```
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
```

```
BBBBBBBB                  CCCCCCCC              DDDDDDDD      IIIIII      SSSSSSSS  PPPPPPPP
BBBBBBBB                  CCCCCCCC              DDDDDDDD      IIIIII      SSSSSSSS  PPPPPPPP
BB      BB              CC                      DD      DD      II      SS          PP      PP
BB      BB              CC                      DD      DD      II      SS          PP      PP
BB      BB              CC                      DD      DD      II      SS          PP      PP
BB      BB              CC                      DD      DD      II      SS          PP      PP
BBBBBBBB                CC                      DD      DD      II        SSSSSS    PPPPPPPP
BBBBBBBB                CC                      DD      DD      II        SSSSSS    PPPPPPPP
BB      BB              CC                      DD      DD      II              SS  PP
BB      BB              CC                      DD      DD      II              SS  PP
BB      BB              CC                      DD      DD      II              SS  PP          ....
BB      BB              CC                      DD      DD      II              SS  PP          ....
BBBBBBBB    ----------    CCCCCCCC  ----------  DDDDDDDD      IIIIII    SSSSSSSS    PP          ....
BBBBBBBB    ----------    CCCCCCCC  ----------  DDDDDDDD      IIIIII    SSSSSSSS    PP          ....

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
```

```
        SUBROUTINE BRIEF_C_DISPATCHER (Lun, Report_type, Record_length,
        1 Summary_qual, summary_flag)
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
C
C       AUTHOR: Elliott A. Drayton              CREATION DATE: 15-Dec-1982
C
C       Functional description:
C
C       This module displays brief reports for error log entries.
C
C       Modified by:
C
C       V03-014 SAR0279         Sharon A. Reynolds      26-jun-1984
C               - Added TMSCP support.
C               - Fixed a bug in the calling parameters to
C               brief_pudriver_mscp_dispatcher.
C
C       V03-013 SAR0253         Sharon A. Reynolds      23-Apr-1984
C               Added a field length for the output of the device
C               name.
C
C       V03-012 SAR0251         Sharon A. Reynolds      11-Apr-1984
C               Removed erf_unkentry and erf_unkclass messages. They
C               are no longer needed due to the way unknown entries
C               are now handled.
C
C       V03-011 SAR0213         Sharon A. Reynolds      28-Mar-1984
C               Added KMS3271 support.
C
C       V03-010 EAD0123         Elliott A. Drayton              23-Mar-1984
C               Changed PA logmessage text for subtypes 7 and 8.
```

```
C
C       V03-009 EAD0116         Elliott A. Drayton      21-Mar-1984
C               Added code to report new HSC logmessage.
C
C       V03-008 SAR0210         Sharon A. Reynolds,     10-Mar-1984
C               - Corrected the check for the call to image loader for
C               /summary support for briefs.
C
C       V03-007 SAR0208         Sharon A. Reynolds,     8-Mar-1984
C               - Fixed a bug in the output of memory entries when
C               there is more than one controller.
C               - Changed the way the call parameters for image loader
C               were passed to work with the new image loader.
C
C       V03-006 SAR0186         Sharon A. Reynolds,     17-Jan-1984
C               - Added message type '8' to the message type
C               dispatching in the logmessage routine.
C               - Added support for memory and volume summaries.
C               - Added last SYE update changes.
C
C       V03-005 SAR0173         Sharon A. Reynolds,     11-Nov-1983
C               - Added brief support for 'logmscp' entries.
C
C       V03-004 SAR0155         Sharon A. Reynolds,     11-Oct-1983
C               - Made "RH7XX" register heralds generic.
C               - Applied an SYE update that modifies the text for
C                 'pudriver' support.
C               - Made "DW7XX" register heralds generic.
C               - Made 'padriver' changes.
C               - Fixed a bug in the output of the pudriver code word
C                 text.
C
C       V03-003 SAR0141         Sharon A. Reynolds,     23-Sep-1983
C               - Put in unknown entry and device class error messages.
C               - Deleted unused routines.
C               - Fixed carriage control for ERF.
C               - Added DR11W support.
C
C       V03-002 SAR0109         Sharon A. Reynolds,     21-Jun-1983
C               Undefined support.
C
C       V03-001 SAR0033         Sharon A. Reynolds,     6-Jun-1983
C               Added brief/cryptic support for memory and mount/dismount
C               entries. Added a few comments.
C--

        Include 'SRC$:MSGHDR.FOR /NOLIST'
        Include 'SRC$:DEVERR.FOR /NOLIST'
        Include 'SRC$:EMBSPDEF.FOR /NOLIST'
        Include 'SRC$:EMBLMDEF.FOR /NOLIST'
        Include 'SRC$:SYECOM.FOR /NOLIST'
        Include 'SRC$:DEV_CLASS.FOR /NOLIST'
        Include 'src$:volmount.for /nolist'
        Include 'SRC$:SSMESSAGE.FOR /nolist'
        Include 'SRC$:PARSERDAT.FOR /nolist'

        External        erf_loaderr
```

```

        Character*1     Report_type

        Byte            Brief_P2,       ! Number of registers to output
        1               Cryp_P2,        !
        1               Brief_32        ! 1 = 32 bit reg's, 0 = 16 bit reg's

        Byte            Lun,            ! Logical unit number for output
        1               Tof,            ! Flag for call to FACTOF
        1               Exit1,          ! Flag to determine if to exit routine
        1               Write1,         ! Flag to determine if to use write
        1               Device_type     ! Value for device type
        1               Device_class    ! Value for device class

        Data            exit1/0/,tof/0/,write1/0/

        Byte            Summary_qual    ! Summary report indicator

        Integer*4       Brief_P3,       ! Address of registers
        1               Brief_P4(2),    ! Parameters for subroutines
        1               Dhead_P2(2),    ! Address of id string
        1               Linchk_P2

        Integer*4       Record_length,  ! Length of error packet being processed
        1               Compress4,      ! External function name
        1               Cpu_type,       ! Value for cpu type
        1               Reg_copy(0:11), ! General area for register values
        1               Length,         ! Length of string in MESSAGE
        1               Xfer_addr,      ! Transfer address for loadable image
        1               Status,         ! Return status
        1               Memory_summ,    ! Memory summary indicator
        1               All_summ,       ! Complete summary indicator
        1               Summary_flag    ! Summary type indicators

        Character*80    Message

        Character*23    image_name
        Data            image_name      /'SYS$SYSTEM:ERFPROC3.EXE'/

C
C RK06 and 07 definitions
C
C       Define the register mnemonics for the BRIEF and CRYPTK reports
C
        CHARACTER*4     RK_REG_HERALD(0:11)
        DATA    RK_REG_HERALD(0)        /'CS1*'/
        DATA    RK_REG_HERALD(1)        /'WC*'/
        DATA    RK_REG_HERALD(2)        /'BA*'/
        DATA    RK_REG_HERALD(3)        /'DA*'/
        DATA    RK_REG_HERALD(4)        /'CS2*'/
        DATA    RK_REG_HERALD(5)        /'DS*'/
        DATA    RK_REG_HERALD(6)        /'ER*'/
        DATA    RK_REG_HERALD(7)        /'AS*'/
        DATA    RK_REG_HERALD(8)        /'DC*'/
        DATA    RK_REG_HERALD(9)        /'MR*'/
        DATA    RK_REG_HERALD(10)       /'EC1*'/
        DATA    RK_REG_HERALD(11)       /'EC2*'/
```

```
C
C RP device type definitions
C
C       Define device types and register mnemonics for the BRIEF
C       and CRYPTK reports
C
        PARAMETER       RM03 = 6
        PARAMETER       RP06 = 5
        PARAMETER       RP05 = 4
        PARAMETER       RP04 = 3
        PARAMETER       RP07 = 7
        PARAMETER       RM80 = 13
        PARAMETER       RM05 = 15

        CHARACTER*4     REG_HERALD1(0:11)
        DATA    REG_HERALD1(0)  /'CS*'/
        DATA    REG_HERALD1(1)  /'DS*'/
        DATA    REG_HERALD1(2)  /'ER1*'/
        DATA    REG_HERALD1(3)  /'MR*'/
        DATA    REG_HERALD1(4)  /'AS*'/
        DATA    REG_HERALD1(5)  /'DA*'/
        DATA    REG_HERALD1(6)  /'DT*'/
        DATA    REG_HERALD1(7)  /'LA*'/
        DATA    REG_HERALD1(8)  /'SN*'/
        DATA    REG_HERALD1(9)  /'OF*'/
        DATA    REG_HERALD1(10) /'DC*'/
        DATA    REG_HERALD1(11) /'CC*'/

        CHARACTER*4     REG_HERALD2(0:11)
        DATA    REG_HERALD2(0)  /'CS*'/
        DATA    REG_HERALD2(1)  /'DS*'/
        DATA    REG_HERALD2(2)  /'ER1*'/
        DATA    REG_HERALD2(3)  /'MR1*'/
        DATA    REG_HERALD2(4)  /'AS*'/
        DATA    REG_HERALD2(5)  /'DA*'/
        DATA    REG_HERALD2(6)  /'DT*'/
        DATA    REG_HERALD2(7)  /'LA*'/
        DATA    REG_HERALD2(8)  /'SN*'/
        DATA    REG_HERALD2(9)  /'OF*'/
        DATA    REG_HERALD2(10) /'DC*'/
        DATA    REG_HERALD2(11) /'HR*'/


C
C RL02 Definitions
C
C       Define the register mnemonics for the BRIEF and CRYPTK reports
C
        CHARACTER*4     RL_REG_HERALD(0:3)
        DATA    RL_REG_HERALD(0)        /'CS*'/
        DATA    RL_REG_HERALD(1)        /'BA*'/
        DATA    RL_REG_HERALD(2)        /'DA*'/
        DATA    RL_REG_HERALD(3)        /'MP*'/
```

```
C
C RX01/2 Definitions
C
C       Define necessary data and the register mnemonics for the BRIEF
C       and CRYPTK reports
C
        Integer*4       extended_registers(2)

        Integer*2       extended_status(4)

        Equivalence     (extended_registers,emb$l_dv_regsav(7))
        Equivalence     (extended_registers,extended_status)

        CHARACTER*5     RX_REG_HERALD(0:5)
        data    rx_reg_herald(0)        /'CS*'/
        data    rx_reg_herald(1)        /'ES*'/
        data    rx_reg_herald(2)        /'ES 1*'/
        data    rx_reg_herald(3)        /'ES 2*'/
        data    rx_reg_herald(4)        /'ES 3*'/
        data    rx_reg_herald(5)        /'ES 4*'/

C
C TU58 Definitions
C
C       Define the register mnemonics for the BRIEF and CRYPTK reports
C
        CHARACTER*5     DD_REG_HERALD(0:3)
        DATA    DD_REG_HERALD(0)        /'RCSR*'/
        DATA    DD_REG_HERALD(1)        /'RBUF*'/
        DATA    DD_REG_HERALD(2)        /'XCSR*'/
        DATA    DD_REG_HERALD(3)        /'XBUF*'/


C
C ML11 Definitions
C
C
C       Define the register mnemonics for the BRIEF and CRYPTK reports
C
        Character*5     ml_reg_herald(0:11)
        Data    ml_reg_herald(0)        /'CS1*'/
        Data    ml_reg_herald(1)        /'DS*'/
        Data    ml_reg_herald(2)        /'ER*'/
        Data    ml_reg_herald(3)        /'MR*'/
        Data    ml_reg_herald(4)        /'AS*'/
        Data    ml_reg_herald(5)        /'DA*'/
        Data    ml_reg_herald(6)        /'DT*'/
        Data    ml_reg_herald(7)        /'SN*'/
        Data    ml_reg_herald(8)        /'E1*'/
        Data    ml_reg_herald(9)        /'E2*'/
        Data    ml_reg_herald(10)       /'EE*'/
        Data    ml_reg_herald(11)       /'EL*'/


C
C RB730 Definitions
C
```

```
C       Define the register mnemonics for the BRIEF and CRYPTK reports
C
        Character*8     rb_reg_herald(0:6)
        data    rb_reg_herald(0)        /'RB CSR*'/
        data    rb_reg_herald(1)        /'RB BAR*'/
        data    rb_reg_herald(2)        /'RB BCR*'/
        data    rb_reg_herald(3)        /'RB DAR*'/
        data    rb_reg_herald(4)        /'RB MPR*'/
        data    rb_reg_herald(5)        /'RB ECC1*'/
        data    rb_reg_herald(6)        /'RB ECC2*'/


C
C TE16, TU45 and TU77 tapes definitions
C
C       Define the register mnemonics for the BRIEF and CRYPTK reports
C
        CHARACTER*4     tape_reg_herald(0:9)
        DATA    tape_reg_herald(0)      /'CS*'/
        DATA    tape_reg_herald(1)      /'DS*'/
        DATA    tape_reg_herald(2)      /'ER*'/
        DATA    tape_reg_herald(3)      /'MR*'/
        DATA    tape_reg_HERALD(4)      /'AS*'/
        DATA    tape_reg_HERALD(5)      /'FC*'/
        DATA    tape_reg_HERALD(6)      /'DT*'/
        DATA    tape_reg_HERALD(7)      /'CK*'/
        DATA    tape_reg_HERALD(8)      /'SN*'/
        DATA    tape_reg_HERALD(9)      /'TC*'/


C
C TS11 Definitions
C
C       Define necessary data and the register mnemonics for the
C       BRIEF and CRYPTK reports
C
        Character*6     ts_reg_herald(0:6)
        data    ts_reg_herald(0)        /'BA*'/
        data    ts_reg_herald(1)        /'SR*'/
        data    ts_reg_herald(2)        /'X0*'/
        data    ts_reg_herald(3)        /'X1*'/
        data    ts_reg_herald(4)        /'X2*'/
        data    ts_reg_herald(5)        /'X3*'/
        data    ts_reg_herald(6)        /'RBPCR*'/

        EQUIVALENCE     (EMB$L_DV_REGSAV(15),MSG_BUF(0))

        INTEGER*4       MSG_BUF(0:6)

C
C TU78 Tape definitions
C
C       Define the register mnemonics for the BRIEF and CRYPTK reports
C
        character*6     mf_reg_herald(0:16)
        data    mf_reg_herald(0)                /'CS1*'/
        data    mf_reg_herald(1)                /'IS*'/
```

```
        data    mf_reg_herald(2)                /'TC*'/
        data    mf_reg_herald(3)                /'MR1*'/
        data    mf_reg_herald(4)                /'AB*'/
        data    mf_reg_herald(5)                /'BC*'/
        data    mf_reg_herald(6)                /'DT*'/
        data    mf_reg_herald(7)                /'DS*'/
        data    mf_reg_herald(8)                /'SN*'/
        data    mf_reg_herald(9)                /'MR2*'/
        data    mf_reg_herald(10)               /'MR3*'/
        data    mf_reg_herald(11)               /'NDTA*'/
        data    mf_reg_herald(12)               /'NDT 0*'/
        data    mf_reg_herald(13)               /'NDT 1*'/
        data    mf_reg_herald(14)               /'NDT 2*'/
        data    mf_reg_herald(15)               /'NDT 3*'/
        data    mf_reg_herald(16)               /'ID*'/


C
C DUP3271
C
C       Define the register mnemonics for the BRIEF and CRYPTK reports
C
        Character*5     dup_reg_herald(0:4)
        Data    dup_reg_herald(0)       /'PCSR*'/
        Data    dup_reg_herald(1)       /'RCSR*'/
        Data    dup_reg_herald(2)       /'RBUF*'/
        Data    dup_reg_herald(3)       /'TCSR*'/
        Data    dup_reg_herald(4)       /'TBUF*'/


C
C SB11
C
C       Define the register mnemonics for the BRIEF and CRYPTK reports
C
        Character*7     sb_reg_herald(0:3)
        Data    sb_reg_herald(0)        /'CSR0*'/
        Data    sb_reg_herald(1)        /'CSR1*'/
        Data    sb_reg_herald(2)        /'CSR2*'/
        Data    sb_reg_herald(3)        /'ERRSTS*'/

C
C KMS3271
C
C       Define the register mnemonics for the BRIEF and CRYPTK reports
C
        Character*5     kms3271_reg_herald(0:9)
        Data    kms3271_reg_herald(0)   /'DSTS*'/
        Data    kms3271_reg_herald(1)   /'LSTA*'/
        Data    kms3271_reg_herald(2)   /'CSR0*'/
        Data    kms3271_reg_herald(3)   /'CSR1*'/
        Data    kms3271_reg_herald(4)   /'CSR2*'/
        Data    kms3271_reg_herald(5)   /'CSR3*'/
        Data    kms3271_reg_herald(6)   /'CSR0*'/
        Data    kms3271_reg_herald(7)   /'CSR1*'/
        Data    kms3271_reg_herald(8)   /'CSR2*'/
        Data    kms3271_reg_herald(9)   /'CSR3*'/
```

```


C
C LPA11K
C
C       Define the register mnemonics for the BRIEF and CRYPTK reports
C
        CHARACTER*4     lpa_reg_HERALD(0:3)
        DATA    lpa_reg_HERALD(0)       /'CS1*'/
        DATA    lpa_reg_HERALD(1)       /'CS2*'/
        DATA    lpa_reg_HERALD(2)       /'RDA*'/
        DATA    lpa_reg_HERALD(3)       /'MR*'/


C
C DR780
C
C       Define necessary data and the register mnemonics for the
C       BRIEF and CRYPTK reports
C
        Character*4     dr780_reg_herald (0:5)
        DATA    dr780_reg_HERALD(0)     /'CR*'/
        DATA    dr780_reg_HERALD(1)     /'UTL*'/
        DATA    dr780_reg_HERALD(2)     /'BA*'/
        DATA    dr780_reg_HERALD(3)     /'SL*'/
        DATA    dr780_reg_HERALD(4)     /'CA1*'/
        DATA    dr780_reg_HERALD(5)     /'CA2*'/


C
C dr750
C
C       Define the register mnemonics for the BRIEF and CRYPTK reports
C
        Character*8     dr750_reg_herald(0:5)
        Data    dr750_reg_herald(0)     /'DCR*'/
        Data    dr750_reg_herald(1)     /'UTL*'/
        Data    dr750_reg_herald(2)     /'BA*'/
        Data    dr750_reg_herald(3)     /'WCS ADR*'/
        Data    dr750_reg_herald(4)     /'WCS DAT*'/
        Data    dr750_reg_herald(5)     /'DSL*'/

C
C Dr11w
C
C       Define necessary data and the register mnemonics for the
C       BRIEF and CRYPTK reports
C
        CHARACTER*4     dr11w_reg_herald(0:4)
        Data    dr11w_reg_herald(0)     /'CSR*'/
        Data    dr11w_reg_herald(1)     /'EIR*'/
        Data    dr11w_reg_herald(2)     /'IDR*'/
        Data    dr11w_reg_herald(3)     /'BA*'/
        Data    dr11w_reg_herald(4)     /'WC*'/


C
```

```
C Pcl11r
C
C       Define necessary data and the register mnemonics for the
C       BRIEF and CRYPTK reports
C
        character*6     pcl11r_reg_herald(0:5)
        data            pcl11r_reg_herald(0)    /'RCR*'/
        data            pcl11r_reg_herald(1)    /'RSR*'/
        data            pcl11r_reg_herald(2)    /'RDDB*'/
        data            pcl11r_reg_herald(3)    /'RDBC*'/
        data            pcl11r_reg_herald(4)    /'RDBA*'/
        data            pcl11r_reg_herald(5)    /'RDCRC*'/


C
C Pcl11t
C
C       Define necessary data and the register mnemonics for the BRIEF
C       and CRYPTK reports
C
        character*6     pcl11t_reg_herald(0:6)
        data            pcl11t_reg_herald(0)    /'TCR*'/
        data            pcl11t_reg_herald(1)    /'TSR*'/
        data            pcl11t_reg_herald(2)    /'TSDB*'/
        data            pcl11t_reg_herald(3)    /'TSBC*'/
        data            pcl11t_reg_herald(4)    /'TSBA*'/
        data            pcl11t_reg_herald(5)    /'TMMR*'/
        data            pcl11t_reg_herald(6)    /'TSCRC*'/


C
C DT07
C
C       Define necessary data and the register mnemonics for the
C       BRIEF and CRYPTK reports
C
        integer*4       ucb$l_bs_cur
        integer*4       ucb$b_bs_errmsg

        equivalence     (emb$l_dv_regsav(2),ucb$l_bs_cur)
        equivalence     (emb$l_dv_regsav(0),ucb$b_bs_errmsg)

        character*5     dt07_reg_herald(0:1)
        data    dt07_reg_herald(0)      /'CUR*'/
        data    dt07_reg_herald(1)      /'PREV*'/


C
C CI780/CI750 data definitions
C
C       Define necessary data and the register mnemonics for the
C       BRIEF and CRYPTK reports
C
        integer*4       padriver_error_type_code
        integer*4       pcnfgr

        equivalence     (emb$l_dv_regsav(0),padriver_error_type_code)
```

```
        equivalence     (emb$l_dv_regsav(1),pcnfgr)

        character*6     ci_reg_herald(0:5)
        data    ci_reg_herald(0)        /'CNFGR*'/
        data    ci_reg_herald(1)        /'PMCSR*'/
        data    ci_reg_herald(2)        /'PSR*'/
        data    ci_reg_herald(3)        /'PFAR*'/
        data    ci_reg_herald(4)        /'PESR*'/
        data    ci_reg_herald(5)        /'PPR*'/

C
C UDA50
C
C       Define data for the BRIEF and CRYPTK reports
C
        integer*2       initialization_handshake(8)

        equivalence     (emb(94),initialization_handshake)

C
C UBA ERRORS/INTERRUPTS
C
C       Define necessary data and the register mnemonics for the
C       BRIEF and CRYPTK reports
C
        INTEGER*4       UBA_CSR, UBA_CR, UBA_SR, UBA_DCR, UBA_FMER
        INTEGER*4       UBA_FUBAR, ADAPTER_TR

        EQUIVALENCE     (EMB(16),UBA_CSR)
        EQUIVALENCE     (EMB(20),UBA_CR)
        EQUIVALENCE     (EMB(24),UBA_SR)
        EQUIVALENCE     (EMB(28),UBA_DCR)
        EQUIVALENCE     (EMB(32),UBA_FMER)
        EQUIVALENCE     (EMB(36),UBA_FUBAR)
        EQUIVALENCE     (EMB(40),ADAPTER_TR)

        CHARACTER*3     TR
        DATA    TR              /'TR '/

        CHARACTER*6     UBA_REG_HERALD(0:5)
        DATA    UBA_REG_HERALD(0)       /'CSR*'/
        DATA    UBA_REG_HERALD(1)       /'CR*'/
        DATA    UBA_REG_HERALD(2)       /'SR*'/
        DATA    UBA_REG_HERALD(3)       /'DCR*'/
        DATA    UBA_REG_HERALD(4)       /'FMER*'/
        DATA    UBA_REG_HERALD(5)       /'FUBAR*'/

C
C MBA interrupts
C
C       Define necessary data and the register mnemonics for the
C       BRIEF and CRYPTK reports
C
        integer*4       rh780_control_register(6)

        equivalence     (emb(16),rh780_control_register)

