```
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
```

FILEID**PADRIVER

```
PPPPPPPP      AAAAAA    DDDDDDDD    RRRRRRRR       IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
PPPPPPPP      AAAAAA    DDDDDDDD    RRRRRRRR       IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
PP      PP  AA      AA  DD      DD  RR      RR       II      VV      VV  EE          RR      RR
PP      PP  AA      AA  DD      DD  RR      RR       II      VV      VV  EE          RR      RR
PP      PP  AA      AA  DD      DD  RR      RR       II      VV      VV  EE          RR      RR
PP      PP  AA      AA  DD      DD  RR      RR       II      VV      VV  EE          RR      RR
PPPPPPPP    AA      AA  DD      DD  RRRRRRRR         II      VV      VV  EEEEEEEE    RRRRRRRR
PPPPPPPP    AA      AA  DD      DD  RRRRRRRR         II      VV      VV  EEEEEEEE    RRRRRRRR
PP          AAAAAAAAAA  DD      DD  RR  RR           II      VV      VV  EE          RR  RR
PP          AAAAAAAAAA  DD      DD  RR  RR           II      VV      VV  EE          RR  RR
PP          AA      AA  DD      DD  RR    RR         II        VV  VV    EE          RR    RR
PP          AA      AA  DD      DD  RR    RR         II        VV  VV    EE          RR    RR
PP          AA      AA  DDDDDDDD    RR      RR     IIIIII        VV      EEEEEEEEEE  RR      RR
PP          AA      AA  DDDDDDDD    RR      RR     IIIIII        VV      EEEEEEEEEE  RR      RR

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

```
C
C Version:      'V04-000'
C
C****************************************************************************
C*                                                                          *
C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
C*  ALL RIGHTS RESERVED.                                                    *
C*                                                                          *
C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
C*  TRANSFERRED.                                                            *
C*                                                                          *
C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
C*  CORPORATION.                                                            *
C*                                                                          *
C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
C*                                                                          *
C*                                                                          *
C****************************************************************************
C

c       Author  Brian Porter                    Creation date   22-FEB-1982

c++
c       Functional description:
c
c       This module displays entries made by the padriver.
c
c       Modified by:
c
c       V03-010 EAD0178         Elliott A. Drayton              24-May-1984
c               Added code to handle zero length HSC datagram message.
c
c       V03-009 EAD0173         Elliott A. Drayton               9-May-1984
c               Added code to prevent HSC datagram format overflow.
c
c       V03-008 EAD0122         Elliott A. Drayton              24-Mar-1984
c               Changed PA error title for subtype 7.
c
c       V03-007 EAD0121         Elliott A. Drayton              24-Mar-1984
c               Add support for new PA errors subtypes 2,7, and 8.
c
C       V03-006 SAR0199         Sharon A. Reynolds,     20-Feb-1984
C                Added an SYE update that:
C                - Fixed an incorrect path number being reported.
C
C       V03-005 SAR0164         Sharon A. Reynolds,     13-Oct-1983
C               - Added an SYE update that implements new spec
C                 changes for PSR/PESR.
C               - Fixed a bug in the padriver_attention_error_code
C                 routine.
```

```
C
C       V03-004 SAR0088         Sharon A. Reynolds,     20-Jun-1983
C               Changed the carriage control in the 'format' statements
C               for use with ERF.
C
C       V03-003 SAR0057         Sharon A. Reynolds,     15-Jun-1983
C               Removed brief/cryptic support.
C
c       v03-002 BP0002          Brian Porter,           20-AUG-1982
c               Added ci750.
c
c       v03-001 BP0001          Brian Porter,           22-JUL-1982
c               Corrected 'ppd$b_flags' conversion error.
c**
c--

        Subroutine PADRIVER_ATTENTION780 (lun)

        include 'src$:msghdr.for /nolist'
        include 'src$:deverr.for /nolist'


        byte            lun

        integer*4       padriver_error_type_code
        integer*4       pcnfgr
        integer*4       pmcsr
        integer*4       psr
        integer*4       pfar
        integer*4       pesr
        integer*4       ppr
        integer*4       pmadr
        integer*4       pmdatr
        integer*4       correct_control_store_value
        integer*4       compress4

        logical*1       diagnostic_mode

        equivalence     (emb$l_dv_regsav(0),padriver_error_type_code)
        equivalence     (emb$l_dv_regsav(1),pcnfgr)
        equivalence     (emb$l_dv_regsav(2),pmcsr)
        equivalence     (emb$l_dv_regsav(3),psr)
        equivalence     (emb$l_dv_regsav(4),pfar)
        equivalence     (emb$l_dv_regsav(5),pesr)
        equivalence     (emb$l_dv_regsav(6),ppr)
        equivalence     (emb$l_dv_regsav(7),pmadr)
        equivalence     (emb$l_dv_regsav(8),pmdatr)
        equivalence     (emb$l_dv_regsav(9),correct_control_store_value)


        call frctof (lun)

        call header (lun)

        call logger (lun,'DEVICE ATTENTION')

        call padriver_attention_error_code (lun,padriver_error_type_code)

        call padriver_initialization (lun,padriver_error_type_code)

        if (lib$extzv(8,7,padriver_error_type_code) .eq. 0) goto 75

c
c       set not diagnostic mode for now
c

        diagnostic_mode = .false.

        if (.not. diagnostic_mode) then

        call ci780_rega (lun,pcnfgr)
        else

        call linchk (lun,2)

        write(lun,5) pcnfgr
5       format(/' ',t8,'CNFGR',t24,z8.8)
        endif

        call ci_pmcsr (lun,pmcsr,diagnostic_mode)

        call ci_psr (lun,psr,diagnostic_mcde)

        call linchk (lun,1)

        write(lun,10) pfar
10      format(' ',t8,'PFAR',t24,z8.8)

        call ci_pesr (lun,pesr,psr,diagnostic_mode)

        call ci_ppr (lun,ppr,psr,diagnostic_mode)

        call ci_control_store_mismatch (lun,pmadr,pmdatr,
        1 correct_control_store_value,padriver_error_type_cade,diagnostic_mode)

        call linchk (lun,1)

        write(lun,15)
15      format(' ',:)

        call ucb$b_ertcnt (lun,lib$extzv(16,8,padriver_error_type_code))

        call ucb$b_ertmax (lun,lib$extzv(24,8,padriver_error_type_code))

        call ucb$l_char (lun,emb$l_dv_char)

        call ucb$w_sts (lun,emb$w_dv_sts)

        call ucb$w_errcnt (lun,emb$w_dv_errcnt)

75      return
        End
```

```
                                                          G 15
PADRIVER_ATTENTION780                                    16-Sep-1984 00:11:24    VAX-11 FORTRAN V3.4-56              Page   4
                                                          5-Sep-1984 14:10:51    DISK$VMSMASTER:[ERF.SRC]PADRIVER.FOR;1

PROGRAM SECTIONS

    Name                                  Bytes   Attributes

  0 $CODE                                   434   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                   82   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                  280   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG
  3 EMB                                     512   PIC OVR REL GBL   SHR NOEXE   RD   WRT LONG

    Total Space Allocated                  1308

ENTRY POINTS

    Address   Type  Name

  0-00000000        PADRIVER_ATTENTION780

VARIABLES

    Address   Type  Name                                     Address   Type  Name

  2-00000004  I*4   COMPRESS4                              3-00000076  I*4   CORRECT_CONTROL_STORE_VALUE
  2-00000000  L*1   DIAGNOSTIC_MODE                        3-0000001C  L*1   EMB$B_DV_CLASS
  3-00000010  L*1   EMB$B_DV_ERTCNT                        3-00000011  L*1   EMB$B_DV_ERTMAX
  3-0000003E  L*1   EMB$B_DV_NAMLNG                        3-0000003A  L*1   EMB$B_DV_SLAVE
  3-0000001D  L*1   EMB$B_DV_TYPE                          3-00000036  I*4   EMB$L_DV_CHAR
  3-00000012  I*4   EMB$L_DV_IOSB1                         3-00000016  I*4   EMB$L_DV_IOSB2
  3-00000026  I*4   EMB$L_DV_MEDIA                         3-0000004E  I*4   EMB$L_DV_NUMREG
  3-0000002E  I*4   EMB$L_DV_OPCNT                         3-00000032  I*4   EMB$L_DV_OWNUIC
  3-0000001E  I*4   EMB$L_DV_RQPID                         3-00000000  I*4   EMB$L_HD_SID
  3-0000003F  CHAR  EMB$T_DV_NAME                          3-00000024  I*2   EMB$W_DV_BCNT
  3-00000022  I*2   EMB$W_DV_BOFF                          3-0000002C  I*2   EMB$W_DV_ERRCNT
  3-0000003C  I*2   EMB$W_DV_FUNC                          3-0000001A  I*2   EMB$W_DV_STS
  3-0000002A  I*2   EMB$W_DV_UNIT                          3-00000004  I*2   EMB$W_HD_ENTRY
  3-0000000E  I*2   EMB$W_HD_ERRSEQ                       AP-00000004@ L*1   LUN
  3-00000052  I*4   PADRIVER_ERROR_TYPE_CODE               3-00000056  I*4   PCNFGR
  3-00000066  I*4   PESR                                   3-00000062  I*4   PFAR
  3-0000006E  I*4   PMADR                                  3-0000005A  I*4   PMCSR
  3-00000072  I*4   PMDATR                                 3-0000006A  I*4   PPR
  3-0000005E  I*4   PSR

ARRAYS

    Address   Type  Name                                   Bytes  Dimensions

  3-00000000  L*1   EMB                                      512  (0:511)
  3-00000052  I*4   EMB$L_DV_REGSAV                          420  (0:104)
  3-00000006  I*4   EMB$Q_HD_TIME                              8  (2)
```

```
                                                       H 15
PADRIVER_ATTENTION780                                  16-Sep-1984 00:11:24    VAX-11 FORTRAN V3.4-56              Page   5
                                                        5-Sep-1984 14:10:51    DISK$VMSMASTER:[ERF.SRC]PADRIVER.FOR;1

