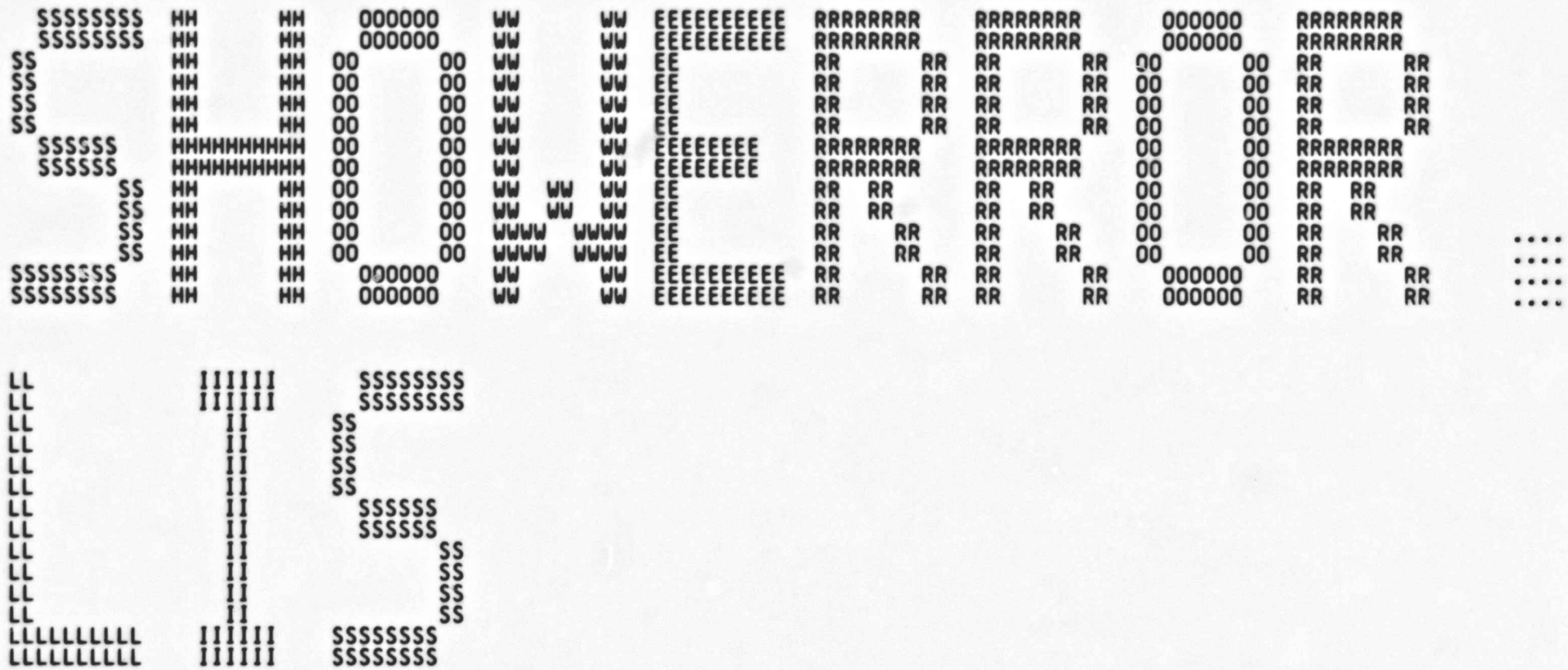
```
   CCCCCCCCCCC  LLL              IIIIIIIII     UUU         UUU  TTTTTTTTTTTTTTT  LLL
   CCCCCCCCCCC  LLL              IIIIIIIII     UUU         UUU  TTTTTTTTTTTTTTT  LLL
   CCCCCCCCCCC  LLL              IIIIIIIII     UUU         UUU  TTTTTTTTTTTTTTT  LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
   CCCCCCCCCCC  LLLLLLLLLLLLLLL     IIIIIIIII     UUUUUUUUUUUUUUU        TTT        LLLLLLLLLLLLLLL
   CCCCCCCCCCC  LLLLLLLLLLLLLLL     IIIIIIIII     UUUUUUUUUUUUUUU        TTT        LLLLLLLLLLLLLLL
   CCCCCCCCCCC  LLLLLLLLLLLLLLL     IIIIIIIII     UUUUUUUUUUUUUUU        TTT        LLLLLLLLLLLLLLL
```

FILEID**SHOWERROR



```
;     1          0001  0 MODULE showerror (IDENT = 'V04-000',
;     2          0002  0                    ADDRESSING_MODE (EXTERNAL = GENERAL)) =
;     3          0003  0
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !
;     7          0007  1 !****************************************************************************
;     8          0008  1 !*                                                                          *
;     9          0009  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;    10          0010  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;    11          0011  1 !*  ALL RIGHTS RESERVED.                                                     *
;    12          0012  1 !*                                                                          *
;    13          0013  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    14          0014  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    15          0015  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    16          0016  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    17          0017  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    18          0018  1 !*  TRANSFERRED.                                                             *
;    19          0019  1 !*                                                                          *
;    20          0020  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    21          0021  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    22          0022  1 !*  CORPORATION.                                                             *
;    23          0023  1 !*                                                                          *
;    24          0024  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    25          0025  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;    26          0026  1 !*                                                                          *
;    27          0027  1 !*                                                                          *
;    28          0028  1 !****************************************************************************
;    29          0029  1 !
;    30          0030  1 !
;    31          0031  1 !++
;    32          0032  1 !
;    33          0033  1 ! FACILITY:  SHOW utility
;    34          0034  1 !
;    35          0035  1 ! ABSTRACT:
;    36          0036  1 !       This module contains the routines for the SHOW ERROR command.
;    37          0037  1 !
;    38          0038  1 ! ENVIRONMENT:
;    39          0039  1 !       VAX native, user and kernel mode
;    40          0040  1 !
;    41          0041  1 ! AUTHOR:  Gerry Smith           CREATION DATE:  28-Jul-1982
;    42          0042  1 !
;    43          0043  1 ! MODIFIED BY:
;    44          0044  1 !
;    45          0045  1 !       V03-002 GAS0149         Gerry Smith             29-Jun-1983
;    46          0046  1 !               Use IOC$CVT_DEVNAM to get the device name.
;    47          0047  1 !
;    48          0048  1 !       V03-001 GAS0117         Gerry Smith             12-Apr-1983
;    49          0049  1 !               Change display to work with long device names.
;    50          0050  1 !
;    51          0051  1 !--
```

```
;    53          0052  1
;    54          0053  1 !
;    55          0054  1 ! Include files
;    56          0055  1 !
;    57          0056  1
;    58          0057  1 LIBRARY 'SYS$LIBRARY:LIB';                  ! VAX/VMS system definitions
;    59          0058  1 REQUIRE 'SRC$:SHOWDEF';                     ! SHOW common definitions
;    60          0157  1
;    61          0158  1 !
;    62          0159  1 ! Define the linkage for the routines to lock and unlock the I/O database,
;    63          0160  1 ! scan the databse, and fabricate the device name.
;    64          0161  1 !
;    65          0162  1 LINKAGE
;    66          0163  1     IOLOCK = JSB (REGISTER = 4),
;    67          0164  1     IOSCAN = JSB (REGISTER = 11,                ! Call with DDB,
;    68          0165  1                   REGISTER = 10;                ! UCB,
;    69          0166  1                   REGISTER = 11,                ! Return with DDB,
;    70          0167  1                   REGISTER = 10),               ! UCB
;    71          0168  1     CVTDEV = JSB (REGISTER = 0,                 ! Length of output buffer,
;    72          0169  1                   REGISTER = 1,                 ! Address of output buffer
;    73          0170  1                   REGISTER = 4,                 ! Format of device name
;    74          0171  1                   REGISTER = 5;                 ! Address of UCB
;    75          0172  1                   REGISTER = 1);                ! Length of final name
;    76          0173  1
;    77          0174  1
;    78          0175  1
;    79          0176  1 !
;    80          0177  1 ! Define the flags longword bits.
;    81          0178  1 !
;    82          0179  1 MACRO
;    83          0180  1     SHOW$V_BRIEF        =       0, 0, 1, 0%,        ! /BRIEF
;    84          0181  1     SHOW$V_FULL         =       0, 1, 1, 0%;        ! /ALL
;    85          0182  1
;    86          0183  1
;    87          0184  1 !
;    88          0185  1 ! Define macros to describe the fields in the scratch area.
;    89          0186  1 !
;    90          0187  1 MACRO
;    91          0188  1     d_l_devlen  =  0, 0, 32, 0%,                ! Device name length
;    92          0189  1     d_a_ptr     =  4, 0, 32, 0%,                ! Pointer to device name
;    93          0190  1     d_l_errcnt  =  8, 0, 32, 0%,                ! Error count
;    94          0191  1     d_t_name    = 12, 0,  8, 0%;                ! Device name
;    95          0192  1 LITERAL d_k_length = $BYTEOFFSET(d_t_name) + 21;
;    96          0193  1
;    97          0194  1 EXTERNAL LITERAL
;    98          0195  1     show$_noerrors;                             ! No device errors found
;    99          0196  1
```

```
;  101          0197  1 !
;  102          0198  1 ! Table of contents
;  103          0199  1 !
;  104          0200  1
;  105          0201  1 FORWARD ROUTINE
;  106          0202  1     show$errors : NOVALUE,
;  107          0203  1     get_data,
;  108          0204  1     print_data : NOVALUE;
;  109          0205  1
;  110          0206  1 EXTERNAL
;  111          0207  1     scs$ga_localsb,
;  112          0208  1     exe$gl_mchkerrs,
;  113          0209  1     exe$gl_memerrs,
;  114          0210  1     ioc$gl_devlist,
;  115          0211  1     sch$gl_curpcb;
;  116          0212  1
;  117          0213  1 EXTERNAL ROUTINE
;  118          0214  1     cli$present,
;  119          0215  1     lib$get_vm,
;  120          0216  1     show$write_line : NOVALUE,
;  121          0217  1     ioc$scan_iodb : IOSCAN,
;  122          0218  1     ioc$cvt_devnam : CVTDEV,
;  123          0219  1     sch$iolockr : IOLOCK,
;  124          0220  1     sch$iounlock : IOLOCK;
```

```
;  126          0221  1 GLOBAL ROUTINE show$errors : NOVALUE =
;  127          0222  2 BEGIN
;  128          0223  2
;  129          0224  2 LOCAL
;  130          0225  2     status,                                 ! General status return
;  131          0226  2     flags : $BBLOCK[4],                     ! Flags longword
;  132          0227  2     arglst : VECTOR[3],                     ! Argument list for $CMKRNL call
;  133          0228  2     data : VECTOR[2];                       ! Address limits of scratch area
;  134          0229  2
;  135          0230  2 !
;  136          0231  2 ! Obtain any qualifiers, if present.
;  137          0232  2 !
;  138          0233  2 flags[show$v_brief] = cli$present(%ASCID 'BRIEF');
;  139          0234  2 flags[show$v_full]  = cli$present(%ASCID 'FULL');
;  140          0235  2
;  141          0236  2 !
;  142          0237  2 ! Allocate a scratch area in which to put data about the errors.
;  143          0238  2 ! The beginning and ending addresses of the area will be returned in DATA.
;  144          0239  2 !
;  145          0240  3 IF NOT (status = lib$get_vm(%REF(64*512),        ! Request 64 pages
;  146          0241  3                           data))               ! Put address limits here
;  147          0242  2 THEN SIGNAL_STOP(show$_insvirmem, 0, .status);  ! Stop if error.
;  148          0243  2 data[1] = .data[0] + 64*512 - 1;                ! Compute end of area
;  149          0244  2 !
;  150          0245  2 ! Call the data-gathering routine in kernel mode, passing the address of the
;  151          0246  2 ! address limits as an argument.  Save the status.
;  152          0247  2 !
;  153          0248  2 arglst[0] = 2;
;  154          0249  2 arglst[1] = data;
;  155          0250  2 arglst[2] = flags;
;  156        P 0251  3 IF NOT (status = $CMKRNL(ROUTIN = get_data,
;  157          0252  3                          ARGLST = arglst))
;  158          0253  2 THEN
;  159          0254  3     BEGIN
;  160          0255  3     IF .status NEQU ss$_vasfull
;  161          0256  3     THEN
;  162          0257  4         BEGIN
;  163          0258  4         SIGNAL(.status);
;  164          0259  4         RETURN;
;  165          0260  3         END;
;  166          0261  2     END;
;  167          0262  2
;  168          0263  2 !
;  169          0264  2 ! Format and print the data.
;  170          0265  2 !
;  171          0266  2 print_data(data, flags);
;  172          0267  2
;  173          0268  2 !
;  174          0269  2 ! If the status return from $CMKRNL is not 1, then signal that error at this
;  175          0270  2 ! time.  Otherwise, simply return.
;  176          0271  2 !
;  177          0272  2 IF .status NEQ 1
;  178          0273  2 THEN SIGNAL(.status);
;  179          0274  2
;  180          0275  2 RETURN;
;  181          0276  1 END;                                          ! End of show$error
```

```
                                                        .TITLE  SHOWERROR
                                                        .IDENT  \V04-000\

                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                00  00  00  46  45  49  52  42  00000 P.AAB:  .ASCII  \BRIEF\<0><0><0>                        ;
                                        010E0005  00008 P.AAA:  .LONG   17694725                             ;
                                        00000000' 0000C         .ADDRESS P.AAB                               ;
                                    4C  4C  55  46  00010 P.AAD:  .ASCII  \FULL\                              ;
                                        010E0004  00014 P.AAC:  .LONG   17694724                             ;
                                        00000000' 00018         .ADDRESS P.AAD                               ;

                                                        .EXTRN  SHOW$_NOERRORS, SCS$GA_LOCALSB
                                                        .EXTRN  EXE$GL_MCHKERRS
                                                        .EXTRN  EXE$GL_MEMERRS, IOC$GL_DEVLIST
                                                        .EXTRN  SCH$GL_CURPCB, CLI$PRESENT
                                                        .EXTRN  LIB$GET_VM, SHOW$WRITE_LINE
                                                        .EXTRN  IOC$SCAN_IODB, IOC$CVT_DEVNAM
                                                        .EXTRN  SCH$IOLOCKR, SCH$IOUNLOCK
                                                        .EXTRN  SYS$CMKRNL

                                                        .PSECT  $CODE$,NOWRT,2

                                        000C 00000      .ENTRY  SHOW$ERRORS, Save R2,R3                      ; 0221
                    53 00000000G    00  9E 00002      MOVAB   CLI$PRESENT, R3                              ;
                    5E              1C  C2 00009      SUBL2   #28, SP                                      ;
                           0000'    CF  9F 0000C      PUSHAB  P.AAA                                        ; 0233
                                63  01  FB 00010      CALLS   #1, CLI$PRESENT                              ;
    04  AE              01          00  50 F0 00013  INSV    R0, #0, #1, FLAGS                            ;
                                    0000' CF  9F 00019  PUSHAB  P.AAC                                       ; 0234
                                63  01  FB 0001D      CALLS   #1, CLI$PRESENT                              ;
    04  AE              01          01  50 F0 00020  INSV    R0, #1, #1, FLAGS                            ;
                                        08  AE  9F 00026  PUSHAB  DATA                                     ; 0240
                                04  AE  8000  8F 3C 00029  MOVZWL  #32768, 4(SP)                           ;
                                        04  AE  9F 0002F  PUSHAB  4(SP)                                    ;
                            00000000G   00  FB 00032      CALLS   #2, LIB$GET_VM                           ;
                                        52  50 D0 00039  MOVL    R0, STATUS                               ;
                                        11  52 E8 0003C  BLBS    STATUS, 1$                               ;
                                            52 DD 0003F  PUSHL   STATUS                                   ; 0242
                                            7E D4 00041  CLRL    -(SP)                                    ;
                                    007812F2 8F DD 00043  PUSHL   #7869170                                  ;
                            00000000G   00  03 FB 00049  CALLS   #3, LIB$STOP                             ;
                0C  AE          08  AE 00007FFF 8F C1 00050 1$:  ADDL3   #32767, DATA, DATA+4             ; 0243
                                10  AE  02 D0 00056  MOVL    #2, ARGLST                                   ; 0248
                                14  AE      08  AE 9E 0005E  MOVAB   DATA, ARGLST+4                       ; 0249
                                18  AE      04  AE 9E 00063  MOVAB   FLAGS, ARGLST+8                      ; 0250
                                            10  AE 9F 00068  PUSHAB  ARGLST                                ; 0252
                                    0000V   CF  9F 0006B      PUSHAB  GET_DATA                              ;
                            00000000G   00  02 FB 0006F      CALLS   #2, SYS$CMKRNL                       ;
                                        52  50 D0 00076  MOVL    R0, STATUS                               ;
                                        09  52 E8 00079  BLBS    STATUS, 2$                               ;
                            00000244    8F      52 D1 0007C  CMPL    STATUS, #580                         ; 0255
                                            10 12 00083  BNEQ    3$                                       ;
                                        04  AE  9F 00085 2$:  PUSHAB  FLAGS                               ; 0266
                                        0C  AE  9F 00088      PUSHAB  DATA                                  ;
                                0000V   CF      02 FB 0008B      CALLS   #2, PRINT_DATA                    ;
```

```
                        01              52  D1 00090          CMPL    STATUS, #1                              ; 0272
                                        09  13 00093          BEQL    4$                                      ;
                                        52  DD 00095 3$:      PUSHL   STATUS                                  ; 0273
                00000000G   00          01  FB 00097          CALLS   #1, LIB$SIGNAL                          ;
                                            04 0009E 4$:      RET                                             ; 0276

; Routine Size:  159 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  183     0277  1 ROUTINE get_data (data, flags) =
;  184     0278  2 BEGIN
;  185     0279  2
;  186     0280  2 LOCAL
;  187     0281  2     status,                                     ! Status return
;  188     0282  2     limit,                                      ! End-of-address limit
;  189     0283  2     scratch : REF $BBLOCK,                      ! Pointer to scratch area
;  190     0284  2     ucb : REF $BBLOCK,                          ! UCB pointer
;  191     0285  2     ddb : REF $BBLOCK;                          ! DDB pointer
;  192     0286  2
;  193     0287  2 MAP
;  194     0288  2     data : REF VECTOR,                          ! The passed argument is two longwords
;  195     0289  2     flags : REF $BBLOCK;                        ! Flags longword
;  196     0290  2 !
;  197     0291  2 ! Set up the scratch area so that is can be addressed easily.  Also, calculate
;  198     0292  2 ! a limit toward the end of the scratch area, so that we don't write beyond the
;  199     0293  2 ! area.  Finally, set up STATUS as 1, to show that we still have room in the
;  200     0294  2 ! scratch area to store more data.
;  201     0295  2 !
;  202     0296  2 scratch = .data[0];                             ! Point to beginning of scratch area
;  203     0297  2 limit = .data[1] - 256;                         ! Set the limit to be halfway in to
;  204     0298  2                                                 ! the last page of the scratch area.
;  205     0299  2
;  206     0300  2 !
;  207     0301  2 ! Lock the I/O data base.  Upon return from the call to SCH$IOLOCKR, the
;  208     0302  2 ! IPL will be 2, so that pagefaults are still allowed.
;  209     0303  2 !
;  210     0304  2 SCH$IOLOCKR(.sch$gl_curpcb);                    ! Lock the I/O database
;  211     0305  2
;  212     0306  2 !
;  213     0307  2 ! For each DDB, examine each UCB associated with it, and see if the error
;  214     0308  2 ! count is non-zero.  If so, or if all devices were requested, then store
;  215     0309  2 ! the pertinent data into the scratch area.  If the scratch area is exhausted,
;  216     0310  2 ! then indicate this by setting STATUS = SS$_VASFULL.
;  217     0311  2 !
;  218     0312  2 status = IOC$SCAN_IODB(0, 0; ddb, ucb);         ! Get to first UCB
;  219     0313  2 WHILE .status DO                                ! Loop thru all UCB's
;  220     0314  3     BEGIN
;  221     0315  3     IF NOT .$BBLOCK[ucb[ucb$l_devchar], dev$v_mbx]      ! Ignore mailboxes
;  222     0316  3     AND NOT .ucb[ucb$v_template]                        ! and template devices
;  223     0317  4     AND (.flags[show$v_full] OR                 ! If all desired, or
;  224     0318  4          .ucb[ucb$w_errcnt] NEQ 0)              ! this one has errors
;  225     0319  3     THEN                                        ! then get information
;  226     0320  4         BEGIN
;  227     0321  4         IF .scratch GEQA .limit
;  228     0322  4         THEN
;  229     0323  5             BEGIN
;  230     0324  5             status = ss$_vasfull;
;  231     0325  5             EXITLOOP
;  232     0326  4             END;
;  233     0327  4         ioc$cvt_devnam(20,                      ! Get device name, max this long
;  234     0328  4                        scratch[d_t_name],       ! put it here,
;  235     0329  4                        -1,                      ! in standard display format
;  236     0330  4                        .ucb;                    ! UCB is here
;  237     0331  4                        scratch[d_l_devlen]);    ! final length here
;  238     0332  4
;  239     0333  4         scratch[d_l_errcnt] = .ucb[ucb$w_errcnt];
```

```
;  240          0334  4          scratch = .scratch + d_k_length;        ! Get to next data block
;  241          0335  3          END;
;  242          0336  3      status = IOC$SCAN_IODB(.ddb, .ucb; ddb, ucb);
;  243          0337  2      END;
;  244          0338  2
;  245          0339  2  !
;  246          0340  2  ! Now to clean up.  Unlock the I/O database, then lower the IPL
;  247          0341  2  ! to zero.
;  248          0342  2  !
;  249          0343  2  SCH$IOUNLOCK(.sch$gl_curpcb);                    ! Unlock I/O database
;  250          0344  2  SET_IPL(0);                                       ! Lower IPL
;  251          0345  2
;  252          0346  2  !
;  253          0347  2  ! If status = 0, then the entire database has been scanned.
;  254          0348  2  !
;  255          0349  2  IF .status EQL 0
;  256          0350  2  THEN status = 1;
;  257          0351  2
;  258          0352  2  RETURN .status;                                   ! Return with status
;  259          0353  1  END;                                             ! End of GET_DATA


                                    0FFC 00000 GET_DATA:
                                                 .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 0277
                          57 00000000G   00  9E 00002    MOVAB   SCH$GL_CURPCB, R7                     ;
                          50       04    AC  D0 00009    MOVL    DATA, R0                              ; 0296
                          52             60  D0 0000D    MOVL    (R0), SCRATCH                         ;
                    56    04    A0 00000100   8F  C3 00010    SUBL3   #256, 4(R0), LIMIT              ; 0297
                          54             67  D0 00019    MOVL    SCH$GL_CURPCB, R4                     ; 0304
                             00000000G   00  16 0001C    JSB     SCH$IOLOCKR                           ;
                                         5A  7C 00022    CLRQ    R10                                   ; 0312
                             00000000G   00  16 00024 1$: JSB     IOC$SCAN_IODB                         ;
                          53             50  D0 0002A    MOVL    R0, STATUS                            ;
                          42             53  E9 0002D    BLBC    STATUS, 4$                            ; 0313
                    EF    3A    AA       04  E0 00030    BBS     #4, 58(UCB), 1$                       ; 0315
                    EA    65    AA       05  E0 00035    BBS     #5, 101(UCB), 1$                      ; 0316
                    06    08    BC       01  E0 0003A    BBS     #1, @FLAGS, 2$                        ; 0317
                                0082   CA  B5 0003F    TSTW    130(UCB)                              ; 0318
                                       DF  13 00043    BEQL    1$                                    ;
                          56           52  D1 00045 2$: CMPL    SCRATCH, LIMIT                        ; 0321
                                       07  1F 00048    BLSSU   3$                                    ;
                          53       0244  8F  3C 0004A    MOVZWL  #580, STATUS                          ; 0324
                                       21  11 0004F    BRB     4$                                    ; 0323
                          51         0C  A2  9E 00051 3$: MOVAB   12(SCRATCH), R1                     ; 0328
                          55             5A  D0 00055    MOVL    UCB, R5                               ; 0331
                          54             01  CE 00058    MNEGL   #1, R4                                ;
                          50             14  D0 0005B    MOVL    #20, R0                               ;
                             00000000G   00  16 0005E    JSB     IOC$CVT_DEVNAM                        ;
                          82             51  D0 00064    MOVL    R1, (SCRATCH)+                        ;
                    04    A2       0082   CA  3C 00067    MOVZWL  130(UCB), 4(SCRATCH)                  ; 0333
                          52             1D  C0 0006D    ADDL2   #29, SCRATCH                          ; 0334
                                         B2  11 00070    BRB     1$                                    ; 0336
                          54             67  D0 00072 4$: MOVL    SCH$GL_CURPCB, R4                     ; 0343
                             00000000G   00  16 00075    JSB     SCH$IOUNLOCK                          ;
```

```
                    12          00  DA 0007B       MTPR    #0, #18                 ; 0344
                                53  D5 0007E       TSTL    STATUS                  ; 0349
                                03  12 00080       BNEQ    5$                      ;
                    53          01  D0 00082       MOVL    #1, STATUS              ; 0350
                    50          53  D0 00085 5$:   MOVL    STATUS, R0              ; 0352
                                    04 00088       RET                             ; 0353

; Routine Size:  137 bytes,    Routine Base:  $CODE$ + 009F
```

```
;  261     0354  1 ROUTINE print_data (data, flags) : NOVALUE =
;  262     0355  2 BEGIN
;  263     0356  2
;  264     0357  2 MAP
;  265     0358  2     data : REF VECTOR,
;  266     0359  2     flags : REF $BBLOCK;
;  267     0360  2
;  268     0361  2 LOCAL
;  269     0362  2     scratch : REF $BBLOCK;                      ! Pointer to scratch area
;  270     0363  2
;  271     0364  2 !
;  272     0365  2 ! Set up the scratch area.
;  273     0366  2 !
;  274     0367  2 scratch = .data[0];                             ! Scratch area begins here.
;  275     0368  2
;  276     0369  2 !
;  277     0370  2 ! Check to see if there are any errors.  If not, go away.  Otherwise,
;  278     0371  2 ! print a nice little heading.
;  279     0372  2 !
;  280     0373  2 IF NOT .flags[show$v_full]                      ! If not a /FULL request
;  281     0374  2 AND .scratch[d_t_name] EQL 0                    ! and no device errors
;  282     0375  2 AND .exe$gl_mchkerrs EQL 0                      ! and no CPU errors
;  283     0376  2 AND .exe$gl_memerrs EQL 0                       ! and no memory errors
;  284     0377  2 THEN
;  285     0378  3     BEGIN
;  286     0379  3     SIGNAL(show$_noerrors);
;  287     0380  3     RETURN;
;  288     0381  2     END;
;  289     0382  2
;  290     0383  2
;  291     0384  2 show$write_line(%ASCID 'Device                Error Count', 0);
;  292     0385  2
;  293     0386  2 !
;  294     0387  2 ! Print CPU or memory errors, if there are any.
;  295     0388  2 !
;  296     0389  2 IF .exe$gl_mchkerrs NEQ 0
;  297     0390  2 OR .flags[show$v_full]
;  298     0391  2 THEN show$write_line(%ASCID '!21<CPU!>   !6UW', %REF(.exe$gl_mchkerrs));
;  299     0392  2
;  300     0393  2 IF .exe$gl_memerrs NEQ 0
;  301     0394  2 OR .flags[show$v_full]
;  302     0395  2 THEN show$write_line(%ASCID '!21<MEMORY!>   !6UW', %REF(.exe$gl_memerrs));
;  303     0396  2
;  304     0397  2 !
;  305     0398  2 ! Loop thru the scratch area, taking the data off in the same way it was
;  306     0399  2 ! stored.  This will continue until a segment is found which contains a
;  307     0400  2 ! device name length of 0.
;  308     0401  2 !
;  309     0402  2 WHILE .scratch[d_l_devlen] NEQ 0
;  310     0403  2 DO
;  311     0404  3     BEGIN
;  312     0405  3     scratch[d_l_devlen] = .scratch[d_l_devlen] - 1;
;  313     0406  3     scratch[d_a_ptr] = scratch[d_t_name] + 1;
;  314     0407  3     show$write_line(%ASCID '!21<!AF!>   !6UW',
;  315     0408  3                     .scratch);
;  316     0409  3     scratch = .scratch + d_k_length;
;  317     0410  2     END;
```

```
;  318          0411  2
;  319          0412  2 RETURN;
;  320          0413  1 END;

                                                                    ! End of PRINT_DATA


                                                                    .PSECT  $PLIT$,NOWRT,NOEXE,2

20 20 20 20 20 20 20 20 20 65 63 69 76 65 44  0001C P.AAF:  .ASCII  \Device                Error Count\
75 6F 43 20 72 6F 72 72 45 20 20 20 20 20 20  0002B
                                        74 6E 0003A
                                  010E0020.   0003C P.AAE:  .LONG   17694752
                                  00000000'   00040         .ADDRESS P.AAF
55 36 21 20 20 20 3E 21 55 50 43 3C 31 32 21  00044 P.AAH:  .ASCII  \!21<CPU!>   !6UW\
                                              57  00053
                                  010E0010.   00054 P.AAG:  .LONG   17694736
                                  00000000'   00058         .ADDRESS P.AAH
20 20 20 3E 21 59 52 4F 4D 45 4D 3C 31 32 21  0005C P.AAJ:  .ASCII  \!21<MEMORY!>   !6UW\<0>
                                  00 57 55 36 21  0006B
                                  010E0013.   00070 P.AAI:  .LONG   17694739
                                  00000000'   00074         .ADDRESS P.AAJ
55 36 21 20 20 20 3E 21 46 41 21 3C 31 32 21  00078 P.AAL:  .ASCII  \!21<!AF!>   !6UW\
                                              57  00087
                                  010E0010.   00088 P.AAK:  .LONG   17694736
                                  00000000'   0008C         .ADDRESS P.AAL


                                                                    .PSECT  $CODE$,NOWRT,2

                                  003C 00000 PRINT_DATA:
                                                    .WORD   Save R2,R3,R4,R5                    ; 0354
                        55 00000000G   00  9E 00002         MOVAB   EXE$GL_MCHKERRS, R5
                        54 00000000G   00  9E 00009         MOVAB   EXE$GL_MEMERRS, R4
                        53 00000000G   00  9E 00010         MOVAB   SHOW$WRITE_LINE, R3
                        5E             04  C2 00017         SUBL2   #4, SP
                        52       04    BC  D0 0001A         MOVL    @DATA, SCRATCH             ; 0367
                1B      08     BC      01  E0 0001E         BBS     #1, @FLAGS, 1$             ; 0373
                                0C      A2  95 00023         TSTB    12(SCRATCH)                ; 0374
                                        16  12 00026         BNEQ    1$
                                        65  D5 00028         TSTL    EXE$GL_MCHKERRS            ; 0375
                                        12  12 0002A         BNEQ    1$
                                        64  D5 0002C         TSTL    EXE$GL_MEMERRS             ; 0376
                                        0E  12 0002E         BNEQ    1$
                           00000000G    8F  DD 00030         PUSHL   #SHOW$_NOERRORS            ; 0379
                00000000G  00           01  FB 00036         CALLS   #1, LIB$SIGNAL
                                            04 0003D         RET                                ; 0378
                                        7E  D4 0003E 1$:     CLRL    -(SP)                      ; 0384
                                0000'   CF  9F 00040         PUSHAB  P.AAE
                        63              02  FB 00044         CALLS   #2, SHOW$WRITE_LINE
                        50              65  D0 00047         MOVL    EXE$GL_MCHKERRS, R0        ; 0389
                                        05  12 0004A         BNEQ    2$
                0C      08     BC       01  E1 0004C         BBC     #1, @FLAGS, 3$             ; 0390
                        6E              50  D0 00051 2$:     MOVL    R0, (SP)                   ; 0391
                                        5E  DD 00054         PUSHL   SP
                                0000'   CF  9F 00056         PUSHAB  P.AAG
                        63              02  FB 0005A         CALLS   #2, SHOW$WRITE_LINE
```

```
                                50              64  D0 0005D 3$:    MOVL    EXE$GL_MEMERRS, R0                      ; 0393
                                                05  12 00060       BNEQ    4$                                      ;
                        0C      08      BC      01  E1 00062       BBC     #1, @FLAGS, 5$                          ; 0394
                                        6E      50  D0 00067 4$:    MOVL    R0, (SP)                                ; 0395
                                                5E  DD 0006A       PUSHL   SP                                      ;
                                        0000'   CF  9F 0006C       PUSHAB  P.AAI                                   ;
                                        63      02  FB 00070       CALLS   #2, SHOW$WRITE_LINE                     ;
                                                62  D5 00073 5$:    TSTL    (SCRATCH)                               ; 0402
                                                15  13 00075       BEQL    6$                                      ;
                                                62  D7 00077       DECL    (SCRATCH)                               ; 0405
                                04      A2      0D  A2 9E 00079     MOVAB   13(R2), 4(SCRATCH)                      ; 0406
                                                52  DD 0007E       PUSHL   SCRATCH                                 ; 0408
                                        0000'   CF  9F 00080       PUSHAB  P.AAK                                   ; 0407
                                        63      02  FB 00084       CALLS   #2, SHOW$WRITE_LINE                     ;
                                        52      21  C0 00087       ADDL2   #33, SCRATCH                            ; 0409
                                                E7  11 0008A       BRB     5$                                      ; 0402
                                                    04 0008C 6$:    RET                                             ; 0413

; Routine Size:  141 bytes,    Routine Base:  $CODE$ + 0128
```

```
;  322          0414  1 END
;  323          0415  0 ELUDOM


                                                    .EXTRN  LIB$SIGNAL, LIB$STOP

;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  $PLIT$                        144  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                        437  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                           Library Statistics
;                                       --------  Symbols  --------      Pages       Processing
;       File                            Total     Loaded   Percent       Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1        18619         20         0         1000        00:01.8



;                                       COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:SHOWERROR/OBJ=OBJ$:SHOWERROR MSRC$:SHOWERROR/UPDATE=(ENH$:SHOWERROR)

; Size:          437 code + 144 data bytes
; Run Time:        00:10.5
; Elapsed Time:    00:29.9
; Lines/CPU Min:    2366
; Lexemes/CPU-Min: 16106
; Memory Used:  95 pages
; Compilation Complete
```

SHOMSGUTL LIS

SHONET LIS

SHOWAUDIT LIS

SHOWID LIS

SHOWLOG LIS

SHOWERROR LIS

SHOWFILES LIS

SHOMEMORY LIS