```

```
        character*4     mba_reg_herald(6)
        data    mba_reg_herald(1)       /'CSR*'/
        data    mba_reg_herald(2)       /'CR*'/
        data    mba_reg_herald(3)       /'SR*'/
        data    mba_reg_herald(4)       /'VAR*'/
        data    mba_reg_herald(5)       /'BCR*'/
        data    mba_reg_herald(6)       /'DR*'/

C
C BUGCHK
C
C       Define necessary data and the register mnemonics for the
C       BRIEF and CRYPTK reports
C
        Include 'src$:modes.for /nolist'

        Integer         i, j, k

        Integer*4       bugmsg_length

        Character*73    bugmsg_text
        Parameter       bugchk_code = 104

C
C PUDRIVER
C
        Integer*2       code_word

        Equivalence     (emb(82),code_word)

        Character*(*)   Msg1, msg2, Msg3, Atten
        Parameter       (
        1 Msg1 = '''AVAILABLE'' ',
        2 Msg2 = '''DUPLICATE UNIT NUMBER'' ',
        3 Msg3 = '''ACCESS PATH'' ',
        4 Atten = 'MSCP ATTENTION MESSAGE')

        Character*(*)   Init, Comp, Fail, Bus_msg1, Bus_msg2, Bus_msg3
        Parameter       (
        1 Init = ', INIT SEQUENCE ',
        2 Comp = 'COMPLETED',
        3 Fail = 'FAILURE',
        4 Bus_msg1 = ', ''SA'' ERROR BIT SET',
        5 Bus_msg2 = ', UBA DATAPATH PURGE ERROR',
        6 Bus_msg3 = ', UCODE REV AND ''PUDRIVER'' MIS-MATCH')

C Logmscp (mscp messages without ucb's)
C
        Integer*2       logmscp_type
        Equivalence     (emb(16),logmscp_type)

C Common memory entry data definitions
```

```
C
        integer*4       memory_registers
        integer*4       buffer(0:120)
        integer*4       mem_reg_buffer(8)
        integer*4       memory_register_0
        integer*4       memory_register_a
        integer*4       controller_count

        equivalence     (mem_reg_buffer(1),memory_register_a)
        equivalence     (emb(16),buffer(0))
        equivalence     (buffer(0),controller_count)
        equivalence     (buffer(2),memory_registers)

        character*5     mem_reg_herald(3)
        data            mem_reg_herald(1)/'CSR0*'/
        data            mem_reg_herald(2)/'CSR1*'/
        data            mem_reg_herald(3)/'CSR2*'/

C  Ma780
C
C       Define necessary data and the register mnemonics for the
C       BRIEF and CRYPTK reports
C
        integer*4       port_configuration_register

        equivalence     (mem_reg_buffer(1),port_configuration_register)

        character*8     ma780_reg_herald(8)
        data            ma780_reg_herald(1)/'PRTCFNG*'/
        data            ma780_reg_herald(2)/'PRTCR*'/
        data            ma780_reg_herald(3)/'PCSR*'/
        data            ma780_reg_herald(4)/'IVDTCR*'/
        data            ma780_reg_herald(5)/'AER*'/
        data            ma780_reg_herald(6)/'CSR0*'/
        data            ma780_reg_herald(7)/'CSR1*'/
        data            ma780_reg_herald(8)/'MAT*'/

C
C  MS780C
C
C       Define necessary data and the register mnemonics for the
C       BRIEF and CRYPTK reports
C
        character*5     ms780c_reg_herald(3)
        data            ms780c_reg_herald(1)/'CSRA*'/
        data            ms780c_reg_herald(2)/'CSRB*'/
        data            ms780c_reg_herald(3)/'CSRC*'/

C
C  MS780E
C
C       Define necessary data and the register mnemonics for the
C       BRIEF and CRYPTK reports
C
        character*5     ms780e_reg_herald(4)
        data            ms780e_reg_herald(1)/'CSRA*'/
        data            ms780e_reg_herald(2)/'CSRB*'/
```

```
        data            ms780e_reg_herald(3)/'CSRC*'/
        data            ms780e_reg_herald(4)/'CSRD*'/

C
C  MS750
C
C       Define necessary data and the register mnemonics for the
C       BRIEF and CRYPTK reports
C
        integer*4       slot_index

        equivalence     (mem_reg_buffer(1),memory_register_0)

C
C  MS730
C
C       Define necessary data and the register mnemonics for the
C       BRIEF and CRYPTK reports
C
        integer*4       memory_csr0

        equivalence     (mem_reg_buffer(1),memory_csr0)

C
C MOUNT/DISMOUNT data definitions
C
C Define the necessary data and the register mnemonics for the
C BRIEF/CRYPTIC reports
C
        logical*1       str$trim

        integer*4       emb$t_vm_label_length
        integer*4       mount_operation_count
        integer*4       mount_error_count
        integer*4       volume_operation_count
        integer*4       volume_error_count

```

```
C
C Determine if a summary report should be output. If so, load
C SYS$SYSTEM:ERFPROC3.EXE for volume and memory summaries.
C
        If (summary_qual .EQ. 1) then

          All_summ = LIB$EXTZV(sum$v_all_summ,sum$s_all_summ,summary_flag)
          Memory_summ = LIB$EXTZV(sum$v_memory,sum$s_memory,summary_flag)

          If ( ((all_summ)
        1 .OR. (memory_summ))
        1 .AND. (emb$w_hd_entry .EQ. 6             ! EMB$K_SE
        1 .OR. emb$w_hd_entry .EQ. 8               ! EMB$K_HE
        1 ) ) then

            Status = IMAGE_LOADER (emb$w_hd_entry,device_class,xfer_addr)

            If (xfer_addr .EQ. 0) then
              Call LIB$SIGNAL (status)
!?? Should I append a message indicating that memory summaries cannot be done.
            Endif
          Endif
        Endif


C
C Via the entry type, dispatch to the appropriate entry handling routine.
C
        Goto (1,2,999,4,5,6,7,8,9,999) emb$w_hd_entry
        Goto (11, 12 ) (emb$w_hd_entry - 10)
        Goto (999,32,999,999,35,36,37,38,39,40 ) (emb$w_hd_entry - 30)
        Goto (41, 42 ) (emb$w_hd_entry - 40)
        Goto (999,999,999,64,65) (emb$w_hd_entry - 60)
        Goto (999,999,999,999,999,96,97,98,99,100 ) (emb$w_hd_entry - 90)
        Goto (101) (emb$w_hd_entry - 100)
        Goto (999,112) (emb$w_hd_entry - 110)

        Return


C DEVICE ERROR, TIMEOUT, and ATTENTION entries
C
C Via device class, dispatch to the appropriate device dispatching routine.
C
1       Continue        ! Device error packet
96      Continue        ! Device timeout packet
98      Continue        ! Device attention packet

          Device_type = lib$extzv (0,8,emb$b_dv_type)

          If ( EMB$B_DV_CLASS .EQ. DC$_DISK ) goto 10000
          If ( EMB$B_DV_CLASS .EQ. DC$_TAPE ) goto 20000
          If ( EMB$B_DV_CLASS .EQ. DC$_SCOM ) goto 30000
          If ( EMB$B_DV_CLASS .EQ. DC$_REALTIME ) goto 40000
          If ( EMB$B_DV_CLASS .EQ. DC$_BUS ) goto 50000

        Return
```

```
C DISKS device dispatching routine.
C
C Via device type dispatch to the appropriate device dependent routine.
C
10000   goto (

        1 10001,                ! - rk06
        1 10002,                ! - rk07

        1 10003,                ! - rp04
        1 10004,                ! - rp05

        1 10005,                ! - rp06
        1 10006,                ! - rm03

        1 10007,                ! - rp07
        1 999,                  ! - return

        1 10009,                ! - rl01
        1 10010,                ! - rl02

        1 10011,                ! - rx02
        1 10012,                ! - rx03

        1 10013,                ! - rm80
        1 10014,                ! - tu58

        1 10015,                ! - rm05
        1 999,                  ! - rx01

        1 10017,                ! - ml11
        1 10018,                ! - rb02

        1 10019                 ! - rb80
                                ! The ra80, ra81, ra60,
                                ! rc25, and rcf25 are logged
                                ! as logmessage entries and will
                                ! be output by that brief code.
        1 ) Device_type

        Return


```

```
C RK611 device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
10001   continue

10002   Brief_P2 = 12
        Brief_P3 = %loc(EMB$L_DV_REGSAV(0))
        Brief_P4(1) = 4
        Brief_P4(2) = %loc(rk_reg_herald)
        Cryp_P2 = 16
        Dhead_P2(1) = 9
        Dhead_P2(2) = %loc ('UBA RK611')

        Goto 1000


C RP 4,5 and 6 device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
10003   continue
10004   continue
10005   continue

10007   Brief_P2 = 12
        Brief_P3 = %loc(EMB$L_DV_REGSAV(7))       ! Drive_CS1
        Brief_P4(1) = 4
        Brief_P4(2) = %loc(reg_herald1)
        Cryp_P2 = 16
        Dhead_P2(1) = 7
        Dhead_P2(2) = %loc ('MASSBUS')

        Goto 1000


C RM 03,05 and 80 device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
10006   continue
10013   continue

10015   Brief_P2 = 12
        Brief_P3 = %loc(EMB$L_DV_REGSAV(7))       ! Drive_CS1
        Brief_P4(1) = 4
        Brief_P4(2) = %loc(reg_herald2)
        Cryp_P2 = 16
        Dhead_P2(1) = 7
        Dhead_P2(2) = %loc ('MASSBUS')

        Goto 1000

```

```
C RL 01 and 02 device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
10009   continue

10010   Brief_P2 = 4
        Brief_P3 = %loc(EMB$L_DV_REGSAV(0))
        Brief_P4(1) = 4
        Brief_P4(2) = %loc(rl_reg_herald)
        Cryp_P2 = 16
        Dhead_P2(1) = 8
        Dhead_P2(2) = %loc ('UBA RL11')

        Goto 1000


C RX02 device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
10011   Continue

10012   Reg_copy(1) = EMB$L_DV_REGSAV(0)          ! rx2_cs
        Reg_copy(2) = EMB$L_DV_REGSAV(1)          ! rx2_es
        Reg_copy(3) = lib$extzv(0,16,extended_registers(1))
        Reg_copy(4) = lib$extzv(16,16,extended_registers(1))
        Reg_copy(5) = lib$extzv(0,16,extended_registers(2))
        Reg_copy(6) = lib$extzv(16,16,extended_registers(2))

        Brief_P2 = 6
        Brief_P3 = %loc(Reg_copy)
        Brief_P4(1) = 5
        Brief_P4(2) = %loc(rx_reg_herald)
        Cryp_P2 = 16
        Dhead_P2(1) = 9
        Dhead_P2(2) = %loc ('UBA RX211')

        Goto 1000

```

```
C TU58 device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
10014   Brief_P2 = 4
        Brief_P3 = %loc(EMB$L_DV_REGSAV(2))      ! RCSR
        Brief_P4(1) = 5
        Brief_P4(2) = %loc(dd_reg_herald)
        Cryp_P2 = 16

C
C Determine whether there is a node name logged.
C
C Search device name string for a '$'.
C
        I = LIB$LOCC (emb$t_dv_name,'$')

C
C If a node name was logged account for and determine whether this
C tu58 entry is for the console tu58.
C
        If (I .NE. 0) then
            If (emb$t_dv_name((I+1):3) .eq. 'CSA') then J = 1
        Else
            If (emb$t_dv_name(1:3) .eq. 'CSA') then J = 1
        Endif

C
C Indicate whether this entry was for a console tu58 or uba tu58.
C
        If (J .EQ. 1) then
            Dhead_P2(1) = 12
            Dhead_P2(2) = %loc ('CONSOLE TU58')
        Else
            Dhead_P2(1) = 8
            Dhead_P2(2) = %loc ('UBA TU58')
        Endif

        Goto 1000


C ML11 device dependent routine
C
C Make a sequential copy of the device registers. Set up the brief
C and cryptic call parameters and go call the brief/cryptic routines.
C
10017   Reg_copy(0) = EMB$L_DV_REGSAV(8)         ! MLCS1
        Reg_copy(1) = EMB$L_DV_REGSAV(9)         ! MLDS
        Reg_copy(2) = EMB$L_DV_REGSAV(10)        ! MLER
        Reg_copy(3) = EMB$L_DV_REGSAV(11)        ! MLMR
        Reg_copy(4) = EMB$L_DV_REGSAV(12)        ! MLAS
        Reg_copy(5) = EMB$L_DV_REGSAV(13)        ! MLDA
        Reg_copy(6) = EMB$L_DV_REGSAV(14)        ! MLDT
        Reg_copy(7) = EMB$L_DV_REGSAV(16)        ! MLSN
        Reg_copy(8) = EMB$L_DV_REGSAV(17)        ! MLE1
        Reg_copy(9) = EMB$L_DV_REGSAV(18)        ! MLE2
```

```
        Reg_copy(10) = EMB$L_DV_REGSAV(21)      ! MLEE
        Reg_copy(11) = EMB$L_DV_REGSAV(22)      ! MLEL

        Brief_P2 = 12
        Brief_P3 = %loc(Reg_copy)
        Brief_P4(1) = 5
        Brief_P4(2) = %loc(ml_reg_herald)
        Cryp_P2 = 16
        Dhead_P2(1) = 7
        Dhead_P2(2) = %loc ('MASSBUS')

        Goto 1000

```

```
C RB 02 and 80 device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
10018   continue

10019   Brief_P2 = 7
        Brief_P3 = %loc(emb$l_dv_regsav(0))     ! Control and status register
        Brief_P4(1) = 8
        Brief_P4(2) = %loc(rb_reg_herald)
        Cryp_P2 = 32
        Dhead_P2(1) = 5
        Dhead_P2(2) = %loc ('RB730')
        Brief_32 = 1

        Goto 1000


```

```
C TAPES device dispatching routine.
C
C Via device type dispatch to the appropriate device dependent routine.
C
20000   continue

        goto (
        1 201,          ! - te16
        1 202,          ! - tu45
        1 203,          ! - tu77
        1 204,          ! - ts11
        1 205           ! - tu78
        1 ) Device_type

        Return


C TE16, TU45, TU77 device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
201     Continue
202     Continue

203     Brief_P2 = 10
        Brief_P3 = %loc(EMB$L_DV_REGSAV(7))       ! Drive_CS1
        Brief_P4(1) = 4
        Brief_P4(2) = %loc(tape_reg_herald)
        Cryp_P2 = 16
        Dhead_P2(1) = 7
        Dhead_P2(2) = %loc ('MASSBUS')

        Goto 1000

C TS11 device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
204     Reg_copy(0) = EMB$L_DV_REGSAV(0)          ! TSBA
        Reg_copy(1) = EMB$L_DV_REGSAV(1)          ! TSSR
        Reg_copy(2) = msg_buf(3)                  ! Xstat0
        Reg_copy(3) = msg_buf(4)                  ! Xstat1
        Reg_copy(4) = msg_buf(5)                  ! Xstat
        Reg_copy(5) = msg_buf(6)                  ! Xstat3
        Reg_copy(6) = msg_buf(2)                  ! Rbpcr

        Brief_P2 = 7
        Brief_P3 = %loc(Reg_copy)
        Brief_P4(1) = 6
        Brief_P4(2) = %loc(ts_reg_herald)
        Cryp_P2 = 16
        Dhead_P2(1) = 8
        Dhead_P2(2) = %loc ('UBA TS11')

        Goto 1000
```

```


C TU78 device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
205     Brief_P2 = 17
        Brief_P3 = %loc(emb$l_dv_regsav(7))     ! MF_CS1
        Brief_P4(1) = 6
        Brief_P4(2) = %loc(mf_reg_herald)
        Cryp_P2 = 16
        Dhead_P2(1) = 7
        Dhead_P2(2) = %loc ('MASSBUS')

        Goto 1000


```

```
C
C SCOM (synchronous communications) device dispatching routine.
C
C Via device type dispatch to the appropriate device dependent routine.
C
30000   continue

        goto (
        1 999,          ! - use other dispatcher
        1 999,          ! - use other dispatcher
        1 303,          ! - dup11 (3271)
        1 999,          ! - use other dispatcher
        1 999,          ! - use other dispatcher
        1 999,          ! - use other dispatcher
        1 307,          ! - sb11
        1 999,          ! - use other dispatcher
        1 999,          ! - use other dispatcher
        1 999,          ! - use other dispatcher
        1 999,          ! - use other dispatcher
        1 999,          ! - use other dispatcher
        1 999,          ! - use other dispatcher
        1 999,          ! - use other dispatcher
        1 999,          ! - use other dispatcher
        1 999,          ! - use other dispatcher
        1 999,          ! - use other dispatcher
        1 318           ! - KMS3271
        1 ) Device_type

        Return


C DUP11/3271 device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
303     If ( Report_type .EQ. 'C' ) Return
        Brief_P2 = 5
        Brief_P3 = %loc(EMB$L_DV_REGSAV(1))      ! PARCSR
        Brief_P4(1) = 5
        Brief_P4(2) = %loc(dup_reg_herald)
        Dhead_P2(1) = 9
        Dhead_P2(2) = %loc ('UBA DUP11')

        Goto 1000

C
C SB11 device dependent routine
C
C Make a sequential copy of the device registers. Set up the brief
C and cryptic call parameters and go call the brief/cryptic routines.
C
307     Reg_copy(0) = LIB$EXTZV (0,8,EMB$L_DV_REGSAV(1)) ! registers
        Reg_copy(1) = LIB$EXTZV (8,8,EMB$L_DV_REGSAV(1))
        Reg_copy(2) = LIB$EXTZV (16,8,EMB$L_DV_REGSAV(1))
        Reg_copy(3) = EMB$L_DV_REGSAV(0)                 ! err_sts

```

```
        Brief_P2 = 4
        Brief_P3 = %loc(reg_copy)
        Brief_P4(1) = 7
        Brief_P4(2) = %loc(sb_reg_herald)
        Cryp_P2 = 32
        Dhead_P2(1) = 15
        Dhead_P2(2) = %loc ('UBA DEC DATAWAY')
        Brief_32 = 1

        Goto 1000

C
C KMS3271 device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
318     Continue
        Brief_p2 = 10
        Brief_p3 = %LOC(emb$l_dv_regsav(0))
        Brief_p4(1) = 5
        Brief_p4(2) = %LOC(kms3271_reg_herald)
        Cryp_p2 = 16
        Dhead_p2(1) = 11
        Dhead_p2(2) = %LOC('UBA KMS3271')

        Goto 1000
```

```
C
C REALTIME device dispatching routine.
C
C Via device type dispatch to the appropriate device dependent routine.
C
40000   continue

        goto (
        1 401,          ! - lpa11
        1 402,          ! - dr780
        1 403,          ! - dr750
        1 404,          ! - dr11w
        1 405,          ! - pcl11 (receiver)
        1 406,          ! - pcl11 (transmitter)
        1 999,          ! - spare
        1 408           ! - dt07
        1 ) Device_type

        Return

C LPA11K device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
401     Brief_P2 = 4
        Brief_P3 = %loc(EMB$L_DV_REGSAV(0))      ! CNTRL_MAINT
        Brief_P4(1) = 4
        Brief_P4(2) = %loc(lpa_reg_herald)
        Cryp_P2 = 16
        Dhead_P2(1) = 10
        Dhead_P2(2) = %loc ('UBA LPA11K')

        Goto 1000


C DR780 device dependent routine
C
C Make a sequential copy of the device registers. Set up the brief
C and cryptic call parameters and go call the brief/cryptic routines.
C
402     Reg_copy(0) = EMB$L_DV_REGSAV(0)         ! DR_CR
        Reg_copy(1) = EMB$L_DV_REGSAV(1)         ! DR_UTL
        Reg_copy(2) = EMB$L_DV_REGSAV(5)         ! DR_BA
        Reg_copy(3) = EMB$L_DV_REGSAV(16)        ! DR_SL
        Reg_copy(4) = EMB$L_DV_REGSAV(9)         ! DR_CA(1)
        Reg_copy(5) = EMB$L_DV_REGSAV(10)        ! DR_CA(2)

        Brief_P2 = 6
        Brief_P3 = %loc(Reg_copy)
        Brief_P4(1) = 4
        Brief_P4(2) = %loc(dr780_reg_herald)
        Cryp_P2 = 32
        Dhead_P2(1) = 4
        Dhead_P2(2) = %loc ('DR32')
        Brief_32 = 1
```

```
        Goto 1000


```

```
C DR750 device dependent routine
C
C Make a sequential copy of the device registers. Set up the brief
C and cryptic call parameters and go call the brief/cryptic routines.
C
403     Reg_copy(0) = EMB$L_DV_REGSAV(0)        ! DR_DCR
        Reg_copy(1) = EMB$L_DV_REGSAV(1)        ! DR_UTL
        Reg_copy(2) = EMB$L_DV_REGSAV(5)        ! DR_CMI_ADR
        Reg_copy(3) = EMB$L_DV_REGSAV(2)        ! DR_WCS_ADR
        Reg_copy(4) = EMB$L_DV_REGSAV(3)        ! DR_WCS_DAT
        Reg_copy(5) = EMB$L_DV_REGSAV(8)        ! DR_DSL
        Brief_P2 = 6
        Brief_P3 = %loc(reg_copy)
        Brief_P4(1) = 8
        Brief_P4(2) = %loc(dr750_reg_herald)
        Cryp_P2 = 32
        Dhead_P2(1) = 4
        Dhead_P2(2) = %loc ('DR32')
        Brief_32 = 1
        Goto 1000
C DR11W device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
404     Brief_P2 = 5
        Brief_P3 = %LOC (emb$l_dv_regsav(0))
        Brief_P4(1) = 4
        Brief_P4(2) = %LOC (dr11w_reg_herald)
        Cryp_P2 = 16
        Dhead_P2(1) = 9
        Dhead_P2(2) = %LOC ('UBA DR11W')
        Goto 1000
C PCL11R device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
405     Brief_P2 = 6
        Brief_P3 = %loc(EMB$L_DV_REGSAV(0))             ! RCR
        Brief_P4(1) = 6
        Brief_P4(2) = %loc(pcl11r_reg_herald)
        Cryp_P2 = 16
        Dhead_P2(1) = 20
        Dhead_P2(2) = %loc ('UBA PCL11 (RECEIVER)')
        Goto 1000
C PCL11T device dependent routine
```

```
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
406     Brief_P2 = 7
        Brief_P3 = %loc(EMB$L_DV_REGSAV(0))           ! TCR
        Brief_P4(1) = 6
        Brief_P4(2) = %loc(pcl11t_reg_herald)
        Cryp_P2 = 16
        Dhead_P2(1) = 23
        Dhead_P2(2) = %loc ('UBA PCL11 (TRANSMITTER)')

        Goto 1000

C UNUSED
C
407     Return


C DT07 device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
408     If ( Report_type .EQ. 'B' ) then

        call dhead1 (lun,'UBA DT07')
        call linchk (lun,2)
        call ucb$$b_bs_errmsg (lun,ucb$b_bs_errmsg)

        call brief16 (lun,(2),ucb$l_bs_cur,dt07_reg_herald,emb$t_dv_name,
        1 emb$w_dv_unit)

        Else

        call cryptk (lun,16,(2),ucb$l_bs_cur,dt07_reg_herald,
        1 emb$t_dv_name,emb$w_dv_unit)
        Endif

        Return
```

```
C
C BUS device dispatching routine.
C
C Via device type dispatch to the appropriate device dependent routine.
C
50000   continue

        goto (
        1 501,  ! - ci780
        1 502,  ! - ci750
        1 503,  ! - uda50
        1 504,  ! - uda50a
        1 999,  ! - rc25
        1 999   ! - tu81p
        1 ) Device_type

        Return


C CI780/CI750 device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
501     Continue
502     Continue

        If ( Report_type .EQ. 'B' ) then

        call header (lun)
        call logger (lun,'DEVICE ATTENTION')
        call padriver_attention_error_code (lun,padriver_error_type_code)

        call brief32 (lun,(6),pcnfgr,ci_reg_herald,
     1   emb$t_dv_name,emb$w_dv_unit)

        Else

        call cryptk (lun,32,(6),pcnfgr,ci_reg_herald,
     1   emb$t_dv_name,emb$w_dv_unit)
        Endif

        Return


C UDA50 device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
503     If ( Report_type .EQ. 'B' ) then
504     Continue

        call header (lun)
        call logger (lun,'DEVICE ATTENTION')
        call linchk (lun,6)
```

```
        Goto (5001, 5002, 5003, 5004, 5005) code_word

        write(lun,5210) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
        1 ', "PUDRIVER" CODE #',code_word,'.'
        Goto 5007

5001    Message = init // comp
        Length = len (init) + len (comp)
        Goto 5006

5002    Message = init // fail
        Length = len (init) + len (fail)
        Goto 5006

5003    Message = bus_msg1
        Length = len (bus_msg1)
        Goto 5006

5004    Message = bus_msg2
        Length = len (bus_msg2)
        Goto 5006

5005    Message = bus_msg3
        Length = len (bus_msg3)

5006    write(lun,5210) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
        1 Message
5210    format(/' ','"DSA" PORT SUB-SYSTEM, UNIT _',a<emb$b_dv_namlng>,
        1 i<compress4 (lib$extzv(0,16,emb$w_dv_unit))>,':',:a,
        1 :i<compress4 (lib$extzv(0,16,code_word))>,:a<length>)

5007    write(lun,5215) 'SA','PSTEP1','HSTEP1','PSTEP2','HSTEP2','PSTEP3',
        1 'HSTEP3','PSTEP4','HSTEP4'
5215    format(/' ',t8,a,t15,a,t22,a,t29,a,t36,a,t43,a,t50,a,t57,a,t64,a)

        write(lun,5220) uda_sa,(initialization_handshake(i),i = 1,8)
5220    format(/' ',t8,z4.4,8('   ',z4.4))

        endif

        Return
```

```
C MACHINE CHECK entries
C
C Output the cryptic report for the above entries.
C
2       If ( Report_type .EQ. 'C' ) Return      ! Mach. check packet

        Call frctof (lun)
        Call header (lun)
        Call logger (lun,'MACHINE CHECK')

        Return

C SBI fault and ASYNC WRITE ERROR entries
C
C Output the cryptic report for the above entries.
C
4       Continue          ! SBI fault packet
7       Continue          ! SBI async write packet

        If ( Report_type .EQ. 'C' ) Return

        Call header (lun)

        Cpu_type = lib$extzv(24,8,emb$l_hd_sid)

        If (
        1 cpu_type .eq. 255
        1 .or.
        1 cpu_type .eq. 1
        1 .OR.
        1 cpu_type .EQ. 7
        1 ) then

        If (emb$w_hd_entry .eq. 7 ) then
          Call logger (lun,'ASYNCHRONOUS WRITE')
        Else
          Call logger (lun,'SBI FAULT')
        ENDIF

        else if ( cpu_type .eq. 2 ) then

        if (emb$w_hd_entry .eq. 7 ) then

        call logger (lun,'WRITE BUS ERROR')
        endif
        endif

        RETURN



C
C UBA INTERRUPT entries
C
C Output the brief and cryptic reports for the above entry type.
C
```

```
9         If ( Report_type .EQ. 'B' ) then
            Call header (lun)
            Call logger (lun,'''DW'' INTERRUPT')
            Call brief32 (lun,(6),uba_csr,uba_reg_herald,tr,adapter_tr)
          Else
            Call cryptk (lun,32,(6),uba_csr,uba_reg_herald,tr,adapter_tr)
          Endif
          Return


C
C UBA ERROR entries
C
C Output the brief and cryptic reports for the above entry type.
C

11        If ( Report_type .EQ. 'B' ) then
            Call header (lun)
            Call logger (lun,'''DW'' ERROR')
            Call brief32 (lun,(1),uba_csr,'UBA CSR','UBA',0)
          Else
            Call cryptk (lun,32,(1),uba_csr,'UBA CSR','UBA',0)
          Endif
          Return


C
C MBA INTERRUPT entries
C
C Output the brief and cryptic reports for the above entry type.
C

12        If ( Report_type .EQ. 'B' ) then
            Call header (lun)
            Call logger (lun,'''RH'' INTERRUPT')
            Call brief32 (lun,(6),rh780_control_register,reg_herald,TR,
          1 adapter_tr)
          Else
            Call cryptk (lun,32,(6),rh780_control_register,reg_herald,TR,
          1 adapter_tr)
          Endif
          Return

```

```
C SYSTEM START-UP and SYSTEM POWER FAIL RECOVERY entries
C
C Output the brief report for the above entry types.
C
32      Continue            ! system start-up
36      Continue            ! system power fail recovery

        Call header (lun)

        If (emb$w_hd_entry .eq. 32 ) then
          Call logger (lun,'SYSTEM START-UP')
        Else if (emb$w_hd_entry .eq. 36 ) then
          Call logger (lun,'POWER-FAIL RESTART')
        Endif
        Return


C SYSTEM POWER FAIL entries
C
C Output the brief report for the above entry type.
C
34      Call header (lun)                   ! system power fail
        Call logger (lun,'SYSTEM POWER-FAIL')
        Return

C NEW-FILE CREATION and TIME STAMP entries
C
C Output the brief report for the above entry types.
C
35      Continue            ! errlog.sys created
38      Continue            ! time stamp

        Call header (lun)

        If (emb$w_hd_entry .eq. 35) then
          Call logger (lun,'ERRLOG.SYS CREATED')
        Else if (emb$w_hd_entry .eq. 38) then
          Call logger (lun,'TIME STAMP')
        Endif

        Return

C BUGCHECK entries
C
C Output the brief report for the above entry types.
C
37      Continue            ! fatal bugcheck
40      Continue            ! non-fatal bugcheck
112     Continue            ! user bugcheck

910     format(/' ',<bugmsg_length>a1)
912     format(/' ',t8,'bugcheck code',t28,2z2.2)

        Call header (lun)
        I = 0

        If (emb$w_hd_entry .eq. '25'x) then
```

```
                 Call logger (lun,'FATAL BUGCHECK')
                 I = 140
                Else if (emb$w_hd_entry .eq. '28'x) then
                 Call logger (lun,'NON-FATAL BUGCHECK')
                Else if (emb$w_hd_entry .eq. '70'x) then
                 call logger (lun,'USER BUGCHECK')
                Endif


                Call cstring (emb(i+bugchk_code),%REF(bugmsg_text),bugmsg_length)

                If (bugmsg_length .ne. 0) then
                 Call linchk (lun,2)
                 Write(lun,910) (bugmsg_text(k:k),k = 1,bugmsg_length)
                Else
                 Call linchk (lun,2)
                 Write(lun,912) (emb(i+bugchk_code+k),k = 1,0,-1)
                Endif

                Return


        C MESSAGE ($SNDERR, OPERATOR, and NETWORK) entries
        C
        C Output the brief report for the above entry types.
        C
        39      Continue            ! $snderr message
        41      Continue            ! operator message
        42      Continue            ! network message

                Call header (lun)

                If (emb$w_hd_entry .eq. 39) then
                Call logger (lun,'$SNDERR MESSAGE')

                Else if (emb$w_hd_entry .eq. 41) then
                Call logger (lun,'OPERATOR MESSAGE')

                Else if (emb$w_hd_entry .eq. 42) then
                Call logger (lun,'NETWORK MESSAGE')
                Endif

                Call linchk (lun,3)

                Write(lun,2605)
        2605    Format(/' ','MESSAGE TEXT',/)

                If (emb$w_ss_msgsz .ne. 0) then

                Do 2615,i = 0,(emb$w_ss_msgsz/56)
                J = I*56

                Call linchk (lun,1)
                Write(lun,2610) (emb$t_ss_msgtxt(max(1,j):
                1 (j + min(55,emb$w_ss_msgsz - j))))
        2610    Format(' ',T8,A)
```

```
2615    Continue

        Else

        Call linchk (lun,1)
        Write(lun,2620)
2620    Format(' ',T8,'[NONE]')
        Endif

        Return
```

```
C MOUNT/DISMOUNT entries
C
C Output the brief and cryptic reports for the above entry type.
C
64      Continue
65      Continue

C
C Neither mount nor dismount entries are supported for a cryptic
C report so return to the calling routine.
C
        If (report_type .EQ. 'C') return

C MOUNT ENTRY
C
C Ensure the volume label is not more then 12 characters in length.
C
        If (.NOT. STR$TRIM (emb$t_vm_label,emb$t_vm_label,
        1 emb$t_vm_label_length)) then

        Emb$t_vm_label_length = 12
        Endif

C
C Call the 'mount volume' routine to record this entry in the volume
C label queue.
C
        Call HEADER (lun)

        If (volume_output) then
        If (emb$w_hd_entry .EQ. 64) then

        Call LABEL (1,emb$l_hd_sid,emb$b_vm_namlng,emb$t_vm_name,
        1 emb$w_vm_unit,emb$t_vm_label,emb$l_vm_oprcnt,emb$l_vm_errcnt)

C
C Construct and output the entry header
C
        Call LOGGER (lun,'MOUNT VOLUME')
        Call LINCHK (lun,4)

C
C Output the entry information.
C
        Write(lun,10) emb$t_vm_name,emb$w_vm_unit,
        1 emb$t_vm_label(1:emb$t_vm_label_length)
10      format(7' ',t8,'UNIT _',a<emb$b_vm_namlng>,
        1 i<compress4 (lib$extzv(0,16,emb$w_vm_unit))>,':, VOLUME LABEL "',a,
        1 '"',/)

        Write(lun,15) emb$l_vm_oprcnt,emb$l_vm_errcnt
15      format(' ',t8,i<compress4 (emb$l_vm_oprcnt)>,
        1 '. QIO OPERATIONS THIS UNIT, ',i<compress4 (emb$l_vm_errcnt)>,
        1 '. ERRORS THIS UNIT')


C DISMOUNT ENTRY
```

```
C Determine if this is a dismount entry and a brief report is selected.
C Initialize some counts and call the 'dismount volume' routine to
C search the volume label queue for a matching mount entry. (Mount operation
C and mount error count will be returned if entry found).
C
        Else if (emb$w_hd_entry .EQ. 65) then

        Call LABEL (2,emb$l_hd_sid,emb$b_vm_namlng,emb$t_vm_name,
        1 emb$w_vm_unit,emb$t_vm_label,emb$l_vm_oprcnt,emb$l_vm_errcnt)


        Mount_operation_count = -1
        Mount_error_count = -1

        Call DISMOUNT_VOLUME (2,emb$l_hd_sid,emb$b_vm_namlng,emb$t_vm_name,
        1 emb$w_vm_unit,emb$t_vm_label,mount_operation_count,
        1 mount_error_count)

C
C Construct and output the entry header.
C
        Call LOGGER (lun,'DISMOUNT VOLUME')
        Call LINCHK (lun,4)

C
C Output the entry information.
C
        Write(lun,10) emb$t_vm_name,emb$w_vm_unit,
        1 emb$t_vm_label(1:emb$t_vm_label_length)

        Write(lun,15) emb$l_vm_oprcnt,emb$l_vm_errcnt

C
C Determine if a matching mount entry was found by checking the mount
C operation and mount error counts. If an entry was found then update
C the volume counts.
C
        If (
        1 mount_operation_count .ne. -1
        1 .and.
        1 mount_error_count .ne. -1
        1 ) then

        Volume_operation_count = emb$l_vm_oprcnt - mount_operation_count
        Volume_error_count = emb$l_vm_errcnt - mount_error_count

C
C Via the volume operation and mount counts determine whether to output
C them for the report.
C
        If (volume_operation_count) 60,50,50
50      If (volume_error_count) 60,55,55

55      Call LINCHK (lun,1)

        Write(lun,56) volume_operation_count,volume_error_count
56      format(' ',t8,i<compress4 (volume_operation_count)>,
```

```
         1 '. QIO OPERATIONS THIS VOLUME, ',i<compress4 (volume_error_count)>,
         1 '. ERRORS THIS VOLUME')
         Endif
         Endif

         Else               ! volume output = false

         if (emb$w_hd_entry .EQ. 64) then

         Call MOUNT_VOLUME (1,emb$l_hd_sid,emb$b_vm_namlng,emb$t_vm_name,
         1 emb$w_vm_unit,emb$t_vm_label,emb$l_vm_oprcnt,emb$l_vm_errcnt)

         Else if (emb$w_hd_entry .EQ. 65) then

         Call DISMOUNT_VOLUME (2,emb$l_hd_sid,emb$b_vm_namlng,emb$t_vm_name,
         1 emb$w_vm_unit,emb$t_vm_label,mount_operation_count,
         1 mount_error_count)
         Endif
         Endif

 60      Return
```

```
C UNDEFINED INTERRUPT entries
C
C Output the brief and cryptic reports for the above entry type.
C
97      Continue            ! undefined interrupt

        If (report_type .EQ. 'C') return

        Call B_UNDEFINED (lun)
        Return

C LOGSTATUS entry
C
C Output the brief and cryptic reports for the above entry type.
C
99      Continue            ! Log status packet

        If ( Report_type .EQ. 'C' ) Return
        Call header (lun)
        Call logger (lun, 'ERL$LOGSTATUS ENTRY')
        Call dhead2 (lun, 'I/O', emb$b_sp_namlng, emb$t_sp_name,
        1 emb$w_sp_unit )
C       Call rollup
        Return


C LOGMESSAGE entry
C
C Output the brief and cryptic reports for the above entry type.
C
100     If ( Report_type .EQ. 'C' ) Return       ! Log message packet
c++
c       Functional description
c
c       This section provides dispatching for entries logged by erl$logmessage.
c       Dispatching is done by emb$w_lm_msgtyp.
c
c--

        Goto (21,22,23,24,25,26,27,28) emb$w_lm_msgtyp

        Tof = 1
        Write1 = 1
        Goto 1010

21      Exit1 = 1                          ! Disk messages
        Goto 1010

22      Exit1 = 1                          ! Tape messages
        Goto 1010


23      Call brief_padriver_logmessage (lun)     ! CI port
        Return

24      Call brief_pudriver_mscp_dispatcher (lun,reccnt) ! UDA port
        Return
```

```
25      tof = 1                             ! Available attention msg
        Linchk_P2 = 2
        Write1 = 0
        Message = msg1 // atten
        Length = len(msg1) + len(atten)
        Goto 1010

26      Tof = 1                             ! Duplicate unit msg
        Linchk_P2 = 2
        Write1 = 0
        Message = msg2 // atten
        Length = len(msg2) + len(atten)
        Goto 1010

27      Exit1 = 1                           ! Invalid command log msg
        Goto 1010

28      Tof = 1                             ! Access path attention msg
        Linchk_P2 = 2
        Write1 = 0
        Message = msg3 // atten
        Length = len(msg3) + len(atten)


1010    If ( tof .eq. 1 ) then
          Call frctof (lun)
          Tof = 0
        Endif

        Call header (lun)
        Call logger (lun,'ERL$LOGMESSAGE ENTRY')
        Call dhead2 (lun,'I/O',emb$b_lm_namlng,emb$t_lm_name,emb$w_lm_unit)
        If ( exit1 .eq. 1 ) then
          Exit1 = 0
          Return
        Endif

        If ( linchk_P2 .ne. 0 ) then
          Call linchk ( lun, linchk_p2 )
          Linchk_P2 = 0
        Endif


        If ( Write1 .EQ. 1 ) then
          Write(lun,2590) 'EMB$W_LM_MSGTYP',(emb$w_lm_msgtyp,i = 1,2)
2590      format(/' ',t8,a,t28,z4.4,/,
        1 t40,a,i<compress4 (lib$extzv(0,16,emb$w_lm_msgtyp))>)
          Call dumpreg (lun,((record_length + 3)/4) - 9,emb(38))

        Else

          Write(lun,2510) Message
2510      Format(/' ',a<length>)
        Endif

        Return
```

```
C LOGMSCP (mscp message without ucb)
C
C Output the brief report for the above entry type.
C
101     If (report_type .EQ. 'C') return

        Call HEADER (lun)
        Call LOGGER (lun,'ERL$LOGMSCP ENTRY')

        if (logmscp_type .eq. 9) then

        Call LINCHK (lun,2)
        write(lun,30) 'UNEXPECTED END MESSAGE ''MSCP$W_STATUS'''
30      format(/' ',t8,a)

        else if (logmscp_type .eq. 10) then

        Call LINCHK (lun,2)
        write(lun,30) 'UNEXPECTED ATTENTION CONTROL MESSAGE'

        else if (logmscp_type .eq. 11) then

        Call LINCHK (lun,2)
        write(lun,30) 'DATAGRAM FOR NON-EXISTING ''UCB'''
        endif

        return



C MEMORY entries
C
C Output the brief and cryptic reports for the above entry type.
C Output the entry header.
C
5       Continue
6       Continue
8       Continue

C
C Determine if a summary report should be output. If so,
C execute the necessary code to produce a memory summary report.
C

        If ((memory_summ) .OR. (all_summ)) then
          Call EXEC_IMAGE(xfer_addr,lun,'R')
        Endif

        If (report_type .NE. 'C') then
        Call HEADER (lun)
C SBI Alert
        If (emb$w_hd_entry .EQ. 5) then
        Call LOGGER (lun,'SBI ALERT')
```

```
C Hard ECC
        Else if (emb$w_hd_entry .EQ. 8) then
        Call LOGGER (lun,'FATAL MEMORY ERROR')

C Soft ECC
        Else
        Call LOGGER (lun,'MEMORY ERROR')
        Endif

        Endif

C Determine the cpu type.
C
        J = 0
        Cpu_type = LIB$EXTZV(24,8,emb$l_hd_sid)

        If ( cpu_type .EQ. 255
        1 .OR.
        2 cpu_type .EQ. 1 ) then

C 11/780 cpu
C  Interpret the information for all controllers in this entry.
C  Determine the memory type.
C
        Do 635, I=1, controller_count
        Memory_type = LIB$EXTZV(5,3,buffer(2+J))

C MS780C device dependent routine
C
610     If (memory_type .EQ. 0) then

C Copy the memory registers into mem_reg_buffer
C
        Call MOVC3 (%val(12),buffer(2+j),mem_reg_buffer)

C Determine if it should be a brief report. Pass the memory registers,
C register herald and the adapter tr to the brief routine for output.
C
        If (report_type .EQ. 'B') then
        Call BRIEF32 (lun,(3),memory_register_a,ms780c_reg_herald,'TR ',
        1 buffer(1+j))

        Else

C Determine if it should be a  register dump report. Pass the memory
C registers, register herald and the adapter tr to the brief routine
C for output.
C
        Call CRYPTK (lun,32,(3),memory_register_a,ms780c_reg_herald,'TR ',
        1 buffer(1+j))
        Endif

        J=J+4
        Endif


C  MA780 device dependent routine
```

```
C
620     If (memory_type .EQ. 2) then

C Copy the memory registers into mem_reg_buffer
C
        Call MOVC3 (%val(32),buffer(2+j),mem_reg_buffer)

C Determine if it should be a brief report. Pass the memory registers,
C register herald and the adapter tr to the brief routine for output.
C
        If (report_type .EQ. 'B') then
        Call BRIEF32 (lun,(8),port_configuration_register,ma780_reg_herald,
        1 'TR ',buffer(1+j))

        Else

C Determine if it should be a  register dump report. Pass the memory
C registers, register herald and the adapter tr to the brief routine
C for output.
C
        Call CRYPTK (lun,32,(8),port_configuration_register,ma780_reg_herald,
        1 'TR ',buffer(1+j))
        Endif

        J=J+9
        Endif


C MS780E device dependent routine
C
630     If (memory_type .EQ. 3) then

C Copy the memory registers into mem_reg_buffer
C
        Call MOVC3 (%val(16),buffer(2+j),mem_reg_buffer)

C Determine if it should be a brief report. Pass the memory registers,
C register herald and the adapter tr to the brief routine for output.
C
        If (report_type .EQ. 'B') then
        Call BRIEF32 (lun,(4),memory_register_a,ms780e_reg_herald,'TR ',
        1 buffer(1+j))

        Else

C Determine if it should be a  register dump report. Pass the memory
C registers, register herald and the adapter tr to the brief routine
C for output.
C
        Call CRYPTK (lun,32,(4),memory_register_a,ms780e_reg_herald,'TR ',
        1 buffer(1+j))
        Endif

        J=J+5
        Endif

635     CONTINUE
```

```
        Return

        Else if (cpu_type .EQ. 2 ) then

C 11/750 cpu
C   MS750 device dependent routine
C
640     Call MOVC3 (%val(12),memory_registers,mem_reg_buffer)

        If (report_type .EQ. 'B') then

        Call BRIEF32 (lun,(3),memory_register_0,mem_reg_herald,'SI ',
        1 slot_index)

        Else

        Call CRYPTK (lun,32,(3),memory_register_0,mem_reg_herald,'SI ',
        1 slot_index)

        Endif
        Return


        Else if ( cpu_type .EQ. 3 ) then

C 11/730 cpu
C   MS730 device dependent routine
C
650     Call MOVC3 (%val(12),memory_register_0,mem_reg_buffer)
        If (report_type .EQ. 'B') then

        Call BRIEF32 (lun,(3),memory_csr0,mem_reg_herald,'MEM',0)

        Else

        Call CRYPTK (lun,32,(3),memory_csr0,mem_reg_herald,'MEM',0)
        Endif

        Endif

        Return
```

```
C
C Determine the report type and call the brief and/or cryptic routines.
C Parameters for call to brief are:
C (lun,number of registers,registers,register heralds,name,unit)
C
1000    If ( Report_type .EQ. 'B' ) then
            Call DHEAD1 (lun, Dhead_P2 )
            If ( Brief_32 .EQ. 1 ) Then
                Call BRIEF32 (lun, Brief_P2, %val(Brief_P3), Brief_P4,
     1                        emb$t_dv_name, emb$w_dv_unit)
                Brief_32 = 0
            Else
                Call BRIEF16 (lun, Brief_P2, %val(Brief_P3), Brief_P4,
     1                        emb$t_dv_name, emb$w_dv_unit)
            Endif
        Else
            Call CRYPTK (lun, Cryp_P2, Brief_P2, %val(Brief_P3), Brief_P4,
     1                   emb$t_dv_name, emb$w_dv_unit)
        Endif
        Return

999     Return

        End
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	6084	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	1571	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	4020	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
4	SYECOM	44	PIC OVR REL GBL SHR NOEXE RD WRT LONG
5	MODE	55	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	12286	

ENTRY POINTS

Address	Type	Name
0-00000000		BRIEF_C_DISPATCHER

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000028	I*4	ADAPTER_TR	2-000005B8	I*4	ALL_SUMM
2-000004E1	L*1	BRIEF_32	2-000004DF	L*1	BRIEF_P2
2-0000059C	I*4	BRIEF_P3	2-000005C8	I*4	BUGMSG_LENGTH
2-00000550	CHAR	BUGMSG_TEXT	2-0000009F	I*4	CMD$A_POINTER
2-0000009E	L*1	CMD$B_CLASS	2-0000009D	L*1	CMD$B_TYPE
2-0000009B	I*2	CMD$W_SIZE	3-00000052	I*2	CODE_WORD
3-00000010	I*4	CONTROLLER_COUNT	2-000005A4	I*4	CPU_TYPE
4-00000012	L*1	CP_11750	4-00000011	L*1	CP_11780
4-00000013	L*1	CP_117ZZ	4-00000014	L*4	CRYPTK_FLAG
2-000004E0	L*1	CRYP_P2	2-00000024	I*4	DEV$A_BLINK
2-00000020	I*4	DEV$A_FLINK	2-0000003D	L*1	DEV$B_CLASS
2-00000028	L*1	DEV$B_DEV_NAME_LENGTH	2-00000029	CHAR	DEV$T_DEV_NAME
2-00000029	I*2	DEV$T_DEV_NAME_LEN	2-0000002B	CHAR	DEV$T_DEV_NAME_TXT
2-0000003B	I*2	DEV$W_UNIT	2-000005E4	R*4	DEVICE_CLASS
2-000005E8	R*4	DEVICE_TYPE	2-000004E5	L*1	DEVICE_TYPEDEVICE_CLASS
4-0000000D	I*4	DEV_CHAR	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	3-00000010	L*1	EMB$B_LM_CLASS
3-00000014	L*1	EMB$B_LM_NAMLNG	3-00000011	L*1	EMB$B_LM_TYPE
3-00000010	L*1	EMB$B_SP_CLASS	3-00000040	L*1	EMB$B_SP_NAMLNG
3-00000011	L*1	EMB$B_SP_TYPE	3-0000001E	L*1	EMB$B_VM_NAMLNG
3-00000036	I*4	EMB$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	3-00000014	I*4	EMB$L_SP_BCNT
3-00000038	I*4	EMB$L_SP_CHAR	3-0000003C	I*4	EMB$L_SP_CMDREF
3-00000020	I*4	EMB$L_SP_IOSB1	3-00000024	I*4	EMB$L_SP_IOSB2
3-00000018	I*4	EMB$L_SP_MEDIA	3-0000002C	I*4	EMB$L_SP_OPCNT
3-00000034	I*4	EMB$L_SP_OWNUIC	3-0000001C	I*4	EMB$L_SP_RQPID
3-00000014	I*4	EMB$L_VM_ERRCNT	3-00000018	I*4	EMB$L_VM_OPRCNT

```
  3-00000010  I*4   EMB$L_VM_OWNUIC                3-0000003F  CHAR EMB$T_DV_NAME
  3-00000015  CHAR  EMB$T_LM_NAME                  3-00000041  CHAR EMB$T_SP_NAME
  3-00000012  CHAR  EMB$T_SS_MSGTXT                3-00000032  CHAR EMB$T_VM_LABEL
  2-000005D0  I*4   EMB$T_VM_LABEL_LENGTH          3-0000001F  CHAR EMB$T_VM_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
  3-00000024  I*2   EMB$W_LM_MSGTYP                3-00000012  I*2  EMB$W_LM_UNIT
  3-00000012  I*2   EMB$W_SP_BOFF                  3-00000030  I*2  EMB$W_SP_ERRCNT
  3-00000028  I*2   EMB$W_SP_FUNC                  3-00000032  I*2  EMB$W_SP_STS
  3-0000002A  I*2   EMB$W_SP_UNIT                  3-00000010  I*2  EMB$W_SS_MSGSZ
  3-00000030  I*2   EMB$W_VM_NUMSET                3-0000001C  I*2  EMB$W_VM_UNIT
  3-0000002E  I*2   EMB$W_VM_VOLNUM                4-0000001E  L*1  END_VALUE
  4-0000001D  L*1   EOF_FLAG                       2-0000004A  L*1  ERL$B_RPT_TYPE
  2-00000063  I*4   ERL$L_END_ENTRY                2-00000067  I*4  ERL$L_MBX_CHANNEL
  2-0000006B  I*4   ERL$L_MBX_UNIT                 2-0000004B  I*4  ERL$L_SID_SELECTION
  2-0000005F  I*4   ERL$L_START_ENTRY              2-0000006F  CHAR ERL$T_GETCHN_BUFFER
  2-000004E3  L*1   EXIT1                          4-00000004  L*4  FORMS
  2-000005BC  I*4   I                              2-00000536  CHAR IMAGE_NAME
  2-000005C0  I*4   J                              2-000005C4  I*4  K
  2-000005A8  I*4   LENGTH                         2-000005A0  I*4  LINCHK_P2
  4-0000000C  L*1   LINES                          3-00000010  I*2  LOGMSCP_TYPE
  4-00000027  I*4   LSTLUN                        AP-00000004@ L*1  LUN
  4-0000001F  I*4   MAILBOX_CHANNEL                2-00000000  I*4  MEMORY_CSR0
  3-00000018  I*4   MEMORY_REGISTERS               2-00000000  I*4  MEMORY_REGISTER_0
  2-00000000  I*4   MEMORY_REGISTER_A              2-000005B4  I*4  MEMORY_SUMM
  2-000005F8  I*4   MEMORY_TYPE                    2-000004E6  CHAR MESSAGE
  2-000005D8  I*4   MOUNT_ERROR_COUNT              2-000005D4  I*4  MOUNT_OPERATION_COUNT
  4-0000002B  CHAR  OPTIONS                        3-00000052  I*4  PADRIVER_ERROR_TYPE_CODE
  3-00000056  I*4   PCNFGR                         2-00000000  I*4  PORT_CONFIGURATION_REGISTER
  4-00000008  L*4   PRINTER                        4-00000000  I*4  RECCNT
 AP-0000000C@ I*4   RECORD_LENGTH                  4-00000023  I*4  RECORD_SIZE
  2-000005F4  R*4   REG_HERALD                    AP-00000008@ CHAR REPORT_TYPE
  2-000005CC  I*4   SLOT_INDEX                     2-000005B0  I*4  STATUS
 AP-00000014@ I*4   SUMMARY_FLAG                  AP-00000010@ L*1  SUMMARY_QUAL
  2-000005EC  R*4   THENJ                          2-000004E2  L*1  TOF
  2-0000054D  CHAR  TR                             3-00000014  I*4  UBA_CR
  3-00000010  I*4   UBA_CSR                        3-0000001C  I*4  UBA_DCR
  3-00000020  I*4   UBA_FMER                       3-00000024  I*4  UBA_FUBAR
  3-00000018  I*4   UBA_SR                         3-00000052  I*4  UCB$B_BS_ERRMSG
  3-0000005A  I*4   UCB$L_BS_CUR                   2-000005F0  R*4  UDA_SA
  4-00000019  L*1   VALID_CLASS                    4-0000001A  L*1  VALID_CPU
  4-0000001B  L*1   VALID_ENTRY                    4-0000001C  L*1  VALID_TYPE
  2-000005E0  I*4   VOLUME_ERROR_COUNT             2-000005DC  I*4  VOLUME_OPERATION_COUNT
  4-00000018  L*1   VOLUME_OUTPUT                  2-000004E4  L*1  WRITE1
  2-000005AC  I*4   XFER_ADDR
```

ARRAYS

```
    Address   Type  Name                                        Bytes  Dimensions

  2-000000A4  I*4   BRIEF_P4                                        8  (2)
  3-00000010  I*4   BUFFER                                        484  (0:120)
  2-00000049  L*1   BUG$R_BUGCHK_FLAGS                              1  (1)
  2-00000049  L*1   BUGCHK_DEF                                      1  (1)
```

```
 2-0000040D  CHAR CI_REG_HERALD                       36  (0:5)
 2-000001A2  CHAR DD_REG_HERALD                       20  (0:3)
 2-0000003E  L*1  DEV$R_FLAGS_OVERLAY                  1  (1)
 2-0000003E  L*1  DEV$R_FLAG_BITS                      1  (1)
 2-00000020  L*1  DEV_QUEUE                           31  (31)
 2-000000AC  I*4  DHEAD_P2                             8  (2)
 2-000003A1  CHAR DR11W_REG_HERALD                    20  (0:4)
 2-00000371  CHAR DR750_REG_HERALD                    48  (0:5)
 2-00000359  CHAR DR780_REG_HERALD                    24  (0:5)
 2-00000403  CHAR DT07_REG_HERALD                     10  (0:1)
 2-000002E2  CHAR DUP_REG_HERALD                      25  (0:4)
 3-00000000  L*1  EMB                                512  (0:511)
 3-00000026  L*1  EMB$B_LM_MSGTXT                    460  (460)
 3-00000052  I*4  EMB$L_DV_REGSAV                    420  (0:104)
 3-00000006  I*4  EMB$Q_HD_TIME                        8  (2)
 2-0000004F  I*4  ERL$Q_END_DATE                       8  (2)
 2-00000057  I*4  ERL$Q_START_DATE                     8  (2)
 2-0000009B  L*1  ERL$R_CMD_LINE                       8  (8)
 2-00000043  L*1  EXC$R_EXCLUDE_FLAGS                  3  (3)
 2-00000043  L*1  EXCDEF                               3  (3)
 3-0000006E  I*4  EXTENDED_REGISTERS                   8  (2)
 3-0000006E  I*2  EXTENDED_STATUS                      8  (4)
 2-00000040  L*1  INC$R_INCLUDE_FLAGS                  3  (3)
 2-00000040  L*1  INCDEF                               3  (3)
 3-0000005E  I*2  INITIALIZATION_HANDSHAKE            16  (8)
 2-00000317  CHAR KMS3271_REG_HERALD                  50  (0:9)
 2-00000349  CHAR LPA_REG_HERALD                      16  (0:3)
 2-0000047C  CHAR MA780_REG_HERALD                    64  (8)
 2-00000455  CHAR MBA_REG_HERALD                      24  (6)
 2-00000000  I*4  MEM_REG_BUFFER                      32  (8)
 2-0000046D  CHAR MEM_REG_HERALD                      15  (3)
 2-0000027C  CHAR MF_REG_HERALD                      102  (0:16)
 2-000001B6  CHAR ML_REG_HERALD                       60  (0:11)
 5-00000000  CHAR MODES                               55  (0:4)
 2-000004BC  CHAR MS780C_REG_HERALD                   15  (3)
 2-000004CB  CHAR MS780E_REG_HERALD                   20  (4)
 3-0000008E  I*4  MSG_BUF                             28  (0:6)
 2-00000046  L*1  OPT$R_OPT_FLAGS                      3  (3)
 2-00000046  L*1  OPTDEF                               3  (3)
 2-000003B5  CHAR PCL11R_REG_HERALD                   36  (0:5)
 2-000003D9  CHAR PCL11T_REG_HERALD                   42  (0:6)
 2-0000004A  L*1  PRS_DATA                            81  (81)
 2-0000009B  L*1  PRS_TABLE                            8  (8)
 2-000001F2  CHAR RB_REG_HERALD                       56  (0:6)
 2-000000B4  I*4  REG_COPY                            48  (0:11)
 2-00000114  CHAR REG_HERALD1                         48  (0:11)
 2-00000144  CHAR REG_HERALD2                         48  (0:11)
 3-00000010  I*4  RH780_CONTROL_REGISTER              24  (6)
 2-000000E4  CHAR RK_REG_HERALD                       48  (0:11)
 2-00000174  CHAR RL_REG_HERALD                       16  (0:3)
 2-00000184  CHAR RX_REG_HERALD                       30  (0:5)
 2-000002FB  CHAR SB_REG_HERALD                       28  (0:3)
 2-0000003F  L*1  SUM$R_SUMMARY_FLAGS                  1  (1)
 2-0000003F  L*1  SUMDEF                               1  (1)
 2-0000022A  CHAR TAPE_REG_HERALD                     40  (0:9)
 2-00000252  CHAR TS_REG_HERALD                       42  (0:6)
 2-00000431  CHAR UBA_REG_HERALD                      36  (0:5)
```

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
0-00000111	1	0-00000A31	2	0-00000A67	4	0-000013D3	5	0-000013D3	6	0-00000A67	7
0-000013D3	8	0-00000AEA	9	1-00000441	10'	0-00000B44	11	0-00000B9E	12	1-00000474	15'
0-00001167	21	0-0000116F	22	0-00001177	23	0-00001187	24	0-00001197	25	0-000011B9	26
0-000011DB	27	0-000011E2	28	1-00000522	30'	0-00000BF8	32	**	34	0-00000C3A	35
0-00000BF8	36	0-00000C7C	37	0-00000C3A	38	0-00000D9A	39	0-00000C7C	40	0-00000D9A	41
0-00000D9A	42	**	50	**	55	1-000004B8	56'	0-000010EB	60	0-00000ECD	64
0-00000ECD	65	0-00000111	96	0-000010EC	97	0-00000111	98	0-00001104	99	0-0000113A	100
0-000012F7	101	0-00000C7C	112	0-000003FB	201	0-000003FB	202	0-000003FB	203	0-00000429	204
0-0000048B	205	0-000004E6	303	0-00000517	307	0-0000057B	318	0-000005C2	401	0-000005F0	402
0-0000064D	403	0-000006A9	404	0-000006D7	405	0-00000705	406	**	407	0-00000733	408
0-000007B1	501	0-000007B1	502	0-0000081A	503	0-00000824	504	**	610	**	620
**	630	**	635	**	640	**	650	1-000003EF	910'	1-000003FB	912'
0-000016DB	999	0-00001676	1000	0-00001202	1010	1-00000517	2510'	1-00000500	2590'	1-00000418	2605'
1-0000042C	2610'	**	2615	1-00000433	2620'	0-000008C6	5001	0-000008DB	5002	0-000008F0	5003
0-00000904	5004	0-00000918	5005	0-0000092A	5006	0-0000097A	5007	1-00000375	5210'	1-000003B8	5215'
1-000003D8	5220'	0-00000152	10000	0-00000181	10001	0-00000181	10002	0-000001AF	10003	0-000001AF	10004
0-000001AF	10005	0-000001DD	10006	0-000001AF	10007	0-0000020B	10009	0-0000020B	10010	0-00000239	10011
0-00000239	10012	0-000001DD	10013	0-000002AF	10014	0-000001DD	10015	0-00000332	10017	0-000003B5	10018
0-000003B5	10019	0-000003E8	20000	0-000004B9	30000	0-000005A9	40000	0-0000079C	50000		

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	BRIEF16		BRIEF32		BRIEF_PADRIVER_LOGMESSAGE
	BRIEF_PUDRIVER_MSCP_DISPATCHER		B_UNDEFINED	I*4	COMPRESS4
	CRYPTR		CSTRING		DHEAD1
	DHEAD2		DISMOUNT_VOLUME		DUMPREG
	ERF_LOADERR		EXEC_IMAGE		FRCTOF
	HEADER	I*4	IMAGE_LOADER		LABEL
I*4	LIB$EXTZV	I*4	LIB$LOCC		LIB$SIGNAL
	LINCHK		LOGGER		MOUNT_VOLUME
	MOVC3		PADRIVER_ATTENTION_ERROR_CODE	L*1	STR$TRIM
	UCB$$B_BS_ERRMSG				

```
        Subroutine BRIEF_PADRIVER_LOGMESSAGE (lun)


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

        byte            lun

        equivalence     (emb$b_lm_msgtxt(1),padriver_error_type_code)

        integer*4       padriver_error_type_code
        integer*4       error_subtype
        integer*4       compress4
        integer*4       error_type


        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$w_lm_unit,'UNRECOGNIZED "SCA" PACKET'
   10       format(/' ','CI SUB-SYSTEM, _',a<emb$b_lm_namlng>,
        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
          Write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
     1   emb$w_lm_unit,'DATA CABLE(S) CHANGE OF STATE'
          Call linchk (lun,2)
          If (error_subtype .eq. 0) then
            write(lun,12) 'PATH #0. HAS GONE FROM GOOD TO BAD'
 12         format(/' ',t8,a)
          Else
            Goto (200,205,210,215,220,225,230,235,240,245,250)
     1      error_subtype
 999        Continue
          Endif
        else

        write(lun,25) emb$t_lm_name(1:emb$b_lm_namlng),
        1 emb$w_lm_unit,'''PADRIVER'' ERROR TYPE #',error_type,
        1 '., ERROR SUB-TYPE #',error_subtype,'.'
 25     format(/' ','CI SUB-SYSTEM, ',a<emb$b_lm_namlng>,
        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)

        Return
 200        write(lun,12) 'PATH #1. HAS GONE FROM GOOD TO BAD'
            goto 999

 205        write(lun,12) 'PATH #0. HAS GONE FROM BAD TO GOOD'
            goto 999

 210        write(lun,12) 'PATH #1. HAS GONE FROM BAD TO GOOD'
            goto 999

 215        write(lun,12) 'CABLES HAVE GONE FROM UNCROSSED TO CROSSED'
            goto 999

 220        write(lun,12) 'CABLES HAVE GONE FROM CROSSED TO UNCROSSED'
            goto 999
```

```
 225            write(lun,12) 'PATH #0. LOOPBACK HAS GONE FROM GOOD TO BAD'
                goto 999

 230            write(lun,12) 'PATH #1. LOOPBACK HAS GONE FROM GOOD TO BAD'
                goto 999

 235            write(lun,12)'PATH #0. LOOPBACK HAS BECOME GOOD, UNCROSSED'
                goto 999

 240            write(lun,12)'PATH #1. LOOPBACK HAS BECOME GOOD, UNCROSSED'
                goto 999

 245            write(lun,12)
          1       'PATH #0. HAS BECOME WORKING BUT CROSSED TO PATH #1.'
                goto 999

 250            write(lun,12)
          1       'PATH #1. HAS BECOME WORKING BUT CROSSED TO PATH #0.'
                goto 999
          End
```

PROGRAM SECTIONS

```
    Name                                        Bytes   Attributes

  0 $CODE                                        1625   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                       1040   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                        332   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG
  3 EMB                                           512   PIC OVR REL GBL   SHR NOEXE   RD   WRT LONG

    Total Space Allocated                        3509
```

ENTRY POINTS

```
    Address   Type  Name

  0-00000000        BRIEF_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                       2-00000000  I*4   ERROR_SUBTYPE
  2-00000004  I*4   ERROR_TYPE                         AP-00000004@ L*1   LUN
  3-00000026  I*4   PADRIVER_ERROR_TYPE_CODE
```

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)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-0000039D	10'	1-000003D0	12'	1-000003D8	25'	0-00000435	200	0-0000045B	205	0-00000481	210
0-000004A7	215	0-000004CD	220	0-000004F3	225	0-00000519	230	0-0000053F	235	0-00000565	240
0-0000058B	245	0-000005B1	250	0-000003AD	999						

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4		HEADER	I*4	LIB$EXTZV
	LINCHK		LOGGER		PADRIVER_INITIALIZATION

0001

```
        Subroutine BRIEF_PUDRIVER_MSCP_DISPATCHER (lun,reccnt)

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

        byte            lun
        byte            mslg$b_format

        Integer*4       reccnt

        equivalence     (emb(46),mslg$b_format)


        if (emb$w_hd_entry .eq. 100) then

           call FRCTOF (lun)
           call HEADER2 (lun,reccnt)
           call LOGGER (lun,'ERL$LOGMESSAGE ENTRY')
           call DHEAD2 (lun,'''DSA'' PORT',emb$b_lm_namlng,emb$t_lm_name,
        1               emb$w_lm_unit)

        else if (emb$w_hd_entry .eq. 99) then

                call FRCTOF (lun)
                call HEADER2 (lun,reccnt)
                call LOGGER (lun,'ERL$LOGSTATUS ENTRY')
                call DHEAD2 (lun,'''DSA'' PORT',emb$b_sp_namlng,emb$t_sp_name,
        1                    emb$w_sp_unit)

        Endif

        Return
        End
```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 $CODE	148	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 $PDATA	52	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 $LOCAL	132	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3 EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
Total Space Allocated	844	

ENTRY POINTS

Address	Type	Name
0-00000000		BRIEF_PUDRIVER_MSCP_DISPATCHER

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$B_LM_TYPE	3-00000010	L*1	EMB$B_SP_CLASS
3-00000040	L*1	EMB$B_SP_NAMLNG	3-00000011	L*1	EMB$B_SP_TYPE
3-00000000	I*4	EMB$L_HD_SID	3-00000014	I*4	EMB$L_SP_BCNT
3-00000038	I*4	EMB$L_SP_CHAR	3-0000003C	I*4	EMB$L_SP_CMDREF
3-00000020	I*4	EMB$L_SP_IOSB1	3-00000024	I*4	EMB$L_SP_IOSB2
3-00000018	I*4	EMB$L_SP_MEDIA	3-0000002C	I*4	EMB$L_SP_OPCNT
3-00000034	I*4	EMB$L_SP_OWNUIC	3-0000001C	I*4	EMB$L_SP_RQPID
3-00000015	CHAR	EMB$T_LM_NAME	3-00000041	CHAR	EMB$T_SP_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
3-00000012	I*2	EMB$W_SP_BOFF	3-00000030	I*2	EMB$W_SP_ERRCNT
3-00000028	I*2	EMB$W_SP_FUNC	3-00000032	I*2	EMB$W_SP_STS
3-0000002A	I*2	EMB$W_SP_UNIT	AP-00000004@	L*1	LUN
3-0000002E	L*1	MSLG$B_FORMAT	AP-00000008@	I*4	RECCNT

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)

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name
	DHEAD2		FRCTOF		HEADER2		LOGGER

0001

COMMAND QUALIFIERS

```
  FORTRAN /LIS=LIS$:B_C_DISP/OBJ=OBJ$:B_C_DISP MSRC$:B_C_DISP

  /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:           30.87 seconds
  Elapsed Time:       81.18 seconds
  Page Faults:        595
  Dynamic Memory:     455 pages
```

0146 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

ERFPROC4 MAP

EMBETDEF REQ

BUGCHK LIS

DEVERR FOR

MODES FOR

SSMESSAGE FOR

B_C_DISP LIS

ERFSHR MAP

BSDRIVER LIS

ERFPROC3 MAP

ERFDEF REQ

MSGHDR FOR

SYECOM FOR

ERFRLTIM MAP

ERFTAPE MAP

PARSERDAT SDL

DEV_CLASS FOR

DR32COM FOR

OPCODES FOR

VOLMOUNT FOR

ERFSUMM MAP

EMBLMDEF FOR

QIOCOMMON FOR

ERFPROC5 MAP

DEVERRDEF REQ

RECSELDEF REQ

BRIEF LIS

EMBSPDEF FOR

0147 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CLASSIFY
LIS

DR750
LIS

CSTRING
LIS

DHEADS
LIS

DR780
LIS

DR11W
LIS

COMPRESS
LIS

DTAILS
LIS

DECODECC
LIS

DUMPREG
LIS

DUTUDRIVR
LIS

CALCMAP
LIS

DUP3271
LIS

CRYPTK
LIS

DQDISKS
LIS

DUP11
LIS