LABELS

    Address    Label        Address    Label        Address    Label        Address    Label

  1-00000029   5'         1-0000003C   10'        1-0000004D   15'        0-000001B1   75

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name                                    Type  Name                                     Type  Name

        CI780_REGA                                    CI_CONTROL_STORE_MISMATCH                      CI_PESR
        CI_PMCSR                                      CI_PPR                                         CI_PSR
        FRCTOF                                        HEADER                                    I*4  LIB$EXTZV
        LINCHK                                        LOGGER                                         PADRIVER_ATTENTION_ERROR_CODE
        PADRIVER_INITIALIZATION                       UCB$B_ERTCNT                                   UCB$B_ERTMAX
        UCB$L_CHAR                                    UCB$W_ERRCNT                                   UCB$W_STS


        Subroutine PADRIVER_ATTENTION750 (lun)


        include 'src$:msghdr.for /nolist'
        include 'src$:deverr.for /nolist'


        byte            lun

        integer*4       padriver_error_type_code
        integer*4       pcnfgr
        integer*4       pmcsr
        integer*4       psr
        integer*4       pfar
        integer*4       pesr
        integer*4       ppr
        integer*4       pmadr
        integer*4       pmdatr
        integer*4       correct_control_store_value
        integer*4       compress4

        logical*1       diagnostic_mode

        equivalence     (emb$l_dv_regsav(0),padriver_error_type_code)
        equivalence     (emb$l_dv_regsav(1),pcnfgr)
        equivalence     (emb$l_dv_regsav(2),pmcsr)
        equivalence     (emb$l_dv_regsav(3),psr)
        equivalence     (emb$l_dv_regsav(4),pfar)
        equivalence     (emb$l_dv_regsav(5),pesr)
        equivalence     (emb$l_dv_regsav(6),ppr)
        equivalence     (emb$l_dv_regsav(7),pmadr)
        equivalence     (emb$l_dv_regsav(8),pmdatr)
        equivalence     (emb$l_dv_regsav(9),correct_control_store_value)


        call frctof (lun)

        call header (lun)

        call logger (lun,'DEVICE ATTENTION')

        call padriver_attention_error_code (lun,padriver_error_type_code)

        call padriver_initialization (lun,padriver_error_type_code)

        if (lib$extzv(8,7,padriver_error_type_code) .eq. 0) goto 20

c
c       set not diagnostic_mode for now
c

        diagnostic_mode = .false.

        If (LIB$EXTZV(14,1,pcnfgr) .EQ. 1) then

```

```
                Diagnostic_mode = .true.
                Endif

                if (.not. diagnostic_mode) then

                call ci750_cnfgr (lun,pcnfgr)
                else

                call linchk (lun,3)

                write(lun,5) pcnfgr
        5       format(/' ',t8,'CNFGR',t24,z8.8,/,
                1 T40,'DIAGNOSTIC MODE')
                endif

                call ci_pmcsr (lun,pmcsr,diagnostic_mode)

                call ci_psr (lun,psr,diagnostic_mode)

                call linchk (lun,1)

                write(lun,10) pfar
        10      format(' ',t8,'PFAR',t24,z8.8)

                call ci_pesr (lun,pesr,psr,diagnostic_mode)

                call ci_ppr (lun,ppr,psr,diagnostic_mode)

                call ci_control_store_mismatch (lun,pmadr,pmdatr,
                1 correct_control_store_value,padriver_error_type_code,diagnostic_mode)

                call linchk (lun,1)

                write(lun,15)
        15      format(' ',:)

                call ucb$b_ertcnt (lun,lib$extzv(16,8,padriver_error_type_code))

                call ucb$b_ertmax (lun,lib$extzv(24,8,padriver_error_type_code))

                call ucb$l_char (lun,emb$l_dv_char)

                call ucb$w_sts (lun,emb$w_dv_sts)

                call ucb$w_errcnt (lun,emb$w_dv_errcnt)

        20      return
                End
```

```
PROGRAM SECTIONS

    Name                                        Bytes   Attributes

  0 $CODE                                         454   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                        106   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                        296   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG
  3 EMB                                           512   PIC OVR REL GBL   SHR NOEXE   RD   WRT LONG

    Total Space Allocated                        1368


ENTRY POINTS

    Address  Type  Name

  0-00000000       PADRIVER_ATTENTION750


VARIABLES

    Address  Type  Name                                      Address  Type  Name

  2-00000004  I*4  COMPRESS4                              3-00000076  I*4  CORRECT_CONTROL_STORE_VALUE
  2-00000000  L*1  DIAGNOSTIC_MODE                        3-0000001C  L*1  EMB$B_DV_CLASS
  3-00000010  L*1  EMB$B_DV_ERTCNT                        3-00000011  L*1  EMB$B_DV_ERTMAX
  3-0000003E  L*1  EMB$B_DV_NAMLNG                        3-0000003A  L*1  EMB$B_DV_SLAVE
  3-0000001D  L*1  EMB$B_DV_TYPE                          3-00000036  I*4  EMB$L_DV_CHAR
  3-00000012  I*4  EMB$L_DV_IOSB1                         3-00000016  I*4  EMB$L_DV_IOSB2
  3-00000026  I*4  EMB$L_DV_MEDIA                         3-0000004E  I*4  EMB$L_DV_NUMREG
  3-0000002E  I*4  EMB$L_DV_OPCNT                         3-00000032  I*4  EMB$L_DV_OWNUIC
  3-0000001E  I*4  EMB$L_DV_RQPID                         3-00000000  I*4  EMB$L_HD_SID
  3-0000003F  CHAR EMB$T_DV_NAME                          3-00000024  I*2  EMB$W_DV_BCNT
  3-00000022  I*2  EMB$W_DV_BOFF                          3-0000002C  I*2  EMB$W_DV_ERRCNT
  3-0000003C  I*2  EMB$W_DV_FUNC                          3-0000001A  I*2  EMB$W_DV_STS
  3-0000002A  I*2  EMB$W_DV_UNIT                          3-00000004  I*2  EMB$W_HD_ENTRY
  3-0000000E  I*2  EMB$W_HD_ERRSEQ                       AP-00000004@ L*1  LUN
  3-00000052  I*4  PADRIVER_ERROR_TYPE_CODE               3-00000056  I*4  PCNFGR
  3-00000066  I*4  PESR                                   3-00000062  I*4  PFAR
  3-0000006E  I*4  PMADR                                  3-0000005A  I*4  PMCSR
  3-00000072  I*4  PMDATR                                 3-0000006A  I*4  PPR
  3-0000005E  I*4  PSR


ARRAYS

    Address  Type  Name                                      Bytes  Dimensions

  3-00000000  L*1  EMB                                         512  (0:511)
  3-00000052  I*4  EMB$L_DV_REGSAV                             420  (0:104)
  3-00000006  I*4  EMB$Q_HD_TIME                                 8  (2)
```

```
LABELS

    Address   Label        Address   Label        Address   Label        Address   Label

  1-0000002D  5'         1-00000054  10'        1-00000065  15'        0-000001C5  20

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name                           Type  Name                           Type  Name

        CI750_CNFGR                          CI_CONTROL_STORE_MISMATCH            CI_PESR
        CI_PMCSR                             CI_PPR                               CI_PSR
        FRCTOF                               HEADER                         I*4   LIB$EXTZV
        LINCHK                               LOGGER                               PADRIVER_ATTENTION_ERROR_CODE
        PADRIVER_INITIALIZATION              UCB$B_ERTCNT                         UCB$B_ERTMAX
        UCB$L_CHAR                           UCB$W_ERRCNT                         UCB$W_STS
```

```
        Subroutine PADRIVER_ATTENTION_ERROR_CODE (lun,padriver_error_type_code)

        include 'src$:msghdr.for /nolist'
        include 'src$:deverr.for /nolist'

        byte            lun

        integer*4       padriver_error_type_code
        integer*4       error_type
        integer*4       error_subtype
        integer*4       compress4, Length

        Character*(80)  Message
        Character*(*)   Msg_free, Gram_free, Hi, Lo, Prio_cmd,
        1               Q_ins_fail, Q_rem_fail, Resp,
        1               Msg1, Msg2, Msg3, Msg4, Msg5,
        1               Msg6, Msg7, Msg8, Msg9, Msg10,
        1               Msg11,Msg12,Msg13

        Parameter       (
        1 Msg_free = 'MESSAGE FREE ',
        2 Gram_free = 'DATAGRAM FREE ',
        3 Hi = 'HIGH ',
        4 Lo = 'LOW ',
        5 Prio_cmd = 'PRIORITY COMMAND ',
        6 Q_ins_fail = 'QUEUE INSERT FAILURE',
        7 Q_rem_fail = 'QUEUE REMOVE FAILURE',
        8 Resp = 'RESPONSE ',
        9 Msg1 = 'INSUFFICIENT NON-PAGED POOL FOR INITIALIZATION',
        1 Msg2 = 'FAILED TO LOCATE PORT MICRO-CODE IMAGE',
        2 Msg3 = 'MICRO-CODE VERIFICATION ERROR',
        3 Msg4 = 'NO TRANSITION FROM ''UNINITIALIZED'' TO ''DISABLED''',
        4 Msg5 = 'PORT ERROR BIT(S) SET',
        5 Msg6 = 'PORT POWER DOWN',
        6 Msg7 = 'PORT POWER UP',
        7 Msg8 = 'UNEXPECTED INTERRUPT',
        8 Msg9 = 'SCSSYSTEMID MUST BE SET TO A NON-ZERO VALUE.',
        9 Msg10 = 'CI PORT MICROCODE REV NOT ',
        1 Msg11 = 'SUPPORTED',
        2 Msg12 = 'CURRENT, BUT SUPPORTED',
        3 Msg13 = '11/750 CPU MICROCODE NOT ADEQUATE FOR CI')

