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
```

```
**FILE**ID**PAERROR                                    J 12
```

```
PPPPPPPP      AAAAAA    EEEEEEEEEE  RRRRRRRR    RRRRRRRR      OOOOOO    RRRRRRRR
PPPPPPPP      AAAAAA    EEEEEEEEEE  RRRRRRRR    RRRRRRRR      OOOOOO    RRRRRRRR
PP      PP  AA      AA  EE          RR      RR  RR      RR  OO      OO  RR      RR
PP      PP  AA      AA  EE          RR      RR  RR      RR  OO      OO  RR      RR
PP      PP  AA      AA  EE          RR      RR  RR      RR  OO      OO  RR      RR
PP      PP  AA      AA  EE          RR      RR  RR      RR  OO      OO  RR      RR
PPPPPPPP    AA      AA  EEEEEEEE    RRRRRRRR    RRRRRRRR    OO      OO  RRRRRRRR
PPPPPPPP    AA      AA  EEEEEEEE    RRRRRRRR    RRRRRRRR    OO      OO  RRRRRRRR
PP          AAAAAAAAAA  EE          RR  RR      RR  RR      OO      OO  RR  RR
PP          AAAAAAAAAA  EE          RR  RR      RR  RR      OO      OO  RR  RR
PP          AA      AA  EE          RR    RR    RR    RR    OO      OO  RR    RR        ....
PP          AA      AA  EE          RR    RR    RR    RR    OO      OO  RR    RR        ....
PP          AA      AA  EEEEEEEEEE  RR      RR  RR      RR    OOOOOO    RR      RR      ....
PP          AA      AA  EEEEEEEEEE  RR      RR  RR      RR    OOOOOO    RR      RR      ....

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

```
0000            .TITLE  PAERROR Error Handling & Logging Routines
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:  ROUTINES TO HANDLE CI VIRTUAL CIRCUIT RECOVERY
0000    ;
0000    ; AUTHOR:  N. KRONENBERG,  DECEMBER 1981
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V04-001 NPK3066         N. Kronenberg                7-Sep-1984
0000    ;               Disable invalid buffer name bugcheck since bug is found.
0000    ;               With this edit, all optional bugchecks are disabled
0000    ;               and maximum error recovery enabled.
0000    ;
0000    ;       V03-040 NPK3065         N. Kronenberg               23-Aug-1984
0000    ;               Disable MFQE optional bugcheck since bug is found.
0000    ;
0000    ;       V03-039 NPK3061         N. Kronenberg                9-Aug-1984
0000    ;               Remove optional debug bugcheck on unexpected port
0000    ;               interrupt.
0000    ;
0000    ;       V03-038 NPK3060         N. Kronenberg                1-Aug-1984
0000    ;               Remove remote port from OPA0 messages concerning
0000    ;               loopback dgs since no remote port is applicable.
0000    ;               Make loopback dg OPA0 messages be reported always.
0000    ;
0000    ;       V03-037 NPK3058         N. Kronenberg               25-Jul-1984
0000    ;               Add MFQE optional bugcheck enable flag and enable
```

```
0000  ;                 three kinds of optional bugchecks.
0000  ;
0000  ;         V03-036 NPK3057         N. Kronenberg           23-Jul-1984
0000  ;                 Change the OPA0 message descriptors for cpu/port
0000  ;                 ucode not at required rev level not to include
0000  ;                 offline messages since these are generated separately
0000  ;                 in PAINIT, CLEANUP_PDT.
0000  ;
0000  ;         V03-035 NPK3055         N. Kronenberg           14-Jul-1984
0000  ;                 Change OPA0 error log msgs for cpu/port ucode rev
0000  ;                 error to include port offline msg.  Change wording
0000  ;                 of cpu ucode rev error msg to say that rev is insufficient
0000  ;                 for CI activity.
0000  ;                 Add separate port ucode rev warning msg that does not
0000  ;                 include offline announcement.
0000  ;                 Add ELOG$CPU_REV, ELOG$UCODE_ERR, ELOG_UCODE_WARN.
0000  ;
0000  ;         V03-034 NPK3054         N. Kronenberg           24-Jun-1984
0000  ;                 Add OPA0 messages to warn operator of either CPU
0000  ;                 rev level insufficient to support ci, or the ci
0000  ;                 ucode rev level is insufficient.
0000  ;
0000  ;         V03-033 NPK3053         N. Kronenberg           17-May-1984
0000  ;                 Fix branch error in NPK3052.
0000  ;
0000  ;         V03-032 NPK3052         N. Kronenberg           4-May-1984
0000  ;                 Fix ERR$PWF_RECOV to properly handle a port failure
0000  ;                 for a port with circuits in VC_FAIL state.
0000  ;
0000  ;         V03-031 NPK3048         N. Kronenberg           9-Apr-1984
0000  ;                 Add two new $DEBUGCHECK enable flags.
0000  ;
0000  ;         V03-030 TMK0005         Todd M. Katz            25-Mar-1984
0000  ;                 Change the text of the remote system conflicts _OPA0 error
0000  ;                 logging message.
0000  ;
0000  ;         V03-029 TMK0004         Todd M. Katz            24-Mar-1984
0000  ;                 When it is decided to log an error condition to _OPA0, a fork
0000  ;                 process is created to format and broadcast an appropriate
0000  ;                 message. It is absolutely necessary that all messages be
0000  ;                 formated at fork IPL. This is because there is only one copy
0000  ;                 of each message, but there maybe multiple CI ports making use
0000  ;                 of each message.
0000  ;
0000  ;                 However, what is incorrect is that the optional data which maybe
0000  ;                 used for formatting a _OPA0 error log message is being extracted
0000  ;                 from the UCB error logging buffer or from the device registers
0000  ;                 within the context of the fork process. By the time the fork
0000  ;                 process gets a chance to execute and make use of this optional
0000  ;                 data for formatting a message, it is possible (and in the case
0000  ;                 of device registers certain) that the values stored in these
0000  ;                 locations will have changed.
0000  ;
0000  ;                 The solution to this problem is to store the needed information
0000  ;                 within UCB$T_OPA0_TEMP (a new UCB field three longwards in size)
0000  ;                 just before the creation of the fork process within OPA0_LOG.
0000  ;                 Then, whenever optional formatting of an _OPA0 error log message
```

```
0000 ;          is required, the routines which perform the formatting make use
0000 ;          of the information stored in this UCB location.
0000 ;
0000 ;          Three types of information maybe required for additional
0000 ;          formatting - device registers, a remote port number, or CI
0000 ;          packet information. I have defined a OPA0 error logging control
0000 ;          flag for each information type. For a given error condition the
0000 ;          setting of these control flags will direct what information is
0000 ;          saved within this new UCB location, before the fork process is
0000 ;          created, to be used in the formatting of the appropriate _OPA0
0000 ;          error log message.
0000 ;
0000 ;  V03-028 TMK0003         Todd M. Katz            06-Mar-1984
0000 ;          Add support for _OPA0 error logging. This involves determining,
0000 ;          whenever error logging is to be done, whether or not an attempt
0000 ;          should also be made to log the error condition at _OPA0. Such
0000 ;          logging will always be attempted for certain error conditions,
0000 ;          and it will also be done whenever it is found that the system
0000 ;          device, which is presumed to also be the error logging device,
0000 ;          is currently unavailable.
0000 ;
0000 ;          A table driven routine, OPA0_LOG, is used to determine whether
0000 ;          or not _OPA0 error logging should always be done for a given
0000 ;          error condition as well as to provide the error logging message
0000 ;          to be broadcast to _OPA0 and optional formatting information.
0000 ;          When a decision is made to perform this error logging, the UCB's
0000 ;          message fork block is used to create a fork process provided it
0000 ;          is not already in use (in which case _OPA0 error logging will be
0000 ;          bypassed for this error condition). When this fork process
0000 ;          resumes control at OPA0_LOG_FORK, it proceeds to format an error
0000 ;          logging message and broadcast it to _OPA0. In the case of
0000 ;          certain unrecoverable port initialization errors, this fork
0000 ;          process will also broadcast a second message indicating that the
0000 ;          port will be left offline.
0000 ;
0000 ;  V03-027 TMK0002         Todd M. Katz            21-Feb-1984
0000 ;          Make the following changes to fix several bugs, and in support
0000 ;          of allowing port initialization to proceed at IPL 8 instead of
0000 ;          at IPL$_POWER:
0000 ;
0000 ;          1. Do not disable all interrupts by raising IPL to IPL$_POWER
0000 ;             before calling INI$PORT from within ERR$INIPORT. Port
0000 ;             initialization is now being done at fork IPL instead of at
0000 ;             IPL$_POWER.
0000 ;
0000 ;          2. Disable device interrupts within ERR$INIPORT before calling
0000 ;             INI$PORT to re-initialize the port. This is done by
0000 ;             explicitely placing the port within the un-initialized state.
0000 ;             If this is not done it is possible that the port maybe in the
0000 ;             un-initialized state but with device interrupts enabled when
0000 ;             port re-initializtion begins. Then if a device interrupt
0000 ;             occurs during port re-initialization it may prevent the
0000 ;             un-initialized -> disabled state transition from occurring
0000 ;             at the proper time. The end result is that a second attempt
0000 ;             at re-initializing the port will be required.
0000 ;
0000 ;          3. The way in which ERR$PWF_RECOV is forking is incorrect.
```

```
0000  ;               It does not make proper use of the UCB_V_FKLOCK fork block
0000  ;               interlock bit. It never sets the interlock bit before using
0000  ;               the fork block if the fork block is currently not in use.
0000  ;               This may result in this same fork block being used twice
0000  ;               in succession. In such a situation the context saved by the
0000  ;               first fork, the fork initiated by ERR$PWF_RECOV, would be
0000  ;               overwritten by the context of the second fork.
0000  ;
0000  ;               I have corrected this problem by utilizing the new routine
0000  ;               INI$FORK to control the forking. This routine knows how to
0000  ;               extract the fork block from the appropriate fork queue in an
0000  ;               atomic fashion, and how to make proper use of the fork block
0000  ;               interlock bit. This routine always returns control at fork
0000  ;               IPL by jumping to the address provided it as input in R3.
0000  ;
0000  ;            4. I have also corrected an error in how ERR$PWF_RECOV cleans up
0000  ;               a local port's path blocks, and crashes the local port.
0000  ;               This routine should only be crashing the port after every
0000  ;               SYSAP with a connection over the port has been notified and
0000  ;               has had a chance to issue a DISCONNECT. A DISCONNECT, under
0000  ;               such a circumstance, would result in the path block being
0000  ;               deleted, and the count of path blocks associated with the
0000  ;               port being decremented, if the disconnected connection
0000  ;               represented the path's last connection. Therefore,
0000  ;               ERR$PWF_RECOV should only be crashing the port when the count
0000  ;               of path blocks associated with the port reaches zero
0000  ;               indicating that every SYSAP which had a connection over this
0000  ;               port has been notified and issued a DISCONNECT.
0000  ;
0000  ;               Unfortunately when the co-routine CNF$LKP_PB_PDT encounters
0000  ;               the end of the PB list, ERR$PWF_RECOV immediately crashes the
0000  ;               port regardless of the number of path blocks still associated
0000  ;               with the port. I have corrected this routine so that when the
0000  ;               end of the port's path block list is encountered,
0000  ;               ERR$PWF_RECOV will only crash the port if the count of the
0000  ;               port's assocated path blocks is zero.
0000  ;
0000  ;    V03-026 TMK0001          Todd M. Katz           14-Feb-1984
0000  ;            Add support for error logging of the refusals of the local
0000  ;            port to open up a virtual circuit to a remote port because of
0000  ;            conflictions between information provided by the remote system
0000  ;            and a known system within the system-wide configuration data
0000  ;            base. This support involves modification to ELOG$PACKET so that
0000  ;            a special type of packet is logged whenever this event occurs.
0000  ;            Instead of logging a data packet, this event results in the
0000  ;            logging of the known system ID, the known system nodename, and
0000  ;            the remote system nodename in addition to the usaul stuff which
0000  ;            is always logged (local station address, etc... ).
0000  ;
0000  ;            Also, fix two small bugs within ELOG$PACKET. Currently, the
0000  ;            entire message logging area is not being used (or is not being
0000  ;            zeroed out if there is no packet to be logged). This is because
0000  ;            the destination sizes used in the MOVC5s only include 4 bytes
0000  ;            of the 8 bytes of CI packet command/control/status
0000  ;            information, CI packet PPD type, and CI packet message data
0000  ;            length.
0000  ;
```

```
0000    ;       V03-025 NPK3044         N. Kronenberg           6-Feb-1984
0000    ;               Add ELOG$ERROR_DG to log an error datagram.  Modify
0000    ;               ELOG$$LOG_LM to handle error log datagrams which are
0000    ;               larger than other logged messages.
0000    ;               Disable all optional bugchecks in ERR$DEBUGCHECK.
0000    ;
0000    ;       V03-024 NPK3043         N. Kronenberg           6-Feb-1984
0000    ;               Fix ELOG$$LOG_LM to copy all 6 bytes of local sysid.
0000    ;
0000    ;       V03-023 NPK3039         N. Kronenberg           11-Jan-1984
0000    ;               Zero PB$L_CLSCKT_DG when closing vc in ERR$CRASHVC.
0000    ;               Add ERR$V_DEB_PSRX flag for enabling/disabling bugcheck
0000    ;               on interrupt with undefined bits set in PSR.
0000    ;
0000    ;       V03-022 NPK3038         N. Kronenberg           6-Dec-1983
0000    ;               Disable the ERR$DEBUGCHECK flags for connect request
0000    ;               with no path block and SCS bookkeeping with no path
0000    ;               block.
0000    ;
0000    ;       V03-021 NPK3037         N. Kronenberg           11-Nov-1983
0000    ;               Add ERR$DEBUGCHECK flags definitions and flags longwd.
0000    ;               Make subroutine CLEANUP_PKTS a global routine,
0000    ;               ERR$CLEANUP_PKT.
0000    ;               Make subroutine CALL_INIT_PORT a global routine,
0000    ;               ERR$INIPORT.
0000    ;               Remove queue interlock clear from FLUSH_Q since it
0000    ;               is already done in routine UNLOCK_BADQ.
0000    ;
0000    ;       V03-020 NPK3029         N. Kronenberg           22-Jul-1983
0000    ;               Enhancements for V4.0:
0000    ;               Change ERR$CRASH_PORT to not fake a power off to
0000    ;               prevent reinit of port if ERTCNT is exhausted
0000    ;               (INI$PORT now handles that.)
0000    ;               Change IOFORK to FORK in ERR$PWF_RECOV.
0000    ;               Remove references to PB$L_SB in favor of PB$L_SBLINK.
0000    ;
0000    ;       V03-019 NPK3024         N. Kronenberg           18-May-1983
0000    ;               Add logic for variable net header size to routine
0000    ;               ELOG$LOG_LM.
0000    ;
0000    ;       V03-018 KTA3046         Kerbey T. Altmann       30-Mar-1983
0000    ;               Redo for SCS/PPD split.
0000    ;
0000    ;       V03-017 NPK3011         N. Kronenberg           22-Nov-1982
0000    ;               Fix ERR$CRASH_PORT to call ERR$PWF_RECOV at device IPL.
0000    ;
0000    ;       V03-016 ROW0133         Ralph O. Weber          14-OCT-1982
0000    ;               Correct PPD$W_LENGTH reference in ELOG$$LOG_LM to PPD$W_SIZE.
0000    ;               This causes the allocated pool size value to be used, as
0000    ;               documented, when the maximum size of the message region to be
0000    ;               error logged is calculated.
0000    ;               This change will be distributed in Version 3.2.
0000    ;
0000    ;       V03-015 NPK3006         N. Kronenberg           9-Sep-1982
0000    ;               Comment possible aux status input to ERR$PWF_RECOV better.
0000    ;               Fix data structure error path by zeroing locked queue
0000    ;               headers in ERR$PWF_RECOV prior to forking down from
```

```
0000  ;                 device IPL.
0000  ;
0000  ;         V03-014 ROW0119         Ralph O. Weber          9-AUG-1982
0000  ;                 Modify ELOG$$LOG_LM so that it does not copy anything beyond
0000  ;                 the space allocated to a message packet as shown in the size
0000  ;                 word field of the standard pool unit header.
0000  ;                 This change will be in a new driver image shipped in V3.1.
0000  ;
0000  ;         V03-013 ROW0115         Ralph O. Weber          30-JUN-1982
0000  ;                 Modify ELOG$$LOG_LM to always copy first 68 bytes of message
0000  ;                 into UCB logged message buffer and to specially zero the
0000  ;                 buffer when no message packet exists.  Also replace
0000  ;                 ELOG$$LOG_LM system block search code with use of new
0000  ;                 PB$L_SBLINK pointer to SB.
0000  ;                 This change will be in a new driver image shipped in V3.1.
0000  ;
0000  ;         V03-12  NPK3001         N. Kronenberg           28-Jun-1982
0000  ;                 Clear UCB fork blk lock following power fail fork.
0000  ;
0000  ;         V03-011 ROW0111         Ralph O. Weber          27-JUN-1982
0000  ;                 Add ELOG$CABLES, a routine like ELOG$PACKET only with change
0000  ;                 of cable state error type.  This routine required for loopback
0000  ;                 datagram logging.  Add a clear for UCB$L_CICMD when there is
0000  ;                 no message packet so that it will be zero just like everything
0000  ;                 else.
0000  ;                 This change will be in a new driver image shipped in V3.1.
0000  ;
0000  ;         V03-010 ROW0110         Ralph O. Weber          24-JUN-1982
0000  ;                 Fix ELOG$$LOG_LM to adjust error count up by one while copying
0000  ;                 it into the UCB log message buffer, since UCB$W_ERRCNT has not
0000  ;                 yet been incremented.
0000  ;                 This change will be in a new driver image shipped in V3.1.
0000  ;
0000  ;         V03-009 ROW0108         Ralph O. Weber          24-JUN-1982
0000  ;                 Fix ELOG$PACKET and ELOG$$LOG_LM to handle case where no
0000  ;                 packet exists.  Also correct ELOG$PACKET so that error sub-
0000  ;                 type information is retrieved after CNF$LKP_PB_MSG is called.
0000  ;                 This change will be shipped with VAX/VMS Version 3.1.
0000  ;
0000  ;         V03-008 NPK3001         N. Kronenberg           22-Jun-1982
0000  ;                 Fix to keep UCB fork block locked on power fail
0000  ;                 recovery fork.
0000  ;
0000  ;         V03-007 ROW0098         Ralph O. Weber          7-JUN-1982
0000  ;                 Add call to error appropriate error logging routine at
0000  ;                 CONFIG_ERR in ERR$VCCLOSED_MSG.
0000  ;                 This change will be in a new driver image shipped in V3.1.
0000  ;
0000  ;         V03-006 ROW0092         Ralph O. Weber          3-JUN-1982
0000  ;                 Add error logging routines which generate logged message error
0000  ;                 log entries; ELOG$PACKET, ELOG$PTH_ST_CHG, and ELOG$CBL_X_CHG.
0000  ;                 Also added necessary definition macro references.
0000  ;                 This change will be in a new driver image shipped in V3.1.
0000  ;
0000  ;         V03-005 ROW0089         Ralph O. Weber          20-MAY-1982
0000  ;                 Add error logging routines which generate device attention
0000  ;                 error log entries; ELOG$INIT_SWERR, ELOG$UCODE_NORD,
0000   ;               ELOG$HARDWARE, and ELOG$INTRLOCK.  Also add register dump
0000   ;               routine, ELOG$REGDUMP.  Add necessary definition macro
0000   ;               references too.
0000   ;               This change will be in a new driver image shipped in V3.1.
0000   ;
0000   ;       V03-004 NPK2019         N. Kronenberg           6-Apr-1982
0000   ;               Changed DISP_ENTRY to global ERR$DISP_ENTRY.
0000   ;               Add routine ERR$CRASH_PORT.
0000   ;               Fix illegal CDT state in NOTIFY_SYSAP to be nonfatal
0000   ;               bugcheck with recovery rather than fatal bugcheck.
0000   ;               Fix PB lookup failure in ERR$VCCLOSED_MSG to crash VC.
0000   ;               Change queue interlock failure in FLUSH_Q to be non
0000   ;               fatal bugcheck.
0000   ;               Fix CHK_NO_CDTS to get remote port from PB and use
0000   ;               $TURNMSG.
0000   ;               Fix CLEANUP_PKTS to reset logout area longwd immediately
0000   ;               after processing entry.
0000   ;
0000   ;       V03-003 NPK2018         N. Kronenberg           29-Mar-1982
0000   ;               Modified ERR$CRASHVC_PB to use dg buffer in PB for
0000   ;               SETCKT instead of allocating buffer.
0000   ;               Broke ERR$DISC_VCFAIL into main routine and new
0000   ;               subroutine, CHR_NO_CDTS.
0000   ;               Made disconnect on power failure synchronous --
0000   ;               it suspends till CDT is actually removed.
