L 12

```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
```

```
_$

Ps
--

YZ

Z$

Z$

Z$

Z$

Z$

Z$

Z$

Z$

Z$

Z$

Z$

Z$
```

```
**FILE**ID**ERRORLOG                                   G  4
```

```
EEEEEEEEEE  RRRRRRRR    RRRRRRRR      OOOOOO    RRRRRRRR    LL            OOOOOO      GGGGGGGG
EEEEEEEEEE  RRRRRRRR    RRRRRRRR      OOOOOO    RRRRRRRR    LL            OOOOOO      GGGGGGGG
EE          RR      RR  RR      RR  OO      OO  RR      RR  LL          OO      OO  GG
EE          RR      RR  RR      RR  OO      OO  RR      RR  LL          OO      OO  GG
EE          RR      RR  RR      RR  OO      OO  RR      RR  LL          OO      OO  GG
EE          RR      RR  RR      RR  OO      OO  RR      RR  LL          OO      OO  GG
EEEEEEEE    RRRRRRRR    RRRRRRRR    OO      OO  RRRRRRRR    LL          OO      OO  GG
EEEEEEEE    RRRRRRRR    RRRRRRRR    OO      OO  RRRRRRRR    LL          OO      OO  GG
EE          RR  RR      RR  RR      OO      OO  RR  RR      LL          OO      OO  GG  GGGGGG
EE          RR  RR      RR  RR      OO      OO  RR  RR      LL          OO      OO  GG  GGGGGG
EE          RR    RR    RR    RR    OO      OO  RR    RR    LL          OO      OO  GG      GG      ....
EE          RR    RR    RR    RR    OO      OO  RR    RR    LL          OO      OO  GG      GG      ....
EEEEEEEEEE  RR      RR  RR      RR    OOOOOO    RR      RR  LLLLLLLLLL    OOOOOO      GGGGGG        ....
EEEEEEEEEE  RR      RR  RR      RR    OOOOOO    RR      RR  LLLLLLLLLL    OOOOOO      GGGGGG        ....

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

ER
VO

Table of contents

```
0000            .TITLE  ERRORLOG - ERROR LOG SUPPORT ROUTINES
0000            .IDENT  'V04-000'
0000
0000     ;****************************************************************************
0000     ;*                                                                          *
0000     ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000     ;*  ALL RIGHTS RESERVED.                                                    *
0000     ;*                                                                          *
0000     ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000     ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000     ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000     ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000     ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000     ;*  TRANSFERRED.                                                            *
0000     ;*                                                                          *
0000     ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000     ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000     ;*  CORPORATION.                                                            *
0000     ;*                                                                          *
0000     ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000     ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000     ;*                                                                          *
0000     ;*                                                                          *
0000     ;****************************************************************************
0000     ;
0000     ; D. N. CUTLER 7-MAR-77
0000     ;
0000     ; ERROR LOG SUPPORT ROUTINES
0000     ;
0000     ; MODIFIED BY:
0000     ;
0000     ;       V03-012 EAD0162         Elliott A. Drayton      26-Apr-1984
0000     ;               Correct ADDB3 in routine GETFULLNAME to use R0.
0000     ;
0000     ;       V03-011 EAD0160         Elliott A. Drayton      16-Apr-1984
0000     ;               Added a test for the system block address not being there.
0000     ;
0000     ;       V03-010 EAD0137         Elliott A. Drayton      11-Apr-1984
0000     ;               Changed code to log full device names. NODE NAME + DEVICE.
0000     ;
0000     ;       V03-009 LMP0221         L. Mark Pilant,         30-Mar-1984  13:57
0000     ;               Change UCB$L_OWNUIC to ORB$L_OWNER and UCB$W_VPROT to
0000     ;               ORB$W_PROT.
0000     ;
0000     ;       V03-008 KPL0100         Peter Lieberwirth       22-Mar-1984
0000     ;               Use CONFREGL instead of CONFREG.  Anticipate SBICONF
0000     ;               containing a PFN instead of a VA if BI adapter
0000     ;               initialization didn't originally recognize the adapter.
0000     ;
0000     ;       V03-007 SSA0007         Stan Amway              2-Feb-1984
0000     ;               Fix broken branch to ERL$ALLOCEMB.
0000     ;
0000     ;       V03-006 LMP0185         L. Mark Pilant,         1-Feb-1984  9:37
0000     ;               Fix some broken branches.
0000     ;
0000     ;       V03-005 ROW0241         Ralph O. Weber          12-OCT-1983
```

```
0000    ;               Correct ERL$LOG_D(T)MSCP to allocate space for the error log
0000    ;               header in addition to the space needed for the logged message
0000    ;               etc.  Also use symbolic size of error log entry header instead
0000    ;               of a constant.
0000    ;
0000    ;       V03-004 RLRMSCP         Robert L. Rappaport     27-Jul-1983
0000    ;               Add two entrypoints, ERL$LOG_DMSCP and ERL$LOG_TMSCP,
0000    ;               to log invalid Disk MSCP and Tape MSCP messages.
0000    ;
0000    ;       V03-003 KDM0051         Kathleen D. Morse       11-Jul-1983
0000    ;               Change references to TODR to use loadable, cpu-dependent
0000    ;               routine, EXE$READ_TODR.
0000    ;       V03-002 BLS0187         Benn Schreiber          24-Sep-1982
0000    ;               Correct broken branch offset due to UCB growing
0000    ;
0000    ;-
```

```
0000    ;
0000    ; MACRO LIBRARY CALLS
0000    ;
0000
0000            $CDDBDEF                                ;DEFINE CDDB OFFSETS
0000            $CDRPDEF                                ;DEFINE CDRP OFFSETS
0000            $DDBDEF                                 ;DEFINE DDB OFFSETS
0000            $DDTDEF                                 ;DEFINE DDT OFFSETS
0000            $DEVDEF                                 ;DEFINE DEVICE CHARACTERISTIC BITS
0000            $EMBDEF <DV,SU,TS,UI,SP,LM,ET>          ;ERROR LOG MESSAGE BUFFERS OFFSETS
0000            $ERLDEF                                 ;DEFINE ERROR ALLOCATION BUFFER OFFSETS
0000            $FCBDEF                                 ;DEFINE FCB OFFSETS
0000            $IODEF                                  ;DEFINE I/O FUNCTION VALUES
0000            $IRPDEF                                 ;DEFINE IRP OFFSETS
0000            $MCHKDEF                                ;DEFINE MACHINE CHECK RECOVERY MASK BITS
0000            $MSCPDEF                                ;DEFINE MSCP OFFSETS
0000            $NDTDEF                                 ;DEFINE NEXUS DEVICE TYPE CODES
0000            $ORBDEF                                 ;DEFINE OBJECT'S RIGHTS BLOCK OFFSETS
0000            $PRDEF                                  ;DEFINE PROCESSOR REGISTER NUMBERS
0000            $SBDEF                                  ;DEFINE SYSTEM BLOCK OFFSETS
0000            $UCBDEF                                 ;DEFINE UCB OFFSETS
0000            $WCBDEF                                 ;DEFINE WCB OFFSETS
0000
0000                                                    ;***
0000    ;DEBUG=1                                        ;***IF DEFINED, ENABLE UNEXPECTED
0000                                                    ;*** INTERRUPT IDENTIFIES VECTOR #
0000
0000    ;
0000    ; LOCAL MACROS
0000    ;
0000
0000    ;
0000    ; MACRO TO DEFINE AN INTERRUPT SERVICE ROUTINE LABEL FOR UNEXPECTED  INTERRUPTS
0000    ;
0000            .MACRO  ISRDEF,VNUM
0000            .ALIGN  LONG                            ; Make all vectors long word alligned
0000    ERL$VEC'VNUM::                                  ;INTERRUPT SERVICE LABEL
0000            .IF     DF,DEBUG                        ;***IF DEBUGGING
0000            BSBW    ERL$UNEXP                       ;***CALL INTERRUPT SERVICE
0000            .BYTE   <VNUM>/2                        ;***IDENTIFY VECTOR OFFSET INTO SCB
0000            .ENDC
0000            .ENDM   ISRDEF
0000    ;
0000    ; MACRO TO DEFINE THE INTERRUPT SERVICE ROUTINE LABELS FOR AN ADAPTER
0000    ;
0000            .MACRO  ADPISR,SLOT
0000    VECTOR = SLOT * 4 + 256
0000            .REPT   4
0000            ISRDEF  \VECTOR
0000    VECTOR = VECTOR + <16 * 4>
0000            .ENDR
0000            .IF     NDF,DEBUG                       ;IF NOT DEBUGGING
0000            BSBB    ADP_HANDLER                     ;CALL INTERRUPT SERVICE
0000            .ENDC
0000            .ENDM   ADPISR
0000    ;
0000    ; LOCAL SYMBOLS
```

```
                 0000   ;
                 0000
                 0000   ;
                 0000   ; MAXIMUM NUMBER OF MESSAGES BEFORE WAKE OF FORMAT PROCESS
                 0000   ;
                 0000
       0000000A  0000   MAXMSG=10                                       ;
                 0000
                 0000   ;
                 0000   ; MAXIMUM TIME IN SECONDS BEFORE WAKE OF FORMAT PROCESS
                 0000   ;
                 0000
       0000001E  0000   MAXTIM=30                                       ;
                 0000
                 0000   ;
                 0000   ; LOCAL DATA
                 0000   ;
                 0000
             00000000           .PSECT  $$$260,QUAD,WRT
                 0000   ;
                 0000   ;       WARNING!!!  The next two bytes must be adjacent and word aligned
                 0000   ;
                 0000           .ALIGN  WORD
             00  0000   BUF1:   .BYTE   0                               ;COUNT OF BUSY MESSAGES IN BUFFER
             00  0001           .BYTE   0                               ;COUNT OF COMPLETED MESSAGES IN BUFFER
             00  0002           .BYTE   0                               ;BUFFER INDICATOR
             00  0003           .BYTE   0                               ;BUFFER CONTROL FLAGS
      0000000C'  0004           .LONG   10$                             ;ADDRESS OF NEXT AVAILABLE SPACE IN BUFFER
      00000200'  0008           .LONG   20$                             ;ADDRESS OF END OF BUFFER + 1
       00000200  000C   10$:    .BLKB   512-ERL$C_LENGTH                ;ACTUAL BUFFER AREA
                 0200   20$:                                            ;REF LABEL
                 0200   ;
                 0200   ;       WARNING!!!  The next two bytes must be adjacent and word aligned
                 0200   ;
                 0200           .ALIGN  WORD
             00  0200   BUF2:   .BYTE   0                               ;COUNT OF BUSY MESSAGES IN BUFFER
             00  0201           .BYTE   0                               ;COUNT OF COMPLETED MESSAGES IN BUFFER
             01  0202           .BYTE   1                               ;BUFFER INDICATOR
             00  0203           .BYTE   0                               ;BUFFER CONTROL FLAGS
      0000020C'  0204           .LONG   10$                             ;ADDRESS OF NEXT AVAILABLE SPACE IN BUFFER
      00000400'  0208           .LONG   20$                             ;ADDRESS OF END OF BUFFER + 1
       00000400  020C   10$:    .BLKB   512-ERL$C_LENGTH                ;ACTUAL BUFFER AREA
                 0400   20$:                                            ;REF LABEL
                 0400
                 0400   ;
                 0400   ; GLOBAL DATA
                 0400   ;
                 0400   ; ERROR LOG DATA BASE
                 0400   ;
                 0400
                 0400   ERL$AL_BUFADDR::                                ;ALLOCATION BUFFER ADDRESS ARRAY
      00000000'  0400           .LONG   BUF1                            ;ADDRESS OF BUFFER 1 DESCRIPTOR
      00000200'  0404           .LONG   BUF2                            ;ADDRESS OF BUFFER 2 DESCRIPTOR
                 0408   ERL$GB_BUFIND::                                 ;CURRENT ALLOCATION BUFFER INDICATOR
             00  0408           .BYTE   0                               ;
                 0409   ERL$GB_BUFFLAG::                                ;BUFFER STATUS FLAGS
             00  0409           .BYTE   0                               ;
```

```
          040A  ERL$GB_BUFPTR::                             ;FORMAT PROCESS BUFFER INDICATOR
      00  040A          .BYTE   0                           ;
          040B  ERL$GB_BUFTIM::                             ;FORMAT PROCESS WAKEUP TIMER
      1E  040B          .BYTE   MAXTIM                      ;
          040C  ERL$GL_ERLPID::                             ;PROCESS ID OF ERROR LOG PROCESS
00000000  040C          .LONG   0                           ;
          0410  ERL$GL_SEQUENCE::                           ;UNIVERSAL ERROR SEQUENCE NUMBER
00000000  0410          .LONG   0                           ;
```

```
                              0414         .SBTTL  UNEXPECTED INTERRUPT SERVICE
                              0414
                              0414  ;+
                              0414  ; ERL$VEC'VNUM - INTERRUPT SERVICE FOR SCB VECTOR VNUM.
                              0414  ; ERL$UNEXP - GENERAL UNEXPECTED INTERRUPT SERVICE
                              0414  ;
                              0414  ; THESE INTERRUPT SERVICE ROUTINES ARE EXECUTED FOR UNUSED SCB VECTORS.
                              0414  ;
                              0414  ; IF DEBUG IS DEFINED, EACH INTERRUPT SERVICE CALLS ERL$UNEXP WITH
                              0414  ; THE <VECTOR OFFSET>/2 INTO THE SCB AS A 1 BYTE ARGUMENT.
                              0414  ;
                              0414  ; IF DEBUG IS NOT DEFINED, ALL CPU INTERRUPT SERVICE ROUTINES COLLAPSE TO
                              0414  ; GLOBAL LABELS EQUAL TO ERL$UNEXP AND ALL ADAPTER INTERRUPT SERVICE
                              0414  ; ROUTINES CALL A ROUTINE THAT SAVES THE ADAPTER TYPE, TRIES TO DISABLE
                              0414  ; FURTHER INTERRUPTS, AND LOGS THE INTERRUPT.
                              0414  ;
                              0414  ; THERE ARE ENOUGH INTERRUPT SERVICE ROUTINES FOR THE ARCHITECTURAL PAGE
                              0414  ; OF THE SCB, I.E., 128 ROUTINES.
                              0414  ;
                              0414  ; INPUTS:
                              0414  ;
                              0414  ;        (SP) = PC AT INTERRUPT
                              0414  ;        4(SP) = PSL AT INTERRUPT
                              0414  ;
                              0414  ; OUTPUTS:
                              0414  ;
                              0414  ;        ERROR IS LOGGED, OR PROCESSOR BUGCHECKS.
                              0414  ;-
                          00000000         .PSECT  $AEXENONPAGED,LONG
                              0000  ;
                              0000  ; UNEXPECTED ADAPTER INTERRUPT HANDLER: IF DEBUG IS DISABLED, SAVE THE
                              0000  ; ADAPTER TYPE, ATTEMPT TO DISABLE FURTHER INTERRUPTS FROM THE ADAPTER,
                              0000  ; AND LOG THE INTERRUPT.  IF DEBUG IS ENABLED, BUGCHECK AS FOR CPU INTERRUPTS.
                              0000  ;
                              0000         .ALIGN  LONG
                              0000  ADP_UNEXP:
                    00000000  0000         NEXUS = 0                       ;FIRST ADAPTER = 0
                              0000         .REPT   16                      ;ISR'S FOR 16 ADAPTERS ONLY
                              0000         ADPISR  \NEXUS                  ;DEFINE ERL$INT'VNUM LABELS AND ISRS
                              0000         NEXUS = NEXUS + 1               ;NEXT ADAPTER
                              0000         .ENDR                           ;
                              003E  ADP_HANDLER:
     6E   00000002'8F     C2  003E         SUBL    #ADP_UNEXP+2,(SP)       ;COMPUTE ADAPTER OFFSET
              6E   04     C6  0045         DIVL    #4,(SP)                 ;COMPUTE ADAPTER SLOT/TR NUMBER
                   1F     BB  0048         PUSHR   #^M<R0,R1,R2,R3,R4>     ;SAVE REGISTERS
           53    14 AE    D0  004A         MOVL    5*4(SP),R3              ;RETRIEVE SLOT NUMBER
  54    00000000'FF43     D0  004E         MOVL    @MMG$GL_SBICONF[R3],R4  ;GET ADDRESS OF ADAPTER REGISTERS
                   5C     18  0056         BGEQ    100$                    ;GEQ MEANS SBICONF DOES NOT CONTAIN
                              0058                                         ; A SYSTEM VA, MUST BE PFN OR 0
                              0058
                              0058         $PRTCTINI B^5$,#<MCHK$M_NEXM!MCHK$M_LOG>
                              0064
                04 A4     D4  0064         CLRL    4(R4)                   ;DISABLE ADAPTER INTERRUPTS (HOPEFULLY)
              51   64     D0  0067         MOVL    (R4),R1                 ;GET ADAPTER CONFIGURATION REG CONTENTS
              30   51     91  006A         CMPB    R1,#NDT$_DR32           ;IS THIS A DR32?
                   07     12  006D         BNEQ    1$                      ;BRANCH IF NOT
     64   00000500 8F     D0  006F         MOVL    #^X500,(R4)             ;ELSE CLEAR INTERRUPTS IN SPECIAL WAY
```

```
          54      64   D0  0076  1$:     MOVL    (R4),R4                  ;GET THE ADAPTER'S CONFIGURATION REG
                           0079
                           0079          $PRTCTEND 5$
               37 50   E9  007A          BLBC    R0,100$                  ;IF R0 LBC, THEN NO ADPATER PRESENT
                           007D
     00000000'FF43     D5  007D          TSTL    @EXE$GL_CONFREGL[R3]     ;ALREADY CONFIGURED?
                  08   12  0084          BNEQ    10$                      ;IF NEQ, YES
  00000000'FF43   54   9A  0086          MOVZBL  R4,@EXE$GL_CONFREGL[R3] ;SAVE THE ADAPTER TYPE
                           008E  10$:
          51      18   D0  008E          MOVL    #EMB$C_UI_LENGTH,R1      ;SET SIZE OF MESSAGE TO ALLOCATE
        00000251'EF    16  0091          JSB     ERL$ALLOCEMB             ;ALLOCATE AN ERROR LOG BUFFER
               14 50   E9  0097          BLBC    R0,20$                   ;BRANCH IF NONE AVAILABLE
    04 A2    0061 8F   B0  009A          MOVW    #EMB$C_UI,EMB$W_UI_ENTRY(R2) ;SET MESSAGE TYPE
          10 A2   53   D0  00A0          MOVL    R3,EMB$L_UI_TR(R2)       ;SET SLOT/TR NUMBER
          14 A2   54   D0  00A4          MOVL    R4,EMB$L_UI_CSR(R2)      ;SET CONFIGURATION REGISTER VALUE
        00000325'EF    16  00A8          JSB     ERL$RELEASEMB            ;RELEASE BUFFER
                  1F   BA  00AE  20$:    POPR    #^M<R0,R1,R2,R3,R4>      ;RESTORE REGISTERS
          5E      04   C0  00B0          ADDL    #4,SP                    ;REMOVE SLOT NUMBER
                       02  00B3          REI                              ;
                           00B4
                  54   D4  00B4  100$:   CLRL    R4                       ;FLAG NO ADAPTER PRESETN
                  D6   11  00B6          BRB     10$                      ;JOIN COMMON CODE
                           00B8
                           00B8  ;
                           00B8  ; UNEXPECTED CPU INTERRUPT HANDLER:  IF DEBUG IS ENABLED, BUGCHECK WITH
                           00B8  ; <VECTOR OFFSET>/2 INTO SCB AS TOP BYTE ON STACK. IF DEBUG IS DISABLED,
                           00B8  ; JUST BUGCHECK.
                           00B8  ;
                           00B8          .ALIGN  LONG
                           00B8  CPU_UNEXP:
                 00000000  00B8          VNUM=000                         ;FIRST VECTOR = 0
                           00B8          .REPT   64                       ;ISR'S FOR CPU INTERRUPTS ONLY
                           00B8          ISRDEF  \VNUM                    ;DEFINE ERL$INT'VNUM LABEL AND ISR
                           00B8          VNUM=VNUM+4                      ;NEXT VECTOR
                           00B8          .ENDR                            ;
                           00B8
                           00B8  ERL$UNEXP::                              ;
                           00B8          .IF     DF,DEBUG                 ;***IF VECTOR ID ENABLED,...
                           00B8          MOVZBL  @(SP),(SP)               ;***OVERLAY RETURN WITH VECTOR OFFSET
                           00B8          MULL    #2,(SP)                  ;***CONVERT ARG TO VECTOR OFFSET
                           00B8          .IFTF
                           00B8          BUG_CHECK UNXINTEXC              ;BUGCHECK
                           00BC          .IFT
                           00BC          TSTL    (SP)+                    ;***CLEAN STACK
                           00BC          .ENDC                            ;
                       02  00BC          REI                              ;RETURN FROM INTERRUPT
                           00BD
                           00BD  ;
                           00BD  ; Vector entry for counting unexpected interrupts, rather than logging
                           00BD  ; them.  Used on 11'780 for passive release on the DW780 and for the
                           00BD  ; CVTP microcode bug.
                           00BD  ;
                           00BD          .ALIGN  LONG
                           00C0
                           00C0  ERL$VEC_RETURN::
        00000000'EF    D6  00C0          INCL    IO$GL_SCB_INTO           ; Increment counter
                       02  00C6          REI                              ; And return
```

```
      00C7   312
      00C7   313
```

```
                            00C7          .SBTTL  LOG DEVICE ERRORS
                            00C7     ;+
                            00C7     ; ERL$DEVICERR - LOG DEVICE CONTROLLER AND/OR DRIVE ERROR
                            00C7     ; ERL$DEVICTMO - LOG DEVICE TIMEOUT ERROR
                            00C7     ;
                            00C7     ; THIS ROUTINE IS CALLED TO LOG A DEVICE TIMEOUT OR DEVICE CONTROLLER
                            00C7     ; AND/OR DRIVE ERROR.
                            00C7     ;
                            00C7     ; INPUTS:
                            00C7     ;
                            00C7     ;       R5 = DEVICE UNIT UCB ADDRESS.
                            00C7     ;
                            00C7     ; OUTPUTS:
                            00C7     ;
                            00C7     ;       IF AN ERROR LOG ENTRY IS NOT ALREADY IN PROGRESS ON THE UNIT, ERROR
                            00C7     ;       LOGGING IS ENABLED FOR THE UNIT, AND THE CURRENT REQUEST DOES NOT
                            00C7     ;       INHIBIT ERROR LOGGING, THEN AN ERROR MESSAGE BUFFER IS ALLOCATED AND
                            00C7     ;       FILLED IN WITH PERTINENT REQUEST INFORMATION FOLLOWED BY A DUMP OF
                            00C7     ;       THE DEVICE REGISTERS.
                            00C7     ;
                            00C7     ;       ALL REGISTERS ARE PRESERVED ACROSS CALL.
                            00C7     ;-
                            00C7
                        00000000          .PSECT  WIONONPAGED
                            0000          .ENABL  LSB
                            0000 ERL$DEVICERR::                        ;LOG DEVICE CONTROLLER AND/OR DRIVE ERROR
               01       DD  0000          PUSHL   #EMB$C_DE            ;SET FOR DEVICE ERROR
               05       11  0002          BRB     10$                  ;
                            0004 ERL$DEVICTMO::                        ;LOG DEVICE TIMEOUT EPROR
      7E   0060 8F      3C  0004          MOVZWL  #EMB$C_DT,-(SP)      ;SET FOR DEVICE TIMEOUT
   03 38 A5     16      E0  0009 10$:     BBS     #DEV$V_ELG,UCB$L_DEVCHAR(R5),15$ ;IF SET, ERROR LOG ENABLED
             0081       31  000E 12$:     BRW     40$                  ;ERROR LOG DISABLED
 F7 009A C5     0B      E0  0011 15$:     BBS     #IO$V_INHERLOG,UCB$W_FUNC(R5),12$ ;IF SET, ERROR LOG INHIBITED
          0082 C5       B6  0017          INCW    UCB$W_ERRCNT(R5)     ;INCREMENT NUMBER OF DEVICE ERRORS
   72 64 A5     02      E0  001B          BBS     #UCB$V_ERLOGIP,UCB$W_STS(R5),40$ ;IF SET, ERROR IN PROGRESS
          004F 8F       BB  0020          PUSHR   #^M<R0,R1,R2,R3,R6>  ;SAVE REGISTERS
      53    28 A5       D0  0024          MOVL    UCB$L_DDB(R5),R3     ;GET ADDRESS OF DDB
      56  0088 C5       D0  0028          MOVL    UCB$L_DDT(R5),R6     ;GET ADDRESS OF DDT (from UCB not DDB)
      51    16 A6       3C  002D          MOVZ'/L DDT$W_ERRORBUF(R6),R1 ;GET SIZE OF ERROR LOG BUFFER IN BYTES
      00000251'EF       16  0031          JSB     ERL$ALLOCEMB         ;ALLOCATE ERROR MESSAGE BUFFER
            54 50       E9  0037          BLBC    R0,30$               ;IF LBC ALLOCATION FAILURE
    0094 C5     52      D0  003A          MOVL    R2,UCB$L_EMB(R5)     ;SAVE ADDRESS OF ERROR MESSAGE BUFFER
      64 A5     04      A8  003F          BISW    #UCB$M_ERLOGIP,UCB$W_STS(R5) ;SIGNAL ERROR LOGGING IN PROGRESS
   04 A2     14 AE      B0  0043          MOVW    5*4(SP),EMB$W_DV_ENTRY(R^) ;INSERT ENTRY TYPE
      52        1C      C0  0048          ADDL    #EMB$B_DV_CLASS,R2   ;POINT TO DEVICE CLASS
                            C)4B
                            004B          ASSUME  EMB$B_DV_TYPE   EQ   EMB$B_DV_CLASS+1
      82     40 A5      B0  004B          MOVW    UCB$B_DEVCLASS(R5),(R2)+ ;INSERT DEVICE CLASS AND TYPE
      51     58 A5      D0  004F          MOVL    UCB$L_IRP(R5),R1     ;GET ADDRESS OF I/O PACKET
                            0053
                            0053          ASSUME  EMB$L_DV_RQPID  EQ   EMB$B_DV_TYPE+1
      82     0C A1      D0  0053          MOVL    IRP$L_PID(R1),(R2)+  ;INSERT REQUESTER PROCESS ID
                            0057
                            0057          ASSUME  EMB$W_DV_BOFF   EQ   EMB$L_DV_RQPID+4
                            0057          ASSUME  EMB$W_DV_BCNT   EQ   EMB$W_DV_BOFF+2
      82     30 A1      D0  0057          MOVL    IRP$W_BOFF(R1),(R2)+ ;INSERT TRANSFER PARAMETERS
                            005B
```

```
                          005B              ASSUME  EMB$L_DV_MEDIA  EQ      EMB$W_DV_BCNT+2
82    00BC C5      D0     005B              MOVL    UCB$L_MEDIA(R5),(R2)+   ;INSERT SIZE OF DISK
                          0060
                          0060              ASSUME  EMB$W_DV_UNIT   EQ      EMB$L_DV_MEDIA+4
  82    54 A5      B0     0060              MOVW    UCB$W_UNIT(R5),(R2)+    ;INSERT UNIT NUMBER
                          0064
                          0064              ASSUME  EMB$W_DV_ERRCNT EQ      EMB$W_DV_UNIT+2
82    0082 C5      B0     0064              MOVW    UCB$W_ERRCNT(R5),(R2)+  ;INSERT NUMBER OF DEVICE ERRORS
                          0069
                          0069              ASSUME  EMB$L_DV_OPCNT  EQ      EMB$W_DV_ERRCNT+2
  82    70 A5      D0     0069              MOVL    UCB$L_OPCNT(R5),(R2)+   ;INSERT OPERATIONS COMPLETED
                          006D
                          006D              ASSUME  EMB$L_DV_OWNUIC EQ      EMB$L_DV_OPCNT+4
  50    1C A5      D0     006D              MOVL    UCB$L_ORB(R5),R0        ;GET ORB ADDRESS
     82    60      D0     0071              MOVL    ORB$L_OWNER(R0),(R2)+   ;INSERT VOLUME OWNER UIC
                          0074
                          0074              ASSUME  EMB$L_DV_CHAR   EQ      EMB$L_DV_OWNUIC+4
  82    38 A5      D0     0074              MOVL    UCB$L_DEVCHAR(R5),(R2)+ ;INSERT DEVICE CHARACTERISTICS
                          0078
                          0078              ASSUME  EMB$B_DV_SLAVE  EQ      EMB$L_DV_CHAR+4
82    0090 C5      9B     0078              MOVZBW  UCB$B_SLAVE(R5),(R2)+   ;INSERT SLAVE UNIT NUMBER
                          007D
                          007D              ASSUME  EMB$W_DV_FUNC   EQ      EMB$B_DV_SLAVE+2
  82    20 A1      B0     007D              MOVW    IRP$W_FUNC(R1),(R2)+    ;INSERT FUNCTION VALUE
                          0081
                          0081              ASSUME  EMB$T_DV_NAME   EQ      EMB$W_DV_FUNC+2
7E    52    10     C1     0081              ADDL3   #EMB$L_DV_REGSAV-EMB$T_DV_NAME,R2,-(SP) ;CALCULATE ADDRESS OF REGIST
         0269      30     0085              BSBW    ERL$GETFULLNAME         ; Copy full device name
           50 8ED0        0088              POPL    R0                      ; Restore address of register dump area
        10 B6      16     008B              JSB     @DDT$L_REGDUMP(R6)      ;CALL REGISTER DUMP ROUTINE
     004F 8F       BA     008E      30$:    POPR    #^M<R0,R1,R2,R3,R6>     ;RESTORE REGISTERS
     5E    04      C0     0092      40$:    ADDL2   #4,SP                   ;REMOVE ENTRY TYPE FROM STACK
                   05     0095              RSB                             ;
                          0096              .DSABL  LSB
```

```
                           0096          .SBTTL  LOG ASYCHRONOUS DEVICE ATTENTIONS
                           0096
                           0096  ;+
                           0096  ; ERL$DEVICEATTN - Log asychronous device attention interrupts that are
                           0096  ;         not related to the current I/O operation that may be in progress.
                           0096  ;
                           0096  ; INPUTS:
                           0096  ;
                           0096  ;         R5 => UCB
                           0096  ;
                           0096  ; OUTPUTS:
                           0096  ;
                           0096  ;         If error logging is enabled for the device, an error log buffer
                           0096  ;         is allocated, filled in and released.  There may be an error log
                           0096  ;         in progress for the current device, but this is not taken into
                           0096  ;         account since the current attention interrupt is not related to
                           0096  ;         the I/O that may be in progress.
                           0096  ;
                           0096  ;-
                           0096
                           0096  ERL$DEVICEATTN::
                           0096
       004F 8F      BB     0096          PUSHR   #^M<R0,R1,R2,R3,R6>      ; Save registers.
  56   0088 C5      D0     009A          MOVL    UCB$L_DDT(R5),R6         ; Get address of DDT.
   51    16 A6      3C     009F          MOVZWL  DDT$W_ERRORBUF(R6),R1    ; R1=size of error log buffer in bytes.
       0082 C5      B6     00A3          INCW    UCB$W_ERRCNT(R5)         ; Increment number of device errors.
            16      E1     00A7          BBC     #DEV$V_ELG,-
     5D 38 A5              00A9                  UCB$L_DEVCHAR(R5),30$    ; If clr, error log disabled.
          01A2      30     00AC          BSBW    ERL$ALLOCEMB             ; Allocate error message buffer.
         57 50      E9     00AF          BLBC    R0,30$                   ; If LBC allocation failure.
            52      DD     00B2          PUSHL   R2                       ; Save address of allocated buffer.
       0062 8F      B0     00B4          MOVW    #EMB$C_DA,-
         04 A2             00B8                  EMB$W_DV_ENTRY(R2)       ; Insert entry type.
         64 A5      B0     00BA          MOVW    UCB$W_STS(R5),-          ; Save device status in buffer.
         1A A2             00BD                  EMB$W_DV_STS(R2)
         12 A2      7C     00BF          CLRQ    EMB$Q_DV_IOSB(R2)        ; Clear irrelevant field.
                           00C2
      52    1C      C0     00C2          ADDL    #EMB$B_DV_CLASS,R2       ; R2 => device class field.
                           00C5
                           00C5          ASSUME  EMB$B_DV_TYPE    EQ      EMB$B_DV_CLASS+1
  82     40 A5      B0     00C5          MOVW    UCB$B_DEVCLASS(R5),(R2)+ ; Insert device class and type.
                           00C9
                           00C9          ASSUME  EMB$L_DV_RQPID   EQ      EMB$B_DV_TYPE+1
                           00C9          ASSUME  EMB$W_DV_BOFF    EQ      EMB$L_DV_RQPID+4
                           00C9          ASSUME  EMB$W_DV_BCNT    EQ      EMB$W_DV_BOFF+2
            82      7C     00C9          CLRQ    (R2)+                    ; Clear PID, BOFF and BCNT.
                           00CB
                           00CB          ASSUME  EMB$L_DV_MEDIA   EQ      EMB$W_DV_BCNT+2
  82   00BC C5      D0     00CB          MOVL    UCB$L_MEDIA(R5),(R2)+    ; Insert size of disk.
                           00D0
                           00D0          ASSUME  EMB$W_DV_UNIT    EQ      EMB$L_DV_MEDIA+4
  82     54 A5      B0     00D0          MOVW    UCB$W_UNIT(R5),(R2)+     ; Insert unit number.
                           00D4
                           00D4          ASSUME  EMB$W_DV_ERRCNT EQ       EMB$W_DV_UNIT+2
  82   0082 C5      B0     00D4          MOVW    UCB$W_ERRCNT(R5),(R2)+   ; Insert number of device errors.
                           00D9
                           00D9          ASSUME  EMB$L_DV_OPCNT   EQ      EMB$W_DV_ERRCNT+2
      82    70 A5    D0  00D9           MOVL    UCB$L_OPCNT(R5),(R2)+    ; Insert operations completed.
                         00DD
                         00DD           ASSUME  EMB$L_DV_OWNUIC EQ       EMB$L_DV_OPCNT+4
      50    1C A5    D0  00DD           MOVL    UCB$L_ORB(R5),R0         ;GET ORB ADDRESS
         82    60    D0  00E1           MOVL    ORB$L_OWNER(R0),(R2)+    ; Insert volume owner uic.
                         00E4
                         00E4           ASSUME  EMB$L_DV_CHAR   EQ       EMB$L_DV_OWNUIC+4
      82    38 A5    D0  00E4           MOVL    UCB$L_DEVCHAR(R5),(R2)+ ; Insert device characteristics.
                         00E8
                         00E8           ASSUME  EMB$B_DV_SLAVE  EQ       EMB$L_DV_CHAR+4
     82   0090 C5    9B  00E8           MOVZBW  UCB$B_SLAVE(R5),(R2)+    ; Insert slave unit number.
                         00ED
                         00ED           ASSUME  EMB$W_DV_FUNC   EQ       EMB$B_DV_SLAVE+2
               82    B4  00ED           CLRW    (R2)+                    ; Clear irrelevant function value.
                         00EF
                         00EF           ASSUME  EMB$T_DV_NAME   EQ       EMB$W_DV_FUNC+2
     7E   52    10   C1  00EF           ADDL3   #EMB$C_DV_REGSAV-EMB$T_DV_NAME,R2,-(SP) ;CALCULATE ADDRESS OF REGIST
      53    28 A5    D0  00F3           MOVL    UCB$L_DDB(R5),R3         ; Get address of DDB
          01F7       30  00F7           BSBW    ERL$GETFULLNAME          ; Copy full device name
             50 8ED0     00FA           POPL    R0                       ; Restore address of register dump area
      51    10 A6    D0  00FD           MOVL    DDT$L_REGDUMP(R6),R1     ; R1 => register dump routine.
            61       16  0101           JSB     (R1)                     ; Call register dump routine.
             52 8ED0     0103           POPL    R2                       ; Restore address of allocated buffer.
          021C       30  0106           BSBW    ERL$RELEASEMB            ; Release this error log buffer.
        004F 8F      BA  0109  30$:     POPR    #^M<R0,R1,R2,R3,R6>      ; Restore registers.
                     05  010D           RSB                              ;
                         010E           .DSABL  LSB
```

```
                        010E         .SBTTL  LOG SOFTWARE STATUS
                        010E
                        010E ;+
                        010E ; ERL$LOGSTATUS - Log software status corresponding to a logged message.
                        010E ;
                        010E ; INPUTS:
                        010E ;       R0-R1 contain final I/O status
                        010E ;       R2 => MSCP end message
                        010E ;       R3 => UCB
                        010E ;       R5 => CDRP
                        010E ;
                        010E ; OUTPUTS:
                        010E ;       An error log message (format EMBSPDEF) is allocated and filled in.
                        010E ;       All registers are preserved.
                        010E ;-
                        010E
                        010E ERL$LOGSTATUS::
                        010E
        0082 C3     B6  010E         INCW    UCB$W_ERRCNT(R3)          ; Increment number of device errors.
              16    E1  0112         BBC     #DEV$V_ELG,-              ; If clear, error log disabled.
       71 38 A3         0114                 UCB$L_DEVCHAR(R3),20$
                        0117
          7E  50    7D  0117         MOVQ    R0,-(SP)                  ; Save R0, R1, R2.
              52    DD  011A         PUSHL   R2
   51   0050 8F     3C  011C         MOVZWL  #EMB$K_SP_LENGTH,R1       ; R1 contains length of buffer to alloc
            012D    30  0121         BSBW    ERL$ALLOCEMB              ; Allocate error message buffer.
           5B 50    E9  0124         BLBC    R0,10$                    ; LBC implies allocation failure.
                        0127
        0063 8F     B0  0127         MOVW    #EMB$C_SP,-               ; Indicate type of error log buffer.
           04 A2        012B                 EMB$W_SP_ENTRY(R2)
   50      10 A2    9E  012D         MOVAB   EMB$B_SP_CLASS(R2),R0     ; R0 => where to begin filling.
                        0131
                        0131         ASSUME  UCB$B_DEVTYPE   EQ      UCB$B_DEVCLASS+1
                        0131         ASSUME  EMB$B_SP_TYPE   EQ      EMB$B_SP_CLASS+1
   80      40 A3    B0  0131         MOVW    UCB$B_DEVCLASS(R3),(R0)+; Move Device type and class.
                        0135
                        0135         ASSUME  EMB$W_SP_BOFF   EQ      EMB$B_SP_TYPE+1
   80      D0 A5    B0  0135         MOVW    CDRP$W_BOFF(R5),(R0)+     ; Copy BOFF.
                        0139
                        0139         ASSUME  EMB$L_SP_BCNT   EQ      EMB$W_SP_BOFF+2
   80      D2 A5    D0  0139         MOVL    CDRP$L_BCNT(R5),(R0)+     ; Also byte count.
                        013D
                        013D         ASSUME  EMB$L_SP_MEDIA  EQ      EMB$L_SP_BCNT+4
   80      D8 A5    D0  013D         MOVL    CDRP$L_MEDIA(R5),(R0)+    ; Move media address (LBN).
                        0141
                        0141         ASSUME  EMB$L_SP_RQPID  EQ      EMB$L_SP_MEDIA+4
   80      AC A5    D0  0141         MOVL    CDRP$L_PID(R5),(R0)+      ; Copy requesting PID.
                        0145
                        0145         ASSUME  EMB$Q_SP_IOSB   EQ      EMB$L_SP_RQPID+4
   80      04 AE    7D  0145         MOVQ    4(SP),(R0)+               ; Copy saved I/O status to buffer.
                        0149
                        0149         ASSUME  EMB$W_SP_FUNC   EQ      EMB$Q_SP_IOSB+8
   80      C0 A5    B0  0149         MOVW    CDRP$W_FUNC(R5),(R0)+     ; Copy I/O function code.
                        014D
                        014D         ASSUME  EMB$W_SP_UNIT   EQ      EMB$W_SP_FUNC+2
   80      54 A3    B0  014D         MOVW    UCB$W_UNIT(R3),(R0)+      ; Copy unit number.
                        0151
```

```
                          0151   549            ASSUME  EMB$L_SP_OPCNT  EQ      EMB$W_SP_UNIT+2
       80    70 A3   D0   0151   550            MOVL    UCB$L_OPCNT(R3),(R0)+   ; Copy cummulative operation count.
                          0155   551
                          0155   552            ASSUME  EMB$W_SP_ERRCNT EQ      EMB$L_SP_OPCNT+4
     80    0082 C3   B0   0155   553            MOVW    UCB$W_ERRCNT(R3),(R0)+  ; And also cummulative error count.
                          015A   554
                          015A   555            ASSUME  EMB$W_SP_UCBSTS EQ      EMB$W_SP_ERRCNT+2
       80    64 A3   B0   015A   556            MOVW    UCB$W_STS(R3),(R0)+     ; Copy UCB STS field.
                          015E   557
                          015E   558            ASSUME  EMB$L_SP_OWNUIC EQ      EMB$W_SP_UCBSTS+2
       51    1C A3   D0   015E   559            MOVL    UCB$L_ORB(R3),R1        ;GET ORB ADDRESS
          80    61   D0   0162   560            MOVL    ORB$L_OWNER(R1),(R0)+   ; Copy device owner UIC.
                          0165   561
                          0165   562            ASSUME  EMB$L_SP_CHAR   EQ      EMB$L_SP_OWNUIC+4
       80    38 A3   D0   0165   563            MOVL    UCB$L_DEVCHAR(R3),(R0)+ ; Copy device characteristics.
                          0169   564
                          0169   565            ASSUME  EMB$L_SP_CMDREF EQ      EMB$L_SP_CHAR+4
          51    6E   D0   0169   566            MOVL    (SP),R1                 ; R1 => MSCP end message.
          80    61   D0   016C   567            MOVL    MSCP$L_CMD_REF(R1),(R0)+; Copy command reference number (RSPID).
                          016F   568
                          016F   569            ASSUME  EMB$T_SP_DEVNAM EQ      EMB$L_SP_CMDREF+4
          7E    52   7D   016F   570            MOVQ    R2,-(SP)                ; Save UCB & buffer base address(R2,R3)
          52    50   D0   0172   571            MOVL    R0,R2                   ; Get buffer adderss
       53    28 A3   D0   0175   572            MOVL    UCB$L_DDB(R3),R3        ; Get DDB address
              0175   30   0179   573            BSBW    ERL$GETFULLNAME         ; Copy full device name
          52    8E   7D   017C   574            MOVQ    (SP)+,R2                ; Restore R2 and R3
                          017F   575
              01A3   30   017F   576            BSBW    ERL$RELEASEMB           ; Release filled in error buffer.
                          0182   577 10$:
               52 8ED0    0182   578            POPL    R2                      ; Restore registers R2, R1, R0.
          50    8E   7D   0185   579            MOVQ    (SP)+,R0
                          0188   580 20$:
                     05   0188   581            RSB                             ; Return to caller.
```

```
                         0189   583             .SBTTL  LOG DRIVER MESSAGE
                         0189   584
                         0189   585 ;+
                         0189   586 ; ERL$LOGMESSAGE - Subroutine to allocate a message buffer, fill in a
                         0189   587 ;               standard header, and then copy caller specified test to the rest
                         0189   588 ;               of the buffer.
                         0189   589 ;
                         0189   590 ; INPUTS:
                         0189   591 ;               R0 =  Code specifying message sub type.
                         0189   592 ;               R1 =  length of caller specified text
                         0189   593 ;               R2 => caller text
                         0189   594 ;               R3 => UCB
                         0189   595 ;
                         0189   596 ; OUTPUTS:
                         0189   597 ;               Message allocated and filled.  All registers preserved.
                         0189   598 ;-
                         0189   599
                         0189   600 ERL$LOGMESSAGE::
                         0189   601
        0082 C3    B6    0189   602             INCW    UCB$W_ERRCNT(R3)          ; Increment total number of errors.
          16       E1    018D   603             BBC     #DEV$V_ELG,-              ; Clear means error logging inhibited.
       55 38 A3          018F   604                     UCB$L_DEVCHAR(R3),20$
                         0192   605
       7E    50    7D    0192   606             MOVQ    R0,-(SP)                  ; Save registers R0-R5.
       7E    52    7D    0195   607             MOVQ    R2,-(SP)
       7E    54    7D    0198   608             MOVQ    R4,-(SP)
       51    26    C0    019B   609             ADDL    #EMB$K_LM_LENGTH,R1       ; Add message header to caller's length.
          00B0     30    019E   610             BSBW    ERL$ALLOCEMB              ; Allocate buffer.
          3A 50    E9    01A1   611             BLBC    R0,10$                    ; LBC means allocation failure.
                         01A4   612
             52    DD    01A4   613             PUSHL   R2                        ; Save address of buffer.
        0064 8F    B0    01A6   614             MOVW    #EMB$C_LM,-               ; Indicate type of error log buffer.
          04 A2          01AA   615                     EMB$W_LM_ENTRY(R2)
                         01AC   616
                         01AC   617             ASSUME  UCB$B_DEVTYPE   EQ      UCB$B_DEVCLASS+1
                         01AC   618             ASSUME  EMB$B_LM_TYPE   EQ      EMB$B_LM_CLASS+1
          40 A3    B0    01AC   619             MOVW    UCB$B_DEVCLASS(R3),-      ; Begin to fill in buffer.  Copy
          10 A2          01AF   620                     EMB$B_LM_CLASS(R2)        ;  Device type and class.
                         01B1   621
          54 A3    B0    01B1   622             MOVW    UCB$W_UNIT(R3),-          ; Also copy device unit number.
          12 A2          01B4   623                     EMB$W_LM_UNIT(R2)
                         01B6   624
             53    DD    01B6   625             PUSHL   R3                        ; Save UCB
     52   14 A2    DE    01B8   626             MOVAL   EMB$T_LM_DEVNAM(R2),R2    ; Get buffer address for device name
     53   28 A3    D0    01BC   627             MOVL    UCB$L_DDB(R3),R3          ; Get DDB address
          012E     30    01C0   628             BSBW    ERL$GETFULLNAME           ; Copy full device name
             53  8ED0    01C3   629             POPL    R3                        ; Restore UCB
       52    6E    D0    01C6   630             MOVL    (SP),R2                   ; Restore buffer base address
                         01C9   631
 24 A2    14 AE    B0    01C9   632             MOVW    20(SP),EMB$W_LM_MSGTYP(R2)          ; Copy message subtype.
    51    0C AE    D0    01CE   633             MOVL    12(SP),R1                           ; R1 => caller's text.
26 A2   61   18 AE    28    01D2   634             MOVC3   24(SP),(R1),EMB$W_LM_MSGTYP+2(R2); Copy caller's text.
             52  8ED0    01D8   635             POPL    R2                                  ; R2 => allocated buffer.
          0147     30    01DA   636             BSBW    ERL$RELEASEMB                       ; Release buffer.
                         01DE   637 10$:
       54    8E    7D    01DE   638             MOVQ    (SP)+,R4                  ; Restore Registers R0-R5.
       52    8E    7D    01E1   639             MOVQ    (SP)+,R2
```

```
50   8E   7D  01E4            MOVQ    (SP)+,R0
              01E7      20$:
          05  01E7            RSB                             ; Return to caller.
```

```
                        01E8          .SBTTL  ERL$LOG_DMSCP and ERL$LOG_TMSCP
                        01E8
                        01E8  ;+
                        01E8  ; Routines that respectively log invalid Disk and Tape MSCP messages.
                        01E8  ;
                        01E8  ; Inputs:
                        01E8  ;       R0 =  type of message
                        01E8  ;       R1 =  length of message
                        01E8  ;       R2 => message
                        01E8  ;       R3 => CDDB
                        01Ed  ;
                        01E8  ; Outputs:
                        01E8  ;       All registers preserved.
                        01E8  ;
                        01E8  ;       We want to log the following items in addition to the message
                        01E8  ;       and its type:
                        01E8  ;
                        01E8  ;               1. CDDB$B_SYSTEMID (6 bytes)
                        01E8  ;               2. The ASCII string "DISK" (4 bytes) or "TAPE" (4 bytes)
                        01E8  ;               3. CDDB$Q_CNTRLID (8 bytes)
                        01E8  ;
                        01E8
                        01E8          .enabl  lsb
                        01E8
                        01E8  ERL$LOG_TMSCP::
                        01E8
               3F   BB  01E8          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers.
 54    45504154 8F   D0  01EA          MOVL    #^A/TAPE/,R4            ; R4 has string "TAPE".
               09   11  01F1          BRB     10$                     ; Branch around to common code.
                        01F3  ERL$LOG_DMSCP::
                        01F3
               3F   BB  01F3          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers.
 54    4B534944 8F   D0  01F5          MOVL    #^A/DISK/,R4            ; R4 has string "DISK".
                        01FC  10$:
          51   24   C0  01FC          ADDL    #<2+4+6+8 -             ; R1 has length which is bumped by
                        01FF                   +EMB$K_HD_LENGTH>,R1   ;  2 for the type, 4 for "DISK" or
                        01FF                                          ;  "TAPE", 6 for SYSTEMID, 8 for
                        01FF                                          ;  CNTRLID, and errorlog entry header
                        01FF                                          ;  size.
             004F   30  01FF          BSBW    ERL$ALLOCEMB            ; Allocate Errorlog Buffer.
            27 50   E9  0202          BLBC    R0,20$                  ; LBC means no allocate.
               52   DD  0205          PUSHL   R2                      ; Save R2=>Buffer.
          0065 8F   B0  0207          MOVW    #EMB$C_LOGMSCP,-        ; Copy message class to buffer header.
            04 A2       020B                  EMB$W_HD_ENTRY(R2)
          52   10   C0  020D          ADDL    #EMB$K_HD_LENGTH,R2     ; R2 => beyond header.
       82   04 AE   B0  0210          MOVW    4(SP),(R2)+             ; Copy message type (from saved regs).
          82   54   D0  0214          MOVL    R4,(R2)+                ; Copy Class driver type.
       82   20 A3   7D  0217          MOVQ    CDDB$Q_CNTRLID(R3),(R2)+; Controller identifier.
       82   0C A3   7D  021B          MOVQ    CDDB$B_SYSTEMID(R3),(R2)+;And System ID.
            08 AE   28  021F          MOVC3   8(SP),-                 ; Get length from saved registers.
            0C BE       0222                  @12(SP),-               ;  also source address.
            FE A2       0224                  -2(R2)                  ; Target is -2 since SYSTEMID is 6 bytes.
               52 8ED0  0226          POPL    R2                      ; Restore R2=>Buffer.
             00F9   30  0229          BSBW    ERL$RELEASEMB           ; Free Errorlog buffer.
                        022C  20$:
               3F   BA  022C          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers.
                    05  022E          RSB
```

```
         022F   701         .dsabl  lsb
```

```
                  022F          .SBTTL  BUILD STARTUP AND POWERFAIL MESSAGES
                  022F  ;+
                  022F  ; ERL$COLDSTART - LOG COLDSTART (SYSTEMBOOT)
                  022F  ;
                  022F  ; THIS ROUTINE IS CALLED BY SYSINIT AFTER CORRECTLY SETTING THE SYSTEM
                  022F  ; TIME TO LOG THE BOOTING OF THE SYSTEM.
                  022F  ;
                  022F  ; ERL$WARMSTART - LOG WARMSTART (POWER RECOVERY)
                  022F  ;
                  022F  ; THIS ROUTINE IS CALLED BY POWERFAIL AFTER CORRECTING THE SYSTEM TIME
                  022F  ; TO LOG THE POWER FAIL AND RECOVERY.
                  022F  ;
                  022F  ; INPUTS:
                  022F  ;       NONE
                  022F  ;
                  022F  ; OUTPUTS:
                  022F  ;
                  022F  ; AN ERROR LOG BUFFER IS ALLOCATED AND FILLED WITH THE APPROPRIATE MESSAGE
                  022F  ; IF POSSIBLE AND THE ERROR LOG PROCESS AWAKENED IF NECESSARY.
                  022F  ;-
                  022F          .ENABL  LSB
                  022F  ERL$COLDSTART::                        ;
    53   20   3C  022F          MOVZWL  #EMB$C_CS,R3           ;SET TYPE OF MESSAGE TO COLDSTART
         03   11  0232          BRB     10$                    ;
                  0234  ERL$WARMSTART::                        ;
    53   24   3C  0234          MOVZWL  #EMB$C_WS,R3           ;SET TYPE OF MESSAGE TO WARMSTART
    51   14   3C  0237  10$:    MOVZWL  #EMB$C_SU_LENGTH,R1    ;SET SIZE OF MESSAGE TO ALLOCATE
         15   10  023A          BSBB    ERL$ALLOCEMB           ;ALLOCATE AN ERROR LOG BUFFER
      11 50   E9  023C          BLBC    R0,20$                 ;BRANCH IF NONE AVAILABLE
 00000000'EF  16  023F          JSB     EXE$READ_TODR          ;GET TIME TO LOG
   10 A2  50  D0  0245          MOVL    R0,EMB$L_SU_DAYTIM(R2) ;LOG TIME OF DAY CLOCK
   04 A2  53  B0  0249          MOVW    R3,EMB$W_SU_ENTRY(R2)  ;SET MESSAGE TYPE
       00D5   30  024D          BSBW    ERL$RELEASEMB          ;RELEASE BUFFER
              05  0250  20$:    RSB                            ;
                  0251          .DSABL  LSB                    ;
```

```
                                    0251          .SBTTL  ALLOCATE ERROR MESSAGE BUFFER
                                    0251     ;+
                                    0251     ; ERL$ALLOCEMB - ALLOCATE ERROR MESSAGE BUFFER
                                    0251     ;
                                    0251     ; THIS ROUTINE IS CALLED TO ALLOCATE AN ERROR LOG MESSAGE BUFFER AND
                                    0251     ; INITIALIZE ITS HEADER.
                                    0251     ;
                                    0251     ; INPUTS:
                                    0251     ;
                                    0251     ;        R1 = SIZE OF MESSAGE BUFFER REQUIRED IN BYTES.
                                    0251     ;
                                    0251     , OUTPUTS:
                                    0251     ;
                                    0251     ;        R0 LOW BIT CLEAR INDICATES AN ALLOCATION FAILURE.
                                    0251     ;
                                    0251     ;        R0 LOW BIT SET INDICATES SUCCESSFUL ALLOCATION WITH:
                                    0251     ;
                                    0251     ;               R1 = ERROR SEQUENCE NUMBER.
                                    0251     ;               R2 = ADDRESS OF ALLOCATED ERROR MESSAGE BUFFER.
                                    0251     ;
                                    0251     ;        IN EITHER CASE THE UNIVERSAL ERROR SEQUENCE NUMBER IS INCREMENTED
                                    0251     ;        AND STORED IN THE BUFFER AT THE STANDARD PLACE, ALONG WITH THE TIME.
                                    0251     ;        AND THE ERROR LOG PROCESS MAY BE AWAKENED IF AN ERROR ALLOCATION
                                    0251     ;        BUFFER IS FOUND TO BE FULL.
                                    0251     ;
                                    0251     ;        R3 IS PRESERVED ACROSS CALL.
                                    0251     ;-
                                    0251
                                    0251    ERL$ALLOCEMB::                                  ;ALLOCATE ERROR MESSAGE BUFFER
                                    0251            DSBINT                                  ;DISABLE ALL INTERRUPTS
              51     04       C0    0257            ADDL    #EMB$K_LENGTH,R1                ; Add in size of header for message
     50    00000408'EF       9A    025A            MOVZBL  ERL$GB_BUFIND,R0                ;GET CURRENT ALLOCATION BUFFER INDICATOR
    50    00000400'EF40      D0    0261            MOVL    ERL$AL_BUFADDR[R0],R0           ;GET ADDRESS OF ALLOCATION BUFFER DESCRIPTOR
         1E 03 A0    00      E0    0269            BBS     #ERL$V_LOCK,ERL$B_FLAGS(R0),15$; IF SET, BUFFER BEING COPIED
            52   04 A0       D0    026E    10$:    MOVL    ERL$L_NEXT(R0),R2               ;GET ADDRESS OF NEXT AVAILABLE SPACE
      04 A0    52    51      C1    0272            ADDL3   R1,R2,ERL$L_NEXT(R0)            ;CALCULATE ADDRESS OF NEXT AVAILABLE SPACE
        04 A0     08 A0      D1    0277            CMPL    ERL$L_END(R0),ERL$L_NEXT(R0) ;ENTRY FIT WITHIN BUFFER?
                     42      1E    027C            BGEQU   20$                             ;IF GEQU YES
    00000409'EF      02      88    027E            BISB    #ERL$M_TIMER,ERL$GB_BUFFLAG ;SET TIMER ACTIVE
    0000040B'EF      01      90    0285            MOVB    #1,ERL$GB_BUFTIM                ;FORCE ERROR LOG PROCESS WAKE
        04 A0     08 A0      D0    028C    15$:    MOVL    ERL$L_END(R0),ERL$L_NEXT(R0) ;INDICATE THAT BUFFER IS FULL
    00000408'EF      01      8C    0291            XORB    #1,ERL$GB_BUFIND                ;SWITCH TO ALTERNATE BUFFER
     50    00000408'EF       9A    0298            MOVZBL  ERL$GB_BUFIND,R0                ;GET NEW BUFFER INDICATOR
    50    00000400'EF40      D0    029F            MOVL    ERL$AL_BUFADDR[R0],R0           ;GET ADDRESS OF ALLOCATION BUFFER DESCRIPTOR
         0B 03 A0    00      E0    02A7            BBS     #ERL$V_LOCK,ERL$B_FLAGS(R0),17$; IF SET, BUFFER BEING COPIED
      52    04 A0    51      C1    02AC            ADDL3   R1,ERL$L_NEXT(R0),R2            ;CALCULATE ADDRESS OF NEXT AVAILABLE SPACE
            52    08 A0      D1    02B1            CMPL    ERL$L_END(R0),R2                ;ENTRY FIT WITHIN BUFFER?
                     B7      1E    02B5            BGEQU   10$                             ;IF GEQU YES
        04 A0     08 A0      D0    02B7    17$:    MOVL    ERL$L_END(R0),ERL$L_NEXT(R0) ;INDICATE THAT BUFFER IS FULL
                     50      D4    02BC            CLRL    R0                              ;INDICATE ALLOCATION FAILURE
                     27      11    02BE            BRB     30$                             ;
              52     04      C0    02C0    20$:    ADDL    #EMB$K_LENGTH,R2                ; Point past the message header
              62     3E      DB    02C3            MFPR    #PR$_SID,EMB$L_HD_SID(R2)         ; Set system ID into message
           FC A2     51      3C    02C6            MOVZWL  R1,EMB$W_SIZE(R2)               ; Set size in message header
        FE A2     02 A0      90    02CA            MOVB    ERL$B_BUFIND(R0),EMB$B_BUFIND(R2) ;SET RESPECTIVE BUFFER INDICATOR
                     60      96    02CF            INCB    ERL$B_BUSY(R0)                  ;INCREMENT MESSAGE BUSY COUNT
     51    00000410'EF       D0    02D1            MOVL    ERL$GL_SEQUENCE,R1              ;GET CURRENT ERROR SEQUENCE NUMBER
```

```
         06 A2    00000000'EF    7D  02D8    796          MOVQ     EXE$GQ_SYSTIME,EMB$Q_DV_TIME(R2) ;INSERT CURRENT TIME
               0E A2     51      B0  02E0    797          MOVW     R1,EMB$W_DV_ERRSEQ(R2)  ;INSERT ERROR SEQUENCE NUMBER
                   50    01      D0  02E4    798          MOVL     #1,R0                   ;SET SUCCESS INDICATOR
                  00000410'EF    D6  02E7    799 30$:     INCL     ERL$GL_SEQUENCE         ;INCREMENT UNIVERSAL ERROR SEQUENCE NUMBER
                                     02ED    800          ENBINT                           ;ENABLE INTERRUPTS
                                 05  02F0    801          RSB                              ;
```

```
                          02F1   803          .SBTTL  GET FULL DEVICE NAME
                          02F1   804  ;+
                          02F1   805  ; ERL$GETFULLNAME - GET FULL DEVICE NAME
                          02F1   806  ;
                          02F1   807  ; THIS ROUTINE IS CALLED TO COPY THE FULL DEVICE NAME (NODE NAME + DEVICE NAME)
                          02F1   808  ; TO THE ERROR LOG BUFFER.
                          02F1   809  ;
                          02F1   810  ; INPUTS:
                          02F1   811  ;
                          02F1   812  ;       R3 = address of DDB
                          02F1   813  ;       R2 = address of error log buffer
                          02F1   814  ;
                          02F1   815  ; OUTPUTS:
                          02F1   816  ;
                          02F1   817  ;       If a node name exist in the system block, it is copied with the
                          02F1   818  ;       device name to the error log buffer.
                          02F1   819  ;
                          02F1   820  ;       R0, R1, AND R3 ARE DESTROYED ACROSS CALL.
                          02F1   821  ;-
                          02F1   822
                          02F1   823  ERL$GETFULLNAME::
      51    14 A3    9E   02F1   824          MOVAB   DDB$T_NAME(R3),R1        ; Get address of device name.
         7E    81    9A   02F5   825          MOVZBL  (R1)+,-(SP)              ; Save the string length
      53    34 A3    D0   02F8   826          MOVL    DDB$L_SB(R3),R3          ; Get address of system block
               1A    13   02FC   827          BEQL    20$                      ; If EQL, go to move device name
      53    44 A3    9E   02FE   828          MOVAB   SB$T_NODENAME(R3),R3     ; Get address of nodename
         50    83    9A   0302   829          MOVZBL  (R3)+,R0                 ; Get nodename length
               11    13   0305   830          BEQL    20$                      ; If eql, go move device name
    62    6E    50    81  0307   831          ADDB3   R0,(SP),(R2)             ; Nodename length + device name
               82    96   030B   832          INCB    (R2)+                    ; Total string len. + 1 for "$"
         82    83    90   030D   833  10$:    MOVB    (R3)+,(R2)+              ; Copy nodename
            FA 50    F5   0310   834          SOBGTR  R0,10$
         82    24    90   0313   835          MOVB    #^A/$/,(R2)+             ; Insert the "$"
               03    11   0316   836          BRB     30$                      ; Go move device name
         82    6E    90   0318   837  20$:    MOVB    (SP),(R2)+               ; Move dev. name len. to buffer
         50    8E    D0   031B   838  30$:    MOVL    (SP)+,R0                 ; Get dev. name length
         82    81    90   031E   839  40$:    MOVB    (R1)+,(R2)+              ; Move device name into buffer
            FA 50    F5   0321   840          SOBGTR  R0,40$
                     05   0324   841          RSB                              ; Return to caller
                          0325   842
```

```
                                   0325         .SBTTL  RELEASE ERROR MESSAGE BUFFER
                                   0325  ;+
                                   0325  ; ERL$RELEASEMB - RELEASE ERROR MESSAGE BUFFER
                                   0325  ;
                                   0325  ; THIS ROUTINE IS CALLED TO RELEASE AN ERROR MESSAGE BUFFER FOR PROCESSING
                                   0325  ; BY THE ERROR LOG PROCESS.
                                   0325  ;
                                   0325  ; INPUTS:
                                   0325  ;
                                   0325  ;       R2 = ADDRESS OF ERROR MESSAGE BUFFER.
                                   0325  ;
                                   0325  ; OUTPUTS:
                                   0325  ;
                                   0325  ;       THE COMPLETED ERROR MESSAGE COUNT IS INCREMENTED IN THE RESPECTIVE
                                   0325  ;       ALLOCATION BUFFER HEADER, THE MESSAGE IS SET VALID, AND THE BUSY
                                   0325  ;       MESSAGE COUNT IS DECREMENTED IN THE RESPECTIVE ALLOCATION BUFFER
                                   0325  ;       HEADER.
                                   0325  ;
                                   0325  ;       R3 IS PRESERVED ACROSS CALL.
                                   0325  ;-
                                   0325
                                   0325  ERL$RELEASEMB::                                 ;RELEASE ERROR MESSAGE BUFFER
                 FF A2     96      0325         INCB    EMB$B_VALID(R2)         ;SET MESSAGE BUFFER VALID
            50   FE A2     9A      0328         MOVZBL  EMB$B_BUFIND(R2),R0     ;GET BUFFER INDICATOR OF ALLOCATION BUFFER
     50   00000400'EF40    D0      032C         MOVL    ERL$AL_BUFADDR[R0],R0   ;GET ADDRESS OF ALLOCATION BUFFER DESCRIPTOR
          60    00FF 8F    58      0334         ADAWI   #^XFF,ERL$B_BUSY(R0)    ;ADJUST BUSY AND COMPLETED MESSAGE COUNT
  0D 00000409'EF      01   E3      0339         BBCS    #ERL$V_TIMER,ERL$GB_BUFFLAG,10$ ;IF CLR, NO TIMER RUNNING
            01 A0     0A   91      0341         CMPB    #MAXMSG,ERL$B_MSGCNT(R0) ;MAXIMUM NUMBER OF MESSAGES EXCEEDED?
                      07   1A      0345         BGTRU   10$                     ;IF GTRU NO
     0000040B'EF      01   90      0347         MOVB    #1,ERL$GB_BUFTIM        ;FORCE ERROR LOG PROCESS WAKE
                           05      034E  10$:   RSB                             ;
```

```
                              034F   876         .SBTTL  WAKE ERROR LOG FORMAT PROCESS
                              034F   877 ;+
                              034F   878 ; ERL$WAKE - WAKE ERROR LOG FORMAT PROCESS
                              034F   879 ;
                              034F   880 ; THIS ROUTINE IS CALLED ONCE A SECOND WHEN THE ERROR BUFFER TIMER IS ACTIVE.
                              034F   881 ;
                              034F   882 ; INPUTS:
                              034F   883 ;
                              034F   884 ;       NONE.
                              034F   885 ;
                              034F   886 ; OUTPUTS:
                              034F   887 ;
                              034F   888 ;       THE ERROR BUFFER TIMER IS DECREMENTED AND IF THE RESULT IS ZERO THE
                              034F   889 ;       ERROR LOG FORMAT PROCESS IS AWAKENED.
                              034F   890 ;-
                              034F   891
                              034F   892 ERL$WAKE::                              ;WAKE ERROR LOG FORMAT PROCESS
        0000040B'EF      97   034F   893         DECB    ERL$GB_BUFTIM           ;DECREMENT TIMER
                 18      12   0355   894         BNEQ    10$                     ;
 00000409'EF     02      8A   0357   895         BICB    #ERL$M_TIMER,ERL$GB_BUFFLAG ;CLEAR TIMER ACTIVE FLAG
 0000040B'EF     1E      90   035E   896         MOVB    #MAXTIM,ERL$GB_BUFTIM   ;RESET TIMER VALUE
 51     0000040C'EF      D0   0365   897         MOVL    ERL$GL_ERLPID,R1        ;GET ERROR LOG PROCESS ID
              FC91'      30   036C   898         BSBW    SCH$WAKE                ;WAKE ERROR LOG PROCESS
                         05   036F   899 10$:    RSB                             ;
                              0370   900
                              0370   901         .END
```

```
ADP_HANDLER                       0000003E R      03           EMB$L_UI_CSR                = 00000014
ADP_UNEXP                         00000000 R      03           EMB$L_UI_TR                 = 00000010
BUF1                              00000000 R      02           EMB$Q_DV_IOSB               = 00000012
BUF2                              00000200 R      02           EMB$Q_DV_TIME               = 00000006
BUG$_UNXINTEXC                    ********     X  03           EMB$Q_SP_IOSB               = 00000020
CDDB$B_SYSTEMID                 = 0000000C                     EMB$T_DV_NAME               = 0000003E
CDDB$Q_CNTRLID                  = 00000020                     EMB$T_LM_DEVNAM             = 00000014
CDRP$L_BCNT                     = FFFFFFD2                     EMB$T_SP_DEVNAM             = 00000040
CDRP$L_MEDIA                    = FFFFFFD8                     EMB$W_DV_BCNT               = 00000024
CDRP$L_PID                      = FFFFFFAC                     EMB$W_DV_BOFF               = 00000022
CDRP$W_BOFF                     = FFFFFFD0                     EMB$W_DV_ENTRY              = 00000004
CDRP$W_FUNC                     = FFFFFFC0                     EMB$W_DV_ERRCNT             = 0000002C
CPU_UNEXP                         000000B8 R      03           EMB$W_DV_ERRSEQ             = 0000000E
DDB$L_SB                        = 00000034                     EMB$W_DV_FUNC               = 0000003C
DDB$T_NAME                      = 00000014                     EMB$W_DV_STS                = 0000001A
DDT$L_REGDUMP                   = 00000010                     EMB$W_DV_UNIT               = 0000002A
DDT$W_ERRORBUF                  = 00000016                     EMB$W_HD_ENTRY              = 00000004
DEV$V_ELG                       = 00000016                     EMB$W_LM_ENTRY              = 00000004
EMB$B_BUFIND                    = FFFFFFFE                     EMB$W_LM_MSGTYP             = 00000024
EMB$B_DV_CLASS                  = 0000001C                     EMB$W_LM_UNIT               = 00000012
EMB$B_DV_SLAVE                  = 0000003A                     EMB$W_SIZE                  = FFFFFFFC
EMB$B_DV_TYPE                   = 0000001D                     EMB$W_SP_BOFF               = 00000012
EMB$B_LM_CLASS                  = 00000010                     EMB$W_SP_ENTRY              = 00000004
EMB$B_LM_TYPE                   = 00000011                     EMB$W_SP_ERRCNT             = 00000030
EMB$B_SP_CLASS                  = 00000010                     EMB$W_SP_FUNC               = 00000028
EMB$B_SP_TYPE                   = 00000011                     EMB$W_SP_JCBSTS             = 00000032
EMB$B_VALID                     = FFFFFFFF                     EMB$W_SP_UNIT               = 0000002A
EMB$C_CS                        = 00000020                     EMB$W_SU_ENTRY              = 00000004
EMB$C_DA                        = 00000062                     EMB$W_UI_ENTRY              = 00000004
EMB$C_DE                        = 00000001                     ERL$ALLOCEMB                  00000251 RG     04
EMB$C_DT                        = 00000060                     ERL$AL_BUFADDR                00000400 RG     02
EMB$C_LM                        = 00000064                     ERL$B_BUFIND                = 00000002
EMB$C_LOGMSCP                   = 00000065                     ERL$B_BUSY                  = 00000000
EMB$C_SP                        = 00000063                     ERL$B_FLAGS                 = 00000003
EMB$C_SU_LENGTH                 = 00000014                     ERL$B_MSGCNT                = 00000001
EMB$C_UI                        = 00000061                     ERL$COLDSTART                 0000022F RG     04
EMB$C_UI_LENGTH                 = 00000018                     ERL$C_LENGTH                = 0000000C
EMB$C_WS                        = 00000024                     ERL$DEVICEATTN                00000096 RG     04
EMB$K_HD_LENGTH                 = 00000010                     ERL$DEVICERR                  00000000 RG     04
EMB$K_LENGTH                    = 00000004                     ERL$DEVICTMO                  00000004 RG     04
EMB$K_LM_LENGTH                 = 00000026                     ERL$GB_BUFFLAG                00000409 RG     02
EMB$K_SP_LENGTH                 = 00000050                     ERL$GB_BUFIND                 00000408 RG     02
EMB$L_DV_CHAR                   = 00000036                     ERL$GB_BUFPTR                 0000040A RG     02
EMB$L_DV_MEDIA                  = 00000026                     ERL$GB_BUFTIM                 0000040B RG     02
EMB$L_DV_OPCNT                  = 0000002E                     ERL$GETFULLNAME               000002F1 RG     04
EMB$L_DV_OWNUIC                 = 00000032                     ERL$GL_ERLPID                 0000040C PG     02
EMB$L_DV_REGSAV                 = 0000004E                     ERL$GL_SEQUENCE               00000410 RG     02
EMB$L_DV_RQPID                  = 0000001E                     ERL$LOGMESSAGE                00000189 RG     04
EMB$L_HD_SID                    = 00000000                     ERL$LOGSTATUS                 0000010E RG     04
EMB$L_SP_BCNT                   = 00000014                     ERL$LOG_DMSCP                 000001F3 RG     04
EMB$L_SP_CHAR                   = 00000038                     ERL$LOG_TMSCP                 000001E8 RG     04
EMB$L_SP_CMDREF                 = 0000003C                     ERL$L_END                   = 00000008
EMB$L_SP_MEDIA                  = 00000018                     ERL$L_NEXT                  = 00000004
EMB$L_SP_OPCNT                  = 0000002C                     ERL$M_TIMER                 = 00000002
EMB$L_SP_OWNUIC                 = 00000034                     ERL$RELEASEMB                 00000325 RG     04
EMB$L_SP_RQPID                  = 0000001C                     ERL$UNEXP                     000000B8 RG     03
EMB$L_SU_DAYTIM                 = 00000010                     ERL$VECO                      000000B8 RG     03
```

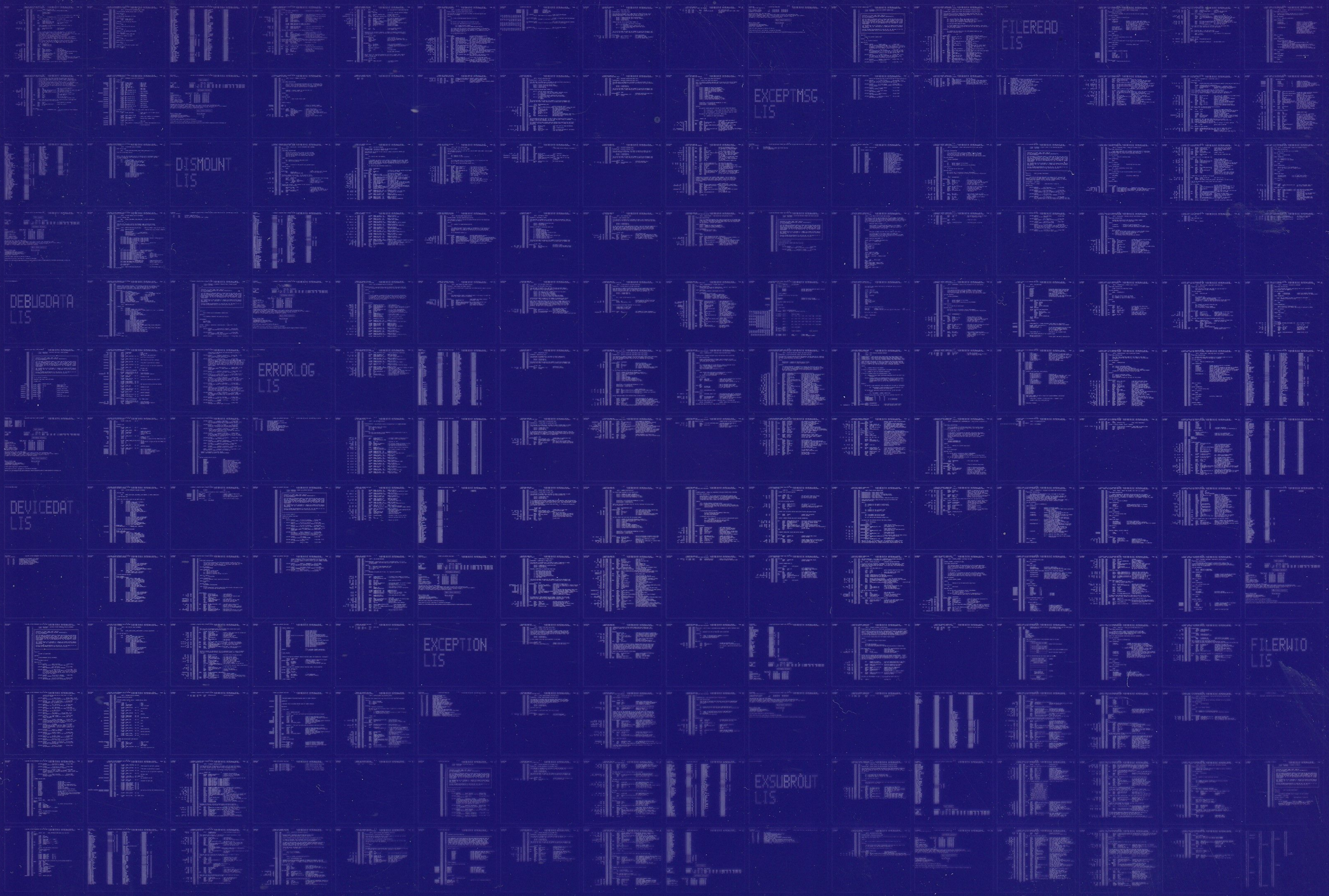
```
ERL$VEC100                  000000B8 RG     03        ERL$VEC308                  00000034 RG     03
ERL$VEC104                  000000B8 RG     03        ERL$VEC312                  00000038 RG     03
ERL$VEC108                  000000B8 RG     03        ERL$VEC316                  0000003C RG     03
ERL$VEC112                  000000B8 RG     03        ERL$VEC32                   000000B8 RG     03
ERL$VEC116                  000000B8 RG     03        ERL$VEC320                  00000000 RG     03
ERL$VEC12                   000000B8 RG     03        ERL$VEC324                  00000004 RG     03
ERL$VEC120                  000000B8 RG     03        ERL$VEC328                  00000008 RG     03
ERL$VEC124                  000000B8 RG     03        ERL$VEC332                  0000000C RG     03
ERL$VEC128                  000000B8 RG     03        ERL$VEC336                  00000010 RG     03
ERL$VEC132                  000000B8 RG     03        ERL$VEC340                  00000014 RG     03
ERL$VEC136                  000000B8 RG     03        ERL$VEC344                  00000018 RG     03
ERL$VEC140                  000000B8 RG     03        ERL$VEC348                  0000001C RG     03
ERL$VEC144                  000000B8 RG     03        ERL$VEC352                  00000020 RG     03
ERL$VEC148                  000000B8 RG     03        ERL$VEC356                  00000024 RG     03
ERL$VEC152                  000000B8 RG     03        ERL$VEC36                   000000B8 RG     03
ERL$VEC156                  000000B8 RG     03        ERL$VEC360                  00000028 RG     03
ERL$VEC16                   000000B8 RG     03        ERL$VEC364                  0000002C RG     03
ERL$VEC160                  000000B8 RG     03        ERL$VEC368                  00000030 RG     03
ERL$VEC164                  000000B8 RG     03        ERL$VEC372                  00000034 RG     03
ERL$VEC168                  000000B8 RG     03        ERL$VEC376                  00000038 RG     03
ERL$VEC172                  000000B8 RG     03        ERL$VEC380                  0000003C RG     03
ERL$VEC176                  000000B8 RG     03        ERL$VEC384                  00000000 RG     03
ERL$VEC180                  000000B8 RG     03        ERL$VEC388                  00000004 RG     03
ERL$VEC184                  000000B8 RG     03        ERL$VEC392                  00000008 RG     03
ERL$VEC188                  000000B8 RG     03        ERL$VEC396                  0000000C RG     03
ERL$VEC192                  000000B8 RG     03        ERL$VEC4                    000000B8 RG     03
ERL$VEC196                  000000B8 RG     03        ERL$VEC40                   000000B8 RG     03
ERL$VEC20                   000000B8 RG     03        ERL$VEC400                  00000010 RG     03
ERL$VEC200                  000000B8 RG     03        ERL$VEC404                  00000014 RG     03
ERL$VEC204                  000000B8 RG     03        ERL$VEC408                  00000018 RG     03
ERL$VEC208                  000000B8 RG     03        ERL$VEC412                  0000001C RG     03
ERL$VEC212                  000000B8 RG     03        ERL$VEC416                  00000020 RG     03
ERL$VEC216                  000000B8 RG     03        ERL$VEC420                  00000024 RG     03
ERL$VEC220                  000000B8 RG     03        ERL$VEC424                  00000028 RG     03
ERL$VEC224                  000000B8 RG     03        ERL$VEC428                  0000002C RG     03
ERL$VEC228                  000000B8 RG     03        ERL$VEC432                  00000030 RG     03
ERL$VEC232                  000000B8 RG     03        ERL$VEC436                  00000034 RG     03
ERL$VEC236                  000000B8 RG     03        ERL$VEC44                   000000B8 RG     03
ERL$VEC24                   000000B8 RG     03        ERL$VEC440                  00000038 RG     03
ERL$VEC240                  000000B8 RG     03        ERL$VEC444                  0000003C RG     03
ERL$VEC244                  000000B8 RG     03        ERL$VEC448                  00000000 RG     03
ERL$VEC248                  000000B8 RG     03        ERL$VEC452                  00000004 RG     03
ERL$VEC252                  000000B8 RG     03        ERL$VEC456                  00000008 RG     03
ERL$VEC256                  00000000 RG     03        ERL$VEC460                  0000000C RG     03
ERL$VEC260                  00000004 RG     03        ERL$VEC464                  00000010 RG     03
ERL$VEC264                  00000008 RG     03        ERL$VEC468                  00000014 RG     03
ERL$VEC268                  0000000C RG     03        ERL$VEC472                  00000018 RG     03
ERL$VEC272                  00000010 RG     03        ERL$VEC476                  0000001C RG     03
ERL$VEC276                  00000014 RG     03        ERL$VEC48                   000000B8 RG     03
ERL$VEC28                   000000B8 RG     03        ERL$VEC480                  00000020 RG     03
ERL$VEC280                  00000018 RG     03        ERL$VEC484                  00000024 RG     03
ERL$VEC284                  0000001C RG     03        ERL$VEC488                  00000028 RG     03
ERL$VEC288                  00000020 RG     03        ERL$VEC492                  0000002C RG     03
ERL$VEC292                  00000024 RG     03        ERL$VEC496                  00000030 RG     03
ERL$VEC296                  00000028 RG     03        ERL$VEC500                  00000034 RG     03
ERL$VEC300                  0000002C RG     03        ERL$VEC504                  00000038 RG     03
ERL$VEC304                  00000030 RG     03        ERL$VEC508                  0000003C RG     03
```

```
ERL$VEC52                         000000B8 RG    03        VECTOR                        = 0000023C
ERL$VEC56                         000000B8 RG    03        VNUM                          = 00000100
ERL$VEC60                         000000B8 RG    03
ERL$VEC64                         000000B8 RG    03
ERL$VEC68                         000000B8 RG    03
ERL$VEC72                         000000B8 RG    03
ERL$VEC76                         000000B8 RG    03
ERL$VEC8                          000000B8 RG    03
ERL$VEC80                         000000B8 RG    03
ERL$VEC84                         000000B8 RG    03
ERL$VEC88                         000000B8 RG    03
ERL$VEC92                         000000B8 RG    03
ERL$VEC96                         000000B8 RG    03
ERL$VEC_RETURN                    000000C0 RG    03
ERL$V_LOCK                      = 00000000
ERL$V_TIMER                     = 00000001
ERL$WAKE                          0000034F RG    04
ERL$WARMSTART                     00000234 RG    04
EXE$GL_CONFREGL                   ********    X  03
EXE$GQ_SYSTIME                    ********    X  04
EXE$MCHK_PRTCT                    ********    X  03
EXE$READ_TODR                     ********    X  04
IO$GL_SCB_INTO                    ********    X  03
IO$V_INHERLOG                   = 0000000B
IRP$L_PID                       = 0000000C
IRP$W_BOFF                      = 00000030
IRP$W_FUNC                      = 00000020
MAXMSG                          = 0000000A
MAXTIM                          = 0000001E
MCHK$M_LOG                      = 00000001
MCHK$M_NEXM                     = 00000004
MMG$GL_SBICONF                    ********    X  03
MSCP$L_CMD_REF                  = 00000000
NDT$_DR32                       = 00000030
NEXUS                           = 00000010
ORB$L_OWNER                     = 00000000
PR$_IPL                         = 00000012
PR$_SID                         = 0000003E
SB$T_NODENAME                   = 00000044
SCH$WAKE                          ********    X  04
UCB$B_DEVCLASS                  = 00000040
UCB$B_DEVTYPE                   = 00000041
UCB$B_SLAVE                     = 00000090
UCB$L_DDB                       = 00000028
UCB$L_DDT                       = 00000088
UCB$L_DEVCHAR                   = 00000038
UCB$L_EMB                       = 00000094
UCB$L_IRP                       = 00000058
UCB$L_MEDIA                     = 000000BC
UCB$L_OPCNT                     = 00000070
UCB$L_ORB                       = 0000001C
UCB$M_ERLOGIP                   = 00000004
UCB$V_ERLOGIP                   = 00000002
UCB$W_ERRCNT                    = 00000082
UCB$W_FUNC                      = 0000009A
UCB$W_STS                       = 00000064
UCB$W_UNIT                      = 00000054
```

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

PSECT name                  Allocation          PSECT No.  Attributes
----------                  ----------          ---------  ----------
.  ABS  .                   00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                       00000000  (    0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
$$$260                      00000414  ( 1044.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC QUAD
$AEXENONPAGED               000000C7  (  199.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC LONG
WIONONPAGED                 00000370  (  880.)  04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE

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

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                    36    00:00:00.07    00:00:01.74
Command processing               108    00:00:00.49    00:00:04.75
Pass 1                           549    00:00:23.65    00:01:09.35
Symbol table sort                  0    00:00:03.40    00:00:11.63
Pass 2                           174    00:00:04.51    00:00:14.71
Symbol table output               34    00:00:00.28    00:00:01.92
Psect synopsis output              2    00:00:00.02    00:00:00.02
Cross-reference output             0    00:00:00.00    00:00:00.00
Assembler run totals             905    00:00:32.44    00:01:44.15

The working set limit was 1950 pages.
130939 bytes (256 pages) of virtual memory were used to buffer the intermediate code.
There were 120 pages of symbol table space allocated to hold 2271 non-local and 34 local symbols.
901 source lines were read in Pass 1, producing 25 object records in Pass 2.
41 pages of virtual memory were used to define 40 macros.

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

Macro library name                         Macros defined
------------------                         --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                       28
_$255$DUA28:[SYSLIB]STARLET.MLB;2                     7
TOTALS (all libraries)                               35

2304 GETS were required to define 35 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:ERRORLOG/OBJ=OBJ$:ERRORLOG MSRC$:ERRORLOG/UPDATE=(ENH$:ERRORLOG)+EXECML$/LIB
```

0374 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



FILEREAD
LIS

EXCEPTMSG
LIS

DISMOUNT
LIS

DEBUGDATA
LIS

ERRORLOG
LIS

DEVICEDAT
LIS

EXCEPTION
LIS

FILERWIO
LIS

EXSUBROUT
LIS