        Error_subtype = lib$extzv(0,8,padriver_error_type_code)
        Error_type = lib$extzv(8,7,padriver_error_type_code)

        Call linchk (lun,2)

        Goto ( 100, 200 ) error_type

        If (error_type .eq. 0) then
         If (error_subtype .eq. 0) then
           Message = msg1
           Length = len (msg1)
           Goto 990
        Else if (error_subtype .eq. 1) then
                Message = msg2
                Length = len (msg2)
```

```
                         Goto 990
         Else if (error_subtype .eq. 2) then
                         Message = msg9
                         Goto 990
         Endif

        Else
         Write(lun,995) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
        1 '''PADRIVER'' ERROR TYPE #',error_type,'., ERROR SUB-TYPE #',
        1 error_subtype,'.'
995      Format(/' ','CI SUB-SYSTEM, ',a,
        1 i<compress4 (lib$extzv(0,16,emb$w_dv_unit))>,': - ',
        1 a,i<compress4 (error_type)>,a,i<compress4 (error_subtype)>,a)

        Endif

        Return

100     Goto ( 5, 10, 15, 20, 25, 30, 35, 40 ) error_subtype

        If (error_subtype .eq. 0) then
          Message = msg3
          Length = len (msg3)
          Goto 990
        Endif
        Return

5       Message = msg4
        Length = len (msg4)
        Goto 990

10      Message = msg5
        Length = len (msg5)
        Goto 990

15      Message = msg6
        Length = len (msg6)
        Goto 990

20      Message = msg7
        Length = len (msg7)
        Goto 990

25      Message = msg8
        Length = len (msg8)
        Goto 990

30      Message = msg10 // msg11
        Goto 990

35      Message = msg13
        Goto 990

40      Message = msg10 // msg12
        Goto 990

200     Goto ( 210,220,230,240,250,260 ) error_subtype
```

```
        If (error_subtype .eq. 0) then
          Message = msg_free // q_rem_fail
          Length = len (msg_free) + len ( q_rem_fail)
          Goto 990                        ! Go Write
        Endif
        Return

210     Message = gram_free // q_rem_fail
        length = len (gram_free) + len (q_rem_fail)
        Goto 990

220     Message = resp // q_rem_fail
        Length = len (resp) + len (q_rem_fail)
        Goto 990

230     Message = hi // prio_cmd // q_ins_fail
        Length = len (hi) + len (prio_cmd) + len (q_ins_fail)
        Goto 990

240     Message = lo // prio_cmd // q_ins_fail
        Length = len (lo) + len (prio_cmd) + len (q_ins_fail)
        Goto 990

250     Message = msg_free // q_ins_fail
        Length = len (msg_free) + len (q_ins_fail)
        Goto 990

260     Message = gram_free // q_ins_fail
        Length = len (gram_free) + len (q_ins_fail)

990     write(lun,991) emb$t_dv_name(1:emb$b_dv_namlng),
        1 emb$w_dv_unit, Message

991     format(/' ','CI SUB-SYSTEM, ',a,
        1 i<compress4 (lib$extzv(0,16,emb$w_dv_unit))>,': - ',a,
        1 :i<compress4 (error_subtype)>,:a)

        Return
        End
```

```
                                                                         D 16
PADRIVER_ATTENTION_ERROR_CODE                                           16-Sep-1984 00:11:24    VAX-11 FORTRAN V3.4-56           Page  14
                                                                         5-Sep-1984 14:10:51    DISK$VMSMASTER:[ERF.SRC]PADRIVER.FOR;1

PROGRAM SECTIONS

    Name                                    Bytes   Attributes

  0 $CODE                                     809   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                    803   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                    216   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG
  3 EMB                                       512   PIC OVR REL GBL   SHR NOEXE   RD   WRT LONG

    Total Space Allocated                    2340


ENTRY POINTS

    Address   Type  Name

  0-00000000        PADRIVER_ATTENTION_ERROR_CODE


VARIABLES

    Address   Type  Name                                   Address   Type  Name

  3-0000001C  L*1  EMB$B_DV_CLASS                       3-00000010  L*1  EMB$B_DV_ERTCNT
  3-00000011  L*1  EMB$B_DV_ERTMAX                      3-0000003E  L*1  EMB$B_DV_NAMLNG
  3-0000003A  L*1  EMB$B_DV_SLAVE                       3-0000001D  L*1  EMB$B_DV_TYPE
  3-00000036  I*4  EMB$L_DV_CHAR                        3-00000012  I*4  EMB$L_DV_IOSB1
  3-00000016  I*4  EMB$L_DV_IOSB2                       3-00000026  I*4  EMB$L_DV_MEDIA
  3-0000004E  I*4  EMB$L_DV_NUMREG                      3-0000002E  I*4  EMB$L_DV_OPCNT
  3-00000032  I*4  EMB$L_DV_OWNUIC                      3-0000001E  I*4  EMB$L_DV_RQPID
  3-00000000  I*4  EMB$L_HD_SID                         3-0000003F  CHAR EMB$T_DV_NAME
  3-00000024  I*2  EMB$W_DV_BCNT                        3-00000022  I*2  EMB$W_DV_BOFF
  3-0000002C  I*2  EMB$W_DV_ERRCNT                      3-0000003C  I*2  EMB$W_DV_FUNC
  3-0000001A  I*2  EMB$W_DV_STS                         3-0000002A  I*2  EMB$W_DV_UNIT
  3-00000004  I*2  EMB$W_HD_ENTRY                       3-0000000E  I*2  EMB$W_HD_ERRSEQ
  2-00000054  I*4  ERROR_SUBTYPE                        2-00000050  I*4  ERROR_TYPE
  2-00000058  I*4  LENGTH                              AP-00000004@ L*1  LUN
  2-00000000  CHAR MESSAGE                             AP-00000008@ I*4  PADRIVER_ERROR_TYPE_CODE


ARRAYS

    Address   Type  Name                                     Bytes  Dimensions

  3-00000000  L*1  EMB                                        512  (0:511)
  3-00000052  I*4  EMB$L_DV_REGSAV                            420  (0:104)
  3-00000006  I*4  EMB$Q_HD_TIME                                8  (2)


LABELS

    Address   Label        Address   Label        Address   Label        Address   Label        Address   Label        Address   Label

  0-00000131  5          0-00000145  10         0-00000159  15         0-0000016D  20         0-00000181  25         0-00000195  30
  0-000001A5  35         0-000001B5  40         0-00000102  100        0-000001C5  200        0-000001F0  210        0-00000204  220
  0-00000218  230        0-00000230  240        0-00000248  250        0-0000025C  260        0-0000026E  990        1-00000072  991'
  1-0000003F  995'
```

```
FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name              Type  Name             Type  Name

   I*4  COMPRESS4          I*4  LIB$EXTZV              LINCHK
```

```



      Subroutine PADRIVER_INITIALIZATION (lun,padriver_error_type_code)

      byte            lun

      integer*4       padriver_error_type_code
      integer*4       initialization_retry_count
      integer*4       initialization_maxtry_count
      integer*4       compress4

      logical*1       port_reinitialization


      port_reinitialization = .false.

      if (lib$extzv(15,1,padriver_error_type_code) .eq. 1)
      1 port_reinitialization = .true.

      initialization_retry_count = lib$extzv(16,8,padriver_error_type_code)
      initialization_maxtry_count = lib$extzv(24,8,padriver_error_type_code)

      if (port_reinitialization) then

      call linchk (lun,2)

      if (initialization_retry_count .gt. 0) then

      write(lun,10) 'PORT WILL BE RESTARTED, ',
      1 initialization_retry_count,'. OF ',initialization_maxtry_count,
      1 '. RETRIES REMAINING'
10    format(/' ',t8,a,i<compress4 (initialization_retry_count)>,a,
      1 i<compress4 (initialization_maxtry_count)>,a)
      else

      write(lun,15) '0. RETRIES REMAINING, PORT WILL BE DISABLED'
15    format(/' ',t8,a)
      endif
      endif

      return

      end
```

```
PROGRAM SECTIONS

    Name                                     Bytes   Attributes

  0 $CODE                                      226   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                     145   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                     120   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG

    Total Space Allocated                      491

ENTRY POINTS

    Address   Type  Name

  0-00000000        PADRIVER_INITIALIZATION

VARIABLES

    Address   Type  Name                                   Address   Type  Name

  2-00000008  I*4   INITIALIZATION_MAXTRY_COUNT          2-00000004  I*4   INITIALIZATION_RETRY_COUNT
 AP-00000004@ I*1   LUN                                 AP-00000008@ I*4   PADRIVER_ERROR_TYPE_CODE
  2-00000000  L*1   PORT_REINITIALIZATION

LABELS

    Address   Label       Address   Label

  1-00000073  10'       1-00000089  15'

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name             Type  Name              Type  Name

   I*4  COMPRESS4         I*4  LIB$EXTZV               LINCHK
```

```



        Subroutine CI_PESR (lun,pesr,psr,diagnostic_mode)

        byte            lun

        integer*4       pesr
        integer*4       psr
        integer*4       compress4
        Integer*4       pesr_value

        logical*1       diagnostic_mode


        call linchk (lun,1)

        write(lun,25) pesr
25      format(' ',t8,'PESR',t24,z8.8)

        if (.not. diagnostic_mode) then

        if (lib$extzv(4,1,psr) .eq. 1) then

        Pesr_value = LIB$EXTZV(0,20,pesr)

        If (pesr_value .NE. 0) then
        Call LINCHK (lun,1)
        Endif

        IF (pesr_value .EQ. 1) then

        write(lun,30) 'ILLEGAL SYSTEM VIRT ADDR FORMAT'
30      format(' ',t40,a,:i<compress4 (pesr_value)>,:a)

        else if (pesr_value .eq. 2) then
        write(lun,30) 'NON-EXISTENT SYSTEM VIRTUAL ADDR'

        else if (pesr_value .eq. 3) then
        write(lun,30) 'INVALID SYSTEM "PTE"'

        else if (pesr_value .eq. 4) then
        write(lun,30) 'INVALID BUFFER "PTE"'

        else if (pesr_value .eq. 5) then
        write(lun,30) 'NON-EXISTENT SYSTEM GBL VIRT ADDR'

        else if (pesr_value .eq. 6) then
        write(lun,30) 'NON-EXISTENT BUFFER GBL VIRT ADDR'

        else if (pesr_value .eq. 7) then
        write (lun,30) 'INVALID SYSTEM GLOBAL "PTE"'

        else if (pesr_value .eq. 8) then
        write(lun,30) 'INVALID BUFFER GLOBAL "PTE"'

        else if (pesr_value .eq. 9) then
        write(lun,30) 'INVALID SYSTEM GBL ''PTE'' MAPPING'

        else if (pesr_value .eq. 10) then
        write(lun,30) 'INVALID BUFFER GBL ''PTE'' MAPPING'

        else if (pesr_value .eq. 11) then
        write(lun,30) 'QUEUE INTERLOCK RETRY FAILURE'

        else if (pesr_value .eq. 12) then
        write(lun,30) 'ILLEGAL QUEUE OFFSET ALIGNMENT'

        else if (pesr_value .eq. 13) then
        write(lun,30) 'ILLEGAL ''PQB'' FORMAT'

        else if (pesr_value .eq. 14) then
        write(lun,30) 'REGISTER PROTOCOL VIOLATION'
        else

        write(lun,30) 'ERROR STATUS CODE #',pesr_value,'.'
        endif
        endif

        If (LIB$EXTZV(7,1,psr) .EQ. 1) then

        Pesr_value = LIB$EXTZV(16,5,pesr)

        If (pesr_value .NE. 0) then
        Call LINCHK (lun,1)
        Endif

        If (pesr_value .EQ. 1) then

        write(lun,30) 'RECEIVE BUFFERS EMPTY, FLAG SET'

        else if (pesr_value .eq. 2) then

        write(lun,30) 'INTERNAL PACKET IN ILLEGAL STATE'

        else if (pesr_value .eq. 3) then

        write(lun,30) 'PORT STATUS, ENABLED AND DISABLED'

        else if (pesr_value .eq. 4) then

        write(lun,30) 'COMMAND, COMPLETE AND INCOMPLETE'

        else if (pesr_value .eq. 5) then

        write(lun,30) 'INTERNAL QUEUE RETRY EXPIRED'

        else if (pesr_value .eq. 6) then

        write(lun,30) 'INTERNAL TRANSMIT, NO PATH'

        else if (pesr_value .eq. 7) then

        write(lun,30) 'RECEIVE PACKET, ACK AND NACK'

        else if (pesr_value .eq. 8) then

        write(lun,30) 'PATH FAILURE, BOTH AVAILABLE'

        else if (pesr_value .eq. 9) then

        write(lun,30) 'UNKNOWN MAINTENANCE OPCODE'

        else if (pesr_value .eq. 10) then

        write(lun,30) 'BOTH PATHS BEING FORCED'

        else if (pesr_value .eq. 11) then

        write(lun,30) 'ILLEGAL CSB STATE'
        else

        write(lun,30) 'ERROR STATUS CODE #',pesr_value,'.'
        endif
        endif
        endif

        return

        End
```

```
PROGRAM SECTIONS

    Name                                 Bytes   Attributes

  0 $CODE                                 1357   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                 778   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                 304   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG

    Total Space Allocated                 2439

ENTRY POINTS

    Address   Type  Name

  0-00000000        CI_PESR

VARIABLES

    Address   Type  Name                                Address   Type  Name

 AP-00000010@ L*1  DIAGNOSTIC_MODE                  AP-00000004@ L*1  LUN
 AP-00000008@ I*4  PESR                              2-00000C00  I*4  PESR_VALUE
 AP-0000000C@ I*4  PSR
```

```
LABELS

    Address   Label        Address   Label

  1-000002E9  25'        1-000002FA  30'

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name            Type  Name            Type  Name

   I*4  COMPRESS4        I*4  LIB$EXTZV             LINCHK
```

```



        Subroutine CI_PMCSR (lun,pmcsr,diagnostic_mode)


        byte            lun

        integer*4       pmcsr

        logical*1       diagnostic_mode

        character*29    v1pmcsr(0:4)
        data    v1pmcsr(0)      /'MAINTENANCE INITIALIZE*'/
        data    v1pmcsr(1)      /'MAINTENANCE TIMER DISABLE*'/
        data    v1pmcsr(2)      /'MAINTENANCE INTERRUPT ENABLE*'/
        data    v1pmcsr(3)      /'MAINTENANCE INTERRUPT FLAG*'/
        data    v1pmcsr(4)      /'WRONG PARITY*'/

        character*30    v2pmcsr(6:15)
        data    v2pmcsr(6)      /'PROGRAMMABLE STARTING ADDRESS*'/
        data    v2pmcsr(7)      /'UNINITIALIZED STATE*'/
        data    v2pmcsr(8)      /'TRANSMIT BUFFER PARITY ERROR*'/
        data    v2pmcsr(9)      /'OUTPUT PARITY ERROR*'/
        data    v2pmcsr(10)     /'INPUT PARITY ERROR*'/
        data    v2pmcsr(11)     /'TRANSMIT BUFFER PARITY ERROR*'/
        data    v2pmcsr(12)     /'RECEIVE BUFFER PARITY ERROR*'/
        data    v2pmcsr(13)     /'LOCAL STORE PARITY ERROR*'/
        data    v2pmcsr(14)     /'CONTROL STORE PARITY ERROR*'/
        data    v2pmcsr(15)     /'PARITY ERROR*'/


        call linchk (lun,1)

        write(lun,5) pmcsr
5       format(' ',t8,'PMCSR',t24,z8.8)

        if (.not. diagnostic_mode) then

        call output (lun,pmcsr,v1pmcsr,0,0,4,'0')

        call output (lun,pmcsr,v2pmcsr,6,6,15,'0')
        endif

        return

        End
```

```
PROGRAM SECTIONS

    Name                                        Bytes   Attributes

  0 $CODE                                          98   PIC CON REL LCL   SHR    EXE   RD NOWRT LONG
  1 $PDATA                                         40   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                        596   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG

    Total Space Allocated                         734

ENTRY POINTS

    Address   Type  Name

  0-00000000        CI_PMCSR

VARIABLES

    Address   Type  Name                                     Address   Type  Name

 AP-0000000C@ L*1   DIAGNOSTIC_MODE                       AP-00000004@ L*1   LUN
 AP-00000008@ I*4   PMCSR

ARRAYS

    Address   Type  Name                      Bytes  Dimensions

  2-00000000  CHAR V1PMCSR                     145  (0:4)
  2-00000091  CHAR V2PMCSR                     300  (6:15)

LABELS

    Address   Label

  1-00000016  5'

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name             Type  Name

        LINCHK                 OUTPUT
        Subroutine CI_PSR (lun,psr,diagnostic_mode)


        byte            lun

        integer*4       psr

        logical*1       diagnostic_mode

        character*29    v1psr(0:7)
        data    v1psr(0)        /'RESPONSE QUEUE AVAILABLE*'/
        data    v1psr(1)        /'MESSAGE FREE QUEUE EMPTY*'/
        data    v1psr(2)        /'PORT DISABLE COMPLETE*'/
        data    v1psr(3)        /'PORT INITIALIZATION COMPLETE*'/
        data    v1psr(4)        /'DATA STRUCTURE ERROR*'/
        data    v1psr(5)        /'MEMORY SYSTEM ERROR*'/
        data    v1psr(6)        /'MAINTENANCE TIMER EXPIRATION*'/
        Data    v1psr(7)        /'MISCELLANEOUS ERROR DETECTED*'/

        character*18    v2psr(31:31)
        data    v2psr(31)       /'MAINTENANCE ERROR*'/


        call linchk (lun,1)

        write(lun,5) psr
5       format(' ',t8,'PSR',t24,z8.8)

        if (.not. diagnostic_mode) then

        call output (lun,psr,v1psr,0,0,7,'0')

        call output (lun,psr,v2psr,31,31,31,'0')
        endif

        return

        End
```

```
PROGRAM SECTIONS

    Name                                      Bytes   Attributes

  0 $CODE                                        98   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                       34   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                      400   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG

    Total Space Allocated                       532

ENTRY POINTS

    Address   Type  Name

  0-00000000        CI_PSR

VARIABLES

    Address   Type  Name                                  Address   Type  Name

 AP-0000000C@ L*1   DIAGNOSTIC_MODE                     AP-00000004@ L*1   LUN
 AP-00000008@ I*4   PSR

ARRAYS

    Address   Type  Name                    Bytes  Dimensions

  2-00000000  CHAR V1PSR                      232  (0:7)
  2-000000E8  CHAR V2PSR                       18  (31:31)

LABELS

    Address   Label

  1-00000012  5'

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name            Type  Name

        LINCHK                OUTPUT
        Subroutine CI_PPR (lun,ppr,psr,diagnostic_mode)

        byte            lun

        integer*4       ppr
        integer*4       psr
        integer*4       node_number
        integer*4       internal_buffer_size
        integer*4       compress4

        logical*1       diagnostic_mode


        call linchk (lun,1)

        write(lun,35) ppr
35      format(' ',t8,'PPR',t24,z8.8)

        if (.not. diagnostic_mode) then

        if (lib$extzv(3,1,psr) .eq. 1) then

        node_number = lib$extzv(0,8,ppr)

        call linchk (lun,1)

        write(lun,40) node_number
40      format(' ',t40,'"CI" NODE #',i<compress4 (node_number)>,'.')

        internal_buffer_size = lib$extzv(16,12,ppr)

        call linchk (lun,1)

        write(lun,45) internal_buffer_size
45      format(' ',t40,'INTERNAL BUFFER SIZE, ',
        1 i<compress4 (internal_buffer_size)>,'. BYTES')

        call linchk (lun,1)

        if (lib$extzv(31,1,ppr) .eq. 0) then

        write(lun,50) '16'
50      format(' ',t40,a,' NODE MAXIMUM THIS "CI"')
        else

        write(lun,50) '224'
        endif
        endif
        endif

        return

        End
```

```
PROGRAM SECTIONS

    Name                                     Bytes   Attributes

  0 $CODE                                      339   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                     154   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                     116   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG

    Total Space Allocated                      609

ENTRY POINTS

    Address   Type  Name

  0-00000000        CI_PPR

VARIABLES

    Address   Type  Name                                   Address   Type  Name

 AP-00000010@ L*1   DIAGNOSTIC_MODE                      2-00000004  I*4   INTERNAL_BUFFER_SIZE
 AP-00000004@ L*1   LUN                                  2-00000000  I*4   NODE_NUMBER
 AP-00000008@ I*4   PPR                                 AP-0000000C@ I*4   PSR

LABELS

    Address   Label        Address   Label        Address   Label        Address   Label

  1-00000021  35'       1-00000031  40'       1-0000004D  45'       1-0000007A  50'

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name             Type  Name             Type  Name

   I*4  COMPRESS4         I*4  LIB$EXTZV              LINCHK
```

```



        Subroutine CI_CONTROL_STORE_MISMATCH (lun,pmadr,pmdatr,
        1 correct_control_store_value,padriver_error_type_code,diagnostic_mode)


        byte            lun

        integer*4       pmadr
        integer*4       pmdatr
        integer*4       correct_control_store_value
        integer*4       padriver_error_type_code

        logical*1       diagnostic_mode


        if (
        1 lib$extzv(8,7,padriver_error_type_code) .eq. 1
        1 .and.
        1 lib$extzv(0,8,padriver_error_type_code) .eq. 0
        1 ) then

        call linchk (lun,1)

        write(lun,55) pmadr
55      format(' ',t8,'PMADR',t24,z8.8)

        if (.not. diagnostic_mode) then

        call linchk (lun,4)

        write(lun,60) pmdatr,correct_control_store_value
60      format(' ',t8,'PMDATR',t24,z8.8,/,
        1 t40,'BAD DATA',/,
        1 t24,z8.8,/,
        1 t40,'GOOD DATA')
        else

        call linchk (lun,2)

        write(lun,65) pmdatr,correct_control_store_value
65      format(' ',t8,'PMDATR',t24,z8.8,/,
        1 t24,z8.8)
        endif
        endif

        return

        End
```

```
PROGRAM SECTIONS

    Name                                    Bytes   Attributes

  0 $CODE                                     217   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                    119   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                     68   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG

    Total Space Allocated                     404


ENTRY POINTS

    Address  Type  Name

  0-00000000       CI_CONTROL_STORE_MISMATCH


VARIABLES

    Address  Type  Name                                Address  Type  Name

 AP-00000010@ I*4  CORRECT_CONTROL_STORE_VALUE     AP-00000018@ L*1  DIAGNOSTIC_MODE
 AP-00000004@ L*1  LUN                             AP-00000014@ I*4  PADRIVER_ERROR_TYPE_CODE
 AP-00000008@ I*4  PMADR                           AP-0000000C@ I*4  PMDATR


LABELS

    Address   Label       Address   Label       Address   Label

  1-00000018  55'       1-0000002A  60'       1-0000005E  65'


FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name              Type  Name

   I*4  LIB$EXTZV               LINCHK
```

```
                                                H  1
                                               1c-Sep-1984 00:11:24    VAX-11 FORTRAN V3.4-56              Page  30
                                                5-Sep-1984 14:10:51    DISK$VMSMASTER:[ERF.SRC]PADRIVER.FOR;1

0001
```

```
PA

08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
```

```
        Subroutine PADRIVER_LOGMESSAGE (lun,option)

        include 'src$:msghdr.for /nolist'
        include 'src$:emblmdef.for /nolist'


        byte            lun

        character*1     option

        integer*4       padriver_error_type_code
        integer*4       ucb$l_errcnt
        integer*4       remote_station_address031
        integer*4       remote_system_id031
        integer*4       first_68_bytes_of_message(17)
        integer*4       error_subtype
        integer*4       error_type
        integer*4       path
        integer*4       remote_node_number
        integer*4       operation_code
        integer*4       compress4

        logical*1       response

        integer*2       local_station_address(3)
        integer*2       local_system_id(3)
        integer*2       remote_station_address(3)
        integer*2       remote_system_id(3)
        integer*2       remote_station_address3247, hsc$w_msglen
        integer*2       remote_system_id3247, hsc$w_errlog_dg

        byte            ppd$b_port
        byte            ppd$b_status
        byte            ppd$b_opc
        byte            ppd$b_flags

        equivalence     (remote_station_address(3),remote_station_address3247)
        equivalence     (remote_station_address,remote_station_address031)
        equivalence     (remote_system_id,remote_system_id031)
        equivalence     (remote_system_id(3),remote_system_id3247)

        equivalence     (emb$b_lm_msgtxt(1),padriver_error_type_code)
        equivalence     (emb$b_lm_msgtxt(5),ucb$l_errcnt)
        equivalence     (emb$b_lm_msgtxt(9),local_station_address)
        equivalence     (emb$b_lm_msgtxt(15),local_system_id)
        equivalence     (emb$b_lm_msgtxt(21),remote_station_address)
        equivalence     (emb$b_lm_msgtxt(27),remote_system_id)
        equivalence     (emb$b_lm_msgtxt(33),ppd$b_port)
        equivalence     (emb$b_lm_msgtxt(34),ppd$b_status)
        equivalence     (emb$b_lm_msgtxt(35),ppd$b_opc)
        equivalence     (emb$b_lm_msgtxt(36),ppd$b_flags)
        equivalence     (emb$b_lm_msgtxt(37),first_68_bytes_of_message)
        equivalence     (emb$b_lm_msgtxt(39),hsc$w_errlog_dg)
        equivalence     (emb$b_lm_msgtxt(49),hsc$t_nodename)
        equivalence     (emb$b_lm_msgtxt(57),hsc$w_msglen)
        equivalence     (emb$b_lm_msgtxt(59),hsc$t_message)

```

PA
PR
EN
VA
A
A
AR

```

        character*(200) hsc$t_message
        character*(8)   hsc$t_nodename
        character*(50)  Message_string
        character*(*)   Msg1,msg2,msg3,msg4,msg5,msg6,msg7,msg8,msg9,msg10,
        1 msg11,msg12,msg13

        Integer*4       str$position, start_index, end_loc
        Character*1     sub_str
        Data sub_str/13/

        parameter       (
        1 msg1  = 'DATA CABLE(S) CHANGE OF STATE',
        2 msg2  = 'PATH #0. HAS GONE FROM GOOD TO BAD',
        3 msg3  = 'PATH #1. HAS GONE FROM GOOD TO BAD',
        4 msg4  = 'PATH #0. HAS GONE FROM BAD TO GOOD',
        5 msg5  = 'PATH #1. HAS GONE FROM BAD TO GOOD',
        6 msg6  = 'CABLES HAVE GONE FROM UNCROSSED TO CROSSED',
        7 msg7  = 'CABLES HAVE GONE FROM CROSSED TO UNCROSSED',
        8 msg8  = 'PATH #0. LOOPBACK HAS GONE FROM GOOD TO BAD',
        9 msg9  = 'PATH #1. LOOPBACK HAS GONE FROM GOOD TO BAD',
        2 msg10 = 'PATH #0. LOOPBACK HAS BECOME GOOD, UNCROSSED',
        2 msg11 = 'PATH #1. LOOPBACK HAS BECOME GOOD, UNCROSSED',
        2 msg12 = 'PATH #0. HAS BECOME WORKING BUT CROSSED TO PATH #1.',
        2 msg13 = 'PATH #1. HAS BECOME WORKING BUT CROSSED TO PATH #0.')

        if (option .eq. 'S') call frctof (lun)

        call header (lun)

        call logger (lun,'ERL$LOGMESSAGE ENTRY')

        error_subtype = lib$extzv(0,8,padriver_error_type_code)
        error_type = lib$extzv(8,7,padriver_error_type_code)

        call linchk (lun,2)

        if (error_type .eq. 64) then

            if (error_subtype .eq. 0) then
               write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
        1      emb$w_lm_unit,'UNRECOGNIZED ''SCA'' PACKET'
10             format('0',' ','CI SUB-SYSTEM, _',a,
        1      i<compress4 (lib$extzv(0,16,emb$w_lm_unit))>,': - ',
        1      a,:i<compress4 (error_subtype)>,:a)

            else if (error_subtype .eq. 1) then
                 write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
        1        emb$w_lm_unit,'PORT HAS CLOSED ''VIRTUAL CIRCUIT'''

            else if (error_subtype .eq. 2) then
                 write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
        1        emb$w_lm_unit,'SOFTWARE SHUTTING DOWN PORT'

            else if (error_subtype .eq. 3) then
                 write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
        1        emb$w_lm_unit,'SOFTWARE IS CLOSING ''VIRTUAL CIRCUIT'''
```

```

             else if (error_subtype .eq. 4) then
                  write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
        1         emb$w_lm_unit,'RECEIVED ''CONNECT'' WITHOUT PATH-BLOCK'

             else if (error_subtype .eq. 5) then
                  write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
        1         emb$w_lm_unit,'INAPPROPRIATE ''SCA'' CONTROL MESSAGE'

             else if (error_subtype .eq. 6) then
                  write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
        1         emb$w_lm_unit,'NO PATH-BLOCK DURING ''VIRTUAL CIRCUIT'' CLOSE'

             else if (error_subtype .eq. 7) then
                  write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
        1         emb$w_lm_unit,'HSC ERROR LOGGING DATAGRAM RECEIVED.'

             else if (error_subtype .eq. 8) then
                  write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
        1         emb$w_lm_unit,'REMOTE SYSTEM CONFLICTS WITH KNOWN SYSTEM.'

             endif

        else if (error_type .eq. 65) then
             message_string = msg1

             write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
        1    emb$w_lm_unit,message_string

             call linchk (lun,2)

             Go to (310,315,320,325,330,335,
        1           340,345,350,355,360) error_subtype

             if (error_subtype .eq. 0) then
                message_string = msg2
                go to 990
             else
c               ERROR_SUBTYPE value did not match any known value so
                go to 992
             endif

310          message_string = msg3
             go to 990

315          message_string = msg4
             go to 990

320          message_string = msg5
             go to 990

325          message_string - msg6
             go to 990

330          message_string = msg7
             go to 990

```

```
335             message_string = msg8
                go to 990

340             message_string = msg9
                go to 990

345             message_string = msg10
                go to 990

350             message_string = msg11
                go to 990

355             message_string = msg12
                go to 990

360             message_string = msg13

990             write(lun,12) message_string
12              format(/' ',t8,a)
992             continue
          else

                write(lun,25) embSt_lm_name(1:emb$b_lm_namlng),
          1     emb$w_lm_unit,'"PADRIVER" ERROR TYPE #',error_type,
          1     '., ERROR SUB-TYPE #',error_subtype,'.'
25              format(/' ','CI SUB-SYSTEM, _',a,
          1     i<compress4 (lib$extzv(0,16,emb$w_lm_unit))>,': - ',
          1     a,i<compress4 (error_type)>,a,i<compress4 (error_subtype)>,a)
          endif

          call padriver_initialization (lun,padriver_error_type_code)

          if (option .eq. 'B') return

          call linchk (lun,2)

          write(lun,30) 'LOCAL STATION ADDRESS, ',
          1 (local_station_address(i),i = 3,1,-1)
30        format(/' ',t8,a,3(z4.4),' (HEX)')

          call linchk (lun,2)

          write(lun,30) 'LOCAL SYSTEM ID, ',(local_system_id(i),i = 3,1,-1)

          message = .false.

          call linchk (lun,2)

          if (remote_station_address031 - 0) 35,40,40

35        if (remote_station_address3247 - 0) 45,40,40

40        write(lun,30) 'REMOTE STATION ADDRESS, ',
          1 (remote_station_address(i),i = 3,1,-1)

          message = .true.
```

```
      goto 55

 45   write(lun,50) 'REMOTE STATION ADDRESS UNAVAILABLE'
 50   format(/' ',t8,a)

 55   continue

      call linchk (lun,2)

      if (remote_system_id031 - 0) 70,65,70

 65   if (remote_system_id3247 - 0) 70,75,70

 70   write(lun,30) 'REMOTE SYSTEM ID, ',(remote_system_id(i),i = 3,1,-1)

      goto 80

 75   write(lun,50) 'REMOTE SYSTEM ID UNAVAILABLE'

 80   continue

      call linchk (lun,1)

      write(lun,85)
 85   format(' ',:)

      call ucb$b_ertcnt (lun,lib$extzv(16,8,padriver_error_type_code))

      call ucb$b_ertmax (lun,lib$extzv(24,8,padriver_error_type_code))

      call ucb$w_errcnt (lun,ucb$l_errcnt)

      if (.NOT. message) return

      call linchk (lun,1)

      write(lun,90) ppd$b_port
 90   format(' ',t8,'PPD$B_PORT',t30,z2.2)

      remote_node_number = lib$extzv(0,8,ppd$b_port)

      call linchk (lun,1)

      write(lun,95) remote_node_number
 95   format(' ',t40,'REMOTE NODE #',i<compress4 (remote_node_number)>,
     1 '.')

      call linchk (lun,1)

      write(lun,97) ppd$b_status
 97   format(' ',t8,'PPD$B_STATUS',t30,z2.2)

      response = .false.

      if (ppd$b_status .ne. 0) response = .true.

      if (lib$extzv(5,1,ppd$b_opc) .eq. 1) response = .true.
```

```

                if (response) then

                call status (lun,ppd$b_status)
                endif

                call linchk (lun,1)

                write(lun,99) ppd$b_opc
        99      format(' ',t8,'PPD$B_OPC',t30,z2.2)

                operation_code = lib$extzv(0,8,ppd$b_opc)

                call linchk (lun,1)

                if (operation_code .eq. 1) then

                   if (.not. response) then

                     write(lun,105) 'SNDDG'
        105          format(' ',t40,a)
                   else

                     write(lun,105) 'DGSNT'
                   endif

                call flags_pf (lun,ppd$b_flags)

                else if (operation_code .eq. 2) then

                if (.not. response) then

                write(lun,105) 'SNDMSG'
                else

                write(lun,105) 'MSGSNT'
                endif

                call flags_pf (lun,ppd$b_flags)

                else if (operation_code .eq. 3) then

                if (.not. response) then

                write(lun,105) 'RETCNF'
                else

                write(lun,105) 'CNFRET'
                endif

                call flags (lun,ppd$b_flags)

                else if (operation_code .eq. 5) then

                if (.not. response) then

                write(lun,105) 'REQID'
```

```
            else

            write(lun,105) 'IDREQ'
            endif

            call flags (lun,ppd$b_flags)

            else if (operation_code .eq. 6) then

            if (.not. response) then

            write(lun,105) 'SNDRST'
            else

            write(lun,105) 'RSTSNT'
            endif

            call flags_f (lun,ppd$b_flags)

            else if (operation_code .eq. 7) then

            if (.not. response) then

            write(lun,105) 'SNDSTRT'
            else

            write(lun,105) 'STRTSNT'
            endif

            call flags_ds (lun,ppd$b_flags)

            else if (operation_code .eq. 8) then

            if (.not. response) then

            write(lun,105) 'REQDAT0'
            else

            write(lun,105) 'DATREQ0'
            endif

            call flags_p (lun,ppd$b_flags)

            else if (operation_code .eq. 9) then

            if (.not. response) then

            write(lun,105) 'REQDAT1'
            else

            write(lun,105) 'DATREQ1'
            endif

            call flags_p (lun,ppd$b_flags)

            else if (operation_code .eq. 10) then

            if (.not. response) then

            write(lun,105) 'REQDAT2'
            else

            write(lun,105) 'DATREQ2'
            endif

            call flags_p (lun,ppd$b_flags)

            else if (operation_code .eq. 13) then

            if (.not. response) then

            write(lun,105) 'SNDLB'
            else

            write(lun,105) 'LBSNT'
            endif

            call flags_pf (lun,ppd$b_flags)

            else if (operation_code .eq. 14) then

            if (.not. response) then

            write(lun,105) 'REQMDAT'
            else

            write(lun,105) 'MDATREQ'
            endif

            call flags_p (lun,ppd$b_flags)

            else if (operation_code .eq. 16) then

            if (.not. response) then

            write(lun,105) 'SNDDAT'
            else

            write(lun,105) 'DATSNT'
            endif

            call flags_p (lun,ppd$b_flags)

            else if (operation_code .eq. 17) then

            if (.not. response) then

            write(lun,105) 'RETDAT'
            else

            write(lun,105) 'DATRET'
            endif

            call flags_p (lun,ppd$b_flags)
```

```

          else if (operation_code .eq. 18) then

          if (.not. response) then

          write(lun,105) 'SNDMDAT'
          else

          write(lun,105) 'MDATSNT'
          endif

          call flags_p (lun,ppd$b_flags)

          else if (operation_code .eq. 24) then

          if (.not. response) then

          write(lun,105) 'INVTC'
          else

          write(lun,105) 'TCINV'
          endif

          call flags (lun,ppd$b_flags)

          else if (operation_code .eq. 25) then

          if (.not. response) then

          write(lun,105) 'SETCKT'
          else

          write(lun,105) 'CKTSET'
          endif

          call flags (lun,ppd$b_flags)

          else if (operation_code .eq. 26) then

          if (.not. response) then

          write(lun,105) 'RDCNT'
          else

          write(lun,105) 'CNTRD'
          endif

          call flags (lun,ppd$b_flags)

          else if (operation_code .eq. 33) then

          write(lun,105) 'DGREC'

          call flags_pf (lun,ppd$b_flags)

          else if (operation_code .eq. 34) then

```

```
                write(lun,105) 'MSGREC'

                call flags_pf (lun,ppd$b_flags)

                else if (operation_code .eq. 35) then

                write(lun,105) 'CNFREC'

                call linchk (lun,1)

                write(lun,111) ppd$b_flags

                else if (operation_code .eq. 49) then

                write(lun,105) 'DATREC'

                call linchk (lun,1)

                write(lun,111) ppd$b_flags

                else if (operation_code .eq. 45) then

                write(lun,105) 'LBREC'

                call linchk (lun,1)

                write(lun,111) ppd$b_flags

                path = lib$extzv(1,2,ppd$b_flags)

                path = path - 1

                if (path .ge. 0) then

                call linchk (lun,1)

                write(lun,110) 'LOOPBACK RECEIVED ON PATH #',path
        110     format(' ',t40,a,i<compress4 (path)>,'.')
                endif

                else if (operation_code .eq. 43) then

                write(lun,105) 'IDREC'

                call linchk (lun,1)

                write(lun,111) ppd$b_flags
        111     format(' ',t8,'PPD$B_FLAGS',t30,z2.2)

                path = lib$extzv(1,2,ppd$b_flags)

                path = path - 1

                if (path .GE. 0) then

                call linchk (lun,1)

                write(lun,110) 'RECEIVE PATH #',path

                else

                call linchk (lun,1)

                write(lun,112) 'RECEIVE'
        112     format(' ',t40,a,' PATH, INTERNAL LOOPBACK')
                endif

                path = lib$extzv(4,2,ppd$b_flags)

                path = path - 1

                if (path .GE. 0) then

                call linchk (lun,1)

                write(lun,110) 'SEND PATH #',path

                else

                call linchk (lun,1)

                write(lun,112) 'SEND'
                endif

                else if (operation_code .eq. 36) then

                write(lun,105) 'MCNFREC'

                call linchk (lun,1)

                write(lun,111) ppd$b_flags

                else if (operation_code .eq. 51) then

                write(lun,105) 'MDATREC'

                call linchk (lun,1)

                write(lun,111) ppd$b_flags

                else if (operation_code .eq. 11) then

                write(lun,105) 'ID'

                call linchk (lun,1)

                write(lun,111) ppd$b_flags

                else if (operation_code .eq. 19) then

                write(lun,105) 'RETMDAT'

                call linchk (lun,1)
                write(lun,111) ppd$b_flags

                else if (operation_code .eq. 4) then

                write(lun,105) 'MCNF'

                call linchk (lun,1)

                write(lun,111) ppd$b_flags
                else

                if (.not. response) then

                write(lun,115) 'COMMAND, ',operation_code,'.'
        115     format(' ',t40,'PORT ',a,i<compress4 (operation_code)>,a)

                call linchk (lun,1)

                write(lun,111) ppd$b_flags
                else

                write(lun,115) 'RESPONSE, ',operation_code,'.'

                call linchk (lun,1)

                write(lun,111) ppd$b_flags
                endif
                endif

                if (message) then

                do 123,i = 1,17

                if (first_68_bytes_of_message(i) .ne. 0) goto 124

        123     continue

                goto 140

        124     If ((error_subtype .eq. 7) .AND. (error_type .eq. 64)) then

                    If ((hsc$w_errlog_dg .eq. 5) .AND. (hsc$w_msglen .gt. 2)) then
                       call linchk (lun,3)
                       write(lun,85)    ! Write a blank line
                       write(lun,125) '"HSC" ERROR LOG DATAGRAM'
                       write(lun,125) hsc$t_nodename(1:8)

                       start_index = 1
                       end_loc = hsc$w_msglen - 2

        1111           j = STR$POSITION (hsc$t_message, sub_str, start_index)

        c              If the search find the sub string past the end of the message
        c               then the search failed.

                       if (j .gt. (hsc$w_msglen - 2) ) then
                        j = 0
                endif

                if (j .eq. 0) then
                  end_loc = j
                  j = hsc$w_msglen - 1
                end if

                write(lun,2126) hsc$t_message(start_index:(j-1))
2126            format (' ',t8,a)

                if (end_loc .ne. 0) then
                   start_index = j + 2
                   goto 1111
                end if
             else
                call linchk (lun,3)
                write (lun,125) 'UNRECOGNIZED ''HSC'' ERROR LOG DATAGRAM'
             endif

        else

            call linchk (lun,3)
            write(lun,125) '''CI'' MESSAGE'
125         format(/' ',t8,a)

            write(lun,85)

            do 135,i = 1,17

            call linchk (lun,1)

            write(lun,130) first_68_bytes_of_message(i)
130         format(' ',t24,z8.8)

135         continue
        endif

140     continue
        endif

        return
        End
```

```
PROGRAM SECTIONS

    Name                                         Bytes   Attributes

  0 $CODE                                         6069   PIC CON REL LCL   SHR   EXE    RD NOWRT LONG
  1 $PDATA                                        1877   PIC CON REL LCL   SHR NOEXE    RD NOWRT LONG
  2 $LOCAL                                        1112   PIC CON REL LCL NOSHR NOEXE    RD   WRT LONG
  3 EMB                                            512   PIC OVR REL GBL   SHR NOEXE    RD   WRT LONG

    Total Space Allocated                         9570


ENTRY POINTS

    Address   Type  Name

  0-00000000        PADRIVER_LOGMESSAGE


VARIABLES

    Address   Type  Name                                     Address   Type  Name

  3-00000010  L*1   EMB$B_LM_CLASS                         3-00000014  L*1   EMB$B_LM_NAMLNG
  3-00000011  L*1   EMB$B_LM_TYPE                          3-00000000  I*4   EMB$L_HD_SID
  3-00000015  CHAR  EMB$T_LM_NAME                          3-00000004  I*2   EMB$W_HD_ENTRY
  3-0000000E  I*2   EMB$W_HD_ERRSEQ                        3-00000024  I*2   EMB$W_LM_MSGTYP
  3-00000012  I*2   EMB$W_LM_UNIT                          2-0000004C  I*4   END_LOC
  2-00000034  I*4   ERROR_SUBTYPE                          2-00000038  I*4   ERROR_TYPE
  3-00000060  CHAR  HSC$T_MESSAGE                          3-00000056  CHAR  HSC$T_NODENAME
  3-0000004C  I*2   HSC$W_ERRLOG_DG                        3-0000005E  I*2   HSC$W_MSGLEN
  2-00000050  I*4   I                                      2-00000058  I*4   J
 AP-00000004@ L*1   LUN                                    2-00000054  I*4   MESSAGE
  2-00000001  CHAR  MESSAGE_STRING                         2-00000044  I*4   OPERATION_CODE
 AP-00000008@ CHAR  OPTION                                 3-00000026  I*4   PADRIVER_ERROR_TYPE_CODE
  2-0000003C  I*4   PATH                                   3-00000049  L*1   PPD$B_FLAGS
  3-00000048  L*1   PPD$B_OPC                              3-00000046  L*1   PPD$B_PORT
  3-00000047  L*1   PPD$B_STATUS                           2-00000040  I*4   REMOTE_NODE_NUMBER
  3-0000003A  I*4   REMOTE_STATION_ADDRESS031              3-0000003E  I*2   REMOTE_STATION_ADDRESS3247
  3-00000040  I*4   REMOTE_SYSTEM_ID031                    3-00000044  I*2   REMOTE_SYSTEM_ID3247
  2-00000000  L*1   RESPONSE                               2-00000048  I*4   START_INDEX
  2-00000033  CHAR  SUB_STR                                3-0000002A  I*4   UCB$L_ERRCNT


ARRAYS

    Address   Type  Name                                       Bytes  Dimensions

  3-00000000  L*1   EMB                                          512  (0:511)
  3-00000026  L*1   EMB$B_LM_MSGTXT                              460  (460)
  3-00000006  I*4   EMB$Q_HD_TIME                                  8  (2)
  3-0000004A  I*4   FIRST_68_BYTES_OF_MESSAGE                     68  (17)
  3-0000002E  I*2   LOCAL_STATION_ADDRESS                          6  (3)
  3-00000034  I*2   LOCAL_SYSTEM_ID                                6  (3)
  3-0000003A  I*2   REMOTE_STATION_ADDRESS                         6  (3)
  3-00000040  I*2   REMOTE_SYSTEM_ID                               6  (3)
```

```
LABELS

    Address    Label        Address    Label        Address    Label        Address    Label        Address    Label        Address    Label

  1-000003D9   10'        1-00000407   12'        1-0000040F   25'        1-00000442   30'             **        35        0-000005CE   40
  0-00000605   45         1-0000045A   50'        0-00000628   55              **      65        0-00000647   70        0-0000067A   75
  0-0000069D   80         1-00000462   85'        1-00000467   90'        1-0000047E   95'       1-0000049C   97'       1-000004B5   99'
  1-000004CB   105'       1-000004D2   110'       1-000004E2   111'       1-000004FA   112'      1-0000051B   115'           **       123
  0-00001527   124        1-00000537   125'       1-0000053F   130'            **      135       0-0000171C   140       0-000003E3   310
  0-000003F0   315        0-000003FD   320        0-0000040A   325        0-00000417   330       0-00000424   335       0-00000431   340
  0-0000043E   345        0-0000044B   350        0-00000458   355        0-00000463   360       0-0000046C   990       0-0000048E   992
  0-000015D8   1111       1-00000530   2126'

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name                                   Type  Name                                   Type  Name

   I*4  COMPRESS4                                    FLAGS                                        FLAGS_DS
        FLAGS_F                                      FLAGS_P                                      FLAGS_PF
        FRCTOF                                       HEADER                                  I*4  LIB$EXTZV
        LINCHK                                       LOGGER                                       PADRIVER_INITIALIZATION
        STATUS                                  I*4  STR$POSITION                                 UCB$B_ERTCNT
        UCB$B_ERTMAX                                 UCB$W_ERRCNT


        Subroutine FLAGS (lun,ppd$b_flags)

        byte            lun
        byte            ppd$b_flags

        integer*4       path_select


        call linchk (lun,1)

        write(lun,5) ppd$b_flags
5       format(' ',t8,'PPD$B_FLAGS',t30,z2.2)

        if (lib$extzv(0,1,ppd$b_flags) .eq. 1) then

        call linchk (lun,1)

        write(lun,10) 'RESPONSE QUEUE BIT'
10      format(' ',t40,a)
        endif

        path_select = lib$extzv(1,2,ppd$b_flags)

        call linchk (lun,1)

        if (path_select .eq. 1) then
        write(lun,10) 'SELECT PATH #0.'

        else if (path_select .eq. 2) then
        write(lun,10) 'SELECT PATH #1.'

        endif

        return
        End
```

```
PROGRAM SECTIONS

    Name                                 Bytes   Attributes

  0 $CODE                                  230   PIC CON REL LCL    SHR   EXE   RD NOWRT LONG
  1 $PDATA                                  91   PIC CON REL LCL    SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                  72   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG

    Total Space Allocated                  393


ENTRY POINTS

    Address   Type  Name

  0-00000000        FLAGS


VARIABLES

    Address   Type  Name                 Address   Type  Name                Address   Type  Name

 AP-00000004@ L*1   LUN              2-00000000  I*4   PATH_SELECT     AP-00000008@ L*1   PPD$B_FLAGS


LABELS

    Address   Label       Address   Label

  1-0000003C  5'       1-00000054  10'


FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name              Type  Name

   I*4  LIB$EXTZV               LINCHK
```

```

        Subroutine FLAGS_PF (lun,ppd$b_flags)

        byte            lun
        byte            ppd$b_flags


        call flags (lun,ppd$b_flags)

        call linchk (lun,1)

        if (lib$extzv(8,1,ppd$b_flags) .eq. 1) then

        write(lun,5) '"NIBBLE" PACKED'
  5     format(' ',t40,a)
        else

        write(lun,5) '"LONGWORD" PACKED'
        endif

        return

        End
```

```
PROGRAM SECTIONS

    Name                                  Bytes   Attributes

  0 $CODE                                   127   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                   47   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                   56   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG

    Total Space Allocated                   230

ENTRY POINTS

    Address   Type  Name

  0-00000000        FLAGS_PF

VARIABLES

    Address   Type  Name                 Address   Type  Name

 AP-00000004@ L*1   LUN              AP-00000008@ L*1   PPD$B_FLAGS
```

```
LABELS

    Address   Label

  1-00000028  5'

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name            Type  Name             Type  Name

        FLAGS            I*4  LIB$EXTZV              LINCHK



        Subroutine FLAGS_P (lun,ppd$b_flags)


        byte            lun
        byte            ppd$b_flags

        integer*4       packet_multiple
        integer*4       packet_base_size
        integer*4       packet_size
        integer*4       compress4


        call flags (lun,ppd$b_flags)

        packet_multiple = lib$extzv (5,3,ppd$b_flags)
        packet_base_size = 512

        if (lib$extzv(8,1,ppd$b_flags) .eq. 1) packet_base_size = 576

        packet_size = packet_base_size * (packet_multiple + 1)

        call linchk (lun,2)

        write(lun,5) 'PACKET MULTIPLE ',packet_multiple,
        1 ' - PACKET SIZE ',packet_size,'. BYTES'
5       format(' ',t40,a,i<compress4 (packet_multiple)>,/,
        1 t40,a,i<compress4 (packet_size)>,a)

        return

        End
```

```
PROGRAM SECTIONS

    Name                                     Bytes   Attributes

  0 $CODE                                      198   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                      82   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                     108   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG

    Total Space Allocated                      388


ENTRY POINTS

    Address   Type  Name

  0-00000000        FLAGS_P


VARIABLES

    Address   Type  Name                                 Address   Type  Name

 AP-00000004@ L*1   LUN                                2-00000004  I*4   PACKET_BASE_SIZE
  2-00000000  I*4   PACKET_MULTIPLE                    2-00000008  I*4   PACKET_SIZE
 AP-00000008@ L*1   PPD$B_FLAGS


LABELS

    Address   Label

  1-0000003A  5'


FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name            Type  Name            Type  Name            Type  Name

   I*4  COMPRESS4             FLAGS            I*4  LIB$EXTZV             LINCHK
```

```


        Subroutine FLAGS_F (lun,ppd$b_flags)

        byte            lun
        byte            ppd$b_flags


        call flags (lun,ppd$b_flags)

        if (lib$extzv(8,1,ppd$b_flags) .eq. 1) then

        call linchk (lun,1)

        write(lun,5) 'FORCE RESET'
   5    format(' ',t40,a)
        endif

        return

        End
```

```
PROGRAM SECTIONS

    Name                                    Bytes   Attributes

  0 $CODE                                      91   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                     26   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                     48   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG

    Total Space Allocated                     165

ENTRY POINTS

    Address   Type  Name

  0-00000000        FLAGS_F

VARIABLES

    Address   Type  Name                  Address   Type  Name

 AP-00000004@ L*1   LUN              AP-00000008@ L*1   PPD$B_FLAGS

LABELS

    Address    Label

  1-00000013   5'
```

```
FLAGS_F                                           E  3
                                                 16-Sep-1984 00:11:24    VAX-11 FORTRAN V3.4-56              Page  53
                                                  5-Sep-1984 14:10:51    DISK$VMSMASTER:[ERF.SRC]PADRIVER.FOR,1

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name              Type  Name             Type  Name

        FLAGS              I*4  LIB$EXTZV              LINCHK
```

```



            Subroutine FLAGS_DS (lun,ppd$b_flags)

            byte            lun
            byte            ppd$b_flags


            call flags (lun,ppd$b_flags)

            if (lib$extzv(8,1,ppd$b_flags) .eq. 1) then

            call linchk (lun,1)

            write(lun,5) 'DEFAULT STARTING ADDRESS'
    5       format(' ',t40,a)
            endif

            return

            End
```

```
PROGRAM SECTIONS

    Name                                 Bytes   Attributes

  0 $CODE                                   91   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                  39   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                  48   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG

    Total Space Allocated                  178

ENTRY POINTS

    Address   Type  Name

  0-00000000        FLAGS_DS

VARIABLES

    Address   Type  Name               Address   Type  Name

 AP-00000004@ L*1  LUN             AP-00000008@ L*1  PPD$B_FLAGS
```

```
LABELS

    Address   Label

  1-00000020  5'

FUNC' ONS AND SUBROUTINES REFERENCED

  Type  Name             Type  Name             Type  Name

        FLAGS             I*4  LIB$EXTZV              LINCHK


        Subroutine STATUS (lun,ppd$b_status)


        byte            lun
        byte            ppd$b_status

        integer*4       type
        integer*4       pth_1
        integer*4       pth_0
        integer*4       sub_type
        integer*4       compressc

        character*5     v1status(0:0)
        data            v1status(0)     /'FAIL*'/

        character*20    path_status(0:3)
        data            path_status(0)  /'"ACK" OR NOT USED*'/
        data            path_status(1)  /'"NAK"*'/
        data            path_status(2)  /'NO RESPONSE*'/
        data            path_status(3)  /'ARBITRATION TIMEOUT*'/

        character*25    subtype(0:3)
        data            subtype(0)      /'PACKET SIZE VIOLATION*'/
        data            subtype(1)      /'UNRECOGNIZED PACKET*'/
        data            subtype(2)      /'INVALID DESTINATION PORT*'/
        data            subtype(3)      /'UNRECOGNIZED COMMAND*'/

        character*27    types(0:6)
        data            types(0)        /'NORMAL*'/
        data            types(1)        /'VIRTUAL CIRCUIT CLOSED*'/
        data            types(2)        /'INVALID BUFFER NAME*'/
        data            types(3)        /'BUFFER LENGTH VIOLATION*'/
        data            types(4)        /'ACCESS CONTROL VIOLATION*'/
        data            types(5)        /'NO PATH*'/
        data            types(6)        /'BUFFER MEMORY SYSTEM ERROR*'/


        type = lib$extzv(5,3,ppd$b_status)
        pth_1 = lib$extzv(3,2,ppd$b_status)
        pth_0 = lib$extzv(1,2,ppd$b_status)
        sub_type = lib$extzv(1,4,ppd$b_status)

        call output (lun,ppd$b_status,v1status,0,0,0,'0')

        if (type .eq. 7) then

        call linchk (lun,1)

        write(lun,10) subtype(sub_type)
10      format(' ',t40,a<compressc (subtype(sub_type))>)
        else

        call linchk (lun,2)

        write(lun,15) '0',path_status(pth_0)
```

```
      15      format(' ',t40,'PATH #',a,'.. ',
              1 a<compressc (path_status(pth_0))>)

              pth_0 = pth_1

              write(lun,15) '1',path_status(pth_1)

              call linchk (lun,1)

              write(lun,20) types(type)
      20      format(' ',t40,a<compressc (types(type))>)
              endif

              return

              End



PROGRAM SECTIONS

    Name                                    Bytes   Attributes

  0 $CODE                                     488   PIC CON REL LCL   SHR   EXE    RD NOWRT LONG
  1 $PDATA                                     77   PIC CON REL LCL   SHR NOEXE    RD NOWRT LONG
  2 $LOCAL                                    616   PIC CON REL LCL NOSHR NOEXE    RD   WRT LONG

    Total Space Allocated                    1181

ENTRY POINTS

    Address   Type  Name

  0-00000000        STATUS

VARIABLES

    Address   Type  Name                 Address   Type  Name               Address   Type  Name                Address   Type  Name

 AP-00000004@ L*1   LUN               AP-00000008@ L*1   PPD$B_STATUS     2-00000180  I*4   PTH_0            2-0000017C  I*4   PTH_1
  2-00000184  I*4   SUB_TYPE            2-00000178  I*4   TYPE

ARRAYS

    Address   Type  Name                   Bytes  Dimensions

  2-00000005  CHAR PATH_STATUS               80   (0:3)
  2-00000055  CHAR SUBTYPE                  100   (0:3)
  2-000000B9  CHAR TYPES                    189   (0:6)
  2-00000000  CHAR V1STATUS                   5   (0:0)
```

```
LABELS

    Address   Label        Address   Label        Address   Label

  1-0000001B  10'        1-00000027  15'        1-00000041  20'


FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name              Type  Name              Type  Name              Type  Name

   I*4  COMPRESSC          I*4  LIB$EXTZV               LINCHK                  OUTPUT


COMMAND QUALIFIERS

  FORTRAN /LIS=LIS$:PADRIVER/OBJ=OBJ$:PADRIVER MSRC$:PADRIVER

  /CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)
  /DEBUG=(NOSYMBOLS,TRACEBACK)
  /STANDARD=(NOSYNTAX,NOSOURCE_FORM)
  /SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP)
  /F77  /NOG_FLOATING  /I4  /OPTIMIZE  /WARNINGS  /NOD_LINES  /NOCROSS_REFERENCE  /NOMACHINE_CODE  /CONTINUATIONS=19


COMPILATION STATISTICS

  Run Time:           30.65 seconds
  Elapsed Time:       66.15 seconds
  Page Faults:        424
  Dynamic Memory:     330 pages
```

0152 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PADRIVER
LIS

OUTPUT
LIS

NEW_RTN
LIS

MT_DISMT
LIS

OPNOUTFIL
LIS

OUTPUTD
LIS

NEWFILE
LIS

0153 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY