0-R3                           -Destroyed
                          029D  ;       other registers                 -Preserved
                          029D  ;-
                          029D
                          029D          .ENABL  LSB
                          029D
                          029D  CNF$DGREC::
                          029D
      51   0C A2     9A   029D          MOVZBL  PPD$B_PORT(R2),R1       ; Get remote port #
 0114 C4      51     E0   02A1          BBS     R1,PDT$B_PORTMAP(R4),-  ; Look PB existence up in
              03          02A6                  PB_EXISTS               ;  path map; branch if exists
           FD56'     31   02A7          BRW     INT$INS_DFREEQ1         ; Discard datagram and return
                          02AA                                          ;  from there to interrupt service
                          02AA
                          02AA  PB_EXISTS:
                          02AA
 53    0174 C4       DE   02AA          MOVAL   PDT$Q_FORMPB(R4),R3     ; Get formative PB listhead
            054C     30   02AF          BSBW    SEARCH_PATHS            ; Search path list for PB
           09 50     E8   02B2          BLBS    R0,FOUND_PB             ; Branch if success
                          02B5
                          02B5  CONFIG_LIST:
                          02B5
            056A     30   02B5          BSBW    CNF$LKP_PB_MSG          ; Locate PB in open config database
           0A 50     E9   02B8          BLBC    R0,CONFIG_ERR           ; Branch if couldn't find it
        53    51     D0   02BB          MOVL    R1,R3                   ; Else copy PB addr to right reg
                          02BE
                          02BE  FOUND_PB:
                          02BE
      51   12 A2     3C   02BE          MOVZWL  PPD$W_MTYPE(R2),R1      ; Set event = rec'd dg type
            01FE     31   02C2          BRW     ACTION_DISP             ; Transfer to action dispatcher
                          02C5                                          ;  and return from there
                          02C5
                          02C5  CONFIG_ERR:
                          02C5
                          02C5          BUGCHECK        CIPORT,NONFATAL ; Inconsistent database
                          02CC
           FD31'     31   02CC          BRW     INT$INS_DFREEQ1         ; If nonfatal, discard dg
                          02CF                                          ;  and forget it happened
                          02CF
                          02CF          .DSABL  LSB
```

```
                              02CF            .SBTTL  CNF$STOP_VCS,   SEND STOPS TO ALL VCS
                              02CF
                              02CF    ;+
                              02CF    ; This routine is called during a bugcheck.  It is used to notify
                              02CF    ; other systems to which we have circuits open, that this system is
                              02CF    ; shutting down.  Notification is best try only, no guarantees of
                              02CF    ; success.
                              02CF    ;
                              02CF    ; CNF$STOP_VCS first checks if the PDT is offline.  If so, return
                              02CF    ; is taken since the port is not operating.  Otherwise, the port
                              02CF    ; map is examined to determine each port which is known.  For each
                              02CF    ; known port (except self), a shutdown datagram is sent.  After a hang
                              02CF    ; of an adequate number of milliseconds, the port response queue is
                              02CF    ; rummaged for the sent datagram.  If not found, the port is assumed
                              02CF    ; to be not opreating and return is taken without further notifications.
                              02CF    ; If the sent datagram is found, it is removed from the response queue
                              02CF    ; for reuse in the next host shutdown datagram.
                              02CF    ;
                              02CF    ; Inputs:
                              02CF    ;
                              02CF    ;       R4                              -PDT address
                              02CF    ;
                              02CF    ; Outputs:
                              02CF    ;
                              02CF    ;       R0-R3                           -Destroyed
                              02CF    ;
                              02CF    ;       Other registers                 -Preserved
                              02CF    ;-
                              02CF
                              02CF    ;
                              02CF    ; Shutdown datagram is assembled into the PDT.  It must not be
                              02CF    ; allocated from pool since that is too risky during a bugcheck:
                              02CF    ;
                              02CF
                              02CF
                              02CF            .ENABL  LSB
                              02CF
                              02CF    CNF$STOP_VCS::
                              02CF
   50    00DC C4        D0    02CF            MOVL    PDT$L_UCB0(R4),R0         ; Get UCB address
                 04     E1    02D4            BBC     #UCB$V_ONLINE,-           ; Branch if the port
            23 64 A0          02D6                    UCB$W_STS(R0),20$         ;  is offline
            019C C4     DE    02D9            MOVAL   PDT$Q_TEMP_RSPQ(R4),-     ; Init the temorary response
            019C C4           02DD                    PDT$Q_TEMP_RSPQ(R4)       ;  queue to empty
            019C C4     DE    02E0            MOVAL   PDT$Q_TEMP_RSPQ(R4),-     ;
            01A0 C4           02E4                    PDT$Q_TEMP_RSPQ+4(R4)     ;
   52    01B0 C4        DE    02E7            MOVAL   PDT$B_HSHUT_DG(R4),R2     ; Get addr of host shutdown dg
                 62     7C    02EC            CLRQ    (R2)                      ; Zero self relative links to
                              02EE                                              ;  show dg not queued anywhere
     003B0014 8F        D0    02EE            MOVL    #<PDT$C_HSHUT_SIZ + <DYN$C_CIDG@16>>,-
              08 A2           02F4                    PPD$W_SIZE(R2)            ; Set structure size and type just
                              02F6                                              ;  for completeness
              0569      30    02F6            BSBW    CNF$LKP_PB_PDT            ; Look up 1st/next PB on this PDT
              03 50     E8    02F9            BLBS    R0,FOUND_VC               ; Branch if PB found to start of
                              02FC                                              ;  coroutine processing. Coroutine
                              02FC                                              ;  called back from CNF$LKP_PB_PDT
                              02FC
```

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

```
0380          .SBTTL  ACTION DISPATCHING
0380          .SBTTL  -       ACTION TABLE FORMAT
0380
0380   ;+
0380   ; The ACTION_TABLE is a list of action routines to execute for
0380   ; each combination of port-port VC state and event.  The format
0380   ; of the table is a list of VC state entries.  Each state entry
0380   ; is followed by a list of events possible for that state.  Each
0380   ; event entry is followed by a list of actions to be taken for
0380   ; the event.  The table is arranged linearly.
0380   ;
0380   ; The various entries are generated by the macros STATE, EVENT,
0380   ; ACTION, and ENDACTION defined in the next section.  Actions
0380   ; may return status or not.  For actions which do return status,
0380   ; the action dispatcher checks R0 for success/fail status.  In
0380   ; case of failure the action disRatcher calls routine CLEANUP
0380   ; and terminates action routine execution.
0380   ;
0380   ; The format of the various types of entry in the action table:
0380   ;
0380   ;       STATE:              +--------+--------+--------+--------+
0380   ;                           |offset to nxt st |  state code     |
0380   ;                           +--------+--------+--------+--------+
0380   ;
0380   ;       EVENT:              +--------+--------+--------+--------+
0380   ;                           |offset to nxt evt|  event code     |
0380   ;                           +--------+--------+--------+--------+
0380   ;
0380   ;       ACTION:             +--------+--------+--------+--------+
0380   ;                           |offset to routine|  arg   |  code  |
0380   ;                           +--------+--------+--------+--------+
0380   ;
0380   ; Standard inputs to action routines are:
0380   ;
0380   ;       R1                  -Argument in action table entry
0380   ;       R2                  -Addr of IDREC/START/STACK/ACK dg, if any
0380   ;       R3                  -Addr of PB
0380   ;       R4                  -Addr of PDt
0380   ;
0380   ; The end action actin type is special:  it moves the argument
0380   ; into the PB state word and terminates the list of actions.  End
0380   ; action entries are a single word long.
0380   ;-
```

```
0380          .SBTTL  -       ACTION TABLE MACROS
0380  ;
0380  ; Macro to define a state entry:
0380  ;
0380
0380          .MACRO  STATE   CODE
0380           .NOSHOW
0380           $$$=.                                   ; Save start of state entry
0380           .WORD  CODE                             ; State code
0380           .IF DF $$$LAST_STATE                    ; If there was a previous
0380           .=$$$LAST_STATE+ST$W_NEXT               ;   state entry, go back and
0380           .WORD  $$$-$$$LAST_STATE                ;   file in its fwd link
0380           .=$$$+ST$W_NEXT                         ;   and reset pointer to this entry
0380           .ENDC                                   ;
0380           .WORD  0                                ; Allocate word for fwd link
0380           $$$LAST_STATE=$$$                       ; Define start of this entry
0380           $$$LAST_EVENT=0                         ; Reset addr of last event to
0380                                                   ;   show start of new list of events
0380           .SHOW
0380          .ENDM   STATE
0380
0380  ;
0380  ; Macro to define event entry:
0380  ;
0380
0380          .MACRO  EVENT   CODE
0380           .NOSHOW
0380           $$$=.                                   ; Save start of entry
0380           .WORD  CODE                             ; Event code
0380           .IF NE $$$LAST_EVENT                    ; If there was a previous event,
0380           .=$$$LAST_EVENT+EV$W_NEXT               ;   then go back to it and
0380           .WORD  $$$-$$$LAST_EVENT                ;   fill in its fwd link
0380           .=$$$+EV$W_NEXT                         ;   and return to current entry
0380           .ENDC                                   ;
0380           .WORD  0                                ; Allocate word for fwd link
0380           $$$LAST_EVENT=$$$                       ; Define addr of this entry
0380           .SHOW
0380          .ENDM   EVENT
0380
0380  ;
0380  ; Macro to define action entry:
0380  ;
0380
0380          .MACRO  ACTION ROUTINE,FLAG=0,ARG=0,CODE=AC$C_CONTINUE
0380           .NOSHOW
0380           $$$=.                                   ; Save start of entry
0380           .BYTE  CODE!FLAG                        ; Action type code
0380           .BYTE  ARG                              ; Argument
0380           .WORD  ROUTINE-$$$                      ; Offset to action routine
0380           .SHOW
0380          .ENDM   ACTION
0380
0380  ;
0380  ; Macro to define an endaction entry:
0380  ;
0380
0380          .MACRO  ENDACTION NEWSTATE
```

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

```
                0380       .SBTTL  -       ACTION TABLE OFFSETS AND DEFINITIONS
                0380
                0380  ;
                0380  ; Offsets to state, event and action entries in the action
                0380  ; dispatch table:
                0380  ;
                0380
00000000        0380  ST$W_CODE = 0                          ; State code (codes defined in $PBDEF)
00000002        0380  ST$W_NEXT = 2                          ; Offset to next state entry
                0380
00000000        0380  EV$W_CODE = 0                          ; Event code
00000002        0380  EV$W_NEXT = 2                          ; Offset to next event entry
                0380
00000000        0380  AC$B_CODE = 0                          ; Action code
00000001        0380  AC$B_ARG = 1                           ; Action routine argument
00000001        0380  AC$W_NEWST = 1                         ; New path blk state on end action
00000002        0380  AC$W_ACTION = 2                        ; Offset to action routine
                0380
                0380  ;
                0380  ; Event code definitions:
                0380  ;
                0380
                0380                                         ; Following codes (sign bit clear) assumed equal
                0380                                         ;  to the corresponding PPD msg types:
00000000        0380  EV$C_START = 0                         ;  START dg received
00000001        0380  EV$C_STACK = 1                         ;  STACK dg reveived
00000002        0380  EV$C_ACK = 2                           ;  ACK dg received
00000005        0380  EV$C_ELOG = 5                          ;  Error log dg received
00000006        0380  EV$C_HOSTSHUT = 6                      ;  Host shutdown dg received
                0380                                         ; The following codes are assumed to have
                0380                                         ;  no definition as PPD types that we
                0380                                         ;  will ever receive (needs to be in
                0380                                         ;  architecture that sign bit set codes
                0380                                         ;  are reserved.)
00008000        0380  EV$C_SCSMSG = ^X8000                   ; SCS control msg received (connx
                0380                                         ;  management or credit)
00008001        0380  EV$C_TIMEOUT = ^X8001                  ; Path timer expired
00008002        0380  EV$C_SEND_START = ^X8002               ; Send 1st START, initiate handshake
                0380
                0380  ;
                0380  ; Action code definitions:
                0380  ;
                0380
00000000        0380  AC$C_END = 0                           ; No more action routines, update PB state
00000001        0380  AC$C_CONTINUE = 1                      ; More action routines.
00000080        0380  STATUS = ^X80                          ; If set, action routine returns status
```

```
0380           .SBTTL  -       ACTION TABLE
0380
0380
0380   ACTION_TABLE::
0380
0380           STATE   PB$C_CLOSED                             ; New PB just created
0384
0384             EVENT EV$C_SEND_START                         ; Initiate START handshake
0388               ACTION      SEND_1ST_START                  ; Send 1st START dg
038C               ACTION      START_TIMER                     ; Enable timer
0390               ENDACTION   PB$C_ST_SENT                    ; State moves to start sent
0393
0393             EVENT EV$C_ELOG                               ; Error log dg received
0397               ACTION      REC_ERROR_DG                    ; Go log it
039B               ENDACTION   PB$C_CLOSED                     ; State unchanged
039E
039E           STATE   PB$C_ST_SENT                            ; State= start sent
03A2
03A2             EVENT         EV$C_STACK                      ; Received STACK dg
03A6               ACTION      STOP_TIMER                      ; Disable timer
03AA               ACTION      BUILD_SB,STATUS                 ; Build a formative SB
03AE               ACTION      SET_CIRCUIT,STATUS              ; Tell port to open VC
03B2               ACTION      ENTER_PB,STATUS                 ; Move PB to system database
03B6               ACTION      SEND_ACK                        ; Send ACK
03BA               ENDACTION   PB$C_OPEN                       ; Move PB state to open
03BD
03BD             EVENT         EV$C_START                      ; Received START dg
03C1               ACTION      BUILD_SB,STATUS                 ; Build formative SB
03C5               ACTION      SET_CIRCUIT,STATUS              ; Tell port to open VC
03C9               ACTION      SEND_1ST_STACK                  ; Send STACK dg
03CD               ACTION      START_TIMER                     ; Start a timer
03D1               ENDACTION   PB$C_ST_REC                     ; Move PB state to start rec'd
03D4
03D4             EVENT         EV$C_TIMEOUT                    ; Timer expired
03D8               ACTION      SEND_START,STATUS               ; Retry send of START dg
03DC               ACTION      START_TIMER                     ; Restart timer
03E0               ENDACTION   PB$C_ST_SENT                    ; PB state stays start sent
03E3
03E3             EVENT         EV$C_ELOG                       ; Error log dg received
03E7               ACTION      REC_ERROR_DG                    ; Go log it
03EB               ENDACTION   PB$C_ST_SENT                    ; State unchanged
03EE
03EE           STATE           PB$C_ST_REC                     ; State is start rec'd
03F2
03F2             EVENT         EV$C_ACK                        ; Rec'd ACK dg
03F6               ACTION      IGNORE_DG                       ; Return dg to DFREEQ
03FA               ACTION      STOP_TIMER                      ; Disable timer
03FE               ACTION      ENTER_PB,STATUS                 ; Move PB to system database
0402               ENDACTION   PB$C_OPEN                       ; Move PB state to open
0405
0405             EVENT         EV$C_SCSMSG                     ; Rec'd SCS control msg
0409               ACTION      STOP_TIMER                      ; Stop timer
040D               ACTION      ENTER_PB,STATUS                 ; Move PB to system database
0411               ENDACTION   PB$C_OPEN                       ; Move PB state to open
0414
0414             EVENT         EV$C_STACK                      ; Rec'd STACK dg
0418               ACTION      STOP_TIMER                      ; Disable timer
```

```
041C              ACTION        UPDATE_SWINCARN         ; Copy new incarn # to SB
0420              ACTION        ENTER_PB,STATUS         ; Move PB to system database
0424              ACTION        SEND_ACK                ; Send ACK dg
0428              ENDACTION     PB$C_OPEN               ; Move PB state to open
042B
042B            EVENT           EV$C_START              ; Rec'd START dg
042F              ACTION        UPDATE_SWINCARN         ; Copy new incarn # to SB
0433              ACTION        SEND_1ST_STACK          ; Send STACK dg
0437              ACTION        START_TIMER             ; Start timer
043B              ENDACTION     PB$C_ST_REC             ; PB state stays same
043E
043E            EVENT           EV$C_TIMEOUT            ; Timer expired
0442              ACTION        SEND_STACK,STATUS       ; Try another STACK dg
0446              ACTION        START_TIMER             ; Start up the timer again
044A              ENDACTION     PB$C_ST_REC             ; PB state stays same
044D
044D            EVENT           EV$C_ELOG               ; Error log dg received
0451              ACTION        REC_ERROR_DG            ; Go log it
0455              ENDACTION     PB$C_ST_REC             ; State unchanged
0458
0458          STATE             PB$C_OPEN               ; Path state is open
045C
045C            EVENT           EV$C_STACK              ; Rec'd STACK dg
0460              ACTION        SEND_ACK                ; Send ACK dg
0464              ENDACTION     PB$C_OPEN               ; Leave PB state open
0467
0467
0467            EVENT           EV$C_ACK                ; Rec'd ACK dg
046B              ACTION        IGNORE_DG               ; Return dg to DFREEQ
046F              ENDACTION     PB$C_OPEN               ; Leave PB state open
0472
0472            EVENT           EV$C_START              ; Rec'd START dg on open VC
0476              ACTION        BREAK_PATH              ; Collapse path
047A              ENDACTION     PB$C_VC_FAIL            ;  leaving PB state as set
047D                                                    ;  by BREAK_PATH
047D
047D            EVENT           EV$C_ELOG               ; Error log dg received
0481              ACTION        REC_ERROR_DG            ; Go log it
0485              ENDACTION     PB$C_OPEN               ; State unchanged
0488
0488            EVENT           EV$C_HOSTSHUT           ; Host shutdown received
048C              ACTION        BREAK_HOST              ; Go close VC with special status
0490              ENDACTION     PB$C_VC_FAIL            ; State is vc fail
0493
0493          STATE             PB$C_VC_FAIL            ; VC failure in progress
0497
0497            EVENT           EV$C_START              ; Rec'd START dg
049B              ACTION        IGNORE_DG               ; Discard without action
049F              ENDACTION     PB$C_VC_FAIL            ;
04A2
04A2            EVENT           EV$C_STACK              ; Rec'd STACK dg
04A6              ACTION        IGNORE_DG               ; Discard without action
04AA              ENDACTION     PB$C_VC_FAIL            ;
04AD
04AD            EVENT           EV$C_ACK                ; Rec'd ACK dg
04B1              ACTION        IGNORE_DG               ; Discard without action
04B5              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
                        04C3          .SBTTL  -       ACTION_DISP, ACTION DISPATCHER
                        04C3
                        04C3   ;+
                        04C3   ; The action dispatcher looks up in the action table the list of
                        04C3   ; action routines to execute for the current path block state and
                        04C3   ; the event that occurred.  If an action routine specifies that it
                        04C3   ; returns status, the R0 is checked upon return for success (LBS)
                        04C3   ; or failure (LBC).  On failure the cleanup routine, CLEANUP, is called
                        04C3   ; and ACTION_DISP exits.  Normally, action routines are executed
                        04C3   ; until an end action routine is encountered.  The end action automatically
                        04C3   ; sets the path block state to the value specified in the end action
                        04C3   ; argument.
                        04C3   ;
                        04C3   ; The following register conventions apply for action routines:
                        04C3   ;
                        04C3   ;      R2                -Addr of START/STACK/ACK/IDREC dg, if any
                        04C3   ;      R3                -Addr of formative PB
                        04C3   ;      R4                -Addr of PDT
                        04C3   ;      R5                -Addr of current action entry
                        04C3   ;
                        04C3   ; Actions may use R0 and R1, but must use R2 with care.  Action
                        04C3   ; routines must preserve all other registers.
                        04C3   ;
                        04C3   ; Inputs to ACTION_DISP:
                        04C3   ;
                        04C3   ;      R1                           -Event code
                        04C3   ;      R2-R4                        -As shown above
                        04C3   ;
                        04C3   ; Outputs:
                        04C3   ;
                        04C3   ;      R0-R2                        -Destroyed
                        04C3   ;      other registers              -Preserved
                        04C3   ;-
                        04C3
                        04C3   ASSUME  EV$C_START      EQ  0           ; Assume that events START and
                        04C3   ASSUME  EV$C_STACK      EQ  1           ;  STACK are .LE. 1
                        04C3   ASSUME  EV$C_ACK        EQ  2           ; Assume that events associated with
                        04C3                                           ;  rec'd dgs are .LE. 2
                        04C3
                        04C3   ASSUME  PB$C_CLOSED     EQ  0           ; Assume that all the
                        04C3   ASSUME  PB$C_ST_SENT    EQ  1           ;  formative path block states
                        04C3   ASSUME  PB$C_ST_REC     EQ  2           ;  are .LE. 2
                        04C3
                        04C3          .ENABL  LSB
                        04C3
                        04C3   ACTION_DISP:
                        04C3
             55    DD   04C3          PUSHL   R5                      ; Save a register
             51    DD   04C5          PUSHL   R1                      ; Save event code
   55   FEB5 CF    DE   04C7          MOVAL   ACTION_TABLE,R5         ; Get addr of action table
                        04CC
                        04CC   NEXT_STATE:
                        04CC
        50   65    B0   04CC          MOVW    ST$W_CODE(R5),R0        ; Get next state code
     12 A3   50    B1   04CF          CMPW    R0,PB$W_STATE(R3)       ; State codes match?
             0B    13   04D3          BEQL    LOOKUP_EVENT            ; Branch if so
   50     02 A5    32   04D5          CVTWL   ST$W_NEXT(R5),R0        ; Get offset to next state
```

```
               4C    13  04D9               BEQL    PB_STATE_ERR              ; Branch if no more states
          55   50    C0  04DB               ADDL    R0,R5                     ; Else step to nxt state entry
               EC    11  04DE               BRB     NEXT_STATE                ;  and try it
                         04E0
                         04E0  LOOKUP_EVENT:
                         04E0
               85    D5  04E0               TSTL    (R5)+                     ; Step to start of event list
                         04E2
                         04E2  NEXT_EVENT:
                         04E2
          51   65    B1  04E2               CMPW    EV$W_CODE(R5),R1          ; Event codes match?
               0B    13  04E5               BEQL    NEXT_ACTION               ; Branch if yes
      50    02 A5    32  04E7               CVTWL   EV$W_NEXT(R5),R0          ; Get offset to next event
               3A    13  04EB               BEQL    PB_STATE_ERR              ; Branch if no more events
          55   50    C0  04ED               ADDL    R0,R5                     ; Else step to next event entry
               F0    11  04F0               BRB     NEXT_EVENT                ;  and try it
                         04F2
                         04F2  NEXT_ACTION:
                         04F2
               85    D5  04F2               TSTL    (R5)+                     ; Step to 1st/next action entry
               65    95  04F4               TSTB    (R5)                      ; end of action routines?
               23    13  04F6               BEQL    END_ACTION                ; Branch if so
      51    01 A5    9A  04F8               MOVZBL  AC$B_ARG(R5),R1           ; Pick up argument
      50    02 A5    32  04FC               CVTWL   AC$W_ACTION(R5),R0        ; Get offset to routine
             6540    16  0500               JSB     (R5)[R0]                  ; Call action routine
               65    95  0503               TSTB    (R5)                      ; Does routine return status?
               EB    14  0505               BGTR    NEXT_ACTION               ; Branch if not
          E8   50    E8  0507               BLBS    R0,NEXT_ACTION            ; Branch if status good
             51 8ED0     050A               POPL    R1                        ; Retreive event code
        01     51    D1  050D               CMPL    R1,#EV$C_STACK            ; Is it rec'd START or STACK dg?
               03    14  0510               BGTR    10$                       ; Branch if not
           FAEB'     30  0512               BSBW    INT$INS_DFREEQ1           ; Else must return rec'd dg to
                         0515                                                 ;  free queue to prevent depletion
                         0515
             55 8ED0     0515  10$:         POPL    R5                        ; Restore R5
             02C3    31  0518               BRW     CLEANUP                   ; Else xfer to PB/SB cleanup and
                         051B                                                 ;  return from there
                         051B
                         051B  END_ACTION:
                         051B
            01 A5    B0  051B               MOVW    AC$W_NEWST(R5),-          ; Update state of path block
         12 A3           051E                       PB$W_STATE(R3)            ;
             51 8ED0     0520               POPL    R1                        ; Clear event type code from stack
                         0523
             55 8ED0     0523  20$:         POPL    R5                        ; Restore R5
                     05  0526               RSB                               ; Return
                         0527
                         0527  PB_STATE_ERR:
                         0527
             51 8ED0     0527               POPL    R1                        ; Retreive event code
               51    D5  052A               TSTL    R1                        ; Indicate that dg is held?
               03    19  052C               BLSS    30$                       ; Branch if not
           FACF'     30  052E               BSBW    INT$INS_DFREEQ1           ; Else return PPD handshake dg
                         0531                                                 ;  to free queue
                         0531
         12 A3       B1  0531  30$:         CMPW    PB$W_STATE(R3),-          ; Is path state in formative
               02        0534                       #PB$C_ST_REC              ;  state?
```

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

```
                 0539              .SBTTL  ACTION ROUTINES
                 0539              .SBTTL  -        SEND_1ST_START, SEND 1ST START DG
                 0539              .SBTTL  -        SEND_START, SEND A START DATAGRAM
                 0539
                 0539      ;+
                 0539      ; SEND_START allocates a dtagram buffer from nonpaged pool,
                 0539      ; formats a START message in it and sends the datagram.  The data
                 0539      ; that goes into the START message is assembled into the message
                 0539      ; by routine FMT_START_DATA.
                 0539      ;
                 0539      ; SEND_START has two entries:  SEND_1ST_START which resets the START
                 0539      ; retry count and SEND_START which decrements and checks the retry
                 0539      ; count before sending the datagram.
                 0539      ;
                 0539      ; The retries must continue until the target remote port is polled
                 0539      ; again.  This time depends on the interval between poller wakeups,
                 0539      ; the number of ports being polled at each poller wakeup, the total
                 0539      ; number of ports to be polled, and the time between retries
                 0539      ; (SCS$GW_PASTMOUT) as follows:
                 0539      ;
                 0539      ;       # retries = (SCS$GB_PAMXPORT * SCS$GW_PAPOLINT) /
                 0539      ;                       (SCS$GB_PANPOLL * SCS$GW_PASTMOUT)
                 0539      ;
                 0539      ; The retry count is computed each time it's set since the dependent
                 0539      ; variables are dynamic SYSGEN parameters.
                 0539      ;
                 0539      ; SEND_START may fail for two reasons:  insufficient pool to
                 0539      ; allocate the datagram buffer, or retry count exceeded.
                 0539      ;
                 0539      ; Inputs:
                 0539      ;
                 0539      ;       R2                              -Addr of datagram to turn around (1ST_START)
                 0539      ;       R3                              -Addr of PB
                 0539      ;       R4                              -Addr of PDT
                 0539      ;
                 0539      ; Outputs:
                 0539      ;
                 0539      ;       R0                              -0/1 for fail/success (SEND_START only)
                 0539      ;       R1,R2                           -Destroyed
                 0539      ;       other registers                 -Preserved
                 0539      ;-
                 0539
                 0539      ;
                 0539      ; PPD message format assumption:
                 0539      ;
                 0539
                 0539      ASSUME  PPD$W_LENGTH+2  EQ PPD$W_MTYPE
                 0539
                 0539              .ENABL  LSB
                 0539
                 0539      SEND_1ST_START:
                 0539
        3E   D0  0539              MOVL    #<PPD$C_START@16 + PPD$C_START_LEN>,-
     10 A2       053B              PPD$W_LENGTH(R2)        ; Set dg size and type
        24   11  053D              BRB     COM_SEND_1              ; Go do it
                 053F
                 053F      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
```

```
                                   055B          .SBTTL  -       SEND_STACK, SEND A STACK DATAGRAM
                                   055B
                                   055B    ;+
                                   055B    ; This routine has two entries:
                                   055B    ;
                                   055B    ; SEND_1ST_STACK resets the retry count for sending STACK's and
                                   055B    ; recycles the received START datagram into a STACK message.
                                   055B    ; See SEND_1ST_START comments regarding calculation of the
                                   055B    ; retry count.   This entry always completes with success.
                                   055B    ;
                                   055B    ; SEND_STACK is called when the timer expires and a retry is
                                   055B    ; necessary.  It decrements and checks the retry count.  If more retries
                                   055B    ; remain, it allocates a datagram buffer from pool.  This entry can
                                   055B    ; fail due to expired retry count or insufficient pool.
                                   055B    ;
                                   055B    ; Both entries wind up by formatting and sending a STACK datagram.
                                   055B    ;
                                   055B    ; Inputs:
                                   055B    ;
                                   055B    ;       R2                              -Addr of rec'd datagram (if 1ST_STACK)
                                   055B    ;       R3                              -Addr of PB
                                   055B    ;       R4                              -Addr of PDT
                                   055B    ;
                                   055B    ; Outputs:
                                   055B    ;
                                   055B    ;       R0                              -0/1 for fail/success
                                   055B    ;       R1,R2                           -Destroyed
                                   055B    ;       other registers                 -Preserved
                                   055B    ;-
                                   055B
                                   055B    ;
                                   055B    ; PPD message format assumption:
                                   055B    ;
                                   055B
                                   055B
                                   055B          .ENABL  LSB
                                   055B
                                   055B    SEND_1ST_STACK:
                                   055B
           0001003E 8F     D0      055B          MOVL    #<PPD$C_STACK@16 + PPD$C_STACK_LEN>,-
                  10 A2            0561                  PPD$W_LENGTH(R2)         ; Set dg size and type
                                   0563
                                   0563    COM_SEND_1:
                                   0563
    50     00000000'GF     9A      0563          MOVZBL  G^SCS$GB_PAMXPORT,R0     ; Get maximum number of ports
    50     00000000'GF     A4      056A          MULW2   G^SCS$GW_PAPOLINT,R0     ; Compute maximum port #
                                   0571                                           ;  * poller interval
    51     00000000'GF     9A      0571          MOVZBL  G^SCS$GB_PANPOLL,R1      ; Get # ports to poll per interval
    51     00000000'GF     A4      0578          MULW2   G^SCS$GW_PASTMOUT,R1     ; Compute # ports to poll per
                                   057F                                           ;  interval * start timeout
        50     50     51   C7      057F          DIVL3   R1,R0,R0                 ; Divide, increment in case of
     22 A3     50     01   A1      0583          ADDW3   #1,R0,PB$W_RETRY(R3)     ;  remainder, and save retry count
                  0204     30      0588          BSBW    FMT_START_DATA           ; Set up start data
                  037A     30      058B          BSBW    SNDDG_NORET              ; Send dg with RETFLAG=FALSE
                                   058E                                           ;  to channel dg buffer back to
                                   058E                                           ;  free queue
                    C4     11      058E          BRB     SEND_SUCCESS             ; Take success exit
```

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

```
                            05AC         .SBTTL  -       SEND_ACK, SEND ACK DATAGRAM
                            05AC
                            05AC  ;+
                            05AC  ; SEND_ACK turns a previously received STACK datagram into an
                            05AC  ; ACK and sends the datagram.  No failures are possible.
                            05AC  ;
                            05AC  ; Inputs:
                            05AC  ;
                            05AC  ;      R2                          -Addr of dg being turned around
                            05AC  ;      R3                          -Addr of PB
                            05AC  ;      R4                          -Addr of PDT
                            05AC  ;
                            05AC  ; Outputs:
                            05AC  ;
                            05AC  ;      R0,R1                       -Destroyed
                            05AC  ;      other registers             -Preserved
                            05AC  ;-
                            05AC
                            05AC  ;
                            05AC  ; PPD message format assumption:
                            05AC  ;
                            05AC
                            05AC  ASSUME  PPD$W_LENGTH+2  EQ PPD$W_MTYPE
                            05AC
                            05AC         .ENABL  LSB
                            05AC
                            05AC  SEND_ACK:
                            05AC
00020004 8F         D0      05AC         MOVL    #<PPD$C_ACK@16 + PPD$C_ACK_LEN>,-
      10 A2                 05B2                 PPD$W_LENGTH(R2)          ; Set dg size and type
      0351          31      05B4         BRW     SNDDG_NORET               ; Send dg with RETFLAG=FALSE
                            05B7                                           ;  to channel dg buffer back
                            05B7                                           ;  free queue.
                            05B7
                            05B7         .DSABL  LSB
```

```
                  05B7          .SBTTL  -       UPDATE_INCARN,  UPDATE SW INCARN FROM
                  05B7          .SBTTL  -                       2ND START/STACK
                  05B7
                  05B7  ;+
                  05B7  ; This routine exists primarily for the convenience of the HSC
                  05B7  ; which wants to sent its incarnation to its startup time, but
                  05B7  ; does not have a clock.  The HSC uses the first PPD$Q_CURTIME
                  05B7  ; it sees in a START/STACK that is nonzero as its start time.
                  05B7  ; Until it receives the time from some system in the cluster,
                  05B7  ; it conducts start handshakes with a software incarnation number
                  05B7  ; of zero.
                  05B7  ;
                  05B7  ; If VMS receives a START from the HSC before the HSC has set
                  05B7  ; its start time, then the received START has an incarnation number
                  05B7  ; of zero.  A subsequent START/STACK from the HSC will however have
                  05B7  ; a proper incarnation number which is used by this routine to
                  05B7  ; revise the formative SB.
                  05B7  ;
                  05B7  ; Inputs:
                  05B7  ;
                  05B7  ;       R2                      -Addr of START/STACK dg
                  05B7  ;       R3                      -Addr of formative PB
                  05B7  ;       R4                      -Addr of PDT
                  05B7  ;
                  05B7  ; Outputs:
                  05B7  ;
                  05B7  ;       R0                      -Destroyed
                  05B7  ;       Other registers         -Preserved
                  05B7  ;-
                  05B7
                  05B7          .ENABL  LSB
                  05B7
                  05B7  UPDATE_SWINCARN:
                  05B7
50    30 A3   D0  05B7          MOVL    PB$L_SBLINK(R3),R0      ; Get formative SB
      28 A2   7D  05BB          MOVQ    PPD$Q_SWINCARN(R2),-    ; Update formative SB with
      2C A0       05BE                  SB$Q_SWINCARN(R0)       ;  latest SW incarnation #
              05  05C0          RSB                             ; Return
                  05C1
                  05C1          .DSABL  LSB
```

```
05C1        .SBTTL  -       ENTER_PB, MOVE PB (AND SB) FROM FORMATIVE
05C1        .SBTTL  -                 LISTS TO SYSTEM WIDE DATABASE
05C1
05C1  ;+
05C1  ; ENTER_PB moves a pathblock and, if necessary, its associated system
05C1  ; block from the formative pathblock list to the system wide
05C1  ; configuration database.  In the process, and SCS send message
05C1  ; buffer and receive buffer, and SETCKT dg are allocated.  The send
05C1  ; buffer address is stored in the PB and the receive buffer is queued to
05C1  ; the port. If the allocation fails, the path block ad system block remain
05C1  ; on the formative list and error exit is taken.
05C1  ;
05C1  ; What happens to the formative system block depends upon the current
05C1  ; database:
05C1  ;
05C1  ;       -If a matching SB does not already exist,
05C1  ;        then the formative SB is inserted in the database along
05C1  ;        with its formative PB.
05C1  ;
05C1  ;       -If a matching system exists, then check if the
05C1  ;        existing SB has any PB's linked to it.  If not, refresh the
05C1  ;        old SB with information from the formative SB and link the
05C1  ;        formative PB to the refreshed SB.
05C1  ;
05C1  ;       -If the existing matching SB has paths to it, check if the
05C1  ;        existing SB and formative SB have the same software incarnation.
05C1  ;        If not, then two different systems must be masquarading as the
05C1  ;        same system ID and the formative SB and PB are thrown away
05C1  ;        (we refuse to talk to the newcomer.)
05C1  ;
05C1  ;        If the incarnation numbers match, then we just add the formative
05C1  ;        PB to the existing SB's list of paths and discard the formative
05C1  ;        SB.
05C1  ;
05C1  ; A matching system means one that matches in both system ID and node
05C1  ; name.  SB's that match in one, but not the other are rejected and no
05C1  ; vc will be opened to such a system.
05C1  ;
05C1  ; Naturally, there is an exception to the rule excluding systems with
05C1  ; the same node name.  Version 3.x systems with matching node names
05C1  ; but unique system ID's will be permitted to enter the database.
05C1  ; This is because 3.x systems all had the same node name (all blanks)
05C1  ; and their presence will have no effect on the VAXcluster sysap
05C1  ; in a 4.x system.
05C1  ;
05C1  ; Inputs:
05C1  ;
05C1  ;       R3                              -Addr of formative PB
05C1  ;       R4                              -Addr of PDT
05C1  ;
05C1  ; Outputs:
05C1  ;
05C1  ;       R0                              -0/1 for fail/success
05C1  ;       R1                              -Destroyed
05C1  ;       other registers                 -Preserved
05C1  ;-
05C1
```

```
                           05C1  ;
                           05C1  ; System Block adjacency assumptions:
                           05C1  ;
                           05C1
                           05C1  ASSUME   SB$B_SYSTEMID+8 EQ   SB$W_MAXDG
                           05C1  ASSUME   SB$W_MAXDG+2    EQ   SB$W_MAXMSG
                           05C1  ASSUME   SB$W_MAXMSG+2   EQ   SB$T_SWTYPE
                           05C1  ASSUME   SB$T_SWTYPE+4   EQ   SB$T_SWVERS
                           05C1  ASSUME   SB$T_SWVERS+4   EQ   SB$Q_SWINCARN
                           05C1  ASSUME   SB$Q_SWINCARN+8 EQ   SB$T_HWTYPE
                           05C1  ASSUME   SB$T_HWTYPE+4   EQ   SB$B_HWVERS
                           05C1  ASSUME   SB$B_HWVERS+12  EQ   SB$T_NODENAME
                           05C1  ASSUME   SB$T_NODENAME+16 EQ  SB$L_DDB
                           05C1
           0000003C        05C1  UPDATE_LEN = SB$L_DDB-SB$B_SYSTEMID
                           05C1
                           05C1           .ENABL  LSB
                           05C1
                           05C1  ENTER_PB:
                           05C1
                52    DD   05C1           PUSHL   R2                          ; Save R2
              FA3A'   30   05C3           BSBW    INT$ALLOC_MSG               ; Allocate a msg buffer
             03 50    E8   05C6           BLBS    R0,10$                      ; Branch if got it
              0114    31   05C9           BRW     ENTER_ERR                   ; Else go to error
                           05CC
         40 A3   52   D0   05CC  10$:     MOVL    R2,PB$L_SCSMSG(R3)          ; Assign buffer to PB for SCS
                           05D0                                               ;  control messages sent
              FA2D'   30   05D0           BSBW    INT$ALLOC_PPDDG             ; Allocate a PPD dg buffer
             03 50    E8   05D3           BLBS    R0,30$                      ; Branch if got it
              00D8    31   05D6           BRW     ENTER_ERR1                  ; Else go clean up
                           05D9
         54 A3   52   D0   05D9  30$:     MOVL    R2,PB$L_CLSCKT_DG(R3)       ; Save addr of PPD dg
              FA20'   30   05DD           BSBW    INT$ALLOC_MSG               ; Allocate a msg buffer for
                           05E0                                               ;  SCS control msg receive
             03 50    E8   05E0           BLBS    R0,40$                      ; Branch if got it
              00CB    31   05E3           BRW     ENTER_ERR2                  ; Else handle error
                           05E6
              FA17'   30   05E6  40$:     BSBW    INT$INS_MFREEQ              ; Queue buffer to port
       50     30 A3   D0   05E9           MOVL    PB$L_SBLINK(R3),R0          ; Get addr of formative SB
  52   00000000'GF    DE   05ED           MOVAL   G^SCS$GQ_CONFIG,R2          ; Get SB listhead
          51     52   D0   05F4           MOVL    R2,R1                       ; Hold starting point
                           05F7
                           05F7  CMP_EXIST_SBS:
                           05F7
          52     62   D0   05F7           MOVL    (R2),R2                     ; Get next SB in list
          51     52   D1   05FA           CMPL    R2,R1                       ; Back where we started?
                 75   13   05FD           BEQL    MOVE_SB                     ; Branch if so, this system
                           05FF                                               ;  isn't here
             18 A0    D1   05FF           CMPL    SB$B_SYSTEMID(R0),-         ; Check for system ID match
             18 A2         0602                   SB$B_SYSTEMID(R2)           ;  on low 4 bytes
                 07   12   0604           BNEQ    50$                         ; Branch if no match
             1C A0    B1   0606           CMPW    SB$B_SYSTEMID+4(R0),-       ; Check for system ID match
             1C A2         0609                   SB$B_SYSTEMID+4(R2)         ;
                 16   13   060B           BEQL    55$                         ; Branch if matches
                           060D
             29 A0    B1   060D  50$:     CMPW    SB$T_SWVERS+1(R0),-         ; Is the formative system block
           2E33 8F         0610                   #^A/3./                     ;  for a V3.n system?
```

```
                 E2   13  0613           BEQL    CMP_EXIST_SBS               ; Branch if so and bypass node name
                          0615                                               ;  uniqueness test
                 0F   BB  0615           PUSHR   #^M<R0,R1,R2,R3>            ; Save registers destroyed in CMPC
         44 A0   10   29  0617           CMPC3   #16,SB$T_NODENAME(R0),-     ; Are node names the same?
            44 A2         061B                   SB$T_NODENAME(R2)           ;
                 0E   13  061D           BEQL    56$                         ; Branch if node names are same,
                          061F                                               ;  but SYSIDs are not -- can't
                          061F                                               ;  talk to this system because
                          061F                                               ;  there is a configuration error
                 0F   BA  061F           POPR    #^M<R0,R1,R2,R3>            ; Restore registers
                 D4   11  0621           BRB     CMP_EXIST_SBS               ; Continue searching existing SBs
                          0623
                 0F   BB  0623  55$:     PUSHR   #^M<R0,R1,R2,R3>            ; Save reg destroyed by cmpc
         44 A0   10   29  0625           CMPC3   #16,SB$T_NODENAME(R0),-     ; Do the system's node names
            44 A2         0629                   SB$T_NODENAME(R2)           ;  match?
                 03   13  062B           BEQL    57$                         ; Continue if so
               00AC   31  062D  56$:     BRW     ENTER_ERR4                  ; Branch if not -- don't talk to
                          0630                                               ;  this system
                 0F   BA  0630  57$:     POPR    #^M<R0,R1,R2,R3>            ; Restore destroyed registers
            14 A2     D5  0632           TSTL    SB$L_PBCONNX(R2)            ; Does existing SB have paths?
                 27   12  0635           BNEQ    CHK_INCARN_ERR              ; If so, go check for
                          0637                                               ;  inconsistent incarnations
                          0637
                          0637  REFRESH_SB:
                          0637
 00000000'8F     52   D1  0637           CMPL    R2,#SCS$GA_LOCALSB          ; Is this the local SB?
                 0E   12  063E           BNEQ    DO_REFRESH                  ; Branch if not
            2C A0     D1  0640           CMPL    SB$Q_SWINCARN(R0),-         ; Else is the new incarnation the
            2C A2         0643                   SB$Q_SWINCARN(R2)           ;  same as the old?
                 73   12  0645           BNEQ    ENTER_ERR3                  ; Branch if not -- this must be
            30 A0     D1  0647           CMPL    SB$Q_SWINCARN+4(R0),-       ;  a different host masquerading
            30 A2         064A                   SB$Q_SWINCARN+4(R2)         ;  as us
                 6C   12  064C           BNEQ    ENTER_ERR3                  ;
                          064E
                          064E  DO_REFRESH:
                          064E
         14 A2   53   D0  064E           MOVL    R3,SB$L_PBCONNX(R2)         ; Set formative PB as first path
                          0652                                               ;  to use for a connx in old SB
                 3F   BB  0652           PUSHR   #^M<R0,R1,R2,R3,R4,R5>      ; Save regs destroyed by movc
                 3C   28  0654           MOVC3   #UPDATE_LEN,-               ; Update old SB with new
            18 A0         0656                   SB$B_SYSTEMID(R0),-         ;  SB info
            18 A2         0658                   SB$B_SYSTEMID(R2)           ;   from start handshake dg
                 3F   BA  065A           POPR    #^M<R0,R1,R2,R3,R4,R5>      ; Restore registers destroyed
                 0E   11  065C           BRB     DELETE_SB                   ; Go delete new SB and complete
                          065E                                               ;  entering PB in database
                          065E
                          065E  CHK_INCARN_ERR:
                          065E
            2C A0     D1  065E           CMPL    SB$Q_SWINCARN(R0),-         ; Is this the same incarnation of
            2C A2         0661                   SB$Q_SWINCARN(R2)           ;  of the system we've already got?
                 55   12  0663           BNEQ    ENTER_ERR3                  ; Branch if not because this means
            30 A0     D1  0665           CMPL    SB$Q_SWINCARN+4(R0),-       ;  the system is really a different
            30 A2         0668                   SB$Q_SWINCARN+4(R2)         ;  system with the same system ID
                 4E   12  066A           BNEQ    ENTER_ERR3                  ;
                          066C
                          066C  ;
                          066C  ; This system already has an SB in the database.  Delete formative
```

```
                                066C  ; SB and insert formative path block only into the system wide
                                066C  ; configuration database.  R0 has the address of the formative SB.
                                066C  ;
                                066C
                                066C  DELETE_SB:
                                066C
          00000000'GF     16    066C          JSB     G^COM$DRVDEALMEM          ; Deallocate it to pool
                 0B       11    0672          BRB     MOVE_PB                   ; Join common PB move
                                0674
                                0674  ;
                                0674  ; This system is new.  Move the formative SB to the system wide
                                0674  ; configuration database and link formative PB to it.  R0 has the
                                0674  ; address of the formative SB.
                                0674  ;
                                0674
                                0674  MOVE_SB:
                                0674
                 52   50  D0    0674          MOVL    R0,R2                     ; Copy addr of formative SB
              04 B1   62  0E    0677          INSQUE  (R2),@4(R1)               ; Insert formative SB on tail of
                                067B                                            ;  system configuration list
              14 A2   53  D0    067B          MOVL    R3,SB$L_PBCONNX(R2)       ; Set formative PB as first
                                067F                                            ;  path to use for a connection
                                067F
                                067F  MOVE_PB:
                                067F
                 53   63  0F    067F          REMQUE  (R3),R3                   ; Remove formative path block
              10 B2   63  0E    0682          INSQUE  (R3),@SB$L_PBBL(R2)       ;  and link to system block
                      06  12    0686          BNEQ    60$                       ; Branch if not block in list
                                0688  ;
                                0688  ; Give notification that the SB is new or reused
                                0688  ;
                                0688  ;       R2 -> SB
                                0688  ;       R0,R1 need not be preserved
                                0688  ;
          00000000'GF     16    0688          JSB     G^SCS$NEW_SB              ; Note the new SB
                                068E  60$:
              30 A3   52  D0    068E          MOVL    R2,PB$L_SBLINK(R3)        ; Save final SB addr in PB
                   38 A3  DE    0692          MOVAL   PB$L_WAITQFL(R3),-        ; Set PB general wait queue
                   38 A3        0695                  PB$L_WAITQFL(R3)          ;  to no entries
                   38 A3  DE    0697          MOVAL   PB$L_WAITQFL(R3),-        ;
                   3C A3        069A                  PB$L_WAITQBL(R3)          ;
                 0112 C4  B6    069C          INCW    PDT$W_PBCOUNT(R4)         ; Step count of PB's on this PDT
              50   0C A3  D0    06A0          MOVL    PB$B_RSTATION(R3),R0      ; Retrieve the remote port number
         00 0134 C4   50  E5    06A4          BBCC    R0,PDT$B_PLOGMAP(R4),65$; Clear bit in error logging mask
                                06AA                                            ; corresponding to remote port number
                 50   01  3C    06AA  65$:    MOVZWL  #SS$_NORMAL,R0            ; Set status = success
                                06AD
                                06AD  ENTER_DONE:
                                06AD
                     52 8ED0    06AD          POPL    R2                        ; Restore saved register
                          05    06B0          RSB                               ; Return
                                06B1
                                06B1  ENTER_ERR1:
                                06B1  ENTER_ERR2:
                                06B1
              52   40 A3  D0    06B1          MOVL    PB$L_SCSMSG(R3),R2        ; Get addr of SCS send buffer
                   F948'  30    06B5          BSBW    INT$DEAL_MSG              ;  and return to pool
              26    11  06B8          BRB     ENTER_ERR                 ; Join common error exit
                        06BA
                        06BA  ENTER_ERR3:
                        06BA
         51   0C A3 D0  06BA          MOVL    PB$B_RSTATION(R3),R1      ; Retrieve the remote port number
   13 0134 C4   51  E2  06BE          BBSS    R1,PDT$B_PLOGMAP(R4),70$; Branch if remote port already logged
                55  DD  06C4          PUSHL   R5                        ; Otherwise save R5
         55     52  D0  06C6          MOVL    R2,R5                     ; Move known system SB address into R5
                52  D4  06C9          CLRL    R2                        ; Indicate that there is no packet
         51     53  D0  06CB          MOVL    R3,R1                     ; Move remote PB address address into R3
         50     08  9A  06CE          MOVZBL  #PAER$K_ES_RSCKS,R0       ; Set the appropriate error subtype
           F92C'    30  06D1          BSBW    ELOG$PACKET               ; Go log conflict
              55 8ED0   06D4          POPL    R5                        ; Restore R5
                        06D7
           F926'    30  06D7  70$:    BSBW    INT$MFQ2POOL              ; Remove queued SCS recv buffer
              D5    11  06DA          BRB     ENTER_ERR2                ; Join rest of error handling
                        06DC
                        06DC  ENTER_ERR4:
                        06DC
              0F    BA  06DC          POPR    #^M<R0,R1,R2,R3>          ; Restore reg lost in node name
                        06DE                                            ;   comparison
              DA    11  06DE          BRB     ENTER_ERR3                ; Join common cleanup
                        06E0
                        06E0  ENTER_ERR:
                        06E0
       52     54 A3 D0  06E0          MOVL    PB$L_CLSCKT_DG(R3),R2     ; Get the close circuit dg addr
              06    12  06E4          BNEQ    80$                       ; Branch if got one
           F917'    30  06E6          BSBW    INT$ALLOC_PPDDG           ; Else allocate a dg buffer
           C1 50    E9  06E9          BLBC    R0,ENTER_DONE             ; Branch if no pool -- this vc will
                        06EC                                            ;   dangle till somebody tries to use
                        06EC                                            ;   it by sending a connect request.
                        06EC                                            ;   At that time we have another chance
                        06EC                                            ;   to set it closed.
                        06EC
                    C9  06EC  80$:    BISL3   #<PPD$M_RSP@24>!-         ; Format the dg into a SETCKT
                        06ED                  <PPD$C_SETCKT@16>,-       ;
  0C A3   01190000 8F       06ED              PB$B_RSTATION(R3),-       ;
                0C A2   06F4                  PPD$B_PORT(R2)            ;
    10 A2   8000 8F 3C  06F6          MOVZWL  #PPD$M_CST,PPD$W_MASK(R2)
            14 A2   D4  06FC          CLRL    PPD$W_M_VAL(R2)           ;  and ask for vc state to be closed
           F8FE'    30  06FF          BSBW    INT$INS_COMQH             ; Do it at high priority
              50    D4  0702          CLRL    R0                        ; Set status to failed
              A7    11  0704          BRB     ENTER_DONE                ; Go to exit routine
                        0706
                        0706          .DSABL  LSB
```

```
                                   0706        .SBTTL  -       BUILD_SB, BUILD A FORMATIVE SYSTEM BLOCK
                                   0706
                                   0706   ;+
                                   0706   ; BUILD_SB allocates a system block from nonpaged pool and sets
                                   0706   ; it up with information from the received START or STACK datagram.
                                   0706   ; If insufficient pool is available, then the routine returns failure.
                                   0706   ;
                                   0706   ; Inputs:
                                   0706   ;
                                   0706   ;       R2                              -Addr of START/STACK dg
                                   0706   ;       R3                              -Addr of formative PB
                                   0706   ;       R4                              -Addr of PDT
                                   0706   ;
                                   0706   ; Outputs:
                                   0706   ;
                                   0706   ;       R0                              -0/1 for fail/success
                                   0706   ;       R1                              -Destroyed
                                   0706   ;       other registers                 -Preserved
                                   0706   ;-
                                   0706
                                   0706   ;
                                   0706   ; Data structure adjacency assumptions:
                                   0706   ;
                                   0706
                                   0706   ASSUME  SB$B_SYSTEMID+8 EQ SB$W_MAXDG
                                   0706   ASSUME  SB$W_MAXDG+2    EQ SB$W_MAXMSG
                                   0706   ASSUME  SB$W_MAXMSG+2   EQ SB$T_SWTYPE
                                   0706   ASSUME  SB$T_SWTYPE+4   EQ SB$T_SWVERS
                                   0706   ASSUME  SB$T_SWVERS+4   EQ SB$Q_SWINCARN
                                   0706   ASSUME  SB$Q_SWINCARN+8 EQ SB$T_HWTYPE
                                   0706   ASSUME  SB$T_HWTYPE+4   EQ SB$B_HWVERS
                                   0706   ASSUME  SB$T_NODENAME+16 EQ SB$L_DDB
                                   0706
                                   0706   ASSUME  PPD$B_SYSTEMID+8 EQ PPD$W_MAXDG
                                   0706   ASSUME  PPD$W_MAXDG+2   EQ PPD$W_MAXMSG
                                   0706   ASSUME  PPD$W_MAXMSG+2  EQ PPD$T_SWTYPE
                                   0706   ASSUME  PPD$T_SWTYPE+4  EQ PPD$T_SWVERS
                                   0706   ASSUME  PPD$T_SWVERS+4  EQ PPD$Q_SWINCARN
                                   0706   ASSUME  PPD$Q_SWINCARN+8 EQ PPD$T_HWTYPE
                                   0706   ASSUME  PPD$T_HWTYPE+4  EQ PPD$B_HWVERS
                                   0706   ASSUME  PPD$Q_NODENAME+8 EQ PPD$Q_CURTIME
                                   0706
                         0000002C  0706   DATA_LEN = <SB$B_HWVERS+12> - <SB$B_SYSTEMID>
                                   0706
                                   0706           .ENABL  LSB
                                   0706
                                   0706   BUILD_SB:
                                   0706
                   3C      BB      0706           PUSHR   #^M<R2,R3,R4,R5>            ; Save a bunch of registers
      51   00000060 8F     D0      0708           MOVL    #SB$K_LENGTH,R1             ; Get size of SB
           00000000'GF     16      070F           JSB     G^EXE$ALONONPAGED           ; Allocate from nonpaged pool
                 54 50     E9      0715           BLBC    R0,SB_DONE                  ; Branch if no pool
              08 A2  51    B0      0718           MOVW    R1,SB$W_SIZE(R2)            ; Set struct size
              0760 8F      B0      071C           MOVW    #DYN$C_SCS+<DYN$C_SCS_SB@8>,-   ; Set structure type
                 0A A2             0720                   SB$B_TYPE(R2)               ;   and subtype
                 0C A2     DE      0722           MOVAL   SB$L_PBFL(R2),-             ; Set path block list head
                 0C A2             0725                   SB$L_PBFL(R2)               ;  to empty
           0C A2   DE  0727          MOVAL   SB$L_PBFL(R2),-            ;
           10 A2       072A                  SB$L_PBBL(R2)              ;
           51  52  D0  072C          MOVL    R2,R1                      ; Copy SB addr to R1
           52  6E  D0  072F          MOVL    (SP),R2                    ; Retreive dg addr
        53  04 AE  D0  0732          MOVL    4(SP),R3                   ;  and PB addr
        30 A3  51  D0  0736          MOVL    R1,PB$L_SBLINK(R3)         ; Link new SB to PB
           1A A2   90  073A          MOVB    PPD$B_PROTOCOL(R2),-       ; Save PPD protocol level in
           48 A3       073D                  PB$B_PROTOCOL(R3)          ;  formative PB
           7E  51  7D  073F          MOVQ    R1,-(SP)                   ; Save regs destroyed by movc
               2C  28  0742          MOVC3   #DATA_LEN,-                ; Copy system ID, dg and msg
           14 A2       0744                  PPD$B_SYSTEMID(R2),-       ;  sizes, sw type, version,
           18 A1       0746                  SB$B_SYSTEMID(R1)          ;  incarnation, HW type and version
        52  04 AE  D0  0748          MOVL    4(SP),R2                   ; Retreive START/STACK dg addr
           08  20  3A  074C          LOCC    #^A/ /,#8,-                ; Compute # characters prior
           40 A2       074F                  PPD$Q_NODENAME(R2)         ;  to blank fill
       50  08  50  C3  0751          SUBL3   R0,#8,R0                   ;  in node name
           51  8E  7D  0755          MOVQ    (SP)+,R1                   ; Retreive saved registers
        44 A1  50  90  0758          MOVB    R0,SB$T_NODENAME(R1)       ; Set count of characters
           5C A1   D4  075C          CLRL    SB$L_CSB(R1)               ; Zero link to newest CSB.
        40 A2  50  2C  075F          MOVC5   R0,PPD$Q_NODENAME(R2),-    ; Copy ASCII characters into
  45 A1    0F  00      0763                  #0,#15,SB$T_NODENAME+1(R1) ;  counted string node name in SB
               63  D4  0767          CLRL    (R3)                       ; Zero link to DDB chain for new SB
           50  01  3C  0769          MOVZWL  #SS$_NORMAL,R0             ; Set status = success
                       076C
                       076C  SB_DONE:
                       076C
               3C  BA  076C          POPR    #^M<R2,R3,R4,R5>           ; Restore registers
                   05  076E          RSB                                ; Return
                       076F
                       076F          .DSABL  LSB
```

```
                     076F         .SBTTL  -       BREAK_PATH,     INITIATE CRASH
                     076F         .SBTTL  -                       OF VIRTUAL CIRCUIT
                     076F         .SBTTL  -       BREAK_HOST,     HOST SHUTDOWN REC'D
                     076F
                     076F  ;+
                     076F  ; BREAK_PATH is the action routine called when a START is received
                     076F  ; on a VC we think is open.  The START implies that the remote system
                     076F  ; has crashed the VC and that we should do the same.  Therefore, the
                     076F  ; start datagram is discarded and ERR$CRASHVC is called to start
                     076F  ; the process of crashing the virtual circuit.
                     076F  ;
                     076F  ; BREAK_HOST is the action routine called when a host shutdown
                     076F  ; dg is received.  It does the same as BREAK_PATH, but saves
                     076F  ; a special reason code in the path block to be used later when
                     076F  ; notifying SYSAP's of the circuit failure.
                     076F  ;
                     076F  ; Inputs:
                     076F  ;
                     076F  ;        R2                            -Addr of START/Host shutdown dg
                     076F  ;        R3                            -Addr of PB
                     076F  ;        R4                            -Addr of PDT
                     076F  ;
                     076F  ; Outputs:
                     076F  ;
                     076F  ;        R0-R2                         -Destroyed
                     076F  ;        Other registers               -Preserved
                     076F  ;-
                     076F
                     076F         .ENABL  LSB
                     076F
                     076F  BREAK_HOST:
                     076F
  028C 8F     B0     076F         MOVW    #SS$_NOSUCHNODE,-           ; Save vc fail reason for
    46 A3            0773                 PB$W_VCFAIL_RSN(R3)         ;  later reporting to SYSAPs
                     0775                                             ;  as the aux status
                     0775  BREAK_PATH:
                     0775
     F888'    30     0775         BSBW    INT$INS_DFREEQ1             ; Return dg buffer to
                     0778                                             ;  free queue
  51   53     D0     0778         MOVL    R3,R1                       ; Transfer PB address
     F882'    31     077B         BRW     ERR$CRASHVC                 ; Start crash of VC on its way
                     077E
                     077E         .DSABL  LSB
```

```
                          077E        .SBTTL  -       REC_ERROR_DG,           LOG ERROR DG
                          077E
                          077E  ;+
                          077E  ; REC_ERROR_DG is the action routine called for an error log datagram
                          077E  ; PPD type.  These are datagrams received from hosts that have minimal
                          077E  ; error logging capability, do not have an scs connection over which
                          077E  ; to send an application datagram containing error info, and choose to
                          077E  ; send the info in one of these 'out of band' datagrams instead.
                          077E  ;
                          077E  ; Inputs:
                          077E  ;
                          077E  ;       R2                      -Address of start of dg
                          077E  ;       R3                      -Address of PB
                          077E  ;       R4                      -Address of PDT
                          077E  ;
                          077E  ; Outputs:
                          077E  ;
                          077E  ;       R0                      -Destroyed
                          077E  ;       Other registers         -Preserved
                          077E  ;-
                          077E
                          077E        .ENABL  LSB
                          077E
                          077E  REC_ERROR_DG:
                          077E
            F87F'  30     077E        BSBW    ELOG$ERROR_DG           ; Go log it
   50   00DC C4    D0     0781        MOVL    PDT$L_UCB0(R4),R0       ; Get UCB address
        0082 C0    B7     0786        DECW    UCB$W_ERRCNT(R0)        ; Decr error count incremented
                          078A                                        ;  by error logger
              00   11     078A        BRB     IGNORE_DG               ; Go recycle to dg free queue
                          078C
                          078C        .DSABL  LSB
              078C          .SBTTL  -       IGNORE_DG,                  DISCARD DATAGRAM WITHOUT ACTION
              078C
              078C    ;+
              078C    ; IGNORE_DG is the action routine called for received start handshake datagrams
              078C    ; for a path block with VC failure in progress.  The datagram is returned to
              078C    ; the free queue and no further action taken.
              078C    ;
              078C    ; Inputs:
              078C    ;
              078C    ;       R2                              -Addr of handshake dg
              078C    ;
              078C    ; Outputs:
              078C    ;
              078C    ;       R0                              -Destroyed
              078C    ;       Other registers                 -Preserved
              078C    ;-
              078C
              078C          .ENABL  LSB
              078C
              078C    IGNORE_DG:
              078C
  F871'   31  078C          BRW     INT$INS_DFREEQ1         ; Return dg to free queue
              078F
              078F          .DSABL  LSB
```

```
                             078F            .SBTTL  UTILITY ROUTINES
                             078F            .SBTTL  -       FMT_START_DATA, FORMAT START DATA IN A
                             078F            .SBTTL  -                       START/STACK DATAGRAM
                             078F
                             078F     ;+
                             078F     ; FMT_START_DATA fills in the start data in a STACK or START datagram.
                             078F     ; Data is drqwn from sysgen paramters, SCS global locations, the
                             078F     ; system ID register, and constants.
                             078F     ;
                             078F     ; Inputs:
                             078F     ;
                             078F     ;       R2                              -Addr of datagram
                             078F     ;       R3                              -Addr of PB
                             078F     ;       R4                              -Addr of PDT
                             078F     ;
                             078F     ; Outputs:
                             078F     ;
                             078F     ;       R0,R1                           -Destroyed
                             078F     ;       other registers                 -Preserved
                             078F     ;-
                             078F     ;
                             078F     ; Message format adjacency assumptions:
                             078F     ;
                             078F
                             078F            ASSUME  PPD$B_SYSTEMID+6 EQ PPD$B_PROTOCOL
                             078F            ASSUME  PPD$B_PROTOCOL+2 EQ PPD$W_MAXDG
                             078F            ASSUME  PPD$W_MAXDG+2   EQ PPD$W_MAXMSG
                             078F            ASSUME  PPD$W_MAXMSG+2  EQ PPD$T_SWTYPE
                             078F            ASSUME  PPD$T_SWTYPE+4  EQ PPD$T_SWVERS
                             078F            ASSUME  PPD$T_SWVERS+4  EQ PPD$Q_SWINCARN
                             078F            ASSUME  PPD$Q_SWINCARN+8 EQ PPD$T_HWTYPE
                             078F            ASSUME  PPD$T_HWTYPE+4  EQ PPD$B_HWVERS
                             078F            ASSUME  PPD$B_HWVERS+12 EQ PPD$Q_NODENAME
                             078F            ASSUME  PPD$Q_NODENAME+8 EQ PPD$Q_CURTIME
                             078F            ASSUME  PPD$Q_CURTIME+8 EQ PPD$C_MIN_DGSIZ
                             078F
                             078F            .ENABL  LSB
                             078F
                             078F     FMT_START_DATA:
                             078F
      50   14 A2    DE       078F            MOVAL   PPD$B_SYSTEMID(R2),R0    ; Get system ID field addr
80    00000000'GF   7D       0793            MOVQ    G^SCS$GB_SYSTEMID,(R0)+  ; Copy system ID
      FE A0    01   9B       079A            MOVZBW  #PPD$C_PRT_ELOG,-2(R0)   ; Set current protocol rev supported
80    00000000'GF   D0       079E            MOVL    G^SCS$GW_MAXDG,(R0)+     ; Specify max bytes of dg and
                             07A5                                             ;  msg application data
80    20534D56 8F   D0       07A5            MOVL    #^A/VMS /,(R0)+          ; Set operating system name
80    00000000'GF   D0       07AC            MOVL    G^SYS$GQ_VERSION,(R0)+   ; Set operating system version
      0000002C'GF   7D       07B3            MOVQ    G^SCS$GA_LOCALSB+ -
               80            07B9                    SB$Q_SWINCARN,(R0)+      ; Set system boot seq #
80    00000000'EF   D0       07BA            MOVL    INI$T_HWTYPE,(R0)+       ; Set processor name
80    00000000'GF   7D       07C1            MOVQ    G^EXE$GB_CPUDATA,(R0)+   ; Copy CPU data (hardware/ ucode
80    00000008'GF   D0       07C8            MOVL    G^EXE$GB_CPUDATA+8,(R0)+ ;   rev levels)
80    00000000'GF   7D       07CF            MOVQ    G^SCS$GB_NODENAME,(R0)+  ; Null node name, blank filled
80    00000000'GF   7D       07D6            MOVQ    G^EXE$GQ_SYSTIME,(R0)+   ; Set current system time
               05            07DD            RSB                              ; Return
                             07DE
                             07DE            .DSABL  LSB
```

```
                           07DE          .SBTTL  -       CLEANUP, REMOVE FORMATIVE PB AND SB
                           07DE
                           07DE  ;+
                           07DE  ; CLEANUP is called by the ACTION_DISP routine when fail status
                           07DE  ; has been returned by an action routine.  The action routine
                           07DE  ; detecting the error is expected to perform all cleanup other
                           07DE  ; than deleting the formative path block and system block.  CLEANUP
                           07DE  ; deletes the formative system block (if any) and formative
                           07DE  ; path block.  The start handshake is simply abandoned to be
                           07DE  ; restarted by a new IDREC later.
                           07DE  ;
                           07DE  ; Inputs:
                           07DE  ;
                           07DE  ;       R3                              -Addr of formative PB
                           07DE  ;       R4                              -Addr of PDT
                           07DE  ;
                           07DE  ; Outputs:
                           07DE  ;
                           07DE  ;       R0                              -Destroyed
                           07DE  ;       other registers                 -Preserved
                           07DE  ;-
                           07DE
                           07DE          .ENABL  LSB
                           07DE
                           07DE  CLEANUP:
                           07DE
       50    30 A3   D0    07DE          MOVL    PB$L_SBLINK(R3),R0       ; Get addr of formative SB
                02   13    07E2          BEQL    10$                      ; Branch if none
                11   10    07E4          BSBB    CLEAN2                   ; Else deallocate SB
                           07E6
            0C A3    E5    07E6  10$:    BBCC    PB$B_RSTATION(R3),-      ; Mark no PB in path map
       00 0114 C4          07E9                  PDT$B_PORTMAP(R4),20$    ;
          019A C4    B7    07ED  20$:    DECW    PDT$W_STDGUSED(R4)       ; Decr count of # ports likely
                           07F1                                           ;  to send IDREC's and need
                           07F1                                           ;  start handshake
              011D   30    07F1          BSBW    LB_ENABLE                ; Enable loopback dg's if necessary
       50       63   0F    07F4          REMQUE  (R3),R0                  ; Remove PB from formative list
                           07F7
   00000000'GF       16    07F7  CLEAN2: JSB     G^COM$DRVDEALMEM         ; Deallocate PB
                     05    07FD          RSB                              ; Return
                           07FE
                           07FE          .DSABL  LSB
```

```
                            07FE          .SBTTL  -       SEARCH_PATHS, SEARCH FOR PB WITH STATION ADDR MATCH
                            07FE
                            07FE     ;+
                            07FE     ; SEARCH_PATHS searches a doubly linked list of PB's for the first
                            07FE     ; PB with station address matching a specified station address.  The
                            07FE     ; match is done only on the low order 8 bits of station address since
                            07FE     ; CI station addresses are known to fit in 8 bits.
                            07FE     ;
                            07FE     ; Inputs:
                            07FE     ;
                            07FE     ;       R1                              -Station address to match
                            07FE     ;       R3                              -Addr of PB listhead
                            07FE     ;
                            07FE     ; Outputs:
                            07FE     ;
                            07FE     ;       R0                              -0/1 for fail/success on search
                            07FE     ;       R3                              -PB address if success
                            07FE     ;       other registers                 -Preserved
                            07FE     ;-
                            07FE
                            07FE          .ENABL  LSB
                            07FE
                            07FE    SEARCH_PATHS:
                            07FE
             50     53  D0  07FE          MOVL    R3,R0                           ; Hold start point
                            0801
                            0801    SEARCH_CONT:
                            0801
             53     63  D0  0801          MOVL    (R3),R3                         ; Get next PB
             50     53  D1  0804          CMPL    R3,R0                           ; Back at start?
                    0A  13  0807          BEQL    20$                             ; Branch if so
       51    0C     A3  91  0809          CMPB    PB$B_RSTATION(R3),R1            ; Low byte matches?
                    F2  12  080D          BNEQ    SEARCH_CONT                     ; Branch if not
             50     01  3C  080F          MOVZWL  #SS$_NORMAL,R0                  ; Else return success
                        05  0812          RSB                                     ; Return
                            0813
                    50  D4  0813    20$:  CLRL    R0                              ; Status = fail
                        05  0815          RSB                                     ; Return
                            0816
                            0816          .DSABL  LSB
```

```
                           0816         .SBTTL  -       CNF$LKP_PB_MSG, LOOK UP THE PB CORRESPONDING
                           0816         .SBTTL  -                       TO A PDT AND REMOTE STATION ADDR
                           0816
                           0816  ;+
                           0816  ; CNF$LKP_PB_MSG extracts the remote station addr from a received message
                           0816  ; and looks through the system wide configuration database for the
                           0816  ; PB correxponding to the remote station and PDT.  Only the low order
                           0816  ; 8 bits of the station address are matched since CI station addresses
                           0816  ; always fit in 8 bits.
                           0816  ;
                           0816  ; Inputs:
                           0816  ;
                           0816  ;       R2                      -Addr of message
                           0816  ;       R4                      -Addr of PDT
                           0816  ;
                           0816  ; Outputs:
                           0816  ;
                           0816  ;       R0                      -0/1 for fail/success on search
                           0816  ;       R1                      -PB addr if success
                           0816  ;       Other registers         -Preserved
                           0816  ;-
                           0816
                           0816         .ENABL  LSB
                           0816
                           0816  CNF$LKP_PB_MSG2::
                           0816
  51   52   00B4 C4    C3  0816         SUBL3   PDT$L_MSGHDRSZ(R4),R2,R1; Back up to top of PPD layer
       51     0C A1    9A  081C         MOVZBL  PPD$B_PORT(R1),R1       ; Get remote station addr
                 04    11  0820         BRB     5$
                           0822
                           0822  CNF$LKP_PB_MSG::
                           0822
       51     0C A2    9A  0822         MOVZBL  PPD$B_PORT(R2),R1       ; Get remote station addr
                           0826
                 55    DD  0826  5$:    PUSHL   R5                      ; Save a couple of registers
                 53    DD  0828         PUSHL   R3                      ;
  55     00000000'GF   DE  082A         MOVAL   G^SCS$GQ_CONFIG,R5      ; Get addr of listhead for system
                           0831                                         ;  configuration database
                           0831
            55   65    D0  0831  10$:   MOVL    (R5),R5                 ; Get next system block
    00000000'8F  55    D1  0834         CMPL    R5,#SCS$GQ_CONFIG       ; Back at start of list?
                 21    13  083B         BEQL    PB_NOT_FOUND            ; Branch if so
       53     0C A5    DE  083D         MOVAL   SB$L_PBFL(R5),R3        ; Get addr of PB listhead
                 BB    10  0841         BSBB    SEARCH_PATHS            ; See if there is matching station
                           0843
              EB 50    E9  0843  20$:   BLBC    R0,10$                  ; Branch if no matching station
          54  2C A3    D1  0846         CMPL    PB$L_PDT(R3),R4         ; Is this path block a path from
                           084A                                         ;  the same PDT?
                 08    13  084A         BEQL    PB_FOUND                ; Branch if yes
       50     0C A5    DE  084C         MOVAL   SB$L_PBFL(R5),R0        ; Else set up PB listhead addr again
                 AF    10  0850         BSBB    SEARCH_CONT             ; Continue PB search
                 EF    11  0852         BRB     20$                     ;  and check results
                           0854
                           0854  PB_FOUND:
                           0854
            51   53    D0  0854         MOVL    R3,R1                   ; Move PB addr to R1
                           0857
```

```
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            .SBTTL  -       CNF$LKP_PB_PDT, LOOK UP FIRST/NEXT
                              0862            .SBTTL  -                       PB ASSOC WITH PDT
                              0862
                              0862   ;+
                              0862   ; CNF$LKP_PB_PDT looks through the configuration database for PB's
                              0862   ; associated with a specified PDT.  For each one found, the caller is
                              0862   ; called back with the PB address in R3.  When the whole database has
                              0862   ; been searched, return is taken to the caller with failure status in R0.
                              0862   ;
                              0862   ; This routine is called during power failure to cleanup PB's and SB's
                              0862   ; associated with the local failing port.  Therefore, when a PB is
                              0862   ; delivered to the caller, the PB and its SB may have been deleted
                              0862   ; upon return from the coroutine.  The forward links to the next PB and
                              0862   ; next SB in the configuration database will be destroyed in this case.
                              0862   ; Whenever an SB is being processed, the link to the next SB is saved on
                              0862   ; the stack.  When a PB is about to be delivered to the coroutine, the
                              0862   ; link to the next PB is saved on the stack and, upon return, the saved
                              0862   ; link used as the address of the next PB to look at.
                              0862   ;
                              0862   ; Inputs:
                              0862   ;
                              0862   ;        R4                              -PDT addr
                              0862   ;
                              0862   ; Outputs:
                              0862   ;
                              0862   ;        R0                              -Status:  LBS/C if PB found/not found
                              0862   ;        R3                              -PB addr if success
                              0862   ;        R1,R2                           -Destroyed
                              0862   ;        Other registers                 -Preserved
                              0862   ;-
                              0862
                              0862   ASSUME  PB$L_FLINK      EQ 0
                              0862   ASSUME  SB$L_FLINK      EQ 0
                              0862
                              0862            .ENABL  LSB
                              0862
                              0862   CNF$LKP_PB_PDT::
                              0862
 52   00000000'GF       DE    0862            MOVAL   G^SCS$GQ_CONFIG,R2         ; Get configuration database ptr
             52  62     D0    0869            MOVL    (R2),R2                    ; Get next system blk
                              086C
00000000'8F      52     D1    086C   10$:     CMPL    R2,#SCS$GQ_CONFIG          ; Back at header?
                 2D     13    0873            BEQL    NOT_FOUND                  ; Branch if so
                 62     DD    0875            PUSHL   (R2)                       ; Save link to next SB
         53   0C A2     DE    0877            MOVAL   SB$L_PBFL(R2),R3           ; Get PB list header
             51  53     D0    087B            MOVL    R3,R1                      ; Save listhead addr
                              087E
             53  63     D0    087E   20$:     MOVL    (R3),R3                    ; Get next PB
                              0881
             51  53     D1    0881   30$:     CMPL    R3,R1                      ; Back at start of list?
                 17     13    0884            BEQL    NEXT_SB                    ; Branch if so -- move to next SB
         54   2C A3     D1    0886            CMPL    PB$L_PDT(R3),R4            ; Is PB on this PDT?
                 F2     12    088A            BNEQ    20$                        ; Branch if not
             50  01     3C    088C            MOVZWL  #SS$_NORMAL,R0             ; Set success status for caller
                              088F                                               ;  coroutine
                 63     DD    088F            PUSHL   (R3)                       ; Save link to next PB
                 06     BB    0891            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          .SBTTL  -       CNF$REMOVE_PB,  REMOVE PB(SB) FROM
                     08A5          .SBTTL  -                       CONFIG DATABASE
                     08A5
                     08A5  ;+
                     08A5  ; CNF$REMOVE_PB is called by ERR$VCCLOSED_MSG/PB or ERR$VC_CACHECLR
                     08A5  ; when all connections associated with a failing path block have
                     08A5  ; been cleaned up.  CNF$REMOVE_PB marks the remote port as unknown in
                     08A5  ; the port bitmap.  If this is a virutal circuit failure  due to reasons
                     08A5  ; other than local port/system power failure, then the path block SCS
                     08A5  ; receive buffer and, if available, the SCS send buffer, are reclaimed from
                     08A5  ; the message free queue and returned to pool.  In the case of a power
                     08A5  ; failure this step is omitted because all queue elements for all
                     08A5  ; paths on the local port are collected together later.
                     08A5  ;
                     08A5  ; Finally, the path block is unlinked from the system block.  If this
                     08A5  ; leaves the SB with no paths, then the SB link to the next PB to
                     08A5  ; use in a connection is zeroed.  The PB is returned to pool and return taken.
                     08A5  ;
                     08A5  ; Inputs:
                     08A5  ;
                     08A5  ;        IPL                             -Fork IPL
                     08A5  ;
                     08A5  ;        R3                              -PB addr
                     08A5  ;        R4                              -PDT addr
                     08A5  ;
                     08A5  ; Outputs:
                     08A5  ;
                     08A5  ;        R0-R2                           -Destroyed
                     08A5  ;        Other registers                 -Preserved
                     08A5  ;-
                     08A5
                     08A5          .ENABL  LSB
                     08A5
                     08A5  CNF$REMOVE_PB::
                     08A5
      34 A3    D5    08A5          TSTL    PB$L_CDTLST(R3)         ; Verify no CDT's remain
         03    13    08A8          BEQL    10$                     ; Branch if none do
       FA18    31    08AA          BRW     CONFIG_ERR              ; Else inconsistent database
                     08AD
      0C A3    E5    08AD  10$:    BBCC    PB$B_RSTATION(R3),-     ; Mark the remote port unknown
 00 0114 C4          08B0                  PDT$B_PORTMAP(R4),20$   ;  to poller
                     08B4
    019A C4    B7    08B4  20$:    DECW    PDT$W_STDGUSED(R4)      ; Decr #ports that will likely
                     08B8                                          ;  send us IDREC's for a while
       0056    30    08B8          BSBW    LB_ENABLE               ; Enable loopback dg's if necessary
    0112 C4    B7    08BB          DECW    PDT$W_PBCOUNT(R4)       ; Decr count of PB's on this PDT
      12 A3    B1    08BF          CMPW    PB$W_STATE(R3),-        ; Is this a power fail recovery?
    4000 8F          08C2                  #PB$C_PWR_FAIL          ;
         13    13    08C5          BEQL    40$                     ; Branch if so
 52   40 A3    D0    08C7          MOVL    PB$L_SCSMSG(R3),R2      ; Else get SCS send buffer
         07    12    08CB          BNEQ    30$                     ; Branch if available
      F730'    30    08CD          BSBW    INT$MFQ2POOL            ; If unavailable, get it from
         08    1D    08D0          BVS     40$                     ;  message free queue
         03    11    08D2          BRB     35$
                     08D4
      F729'    30    08D4  30$:    BSBW    INT$DEAL_MSG            ; Deallocate to pool
      F726'    30    08D7  35$:    BSBW    INT$MFQ2POOL            ; Get SCS receive buffer from free q
```

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

```
                 08FF          .SBTTL  -       SNDDG_RET,      SEND DG, RETURN BUFFER
                 08FF          .SBTTL  -                       TO RESPONSE QUEUE
                 08FF          .SBTTL  -       SNDDG_NORET,    SEND DG, RETURN BUFFER
                 08FF          .SBTTL  -                       TO FREE QUEUE
                 08FF
                 08FF     ;+
                 08FF     ; The datagram is put on the low priority command queue with
                 08FF     ; the response flag set/clear for the SEND_RET/NORET call.
                 08FF     ;
                 08FF     ; Inputs:
                 08FF     ;
                 08FF     ;        R2                              -Addr of dg buffer
                 08FF     ;        R3                              -Addr of PB
                 08FF     ;        R4                              -Addr of PDT
                 08FF     ;
                 08FF     ; Outputs:
                 08FF     ;
                 08FF     ;        R0                              -Destroyed
                 08FF     ;        Other registers                 -Preserved
                 08FF     ;-
                 08FF
                 08FF          .ENABL  LSB
                 08FF
                 08FF     SNDDG_RET:
                 08FF
51    53   D0    08FF          MOVL    R3,R1                   ; Transfer PB address
50    02   D0    0902          MOVL    #SYSAP$C_DISPPO,R0      ; RETFLAG=TRUE, DISP=POOL
    F6F8'  31    0905          BRW     INT$SNDDG1              ; Send it
                 0908
                 0908     SNDDG_NORET:
                 0908
51    53   D0    0908          MOVL    R3,R1                   ; Transfer PB address
50    00   D0    090B          MOVL    #SYSAP$C_DISPQ,R0       ; RETFLAG=FALSE
    F6EF'  31    090E          BRW     INT$SNDDG1              ; Send it
                 0911
                 0911          .DSABL  LSB
```

```
                         0911         .SBTTL  -       LB_ENABLE,      ENABLE LB DG SENDS
                         0911         .SBTTL  -                       IF NECESSARY
                         0911
                         0911 ;+
                         0911 ; Called whenever a virtual circuit is lost to check and see if
                         0911 ; there are now no remote ports known besides self.  (Known means
                         0911 ; virtual circuits open or formative paths.) If there are no remote
                         0911 ; ports known besides self, then the loopback dg test is enabled.
                         0911 ; Otherwise, the loopback test flag is left alone.
                         0911 ;
                         0911 ; Inputs:
                         0911 ;
                         0911 ;       R4                      -PDT addr
                         0911 ;       PDT$B_PORTMAP(R4)       -32 byte bit map of known ports
                         0911 ;       PDT$B_PORT_NUM(R4)      -# of local port
                         0911 ;
                         0911 ; Outputs:
                         0911 ;
                         0911 ;       R0                      -Destroyed
                         0911 ;       Other registers         -Preserved
                         0911 ;       PDT$W_LPORT_STS         -PDT$M_LBDG set if no other
                         0911 ;                                ports known; else unchanged
                         0911 ;-
                         0911
                         0911         .ENABL  LSB
                         0911
                         0911 LB_ENABLE:
                         0911
         7E   51   7D    0911         MOVQ    R1,-(SP)                      ; Save two registers for caller
              52   D4    0914         CLRL    R2                            ; Zero count of # bytes in map
                         0916
         51   01   CE    0916 10$:    MNEGL   #1,R1                         ; Init prev known port #, modulo 32
                         0919
              51   D6    0919 20$:    INCL    R1                            ; Incr prev known port #, mod 32
         20   51   EA    091B         FFS     R1,#32,-                      ; Find next known port, mod 32
   51   0114 C442        091E                 PDT$B_PORTMAP(R4)[R2],R1 ;  in this longwd of port map
              10   13    0923         BEQL    40$                           ; Branch if none found
    50   52   08   78    0925         ASHL    #32/4,R2,R0                   ; Convert port # mod 32 to
         51   50   C0    0929         ADDL    R0,R1                         ;   actual port number
   017D C4   51   91     092C         CMPB    R1,PDT$B_PORT_NUM(R4)         ; Is known port = self?
              E6   13    0931         BEQL    20$                           ; Branch if so to search more
              0D   11    0933         BRB     50$                           ; Else return without doing anything
                         0935
         52   04   C0    0935 40$:    ADDL    #4,R2                         ; Step offset in port map to next longwd
         20   52   D1    0938         CMPL    R2,#32                        ; Past last longwd in map?
              D9   1F    093B         BLSSU   10$                           ; Branch if not
              04   A8    093D         BISW    #PDT$M_LBDG,-                 ; Else no port other than
        0110 C4          093F                 PDT$W_LPORT_STS(R4)           ;  self known, so enable LB dgs
                         0942
         51   8E   7D    0942 50$:    MOVQ    (SP)+,R1                      ; Restore caller's registers
                   05    0945         RSB                                   ; Return
                         0946
                         0946         .DSABL  LSB
```

```
                     0946          .SBTTL  -        CHECK_PORT_REV,         CHECK PORT
                     0946          .SBTTL  -                                UCODE REV LEVEL
                     0946
                     0946     ;+
                     0946     ; Given and IDREC packet, check the port RAM and ROM rev levels
                     0946     ; to make sure they are adequate.  If not, log an error, print a
                     0946     ; message on OPA0, and (for now) continue.
                     0946     ;
                     0946     ; The algorithm for checking is to look up the ROM/RAM level read
                     0946     ; from the ID in a table of legal ROM/REAM combinations.  If it isn't
                     0946     ; in the table, then check to see if either the ROM or RAM level
                     0946     ; exceeds the maximum the table knows about.  If either exceeds
                     0946     ; the maximum in the table.  If either exceeds the max, do no
                     0946     ; futher checking on the assumption that new ucode is being run
                     0946     ; that VMS hasn't been taught to judge.  If neither exceeds the
                     0946     ; max, then the ucode fails the test.
                     0946     ;
                     0946     ; If the rev level is found in the legal table, then check the
                     0946     ; cautionary rev table to see if we should print a warning before
                     0946     ; continuing.  A flag is set in the cautionary table for rev's
                     0946     ; which are known to have problems, but which have not yet been
                     0946     ; replaced by the fixed ucode in the field yet.  The cautionary
                     0946     ; message on OPA0 alerts customers to ask field service to install
                     0946     ; fixes.
                     0946     ;
                     0946     ; To add new legal rev combinations to the table, patch or extend
                     0946     ; LEGAL_REV_TABLE with the new legal combination(s), and patch
                     0946     ; MAX_RAM/ROM_REV.
                     0946     ;
                     0946     ; Inputs:
                     0946     ;
                     0946     ;       R2                      -Addr of IDREC packet
                     0946     ;       R4                      -PDT addr
                     0946     ;
                     0946     ; Outputs:
                     0946     ;
                     0946     ;       R0                      -Destroyed
                     0946     ;       Other registers         -Preserved
                     0946     ;-
                     0946
                     0946     LEGAL_REV_TABLE:
                     0946
                     0946                                               ; .WORD n,n = RAM/ROM level
                     0946
          0002 0002  0946          .WORD   2,2                          ; Current as of June, 1984
          0003 0003  094A          .WORD   3,3                          ; Next rev known to need fixes
                     094E                                               ;  in both RAM and ROM
          0000 0000  094E          .WORD   0,0                          ; Patch space for future revs
          0000 0000  0952          .WORD   0,0
                     0956
           00000004  0956     REV_TABLE_SIZ = <.- LEGAL_REV_TABLE>/4
                     0956
                     0956     CAUTION_REV:
                     0956
                     0956                                               ; .BYTE nonzero/0 for caution/
                     0956                                               ;  caution message needed
                 00  0956          .BYTE   0                            ; Rev 2,2 -- no caution
```

```
                    00  0957          .BYTE   0                            ; Rev 3,3 -- no caution
                    00  0958          .BYTE   0                            ; Future revs...
                    00  0959          .BYTE   0
                        095A
                        095A  MAX_RAM_REV:
                        095A
                  0003  095A          .WORD   3                            ; Max RAM level in table
                        095C
                        095C  MAX_ROM_REV:
                        095C
                  0003  095C          .WORD   3                            ; Max ROM level in table
                        095E
                        095E          .ENABL  LSB
                        095E
                        095E  CHECK_PORT_REV:
                        095E
              31    BB  095E          PUSHR   #^M<R0,R4,R5>                ; Save caller's registers
    55   00DC C4    D0  0960          MOVL    PDT$L_UCB0(R4),R5            ; Get UCB in case error logging needed
      51    DE AF   DE  0965          MOVAL   LEGAL_REV_TABLE,R1           ; Get addr of legal rev table
              50    D4  0969          CLRL    R0                           ; Zero index into table
                        096B
   1C A2      81    D1  096B  10$:    CMPL    (R1)+,PPD$L_RPORT_REV(R2)    ; Is rev being checked in table?
              22    13  096F          BEQL    CHECK_CAUTION                ; Branch if so
    F6 50     04    F2  0971          AOBLSS  #REV_TABLE_SIZ,R0,10$        ; Branch if not, continue check
      1E A2         B1  0975          CMPW    PPD$L_RPORT_REV+2(R2),-      ; Is RAM level bigger than we know about?
      E0 AF             0978                  MAX_RAM_REV
              23    1A  097A          BGTRU   REV_OK                       ; Branch if so
      1C A2         B1  097C          CMPW    PPD$L_RPORT_REV(R2),-        ; Is ROM level bigger than we know about?
      DB AF             097F                  MAX_ROM_REV
              1C    1A  0981          BGTRU   REV_OK                       ; Branch if so
           F67A'    30  0983          BSBW    ELOG$UCODE_ERR               ; Log problem
     00000000'EF    94  0986          CLRB    INI$PORT_REV                 ; Clear port rev okay flag to force
                        098C                                               ;  more informative UCODEREV bugcheck
                        098C                                               ;  if a bugcheck is done
         0080 C5    94  098C          CLRB    UCB$B_ERTCNT(R5)             ; Take away all port's retries
           F66D'    30  0990          BSBW    ERR$CRASH_PORT               ; Go crash port permanently
                        0993
                        0993  CHECK_CAUTION:
                        0993
      51    C0 AF   DE  0993          MOVAL   CAUTION_REV,R1               ; Get addr of table of caution flags
            6140    95  0997          TSTB    (R1)[R0]                     ; Rev legal, check if caution msg needed
              03    13  099A          BEQL    REV_OK                       ; Branch if completely okay
           F661'    30  099C          BSBW    ELOG$UCODE_WARN              ; Log warning
                        099F
                        099F  REV_OK:
                        099F
              32    BA  099F          POPR    #^M<R1,R4,R5>                ; Restore caller's registers
                    05  09A1          RSB                                  ; Return.
                        09A2
                        09A2          .DSABL  LSB
```

```
                          09A2          .SBTTL  CNF$TIMER,       PERIODIC WAKEUP ROUTINE
                          09A2          .SBTTL  CNF$CALCINTDUE, RESET WAKEUP DUE TIME
                          09A2
                          09A2  ;+
                          09A2  ; CNF$TIMER is called from exec module TIMESCHDL once per n
                          09A2  ; seconds, where n is the basic CI interval timeout.  Timer
                          09A2  ; intervals are specifed in SYSGEN as follows:
                          09A2  ;
                          09A2  ;  Parameter name                 Units                        Variable name
                          09A2  ;
                          09A2  ;         PASIMTOUT        seconds (2, 2^15-1)      SCS$GW_PASTMOUT
                          09A2  ;         PAPOLLINTERVAL  seconds (2, 2^15-1)      SCS$GW_PAPOLINT
                          09A2  ;         PAPOOL_INTERVAL seconds (2, 2^15-1)      SCS$GW_PAPOOLIN
                          09A2  ;
                          09A2  ; Note that if the poller interval and pool checking interval are not
                          09A2  ; exact multiples of the basic interval, then they will be effecitvely
                          09A2  ; rounded up to the nearest multiple of the basic interval.  The basic
                          09A2  ; interval is equal to the start handshake timeout interval.
                          09A2  ;
                          09A2  ; Inputs:
                          09A2  ;
                          09A2  ;         R3                              -Addr of CRB
                          09A2  ;         IPL                             -IPL$_POWER
                          09A2  ;
                          09A2  ; Outputs:
                          09A2  ;
                          09A2  ;         IPL                             -IPL$_SCS
                          09A2  ;         R0-R2,R4,R5                     -Destroyed
                          09A2  ;         Other registers                 -Preserved
                          09A2  ;
                          09A2  ; Entry CNF$CALCINTDUE computes the due time for the next basic interval wakeup.
                          09A2  ; It expects as inputs R3/CRB, R4/PDT and destroys R0.
                          09A2  ;
                          09A2  ;-
                          09A2
                          09A2          .ENABL  LSB
                          09A2
                          09A2  CNF$TIMER::
                          09A2
    54    10 A3     D0    09A2          MOVL    CRB$L_AUXSTRUC(R3),R4   ; Get PDT address
               01   12    09A6          BNEQ    5$                      ; Branch if there is a PDT
                    05    09A8          RSB                             ; Else port init aborted, can't
                          09A9                                          ;  use port
                          09A9
    55  00DC C4     D0    09A9  5$:     MOVL    PDT$L_UCB0(R4),R5       ; Get UCB address
               04   E0    09AE          BBS     #UCB$V_ONLINE,-         ; Branch if controller/unit is
         03 64 A5         09B0                  UCB$W_STS(R5),CONT_POLL ;  on line
             00A7   31    09B3          BRW     CNF$CALCINTDUE          ; Else bypass poller and other activity
                          09B6                                          ;  and compute next wakeup time
                          09B6
                          09B6  CONT_POLL:
                          09B6
  0104 D4      01   D0    09B6          MOVL    #1,@PDT$L_MTC(R4)       ; Poke the maint timer in the
                          09BB                                          ;  port to tell it we are alive
                          09BB          SETIPL  #IPL$_SCS               ; Lower IPL for rest of polling, etc.
               53   DD    09BE          PUSHL   R3                      ; Save CRB address
    53  0174 C4     DE    09C0          MOVAL   PDT$Q_FORMPB(R4),R3     ; Get formative PB listhead addr
```

```
                  53   DD  09C5                PUSHL   R3                          ;  and save a copy
            53    63   D0  09C7                MOVL    (R3),R3                     ; Get addr of 1st entry in PB list
                           09CA
                           09CA   SCAN_FORMPB:
                           09CA
            6E    53   D1  09CA                CMPL    R3,(SP)                     ; Back at start of list?
                  1F   13  09CD                BEQL    FORM_PB_DONE                ; Branch if so
            55    63   D0  09CF                MOVL    (R3),R5                     ; Save addr of next PB in
                           09D2                                                    ;  case this one gets deleted
                  00   E1  09D2                BBC     #PB$V_TIM,-                 ; Branch if no timeout
             12 44 A3      09D4                        PB$W_STS(R3),10$            ;  is in progress
               3C A3   D1  09D7                CMPL    PB$L_DUETIME(R3),-          ; Passed this PB's duetime?
          00000000'GF      09DA                        G^EXE$GL_ABSTIM             ;
                  08   1A  09DF                BGTRU   10$                         ; Branch if not
      51    8001 8F    3C  09E1                MOVZWL  #EV$C_TIMEOUT,R1            ; Set event = timed out
                FADA   30  09E6                BSBW    ACTION_DISP                 ; Call action dispatcher for
                           09E9                                                    ;  this PB
                           09E9
            53    55   D0  09E9   10$:         MOVL    R5,R3                       ; Step to next formative  PB
                  DC   11  09EC                BRB     SCAN_FORMPB                 ; Check next PB
                           09EE
                           09EE   FORM_PB_DONE:
                           09EE
                  8E   D5  09EE                TSTL    (SP)+                       ; Clear PB listhd from stack
             0188 C4   D1  09F0                CMPL    PDT$L_POOLDUE(R4),-         ; Passed pool chekcer's time?
          00000000'GF      09F4                        G^EXE$GL_ABSTIM             ;
                  3D   1A  09F9                BGTRU   CHECK_POLLER                ; Branch if not
      55    00B0 C4    DE  09FB                MOVAL   PDT$L_WAITQBL(R4),R5        ; Get pool waiter listhead addr
         FC A5    65   D1  0A00                CMPL    (R5),-4(R5)                 ; List empty?
                  21   13  0A04                BEQL    POOL_DONE                   ; Branch if so
            55    65   D0  0A06                MOVL    (R5),R5                     ; Else get addr of last waiter (if any)
                           0A09
      53    00AC C4    D0  0A09   20$:         MOVL    PDT$L_WAITQFL(R4),R3        ; Get addr of next CDRP we are
                           0A0E                                                    ;  going to try to wake
                           0A0E                $RESUME_FP      -                   ; Resume next waiter
                           0A0E                        @PDT$L_WAITQFL(R4),-        ;
                           0A0E                        QEMPTY=POOL_DONE            ; if none, go to POOL_DONE
            55    53   D1  0A22                CMPL    R3,R5                       ; Was this waiter the last one when
                           0A25                                                    ;  we started scanning the list?
                           0A25                                                    ;  (More on the list now are
                           0A25                                                    ;  repeat failures.)
                  E2   12  0A25                BNEQ    20$                         ; Branch if not
                           0A27
                           0A27   POOL_DONE:
                           0A27
  50    00000000'GF    3C  0A27                MOVZWL  G^SCS$GW_PAPOOLIN,R0        ; Get pool check interval
00000000'GF       50   C1  0A2E                ADDL3   R0,G^EXE$GL_ABSTIM,-        ; Add pool interval to current
             0188 C4       0A35                        PDT$L_POOLDUE(R4)           ;  time and store as due time
                           0A38
                           0A38   CHECK_POLLER:
                           0A38
               53 8ED0     0A38                POPL    R3                          ; Retreive CRB addr
             018C C4   D1  0A3B                CMPL    PDT$L_POLLERDUE(R4),-       ; Passed poller's duetime?
          00000000'GF      0A3F                        G^EXE$GL_ABSTIM             ;
                  17   1A  0A44                BGTRU   CNF$CALCINTDUE              ; Branch if not
                F5B7   30  0A46                BSBW    CNF$POLL                    ; Call poller
                           0A49
      50    00000000'GF   3C  0A49            MOVZWL  G^SCS$GW_PAPOLINT,R0     ; Get poller interval
      00000000'GF    50   C1  0A50            ADDL3   R0,G^EXE$GL_ABSTIM,-     ; Add poll interval to current time and
                 018C C4      0A57                    PDT$L_POLLERDUE(R4)      ;  store as poller duetime
                   0011   30  0A5A            BSBW    CNF$CALC_POLLSW          ; Compute current time it takes
                              0A5D                                             ;  to do a complete poll sweep
                              0A5D                                             ;  over both paths -- this has
                              0A5D                                             ;  to be recomputed periodically because
                              0A5D                                             ;  the parameters are dynamic
                              0A5D
                              0A5D  CNF$CALCINTDUE::
                              0A5D
      50    00000000'GF   3C  0A5D            MOVZWL  G^SCS$GW_PASTMOUT,R0     ; Get basic timer interval
      00000000'GF    50   C1  0A64            ADDL3   R0,G^EXE$GL_ABSTIM,-     ; Add it to current time and
                   18 A3      0A6B                    CRB$L_DUETIME(R3)        ;  and save in CRB
                              0A6D
                          05  0A6D  30$:      RSB                              ; Return
                              0A6E
                              0A6E            .DSABL  LSB
                          0A6E          .SBTTL  CNF$CALC_POLLSW, CALCULATE TIME TO POLL
                          0A6E          .SBTTL  -                       PORT AT LEAST ONCE
                          0A6E
                          0A6E  ;+
                          0A6E  ; This routine computes the number of seconds it takes to poll
                          0A6E  ; every possible port at least once, even if only one path is
                          0A6E  ; working.  This value is used by the VAXcluster sysap.
                          0A6E  ;
                          0A6E  ; The formula is as follows:
                          0A6E  ;
                          0A6E  ;  {(maximum port # +1)/(# ports polled per interval)} * 2 paths * poll interval
                          0A6E  ;    +maximum time to wake up poller
                          0A6E  ;
                          0A6E  ; If the number of ports polled per interval exceeds the number of free
                          0A6E  ; datagrams available to conduct simultaneous start handshakes, then use
                          0A6E  ; the number of free datagrams instead of the number of ports per interval
                          0A6E  ; in the above formula.  The number of free datagrams available is not known
                          0A6E  ; exactly since there is no accounting on the datagrams that can be tied
                          0A6E  ; up doing start handshakes.  The number available is estimated as
                          0A6E  ; PDT$W_STDGDYN(R4).
                          0A6E  ;
                          0A6E  ; Inputs:
                          0A6E  ;
                          0A6E  ;       R4                              -PDT address
                          0A6E  ;
                          0A6E  ;       SCS$GB_PAMXPORT                 -SYSGEN'ed maximum port #
                          0A6E  ;       SCS$GB_PANPOLL                  -# ports to poll per interval
                          0A6E  ;       SCS$GW_PAPOLINT                 -# seconds between polls, poll interval
                          0A6E  ;       SCS$GW_PASTIMOUT                -# seconds it might take to wake up poller
                          0A6E  ;       PDT$B_MAX_PORT(R4)              -maximum port # supported by this CI
                          0A6E  ;
                          0A6E  ; Outputs:
                          0A6E  ;
                          0A6E  ;       R0,R1,R2                        -Destroyed
                          0A6E  ;       Other registers                 -Preserved
                          0A6E  ;
                          0A6E  ;       PDT$L_POLLSWEEP(R4)             -# seconds to poll each port at least once
                          0A6E  ;-
                          0A6E
                          0A6E          .ENABL  LSB
                          0A6E
                          0A6E  CNF$CALC_POLLSW::
                          0A6E
51   00000000'GF      9A  0A6E          MOVZBL  G^SCS$GB_PAMXPORT,R1    ; Get SYSGENed max port #
   50     017C C4     9A  0A75          MOVZBL  PDT$B_MAX_PORT(R4),R0   ; Get hardware supported max port
        50     51     D1  0A7A          CMPL    R1,R0                   ; SYSGENed .GT. hardware max?
               03     15  0A7D          BLEQ    10$                     ; Branch if not
        51     50     D0  0A7F          MOVL    R0,R1                   ; Else hardware value prevails
                          0A82
               51     D6  0A82  10$:    INCL    R1                      ; Convert port # to number of ports
50   00000000'GF      9A  0A84          MOVZBL  G^SCS$GB_PANPOLL,R0     ; Get # ports polled per interval
   52     0198 C4     3C  0A8B          MOVZWL  PDT$W_STDGDYN(R4),R2    ; Get # dgs available for start
                          0A90                                          ;  start handshakes, max.
        52     50     D1  0A90          CMPL    R0,R2                   ; # ports per interval .leq. free dg
                          0A93                                          ;  limit?
               03     1B  0A93          BLEQU   15$                     ; Branch if so
        50     52     D0  0A95          MOVL    R2,R0                   ; Else use free dg limit instead
```

```
                                  0A98
                      52    D4    0A98  15$:    CLRL    R2                              ; Clear h.o. longwd of dividend
       50    51   51  50    7B    0A9A          EDIV    R0,R1,R1,R0                     ; Compute # ports/ # per interval polled
                      50    D5    0A9F          TSTL    R0                              ; If there was a remainder,
                      02    13    0AA1          BEQL    20$                             ;
                      51    D6    0AA3          INCL    R1                              ;  then round quotient up
                                  0AA5
                 51   51    C0    0AA5  20$:    ADDL    R1,R1                           ; Multipy by 2 paths *
       50    00000000'GF    3C    0AA8          MOVZWL  G^SCS$GW_PAPOLINT,R0            ;  the number of seconds between
                 51   50    C4    0AAF          MULL    R0,R1                           ;  polls
       50    00000000'GF    3C    0AB2          MOVZWL  G^SCS$GW_PASTMOUT,R0            ; Get the timer before poller even
                                  0AB9                                                  ;  awakened,
       00D8 C4   51   50    C1    0AB9          ADDL3   R0,R1,PDT$L_POLLSWEEP(R4)       ;   add in and save total in PDT
                            05    0ABF          RSB                                     ; Return
                                  0AC0
                                  0AC0          .DSABL  LSB
                            0AC0                .SBTTL  START_TIMER, START A PATH BLOCK TIMER
                            0AC0        ;+
                            0AC0        ; START_TIMER computes the due time for PB timeout and sets the
                            0AC0        ; timeout in porgress bit (PB$V_TIM in PB$W_STS) for the specified
                            0AC0        ; pathblock.
                            0AC0        ;
                            0AC0        ; Inputs:
                            0AC0        ;
                            0AC0        ;       R3                              -Addr of PB
                            0AC0        ;
                            0AC0        ; Outputs:
                            0AC0        ;
                            0AC0        ;       R0                              -Destroyed
                            0AC0        ;       Other registers                 -Preserved
                            0AC0        ;-
                            0AC0
                            0AC0                .ENABL  LSB
                            0AC0
                            0AC0        START_TIMER:
                            0AC0
50   00000000'GF     3C     0AC0                MOVZWL  G^SCS$GW_PASTMOUT,R0    ; Get basic timer interval
00000000'GF     50   C1     0AC7                ADDL3   R0,G^EXE$GL_ABSTIM,-    ; Add it to the current time
              3C A3         0ACE                        PB$L_DUETIME(R3)        ;  and save in PB due time
                 00  E2     0AD0                BBSS    #PB$V_TIM,-             ; Set timeout in progress
           00 44 A3         0AD2                        PB$W_STS(R3),10$        ;  in pathblock
                     05     0AD5        10$:    RSB                             ; Return
                            0AD6
                            0AD6                .DSABL  LSB
```

```
                       0AD6             .SBTTL  STOP_TIMER, STOP PATH BLOCK TIMER
                       0AD6
                       0AD6      ;+
                       0AD6      ; STOP_TIMER disables path block timeout by clearing the timeout
                       0AD6      ; in progress bit in the pathblock.
                       0AD6      ;
                       0AD6      ; Inputs:
                       0AD6      ;
                       0AD6      ;       R3                      -Addr of PB
                       0AD6      ;
                       0AD6      ; Outputs:
                       0AD6      ;
                       0AD6      ;       All registers           -Preserved
                       0AD6      ;-
                       0AD6
                       0AD6      STOP_TIMER:
                       0AD6
              00  E5   0AD6             BBCC    #PB$V_TIM,-             ; Clear the timeout in progress bit
        00 44 A3       0AD8                     PB$W_STS(R3),10$        ;  in specified pathblock
                  05   0ADB      10$:   RSB                             ; Return
```

```
                               0ADC             .SBTTL  SET_CIRCUIT, PORT OPENS A PORT-PORT VIRTUAL CIRCUIT
                               0ADC
                               0ADC    ;+
                               0ADC    ; SET_CIRCUIT allocates a datagram buffer.  If none are available,
                               0ADC    ; return with error status.  Otherwise, send the SETCKT datagram
                               0ADC    ; to the port.
                               0ADC    ;
                               0ADC    ; Inputs:
                               0ADC    ;
                               0ADC    ;        R2                      -Addr of START/STACK dg
                               0ADC    ;        R3                      -Addr of formative PB
                               0ADC    ;        R4                      -Addr of PDT
                               0ADC    ;
                               0ADC    ; Outputs:
                               0ADC    ;
                               0ADC    ;        R0                      -0/1 for fail/success
                               0ADC    ;        Other registers         -Preserved
                               0ADC    ;-
                               0ADC
                               0ADC             .ENABL  LSB
                               0ADC
                               0ADC    SET_CIRCUIT:
                               0ADC
                   52     DD   0ADC             PUSHL   R2                          ; Save dg addr
                F51F'     30   0ADE             BSBW    INT$ALLOC_PPDDG             ; Allocate a dg buffer
                31 50     E9   0AE1             BLBC    R0,SET_ERR                  ; Branch if none
                          D0   0AE4             MOVL    #<PPD$M_RSP@24>!-           ;
                               0AE5                      <PPD$C_INVTC@16>,-         ;
  0C A2    01180000 8F         0AE5                     PPD$B_PORT(R2)              ; Set opcode and ask for response
                F511'     30   0AEC             BSBW    INT$INS_COMQH               ; Issue the invalidate command
                F50E'     30   0AEF             BSBW    INT$ALLOC_DG1               ; Allocate a datagram buffer
                               0AF2                                                 ;  for the open circuit command
                20 50     E9   0AF2             BLBC    R0,SET_ERR                  ; Branch if insufficient pool
                          C9   0AF5             BISL3   #<PPD$M_RSP@24>!-           ; Open VC, reset sequence #'s
                               0AF6                      <PPD$C_SETCKT@16>,-        ; Get SETCKT back for pool
  0C A3    01190000 8F         0AF6                     PB$B_RSTATION(R3),-         ;
                    0C A2      0AFD                     PPD$B_PORT(R2)              ;
                          3C   0AFF             MOVZWL  #<PPD$M_CST!-               ;
                               0B00                      PPD$M_NR!PPD$M_NS>,-       ;
     10 A2     E000 8F         0B00                     PPD$W_MASK(R2)              ; Set mask
     14 A2     8000 8F    3C   0B05             MOVZWL  #PPD$M_CST,PPD$W_M_VAL(R2)
                F4F2'     30   0B0B             BSBW    INT$INS_COMQH               ; Send it on its way
             50   01      9A   0B0E             MOVZBL  #SS$_NORMAL,R0              ; Set status to success
                               0B11
                52 8ED0        0B11    10$:     POPL    R2                          ; Retreive dg addr
                          05   0B14             RSB                                 ; Return
                               0B15
                               0B15    SET_ERR:
                               0B15
                   50     D4   0B15             CLRL    R0                          ; Set status to failure
                   F8     11   0B17             BRB     10$                         ; Take common exit
                               0B19
                               0B19             .DSABL  LSB
                               0B19
                               0B19
                               0B19
                               0B19             .END
```

```
                                              N 11
PACONFIG                                               16-SEP-1984 01:14:51  VAX/VMS Macro V04-00          Page  71
Symbol table                                           10-SEP-1984 01:16:23  [DRIVER.SRC]PACONFIG.MAR;2         (37)

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

```
LOOKUP_EVENT                000004E0 R   01      PB$V_CUR_CBL                = 00000000
MAX_RAM_REV                 0000095A R   01      PB$V_TIM                    = 00000000
MAX_ROM_REV                 0000095C R   01      PB$W_RETRY                  = 00000022
MOVE_PB                     0000067F R   01      PB$W_SIZE                   = 00000008
MOVE_SB                     00000674 R   01      PB$W_STATE                  = 00000012
NEW_PATH                    00000108 R   01      PB$W_STS                    = 00000044
NEW_PATH_ERR                000001C8 R   01      PB$W_VCFAIL_RSN             = 00000046
NEXT_ACTION                 000004F2 R   01      PB_EXISTS                     000002AA R   01
NEXT_EVENT                  000004E2 R   01      PB_FOUND                      00000854 R   01
NEXT_REQID                  00000089 R   01      PB_NOT_FOUND                  0000085E R   01
NEXT_SB                     0000089D R   01      PB_STATE_ERR                  00000527 R   01
NEXT_STATE                  000004CC R   01      PDT$B_DQIMAP                  00000154
NOT_FOUND                   000008A2 R   01      PDT$B_HSHUT_DG                000001B0
PAER$K_ES_LOBG            = 00000008             PDT$B_MAX_PORT                0000017C
PAER$K_ES_LOGB            = 00000006             PDT$B_NXT_PORT                0000017E
PAER$K_ES_L1BG            = 00000009             PDT$B_PO_LBSTS                00000180
PAER$K_ES_L1GB            = 00000007             PDT$B_P1_LBSTS                00000181
PAER$K_ES_LST0            = 00000003             PDT$B_PLOGMAP                 00000134
PAER$K_ES_LST1            = 00000009             PDT$B_PORTMAP                 00000114
PAER$K_ES_LST2            = 00000007             PDT$B_PORT_NUM                0000017D
PAER$K_ES_LST3            = 00000009             PDT$B_REQIDPS                 0000017F
PAER$K_ES_LST4            = 0000000C             PDT$C_HSHUT_SIZ             = 00000014
PAER$K_ES_RSCKS           = 00000008             PDT$C_LENGTH                = 000000E4
PAER$K_ET_DALT            = 00000003             PDT$C_PAREGBASE               000000E4
PAER$K_ET_LMLT            = 00000042             PDT$C_PAREGEND                00000110
PB$B_CBL_STS              = 00000028             PDT$C_PQB                   = 000001E0
PB$B_PO_STS               = 00000029             PDT$L_CNF                     000000E4
PB$B_P1_STS               = 0000002A             PDT$L_CQ0                     000000F0
PB$B_PROTOCOL             = 00000048             PDT$L_CQ1                     000000F4
PB$B_RSTATE               = 00000021             PDT$L_DFQ                     000000FC
PB$B_RSTATION             = 0000000C             PDT$L_DFQHDR                  00000208
PB$B_RST_PORT             = 00000020             PDT$L_DGHDRSZ                 00000190
PB$B_SUBTYP               = 0000000B             PDT$L_DGNETHD                 00000194
PB$B_TYPE                 = 0000000A             PDT$L_DQELOGOUT               000002E0
PB$C_CLOSED               = 00000000             PDT$L_GPTBASE                 0000022C
PB$C_LENGTH               = 00000054             PDT$L_GPTLEN                  00000230
PB$C_OPEN                 = 00000003             PDT$L_LBDG                    00000184
PB$C_PALENGTH               00000060             PDT$L_MFQ                     00000100
PB$C_PWR_FAIL             = 00004000             PDT$L_MFQHDR                  0000020C
PB$C_ST_REC               = 00000002             PDT$L_MQELOGOUT               00000320
PB$C_ST_SENT              = 00000001             PDT$L_MSGHDRSZ              = 000000B4
PB$C_VC_FAIL              = 00008000             PDT$L_MTC                     00000104
PB$L_CDTLST               = 00000034             PDT$L_PFAR                    00000108
PB$L_CLSCKT_DG              00000054             PDT$L_PMC                     000000E8
PB$L_DUETIME              = 0000003C             PDT$L_POLLERDUE               0000018C
PB$L_FLINK                = 00000000             PDT$L_POLLSWEEP             = 000000D8
PB$L_PDT                  = 0000002C             PDT$L_POOLDUE                 00000188
PB$L_RPORT_FCN            = 0000001C             PDT$L_PPR                     0000010C
PB$L_RPORT_REV            = 00000018             PDT$L_PS                      000000EC
PB$L_RPORT_TYP            = 00000014             PDT$L_PSR                     000000F8
PB$L_SBLINK               = 00000030             PDT$L_SPTBASE                 00000224
PB$L_SCSMSG               = 00000040             PDT$L_SPTLEN                  00000228
PB$L_WAITQBL              = 0000003C             PDT$L_UCB0                  = 000000DC
PB$L_WAITQFL              = 00000038             PDT$L_VBDT                    0000021C
PB$M_CUR_CBL              = 00000001             PDT$L_VPQB                    00000218
PB$M_CUR_PS               = 00000001             PDT$L_WAITQBL               = 000000B0
PB$T_LPORT_NAME           = 00000024             PDT$L_WAITQFL               = 000000AC
```

```
PDT$M_CUR_LBS                  = 00000001                PPD$C_STACK                    = 00000001
PDT$M_LBDG                     = 00000004                PPD$C_STACK_LEN                = 0000003E
PDT$M_PRV_LBS                  = 00000002                PPD$C_START                    = 00000000
PDT$Q_COMQ2                      000001F0                PPD$C_START_LEN                = 0000003E
PDT$Q_COMQ3                      000001F8                PPD$K_LB_LENGTH                  00000046
PDT$Q_COMQBASE                   000001E0                PPD$K_LENGTH                     00000012
PDT$Q_COMQH                      000001E8                PPD$L_BLINK                      00000004
PDT$Q_COMQL                      000001E0                PPD$L_DG_DISC                    00000028
PDT$Q_DFREEQ                     000001D0                PPD$L_FLINK                      00000000
PDT$Q_FORMPB                     00000174                PPD$L_IN_VCD                     00000018
PDT$Q_MFREEQ                     000001D8                PPD$L_LBCRC                      00000042
PDT$Q_RSPQ                       00000200                PPD$L_P0_ACK                     00000010
PDT$Q_TEMP_RSPQ                  0000019C                PPD$L_P0_NAK                     00000014
PDT$V_CUR_LBS                  = 00000000                PPD$L_P0_NRSP                    00000018
PDT$V_LBDG                     = 00000002                PPD$L_P1_ACK                     0000001C
PDT$W_BDTLEN                     00000220                PPD$L_P1_NAK                     00000020
PDT$W_DQELEN                     00000210                PPD$L_P1_NRSP                    00000024
PDT$W_LPORT_STS                  00000110                PPD$L_REC_BOFF                   00000028
PDT$W_MQELEN                     00000214                PPD$L_REC_NAME                   00000024
PDT$W_PBCOUNT                    00000112                PPD$L_RPORT_FCN                  00000020
PDT$W_STDGDYN                    00000198                PPD$L_RPORT_REV                  0000001C
PDT$W_STDGUSED                   0000019A                PPD$L_RPORT_TYP                  00000018
POOL_DONE                        00000A27 R      01      PPD$L_SND_BOFF                   00000020
PPD$B_DEF_ST                     0000001C                PPD$L_SND_NAME                   0000001C
PPD$B_FLAGS                      0000000F                PPD$L_ST_ADDR                    00000018
PPD$B_HWVERS                     00000034                PPD$L_XCT_LEN                    00000018
PPD$B_LBDATA                     00000012                PPD$M_CST                      = 00008000
PPD$B_LCB_0                      00000012                PPD$M_DQI                      = 00001000
PPD$B_LCB_LPORT                  00000010                PPD$M_NR                       = 00004000
PPD$B_LCB_NPORT                  0000000F                PPD$M_NS                       = 00002000
PPD$B_LCB_OPC                    00000011                PPD$M_RSP                      = 00000001
PPD$B_LCB_PORT                   0000000E                PPD$Q_CURTIME                    00000048
PPD$B_OPC                        0000000E                PPD$Q_NODENAME                   00000040
PPD$B_PORT                       0000000C                PPD$Q_SWINCARN                   00000028
PPD$B_PROTOCOL                   0000001A                PPD$Q_XCT_ID                     00000010
PPD$B_RSTATE                     00000025                PPD$S_PS                       = 00000002
PPD$B_RST_PORT                   00000024                PPD$S_RP                       = 00000002
PPD$B_STATUS                     0000000D                PPD$S_SP                       = 00000002
PPD$B_SWFLAG                     0000000B                PPD$S_STATE                    = 00000002
PPD$B_SYSTEMID                   00000014                PPD$T_HWTYPE                     00000030
PPD$B_TYPE                       0000000A                PPD$T_SWTYPE                     00000020
PPD$C_ACK                      = 00000002                PPD$T_SWVERS                     00000024
PPD$C_ACK_LEN                  = 00000004                PPD$V_PS                       = 00000001
PPD$C_ENAB                     = 00000002                PPD$V_RP                       = 00000001
PPD$C_HOSTSHUT                 = 00000006                PPD$V_SP                       = 00000004
PPD$C_HSHUT_LEN                = 00000002                PPD$V_STATE                    = 00000001
PPD$C_INVTC                    = 00000018                PPD$W_LCB_LEN7                   0000000C
PPD$C_LB_LENGTH                  00000046                PPD$W_LENGTH                     00000010
PPD$C_LCB_DATA                   00000013                PPD$W_MASK                       00000010
PPD$C_LENGTH                     00000012                PPD$W_MAXDG                      0000001C
PPD$C_MIN_DGSIZ                  00000050                PPD$W_MAXMSG                     0000001E
PPD$C_PRT_ELOG                 = 00000001                PPD$W_MTYPE                      00000012
PPD$C_PSP0                     = 00000001                PPD$W_M_VAL                      00000014
PPD$C_PSP1                     = 00000002                PPD$W_SIZE                       00000008
PPD$C_REQID                    = 00000005                PR$_IPL                        = 00000012
PPD$C_SETCKT                   = 00000019                REC_ERROR_DG                     0000077E R      01
PPD$C_SNDDG                    = 00000001                REFRESH_SB                       00000637 R      01
```

```
                                                        D 12
PACONFIG                                                             16-SEP-1984 01:14:51  VAX/VMS Macro V04-00          Page  74
Symbol table                                                         10-SEP-1984 01:16:23  [DRIVER.SRC]PACONFIG.MAR;2          (37)

REV_OK                            0000099F R      01       START_TIMER                      00000AC0 R      01
REV_TABLE_SIZ                   = 00000004                 STATUS                         = 00000080
SB$B_HWVERS                     = 00000038                 STOP_NEXT                        00000305 R      01
SB$B_SYSTEMID                   = 00000018                 STOP_TIMER                       00000AD6 R      01
SB$B_TYPE                       = 0000000A                 SYS$GQ_VERSION                   ********   X    01
SB$K_LENGTH                     = 00000060                 SYSAP$C_DISPPO                 = 00000002
SB$L_CSB                        = 0000005C                 SYSAP$C_DISPQ                  = 00000000
SB$L_DDB                        = 00000054                 TRY_TRANSIT                      0000025E R      01
SB$L_FLINK                      = 00000000                 UCB$B_ERTCNT                   = 00000080
SB$L_PBBL                       = 00000010                 UCB$B_LMERTCNT                   000000D2
SB$L_PBLONNX                    = 00000014                 UCB$B_LMERTMAX                   000000D3
SB$L_PBFL                       = 0000000C                 UCB$B_LMEST                      000000D0
SB$Q_SWINCARN                   = 0000002C                 UCB$B_LMET                       000000D1
SB$T_HWTYPE                     = 00000034                 UCB$K_ERRDGBYTS                = 000000B4
SB$T_NODENAME                   = 00000044                 UCB$K_LMPKTBYTS                = 00000040
SB$T_SWTYPE                     = 00000024                 UCB$L_CICMD                      000000F0
SB$T_SWVERS                     = 00000028                 UCB$L_DDB                      = 00000028
SB$W_MAXDG                      = 00000020                 UCB$L_DPC                      = 0000009C
SB$W_MAXMSG                     = 00000022                 UCB$L_MSGFKBLK                   000000A0
SB$W_SIZE                       = 00000008                 UCB$N_LSADDR                     000000D8
SB_DONE                           0000076C R      01       UCB$N_LSID                       000000DE
SCAN_FORMPB                       000009CA R      01       UCB$N_RSADDR                     000000E4
SCS$ALL_FRDGS                     ********   X    01       UCB$N_RSID                       000000EA
SCS$GA_LOCALSB                    ********   X    01       UCB$T_MSGDATA                    000000F8
SCS$GB_NODENAME                   ********   X    01       UCB$T_OPAO_TEMP                  000000B8
SCS$GB_PAMXPORT                   ********   X    01       UCB$V_ONLINE                   = 00000004
SCS$GB_PANOPOLL                   ********   X    01       UCB$W_ERRCNT                   = 00000082
SCS$GB_PANPOLL                    ********   X    01       UCB$W_LMERRCNT                   000000D4
SCS$GB_SYSTEMID                   ********   X    01       UCB$W_MSGBYTCNT                  000000F4
SCS$GQ_CONFIG                     ********   X    01       UCB$W_MSGPPDTYP                  000000F6
SCS$GW_MAXDG                      ********   X    01       UCB$W_STS                      = 00000064
SCS$GW_PAPOLINT                   ********   X    01       UPDATE_CBL_STS                   000001CB R      01
SCS$GW_PAPOOLIN                   ********   X    01       UPDATE_LEN                     = 0000003C
SCS$GW_PASTMOUT                   ********   X    01       UPDATE_SWINCARN                  000005B7 R      01
SCS$NEW_SB                        ********   X    01
SCS$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
STS0_CODE                       = 00000000
STS$W_NEXT                      = 00000002
START_REQID                       00000082 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
------------------                        --------------
_$255$DUA28:[DRIVER.OBJ]PALIB.MLB;1                    8
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                        12
_$255$DUA28:[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
```

0113 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PAEND
LIS

PACONFIG
LIS

PAERROR
LIS