0000   ;               Modified CONNECT_ABO and DCONNECT_OK in NOTIFY_SYSAP
0000   ;               to call CHK_NO_CDTS.
0000   ;
0000   ;       V03-002 NPK2018         N. Kronenberg           25-Mar-1982
0000   ;               Fix ERR$DISC_PWFAIL to purge out command queues again.
0000   ;
0000   ;       V03-001 NPK2016         N. Kronenberg           18-Mar-1982
0000   ;               Fixed .TITLE
0000   ;
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            $CDTDEF                                 ; Connection Descriptor offsets
        0000            $CLUBDEF                                ; Cluster Block offsets
        0000            $CRBDEF                                 ; Channel Request Block offsets
        0000            $DDBDEF                                 ; Device Data Block format
        0000            $DYNDEF                                 ; Dynamic data structures types
        0000            $EMBDEF                                 ; Error log buffer offsets
        0000            $EMBLTDEF                               ; Logged messages subtype values
        0000            $IDBDEF                                 ; Interrupt Data Block offsets
        0000            $IPLDEF                                 ; Define priority levels
        0000            $MCHKDEF                                ; Protect from machine check codes
        0000            $PBDEF                                  ; Path Blk offsets
        0000            $PDTDEF                                 ; Port Descriptor offsets
        0000            $SBDEF                                  ; System Block offsets
        0000            $SSDEF                                  ; System service return codes
        0000            $UCBDEF                                 ; UCB definitions
        0000            $VECDEF                                 ; CRB vector segment offsets
        0000
        0000    ;
        0000    ; PADRIVER definitions (PALIB.MLB):
        0000    ;
        0000
        0000            $PAERDEF                                ; PADRIVER error code definitions
        0000            $PAPBDEF                                ; PA-specific extension to PB
        0000            $PAPDTDEF                               ; PA-specific extension to PDT
        0000            $PAREGDEF                               ; CI port device register defns
        0000            $PAUCBDEF                               ; PA extension to UCB
        0000            $PPDDEF                                 ; PPD layer of msg/dg header
        0000            .cross
```

```
0000          .SBTTL  _OPA0 ERROR LOGGING DATA
0000
0000  ;+
0000  ; The routine which logs errors to _OPA0 is table driven. There are separate
0000  ; tables for device attention and logged message errors. What follows is the
0000  ; the macro that is used to generate each table entry, the two tables, various
0000  ; offsets to the fields within each table entry, and assorted constants.
0000  ;-
0000
0000  ;
0000  ; Macro to generate an entry within an _OPA0 error logging table. The format
0000  ; of each entry is as follows:
0000  ;
C000  ;       .BYTE   <ERROR SUBTYPE>
0000  ;       .BYTE   <ERROR TYPE>
0000  ;       .BYTE   <CONTROL FLAGS>
0000  ;       .BYTE   <OPTIONAL OFFSET TO MSG FIELD TO BE FORMATTED>
0000  ;       .WORD   <OPTIONAL OFFSET (from PA$CTLINIT) TO FORMATTING ROUTINE>
0000  ;       .WORD   <OFFSET (from PA$CTLINIT) TO ERROR MSG>
0000  ;
0000  ; All of the _OPA0 error messages are placed within their own PSECT. Each
0000  ; _OPA0 error logging table must be terminated by a word of -1.
0000  ;
0000
0000          .MACRO  $OPA0_LOG       TYPE,SUBTYPE,FLAGS,FORMAT,MSG
0000
0000          .IF     NB      TYPE
0000          .BYTE   <PAER$K_ES_'SUBTYPE>    ; Error Subtype
0000          .BYTE   <PAER$K_ET_'TYPE>       ; Error Type
0000
0000          .IF     NB      FLAGS
0000          .BYTE   FLAGS                   ; Flags affecting logging to OPA0
0000          .ENDC
0000          .IF     B       FLAGS
0000          .BYTE   0
0000          .ENDC
0000
0000          .IF     NB      FORMAT
0000          .BYTE   %LOCATE(<xx>,MSG)+11    ; Offset to field to be formatted
0000          .WORD   <FORMAT-PA$CTLINIT>     ; Optional formatting routine offset
0000          .ENDC
0000          .IF     B       FORMAT
0000          .BYTE   0
0000          .WORD   0
0000          .ENDC
0000
0000          .SAVE
0000          .PSECT  $$$110_MSGS
0000  $$MSG_PTR = .
0000          .ASCIC  <CR><LF><BELL>"%PAx0, 'MSG'"<CR><LF> ; Message to display at OPA0
0000          .RESTORE
0000          .WORD   <$$MSG_PTR-PA$CTLINIT>  ; OPA0 msg offset
0000          .ENDC
0000
0000          .IF     B       TYPE
0000          .WORD   -1                      ; -1 marks the end of the table
0000          .ENDC
```

```
               0000   479          .ENDM
```

```
                                 0000
                                 0000   ;
                                 0000   ; Offsets to the various fields within a _OPA0 error logging table entry.
                                 0000   ;
                                 0000
                      00000000   0000   SUBTYPE = 0                             ; Offset to Error subtype
                      00000001   0000   TYPE    = 1                             ; Offset to Error type
                      00000002   0000   CFLAGS  = 2                             ; Offset to Control Flags
                      00000003   0000   OFFSET  = 3                             ; Offset to Optional Formatting Offset
                      00000004   0000   FORMAT  = 4                             ; Offset to Optional Format Routine Offset
                      00000006   0000   MSG     = 6                             ; Offset to Error Message Offset
                                 0000
                      00000008   0000   OPA0_LOG_SIZE   = 8                     ; _OPA0 Error Logging Table Entry Size
                                 0000
                                 0000   ;
                                 0000   ; Define the bits within the control flags _OPA0 error logging table field.
                                 0000   ;
                                 0000
                      00000000   0000   V_ALWAYS  = 0                           ; Always print out this error message
                      00000001   0000   M_ALWAYS  = 1
                                 0000
                      00000001   0000   V_OFFLINE = 1                           ; Always print out a second message
                      00000002   0000   M_OFFLINE = 2                           ; (Port has gone Offline)
                                 0000
                      00000002   0000   V_RPORT   = 2                           ; Store the remote port number in the
                      00000004   0000   M_RPORT   = 4                           ; _OPA0 error loggging UCB data area
                                 0000
                      00000003   0000   V_PKT     = 3                           ; Store the CICMD packet information in
                      00000008   0000   M_PKT     = 8                           ; the _OPA0 error loggging UCB data area
                                 0000
                      00000004   0000   V_REGS    = 4                           ; Store the device registers in the
                      00000010   0000   M_REGS    = 16                          ; _OPA0 error loggging UCB data area
                                 0000
                                 0000   ;
                                 0000   ; Define ASCII symbols for various hexadecimal formatting characters.
                                 0000   ;
                                 0000
                      0000000D   0000   CR      = 13                            ; ASCII for carriage return,
                      0000000A   0000   LF      = 10                            ;  linefeed,
                      00000007   0000   BELL    = 7                             ;  and bell
                                 0000
                      00000006   0000   CTRLR_NAME      = 6                     ; Byte offset to device controller
                                 0000                                           ; letter in error logging messages
                                 0000
                                 0000   ;
                                 0000   ; Define table for hexadecimal -> ASCII and hexadecimal -> decimal -> ASCII
                                 0000   ; conversions.
                                 0000   ;
                                 0000
                                 0000   CONV_TABLE:
42 41 39 38 37 36 35 34 33 32 31 30   0000           .ASCII  /0123456789ABCDEF/
                      46 45 44 43   000C
```

```
0010
0010      ;
0010      ; Device Attention _OPA0 Error Logging Table.
0010      ;
0010
0010      DA_OPA0_LOG_TAB:
0010              $OPA0_LOG       INSW,POOL,M_ALWAYS+M_OFFLINE,,-
0010                      <Insufficient Non-paged Pool for Initialization>
0018              $OPA0_LOG       INSW,CODE,M_ALWAYS+M_OFFLINE,,-
0018                      <Failed to Locate Port Micro-code Image>
0020              $OPA0_LOG       INSW,SCSID,M_ALWAYS+M_OFFLINE,,-
0020                      <SCSSYSTEMID has NOT been set to a Non-zero Value>
0028              $OPA0_LOG       HW,UCDW,M_ALWAYS,,-
0028                      <Micro-code Verification Error>
0030              $OPA0_LOG       HW,INIT,M_ALWAYS+M_REGS,FORMAT_REGS,-
0030                      <Port Transition Failure - CNF/PMC/PSR  xxxxxxxx/xxxxxxxx/xxxxxxxx>
0038              $OPA0_LOG       HW,HWER,M_ALWAYS+M_REGS,FORMAT_REGS,-
0038                      <Port Error Bit(s) Set - CNF/PMC/PSR  xxxxxxxx/xxxxxxxx/xxxxxxxx>
0040              $OPA0_LOG       HW,PDWN,M_ALWAYS,,-
0040                      <Port Power Down>
0048              $OPA0_LOG       HW,PUP,M_ALWAYS,,-
0048                      <Port Power Up>
0050              $OPA0_LOG       HW,UXIN,M_ALWAYS+M_REGS,FORMAT_REGS,-
0050                      <Unexpected Interrupt - CNF/PMC/PSR  xxxxxxxx/xxxxxxxx/xxxxxxxx>
0058              $OPA0_LOG       HW,REVER,M_ALWAYS,FORMAT_REV,-
0058                      <CI port ucode not at required rev level.  RAM/PROM rev is xxxx/xxxx
0060              $OPA0_LOG       HW,REVCA,M_ALWAYS,FORMAT_REV,-
0060                      <CI port ucode not at current rev level.  RAM/PROM rev is xxxx/xxxx>
0068              $OPA0_LOG       HW,CPUREV,M_ALWAYS,,-
0068                      <CPU ucode not at required rev level for CI activity>
0070              $OPA0_LOG       ILCK,MQRM,M_ALWAYS,,-
0070                      <Message Free Queue Remove Failure>
0078              $OPA0_LOG       ILCK,DQRM,M_ALWAYS,,-
0078                      <Datagram Free Queue Remove Failure>
0080              $OPA0_LOG       ILCK,RQRM,M_ALWAYS,,-
0080                      <Response Queue Remove Failure>
0088              $OPA0_LOG       ILCK,HCIN,M_ALWAYS,,-
0088                      <High Priority Command Queue Insert Failure>
0090              $OPA0_LOG       ILCK,LCIN,M_ALWAYS,,-
0090                      <Low Priority Command Queue Insert Failure>
0098              $OPA0_LOG       ILCK,MQIN,M_ALWAYS,,-
0098                      <Message Free Queue Insert Failure>
00A0              $OPA0_LOG       ILCK,DQIN,M_ALWAYS,,-
00A0                      <Datagram Free Queue Insert Failure>
00A8              $OPA0_LOG
```

```
00AA
00AA    ;
00AA    ; Logged Message _OPA0 Error Logging Table.
00AA    ;
00AA
00AA    LM_OPA0_LOG_TAB:
00AA                $OPA0_LOG       PKT,UPKT,M_ALWAYS+M_PKT,FORMAT_PKT,-
00AA                        <Unrecognized SCA Packet - FLAGS/OPC/STATUS/PORT  xx/xx/xx/xx>
00B2                $OPA0_LOG       PKT,PCVC,M_ALWAYS+M_RPORT,FORMAT_PORT,-
00B2                        <Port has Closed Virtual Circuit - REMOTE PORT xxx>
00BA                $OPA0_LOG       PKT,CSHP,M_ALWAYS,,-
00BA                        <Software Shutting Down Port>
00C2                $OPA0_LOG       PKT,SCVC,M_ALWAYS+M_RPORT,FORMAT_PORT,-
00C2                        <Software is Closing Virtual Circuit - REMOTE PORT xxx>
00CA                $OPA0_LOG       PKT,CNPB,M_ALWAYS+M_PKT,FORMAT_PKT,-
00CA                        <Received Connect Without Path-Block - FLAGS/OPC/STATUS/PORT  xx/xx/
00D2                $OPA0_LOG       PKT,SCA,M_ALWAYS+M_PKT,FORMAT_PKT,-
00D2                        <Inappropriate SCA Control Message - FLAGS/OPC/STATUS/PORT  xx/xx/xx
00DA                $OPA0_LOG       PKT,NOPB,M_ALWAYS+M_RPORT,FORMAT_PORT,-
00DA                        <No Path-Block During Virtual Circuit Close - REMOTE PORT xxx>
00E2                $OPA0_LOG       PKT,ERRDG,M_RPORT,FORMAT_PORT,-
00E2                        <HSC Error Logging Datagram Received - REMOTE PORT xxx>
00EA                $OPA0_LOG       PKT,RSCKS,M_ALWAYS+M_RPORT,FORMAT_PORT,-
00EA                        <Remote System Conflicts with Known System - REMOTE PORT xxx>
00F2                $OPA0_LOG       CBL,0GB,M_RPORT,FORMAT_PORT,-
00F2                        <Path #0. Has gone from GOOD to BAD - REMOTE PORT xxx>
00FA                $OPA0_LOG       CBL,1GB,M_RPORT,FORMAT_PORT,-
00FA                        <Path #1. Has gone from GOOD to BAD - REMOTE PORT xxx>
0102                $OPA0_LOG       CBL,0BG,M_RPORT,FORMAT_PORT,-
0102                        <Path #0. Has gone from BAD to GOOD - REMOTE PORT xxx>
010A                $OPA0_LOG       CBL,1BG,M_RPORT,FORMAT_PORT,-
010A                        <Path #1. Has gone from BAD to GOOD - REMOTE PORT xxx>
0112                $OPA0_LOG       CBL,UC,M_RPORT,FORMAT_PORT,-
0112                        <Cables have gone from UNCROSSED to CROSSED - REMOTE PORT xxx>
011A                $OPA0_LOG       CBL,CU,M_RPORT,FORMAT_PORT,-
011A                        <Cables have gone from CROSSED to UNCROSSED - REMOTE PORT xxx>
0122                $OPA0_LOG       CBL,L0GB,M_ALWAYS,,-
0122                        <Path #0. Loopback has gone from GOOD to BAD>
012A                $OPA0_LOG       CBL,L1GB,M_ALWAYS,,-
012A                        <Path #1. Loopback has gone from GOOD to BAD>
0132                $OPA0_LOG       CBL,L0BG,M_ALWAYS,,-
0132                        <Path #0. Loopback has gone from BAD to GOOD>
013A                $OPA0_LOG       CBL,L1BG,M_ALWAYS,,-
013A                        <Path #1. Loopback has gone from BAD to GOOD>
0142                $OPA0_LOG       CBL,L0BX,M_RPORT,FORMAT_PORT,-
0142                        <Path #0. Has become working but CROSSED to Path #1. - REMOTE PORT x
014A                $OPA0_LOG       CBL,L1BX,M_RPORT,FORMAT_PORT,-
014A                        <Path #1. Has become working but CROSSED to Path #0. - REMOTE PORT x
0152                $OPA0_LOG
```

```
                            0154              .SBTTL  ERR$CRASHVC,              CRASH VC ON SPECIFIED
                            0154              .SBTTL  -                         PATH BLOCK
                            0154
                            0154      ;+
                            0154      ; These routines are called to crash an open virtual circuit on
                            0154      ; a specific path.  ERR$CRASHVC sets VC failure in progress
                            0154      ; status in the PB and does a SETCKT closed to the remote port.  Return
                            0154      ; is then taken since the SETCKT response will continue the process of
                            0154      ; cleaning up the broken VC.
                            0154      ;
                            0154      ; In case the response pkt is a REQID or other datagram type pkt,
                            0154      ; there may be no path block.  In this case, return is taken without
                            0154      ; doing anything.
                            0154      ;
                            0154      ; Inputs:
                            0154      ;
                            0154      ;       IPL                             -Fork IPL
                            0154      ;       R1                              -Addr of PB
                            0154      ;       R2                              -Addr of msg/dg response
                            0154      ;       R4                              -PDT addr
                            0154      ;
                            0154      ;       VC state                        -open
                            0154      ;
                            0154      ;
                            0154      ; Outputs:
                            0154      ;
                            0154      ;       R0-R1                           -Destroyed
                            0154      ;       Other registers                 -Preserved;  in particular, the msg/dg
                            0154      ;                                        pointed to by R2 is not disposed of --
                            0154      ;                                        that is the caller's responsibility
                            0154      ;-
                            0154
                            0154              .ENABL  LSB
                            0154
                            0154      ERR$CRASHVC::
                            0154
                51    D5    0154              TSTL    R1                    ; Got a valid path block?
                34    13    0156              BEQL    20$                   ; No, just leave
                52    DD    0158              PUSHL   R2                    ; Save caller's R2
             12 A1    B1    015A              CMPW    PB$W_STATE(R1),-      ; Is virtual circuit failure
           8000 8F          015D                      #PB$C_VC_FAIL         ;  already in progress?
                27    13    0160              BEQL    10$                   ; Branch if so
           8000 8F    B0    0162              MOVW    #PB$C_VC_FAIL,-       ; Set VC failure in progress
             12 A1          0166                      PB$W_STATE(R1)        ;  on this PB
        52   54 A1    D0    0168              MOVL    PB$L_CLSCKT_DG(R1),R2 ; Get addr of SETCKT dg in PB
             54 A1    D4    016C              CLRL    PB$L_CLSCKT_DG(R1)    ; Zero dg address to show that port
                            016F                                            ;  owns pkt now
                      C9    016F              BISL3   #<PPD$M_RSP@24>!-     ; Tell port to mark VC closed
                            0170                      <PPD$C_SETCKT@16>,-
   0C A1   01190000 8F      0170                      PB$B_RSTATION(R1),-   ;  to this remote station
             0C A2          0177                      PPD$B_PORT(R2)        ; Do SETCKT at top priority
     10 A2   8000 8F  3C    0179              MOVZWL  #PPD$M_CST,PPD$W_MASK(R2) ;  to close VC
             14 A2    D4    017F              CLRL    PPD$W_M_VAL(R2)       ; Get response to reclaim buffer
                01    90    0182              MOVB    #PPD$M_DISPOSE,-
             0B A2          0184                      PPD$B_SWFLAG(R2)      ; Ask interrupt serv to notify us
             FE77'    30    0186              BSBW    INT$INS_COMQH         ; Do it
              52 8ED0       0189      10$:    POPL    R2                    ; Restore caller's R2
```

```
                            05  018C     686 20$:    RSB                                   ; Return
                                018D     687
                                018D     688         .DSABL  LSB
```

```
                      018D        .SBTTL  ERR$CRASH_PORT,          INIT PORT CRASH
                      018D
                      018D  ;+
                      018D  ; ERR$CRASH_PORT is called by the driver at fork IPL detecting an error
                      018D  ; which might be either a software error or a port hardware or firmware error.
                      018D  ;
                      018D  ; Action is to maintenance init the port to prevent further activity,
                      018D  ; and, if there are any error retries left, to call ERR$PWF_RECOV
                      018D  ; in simulation of a power fail recovery.  If no retries are left,
                      018D  ; then PUP is cleared in PDT$W_LPORT_STS to prevent the port from being
                      018D  ; reinitialized.  ERR$PWF_RECOV initiates a fork process on the UCB
                      018D  ; which takes care of notifying SYSAPs and cleaning up the configuration
                      018D  ; database eventually.  The main difference between deliberately crashing
                      018D  ; the port and a real power failure is that in the crash case, cached
                      018D  ; packets are not written to the logout area by the port and hence may not
                      018D  ; be reclaimed.
                      018D  ;
                      018D  ; Inputs:
                      018D  ;
                      018D  ;       R4                              -PDT addr
                      018D  ;
                      018D  ;       (SP)                            -Caller's PC
                      018D  ;
                      018D  ; Outputs:
                      018D  ;
                      018D  ;       R0,R1                           -Destroyed
                      018D  ;
                      018D  ;       Other registers                 -Preserved
                      018D  ;-
                      018D
                      018D        .ENABL  LSB
                      018D
                      018D  ERR$CRASH_PORT::
                      018D
          00   E2     018D        BBSS    #PDT$V_PWF_CLNUP,-          ; Set PWF cleanup in progress
  2D 0110 C4          018F                PDT$W_[PORT_STS(R4),20$   ;  Branch if set already
          3C   BB     0193        PUSHR   #^M<R2,R3,R4,R5>            ; Save registers
          01   D0     0195        MOVL    #PA_PMC_M_MIN,-             ; Maintenance init the port
     00E8 D4          0197                @PDT$L_PMC(R4)              ;
55   00DC C4   D0     019A        MOVL    PDT$L_UCB0(R4),R5           ; Get UCB addr
          10   AA     019F        BICW    #UCB$M_ONLINE,-             ; Set unit offline to show init
       64 A5          01A1                UCB$W_STS(R5)               ;  in progress
   51     2C   3C     01A3        MOVZWL  #SS$_ABORT,R1               ; Assume we have more retries,
                      01A6                                            ;  but let SYSAP know not to
                      01A6                                            ;  expect cached entries back
     0080 C5   97     01A6        DECB    UCB$B_ERTCNT(R5)            ; Decr retry count
          05   18     01AA        BGEQ    10$                         ; Branch if not out of retries
51   0054 8F   3C     01AC        MOVZWL  #SS$_CTRLERR,R1             ; Else set aux status to tell
                      01B1                                            ;  SYSAP's port won't return
                      01B1
                      01B1  10$:  DSBINT  UCB$B_DIPL(R5)              ; Set IPL up to device to block
                      01B8                                            ;  interrupts
        0006   30     01B8        BSBW    ERR$PWF_RECOV               ; Treat like power failure from here on
                      01BB        ENBINT                              ; Restore IPL to fork IPL
          3C   BA     01BE        POPR    #^M<R2,R3,R4,R5>            ; Restore registers
                      01C0
          05          01C0  20$:  RSB                                 ; Return to caller
```

```
                                    01C1    747
                                    01C1    748         .DSABL  LSB
```

```
01C1         .SBTTL  ERR$PWF_RECOV,          NOTIFY SYSAPS WITH
01C1         .SBTTL  -                       CONNECTIONS ON POWER
01C1         .SBTTL  -                       FAILED PORT
01C1
01C1 ;+
01C1 ; ERR$PWF_RECOV is called by unit initialization on power fail
01C1 ; recovery or by port interrupt service on power down or by ERR$CRASH_PORT.
01C1 ; ERR$PWF_RECOV first checks for packet queues that might be corrupted
01C1 ; and for corrupted queues zeros the queue header, thus preventing
01C1 ; future attempts to remove entries for the queue and causing bugchecks.
01C1 ; ERR$PWF_RECOV then forks to lower IPL to the SCS syncronization
01C1 ; level.  Next, all formative path blocks on this PDT (i.e., START
01C1 ; handshakes in progress) are looked up and formative PB's and SB's are
01C1 ; deallocated to pool.
01C1 ;
01C1 ; ERR$PWF_RECOV then calls CNF$LKP_PB_PDT to look up PB's associated with
01C1 ; the failed PDT.  CNF$LKP_PB_PDT calls us back as a coroutine for each
01C1 ; PB found.  For each PB, the CDT list is searched and, for each open CDT,
01C1 ; the SYSAP error address is called with appropriate status.  SYSAP
01C1 ; DISCONNECTs issued as a result of error routines being called continue
01C1 ; the failure process.  (See routine ERR$DISC_PWFAIL for more info.)
01C1 ;
01C1 ; CDT's in non-open states are handled the same as described in ERR$VCCLOSED_MSG.
01C1 ;
01C1 ; There is a difference between connection cleanup following a VC failure
01C1 ; and connection cleanup following a port failure.  In the VC failure case,
01C1 ; the port is still alive.  As sysap's with connections on the broken vc
01C1 ; are notified and issue disconnects, CDT's are retained in the PB CDT list.
01C1 ; They are retained because queued traffic may still be in the port which will
01C1 ; be completing with appropriate status.  The CDT's are cleaned up after the
01C1 ; last one is disconnected and after the cache clear msg has made it through
01C1 ; the port.
01C1 ;
01C1 ; If the vc is breaking bacause of a port failure, the port is dead and
01C1 ; no further traffic will be processed.  In this case, as sysap's disconnect,
01C1 ; CDT's are cleaned up immediately.  (Implementation note:  this logic
01C1 ; might be simplified overall by handling the simpler port crash case
01C1 ; like the more comples vc failure case.  The two cases probably need to
01C1 ; differ only in their dependency on the cache clear msg.)
01C1 ;
01C1 ; Given the difference in handling, a problem occurs if a port crash
01C1 ; happens in the midst of a vc failure.  The port crash always results in
01C1 ; a call to ERR$PWF_RECOV which forks prior to processing all the path
01C1 ; blocks.  Consequently, the code which notifies all sysap's in the
01C1 ; event of a vc failures is not interrupted by the code in ERR$PWF_RECOV
01C1 ; which processes PB's.  When we arrive at the point of processing each
01C1 ; PB, we are in one of two situations if the PB is in VC_FAIL state:
01C1 ;
01C1 ;          -All CDT'sa re in VC_FAIL state also, and a cache clear has
01C1 ;           been issued which we have just cleaned up of one of the queues.
01C1 ;
01C1 ;          -Some CDT's are in VC_FAIL.  Sysap's have all been notified
01C1 ;           about the rest of the connections, but have not yet disconnected.
01C1 ;
01C1 ; So, if the PB is already in VC_FAIL state, CDT's in VC_FAIL state are
01C1 ; closed out after completing the pending disconnect calls.  If no CDT's
01C1 ; remain after this, PB (and SB) are also deleted and port reinit may
```

```
                    01C1   ; be attempted.  If some CDT's remain,then place the PB in the PWR_FAIL
                    01C1   ; state so that the remaining disconnects behave properly (like a port
                    01C1   ; failure rather than a vc failure.)
                    01C1   ;
                    01C1   ; Inputs:
                    01C1   ;
                    01C1   ;       IPL                         -IPL$_POWER, device IPL
                    01C1   ;
                    01C1   ;       R1                          -Aux status to report to SYSAP:
                    01C1   ;
                    01C1   ;                                    SS$_POWERFAIL if called by unit init
                    01C1   ;                                    following CPU pwr fail recovery;
                    01C1   ;
                    01C1   ;                                    SS$_POWERFAIL if called by int service
                    01C1   ;                                    on port pwr down;
                    01C1   ;
                    01C1   ;                                    SS$_ABORT if called by int service or
                    01C1   ;                                    ERR$CRASH_PORT with error necessitating
                    01C1   ;                                    reinit of port (buffers cached by port lost);
                    01C1   ;
                    01C1   ;                                    SS$_CTRLERR if called by int service or
                    01C1   ;                                    ERR$CRASH_PORT with error necessitating
                    01C1   ;                                    reinit of port, but no retries are left
                    01C1   ;                                    so that the port will remain shutdown
                    01C1   ;                                    (buffers cached by port lost).
                    01C1   ;
                    01C1   ;       R5                          -UCB 0 addr
                    01C1   ;
                    01C1   ;       Port state                  -Uninitialized/maint;  PDT/PQB
                    01C1   ;                                    logout area contains a list of
                    01C1   ;                                    port cached entries.
                    01C1   ;
                    01C1   ;       PDT$W_LPORT_STS             -PWF_CLNUP set to show powerfail
                    01C1   ;                                    cleanup in progress.
                    01C1   ;                                    PUP set if called from system
                    01C1   ;                                    powerfail recovery to show power up.
                    01C1   ;                                    PUP clear if called from port interrupt
                    01C1   ;                                    on power down to show power not
                    01C1   ;                                    recovered yet.
                    01C1   ;
                    01C1   ;       (SP)                        -Return to caller in unit initialization
                    01C1   ;                                    or interrupt service.
                    01C1   ;
                    01C1   ; Outputs:
                    01C1   ;
                    01C1   ;       IPL                         -IPL --> IPL$_SCS and return taken to
                    01C1   ;                                    unit init;  The unit is set offline
                    01C1   ;                                    and registers preserved on return to
                    01C1   ;                                    unit init.
                    01C1   ;-
                    01C1
                    01C1           .ENABL  LSB
                    01C1
                    01C1   ERR$PWF_RECOV::
                    01C1
      10     AA     01C1           BICW    #UCB$M_ONLINE,-          ; Set unit offline to show
   64 A5            01C3                   UCB$W_STS(R5)            ;  that it's uninitialized
```

```
                              01C5
      54   0084 C5    D0      01C5          MOVL    UCB$L_PDT(R5),R4            ; Get PDT addr
      53   01E0 C4    DE      01CA          MOVAL   PDT$Q_COMQBASE(R4),R3       ; Get addr of 1st command queue hdr
                 52   D4      01CF          CLRL    R2                          ; Zero count of command + rsp queues
                              01D1
                              01D1     10$:
               00C1   30      01D1          BSBW    UNLOCK_BADQ                 ; Unlock and handle bad queue
           53    08   C0      01D4          ADDL    #8,R3                       ; Step to next queue hdr
                 04   F3      01D7          AOBLEQ  #<<PDT$Q_RSPQ - PDT$Q_COMQBASE>/8>,-
              F6 52           01D9                  R2,10$                      ; Branch if more queues to check
      53   020C C4    D0      01DB          MOVL    PDT$L_MFQHDR(R4),R3         ; Get addr of free msg queue hdr
               00B2   30      01E0          BSBW    UNLOCK_BADQ                 ; Check it
      53   0208 C4    D0      01E3          MOVL    PDT$L_DFQHDR(R4),R3         ; Get addr of free dg queue hdr
               00AA   30      01E8          BSBW    UNLOCK_BADQ                 ; Check it
                              01EB
           54    51   D0      01EB          MOVL    R1,R4                       ; Copy aux status to reg preserved
                              01EE                                              ; that will be reserved over fork
53    000001F8'EF     9E      01EE          MOVAB   15$,R3                      ; Address of where to resume after fork
               FE08'  31      01F5          BRW     INI$FORK                    ; Fork...
                              01F8
                              01F8     ;
                              01F8     ; Clean up formative path and system blocks on this PDT. From this point on
                              01F8     ; we are at fork IPL.
                              01F8     ;
                              01F8
      54   0084 C5    D0      01F8     15$: MOVL    UCB$L_PDT(R5),R4            ; Restore PDT address
      52   0174 C4    7E      01FD          MOVAQ   PDT$Q_FORMPB(R4),R2         ; Get addr of formative PB
                              0202                                              ;  listhead
           53    62   D0      0202          MOVL    (R2),R3                     ; Get next formative PB
                              0205
           52    53   D1      0205     20$: CMPL    R3,R2                       ; Back at listhead?
                 21   13      0208          BEQL    50$                         ; Branch if so
        50    30 A3   D0      020A          MOVL    PB$L_SBLINK(R3),R0          ; Else get formative SB
                 06   13      020E          BEQL    30$                         ; Branch if no SB
      00000000'GF     16      0210          JSB     G^COM$DRVDEALMEM            ; Else deallocate SB to pool
                              0216
              0C A3   E5      0216     30$: BBCC    PB$B_RSTATION(R3),-         ; Turn off known port bit in
         00 0114 C4           0219                  PDT$B_PORTMAP(R4),40$       ;  bitmap
           50    53   D0      021D     40$: MOVL    R3,R0                       ; Copy PB addr for deallocator
           53    63   D0      0220          MOVL    (R3),R3                     ; Get address of next formative PB
      00000000'GF     16      0223          JSB     G^COM$DRVDEALMEM            ; Deallocate PB to pool
                 DA   11      0229          BRB     20$                         ; Go for next formative PB
                              022B
           62    52   D0      022B     50$: MOVL    R2,(R2)                     ; Set formative pathblock
        04 A2    52   D0      022E          MOVL    R2,4(R2)                    ;  to empty
                              0232
                              0232     ;
                              0232     ; Remove all packets from port command queues, response queue,
                              0232     ; free queues, and the logout area.  All packets are returned to
                              0232     ; pool except send datagrams which are flagged 'return to sysap.'
                              0232     ; These are returned to the SYSAP just as if they had gone out
                              0232     ; normally.
                              0232     ;
                              0232
               00A7   30      0232          BSBW    ERR$CLEANUP_PKT             ; Call packet cleanup routine
                              0235
                              0235     ;
```

```
                        0235    ; Clean up fully open paths and system blocks on this PDT:
                        0235    ;
                        0235
          FDC8'   30    0235            BSBW    CNF$LKP_PB_PDT          ; Look up 1st/next PB
                        0238                                            ; Start of coroutine if PB found:
           48 50  E9    0238            BLBC    R0,115$                 ; Branch if no more PB's
         8000 8F  B1    023B            CMPW    #PB$C_VC_FAIL,-         ; Is PB already cleaning up a
            12 A3       023F                    PB$W_STATE(R3)          ;  vc failure?
              32  12    0241            BNEQ    100$                    ; Branch if not
                        0243
    50     C8 A3  DE    0243    60$:    MOVAL   PB$L_CDTLST-CDT$L_CDTLST(R3),R0
                        0247                                            ; Else set to scan all CDT's on PB
                        0247
    50     6C A0  D0    0247    70$:    MOVL    CDT$L_CDTLST(R0),R0     ; Get next CDT
                        024B
              1C  13    024B    80$:    BEQL    90$                     ; Branch if no more
           28 A0  B1    024D            CMPW    CDT$W_STATE(R0),-       ; SYSAP finished with connection?
              0C        0250                    #CDT$C_VC_FAIL          ;  (I.e., disconnect issued?)
              F4  12    0251            BNEQ    70$                     ; Branch if not
           6C A0  DD    0253            PUSHL   CDT$L_CDTLST(R0)        ; Save pointer to next CDT
              53  DD    0256            PUSHL   R3                      ; Save PB addr
    53        50  D0    0258            MOVL    R0,R3                   ; Put current CDT addr in standard reg
    50        01  3C    025B            MOVZWL  #SS$_NORMAL,R0          ; Set status = success
          FD9F'   30    025E            BSBW    SCS$CLOSE_CDT           ; Complete SYSAP's pending disconnect
                        0261                                            ;  call and deallocate CDT
          53 8ED0       0261            POPL    R3                      ; Retreive PB address
          50 8ED0       0264            POPL    R0                      ;  and addr of following CDT
              E2  11    0267            BRB     80$                     ; Process next CDT, if any
                        0269
         4000 8F  B0    0269    90$:    MOVW    #PB$C_PWR_FAIL,-        ; Change PB state to power fail
            12 A3       026D                    PB$W_STATE(R3)          ;  recovery/port failure in progress
            34 A3 D5    026F            TSTL    PB$L_CDTLST(R3)         ; All CDT's gone?
              0C  13    0272            BEQL    110$                    ; Branch if so
                  05    0274            RSB                             ; Else done -- remaining CDT's
                        0275                                            ;  will be cleaned up via disconnect calls
                        0275
         4000 8F  B0    0275    100$:   MOVW    #PB$C_PWR_FAIL,-        ; Set PB state to pwr fail
            12 A3       0279                    PB$W_STATE(R3)          ;  in progress
            34 A3 D5    027B            TSTL    PB$L_CDTLST(R3)         ; Does this PB have any connections?
              0D  12    027E            BNEQ    120$                    ; Branch if so
                        0280
          FD7D'   30    0280    110$:   BSBW    CNF$REMOVE_PB           ; Else kill of this PB
                        0283
         0112 C4  B5    0283    115$:   TSTW    PDT$W_PBCOUNT(R4)       ; Any PB's left on this PDT?
              0B  12    0287            BNEQ    130$                    ; Branch if so, can't clean up port
            00C9  30    0289            BSBW    ERR$INIPORT             ; Try port hardware init
                  05    028C            RSB                             ; Continue PB search
                        028D
    50     14 A5  3C    028D    120$:   MOVZWL  UCB$L_FR4(R5),R0        ; Set status info for SYSAP err routine
          FD6C'   30    0291            BSBW    SCS$NOTIFY_SYSAP        ; Handle all CDT's in list
                        0294
                  05    0294    130$:   RSB                             ; Return
                        0295
                        0295            .DSABL  LSB
```

```
                      0295          .SBTTL  UNLOCK_BADQ,            ZERO CORRUPTED QUEUE HDRS
                      0295
                      0295  ;+
                      0295  ; UNLOCK_BADQ tests the interlock bit on the queue pointed to by
                      0295  ; R3.  If the queue is locked, it is presumed corrupted and the header
                      0295  ; zeroed so that any entries that should be in the queue are permanently
                      0295  ; lost.  The interlock bit should never be set in a power fail situation.
                      0295  ; If it is, the auxiliary status in R1 is changed from SS$_POWERFAIL to
                      0295  ; SS$_ABORT.  If the queue is not locked, then return is taken without
                      0295  ; altering anything -- the queue is purged later by ERR$CLEANUP_PKT.
                      0295  ;
                      0295  ; Inputs:
                      0295  ;
                      0295  ;       R1                      -Aux status to pass to SYSAPs
                      0295  ;       R3                      -Addr of queue header to check
                      0295  ;
                      0295  ; Outputs:
                      0295  ;
                      0295  ;       All registers           -Preserved
                      0295  ;-
                      0295
                      0295  UNLOCK_BADQ:
                      0295
   0C 63   00   E1    0295          BBC     #0,(R3),Q_UNLOCKED      ; Branch if queue unlocked
           63   7C    0299          CLRQ    (R3)                    ; Else clear header
 0364 8F   51   B1    029B          CMPW    R1,#SS$_POWERFAIL       ; Is this power failure?
           03   12    02A0          BNEQ    Q_UNLOCKED              ; Branch if not
      51   2C   3C    02A2          MOVZWL  #SS$_ABORT,R1           ; Else change status to show
                      02A5                                          ;  SYSAPs that pkts are lost
                      02A5
                      02A5  Q_UNLOCKED:
                      02A5
                05    02A5          RSB                             ; Return
                      02A6
                      02A6
                      02A6          .DSABL  LSB
```

```
                    02A6             .SBTTL  ERR$DISC_PWFAIL,         PROCESS DISCONNECT CALL
                    02A6             .SBTTL  -                        FOR CDT ON POWER
                    02A6             .SBTTL  -                        FAILED PORT
                    02A6
                    02A6     ;+
                    02A6     ; ERR$DISC_PWFAIL is called by FPC$DCONNECT when the SYSAP issues
                    02A6     ; a DISCONNECT for a connection associated with a power failed port.
                    02A6     ; (Path block state = PB$C_PWR_FAIL.)  In this case the local
                    02A6     ; port is nonfunctional and action is to deallocate CDTs as they
                    02A6     ; are DISCONNECTed after purging out the command queues of any SEND's
                    02A6     ; the SYSAP may have done since being notified at its error entry.
                    02A6     ;
                    02A6     ; If this is the last CDT on this path block, the path block (and
                    02A6     ; system block) is removed and an attempt made to reinit the
                    02A6     ; port hardware.
                    02A6     ;
                    02A6     ; Inputs:
                    02A6     ;
                    02A6     ;       IPL                             -Fork IPL
                    02A6     ;
                    02A6     ;       R1                              -Addr of PB
                    02A6     ;       R3                              -Addr of CDT being DISCONNECTed
                    02A6     ;       R4                              -Addr of PDT
                    02A6     ;
                    02A6     ;       CDT$W_STATE                     -Any except CLOSED or VC_FAIL
                    02A6     ;
                    02A6     ;       (SP)                            -Addr of return to FPC$DCONNECT
                    02A6     ;
                    02A6     ; Outputs:
                    02A6     ;
                    02A6     ;       R0-R3                           -Destroyed
                    02A6     ;       Other registers                 -Preserved
                    02A6     ;-
                    02A6
                    02A6             .ENABL  LSB
                    02A6
                    02A6     ERR$DISC_PWFAIL::
                    02A6
          51    DD  02A6             PUSHL   R1                      ; Save PB addr
       28 A3    B1  02A8             CMPW    CDT$W_STATE(R3),-       ; Is this a listener with a
          09        02AB                     #CDT$C_CON_REC          ;  connect in hand?
          05    12  02AC             BNEQ    10$                     ; Branch if not
       FD4F'    30  02AE             BSBW    SCS$FREE_LISTEN         ; Else just put it back to listening
          11    11  02B1             BRB     20$                     ; Join common check for no more CDT's
                    02B3
          53    DD  02B3     10$:    PUSHL   R3                      ; Save CDT addr
        0024    30  02B5             BSBW    ERR$CLEANUP_PKT         ; Purge out the command queues
                    02B8                                             ;  again in case SYSAP error routine
                    02B8                                             ;  did any more SENDs
        53 8ED0     02B8             POPL    R3                      ; Restore CDT addr
       FD42'    30  02BB             BSBW    SCS$DEAL_SCSREC         ; Deallocate CDT's SCS recv buffer
 00000000'GF    16  02BE             JSB     G^SCS$DEALL_CDT         ; Deallocate CDT
                    02C4
        53 8ED0     02C4     20$:    POPL    R3                      ; Retrieve PB addr in R3
       34 A3    D5  02C7             TSTL    PB$L_CDTLST(R3)         ; Any CDT's left on PB?
          0C    12  02CA             BNEQ    30$                     ; Branch if so
       FD31'    30  02CC             BSBW    CNF$REMOVE_PB           ; Else deallocate PB/SB
```

```
        0112 C4   B5  02CF   1070          TSTW    PDT$W_PBCOUNT(R4)          ; Any PB's left on this PDT?
              03   12  02D3   1071          BNEQ    30$                        ; Branch if some left
            007D   30  02D5   1072          BSBW    ERR$INIPORT                ; Try to init port hardware now
                       02D8   1073
        50    01   3C  02D8   1074 30$:     MOVZWL  #SS$_NORMAL,R0             ; Set to return success to SYSAP
                   05  02DB   1075          RSB                                ;
                       02DC   1076
                       02DC   1077          .DSABL  LSB
```

```
                        02DC          .SBTTL  ERR$CLEANUP_PKT           CLEAN UP PACKETS QUEUED TO
                        02DC          .SBTTL  -                         PORT AND IN LOGOUT AREA
                        02DC
                        02DC    ;+
                        02DC    ; ERR$CLEANUP_PKT calls FLUSH_Q to remove and dispose of packets currently
                        02DC    ; on each of the port queues.  It hen extracts each packet address
                        02DC    ; recorded in the logout area and calls ERR$DISP_ENTRY to dispose of the
                        02DC    ; entry.  The rule for disposing of packets is to return all packets
                        02DC    ; to pool except send datagrams flagged as 'return to sysap.'  These
                        02DC    ; are returned to the SYSAP.
                        02DC    ;
                        02DC    ; Inputs:
                        02DC    ;
                        02DC    ;         R4                            -PDT addr
                        02DC    ;
                        02DC    ; Outputs:
                        02DC    ;
                        02DC    ;         R0-R3                         -Destroyed
                        02DC    ;-
                        02DC
                        02DC    ASSUME  PDT$Q_COMQBASE  EQ  PDT$Q_COMQL
                        02DC    ASSUME  PDT$Q_COMQL+8   EQ  PDT$Q_COMQH
                        02DC    ASSUME  PDT$Q_COMQH+8   EQ  PDT$Q_COMQ2
                        02DC    ASSUME  PDT$Q_COMQ2+8   EQ  PDT$Q_COMQ3
                        02DC    ASSUME  PDT$Q_COMQ3+8   EQ  PDT$Q_RSPQ
                        02DC
                        02DC          .ENABL  LSB
                        02DC
                        02DC    ERR$CLEANUP_PKT::
                        02DC
       51   01E0 C4  DE 02DC          MOVAL   PDT$Q_COMQBASE(R4),R1     ; Get adr of 1st command queue
            53   05  D0 02E1          MOVL    #<<PDT$Q_RSPQ - PDT$Q_COMQBASE>/8 + 1>,R3
                        02E4                                            ; Get count of command/rsp queues
                        02E4
                 33  10 02E4    10$:  BSBB    FLUSH_Q                   ; Purge next queue of all entries
            51   08  C0 02E6          ADDL    #8,R1                     ; Step to next queue to flush
            F8   53  F5 02E9          SOBGTR  R3,10$                    ; Branch if more queues
       51   020C C4  D0 02EC          MOVL    PDT$L_MFQHDR(R4),R1       ; Get addr of msg free queue header
                 26  10 02F1          BSBB    FLUSH_Q                   ; Purge all entries
       51   0208 C4  D0 02F3          MOVL    PDT$L_DFQHDR(R4),R1       ; Get addr of dg free queue header
                 1F  10 02F8          BSBB    FLUSH_Q                   ; Purge all entries
       53   02E0 C4  DE 02FA          MOVAL   PDT$L_DQELOGOUT(R4),R3    ; Get base of logout area
            51   20  D0 02FF          MOVL    #<<PDT$C_PALENGTH - PDT$L_DQELOGOUT>/4>,R1
                        0302                                            ; Get count of elmts in logout area
                        0302
            52   83  D0 0302    20$:  MOVL    (R3)+,R2                  ; Get addr of next entry
  FFFFFFFF 8F    52  D1 0305          CMPL    R2,#-1                    ; Port record anything here?
                 07  13 030C          BEQL    30$                       ; Branch if not
               0021  30 030E          BSBW    ERR$DISP_ENTRY            ; Else dispose of entry
         FC A3   00  D2 0311          MCOML   #0,-4(R3)                 ; Reset entry just processed
                        0315
            EA   51  F5 0315    30$:  SOBGTR  R1,20$                    ; Branch if more entries in logout area
                     05 0318          RSB                               ; Return
                        0319
                        0319          .DSABL  LSB
                        0319
                        0319
```

```
                0319            .SBTTL  FLUSH_Q                       REMOVE AND DISPOSE OF
                0319            .SBTTL  -                             ALL QUEUED ENTRIES
                0319            .SBTTL  ERR$DISP_ENTRY                DISPOSE OF A SINGLE ENTRY
                0319
                0319    ;+
                0319    ; FLUSH_Q removes and processes all entries from a specified port queue.
                0319    ;
                0319    ; ERR$DISP_ENTRY processes a removed entry.  All packets are returned to
                0319    ; pool except send datagrams flagged 'return to sysap.'  These are
                0319    ; handled exactly as if they had gone out successfully.
                0319    ;
                0319    ; Inputs:
                0319    ;
                0319    ;       IPL                             -Fork IPL
                0319    ;       R1                              -Addr of queue header (FLUSH_Q)
                0319    ;       R2                              -Pkt addr (ERR$DISP_ENTRY)
                0319    ;
                0319    ; Outputs:
                0319    ;
                0319    ;       R0                              -Destroyed
                0319    ;       R2                              -Destroyed (FLUSH_Q)
                0319    ;       Other registers                 -Preserved
                0319    ;
                0319
                0319            .ENABL  LSB
                0319
                0319    FLUSH_Q:
                0319
                0319            $QRETRY  REMQHI (R1),R2,-      ; Remove next entry from
                0319                     ERROR=FATALQ          ;  queue head
       04  1D   032B            BVS     10$                   ; Branch if no more entries
       03  10   032D            BSBB    ERR$DISP_ENTRY        ; Else dispose of entry
       E8  11   032F            BRB     FLUSH_Q               ; Go for another entry
                0331
           05   0331    10$:    RSB                           ; Return
                0332
                0332
                0332
                0332    ERR$DISP_ENTRY::
                0332
       00  E1   0332            BBC     #PPD$V_RSP,-          ; Anybody expecting pkt?
 0E 0F A2       0334                    PPD$B_FLAGS(R2),20$   ;  Branch if not
    0E A2  91   0337            CMPB    PPD$B_OPC(R2),-       ; Was it a send datagram?
       01       033A                    #PPD$C_SNDDG          ;
       08  12   033B            BNEQ    20$                   ; Branch if not
       2E  BB   033D            PUSHR   #^M<R1,R2,R3,R5>      ; Save registers
    FCBE'  30   033F            BSBW    INT$DISP_SENDDG       ; Else handle as interrupt
       2E  BA   0342            POPR    #^M<R1,R2,R3,R5>      ; Restore destroyed registers
           05   0344            RSB                           ; Return
                0345
    FCB8'  30   0345    20$:    BSBW    INT$DEAL_PKT          ; Return to pool
           05   0348            RSB                           ; Return
                0349
                0349    FATALQ:                               ; Should never get here since
                0349                                          ;  queue lock cleared by UNLOCK_BADQ
                0349            BUGCHECK  CIPORT,NONFATAL     ; Nonfatal bugcheck
                0350
```

```
      61   7C  0350  1194          CLRQ    (R1)                          ; If survive bugcheck, clear queue
               0352  1195                                                ;  header
      8E   D5  0352  1196          TSTL    (SP)+                         ; Clear return from error call
           05  0354  1197          RSB                                   ; Return from FLUSH_Q
               0355  1198
               0355  1199          .DSABL  LSB
```

```
                    0355                .SBTTL  ERR$INIPORT,            CALL PORT HARDWARE INIT
                    0355
                    0355       ;+
                    0355       ; If port has power now, call port initialization routine in PAINIT.
                    0355       ;
                    0355       ; Inputs:
                    0355       ;
                    0355       ;        R4                     -PDT address
                    0355       ;        IPL                    -IPL$_SCS
                    0355       ;
                    0355       ; Outputs:
                    0355       ;
                    0355       ;        R0-R3                  -Destroyed
                    0355       ;        Other registers        -Preserved
                    0355       ;-
                    0355
                    0355                .DSABL  LSB
                    0355
                    0355       ERR$INIPORT::
                    0355
          01   AA   0355                BICW    #PDT$M_PWF_CLNUP,-      ; Show power fail cleanup
     0110 C4        0357                        PDT$W_[PORT_STS(R4)     ;  bookkeeping done
          01   E1   035A                BBC     #PDT$V_PUP,-            ; Has port got power now?
  1C 0110 C4        035C                        PDT$W_[PORT_STS(R4),20$ ;  Branch if not
     7E   54   7D   0360                MOVQ    R4,-(SP)                ; Save PDT addr and R5(
55   00DC C4   D0   0363                MOVL    PDT$L_UCB0(R4),R5       ; Get UCB addr for init
          04   E0   0368                BBS     #UCB$V_ONLINE,-         ; If controller-unit already
    0C 64 A5        036A                        UCB$W_STS(R5),10$       ;  initialized, branch
54   00E4 C4   D0   036D                MOVL    PDT$L_CNF(R4),R4        ; Get config register addr
          01   D0   0372                MOVL    #PA_PMC_M_MIN,-         ; Place port in un-initialized state
       04 A4        0374                        PA_PMC(R4)              ; and disable device interrupts
       FC87'   30   0376                BSBW    INI$PORT                ; Call port initialization
                    0379
     54   8E   7D   0379       10$:     MOVQ    (SP)+,R4                ; Restore our registers
                    037C
               05   037C       20$:     RSB                             ; Return
                    037D
                    037D                .DSABL  LSB
```

```
                               037D        .SBTTL  ERR$BUGCHECK,           RECORD PORT LOCAL STORE
                               037D        .SBTTL  -                       IN MEMORY
                               037D        .SBTTL  ERR$BUGCHECKNF,         RECORD LOCAL STORE CONDITIONALLY
                               037D        .SBTTL  -                       IF NONFATAL BUGCHECKS ARE FATAL
                               037D
                               037D ;+
                               037D ; This routine copies the port local store (device registers, VC
                               037D ; descriptor table, transalation cache, work space, etc.) over the
                               037D ; microcode in pool so that this info will be available in a dump.
                               037D ;
                               037D ; Inputs:
                               037D ;
                               037D ;       R4                      -PDT addr
                               037D ;
                               037D ; Outputs:
                               037D ;
                               037D ;       All registers           -Preserved
                               037D ;-
                               037D
                               037D ASSUME  <^X1000>  LE  PA_C_WCSSIZ*6
                               037D
                               037D        .ENABL  LSB
                               037D
                               037D ERR$BUGCHECKNF::
                               037D
          00000000'8F     E0   037D        BBS     #EXE$V_FATAL_BUG,-      ; Branch if nonfatal bugchecks
          00000000'GF          0383                G^EXE$GL_DEFFLAGS,-     ;  are set to be fatal via
                   14          0388                ERR$BUGCHECK            ;  SYSGEN parameter
                               0389        $DEBUGCHECK #ERR$V_DEB_BUGNF    ; If flag enabled, do a fatal bugcheck
                               039C                                        ;  anyway regardless of SYSGEN param
                          05   039C        RSB                             ; Else return doing nothing
                               039D
                               039D ERR$BUGCHECK::
                               039D
                   0F     BB   039D        PUSHR   #^M<R0,R1,R2,R3>        ; Save caller's registers
        52   00E4 C4      D0   039F        MOVL    PDT$L_CNF(R4),R2        ; Get addr of base of device registers
                               03A4
    53  00000000'GF       D0   03A4 10$:   MOVL    G^SCS$GL_MCADR,R3       ; Get addr of ucode in pool
        51   0400 8F      3C   03AB        MOVZWL  #<^X1000/4>,R1          ; Get size of device register space
                               03B0
              83   82     D0   03B0 20$:   MOVL    (R2)+,(R3)+             ; Copy next long wd of local store
              FA   51     F5   03B3        SOBGTR  R1,20$                  ; Branch if more to copy
                   0F     BA   03B6        POPR    #^M<R0,R1,R2,R3>
                               03B8
                          05   03B8        RSB
                               03B9
                               03B9        .DSABL  LSB
```

```
                     03B9          .SBTTL  ERR$DEBUGCHECK,         DEBUG BUGCHECK ENABLE FLAGS
                     03B9
                     03B9   ;+
                     03B9   ; For the ourpose of tracing intermittant problems that we normally
                     03B9   ; attempt recovery from, a number of CIPORT bugchecks have been added.
                     03B9   ; Each bugcheck is enabled or disabled by a separate flag.  Flags
                     03B9   ; may be turned on or off by a quick patch to location ERR$DEBUGCHECK.
                     03B9   ;-
                     03B9
                     03B9   ;
                     03B9   ; Flags are stored in the following longword:
                     03B9   ;
                     03B9
                     03B9   ERR$DEBUGCHECK::
                     03B9
           00000000  03B9          .LONG   ^X0                     ; The default is all bugchecks
                     03BD                                          ;  are disabled, and recovery enabled
                     03BD
                     03BD
                     03BD   ;
                     03BD   ; Flag definitions by bit number:
                     03BD   ;
                     03BD
           00000000  03BD   ERR$V_DEB_INVBN == 0                   ; Invalid buffer name during blk
                     03BD                                          ;  xfer -- normally crashes port
                     03BD
           00000001  03BD   ERR$V_DEB_BLV   == 1                   ; Local buffer length violation --
                     03BD                                          ;  normally crashes port
                     03BD
           00000002  03BD   ERR$V_DEB_ACCV  == 2                   ; Access violation during blk
                     03BD                                          ;  xfer -- normally crashes port
                     03BD
           00000003  03BD   ERR$V_DEB_PSV   == 3                   ; Packet size violation --
                     03BD                                          ;  normally crashes port
                     03BD
           00000005  03BD   ERR$V_DEB_URP   == 5                   ; Unrecognized packet --
                     03BD                                          ;  normally logged and discarded
                     03BD
           00000006  03BD   ERR$V_DEB_INVDP == 6                   ; Invalid destination port number --
                     03BD                                          ;  normally crashes port
                     03BD
           00000007  03BD   ERR$V_DEB_URC   == 7                   ; Unrecognized local command --
                     03BD                                          ;  normally crashes port
                     03BD
           00000008  03BD   ERR$V_DEB_ABO   == 8                   ; Aborted command (port disabled) --
                     03BD                                          ;  normally crashes port
                     03BD
           00000009  03BD   ERR$V_DEB_NPUPD == 9                   ; No path + SNDMSG + unrecognized
                     03BD                                          ;  PPD type -- normally crashes port
                     03BD
           0000000A  03BD   ERR$V_DEB_VCUPD ==10                   ; VC closed + SNDMSG + unrecognized
                     03BD                                          ;  PPD type -- normally crashes port
                     03BD
           0000000B  03BD   ERR$V_DEB_INVOP ==11                   ; Invalid opcode in response --
                     03BD                                          ;  normally crashes port
                     03BD
           0000000C  03BD   ERR$V_DEB_UNSTS ==12                   ; Undefined status subtype in response --
```

```
          03BD                                        ;  normally crashes port
          03BD
0000000D  03BD  ERR$V_DEB_NOSTS ==13                  ; Unrecognized combination of status,
          03BD                                        ;  opcode, and PPD type --
          03BD                                        ;  normally crashes port
0000000E  03BD  ERR$V_DEB_XCTER ==14                  ; XCT_ID sequence number check fails
          03BD                                        ;  on DATREC/CNFREC.  Normally crashes
          03BD                                        ;  port
0000000F  03BD  ERR$V_DEB_SCERR ==15                  ; Source connection ID check fails --
          03BD                                        ;  normally crashes port on MSGSNT
          03BD                                        ;  and is ignored on DGSNT
00000010  03BD  ERR$V_DEB_NOPB  ==16                  ; Rec'd connect request with no PB --
          03BD                                        ;  normally crashes port
          03BD
00000011  03BD  ERR$V_DEB_CNFER ==17                  ; Entered VC cleanup with no PB --
          03BD                                        ;  normally crashes port
          03BD
00000012  03BD  ERR$V_DEB_ILKQ  ==18                  ; Interlock queue failure --
          03BD                                        ;  normally crashes port
          03BD
00000013  03BD  ERR$V_DEB_NEPQ  ==19                  ; Reiniting port with non empty
          03BD                                        ;  command/response queues --
          03BD                                        ;  normally logged and recovered
00000014  03BD  ERR$V_DEB_BUGNF ==20                  ; Nonfatal bugcheck being logged --
          03BD                                        ;  normally continues
          03BD
00000015  03BD  ERR$V_DEB_PSRX  ==21                  ; Undefined bits in PSR set --
          03BD                                        ;  normally crashes port
          03BD
00000016  03BD  ERR$V_DEB_OSEQ  ==22                  ; Port received response with
          03BD                                        ;  sequence number mismatch.  This
          03BD                                        ;  is either a legitimate discard
          03BD                                        ;  due to duplicate, or a sequence
          03BD                                        ;  number error.  Software normally
          03BD                                        ;  crashes the vc.
          03BD
00000017  03BD  ERR$V_DEB_VCDCL ==23                  ; Port received sequenced message
          03BD                                        ;  with VCD status set to closed.
          03BD                                        ;  Software normally crashes the
          03BD                                        ;  vc.
          03BD
00000018  03BD  ERR$V_DEB_MFQE ==24                   ; Port detected msg free queue
          03BD                                        ;  empty.
          03BD                                        ;  Normally, port crashes.
          03BD
          03BD
```

```
03BD            .SBTTL  ELOG$INIT_SWERR,        LOG SOFTWARE ERROR
03BD            .SBTTL  -                       ENCOUNTERED DURING
03BD            .SBTTL  -                       PORT INITIALIZATION
03BD            .SBTTL  ELOG$UCODE_NORD,        LOG MICROCODE NOT
03BD            .SBTTL  -                       PROPERLY READ BACK
03BD            .SBTTL  -                       ERROR
03BD            .SBTTL  ELOG$HARDWARE,          LOG HARDWARE ERROR
03BD            .SBTTL  ELOG$Q_INTRLOCK,        LOG QUEUE INTERLOCK
03BD            .SBTTL  -                       FAILURE
03BD
03BD    ;+
03BD    ; These routines log those errors which use the device attention, EMB$C_DA,
03BD    ; error-log-entry format.  There are three such error types:
03BD    ;       - Software errors detected during port initialization.
03BD    ;       - Microcode failed to read-back as loaded (this is logged as a special
03BD    ;         type hardware error).
03BD    ;       - CPU or port ucode not at adequate rev level.
03BD    ;       - Hardware error (typical to but more extensive than those found
03BD    ;         in more standard I/O devices).
03BD    ;       - Failures to obtain access to a queue because of its interlock.
03BD    ;
03BD    ; After some entry specific processing, the body of this routine calls OPA0_LOG
03BD    ; to broadcast the error to _OPA0, if indicated, and then uses ERL$DEVICEATTN
03BD    ; to log the error.  ERL$DEVICEATTN will call ELOG$REGDUMP which will actually
03BD    ; copy the appropriate information into the error log.
03BD    ;
03BD    ;
03BD    ; ELOG$INIT_SWERR:
03BD    ;
03BD    ; Inputs:
03BD    ;       R0                      - Error subtype code in bits 0 through 7
03BD    ;                                 Sign bit set indicates that the error will crash port
03BD    ;                                 Sign bit not set indicates that it will not
03BD    ;       R5                      - Address of device UCB
03BD    ;
03BD    ;
03BD    ; ELOG$UCODE_NORD:
03BD    ;
03BD    ; Inputs:
03BD    ;       R0                      - Correct microcode value
03BD    ;       R4                      - Base virtual address of CI port registers
03BD    ;       R5                      - Address of device UCB
03BD    ;
03BD    ;
03BD    ; ELOG$CPU_REV:
03BD    ;
03BD    ; Inputs:
03BD    ;       R1                      -System ID Register which contains CPU rev level
03BD    ;       R5                      -UCB addr
03BD    ;
03BD    ; ELOG$UCODE_ERR, ELOG$UCODE_WARN:
03BD    ;
03BD    ; Inputs:
03BD    ;       R2                      -Addr of IDREC pkt containing port ucode rev
03BD    ;                                level at offset PPD$L_RPORT_REV
03BD    ;       R5                      -UCB addr
03BD    ;
```

```
                   03BD  ; ELOG$HARDWARE:
                   03BD  ;
                   03BD  ; Inputs:
                   03BD  ;       R0                - Error subtype code in bits 0 through 7
                   03BD  ;                           Sign bit set indicates that the error will crash port
                   03BD  ;                           Sign bit not set indicates that it will not
                   03BD  ;       R4                - Base virtual address of CI port registers
                   03BD  ;       R5                - Address of device UCB
                   03BD  ;
                   03BD  ;
                   03BD  ; ELOG$Q_INTRLOCK:
                   03BD  ;
                   03BD  ; Inputs:
                   03BD  ;       R0                - Error subtype code in bits 0 through 7
                   03BD  ;                           Sign bit set indicates that the error will crash port
                   03BD  ;                           Sign bit not set indicates that it will not
                   03BD  ;       R4                - Address of PDT
                   03BD  ;
                   03BD  ;
                   03BD  ; ALL ROUTINES:
                   03BD  ;
                   03BD  ; Outputs:
                   03BD  ;       R0 is destroyed.  All other registers are preserved.  An entry is made
                   03BD  ;       in the error log. The existance of this error might have been broadcast
                   03BD  ;       to _OPA0.
                   03BD  ;
                   03BD  ;
                   03BD  ; SPECIAL NOTES:
                   03BD  ;
                   03BD  ;    Proper operation of this routine, and ELOG$REGDUMP, depends upon
                   03BD  ;    ERL$DEVICEATTN passing R4 and R5 unaltered to ELOG$REGDUMP.  As of this
                   03BD  ;    routines writing, this was the case.
                   03BD  ;-
                   03BD
                   03BD  ;+
                   03BD  ; The following are various values related to or controlling the size of a
                   03BD  ; device attention error log entry for this device.
                   03BD  ;-
                   03BD
        00000006   03BD  PORT_REGS_LOGGED = 6                       ; Number of port registers logged
        00000003   03BD  NUM_EX_LONGWORDS = 3                       ; Number of extra longwords logged
                   03BD  TOTAL_LONGWORDS  =  2 -                    ; Longword count + error type code
                                   + PORT_REGS_LOGGED -             ; + port registers
        0000000B   03BD            + NUM_EX_LONGWORDS               ; + extra longwords
                   03BD
                   03BD  ELOG$K_BYTES == <TOTAL_LONGWORDS * 4> -    ; This is the number of bytes in a
        0000007A   03BD            + EMB$C_DV_REGSAV                ; device attention error log entry
                   03BD                                             ; from the CI as entered in the DDT.
                   03BD
                   03BD          .MACRO  ZERO_EXTRA_LONGWORDS
                   03BD          ASSUME  NUM_EX_LONGWORDS EQ 3
                   03BD          CLRQ    -(SP)
                   03BD          CLRL    -(SP)
                   03BD          .ENDM   ZERO_EXTRA_LONGWORDS
                   03BD
        0000003E   03BD  DA_MASK = ^M<R1,R2,R3,R4,R5>
                   03BD
```

```
                          03BD ELOG$INIT_SWERR::                                  ; Software error during initialization
                          03BD
               3E   BB    03BD          PUSHR   #DA_MASK                          ; Save registers.
               54   D4    03BF          CLRL    R4                                ; Zero port base VA implying don't log
                          03C1                                                    ; port registers.
                          03C1          ZERO_EXTRA_LONGWORDS                      ; No extra longword to log here.
                          03C5          ASSUME  PAER$K_ET_INSW EQ 0
               7E   94    03C5          CLRB    -(SP)                             ; Build error type part of error code.
               72   11    03C7          BRB     ELOG$$LOG_DA                      ; Branch to common code.
                          03C9
                          03C9
                          03C9 ELOG$UCODE_NORD::
                          03C9
               3E   BB    03C9          PUSHR   #DA_MASK                          ; Save registers.
                          03CB          ASSUME  NUM_EX_LONGWORDS EQ 3
               50   DD    03CB          PUSHL   R0                                ; Ex. lw. 3 = correct ucode value.
               7E   7C    03CD          CLRQ    -(SP)                             ; Init ex. lw. 1 & 2 to zero.
         55    5E   D0    03CF          MOVL    SP, R5                            ; Save current stack pointer.
                          03D2          $PRTCTINI -                               ; Protect the following device register
                          03D2                  B^10$, #MCHK$M_NEXM               ; references from machine checks.
   04 A5     18 A4  D0    03DE          MOVL    PA_MDATR(R4),4(R5)                ; Ex. lw. 2 = wrong   ucode value.
      65     14 A4  D0    03E3          MOVL    PA_MADR(R4),(R5)                  ; Ex. lw. 1 = failing ucode address.
                          03E7          $PRTCTEND 10$                             ; If check occurs, leave zero values(s).
      55     40 AE  D0    03E8          MOVL    3*4+4+4*4(SP),R5                  ; Restore previously saved UCB addr.
   50      8000 8F  32    03EC          CVTWL   #<PAER$K_ES_UCDW ! ^X8000>, -     ; Plant error subtype
                          03F1                  R0                                ; w/ crash port code.
               2D   11    03F1          BRB     LOG_AS_HARDWARE                   ; Branch to common hardware error
                          03F3                                                    ; logging code.
                          03F3
                          03F3 ELOG$CPU_REV::
                          03F3
               3E   BB    03F3          PUSHR   #DA_MASK                          ; Save registers
                          03F5          ASSUME  NUM_EX_LONGWORDS EQ 3
               51   DD    03F5          PUSHL   R1                                ; 1st extra longwd gets CPU SID
          8007 8F   32    03F7          CVTWL   #<PAER$K_ES_CPUREV ! ^X8000>,-
               50         03FB                  R0                                ; Set error subtype, port shutting down
               16   11    03FC          BRB     REV_ERROR                         ; Join common rev error logging
                          03FE
                          03FE
                          03FE ELOG$UCODE_ERR::
                          03FE
               3E   BB    03FE          PUSHR   #DA_MASK                          ; Save registers
          8006 8F   32    0400          CVTWL   #<PAER$K_ES_REVER ! ^X8000>,-
               50         0404                  R0                                ; Set error subtype, port shuts down
               05   11    0405          BRB     PORT_UCODE                        ; Join common port rev error logging
                          0407
                          0407
                          0407 ELOG$UCODE_WARN::
                          0407
               3E   BB    0407          PUSHR   #DA_MASK                          ; Save registers
         50    08   9A    0409          MOVZBL  #PAER$K_ES_REVCA,R0               ; Set error subtype, non fatal to port
                          040C
                          040C PORT_UCODE:
                          040C
                          040C          ASSUME  NUM_EX_LONGWORDS EQ 3
            1C A2   DD    040C          PUSHL   PPD$L_RPORT_REV(R2)               ; 1st extra longwd gets port rev level
    00B8 C5    6E   D0    040F          MOVL    (SP),UCB$T_OPA0_TEMP(R5)          ; Save rev level to format in opa0 msg
```

```
                                  0414
                                  0414  REV_ERROR:
                                  0414
                    7E      7C    0414            CLRQ    -(SP)                       ; 2nd and 3rd longwds not used
                    54      D4    0416            CLRL    R4                          ; Zero port CNF addr to avoid logging
                                  0418                                                ;  device registers
                    06      11    0418            BRB     LOG_AS_HARDWARE             ; Join common HW type error logging
                                  041A
                                  041A  ELOG$HARDWARE::
                                  041A
                    3E      BB    041A            PUSHR   #DA_MASK                    ; Save registers.
                                  041C            ZERO_EXTRA_LONGWORDS                ; No extra longword to log here.
                                  0420  LOG_AS_HARDWARE:
              7E    01      90    0420            MOVB    #PAER$K_ET_HW, -(SP)        ; Build error type part of error code.
                    16      11    0423            BRB     ELOG$$LOG_DA                ; Branch to common code.
                                  0425
                                  0425
                                  0425  ELOG$INTRLOCK::
                                  0425
                    3E      BB    0425            PUSHR   #DA_MASK                    ; Save registers.
         55    00DC C4      D0    0427            MOVL    PDT$L_UCB0(R4), R5          ; Obtain UCB address.
           54    24 A5      D0    042C            MOVL    UCB$L_CRB(R5), R4           ; Get base VA of port regieters via
                                  0430            ASSUME  IDB$L_CSR EQ 0              ; UCB ==> CRB ==> IDB ==> CSR.
           54    2C B4      D0    0430            MOVL    @CRB$L_INTD+VEC$L_IDB(R4), R4
                                  0434            ZERO_EXTRA_LONGWORDS                ; No extra longword to log here.
              7E    02      90    0438            MOVB    #PAER$K_ET_ILCK, -(SP)      ; Build error type part of error code.
                                  043B  ;         BRB     ELOG$$LOG_DA                ; Branch to common code.
                                  043B
              00000014            043B  CLN_BYTES = <NUM_EX_LONGWORDS * 4> + 8        ; Number of bytes to clean from stack
                                  043B
                                  043B  ELOG$$LOG_DA:
                                  043B
                    50      D5    043B            TSTL    R0                          ; Is the port going to be crashed?
                    04      18    043D            BGEQ    10$                         ; Branch if no.  Otherwise,
           6E    80 8F      88    043F            BISB    #PAER$M_CPRT, (SP)          ; set the right bit in error code.
              7E    50      90    0443  10$:      MOVB    R0, -(SP)                   ; Add error subtype to error code.
                    7E      B4    0446            CLRW    -(SP)                       ; Longword align the stack.
              7E    54      D0    0448            MOVL    R4, -(SP)                   ; Save VA of port registers.
                                  044B
                    50      D4    044B            CLRL    R0                          ; Clear register
   50    06 AE    8000 8F   AB    044D            BICW3   #^X8000,6(SP),R0            ; Retrieve error subtype and type
            51    FBB8 CF   9E    0454            MOVAB   DA_OPA0_LOG_TAB,R1          ; Retrieve device attention _OPA0 table
              53    55      D0    0459            MOVL    R5,R3                       ; Move UCB address into proper register
                  023B      30    045C            BSBW    OPA0_LOG                    ; Broadcast error to _OPA0 if indicated
      55    000000A0 8F     C2    045F            SUBL2   #UCB$L_MSGFKBLK,R5          ; Compute UCB address
                                  0466
              54    5E      D0    0466            MOVL    SP, R4                      ; Set pointer needed by ELOG$REGDUMP.
            00000000'GF     16    0469            JSB     G^ERL$DEVICEATTN            ; Perform actual error logging.
           5E    14 AE      9E    046F            MOVAB   CLN_BYTES(SP), SP           ; Clean saved information from stack.
                    3E      BA    0473            POPR    #DA_MASK                    ; Restore saved registers
                            05    0475            RSB                                 ; Return to caller.
                            0476                    .SBTTL  ELOG$REGDUMP,              DEVICE ATTENTION
                            0476                    .SBTTL  -                          REGISTER DUMP ROUTINE
                            0476
                            0476            ;+
                            0476            ; This routine is called by ERL$DEVICEATTN (which is called by ELOG$$LOG_DA)
                            0476            ; to copy the appropriate device registers into an error log entry.
                            0476            ;
                            0476            ; Inputs:
                            0476            ;               R0              - Starting address in error log buffer to be filled
                            0476            ;           00(R4)              - Base virtual address of CI port registers
                            0476            ;           04(R4)              - filler word of zeros
                            0476            ;           06(R4)              - Error code type, crash port, subtype fields
                            0476            ;           08(R4)              - NUM_EX_LONGWORDS of additional data to be logged
                            0476            ;               R5              - Address of the device UCB
                            0476            ;
                            0476            ; Outputs:
                            0476            ;           CI port register values and the additional data are copied to the
                            0476            ;           location(s) pointed to by R0.  R0, R1 and R2 are destroyed.  If for
                            0476            ;           any reason the CI port registers are inaccessable, zeros will be
                            0476            ;           logged for thier values.
                            0476            ;-
                            0476
                            0476
                            0476            ELOG$REGDUMP::
                            0476
          52     50     D0  0476                    MOVL    R0, R2                  ; Copy buffer address to a safe place.
          82     0A     9A  0479                    MOVZBL  #<TOTAL_LONGWORDS - 1>, - ; Insert count of longword
                            047C                            (R2)+                   ; "registers" to be logged.
    82       06 A4      B0  047C                    MOVW    6(R4), (R2)+            ; Insert error retry counts, type,
   82      0080 C5      B0  0480                    MOVW    UCB$B_ERTCNT(R5), (R2)+ ; subtype, and crash port information
                            0485                    ASSUME  PORT_REGS_LOGGED EQ 6   ; to form PADRIVER error code.
                 62     7C  0485                    CLRQ     (R2)                   ; Zero places where CI port registers
              08 A2     7C  0487                    CLRQ     8(R2)                  ; may be copied.
              10 A2     7C  048A                    CLRQ    16(R2)
                            048D                    ASSUME  NUM_EX_LONGWORDS EQ 3
  18 A2      08 A4      7D  048D                    MOVQ     8(R4), 24(R2)          ; Copy extra longwords into
  20 A2      10 A4      D0  0492                    MOVL    16(R4), 32(R2)          ; into error log entry.
        51       64     D0  0497                    MOVL    (R4), R1                ; Obtain base VA of CI port registers.
                 31     13  049A                    BEQL    100$                    ; If zero, don't log registers.
                            049C                    $PRTCTINI -                     ; Protect the following device register
                            049C                            B^10$, MCHK$M_NEXM      ; references from machine checks.
          62     61     D0  04AC                    MOVL    PA_CNF(R1),     (R2)    ; Plant configuration register.
  04 A2      04 A1      D0  04AF                    MOVL    PA_PMC(R1),    4(R2)    ; Plant maintenance control/status reg.
  08 A2    0900 C1      D0  04B4                    MOVL    PA_PS(R1),     8(R2)    ; Plant port status register.
  0C A2    0938 C1      D0  04BA                    MOVL    PA_PFAR(R1), 12(R2)     ; Plant failing address register.
  10 A2    093C C1      D0  04C0                    MOVL    PA_PESR(R1), 16(R2)     ; Plant port error status register.
  14 A2    0940 C1      D0  04C6                    MOVL    PA_PPR(R1),  20(R2)     ; Plant port parameter register.
                            04CC                    $PRTCTEND 10$                   ; End protected code.
                        05  04CD            100$:   RSB                             ; Return to ERL$DEVICEATTN.
```

```
04CE          .SBTTL  ELOG$PACKET,            LOG PACKET RELATED
04CE          .SBTTL  -                       ERROR, GENERAL CASE
04CE          .SBTTL  ELOG$CABLES,            LOG CABLE STATUS
04CE          .SBTTL  -                       CHNAGE, GENERAL CASE
04CE          .SBTTL  ELOG$PTH_ST_CHG         LOG PATH STATUS
04CE          .SBTTL  -                       CHANGE
04CE          .SBTTL  ELOG$CBL_X_CHG          LOG CABLES CROSSED OR
04CE          .SBTTL  -                       NOT CROSSED STATUS
04CE          .SBTTL  -                       CHANGE
04CE          .SBTTL  ELOG$ERROR_DG           LOG ERROR LOG DATAGRAM
04CE
04CE   ;+
04CE   ; These routines log those errors which use the logged message, EMB$C_LM,
04CE   ; error-log-entry format.  All such errors result from detection of an
04CE   ; exceptional condition in a data packet.  The error log entry produced by
04CE   ; these routines will include upto 72 bytes of the packet which signaled the
04CE   ; exceptional condition starting with the 12th byte of the packet.
04CE   ;
04CE   ; There is one exceptional case, and that is when what is being logged is the
04CE   ; refusal of the local system to open up a virtual circuit to a remote system
04CE   ; because the information provided by the remote system conflicts with
04CE   ; information that is already present within the system-wide configuration
04CE   ; data base. In such a case what is logged instead of a data packet is the
04CE   ; remote system node name, the known system nodename, and the known system ID.
04CE   ;
04CE   ; Before calling ERL$LOGMESSAGE to log the error condition, these routines call
04CE   ; OPA0_LOG to log the condition to _OPA0, if such a broadcast is warrented.
04CE   ;
04CE   ; As a matter of convenience, there are four entry points to the routine, one
04CE   ; for each of the following conditions:
04CE   ;       - A path status change (good to bad, or bad to good)
04CE   ;       - A cables crossed/uncrossed status change
04CE   ;       - All other errors detected with in a packet
04CE   ;       - An error log datagram, specified by the PPD type = 5 (PPD$C_ELOG)
04CE   ;         These are used for sending an error log message to a system without
04CE   ;         necessarily having a connection to the system over which to send
04CE   ;         error log info.
04CE   ;
04CE   ; ELOG$PTH_ST_CHG:
04CE   ;
04CE   ; Inputs:
04CE   ;       R0                      - Address of previous path status information byte.
04CE   ;                                 In this byte:
04CE   ;                                       PB$M_CUR_PS eq 0 ==> path was broken
04CE   ;                                       PB$M_CUR_PS ne 0 ==> path was good
04CE   ;                                 The address is assumed to be one of PB$B_P0_STS(R1)
04CE   ;                                 or PB$B_P1_STS(R1).  This information is used to
04CE   ;                                 determine which path is being described.
04CE   ;       R1                      - PB address
04CE   ;       R2                      - Packet address
04CE   ;       R4                      - PDT address
04CE   ;
04CE   ;
04CE   ; ELOG$CBL_X_CHG:
04CE   ;
04CE   ; Inputs:
04CE   ;       R1                      - 0 ==> cables currently crossed
```

```
                          04CE  ;                                  1 ==> cables currently uncrossed
                          04CE  ;           R2                   - Packet address
                          04CE  ;           R3                   - PB address
                          04CE  ;           R4                   - PDT address
                          04CE  ;
                          04CE  ;
                          04CE  ; ELOG$PACKET: and ELOG$CABLES:
                          04CE  ;
                          04CE  ; Inputs:
                          04CE  ;           R0                   - Error subtype code in bits 0 through 7
                          04CE  ;                                  Sign bit set indicates that the error will crash port
                          04CE  ;                                  Sign bit not set indicates that it will not
                          04CE  ;           R1                   - PB address (ELOG$PACKET only)
                          04CE  ;           R2                   - Packet address (zero if none exists)
                          04CE  ;           R4                   - PDT address
                          04CE  ;           R5                   - Known system SB address
                          04CE  ;                                  (ELOG$PACKET and subtype = PAER$K_ES_RSCKS only)
                          04CE  ;
                          04CE  ; ELOG$ERROR_DG:
                          04CE  ;
                          04CE  ; Inputs:
                          04CE  ;           R2                   -Error log packet address
                          04CE  ;           R3                   -PB address
                          04CE  ;           R4                   -PDT address
                          04CE  ;
                          04CE  ; ALL ROUTINES:
                          04CE  ;
                          04CE  ; Outputs:
                          04CE  ;           All other registers are preserved.  An entry is made in the error log.
                          04CE  ;           The existance of this error might have been broadcast to _OPA0.
                          04CE  ;-
                          04CE
           0000003F       04CE  LM_MASK = ^M<R0,R1,R2,R3,R4,R5>
           00000014       04CE  SAVEDR5 = 4*5
                          04CE
                          04CE  ELOG$PTH_ST_CHG::                                   ; Path status change
                          04CE
                 3F   BB  04CE          PUSHR   #LM_MASK                            ; Save registers.
                          04D0          ASSUME  PAER$K_ES_0GB EQ 0
                 55   D4  04D0          CLRL    R5                                  ; Assume it went from good to bad.
              03 60   E8  04D2          BLBS    (R0), 10$                           ; Branch if old status was good.
           55    02   9A  04D5          MOVZBL  #PAER$K_ES_0BG, R5                  ; Else, it went from bad to good.
                          04D8  10$:    ; Determine which path was effected by subtracting the address of the
                          04D8          ; path 0 status byte from the address of the status byte passed to us.
                          04D8          ; Then add the good-to-bad or bad-to-good subtype code base to form
                          04D8          ; the error subtype code.
                          04D8          ASSUME  PB$B_P1_STS EQ PB$B_P0_STS+1
                          04D8          ASSUME  PAER$K_ES_1GB EQ PAER$K_ES_0GB+1
                          04D8          ASSUME  PAER$K_ES_1BG EQ PAER$K_ES_0BG+1
        53    29 A1   9E  04D8          MOVAB   PB$B_P0_STS(R1), R3                 ; Get path 0 status byte address.
           50    53   C2  04DC          SUBL    R3, R0                              ; Subtract from passed address.
           50    55   C0  04DF          ADDL    R5, R0                              ; Add error subtype code base.
                 09   11  04E2          BRB     LOG_AS_CHANGE                       ; Branch to common state change code.
                          04E4
                          04E4
                          04E4  ELOG$CBL_X_CHG::                                    ; Cables crossed/uncrossed change
                          04E4
              3F   BB  04E4                PUSHR    #LM_MASK                  ; Save registers.
                       04E6                ASSUME   PAER$K_ES_CU EQ PAER$K_ES_UC+1
                       04E6                ASSUME   PB$M_CUR_CBL EQ 1
     50   51  04   C1  04E6                ADDL3    #PAER$K_ES_UC, R1, R0     ; Form change crossing subtype.
          51  53   D0  04EA                MOVL     R3, R1                    ; Move PB address to right place.
                       04ED  LOG_AS_CHANGE:
     55     41 8F  9A  04ED                MOVZBL   #PAER$K_ET_CBL, R5        ; Set cable status change error type.
              33   11  04F1                BRB      ELOG$$LOG_LM              ; Branch to common code.
                       04F3
                       04F3
                       04F3  ELOG$CABLES::                                    ; Cables change of state, general case
                       04F3
                       04F3                .ENABL   LSB
                       04F3
              3F   BB  04F3                PUSHR    #LM_MASK                  ; Save registers.
     55     41 8F  9A  04F5                MOVZBL   #PAER$K_ET_CBL, R5        ; Set cable status change error type.
              51   D4  04F9                CLRL     R1                        ; Assume no PB
              52   D5  04FB                TSTL     R2                        ; Is there a message?
              14   13  04FD                BEQL     10$                       ; Branch if no message.
           FAFE'   30  04FF                BSBW     CNF$LKP_PB_MSG            ; Attempt to find path block.
              0F   11  0502                BRB      10$                       ; Join common code
                       0504
                       0504
                       0504  ELOG$PACKET::                                    ; Packet error, general case
                       0504
              3F   BB  0504                PUSHR    #LM_MASK                  ; Save registers.
     52     00B4 C4 C2 0506                SUBL     PDT$L_MSGHDRSZ(R4),R2     ; Back the pointer up
              02   11  050B                BRB      5$
                       050D
                       050D  ELOG$PACKET1::                                   ; Packet error, general case
                       050D
              3F   BB  050D                PUSHR    #LM_MASK                  ; Save registers.
     55     40 8F  9A  050F  5$:           MOVZBL   #PAER$K_ET_PKT, R5        ; Set packet error type.
          50  6E   D0  0513  10$:          MOVL     (SP), R0                  ; Restore caller's error subtype.
              0E   11  0516                BRB      ELOG$$LOG_LM              ; Go to common code.
                       0518
                       0518                .DSABL   LSB
                       0518
                       0518
                       0518  ELOG$ERROR_DG::                                  ; Error log datagram to log
                       0518
              3F   BB  0518                PUSHR    #LM_MASK                  ; Save registers
          50  07   9A  051A                MOVZBL   #PAER$K_ES_ERRDG,R0       ; Get error subtype
          51  53   D0  051D                MOVL     R3,R1                     ; Copy PB address
     55     40 8F  9A  0520                MOVZBL   #PAER$K_ET_PKT,R5         ; Get error type
              00   11  0524                BRB      ELOG$$LOG_LM              ; Join common code to set up
                       0526                                                   ;  error log entry and log it
```

```
                              0526  ;+
                              0526  ; At this point the registers have the following values:
                              0526  ;
                              0526  ;        R0      - Error subtype code in bits 0 through 7
                              0526  ;                  Sign bit set indicates that the error will crash port
                              0526  ;                  Sign bit not set indicates that it will not
                              0526  ;        R1      - =0 ==> no PB exists
                              0526  ;                  Otherwise R1 = PB address
                              0526  ;        R2      - Packet address (zero if none exists)
                              0526  ;        R4      - PDT address
                              0526  ;        R5      - Error type code
                              0526  ;
                              0526  ; The following code will build the logged message buffer in a UCB extension,
                              0526  ; and cause it to be placed in the error log.  It will also call OPA0_LOG
                              0526  ; to broadcast the error to _OPA0 if such a broadcast is required.
                              0526  ; Synchronization on use of the UCB extension area for this purpose is
                              0526  ; accomplished via the UCB$M_ERLOGIP bit in UCB$W_STS.
                              0526  ;
                              0526  ; Because some of the entities in a logged message have odd sizes, the
                              0526  ; following code sometimes saves instructions by incorrectly writing longer
                              0526  ; than necessary entities, and later overwriting the high order portions of
                              0526  ; the written data with the correct information.
                              0526  ;-
                              0526
                              0526
                              0526  ELOG$$LOG_LM:
                              0526
        53   00DC C4    D0    0526          MOVL    PDT$L_UCB0(R4), R3      ; Get the UCB address.
     03 64 A3    02     E3    052B          BBCS    #UCB$V_ERLOGIP, -       ; Flag error logging in progress and
                              0530                  UCB$W_STS(R3), 5$       ; branch if none previously in progress.
             0164       31    0530          BRW     90$                     ; Branch if error log is in progress.
      00D0 C3    50     90    0533  5$:     MOVB    R0, UCB$B_LMEST(R3)     ; Plant error subtype value.
      00D1 C3    55     90    0538          MOVB    R5, UCB$B_LMET(R3)      ; Plant error type value.
                 50     D5    053D          TSTL    R0                      ; Is the port going to be crashed?
                 06     18    053F          BGEQ    10$                     ; Branch if no.  Otherwise, set flag
    00D1 C3    80 8F    88    0541          BISB    #PAER$M_CPRT, UCB$B_LMET(R3) ; bit in error code byte.
  00D2 C3    0080 C3    B0    0547  10$:    MOVW    UCB$B_ERTCNT(R3), -     ; Plant error retry and max retry
                              054E                  UCB$B_LMERTCNT(R3)      ; counts.
 50    0082 C3   01     A1    054E          ADDW3   #1, UCB$W_ERRCNT(R3), R0 ; Adjust unincremented error counter,
      00D4 C3    50     3C    0554          MOVZWL  R0, UCB$W_LMERRCNT(R3)  ; plant it, and zero word following it.
                              0559          ASSUME  UCB$S_LSADDR  EQ 6
                              0559          ASSUME  UCB$S_LSID    EQ 6
                              0559          ASSUME  UCB$S_RSADDR  EQ 6
                              0559          ASSUME  UCB$S_RSID    EQ 6
                              0559          ASSUME  SB$S_SYSTEMID EQ 6
                              0559          $PRTCTINI -                     ; Protect the following device register
                              0559                  B^20$, MCHK$M_NEXM      ; reference from machine checks.
   00D8 C3    010C D4   D0    0569          MOVL    @PDT$L_PPR(R4), -       ; Get the local station address
                              0570                  UCB$N_LSADDR(R3)        ; directly from the port.
                              0570          $PRTCTEND 20$                   ; End protected code.
              0C 50     E8    0571          BLBS    R0, 25$                 ; Branch if no machine check occured.
      00D8 C3    01     CE    0574          MNEGL   #1, UCB$N_LSADDR(R3)    ; If couldn't get local station
      00DC C3    01     AE    0579          MNEGW   #1, UCB$N_LSADDR+4(R3)  ; address, put all ones in its place.
                 04     11    057E          BRB     30$                     ; Then, continue with processing.
         00DA C3        D4    0580  25$:    CLRL    UCB$N_LSADDR+2(R3)      ; If got address, clear high order bits.
 00DE C3   00000000'GF  D0    0584  30$:    MOVL    G^SCS$GB_SYSTEMID, -    ; Get local system id from system
                              058D                  UCB$N_LSID(R3)          ; global address.
        00000004'GF   B0 058D        MOVW    G^SCS$GB_SYSTEMID+4,-     ; Copy h.o. 2 bytes of system id
             00E2 C3     0593                UCB$N_LSID+4(R3)          ;
             00EA C3  7C 0596        CLRQ    UCB$N_RSID(R3)            ; Assume remote system id won't be
                         059A                                          ; found and zero it (plus a little).
                         059A
                         059A        ASSUME  UCB$N_RSADDR+6  EQ UCB$N_RSID
                         059A        ASSUME  UCB$N_RSID+6    EQ UCB$L_CICMD
                         059A        ASSUME  SB$S_NODENAME   EQ 16
                         059A
             00D0 C3  B1 059A        CMPW    UCB$B_LMEST(R3),-         ; Logging known-remote system conflict?
             4008 8F     059E                #<PAER$K_ET_PKT@8 + PAER$K_ES_RSCKS>
                   3D 12 05A1        BNEQ    32$                       ; Branch if not
          55    14 AE D0 05A3        MOVL    SAVEDR5(SP),R5            ; Otherwise restore known system SB addr
          52    30 A1 D0 05A7        MOVL    PB$L_SBLINK(R1),R2        ; Retrieve remote system SB address
        50    00E4 C3 9E 05AB        MOVAB   UCB$N_RSADDR(R3),R0       ; Position to remote system address
                         05B0                                          ; field within logged msg working buffer
          80    0C A1 D0 05B0        MOVL    PB$B_RSTATION(R1),(R0)+   ; Store remote station address
                   80 B4 05B4        CLRW    (R0)+
          80    18 A2 D0 05B6        MOVL    SB$B_SYSTEMID(R2),(R0)+   ; Store remote system ID
          80    1C A2 B0 05BA        MOVW    SB$B_SYSTEMID+4(R2),(R0)+
          80    18 A5 D0 05BE        MOVL    SB$B_SYSTEMID(R5),(R0)+   ; Store known system ID
          80    1C A5 B0 05C2        MOVW    SB$B_SYSTEMID+4(R5),(R0)+
          80    44 A5 7D 05C6        MOVQ    SB$T_NODENAME(R5),(R0)+   ; Store known system nodename
          80    4C A5 7D 05CA        MOVQ    SB$T_NODENAME+8(R5),(R0)+
          80    44 A2 7D 05CE        MOVQ    SB$T_NODENAME(R2),(R0)+   ; Store remote system nodename
          80    4C A2 7D 05D2        MOVQ    SB$T_NODENAME+8(R2),(R0)+
                   53 DD 05D6        PUSHL   R3                        ; Save UCB address
          00    63 00 2C 05D8        MOVC5   #0,(R3),#0,-              ; Clear remainder of logged msg buffer
                60 22    05DC                #<UCB$K_LMPKTBYTS-30>,(R0)
                   5D 11 05DE        BRB     66$                       ; Go finish logged message
                         05E0
                   52 D5 05E0  32$:  TSTL    R2                        ; Is there a message packet?
                   18 12 05E2        BNEQ    35$                       ; Branch if there is one.
           00E4 C3 01 CE 05E4        MNEGL   #1, UCB$N_RSADDR(R3)      ; Else, can't get remote station
           00E8 C3 01 AE 05E9        MNEGW   #1, UCB$N_RSADDR+4(R3)    ; address, so put all ones in its place.
                   53 DD 05EE        PUSHL   R3                        ; Save UCB address.
 0048 8F   00   63 00 2C 05F0        MOVC5   #0, (R3), #0, -           ; Zero all of logged message buffer
             00F0 C3     05F7
                         05FA                #<UCB$K_LMPKTBYTS+8>, -   ; in which message packet would
                         05FA                UCB$L_CICMD(R3)           ; normally be put.
                   41 11 05FA        BRB     66$                       ; Go finish logged message.
       00E4 C3  0C A2 9A 05FC  35$:  MOVZBL  PPD$B_PORT(R2), -         ; Get remote station address from
                         0602                UCB$N_RSADDR(R3)          ; packet.
             00E8 C3  B4 0602        CLRW    UCB$N_RSADDR+4(R3)        ; Zero extend it to 48 bits.
                   51 D5 0606        TSTL    R1                        ; Do we have a PB address?
                   12 13 0608        BEQL    50$                       ; Branch if no and none exists.
          50    30 A1 D0 060A        MOVL    PB$L_SBLINK(R1), R0       ; Get SB address from PB.
                   0C 13 060E        BEQL    50$                       ; Branch if no SB available
       00EA C3  18 A0 D0 0610        MOVL    SB$B_SYSTEMID(R0), -      ; Copy system id from system block
                         0616                UCB$N_RSID(R3)            ; to the log entry.
       00EE C3  1C A0 B0 0616        MOVW    SB$B_SYSTEMID+4(R0), -
                         061C                UCB$N_RSID+4(R3)
                   53 DD 061C  50$:  PUSHL   R3                        ; Save UCB address.
          50    08 A2 32 061E        CVTWL   PPD$W_SIZE(R2),R0         ; Get possible neg offset to net hdr
                   0A 18 0622        BGEQ    55$                       ; Branch if no net header
        50    08 A240 9E 0624        MOVAB   PPD$W_SIZE(R2)[R0],R0     ; Else get addr of net header
     50   60    08 A2 A1 0629        ADDW3   PPD$W_SIZE(R2),(R0),R0    ;  and get size stored in net header
```

```
                                 062E                                               ;  - size of net header
            55   50   0C   A3    062E  55$:   SUBW3   #PPD$B_PORT, R0, R5           ; Compute maximum length of message
                                 0632                                               ; based upon allocated pool region.
  00BC 8F   00   0C A2   55   2C 0632         MOVC5   R5, PPD$B_PORT(R2), -          ; Move all interesting parts of the
                   00F0 C3       063A
                                 063D                 #0, #<UCB$K_ERRDGBYTS+8>, - ; message packet to the logged
                                 063D                 UCB$L_CICMD(R3)                ; message buffer.
                      53 8ED0    063D  66$:   POPL    R3                            ; Restore UCB address.
                 50   03   9A    0640         MOVZBL  #EMB$C_PM, R0                 ; Get CI logged message sub-type code.
            51  0068 8F    3C    0643         MOVZWL  #UCB$K_LMBUFSIZ, R1           ; Get size of logged message.
                0000 C3    B1    0648         CMPW    UCB$B_LMEST(R3),-             ; Is it a plain (short) logged msg?
                4007 8F          064C                 #<PAER$K_ET_PKT@8 + PAER$K_ES_ERRDG>
                   18      12    064F         BNEQ    80$                           ; Branch if so
            51  00F4 C3    3C    0651         MOVZWL  UCB$W_MSGBYTCNT(R3),R1        ; Get a copy of the PPD length from
                                 0656                                               ;  the saved message
                 51   26   C0    0656         ADDL    #<UCB$W_MSGPPDTYP - UCB$B_LMEST>,R1
                                 0659                                               ; Add in other parts of error log entry
        000000DC 8F   51   D1    0659         CMPL    R1,#UCB$K_ERRDGSIZ            ; Is it more than we will log?
                      07   15    0660         BLEQ    80$                           ; Branch if not
        51   000000DC 8F   D0    0662         MOVL    #UCB$K_ERRDGSIZ,R1            ; Else put in max errlog entry size
                                 0669
                 7E   50   7D    0669  80$:   MOVQ    R0,-(SP)                      ; Save registers
            51  FA3A CF    9E    066C         MOVAB   LM_OPA0_LOG_TAB,R1            ; Retrieve logged message _OPA0 table
             00008000 8F   CB    0671         BICL3   #^X00008000,-                 ; Retrieve error subtype and type and
            50  00D0 C3          0677                 UCB$B_LMEST(R3),R0            ; clearing port crash indicating bit
                   1D      10    067B         BSBB    OPA0_LOG                      ; Log the error to _OPA0 if indicated
   53   55   000000A0 8F   C3    067D         SUBL3   #UCB$L_MSGFKBLK,R5,R3         ; Compute UCB address
                 50   8E   7D    0685         MOVQ    (SP)+,R0                      ; Restore registers
                                 0688
            52  00D0 C3    9E    0688         MOVAB   UCB$B_LMEST(R3), R2           ; Get starting address of message.
             00000000'GF   16    068D         JSB     G^ERL$LOGMESSAGE              ; Log the message.
              64 A3   04   AA    0693         BICW    #UCB$M_ERLOGIP, UCB$W_STS(R3) ; Clear err. log in progress flag.
                      3F   BA    0697  90$:   POPR    #LM_MASK                      ; Restore saved registers.
                           05    0699         RSB                                   ; Return to caller.
```

```
                          069A          .SBTTL  OPA0_LOG,                   _OPA0 ERROR LOGGING ROUTINE
                          069A
                          069A   ;+
                          069A   ; This routine first determines whether or not _OPA0 error logging should be
                          069A   ; done. Then, if logging to _OPA0 is indicated, this routine saves what optional
                          069A   ; formatting information will be needed and creates a fork process, using the
                          069A   ; port UCB's message fork block, to handle the formatting and broadcasting of
                          069A   ; the appropriate error log message. If this fork block is currently in use,
                          069A   ; presumably for the broadcasting of an earlier error log message, the
                          069A   ; assumption is made that this earlier message is the more important one, and
                          069A   ; the error condition currently being processed is not logged to _OPA0.
                          069A   ;
                          069A   ; Error logging to _OPA0 will be attempted whenever the system device, which
                          069A   ; is assummed to be the same as the error logging device, is currently
                          069A   ; unavailable. Such error logging will also always be done for certain error
                          069A   ; conditions, such as fatal port initialization errors.
                          069A   ;
                          069A   ; Inputs:
                          069A   ;
                          069A   ;       IPL                             -Device or Fork IPL
                          069A   ;       R0                              -High word 0, Error Subtype, Error Type
                          069A   ;       R1                              -Address of an _OPA0 Error Logging Table
                          069A   ;       R3                              -Address of UCB
                          069A   ;
                          069A   ;       It is assummed that the logged message buffer portion of the UCB has
                          069A   ;       been initialized for all error conditions which use the logged
                          069A   ;       message error log entry format.  The contents of device registers
                          069A   ;       are always obtained via the PDT.
                          069A   ;
                          069A   ; Outputs:
                          069A   ;
                          069A   ;       R0-R1, R3-R4                    -Destroyed
                          069A   ;       R5                              -Address of UCB message fork block
                          069A   ;       Other registers                 -Preserved
                          069A   ;-
                          069A
                          069A          .ENABL  LSB
                          069A   OPA0_LOG:
      55  00A0 C3   9E    069A          MOVAB   UCB$L_MSGFKBLK(R3),R5   ; Retrieve fork block address into R5
                          069F
                          069F   ;
                          069F   ; Find the entry in the appropriate _OPA0 error log table that corresponds
                          069F   ; to the error condition currently being processed.
                          069F   ;
                          069F
          61    50  B1    069F   10$:   CMPW    R0,(R1)                 ; Entry for current error condition?
                09  13    06A2          BEQL    20$                     ; Branch if so
                61  B5    06A4          TSTW    (R1)                    ; Have we reached the end of the table?
                2C  19    06A6          BLSS    30$                     ; Don't perform logging if we have
          51    08  C0    06A8          ADDL2   #OPA0_LOG_SIZE,R1       ; Else, position to next table entry
                F2  11    06AB          BRB     10$                     ; Continue search
```

```
                            06AD
                            06AD  ;
                            06AD  ; The current error condition will be logged to _OPA0 under the following
                            06AD  ; circumstances:
                            06AD  ;
                            06AD  ; 1. It is indicated that such error conditions are always to be logged.
                            06AD  ; 2. The system disk has not yet been mounted.
                            06AD  ; 3. The system disk is currently being mounted.
                            06AD  ; 4. The system disk is undergoing mount verification.
                            06AD  ; 5. During mount verification it is discovered that the system disk drive
                            06AD  ;    contains the wrong volume.
                            06AD  ; 6. The system disk has timed out.
                            06AD  ; 7. The local system is participating in a cluster and quorum has been lost.
                            06AD  ;
                            06AD  ; An implicit assumption is that the system and error logging disk are one and
                            06AD  ; the same.
                            06AD  ;
                            06AD
     23 02 A1    00    E0   06AD  20$:    BBS     #V_ALWAYS,CFLAGS(R1),40$; Go log if this error is always logged
                            06B2
50   00000000'GF       D0   06B2          MOVL    G^EXE$GL_SYSUCB,R0        ; Retrieve UCB for system disk
              34 A0    D5   06B9          TSTL    UCB$L_VCB(R0)             ; Has the system disk been mounted?
                 17    13   06BC          BEQL    40$                       ; Go log if it hasn't
                       B3   06BE          BITW    #UCB$M_MOUNTING!-         ; Is system disk in one of these states?
                            06BF                  UCB$M_WRONGVOL!-          ; Currently being mounted?
                            06BF                  UCB$M_MNTVERIP!-          ; Wrong volume in device?
                            06BF                  UCB$M_TIMOUT,-            ; Mount verification in progress?
  64 A0    C240 8F          06BF                  UCB$W_STS(R0)             ; Timed out?
                 0F    12   06C4          BNEQ    40$                       ; Go log if it is
                            06C6
50   00000000'GF       D0   06C6          MOVL    G^CLU$GL_CLUB,R0          ; Retrieve cluster block
                 05    13   06CD          BEQL    30$                       ; No need to log if there isn't one
                 1C    E1   06CF          BBC     #CLUB$V_QUORUM,-          ; Go log if the system is participating
          01 1C A0          06D1                  CLUB$L_FLAGS(R0),40$      ; in a cluster which has lost quorum
                 05         06D4  30$:    RSB                               ; Return
                 02    E2   06D5  40$:    BBSS    #UCB_V_MSGFKLOCK,-        ; Indicate msg fork block now in use
          FA 68 A3          06D7                  UCB$W_DEVSTS(R3),30$      ; If the fork block already in use,
                            06DA                                            ; assume prior error condition is more
                            06DA                                            ; important & skip logging of this one
```

```
                              06DA      ;
                              06DA      ;
                              06DA      ; A decision has been made to log the error condition to _OPA0. First, store
                              06DA      ; within the UCB any optional information which will be required to format the
                              06DA      ; _OPA0 error log message. Finally setup and create a fork process to format
                              06DA      ; and broadcast the appropriate error log message to _OPA0. The fork process is
                              06DA      ; created using the UCB's message fork block.
                              06DA      ;
                              06DA
  09 02 A1     02      E1     06DA                   BBC     #V_RPORT,CFLAGS(R1),50$ ; Remote port number required?
      00E4 C3          9A     06DF                   MOVZBL  UCB$N_RSADDR(R3),-      ; If so, then save the remote port
      00B8 C3                 06E3                           UCB$T_OPA0_TEMP(R3)     ; number in UCB, and go setup and
               42      11     06E6                   BRB     70$                     ; create the fork process
                              06E8
  09 02 A1     03      E1     06E8      50$:         BBC     #V_PKT,CFLAGS(R1),60$   ; CI packet information required?
      00F0 C3          D0     06ED                   MOVL    UCB$L_CICMD(R3),-       ; If so, then save the CI packet
      00B8 C3                 06F1                           UCB$T_OPA0_TEMP(R3)     ; information in the UCB, and go setup
               34      11     06F4                   BRB     70$                     ; and create the fork process
                              06F6
  2F 02 A1     04      E1     06F6      60$:         BBC     #V_REGS,CFLAGS(R1),70$  ; Branch if device regs not required
   54  0084 C3         D0     06FB                   MOVL    UCB$L_PDT(R3),R4        ; Retrieve PDT address
       00B8 C3         7C     0700                   CLRQ    UCB$T_OPA0_TEMP(R3)     ; Clear UCB locations where the device
       00C0 C3         D4     0704                   CLRL    UCB$T_OPA0_TEMP+8(R3)   ; registers will be saved
                              0708                   $PRTCTINI -                     ; Protect device register references
                              0708                           B^65$,#MCHK$M_NEXM      ; from machine checks
      00E4 D4          D0     0714                   MOVL    @PDT$L_CNF(R4),-        ; Store contents of configuration
      00B8 C3                 0718                           UCB$T_OPA0_TEMP(R3)     ; register
      00E8 D4          D0     071B                   MOVL    @PDT$L_PMC(R4),-        ; Store contents of port maintenance
      00BC C3                 071F                           UCB$T_OPA0_TEMP+4(R3)   ; control register
      00EC D4          D0     0722                   MOVL    @PDT$L_PS(R4),-         ; Store contents of port status register
      00C0 C3                 0726                           UCB$T_OPA0_TEMP+8(R3)
                              0729                   $PRTCTEND 65$                   ; If check occurs, leave zero values(s)
                              072A
        54     51      D0     072A      70$:         MOVL    R1,R4                   ; Save table entry for error in R4
   00000739'EF         9F     072D                   PUSHAB  OPA0_LOG_FORK           ; Fork process routine address
   00000000'GF         17     0733                   JMP     G^EXE$FORK              ; Fork ...
                              0739
                              0739                   .DSABL  LSB
```

```
                        0739          .SBTTL  OPA0_LOG_FORK,          _OPA0 ERROR LOGGING
                        0739          .SBTTL  -                       FORK PROCESS ROUTINE
                        0739
                        0739  ;+
                        0739  ; This is the routine which assumes control, within the context of a fork
                        0739  ; process, when an error log message is to be broadcast to _OPA0. This routine
                        0739  ; formats and broadcasts the _OPA0 error log message as follows:
                        0739  ;
                        0739  ; 1. Optionally format the error log message utilizing information contained
                        0739  ;    within the _OPA0 error log table entry for this specific error condition.
                        0739  ;    The address of the appropriate table entry maybe found within R4 on input
                        0739  ;    to the routine.
                        0739  ; 2. Release the message fork block by clearing the interlock bit. This step
                        0739  ;    must be delayed until after the optional formatting is completed because
                        0739  ;    the optional formatting makes use of UCB locations which we can not allow
                        0739  ;    to be overwritten until we are through with them.
                        0739  ; 3. Copy the device controller letter into the error log message.
                        0739  ; 4. Broadcast the _OPA0 error log message.
                        0739  ; 5. Broadcast a second message indicating that the port will be taken offline
                        0739  ;    if this is indicated for this error condition (Fatal port initialization
                        0739  ;    errors only).
                        0739  ;
                        0739  ; Inputs:
                        0739  ;
                        0739  ;       R3                              -Address of UCB
                        0739  ;       R4                              -Address an _OPA0 Error Logging Table Entry
                        0739  ;       R5                              -Address of Message Fork Block
                        0739  ;
                        0739  ;       It is assummed that the three longwords beginning at UCB$T_OPA0_TEMP
                        0739  ;       have been initialized with whatever values will be required to complete
                        0739  ;       any optional formatting of the current _OPA0 error log message.
                        0739  ;
                        0739  ; Outputs:
                        0739  ;
                        0739  ;       R0-R5                           -Destroyed
                        0739  ;       Other registers                 -Preserved
                        0739  ;-
                        0739
                        0739          .ENABL  LSB
                        0739  OPA0_LOG_FORK:
50     24 A3      D0    0739          MOVL    UCB$L_CRB(R3),R0        ; Retrieve CRB address
       30 A0      DD    073D          PUSHL   CRB$L_INTD+-            ; Retrieve and save address of
                        0740                  VEC$L_INITIAL(R0)       ; controller initialization routine
                        0740
52     06 A4      32    0740          CVTWL   MSG(R4),R2              ; Retrieve offset to counted message
    52    6E      C0    0744          ADDL2   (SP),R2                 ; Compute address of counted message
                        0747
50     04 A4      3C    0747          MOVZWL  FORMAT(R4),R0           ; Retrieve offset to formatting routine
          05      13    074B          BEQL    10$                     ; Branch if no special formatting
    50    6E      C0    074D          ADDL2   (SP),R0                 ; Else compute formatting routine addr
          60      16    0750          JSB     (R0)                    ; Perform special formatting
                        0752
          04      8A    0752  10$:    BICB2   #UCB_M_MSGFKLOCK,-      ; Mark message fork block as being
       68 A3            0754                  UCB$W_DEVSTS(R3)        ; no longer in use
                        0756
    51    82      9A    0756          MOVZBL  (R2)+,R1                ; Retrieve size and address of message
          6E      D4    0759          CLRL    (SP)                    ; Assume will not broadcast "Offline"
```

```
      03 02 A4    01    E1  075B         BBC     #V_OFFLINE,CFLAGS(R4),20$; Branch if this is true
            6E    01    D0  0760         MOVL    #1,(SP)                  ; Else this second msg will be broadcast
                            0763
       54    28 A3      D0  0763  20$:   MOVL    UCB$L_DDB(R3),R4         ; Get DDB address into R4
 55     00000000'GF     9E  0767         MOVAB   G^OPA$UCB0,R5            ; Get _OPA0 UCB address into R5
                            076E
             17 A4      90  076E         MOVB    DDB$T_NAME+3(R4),-       ; Copy device controller letter from
             06 A2          0771                 CTRLR_NAME(R2)           ; DDB to ASCII message
        00000000'GF     16  0773         JSB     G^IOC$BROADCAST          ; Send message to terminal driver
                            0779
                  8E    D5  0779         TSTL    (SP)+                    ; Should the "Offline" msg be broadcast?
                  01    12  077B         BNEQ    30$                      ; Go do so if it should
                        05  077D         RSB                              ; Else return
                            077E
 52     00000000'EF     9E  077E  30$:   MOVAB   INI$MSG_OFFL,R2          ; Retrieve counted message address
           51    82     9A  0785         MOVZBL  (R2)+,R1                 ; Retrieve message size and address
             17 A4      90  0788         MOVB    DDB$T_NAME+3(R4),-       ; Copy device controller letter from
             06 A2          078B                 CTRLR_NAME(R2)           ; DDB to ASCII message
        00000000'GF     17  078D         JMP     G^IOC$BROADCAST          ; Send message to terminal driver and
                            0793                                          ; return
                            0793         .DSABL  LSB
```

```
                                     0793        .SBTTL  _OPA0 ERROR LOGGING FORMATTING ROUTINES
                                     0793        .SBTTL  =       ERR$CNV_HEX_DEC ROUTINE TO CONVERT A BINARY NUMBER
                                     0793        .SBTTL  -                           INTO ITS DECIMAL ASCII EQUIVALENCE
                                     0793
                                     0793  ;+
                                     0793  ; This routine takes a binary number, converts it into a decimal number, and
                                     0793  ; then converts the decimal number into its ASCII equivalence. An implicit
                                     0793  ; assumption is made that the binary number to be converted fits in a byte
                                     0793  ; (ie - has a value in the range 0 - 255 decimal).
                                     0793  ;
                                     0793  ; Inputs:
                                     0793  ;
                                     0793  ;       R0                              -Number to convert into its ASCII equivalence
                                     0793  ;       R2                              -Field in which to store the result
                                     0793  ;
                                     0793  ; Outputs:
                                     0793  ;
                                     0793  ;       R0-R1,R3                        -Destroyed
                                     0793  ;       Other registers                 -Preserved
                                     0793  ;-
                                     0793
                                     0793        .ENABL  LSB
                                     0793  ERR$CNV_HEX_DEC::
                 53    F869 CF   9E  0793        MOVAB   CONV_TABLE,R3           ; Retrieve address of conversion table
                 62    2020 8F   B0  0798        MOVW    #^A/  /,(R2)            ; Blank out first two bytes of field
                                     079D
                            51   D4  079D        CLRL    R1                      ; Clear high order longword
 50    51    50    00000064 8F   7B  079F        EDIV    #100,R0,R1,R0           ; Determine number of 100s and remainder
                            04   13  07A8        BEQL    10$                     ; Branch if no 100s
                    62    6341   90  07AA        MOVB    (R3)[R1],(R2)           ; Otherwise store number in 100s place
                                     07AE
                            51   D4  07AE  10$:  CLRL    R1
             50    51    50 0A   7B  07B0        EDIV    #10,R0,R1,R0            ; Determine number of 10s and remainder
                            05   13  07B5        BEQL    20$
                 01 A2    6341   90  07B7        MOVB    (R3)[R1],1(R2)          ; Store number in 10s place
                                     07BC
                 02 A2    6340   90  07BC  20$:  MOVB    (R3)[R0],2(R2)          ; store number in 1s place
                                 05  07C1        RSB                             ; Return
                                     07C2        .DSABL  LSB
```

```
                          07C2           .SBTTL  -         FORMAT_PKT,      ROUTINE TO FORMAT PACKET
                          07C2           .SBTTL  -                          INFORMATION
                          07C2
                          07C2      ;+
                          07C2      ; This routine formats packet information fields within an _OPA0 error log
                          07C2      ; message. The formatted packet field appears in the message as follows:
                          07C2      ;
                          07C2      ;                         FLAGS/OPC/STATUS/PORT   xx/xx/xx/xx
                          07C2      ;
                          07C2      ; The packet fields are formatted from left to right by calling the routine
                          07C2      ; HEX_TO_ASCII for each packet field to be formatted.
                          07C2      ;
                          07C2      ; Inputs:
                          07C2      ;
                          07C2      ;        R2                          -Address of _OPA0 Error Log Message
                          07C2      ;        R3                          -Address of the UCB
                          07C2      ;        R4                          -Address of an _OPA0 Error Logging Table Entry
                          07C2      ;
                          07C2      ;        It is assummed that UCB$T_OPA0_TEMP has been initialized with the packet
                          07C2      ;        information to be formatted.
                          07C2      ;
                          07C2      ;
                          07C2      ; Outputs:
                          07C2      ;
                          07C2      ;        R0-R1                       -Destroyed
                          07C2      ;        Other registers             -Preserved
                          07C2      ;-
                          07C2
                          07C2           .ENABL  LSB
                          07C2      FORMAT_PKT:
          007C 8F    BB   07C2           PUSHR   #^M<R2,R3,R4,R5,R6>         ; Save some registers
      50    03 A4    98   07C6           CVTBL   OFFSET(R4),R0               ; Retrieve offset to field to format
            52 50    C0   07CA           ADDL2   R0,R2                       ; Compute address of field to format
    55    00BC C3    9E   07CD           MOVAB   UCB$T_OPA0_TEMP+4(R3),R5; Get addr of 1st byte past pkt fields
            56 04    9A   07D2           MOVZBL  #4,R6                       ; Num of packets fields to be formatted
                          07D5
            51 75    9A   07D5      10$:  MOVZBL  -(R5),R1                   ; Get contents of next field to format
            50 02    D0   07D8           MOVL    #2,R0                       ; Set number of nibbles in packet field
               6A    10   07DB           BSBB    HEX_TO_ASCII                ; Format the current packet field
               52    D6   07DD           INCL    R2                          ; Step over the delimiter
            F3 56    F5   07DF           SOBGTR  R6,10$                      ; Continue until all fields formatted
                          07E2
          007C 8F    BA   07E2           POPR    #^M<R2,R3,R4,R5,R6>         ; Restore registers
                     05   07E6           RSB                                 ; Return
                          07E7           .DSABL  LSB
```

```
                      07E7          .SBTTL  -       FORMAT_PORT,    ROUTINE TO FORMAT A
                      07E7          .SBTTL  -                       REMOTE PORT NUMBER
                      07E7
                      07E7    ;+
                      07E7    ; This routine formats a remote port number field within an _OPA0 error log
                      07E7    ; message. The remote port number appears as a decimal number after formatting.
                      07E7    ;
                      07E7    ; Inputs:
                      07E7    ;
                      07E7    ;       R2                              -Address of _OPA0 Error Log Message
                      07E7    ;       R3                              -Address of the UCB
                      07E7    ;       R4                              -Address of an _OPA0 Error Logging Table Entry
                      07E7    ;
                      07E7    ;       It is assummed that UCB$T_OPA0_TEMP has been initialized with the
                      07E7    ;       remote port number to be formatted.
                      07E7    ;
                      07E7    ;
                      07E7    ; Outputs:
                      07E7    ;
                      07E7    ;       R0-R1                           -Destroyed
                      07E7    ;       Other registers                 -Preserved
                      07E7    ;-
                      07E7
                      07E7            .ENABL  LSB
                      07E7    FORMAT_PORT:
      7E    52    7D  07E7            MOVQ    R2,-(SP)                ; Save registers
 50    03   A4    98  07EA            CVTBL   OFFSET(R4),R0           ; Retrieve offset to field to format
      52    50    C0  07EE            ADDL2   R0,R2                   ; Compute address of field to format
50  00B8    C3    D0  07F1            MOVL    UCB$T_OPA0_TEMP(R3),R0  ; Retrieve remote port number
            9B    10  07F6            BSBB    ERR$CNV_HEX_DEC         ; Format the remote port number
      52    8E    7D  07F8            MOVQ    (SP)+,R2                ; Restore registers
                  05  07FB            RSB                             ; Return
                      07FC            .DSABL  LSB
```

```
                        07FC          .SBTTL  -        FORMAT_REGS,    ROUTINE TO FORMAT PORT
                        07FC          .SBTTL  -                        REGISTERS
                        07FC
                        07FC     ;+
                        07FC     ; This routine formats the port register fields within an _OPA0 error log
                        07FC     ; message. Only the contents of selected port registers are formatted. The
                        07FC     ; formatted register fields appear in the message as follows:
                        07FC     ;
                        07FC     ;                       CNF/PMC/PSR      xxxxxxxx/xxxxxxxx/xxxxxxxx
                        07FC     ;
                        07FC     ; The port register fields are formatted from left to right by calling the
                        07FC     ; routine HEX_TO_ASCII for each register field to be formatted.
                        07FC     ;
                        07FC     ; Inputs:
                        07FC     ;
                        07FC     ;       R2                              -Address of _OPA0 Error Log Message
                        07FC     ;       R3                              -Address of the UCB
                        07FC     ;       R4                              -Address of an _OPA0 Error Logging Table Entry
                        07FC     ;
                        07FC     ;       It is assummed that the three longwords beginning at UCB$T_OPA0_TEMP
                        07FC     ;       have been initialized with the values of the device registers to be
                        07FC     ;       formatted.
                        07FC     ;
                        07FC     ;
                        07FC     ; Outputs:
                        07FC     ;
                        07FC     ;       R0-R1                           -Destroyed
                        07FC     ;       Other registers                 -Preserved
                        07FC     ;-
                        07FC
                        07FC          .ENABL  LSB
                        07FC     FORMAT_REGS:
        007C 8F   BB    07FC          PUSHR   #^M<R2,R3,R4,R5,R6>     ; Save some registers
   50     03 A4   98    0800          CVTBL   OFFSET(R4),R0           ; Retrieve offset to field to format
      52     50   C0    0804          ADDL2   R0,R2                   ; Compute address of field to format
  55    00B8 C3   9E    0807          MOVAB   UCB$T_OPA0_TEMP(R3),R5  ; Get address of first port register
      56     03   9A    080C          MOVZBL  #3,R6                   ; Num of register fields to be formatted
                        080F
      51     85   D0    080F     10$:  MOVL    (R5)+,R1                ; Get contents of next port register
      50     08   D0    0812          MOVL    #8,R0                   ; Set number of nibbles in packet field
           002F   30    0815          BSBW    HEX_TO_ASCII            ; Format the current port register field
             52   D6    0818          INCL    R2                      ; Step over the delimiter
          F2 56   F5    081A          SOBGTR  R6,10$                  ; Continue until all registers formatted
                        081D
        007C 8F   BA    081D          POPR    #^M<R2,R3,R4,R5,R6>     ; Restore registers
                  05    0821          RSB                             ; Return
                        0822          .DSABL  LSB
                        0822            .SBTTL  -       FORMAT_REV,     FORMAT PORT UCODE REV LEVELS
                        0822
                        0822    ;+
                        0822    ; This routine formats the PROM and RAM revision levels within an OPA0 message.
                        0822    ; The formatted field appears in the message as follows:
                        0822    ;
                        0822    ;       RAM/PROM rev is xxxx/xxxx
                        0822    ;
                        0822    ; The fields are formatted from left to right by calling the routine
                        0822    ; HEX_TO_ASCII for each rev.
                        0822    ;
                        0822    ; Inputs:
                        0822    ;
                        0822    ;       R2                              -Address of OPA0 error message
                        0822    ;       R3                              -Addr of UCB
                        0822    ;       R4                              -Addr of OPA0 error message table entry
                        0822    ;
                        0822    ;       It is assumed that UCB$T_OPA0_TEMP has been initialized with
                        0822    ;       the rev level information to be formatted.
                        0822    ;
                        0822    ; Outputs:
                        0822    ;
                        0822    ;       R0,R1                           -Destroyed
                        0822    ;       Other registers                 -Preserved
                        0822    ;-
                        0822
                        0822            .ENABL  LSB
                        0822
                        0822    FORMAT_REV:
                        0822
         007C 8F    BB  0822            PUSHR   #^M<R2,R3,R4,R5,R6>     ; Save caller's registers
   50      03 A4    98  0826            CVTBL   OFFSET(R4),R0           ; Retreive offset to field to fmt
           52 50    C0  082A            ADDL2   R0,R2                   ; Compute addr of field to fmt
   55    00B8 C3    3E  082D            MOVAW   UCB$T_OPA0_TEMP(R3),R5  ; Get addr of RAM rev
           56 02    9A  0832            MOVZBL  #2,R6                   ; Two rev levels to fmt
                        0835
           51 85    B0  0835    10$:    MOVW    (R5)+,R1                ; Get next rev level
           50 04    D0  0838            MOVL    #4,R0                   ; 4 hex digits/rev level
              0A    10  083B            BSBB    HEX_TO_ASCII            ; Format this rev
              52    D6  083D            INCL    R2                      ; Step past slash delimiter, /
           F3 56    F5  083F            SOBGTR  R6,10$                  ; Continue formatting revs
         007C 8F    BA  0842            POPR    #^M<R2,R3,R4,R5,R6>     ; Restore registers
                    05  0846            RSB                             ; Return to caller
                        0847
                        0847            .DSABL  LSB
```

```
                            0847         .SBTTL  -       HEX_TO_ASCII    ROUTINE TO CONVERT A BINARY NUMBER
                            0847         .SBTTL  -                       INTO ITS ASCII EQUIVALENCE
                            0847
                            0847  ;+
                            0847  ; This routine takes a binary number, converts it into its ASCII equivalence,
                            0847  ; and stores it in the field provided. The nibbles of the binary number are
                            0847  ; processed and stored in their ASCII equivalences from left to right. This
                            0847  ; routine is capable of handling up to a longword at a time in this fashion.
                            0847  ;
                            0847  ; Inputs:
                            0847  ;
                            0847  ;        R0                      -Number of nibbles in field to be converted
                            0847  ;        R1                      -Number to convert into its ASCII equivalence
                            0847  ;        R2                      -Field in which to store the ASCII equivalences
                            0847  ;
                            0847  ; Outputs:
                            0847  ;
                            0847  ;        R0,R3-R4                -Destroyed
                            0847  ;        R2                      -Address of first byte past field
                            0847  ;        Other registers         -Preserved
                            0847  ;-
                            0847
                            0847         .ENABL  LSB
                            0847  HEX_TO_ASCII:
          53   F7B5 CF   9E 0847         MOVAB   CONV_TABLE,R3           ; Retrieve address of conversion table
          50     50  02  78 084C         ASHL    #2,R0,R0                ; Compute bit number of leftmost nibble
                 50  04  C2 0850         SUBL2   #4,R0                   ; which is to be converted
                            0853
     54   51     04  50  EF 0853  10$:   EXTZV   R0,#4,R1,R4             ; Extract the current nibble
             82   6344   90 0858         MOVB    (R3)[R4],(R2)+          ; Move ASCII equivalence into field
FFF0 50      FC 8F   00  9D 085C         ACBB    #0,#-4,R0,10$           ; Continue until all nibbles processed
                         05 0863         RSB                             ; Return
                            0864         .DSABL  LSB
                            0864
                            0864         .END
```

```
                                                          M 16
PAERROR                    Error Handling & Logging Routines      16-SEP-1984 01:16:25  VAX/VMS Macro V04-00           Page  54
Symbol table                                                      10-SEP-1984 01:16:10  [DRIVER.SRC]PAERROR.MAR;2           (35)

$$$CURSIZ                = 000001C4              ERR$V_DEB_ABO            = 00000008  G
$$$NEWSIZ                = 000001D0              ERR$V_DEB_ACCV           = 00000002  G
$$MSG_PTR                = 00000954 R     03     ERR$V_DEB_BLV            = 00000001  G
BELL                     = 00000007              ERR$V_DEB_BUGNF          = 00000014  G
BUG$_CIPORT                ********   X   01     ERR$V_DEB_CNFER          = 00000011  G
CDT$C_CON_REC            = 00000009              ERR$V_DEB_ILKQ           = 00000012  G
CDT$C_VC_FAIL            = 0000000C              ERR$V_DEB_INVBN          = 00000000  G
CDT$L_CDTLST             = 0000006C              ERR$V_DEB_INVDP          = 00000006  G
CDT$W_STATE              = 00000028              ERR$V_DEB_INVOP          = 0000000B  G
CFLAGS                   = 00000002              ERR$V_DEB_MFQE           = 00000018  G
CLN_BYTES                = 00000014              ERR$V_DEB_NEPQ           = 00000013  G
CLU$GL_CLUB                ********   X   01     ERR$V_DEB_NOPB           = 00000010  G
CLUB$L_FLAGS             = 0000001C              ERR$V_DEB_NOSTS          = 0000000D  G
CLUB$V_QUORUM            = 0000001C              ERR$V_DEB_NPUPD          = 00000009  G
CNF$LKP_PB_MSG             ********   X   01     ERR$V_DEB_OSEQ           = 00000016  G
CNF$LKP_PB_PDT             ********   X   01     ERR$V_DEB_PSRX           = 00000015  G
CNF$REMOVE_PB              ********   X   01     ERR$V_DEB_PSV            = 00000003  G
COM$DRVDEALMEM             ********   X   01     ERR$V_DEB_SCERR          = 0000000F  G
CONV_TABLE                 00000000 R     01     ERR$V_DEB_UNSTS          = 0000000C  G
CR                       = 0000000D              ERR$V_DEB_URC            = 00000007  G
CRB$L_INTD               = 00000024              ERR$V_DEB_URP            = 00000005  G
CTRLR_NAME               = 00000006              ERR$V_DEB_VCDCL          = 00000017  G
DA_MASK                  = 0000003E              ERR$V_DEB_VCUPD          = 0000000A  G
DA_OPA0_LOG_TAB            00000010 R     01     ERR$V_DEB_XCTER          = 0000000E  G
DDB$T_NAME               = 00000014              EXE$FORK                   ********   X   01
ELOG$$LOG_DA               0000043B R     01     EXE$GL_DEFFLAGS            ********   X   01
ELOG$$LOG_LM               00000526 R     01     EXE$GL_LOCKRTRY            ********   X   01
ELOG$CABLES                000004F3 RG    01     EXE$GL_SYSUCB              ********   X   01
ELOG$CBL_X_CHG             000004E4 RG    01     EXE$MCHK_PRTCT             ********   X   01
ELOG$CPU_REV               000003F3 RG    01     EXE$V_FATAL_BUG            ********   X   01
ELOG$ERROR_DG              00000518 RG    01     FATALQ                     00000349 R     01
ELOG$HARDWARE              0000041A RG    01     FLUSH_Q                    00000319 R     01
ELOG$INIT_SWERR            000003BD RG    01     FORMAT                   = 00000004
ELOG$INTRLOCK              00000425 RG    01     FORMAT_PKT                 000007C2 R     01
ELOG$K_BYTES             = 0000007A  G           FORMAT_PORT                000007E7 R     01
ELOG$PACKET                00000504 RG    01     FORMAT_REGS                000007FC R     01
ELOG$PACKET1               0000050D RG    01     FORMAT_REV                 00000822 R     01
ELOG$PTH_ST_CHG            000004CE RG    01     HEX_TO_ASCII               00000847 R     01
ELOG$REGDUMP               00000476 RG    01     IDB$L_CSR                = 00000000
ELOG$UCODE_ERR             000003FE RG    01     INI$FORK                   ********   X   01
ELOG$UCODE_NORD            000003C9 RG    01     INI$MSG_OFFL               ********   X   01
ELOG$UCODE_WARN            00000407 RG    01     INI$PORT                   ********   X   01
EMB$C_PM                 = 00000003              INT$DEAL_PKT               ********   X   01
EMB$L_DV_REGSAV          = 0000004E              INT$DISP_SENDDG            ********   X   01
ERL$DEVICEATTN             ********   X   01     INT$INS_COMQH              ********   X   01
ERL$LOGMESSAGE             ********   X   01     IOC$BROADCAST              ********   X   01
ERR$BUGCHECK               0000039D RG    01     LF                       = 0000000A
ERR$BUGCHECKNF             0000037D RG    01     LM_MASK                  = 0000003F
ERR$CLEANUP_PKT            000002DC RG    01     LM_OPA0_LOG_TAB            000000AA R     01
ERR$CNV_HEX_DEC            00000793 RG    01     LOG_AS_CHANGE              000004ED R     01
ERR$CRASHVC                00000154 RG    01     LOG_AS_HARDWARE            00000420 R     01
ERR$CRASH_PORT             0000018D RG    01     MCHK$M_NEXM              = 00000004
ERR$DEBUGCHECK             000003B9 RG    01     MSG                      = 00000006
ERR$DISC_PWFAIL            000002A6 RG    01     M_ALWAYS                 = 00000001
ERR$DISP_ENTRY             00000332 RG    01     M_OFFLINE                = 00000002
ERR$INIPORT                00000355 RG    01     M_PKT                    = 00000008
ERR$PWF_RECOV              000001C1 RG    01     M_REGS                   = 00000010
```

```
M_RPORT                          = 00000004               PAER$K_ET_INSW                  = 00000000
NUM_EX_LONGWORDS                 = 00000003               PAER$K_ET_LMLT                  = 00000042
OFFSET                           = 00000003               PAER$K_ET_PKT                   = 00000040
OPA$UCB0                           ********    X    01    PAER$M_CPRT                     = 00000080
OPA0_LOG                           0000069A R       01    PA_CNF                            00000000
OPA0_LOG_FORK                      00000739 R       01    PA_CQ0                            00000908
OPA0_LOG_SIZE                    = 00000008               PA_CQ1                            0000090C
PA$CTLINIT                         ********    X    01    PA_CQ2                            00000910
PAER$K_ES_0BG                    = 00000002               PA_CQ3                            00000914
PAER$K_ES_0GB                    = 00000000               PA_C_WCSSIZ                     = 00000C00
PAER$K_ES_1BG                    = 00000003               PA_DFQ                            00000928
PAER$K_ES_1GB                    = 00000001               PA_MADR                           00000014
PAER$K_ES_CNPB                   = 00000004               PA_MDATR                          00000018
PAER$K_ES_CODE                   = 00000001               PA_MFQ                            0000092C
PAER$K_ES_CPUREV                 = 00000007               PA_MTC                            00000930
PAER$K_ES_CSMP                   = 00000002               PA_MTEC                           00000934
PAER$K_ES_CU                     = 00000005               PA_PDC                            00000920
PAER$K_ES_DQIN                   = 00000006               PA_PEC                            0000091C
PAER$K_ES_DQRM                   = 00000001               PA_PESR                           0000093C
PAER$K_ES_ERRDG                  = 00000007               PA_PFAR                           00000938
PAER$K_ES_HCIN                   = 00000003               PA_PIC                            00000924
PAER$K_ES_HWER                   = 00000002               PA_PMC                            00000004
PAER$K_ES_INIT                   = 00000001               PA_PMC_M_MIN                    = 00000001
PAER$K_ES_L0BG                   = 00000008               PA_PPR                            00000940
PAER$K_ES_L0BX                   = 0000000A               PA_PQBBR                          00000904
PAER$K_ES_L0GB                   = 00000006               PA_PS                             00000900
PAER$K_ES_L1BG                   = 00000009               PA_PSR                            00000918
PAER$K_ES_L1BX                   = 0000000B               PB$B_P0_STS                     = 00000029
PAER$K_ES_L1GB                   = 00000007               PB$B_P1_STS                     = 0000002A
PAER$K_ES_LCIN                   = 00000004               PB$B_RSTATION                   = 0000000C
PAER$K_ES_LST0                   = 00000003               PB$C_LENGTH                     = 00000054
PAER$K_ES_LST1                   = 00000009               PB$C_PALENGTH                     00000060
PAER$K_ES_LST2                   = 00000007               PB$C_PWR_FAIL                   = 00004000
PAER$K_ES_LST3                   = 00000009               PB$C_VC_FAIL                    = 00008000
PAER$K_ES_LST4                   = 0000000C               PB$L_CDTLST                     = 00000034
PAER$K_ES_MQIN                   = 00000005               PB$L_CLSCKT_DG                    00000054
PAER$K_ES_MQRM                   = 00000000               PB$L_SBLINK                     = 00000030
PAER$K_ES_NOPB                   = 00000006               PB$M_CUR_CBL                    = 00000001
PAER$K_ES_PCVC                   = 00000001               PB$W_STATE                      = 00000012
PAER$K_ES_PDWN                   = 00000003               PDT$B_DQIMAP                      00000154
PAER$K_ES_POOL                   = 00000000               PDT$B_HSHUT_DG                    000001B0
PAER$K_ES_PUP                    = 00000004               PDT$B_MAX_PORT                    0000017C
PAER$K_ES_REVCA                  = 00000008               PDT$B_NXT_PORT                    0000017E
PAER$K_ES_REVER                  = 00000006               PDT$B_P0_LBSTS                    00000180
PAER$K_ES_RQRM                   = 00000002               PDT$B_P1_LBSTS                    00000181
PAER$K_ES_RSCKS                  = 00000008               PDT$B_PLOGMAP                     00000134
PAER$K_ES_SCA                    = 00000005               PDT$B_PORTMAP                     00000114
PAER$K_ES_SCSID                  = 00000002               PDT$B_PORT_NUM                    0000017D
PAER$K_ES_SCVC                   = 00000003               PDT$B_REQIDPS                     0000017F
PAER$K_ES_UC                     = 00000004               PDT$C_LENGTH                    = 000000E4
PAER$K_ES_UCDW                   = 00000000               PDT$C_PALENGTH                  = 00000360
PAER$K_ES_UPKT                   = 00000000               PDT$C_PAREGBASE                   000000E4
PAER$K_ES_UXIN                   = 00000005               PDT$C_PAREGEND                    00000110
PAER$K_ET_CBL                    = 00000041               PDT$C_PQB                       = 000001E0
PAER$K_ET_DALT                   = 00000003               PDT$L_CNF                         000000E4
PAER$K_ET_HW                     = 00000001               PDT$L_CQ0                         000000F0
PAER$K_ET_ILCK                   = 00000002               PDT$L_CQ1                         000000F4
```

```
PDT$L_DFQ                        000000FC                PPD$B_PORT                      0000000C
PDT$L_DFQHDR                     00000208                PPD$B_PROTOCOL                  0000001A
PDT$L_DGHDRSZ                    00000190                PPD$B_RSTATE                    00000025
PDT$L_DGNETHD                    00000194                PPD$B_RST_PORT                  00000024
PDT$L_DQELOGOUT                  000002E0                PPD$B_STATUS                    0000000D
PDT$L_GPTBASE                    0000022C                PPD$B_SWFLAG                    0000000B
PDT$L_GPTLEN                     00000230                PPD$B_SYSTEMID                  00000014
PDT$L_LBDG                       00000184                PPD$B_TYPE                      0000000A
PDT$L_MFQ                        00000100                PPD$C_LB_LENGTH                 00000046
PDT$L_MFQHDR                     0000020C                PPD$C_LCB_DATA                  00000013
PDT$L_MQELOGOUT                  00000320                PPD$C_LENGTH                    00000012
PDT$L_MSGHDRSZ                 = 000000B4                PPD$C_MIN_DGSIZ                 00000050
PDT$L_MTC                        00000104                PPD$C_SETCKT                  = 00000019
PDT$L_PFAR                       00000108                PPD$C_SNDDG                   = 00000001
PDT$L_PMC                        000000E8                PPD$K_LB_LENGTH                 00000046
PDT$L_POLLERDUE                  0000018C                PPD$K_LENGTH                    00000012
PDT$L_POOLDUE                    00000188                PPD$L_BLINK                     00000004
PDT$L_PPR                        0000010C                PPD$L_DG_DISC                   00000028
PDT$L_PS                         000000EC                PPD$L_FLINK                     00000000
PDT$L_PSR                        000000F8                PPD$L_IN_VCD                    00000018
PDT$L_SPTBASE                    00000224                PPD$L_LBCRC                     00000042
PDT$L_SPTLEN                     00000228                PPD$L_P0_ACK                    00000010
PDT$L_UCB0                     = 000000DC                PPD$L_P0_NAK                    00000014
PDT$L_VBDT                       0000021C                PPD$L_P0_NRSP                   00000018
PDT$L_VPQB                       00000218                PPD$L_P1_ACK                    0000001C
PDT$M_PWF_CLNUP                = 00000001                PPD$L_P1_NAK                    00000020
PDT$Q_COMQ2                      000001F0                PPD$L_P1_NRSP                   00000024
PDT$Q_COMQ3                      000001F8                PPD$L_REC_BOFF                  00000028
PDT$Q_COMQBASE                   000001E0                PPD$L_REC_NAME                  00000024
PDT$Q_COMQH                      000001E8                PPD$L_RPORT_FCN                 00000020
PDT$Q_COMQL                      000001E0                PPD$L_RPORT_REV                 0000001C
PDT$Q_DFREEQ                     000001D0                PPD$L_RPORT_TYP                 00000018
PDT$Q_FORMPB                     00000174                PPD$L_SND_BOFF                  00000020
PDT$Q_MFREEQ                     000001D8                PPD$L_SND_NAME                  0000001C
PDT$Q_RSPQ                       00000200                PPD$L_ST_ADDR                   00000018
PDT$Q_TEMP_RSPQ                  0000019C                PPD$L_XCT_LEN                   00000018
PDT$V_PUP                      = 00000001                PPD$M_CST                     = 00008000
PDT$V_PWF_CLNUP                = 00000000                PPD$M_DISPOSE                 = 00000001
PDT$W_BDTLEN                     00000220                PPD$M_RSP                     = 00000001
PDT$W_DQELEN                     00000210                PPD$Q_CURTIME                   00000048
PDT$W_LPORT_STS                  00000110                PPD$Q_NODENAME                  00000040
PDT$W_MQELEN                     00000214                PPD$Q_SWINCARN                  00000028
PDT$W_PBCOUNT                    00000112                PPD$Q_XCT_ID                    00000010
PDT$W_STDGDYN                    00000198                PPD$T_HWTYPE                    00000030
PDT$W_STDGUSED                   0000019A                PPD$T_SWTYPE                    00000020
PORT_REGS_LOGGED               = 00000006                PPD$T_SWVERS                    00000024
PORT_UCODE                       0000040C R      01      PPD$V_RSP                     = 00000000
PPD$B_DEF_ST                     0000001C                PPD$W_LCB_LEN7                  0000000C
PPD$B_FLAGS                      0000000F                PPD$W_LENGTH                    00000010
PPD$B_HWVERS                     00000034                PPD$W_MASK                      00000010
PPD$B_LBDATA                     00000012                PPD$W_MAXDG                     0000001C
PPD$B_LCB_0                      00000012                PPD$W_MAXMSG                    0000001E
PPD$B_LCB_LPORT                  00000010                PPD$W_MTYPE                     00000012
PPD$B_LCB_NPORT                  0000000F                PPD$W_M_VAL                     00000014
PPD$B_LCB_OPC                    00000011                PPD$W_SIZE                      00000008
PPD$B_LCB_PORT                   0000000E                PR$_IPL                         ********  X     01
PPD$B_OPC                        0000000E                Q_UNLOCKED                      000002A5 R      01
```

```
REV_ERROR                        00000414 R     01        UCB$W_DEVSTS                  = 00000068
SAVEDR5                        = 00000014                 UCB$W_ERRCNT                  = 00000082
SB$B_SYSTEMID                  = 00000018                 UCB$W_LMERRCNT                  000000D4
SB$S_NODENAME                  = 00000010                 UCB$W_MSGBYTCNT                 000000F4
SB$S_SYSTEMID                  = 00000006                 UCB$W_MSGPPDTYP                 000000F6
SB$T_NODENAME                  = 00000044                 UCB$W_STS                     = 00000064
SCS$CLOSE_CDT                    ********   X   01        UCB_M_MSGFKLOCK               = 00000004
SCS$DEALL_CDT                    ********   X   01        UCB_V_MSGFKLOCK               = 00000002
SCS$DEAL_SCSREC                  ********   X   01        UNLOCK_BADQ                     00000295 R     01
SCS$FREE_LISTEN                  ********   X   01        VEC$L_IDB                     = 00000008
SCS$GB_SYSTEMID                  ********   X   01        VEC$L_INITIAL                 = 0000000C
SCS$GL_MCADR                     ********   X   01        V_ALWAYS                      = 00000000
SCS$NOTIFY_SYSAP                 ********   X   01        V_OFFLINE                     = 00000001
SIZ...                         = 00000001                 V_PKT                         = 00000003
SS$_ABORT                      = 0000002C                 V_REGS                        = 00000004
SS$_CTRLERR                    = 00000054                 V_RPORT                       = 00000002
SS$_NORMAL                     = 00000001
SS$_POWERFAIL                  = 00000364
SUBTYPE                        = 00000000
TOTAL_LONGWORDS                = 0000000B
TYPE                           = 00000001
UCB$B_DIPL                     = 0000005E
UCB$B_ERTCNT                   = 00000080
UCB$B_LMERTCNT                   000000D2
UCB$B_LMERTMAX                   000000D3
UCB$B_LMEST                      000000D0
UCB$B_LMET                       000000D1
UCB$K_ERRDGBYTS                = 000000B4
UCB$K_ERRDGSIZ                 = 000000DC
UCB$K_LMBUFSIZ                 = 00000068
UCB$K_LMPKTBYTS                = 00000040
UCB$L_CICMD                      000000F0
UCB$L_CRB                      = 00000024
UCB$L_DDB                      = 00000028
UCB$L_DPC                      = 0000009C
UCB$L_FR4                      = 00000014
UCB$L_MSGFKBLK                   000000A0
UCB$L_PDT                      = 00000084
UCB$L_VCB                      = 00000034
UCB$M_ERLOGIP                  = 00000004
UCB$M_MNTVERIP                 = 00004000
UCB$M_MOUNTING                 = 00000200
UCB$M_ONLINE                   = 00000010
UCB$M_TIMOUT                   = 00000040
UCB$M_WRONGVOL                 = 00008000
UCB$N_LSADDR                     000000D8
UCB$N_LSID                       000000DE
UCB$N_RSADDR                     000000E4
UCB$N_RSID                       000000EA
UCB$S_LSADDR                   = 00000006
UCB$S_LSID                     = 00000006
UCB$S_RSADDR                   = 00000006
UCB$S_RSID                     = 00000006
UCB$T_MSGDATA                    000000F8
UCB$T_OPAO_TEMP                  000000B8
UCB$V_ERLOGIP                  = 00000002
UCB$V_ONLINE                   = 00000004
```

```
                                              +----------------+
                                              ! Psect synopsis !
                                              +----------------+

PSECT name                       Allocation          PSECT No.  Attributes
----------                       ----------          ---------  ----------
.  ABS  .                        00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$$$115_DRIVER                    00000864  ( 2148.)  01 (  1.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC LONG
$ABS$                            00000944  ( 2372.)  02 (  2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
$$$110_MSGS                      000009A6  ( 2470.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE

                                           +------------------------+
                                           ! Performance indicators !
                                           +------------------------+

Phase                       Page faults    CPU Time       Elapsed Time
-----                       -----------    --------       ------------
Initialization                       35    00:00:00.02    00:00:02.32
Command processing                  133    00:00:00.46    00:00:03.31
Pass 1                              624    00:00:19.74    00:01:11.93
Symbol table sort                     0    00:00:02.31    00:00:09.17
Pass 2                              412    00:00:05.30    00:00:18.81
Symbol table output                   3    00:00:00.24    00:00:01.58
Psect synopsis output                 2    00:00:00.02    00:00:00.02
Cross-reference output                0    00:00:00.00    00:00:00.00
Assembler run totals               1211    00:00:28.09    00:01:47.14

The working set limit was 2400 pages.
167702 bytes (328 pages) of virtual memory were used to buffer the intermediate code.
There were 120 pages of symbol table space allocated to hold 2150 non-local and 70 local symbols.
2432 source lines were read in Pass 1, producing 33 object records in Pass 2.
50 pages of virtual memory were used to define 47 macros.

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

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA28:[DRIVER.OBJ]PALIB.MLB;1                    9
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                        24
_$255$DUA28:[SYSLIB]STARLET.MLB;2                      8
TOTALS (all libraries)                                41

2482 GETS were required to define 41 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:PAERROR/OBJ=OBJ$:PAERROR MSRC$:PAERROR/UPDATE=(ENH$:PAERROR)+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

0114 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY