```
EEEEEEEEEEEEEEE   RRRRRRRRRRR      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
EEEEEEEEEEEEEEE   RRR          RRR   FFF
```

FILEID**SUMMARY

```
  SSSSSSSS  UU      UU  MM      MM  MM      MM    AAAAAA    RRRRRRRR    YY      YY
  SSSSSSSS  UU      UU  MM      MM  MM      MM   AAAAAA     RRRRRRRR    YY      YY
SS          UU      UU  MMMM  MMMM  MMMM  MMMM  AA    AA   RR      RR  YY      YY
SS          UU      UU  MMMM  MMMM  MMMM  MMMM  AA    AA   RR      RR  YY      YY
SS          UU      UU  MM  MM  MM  MM  MM  MM  AA    AA   RR      RR    YY  YY
SS          UU      UU  MM  MM  MM  MM  MM  MM  AA    AA   RR      RR    YY  YY
  SSSSSS    UU      UU  MM      MM  MM      MM  AA    AA   RRRRRRRR        YY
  SSSSSS    UU      UU  MM      MM  MM      MM  AA    AA   RRRRRRRR        YY
        SS  UU      UU  MM      MM  MM      MM  AAAAAAAAAA   RR  RR          YY
        SS  UU      UU  MM      MM  MM      MM  AAAAAAAAAA   RR  RR          YY
        SS  UU      UU  MM      MM  MM      MM  AA    AA   RR    RR        YY
        SS  UU      UU  MM      MM  MM      MM  AA    AA   RR    RR        YY
SSSSSSSS    UUUUUUUUUU  MM      MM  MM      MM  AA    AA   RR      RR      YY
SSSSSSSS    UUUUUUUUUU  MM      MM  MM      MM  AA    AA   RR      RR      YY

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
```

....
....
....
....

```
0001    C
0002    C Version:      'V04-000'
0003    C
0004    C*****************************************************************************
0005    C*                                                                           *
0006    C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0007    C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0008    C*  ALL RIGHTS RESERVED.                                                     *
0009    C*                                                                           *
0010    C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0011    C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0012    C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0013    C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0014    C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0015    C*  TRANSFERRED.                                                             *
0016    C*                                                                           *
0017    C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0018    C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0019    C*  CORPORATION.                                                             *
0020    C*                                                                           *
0021    C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0022    C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0023    C*                                                                           *
0024    C*                                                                           *
0025    C*****************************************************************************
0026    C
0027    C       AUTHOR  BRIAN PORTER            CREATION DATE   30-JAN-1980
0028    C
0029
0030    c++
0031    c       Functional description
0032    c
0033    c       This routine maintains a list of counts of error log entry types
0034    c       for different CPU's.  The list has absolute linkage and has the
0035    c       following format.
0036    c
0037    c       +-------------------------+
0038    c       I          flink          I
0039    c       +-------------------------+
0040    c       I          blink          I
0041    c       +-------------------------+
0042    c       I       logging SID       I
0043    c       +-------------------------+
0044    c       I    64 bit time of       I
0045    c       +--                     --+
0046    c       I    earliest entry       I
0047    c       +-------------------------+
0048    c       I    64 bit time of       I
0049    c       +--                     --+
0050    c       I     latest entry        I
0051    c       +-------------------------+
0052    c       I                         I
0053    c       +--                     --+
0054    c       I                         I
0055    c       +--   128 entry type    --+
0056    c       I       buffers           I
0057    c       +--                     --+
```

```
0058     c        I                          I
0059     c        +--------------------------+
0060     c        I                          I
0061     c        +--                      --+
0062     c        I                          I
0063     c        +--                      --+
0064     c        I                          I
0065     c        +--      48 byte         --+
0066     c        I    histogram buffer      I
0067     c        +--                      --+
0068     c        I                          I
0069     c        +--                      --+
0070     c        I                          I
0071     c        +--------------------------+
0072     c
0073     c        Each entry buffer within the list has the following format
0074     c
0075     c        +--------------------------+
0076     c        I                          I
0077     c        +-- entry title descr. --+
0078     c        I                          I
0079     c        +--------------------------+
0080     c        I complete entry  count  I
0081     c        +--------------------------+
0082     c        I incomplete entry count I
0083     c        +--------------------------+
0084     c
0085     c        As entries are encountered in the error log file a search of the
0086     c        list is made for a list entry with the same logging SID.  When a
0087     c        match is found then the appropriate entry buffer counter is updated.
0088     c        Valid entry types are in the range 1 to 255.  Complete entry types
0089     c        are from 1 to 127, incomplete types are weighted by 128.
0090     c        Entry title descriptors that are zero indicate that this entry
0091     c        type for the particular logging SID are unknown.
0092     c        The last entry buffer is used to count unknown entry types within
0093     c        and outside of the valid ranges.
0094     c        The histogram buffer is used to display graphically the occurrance
0095     c        of processed records in the file for the given logging sid.
0096     c        The buffer is dimensioned internally into twenty-four 16-bit integer
0097     c        counters.  As a record is processed by SYE the hour of the day
0098     c        value in EMB$Q_HD_TIME is used as an index into the array and the
0099     c        counter incremented.
0100     c
0101     c        Modified by:
0102     c
0103     c        V03-005 EAD0002         Elliott A. Drayton      18-Feb-1984
0104     c                Add UVAX-1 suport.
0105     c
0106     C        V03-004 JMG0006         Joel M. Gringorten,     29-Dec-1983
0107     C                - Delete Histogram output from Entry_summary_output
0108     C                - Create routine Processed_entries_histo_output
0109     C
0110     C        V03-003 SAR0163         Sharon A. Reynolds,     13-Oct-1983
0111     C                Added an SYE update that adds 11/7XX and 'logmscp'
0112     C                support.
0113     C
0114     C        V03-002 SAR0097         Sharon A. Reynolds,     20-Jun-1983
```

```
0115     C                Changed the carriage control in the 'format' statements
0116     C                for use with ERF.
0117     C
0118     c        v03-001 BP0014          Brian Porter,          05-APR--1982
0119     c                Added more mscp.
0120     c**
0121     c--
0122
0123
0124
0125             subroutine entry_summary_update (cpu,entry,class,type)
0126
0127
0128
0129             include 'src$:msghdr.for /nolist'
0189
0190
0191
0192
0193             byte            lun
0194
0195             logical*1       cpu
0196
0197             logical*1       entry
0198
0199             logical*1       class
0200
0201             logical*1       type
0202
0203             integer*4       buffer1(2)
0204
0205             integer*4       root_summary_flink
0206
0207             integer*4       root_summary_blink
0208
0209             equivalence     (root_summary_flink,buffer1(1))
0210
0211             equivalence     (root_summary_blink,buffer1(2))
0212
0213             integer*4       buffer2(531)
0214
0215             integer*4       flink
0216
0217             integer*4       blink
0218
0219             integer*4       logging_sid
0220
0221             integer*4       earliest_entry_time(2)
0222
0223             integer*4       latest_entry_time(2)
0224
0225             integer*4       entry_type_buffer(4,128)
0226
0227             integer*2       processed_entry_histogram(0:23)
0228
0229             parameter       entry_title_descriptor_length = 1
```

```
0230
0231            parameter       entry_title_descriptor_addre: = 2
0232
0233            parameter       unknown_entry = 3
0234
0235            parameter       complete_entry = 3
0236
0237            parameter       incomplete_entry = 4
0238
0239            equivalence     (flink,buffer2(1))
0240
0241            equivalence     (blink,buffer2(2))
0242
0243            equivalence     (logging_sid,buffer2(3))
0244
0245            equivalence     (earliest_entry_time,buffer2(4))
0246
0247            equivalence     (latest_entry_time,buffer2(6))
0248
0249            equivalence     (entry_type_buffer,buffer2(8))
0250
0251            equivalence     (processed_entry_histogram,buffer2(520))
0252
0253            integer*4       summary_list_entry_count
0254
0255            integer*4       lib$extzv
0256
0257            integer*4       summary_list_entry_address
0258
0259            logical*1       lib$get_vm
0260
0261            logical*1       sys$numtim
0262
0263            character*24    time1
0264
0265            character*24    time2
0266
0267            logical*1       possible_invalid_time
0268
0269            integer*4       timcmp
0270
0271            integer*2       time_buffer(7)
0272
0273
0274
0275    c++
0276    c       functional description
0277    c
0278    c       This entry point is used to create and update the list entries
0279    c       as the different error log entries for the various logging CPU's
0280    c       making up the error log file are encountered.
0281    c--
0282
0283
0284
0285
0286            if (root_summary_flink .eq. 0
```

```
0287            1 .and.
0288            2 root_summary_blink .eq. 0) then
0289
0290            root_summary_flink = %loc(root_summary_flink)
0291
0292            root_summary_blink = %loc(root_summary_flink)
0293            endif
0294
0295            summary_list_entry_address = root_summary_flink
0296
0297            do 100,i = 1,summary_list_entry_count
0298
0299            call movc3 (%val(2076),%val(summary_list_entry_address),flink)
0300
0301            if (emb$l_hd_sid .eq. logging_sid) then
0302
0303            possible_invalid_time = .false.
0304
0305    c
0306    c       unknown entry type
0307    c
0308
0309    10      if (.not. entry) then
0310
0311            entry_type_buffer(unknown_entry,128) =
0312            1 entry_type_buffer(unknown_entry,128) + 1
0313
0314            possible_invalid_time = .true.
0315
0316    c
0317    c       incomplete entry type for a known cpu type
0318    c
0319
0320            else if (
0321            1 cpu
0322            1 .and.
0323            1 entry
0324            1 .and.
0325            1 emb$w_hd_entry .ge. 128
0326            1 ) then
0327
0328            entry_type_buffer(incomplete_entry,lib$extzv(0,7,emb$w_hd_entry)) =
0329            1 entry_type_buffer(incomplete_entry,lib$extzv(0,7,emb$w_hd_entry)) + 1
0330
0331            possible_invalid_time = .true.
0332
0333    c
0334    c       device entry for a known cpu with an unknown device
0335    c
0336
0337            else if (
0338            1 cpu
0339            1 .and.
0340            1 ((emb$w_hd_entry .eq. 1
0341            1 .or.
0342            1 emb$w_hd_entry .eq. 96
0343            1 .or.
```

```
0344            1 emb$w_hd_entry .eq. 98
0345            1 .or.
0346            1 emb$w_hd_entry .eq. 99
0347            1 .or.
0348            1 emb$w_hd_entry .eq. 100)
0349            1 .and.
0350            1 (.not. class
0351            1 .or.
0352            1 .not. type))) then
0353
0354            entry_type_buffer(unknown_entry,128) =
0355            1 entry_type_buffer(unknown_entry,128) + 1
0356
0357    c
0358    c        unknown cpu type (implies that all entries are unknown)
0359    c
0360
0361            else if (.not. cpu) then
0362
0363            entry_type_buffer(unknown_entry,128) =
0364            1 entry_type_buffer(unknown_entry,128) + 1
0365
0366            possible_invalid_time = .true.
0367
0368    c
0369    c       known entry type for a known cpu
0370    c
0371
0372            else if (
0373            1 cpu
0374            1 .and.
0375            1 entry
0376            1 ) then
0377
0378            entry_type_buffer(complete_entry,emb$w_hd_entry) =
0379            1 entry_type_buffer(complete_entry,emb$w_hd_entry) + 1
0380            endif
0381
0382            if (.not. possible_invalid_time) then
0383
0384            if (timcmp(emb$q_hd_time,earliest_entry_time)) 20,25,25
0385
0386    20      call movq (emb$q_hd_time,earliest_entry_time)
0387
0388    25      if (timcmp(emb$q_hd_time,latest_entry_time)) 35,35,30
0389
0390    30      call movq (emb$q_hd_time,latest_entry_time)
0391            endif
0392
0393    35      call movc3 (%val(2068),logging_sid,
0394            1 %val(summary_list_entry_address + 8))
0395
0396            return
0397            endif
0398
0399            summary_list_entry_address = flink
0400
```

```
0401     100      continue
0402
0403              if (lib$get_vm(((2124+7)/8)*8,summary_list_entry_address)) then
0404
0405              call movc5 (%val(0),,%val(0),%val(2124),buffer2)
0406
0407              logging_sid = emb$l_hd_sid
0408
0409              earliest_entry_time(1) = 'ffffffff'x
0410
0411              earliest_entry_time(2) = '7fffffff'x
0412
0413              latest_entry_time(1) = '0'x
0414
0415              latest_entry_time(2) = '0'x
0416
0417              call entry_summary_descriptor_load (logging_sid,
0418              1 entry_type_buffer(entry_title_descriptor_length,1))
0419
0420              call insque (%val(summary_list_entry_address),%val(root_summary_blink))
0421
0422              summary_list_entry_count = summary_list_entry_count + 1
0423
0424              goto 10
0425              endif
0426
0427              return
0428
0429
0430
0431              entry entry_summary_output (lun)
0432
0433
0434
0435     c++
0436     c        functional description
0437     c
0438     c        This entry point is used when the processing of the error log
0439     c        file is finished and the entry summaries are to be output.
0440     c--
0441
0442
0443
0444
0445              summary_list_entry_address = root_summary_flink
0446
0447              do 200,i = 1,summary_list_entry_count
0448
0449              call movc3 (%val(2124),%val(summary_list_entry_address),flink)
0450
0451              call frctof (lun)
0452
0453              call linchk (lun,3)
0454
0455              write(lun,110) logging_sid
0456     110      format(/' ','SUMMARY OF ALL ENTRIES LOGGED BY SID ',z8.8,/)
0457
```

```
0458            do 190,j = 1,128
0459
0460            if (entry_type_buffer(entry_title_descriptor_address,j) .ne. 0) then
0461
0462            if (entry_type_buffer(complete_entry,j) .ne. 0
0463            1 .or.
0464            2 entry_type_buffer(incomplete_entry,j) .ne. 0) then
0465
0466            call entry_summary_write (lun,
0467            1 entry_type_buffer(entry_title_descriptor_length,j),
0468            2 entry_type_buffer(complete_entry,j),
0469            3 entry_type_buffer(incomplete_entry,j))
0470            endif
0471            endif
0472
0473    190     continue
0474
0475            call linchk (lun,3)
0476
0477            if (earliest_entry_time(1) .ne. 'ffffffff'x
0478            1 .and.
0479            1 earliest_entry_time(2) .ne. '7fffffff'x
0480            1 .and.
0481            1 latest_entry_time(1) .ne. '0'x
0482            1 .and.
0483            1 latest_entry_time(2) .ne. '0'x) then
0484
0485            call sys$asctim(,time1,earliest_entry_time,%val(0))
0486
0487            call sys$asctim(,time2,latest_entry_time,%val(0))
0488
0489            write(lun,120) time1,time2
0490    120     format(/' ',t8,'DATE OF EARLIEST ENTRY',t40,a,/,
0491            1 t8,'DATE OF LATEST ENTRY',t40,a)
0492            endif
0493
0494            summary_list_entry_address = flink
0495
0496    200     continue
0497
0498            return
0499
0500
0501
0502
0503            entry processed_entries_histo_output (lun)
0504
0505
0506    c++
0507    c       functional description
0508    c
0509    c       This entry point is used when the processing of the error log
0510    c       file is finished and the Histogtram summary is to be output.
0511    c--
0512
0513
0514            summary_list_entry_address = root_summary_flink
```

```
0515
0516            do 220,i = 1,summary_list_entry_count
0517
0518            call movc3 (%val(2124),%val(summary_list_entry_address),flink)
0519
0520            call frctof (lun)
0521
0522            call linchk (lun,3)
0523
0524            do 122,j = 0,23
0525
0526            if (processed_entry_histogram(j) .ne. 0) goto 124
0527
0528    122     continue
0529
0530            goto 150
0531
0532    124     call linchk (lun,27)
0533
0534            write(lun,125) logging_sid
0535    125     format(/' ','PROCESSED ENTRIES HOUR-OF-DAY HISTOGRAM LOGGED BY SID ',
0536           1 z8.8,/)
0537
0538            do 145,j = 0,23
0539
0540            k = min(50,processed_entry_histogram(j))
0541
0542            write(lun,130) j,processed_entry_histogram(j),('*',l = 1,k)
0543    130     format(' ',t8,i2.2,':00',i7,'. ',<k>a1)
0544
0545    145     continue
0546
0547    150     summary_list_entry_address = flink
0548
0549    220     continue
0550
0551            return
0552
0553
0554
0555            entry processed_entries_histo_update
0556
0557
0558    c++
0559    c       Functional description:
0560    c
0561    c       This routine counts the entries that are processed
0562    c       during each hour of the day that they are logged.
0563    c--
0564
0565
0566
0567
0568            summary_list_entry_address = root_summary_flink
0569
0570            do 300,i = 1,summary_list_entry_count
0571
```

```
0572            call movc3 (%val(12),%val(summary_list_entry_address),flink)
0573
0574            if (emb$l_hd_sid .eq. logging_sid) then
0575
0576            call movc3 (%val(48),%val(summary_list_entry_address + 2076),
0577            1 processed_entry_histogram)
0578
0579            if (sys$numtim(time_buffer,emb$q_hd_time)) then
0580
0581            processed_entry_histogram(time_buffer(4)) =
0582            1 processed_entry_histogram(time_buffer(4)) + 1
0583
0584            call movc3 (%val(48),processed_entry_histogram,
0585            1 %val(summary_list_entry_address + 2076))
0586            endif
0587
0588            return
0589            endif
0590
0591            summary_list_entry_address = flink
0592
0593    300     continue
0594
0595            return
0596
0597            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	1149	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	223	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	2552	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	4436	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-000001A9		ENTRY_SUMMARY_OUTPUT	0-00000000		ENTRY_SUMMARY_UPDATE
0-000002E5		PROCESSED_ENTRIES_HISTO_OUTPUT	0-000003F8		PROCESSED_ENTRIES_HISTO_UPDATE

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000004	I*4	BLINK	AP-0000000C@	L*1	CLASS
AP-00000004@	L*1	CPU	3-00000000	I*4	EMB$L_HD_SID
3-00000004	I*2	EMB$W_HD_ENTRY	3-0000000E	I*2	EMB$W_HD_ERRSEQ

```
 AP-00000008@ L*1  ENTRY                          2-00000000  I*4  FLINK
   2-0000089C  I*4  I                             2-000008A0  I*4  J
   2-000008A4  I*4  K                             2-000008A8  I*4  L
   2-00000008  I*4  LOGGING_SID                 AP-00000004@ L*1  LUN
   2-00000892  L*1  POSSIBLE_INVALID_TIME         2-00000850  I*4  ROOT_SUMMARY_BLINK
   2-0000084C  I*4  ROOT_SUMMARY_FLINK            2-00000898  I*4  SUMMARY_LIST_ENTRY_ADDRESS
   2-00000894  I*4  SUMMARY_LIST_ENTRY_COUNT      2-00000862  CHAR TIME1
   2-0000087A  CHAR TIME2                       AP-00000010@ L*1  TYPE

ARRAYS

    Address  Type  Name                                  Bytes  Dimensions

  2-0000084C  I*4  BUFFER1                                 8  (2)
  2-00000000  I*4  BUFFER2                              2124  (531)
  2-0000000C  I*4  EARLIEST_ENTRY_TIME                     8  (2)
  3-00000000  L*1  EMB                                   512  (0:511)
  3-00000006  I*4  EMB$Q_HD_TIME                           8  (2)
  2-0000001C  I*4  ENTRY_TYPE_BUFFER                    2048  (4, 128)
  2-00000014  I*4  LATEST_ENTRY_TIME                       8  (2)
  2-0000081C  I*2  PROCESSED_ENTRY_HISTOGRAM              48  (0:23)
  2-00000854  I*2  TIME_BUFFER                            14  (7)

LABELS

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

  0-00000054  10         **        20       0-0000011E  25         **        30       0-00000132  35         **        100
  1-00000015  110'     1-00000045  120'       **        122      0-00000349  124      1-00000083  125'     1-000000C4  130'
     **       145      0-000003E8  150        **        190        **        200        **        220        **        300

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name                                 Type  Name                                 Type  Name

        ENTRY_SUMMARY_DESCRIPTOR_LOAD              ENTRY_SUMMARY_WRITE                        FRCTOF
        INSQUE                               I*4   LIB$EXTZV                           L*1   LIB$GET_VM
        LINCHK                                     MOVC3                                      MOVC5
        MOVQ                                       SYS$ASCTIM                         L*1   SYS$NUMTIM
  I*4   TIMCMP
```

```
0001
0002
0003
0004
0005            subroutine entry_summary_descriptor_load (logging_sid,buffer)
0006
0007
0008
0009
0010    c++
0011    c       functional description
0012    c
0013    c       This routine constructs the entry type title descriptor
0014    c       for valid entry values for each CPU type.  Unknown entry values
0015    c       for the particular CPU type are left zero.
0016    c--
0017
0018
0019
0020
0021            integer*4       logging_sid
0022
0023            integer*4       buffer(2,2,128)
0024
0025            integer*4       sid_high_byte
0026
0027
0028
0029
0030            character*21    device_error
0031            data    device_error            /'DEVICE ERROR BIT SET*'/
0032
0033            character*14    machine_check
0034            data    machine_check           /'MACHINE CHECK*'/
0035
0036            character*10    sbi_fault
0037            data    sbi_fault               /'SBI FAULT*'/
0038
0039            character*10    sbi_alert
0040            data    sbi_alert               /'SBI ALERT*'/
0041
0042            character*13    soft_ecc
0043            data    soft_ecc                /'MEMORY ERROR*'/
0044
0045            character*23    async_write
0046            data    async_write             /'SBI ASYNCHRONOUS WRITE*'/
0047
0048            character*16    write_bus_error
0049            data    write_bus_error         /'WRITE BUS ERROR*'/
0050
0051            character*19    hard_ecc
0052            data    hard_ecc                /'FATAL MEMORY ERROR*'/
0053
0054            character*13    volume_mount
0055            data    volume_mount            /'VOLUME MOUNT*'/
0056
0057            character*16    volume_dismount
```

```
0058            data    volume_dismount         /'VOLUME DISMOUNT*'/
0059
0060            character*16    start_up
0061            data    start_up                /'SYSTEM START-UP*'/
0062
0063            character*11    power_fail
0064            data    power_fail              /'POWER-FAIL*'/
0065
0066            character*19    new_logfile
0067            data    new_logfile             /'ERRLOG.SYS CREATED*'/
0068
0069            character*19    power_start
0070            data    power_start             /'POWER-FAIL RESTART*'/
0071
0072            character*15    fatal_bugcheck
0073            data    fatal_bugcheck          /'FATAL BUGCHECK*'/
0074
0075            character*11    time_stamp
0076            data    time_stamp              /'TIME-STAMP*'/
0077
0078            character*16    ss_message
0079            data    ss_message              /'$SNDERR MESSAGE*'/
0080
0081            character*16    sys_bugcheck
0082            data    sys_bugcheck            /'SYSTEM BUGCHECK*'/
0083
0084            character*17    op_message
0085            data    op_message              /'OPERATOR MESSAGE*'/
0086
0087            character*16    nt_message
0088            data    nt_message              /'NETWORK MESSAGE*'/
0089
0090            character*19    device_timeout
0091            data    device_timeout          /'DEVICE I/O TIMEOUT*'/
0092
0093            character*20    undefined_int
0094            data    undefined_int           /'UNDEFINED INTERRUPT*'/
0095
0096            character*17    device_attention
0097            data    device_attention        /'DEVICE ATTENTION*'/
0098
0099            character*14    user_bugcheck
0100            data    user_bugcheck           /'USER BUGCHECK*'/
0101
0102            character*14    uba_interrupt
0103            data    uba_interrupt           /'UBA INTERRUPT*'/
0104
0105            character*10    uba_error
0106            data    uba_error               /'UBA ERROR*'/
0107
0108            character*14    mba_interrupt
0109            data    mba_interrupt           /'MBA INTERRUPT*'/
0110
0111            character*19    unknown_entry
0112            data    unknown_entry           /'UNKNOWN ENTRY TYPE*'/
0113
0114            character*15    mscp_message
```

```
0115            data    mscp_message            /'ERL$LOGMESSAGE*'/
0116
0117            Character*12    mscp_message2
0118            Data    mscp_message2           /'ERL$LOGMSCP*'/
0119
0120            character*14    mscp_software_status
0121            data    mscp_software_status    /'ERL$LOGSTATUS*'/
0122
0123            Character*22    environmental_monitor
0124            Data    environmental_monitor   /'ENVIRONMENTAL MONITOR*'/
0125
0126            Character*9     cpu_halt
0127            Data    cpu_halt                /'CPU HALT*'/
0128
0129            Character*15    console_reboot
0130            Data    console_reboot          /'CONSOLE REBOOT*'/
0131
0132
0133            sid_high_byte = lib$extzv (24,8,logging_sid)
0134
0135    c
0136    c       11/780
0137    c
0138
0139            if (sid_high_byte .eq. 255
0140            1 .or.
0141            1 sid_high_byte .eq. 1) then
0142
0143            buffer(2,1,1) = %loc(device_error)
0144
0145            buffer(2,1,2) = %loc(machine_check)
0146
0147            buffer(2,1,4) = %loc(sbi_fault)
0148
0149            buffer(2,1,5) = %loc(sbi_alert)
0150
0151            buffer(2,1,6) = %loc(soft_ecc)
0152
0153            buffer(2,1,7) = %loc(async_write)
0154
0155            buffer(2,1,8) = %loc(hard_ecc)
0156
0157            buffer(2,1,9) = %loc(uba_interrupt)
0158
0159            buffer(2,1,12) = %loc(mba_interrupt)
0160
0161            buffer(2,1,32) = %loc(start_up)
0162
0163            buffer(2,1,34) = %loc(power_fail)
0164
0165            buffer(2,1,35) = %loc(new_logfile)
0166
0167            buffer(2,1,36) = %loc(power_start)
0168
0169            buffer(2,1,37) = %loc(fatal_bugcheck)
0170
0171            buffer(2,1,38) = %loc(time_stamp)
```

```
0172
0173              buffer(2,1,39) = %loc(ss_message)
0174
0175              buffer(2,1,40) = %loc(sys_bugcheck)
0176
0177              buffer(2,1,41) = %loc(op_message)
0178
0179              buffer(2,1,42) = %loc(nt_message)
0180
0181              buffer(2,1,64) = %loc(volume_mount)
0182
0183              buffer(2,1,65) = %loc(volume_dismount)
0184
0185              buffer(2,1,96) = %loc(device_timeout)
0186
0187              buffer(2,1,97) = %loc(undefined_int)
0188
0189              buffer(2,1,98) = %loc(device_attention)
0190
0191              buffer(2,1,99) = %loc(mscp_software_status)
0192
0193              buffer(2,1,100) = %loc(mscp_message)
0194
0195              buffer(2,1,101) = %loc(mscp_message2)
0196
0197              buffer(2,1,112) = %loc(user_bugcheck)
0198
0199     c
0200     c        11/750
0201     c
0202
0203              else if (sid_high_byte .eq. 2) then
0204
0205              buffer(2,1,1) = %loc(device_error)
0206
0207              buffer(2,1,2) = %loc(machine_check)
0208
0209              buffer(2,1,6) = %loc(soft_ecc)
0210
0211              buffer(2,1,7) = %loc(write_bus_error)
0212
0213              buffer(2,1,8) = %loc(hard_ecc)
0214
0215              buffer(2,1,32) = %loc(start_up)
0216
0217              buffer(2,1,34) = %loc(power_fail)
0218
0219              buffer(2,1,35) = %loc(new_logfile)
0220
0221              buffer(2,1,36) = %loc(power_start)
0222
0223              buffer(2,1,37) = %loc(fatal_bugcheck)
0224
0225              buffer(2,1,38) = %loc(time_stamp)
0226
0227              buffer(2,1,39) = %loc(ss_message)
0228
```

```
0229            buffer(2,1,40) = %loc(sys_bugcheck)
0230
0231            buffer(2,1,41) = %loc(op_message)
0232
0233            buffer(2,1,42) = %loc(nt_message)
0234
0235            buffer(2,1,64) = %loc(volume_mount)
0236
0237            buffer(2,1,65) = %loc(volume_dismount)
0238
0239            buffer(2,1,96) = %loc(device_timeout)
0240
0241            buffer(2,1,97) = %loc(undefined_int)
0242
0243            buffer(2,1,98) = %loc(device_attention)
0244
0245            buffer(2,1,99) = %loc(mscp_software_status)
0246
0247            buffer(2,1,100) = %loc(mscp_message)
0248
0249            buffer(2,1,101) = %loc(mscp_message2)
0250
0251            buffer(2,1,112) = %loc(user_bugcheck)
0252
0253     c
0254     c      11/730
0255     c
0256
0257            else if (sid_high_byte .eq. 3) then
0258
0259            buffer(2,1,1) = %loc(device_error)
0260
0261            buffer(2,1,2) = %loc(machine_check)
0262
0263            buffer(2,1,6) = %loc(soft_ecc)
0264
0265            buffer(2,1,8) = %loc(hard_ecc)
0266
0267            buffer(2,1,11) = %loc(uba_error)
0268
0269            buffer(2,1,32) = %loc(start_up)
0270
0271            buffer(2,1,34) = %loc(power_fail)
0272
0273            buffer(2,1,35) = %loc(new_logfile)
0274
0275            buffer(2,1,36) = %loc(power_start)
0276
0277            buffer(2,1,37) = %loc(fatal_bugcheck)
0278
0279            buffer(2,1,38) = %loc(time_stamp)
0280
0281            buffer(2,1,39) = %loc(ss_message)
0282
0283            buffer(2,1,40) = %loc(sys_bugcheck)
0284
0285            buffer(2,1,41) = %loc(op_message)
```

```
0286
0287            buffer(2,1,42) = %loc(nt_message)
0288
0289            buffer(2,1,64) = %loc(volume_mount)
0290
0291            buffer(2,1,65) = %loc(volume_dismount)
0292
0293            buffer(2,1,96) = %loc(device_timeout)
0294
0295            buffer(2,1,97) = %loc(undefined_int)
0296
0297            buffer(2,1,98) = %loc(device_attention)
0298
0299            buffer(2,1,99) = %loc(mscp_software_status)
0300
0301            buffer(2,1,100) = %loc(mscp_message)
0302
0303            buffer(2,1,101) = %loc(mscp_message2)
0304
0305            buffer(2,1,112) = %loc(user_bugcheck)
0306
0307    c
0308    c       11/7xx
0309    c
0310
0311            else if (sid_high_byte .eq. 4) then
0312
0313            buffer(2,1,1) = %loc(device_error)
0314
0315            buffer(2,1,2) = %loc(machine_check)
0316
0317            buffer(2,1,4) = %loc(sbi_fault)
0318
0319            buffer(2,1,5) = %loc(sbi_alert)
0320
0321            buffer(2,1,6) = %loc(soft_ecc)
0322
0323            buffer(2,1,7) = %loc(async_write)
0324
0325            buffer(2,1,9) = %loc(uba_interrupt)
0326
0327            buffer(2,1,12) = %loc(mba_interrupt)
0328
0329            buffer(2,1,15) = %loc(environmental_monitor)
0330
0331            buffer(2,1,16) = %loc(cpu_halt)
0332
0333            buffer(2,1,17) = %loc(console_reboot)
0334
0335            buffer(2,1,32) = %loc(start_up)
0336
0337            buffer(2,1,34) = %loc(power_fail)
0338
0339            buffer(2,1,35) = %loc(new_logfile)
0340
0341            buffer(2,1,36) = %loc(power_start)
0342
```

```
0343            buffer(2,1,37) = %loc(fatal_bugcheck)
0344
0345            buffer(2,1,38) = %loc(time_stamp)
0346
0347            buffer(2,1,39) = %loc(ss_message)
0348
0349            buffer(2,1,40) = %loc(sys_bugcheck)
0350
0351            buffer(2,1,41) = %loc(op_message)
0352
0353            buffer(2,1,42) = %loc(nt_message)
0354
0355            buffer(2,1,64) = %loc(volume_mount)
0356
0357            buffer(2,1,65) = %loc(volume_dismount)
0358
0359            buffer(2,1,96) = %loc(device_timeout)
0360
0361            buffer(2,1,97) = %loc(undefined_int)
0362
0363            buffer(2,1,98) = %loc(device_attention)
0364
0365            buffer(2,1,99) = %loc(mscp_software_status)
0366
0367            buffer(2,1,100) = %loc(mscp_message)
0368
0369            buffer(2,1,101) = %loc(mscp_message2)
0370
0371            buffer(2,1,112) = %loc(user_bugcheck)
0372
0373     c
0374     c      UVAX1
0375     c
0376
0377            else if (sid_high_byte .eq. 7) then
0378
0379                    buffer(2,1,1) = %loc(device_error)
0380
0381                    buffer(2,1,2) = %loc(machine_check)
0382
0383                    buffer(2,1,6) = %loc(soft_ecc)
0384
0385                    buffer(2,1,7) = %loc(async_write)
0386
0387                    buffer(2,1,8) = %loc(hard_ecc)
0388
0389                    buffer(2,1,32) = %loc(start_up)
0390
0391                    buffer(2,1,34) = %loc(power_fail)
0392
0393                    buffer(2,1,35) = %loc(new_logfile)
0394
0395                    buffer(2,1,36) = %loc(power_start)
0396
0397                    buffer(2,1,37) = %loc(fatal_bugcheck)
0398
0399                    buffer(2,1,38) = %loc(time_stamp)
```

```
0400
0401                    buffer(2,1,39) = %loc(ss_message)
0402
0403                    buffer(2,1,40) = %loc(sys_bugcheck)
0404
0405                    buffer(2,1,41) = %loc(op_message)
0406
0407                    buffer(2,1,42) = %loc(nt_message)
0408
0409                    buffer(2,1,64) = %loc(volume_mount)
0410
0411                    buffer(2,1,65) = %loc(volume_dismount)
0412
0413                    buffer(2,1,96) = %loc(device_timeout)
0414
0415                    buffer(2,1,98) = %loc(device_attention)
0416
0417                    buffer(2,1,99) = %loc(mscp_software_status)
0418
0419                    buffer(2,1,100) = %loc(mscp_message)
0420
0421                    buffer(2,1,101) = %loc(mscp_message2)
0422
0423                    buffer(2,1,112) = %loc(user_bugcheck)
0424            endif
0425
0426            buffer(2,1,128) = %loc(unknown_entry)
0427
0428            do 10,i = 1,128
0429
0430            buffer(1,1,i) = 33
0431
0432    10      continue
0433
0434            return
0435
0436            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	1016	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	8	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	572	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	1596	

ENTRY POINTS

Address	Type	Name
0-00000000		ENTRY_SUMMARY_DESCRIPTOR_LOAD

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000044	CHAR	ASYNC_WRITE	2-000001FE	CHAR	CONSOLE_REBOOT
2-000001F5	CHAR	CPU_HALT	2-0000015E	CHAR	DEVICE_ATTENTION
2-00000000	CHAR	DEVICE_ERROR	2-00000137	CHAR	DEVICE_TIMEOUT
2-000001DF	CHAR	ENVIRONMENTAL_MONITOR	2-000000DC	CHAR	FATAL_BUGCHECK
2-0000006B	CHAR	HARD_ECC	2-00000214	I*4	I
AP-0000004@	I*4	LOGGING_SID	2-00000015	CHAR	MACHINE_CHECK
2-00000195	CHAR	MBA_INTERRUPT	2-000001B6	CHAR	MSCP_MESSAGE
2-000001C5	CHAR	MSCP_MESSAGE2	2-000001D1	CHAR	MSCP_SOFTWARE_STATUS
2-000000B6	CHAR	NEW_LOGFILE	2-00000127	CHAR	NT_MESSAGE
2-00000116	CHAR	OP_MESSAGE	2-000000AB	CHAR	POWER_FAIL
2-000000C9	CHAR	POWER_START	2-0000002D	CHAR	SBI_ALERT
2-00000023	CHAR	SBI_FAULT	2-00000210	I*4	SID_HIGH_BYTE
2-00000037	CHAR	SOFT_ECC	2-000000F6	CHAR	SS_MESSAGE
2-0000009B	CHAR	START_UP	2-00000106	CHAR	SYS_BUGCHECK
2-000000EB	CHAR	TIME_STAMP	2-0000018B	CHAR	UBA_ERROR
2-0000017D	CHAR	UBA_INTERRUPT	2-0000014A	CHAR	UNDEFINED_INT
2-000001A3	CHAR	UNKNOWN_ENTRY	2-0000016F	CHAR	USER_BUGCHECK
2-0000008B	CHAR	VOLUME_DISMOUNT	2-0000007E	CHAR	VOLUME_MOUNT
2-0000005B	CHAR	WRITE_BUS_ERROR			

ARRAYS

Address	Type	Name	Bytes	Dimensions
AP-00000008@	I*4	BUFFER	2048	(2, 2, 128)

LABELS

Address	Label
**	10

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name
I*4	LIB$EXTZV

```
0001
0002
0003            subroutine entry_summary_write (lun,summary_title,complete_count,
0004            1 incomplete_count)
0005
0006
0007
0008
0009    C++
0010    C
0011    C       This routine writes the summary totals to the listing device.
0012    C
0013    C--
0014
0015
0016
0017            byte            lun
0018
0019            character*(*)   summary_title
0020
0021            integer*4       complete_count
0022
0023            integer*4       incomplete_count
0024
0025            integer*4       compress4
0026
0027            integer*4       compressc
0028
0029            integer*4       field
0030
0031
0032
0033            call linchk (lun,1)
0034
0035            field = compress4 (complete_count)
0036
0037            write(lun,10) summary_title,complete_count
0038    10      format(' ',t8,a<compressc (summary_title)>,t<47 - field>,
0039            1 i<field>,'.')
0040
0041            if (incomplete_count .ne. 0) then
0042
0043            field = compress4 (incomplete_count)
0044
0045            call linchk (lun,1)
0046
0047            write(lun,20) summary_title,incomplete_count
0048    20      format(' ',t8,'INCOMPLETE ',a<compressc (summary_title)>,
0049            1 t<47 - field>,i<field>,'.')
0050            endif
0051
0052            return
0053
0054            end
0055
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	238	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	71	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	44	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	353	

ENTRY POINTS

Address	Type	Name
0-00000000		ENTRY_SUMMARY_WRITE

VARIABLES

Address	Type	Name	Address	Type	Name
AP-0000000C@	I*4	COMPLETE_COUNT	2-00000000	I*4	FIELD
AP-00000010@	I*4	INCOMPLETE_COUNT	AP-00000004@	L*1	LUN
2-00000004@	CHAR	SUMMARY_TITLE			

LABELS

Address	Label	Address	Label
1-00000004	10'	1-0000001F	20'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	COMPRESSC		LINCHK

COMMAND QUALIFIERS

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

/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:	10.55 seconds
Elapsed Time:	26.46 seconds
Page Faults:	224
Dynamic Memory:	220 pages

0154 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

TIMCMP.LIS
TUTAPE.LIS
SBI.LIS
STSEVENT.LIS
TRNS_BITS.LIS
UBAERR.LIS
TIMRB.LIS
UBA.LIS
TU81SENSE.LIS
UNDEFINED.LIS
SYSPWRFL.LIS
SUMMARY.LIS
UNKN_DISP.LIS
SHRVECTOR.LIS
SYSTARTUP.LIS
UBAINT.LIS
TOF.LIS
UNKNOWN.LIS
TSTAPE.LIS