I 3

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

```
  SSSSSSSS  HH      HH    000000    WW      WW    SSSSSSSS  YY      YY    SSSSSSSS
  SSSSSSSS  HH      HH    000000    WW      WW    SSSSSSSS  YY      YY    SSSSSSSS
SS          HH      HH  00      00  WW      WW  SS          YY      YY  SS
SS          HH      HH  00      00  WW      WW  SS          YY      YY  SS
SS          HH      HH  00      00  WW      WW  SS            YY  YY    SS
SS          HH      HH  00      00  WW      WW  SS            YY  YY    SS
  SSSSSS    HHHHHHHHHH  00      00  WW      WW    SSSSSS        YY        SSSSSS
  SSSSSS    HHHHHHHHHH  00      00  WW      WW    SSSSSS        YY        SSSSSS
        SS  HH      HH  00      00  WW  WW  WW          SS      YY              SS
        SS  HH      HH  00      00  WW  WW  WW          SS      YY              SS
        SS  HH      HH  00      00  WWWW  WWWW          SS      YY              SS      ....
        SS  HH      HH  00      00  WWWW  WWWW          SS      YY              SS      ....
SSSSSSSS    HH      HH    000000    WW      WW  SSSSSSSS        YY      SSSSSSSS        ....
SSSSSSSS    HH      HH    000000    WW      WW  SSSSSSSS        YY      SSSSSSSS        ....

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
```

```
;     1          0001  0 MODULE showsystem (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 SYSTEM 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:  30-Jul-1982
;    42          0042  1 !
;    43          0043  1 ! MODIFIED BY:
;    44          0044  1 !
;    45          0045  1 !       V03-007 AEW0002         Anne E. Warner          27-Feb-1984
;    46          0046  1 !               Reorganize 'Ph.Mem' format to handle increased process
;    47          0047  1 !               working set sizes.
;    48          0048  1 !
;    49          0049  1 !       V03-006 AEW0001         Anne E. Warner          02-Feb-1984
;    50          0050  1 !               Reorganize the SHOW SYSTEM display.
;    51          0051  1 !                  - Make the display fit on an 80 character display by
;    52          0052  1 !                    taking out the UIC.
;    53          0053  1 !                  - Add the qualifier /FULL to display all information
;    54          0054  1 !                    plus add a second line with the UIC.
;    55          0055  1 !                  - Add the system node name to the header.
;    56          0056  1 !                  - Add the number of days to each process CPU time.
;    57          0057  1 !                  - Add buffered I/O to the direct I/O for each process.
```

```
;    58          0058  1 !
;    59          0059  1 !       V03-005 LMP0140         L. Mark Pilant          23-Aug-1983  23:29
;    60          0060  1 !               Add support for alphanumeric UICs.
;    61          0061  1 !
;    62          0062  1 !       V03-004 GAS0117         Gerry Smith             12-Apr-1983
;    63          0063  1 !               Instead of displaying MWAIT, display which resource
;    64          0064  1 !               the process is awaiting, or MUTEX if waiting for that.
;    65          0065  1 !
;    66          0066  1 !       V03-003 CWH1002         CW Hobbs                25-Feb-1983
;    67          0067  1 !               Use extended pids for the process ids.
;    68          0068  1 !
;    69          0069  1 !       V03-002 GAS0107                 Gerry Smith      8-Feb-1983
;    70          0070  1 !               Collect all the qualifiers before making checks on
;    71          0071  1 !               whether or not any were set.
;    72          0072  1 !
;    73          0073  1 !       V03-001 GAS00103                        17-Jan-1983
;    74          0074  1 !               Initialize the PCB before going into the PIX loop.
;    75          0075  1 !
;    76          0076  1 !--
```

```
;    78          0077  1
;    79          0078  1 !
;    80          0079  1 ! Include files
;    81          0080  1 !
;    82          0081  1
;    83          0082  1 LIBRARY 'SYS$LIBRARY:LIB';              ! VAX/VMS system definitions
;    84          0083  1 REQUIRE 'SRC$:SHOWDEF';                 ! SHOW common definitions
;    85          0182  1
;    86          0183  1 ! Define macro to make a string descriptor:  sd_A
;    87          0184  1
;    88          0185  1 MACRO
;    89        M 0186  1     SD[A] =
;    90          0187  1             BIND %NAME('SD_',A) = $DESCRIPTOR(A)%;
;    91          0188  1
;    92          0189  1 !
;    93          0190  1 ! Define the flags for SHOW SYSTEM
;    94          0191  1 !
;    95          0192  1 MACRO
;    96          0193  1     sys$v_proc  = 0, 0, 1, 0%,
;    97          0194  1     sys$v_subp  = 0, 1, 1, 0%,
;    98          0195  1     sys$v_net   = 0, 2, 1, 0%,
;    99          0196  1     sys$v_batch = 0, 3, 1, 0%,
;   100          0197  1     sys$v_full  = 0, 4, 1, 0%;
;   101          0198  1
;   102          0199  1 !
;   103          0200  1 ! Macros to define the layout of the data block to used to
;   104          0201  1 ! hold information on a particular process
;   105          0202  1 !
;   106          0203  1 MACRO
;   107          0204  1     d$l_pid     =  0, 0, 32, 0%,            ! Process ID
;   108          0205  1     d$l_owner   =  4, 0, 32, 0%,            ! Temp owner storage/process name length
;   109          0206  1     d$a_name    =  8, 0, 32, 0%,            ! Pointer to process name string
;   110          0207  1     d$l_state   = 12, 0, 32, 0%,            ! Process state
;   111          0208  1     d$l_pri     = 16, 0, 32, 0%,            ! Current priority
;   112          0209  1     d$l_iocnt   = 20, 0, 32, 0%,            ! Direct plus Buffered I/O count
;   113          0210  1     d$l_cputim  = 24, 0, 32, 0%,            ! CPU time
;   114          0211  1     d$l_pflts   = 28, 0, 32, 0%,            ! Page fault count
;   115          0212  1     d$l_pgcnt   = 32, 0, 32, 0%,            ! Global page count
;   116          0213  1     d$l_sts     = 36, 0, 32, 0%,            ! Status
;   117          0214  1     d$l_uic     = 40, 0, 32, 0%,            ! Process UIC
;   118          0215  1     d$l_lef     = 44, 0, 32, 0%,            ! Local event flag
;   119          0216  1     d$t_name    = 48, 0,  8, 0%;            ! Process name
;   120          0217  1
;   121          0218  1 !
;   122          0219  1 ! The following literal depends on D$T_NAME being the last field in the
;   123          0220  1 ! locked data area.
;   124          0221  1 !
;   125          0222  1 LITERAL d$k_length = $BYTEOFFSET(d$t_name) + pcb$s_lname;
;   126          0223  1
;   127          0224  1 !
;   128          0225  1 ! Define two bits in the data area, D$V_NETWRK and D$V_BATCH, which
;   129          0226  1 ! correspond to PCB$V_NETWORK and PCB$V_BATCH, except that they reference
;   130          0227  1 ! the process status from the PCB status longword, rather than from the
;   131          0228  1 ! beginning of the PCB.
;   132          0229  1 !
;   133          0230  1 MACRO
;   134          0231  1     d$v_netwrk = 0, $BITPOSITION(pcb$v_netwrk), $FIELDWIDTH(pcb$v_netwrk), $EXTENSION(pcb$v_netwrk)%,
```

```
;  135          0232  1     d$v_ba*ch  = 0, $BITPOSITION(pcb$v_batch),  $FIELDWIDTH(pcb$v_batch),  $EXTENSION(pcb$v_batch)%;
```

```
;  137     0233  1 !
;  138     0234  1 ! Construct a table of process states.  THIS IS AN ORDERED TABLE.
;  139     0235  1 !
;  140     0236  1 LITERAL rsn_cnt = 14;
;  141     0237  1 OWN
;  142     0238  1     state_table : VECTOR[14]
;  143     0239  1     INITIAL (cstring('COLPG'),
;  144     0240  1              cstring('MUTEX'),
;  145     0241  1              cstring('CEF'),
;  146     0242  1              cstring('PFW'),
;  147     0243  1              cstring('LEF'),
;  148     0244  1              cstring('LEFO'),
;  149     0245  1              cstring('HIB'),
;  150     0246  1              cstring('HIBO'),
;  151     0247  1              cstring('SUSP'),
;  152     0248  1              cstring('SUSPO'),
;  153     0249  1              cstring('FPG'),
;  154     0250  1              cstring('COM'),
;  155     0251  1              cstring('COMO'),
;  156     0252  1              cstring('CUR')),
;  157     0253  1     rsn_table : VECTOR[rsn_cnt]
;  158     0254  1     INITIAL (cstring('RWAST'),                  ! AST wait
;  159     0255  1              cstring('RWMBX'),                  ! Mailbox full
;  160     0256  1              cstring('RWNPG'),                  ! Non-paged pool
;  161     0257  1              cstring('RWPFF'),                  ! Page file full
;  162     0258  1              cstring('RWPAG'),                  ! Paged pool
;  163     0259  1              cstring('RWBRK'),                  ! Waiting for BROADCAST to finish
;  164     0260  1              cstring('RWIMG'),                  ! Image activation lock
;  165     0261  1              cstring('RWQUO'),                  ! Pooled quota
;  166     0262  1              cstring('RWLCK'),                  ! Lock ID data base
;  167     0263  1              cstring('RWSWP'),                  ! Swap file space
;  168     0264  1              cstring('RWMPE'),                  ! Modified page list empty
;  169     0265  1              cstring('RWMPB'),                  ! Modified page writer busy
;  170     0266  1              cstring('RWSCS'),                  ! SCS wait
;  171     0267  1              cstring('RWCLU'));                 ! Cluster translation wait
;  172     0268  1
;  173     0269  1 !
;  174     0270  1 ! This ASSUME macro makes sure that we have all the miscellaneous resource
;  175     0271  1 ! waits in this table.
;  176     0272  1 !
;  177     0273  1 $ASSUME(rsn_cnt, EQL, rsn$_max-1)
;  178     0274  1
```

```
;  180          0275  1 !
;  181          0276  1 ! Table of contents
;  182          0277  1 !
;  183          0278  1
;  184          0279  1 FORWARD ROUTINE
;  185          0280  1     show$system : NOVALUE,
;  186          0281  1     get_data,
;  187          0282  1     print_data : NOVALUE;
;  188          0283  1
;  189          0284  1 FORWARD
;  190          0285  1     lock_start : VECTOR[0],
;  191          0286  1     lock_end : VECTOR[0];
;  192          0287  1
;  193          0288  1 EXTERNAL
;  194          0289  1     scs$ga_localsb,                    ! Local system block
;  195          0290  1     sys$gq_version,                    ! System version
;  196          0291  1     exe$gl_abstim,                     ! Time system has been up
;  197          0292  1     sch$gl_pcbvec : REF VECTOR,        ! PCB vector
;  198          0293  1     sch$gl_maxpix;                     ! Maximum process index count
;  199          0294  1
;  200          0295  1 EXTERNAL ROUTINE
;  201          0296  1     lib$get_vm,
;  202          0297  1     cli$present,
;  203          0298  1     show$write_line : NOVALUE;
```

```
;  205          0299  1 GLOBAL ROUTINE show$system : NOVALUE =
;  206          0300  2 BEGIN
;  207          0301  2
;  208          0302  2 !---
;  209          0303  2 !
;  210          0304  2 ! This is the driver routine for the SHOW SYSTEM function.  The command
;  211          0305  2 ! qualifiers are gathered, scratch space is allocated, the data-gathering
;  212          0306  2 ! routine is called via $CMKRNL, and then the data is printed.
;  213          0307  2 !
;  214          0308  2 !---
;  215          0309  2
;  216          0310  2 LOCAL
;  217          0311  2     status,                             ! General status return
;  218          0312  2     size,                               ! Size of scratch area
;  219          0313  2     flags : $BBLOCK[2],                 ! Flags byte
;  220          0314  2     desc : VECTOR[2],                   ! Argument list for calls
;  221          0315  2     data : VECTOR[2];                   ! Address limits of scratch area
;  222          0316  2
;  223          0317  2 !
;  224          0318  2 ! Collect qualifiers.
;  225          0319  2 !
;  226          0320  2 flags[sys$v_full]  = cli$present(%ASCID 'FULL');
;  227          0321  2 flags[sys$v_subp]  = cli$present(%ASCID 'SUBPROCESS');
;  228          0322  2 flags[sys$v_net]   = cli$present(%ASCID 'NETWORK');
;  229          0323  2 flags[sys$v_batch] = cli$present(%ASCID 'BATCH');
;  230          0324  3 IF NOT (.flags[sys$v_subp] OR
;  231          0325  3         .flags[sys$v_net] OR
;  232          0326  3         .flags[sys$v_batch])
;  233          0327  2 THEN flags[sys$v_proc] = true;
;  234          0328  2
;  235          0329  2 !
;  236          0330  2 ! Allocate a scratch area in which to put data about the processes.
;  237          0331  2 ! The size of the scratch area is determined by taking the amount of
;  238          0332  2 ! bytes of information per process (D$K_LENGTH), multiplying that by
;  239          0333  2 ! the maximum number of processes in the system, and then adding a few
;  240          0334  2 ! pages for slop.  The beginning and ending addresses of the area will
;  241          0335  2 ! be returned in DATA.
;  242          0336  2 !
;  243          0337  2 size = (.sch$gl_maxpix * d$k_length) + (3 * 512);
;  244          0338  2
;  245          0339  3 IF NOT (status = LIB$GET_VM(size,                ! This many bytes
;  246          0340  3                             data))               ! Put starting address here
;  247          0341  2 THEN SIGNAL_STOP(show$_insvirmem, 0, .status);  ! Stop if error
;  248          0342  2 data[1] = .data[0] + .size - 1;                 ! Put ending address here
;  249          0343  2
;  250          0344  2 !
;  251          0345  2 ! Lock the first page of the scratch area, and the code that runs at elevated
;  252          0346  2 ! IPL, into the process working set.
;  253          0347  2 !
;  254          0348  2 desc[0] = lock_start;
;  255          0349  2 desc[1] = lock_end;
;  256          0350  3 IF NOT (status = $LKWSET(INADR = desc))
;  257          0351  2 THEN SIGNAL_STOP(.status);
;  258          0352  2
;  259          0353  2 desc[0] = .data[0];
;  260          0354  2 desc[1] = .data[0] + d$k_length;
;  261          0355  3 IF NOT (status = $LKWSET(INADR = desc))
```

```
;  262          0356  2 THEN SIGNAL_STOP(.status);
;  263          0357  2
;  264          0358  2 !
;  265          0359  2 ! Call the data-gathering routine in kernel mode, passing the address
;  266          0360  2 ! limits as an argument.
;  267          0361  2 !
;  268          0362  2 desc[0] = 2;
;  269          0363  2 desc[1] = data;
;  270          0364  2 desc[2] = flags;
;  271        P 0365  2 IF NOT (status = $CMKRNL(ROUTIN = get_data,
;  272          0366  3                          ARGLST = desc))
;  273          0367  2 THEN
;  274          0368  3     BEGIN
;  275          0369  3     SIGNAL(.status);
;  276          0370  3     RETURN;
;  277          0371  2     END;
;  278          0372  2
;  279          0373  2 !
;  280          0374  2 ! Format and print the data.
;  281          0375  2 !
;  282          0376  2 print_data(data, flags);
;  283          0377  2
;  284          0378  2 RETURN;
;  285          0379  1 END;                                    ! End of show$error


                                                .TITLE  SHOWSYSTEM
                                                .IDENT  \V04-000\

                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

47 50 4C 4F 43 05  00000 P.AAA:  .ASCII  <5>\COLPG\
58 45 54 55 4D 05  00006 P.AAB:  .ASCII  <5>\MUTEX\
      46 45 43 03  0000C P.AAC:  .ASCII  <3>\CEF\
      57 46 50 03  00010 P.AAD:  .ASCII  <3>\PFW\
      46 45 4C 03  00014 P.AAE:  .ASCII  <3>\LEF\
   4F 46 45 4C 04  00018 P.AAF:  .ASCII  <4>\LEFO\
      42 49 48 03  0001D P.AAG:  .ASCII  <3>\HIB\
   4F 42 49 48 04  00021 P.AAH:  .ASCII  <4>\HIBO\
   50 53 55 53 04  00026 P.AAI:  .ASCII  <4>\SUSP\
4F 50 53 55 53 05  0002B P.AAJ:  .ASCII  <5>\SUSPO\
      47 50 46 03  00031 P.AAK:  .ASCII  <3>\FPG\
      4D 4F 43 03  00035 P.AAL:  .ASCII  <3>\COM\
   4F 4D 4F 43 04  00039 P.AAM:  .ASCII  <4>\COMO\
      52 55 43 03  0003E P.AAN:  .ASCII  <3>\CUR\
54 53 41 57 52 05  00042 P.AAO:  .ASCII  <5>\RWAST\
58 42 4D 57 52 05  00048 P.AAP:  .ASCII  <5>\RWMBX\
47 50 4E 57 52 05  0004E P.AAQ:  .ASCII  <5>\RWNPG\
46 46 50 57 52 05  00054 P.AAR:  .ASCII  <5>\RWPFF\
47 41 50 57 52 05  0005A P.AAS:  .ASCII  <5>\RWPAG\
4B 52 42 57 52 05  00060 P.AAT:  .ASCII  <5>\RWBRK\
47 4D 49 57 52 05  00066 P.AAU:  .ASCII  <5>\RWIMG\
4F 55 51 57 52 05  0006C P.AAV:  .ASCII  <5>\RWQUO\
4B 43 4C 57 52 05  00072 P.AAW:  .ASCII  <5>\RWLCK\
50 57 53 57 52 05  00078 P.AAX:  .ASCII  <5>\RWSWP\
45 50 4D 57 52 05  0007E P.AAY:  .ASCII  <5>\RWMPE\
42 50 4D 57 52 05  00084 P.AAZ:  .ASCII  <5>\RWMPB\
```

```
                                                53 43 53 57 52 05  0008A P.ABA:  .ASCII  <5>\RWSCS\                                  ;
                                                55 4C 43 57 52 05  00090 P.ABB:  .ASCII  <5>\RWCLU\                                  ;
                                                                   00096        .BLKB   2                                            ;
                                                      4C 4C 55 46  00098 P.ABD:  .ASCII  \FULL\                                     ;
                                                         010E0004  0009C P.ABC:  .LONG   17694724                                   ;
                                                         00000000' 000A0        .ADDRESS P.ABD                                       ;
                  00 00 53 53 45 43 4F 52 50 42 55 53            000A4 P.ABF:  .ASCII  \SUBPROCESS\<0><0>                          ;
                                                         010E000A  000B0 P.ABE:  .LONG   17694730                                   ;
                                                         00000000' 000B4        .ADDRESS P.ABF                                       ;
                                 00 4B 52 4F 57 54 45 4E         000B8 P.ABH:  .ASCII  \NETWORK\<0>                                ;
                                                         010E0007  000C0 P.ABG:  .LONG   17694727                                   ;
                                                         00000000' 000C4        .ADDRESS P.ABH                                       ;
                                 00 00 00 48 43 54 41 42         000C8 P.ABJ:  .ASCII  \BATCH\<0><0><0>                            ;
                                                         010E0005  000D0 P.ABI:  .LONG   17694725                                   ;
                                                         00000000' 000D4        .ADDRESS P.ABJ                                       ;

                                                                                .PSECT  $OWN$,NOEXE,2

00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00000 STATE_TABLE:
                                                                                .ADDRESS P.AAA, P.AAB, P.AAC, P.AAD, P.AAE, -        ;
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00018               P.AAF, P.AAG, P.AAH, P.AAI, P.AAJ, P.AAK, -          ;
                                                00000000' 00000000' 00030       P.AAL, P.AAM, P.AAN                                  ;
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00038 RSN_TABLE:
                                                                                .ADDRESS P.AAO, P.AAP, P.AAQ, P.AAR, P.AAS, -        ;
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 00050               P.AAT, P.AAU, P.AAV, P.AAW, P.AAX, P.AAY, -          ;
                                                00000000' 00000000' 00068       P.AAZ, P.ABA, P.ABB                                  ;

                                                                                .EXTRN  SCS$GA_LOCALSB, SYS$GQ_VERSION
                                                                                .EXTRN  EXE$GL_ABSTIM, SCH$GL_PCBVEC
                                                                                .EXTRN  SCH$GL_MAXPIX, LIB$GET_VM
                                                                                .EXTRN  CLI$PRESENT, SHOW$WRITE_LINE
                                                                                .EXTRN  SYS$LKWSET, SYS$CMKRNL

                                                                                .PSECT  $CODE$,NOWRT,2

                                                   003C 00000                   .ENTRY  SHOW$SYSTEM, Save R2,R3,R4,R5                ; 0299
                                 55 00000000G   00  9E 00002                   MOVAB   SYS$LKWSET, R5                               ;
                                 54 00000000G   00  9E 00009                   MOVAB   LIB$STOP, R4                                 ;
                                 53 00000000G   00  9E 00010                   MOVAB   CLI$PRESENT, R3                              ;
                                 5E                 18  C2 00017                SUBL2   #24, SP                                      ;
                                          0000' CF  9F 0001A                   PUSHAB  P.ARC                                        ; 0320
                                 63             01  FB 0001E                   CALLS   #1, CLI$PRESENT                              ;
   04   AE             01       04             50  F0 00021                   INSV    R0, #4, #1, FLAGS                            ;
                                          0000' CF  9F 00027                   PUSHAB  P.ABE                                        ; 0321
                                 63             01  FB 0002B                   CALLS   #1, CLI$PRESENT                              ;
   04   AE             01       01             50  F0 0002E                   INSV    R0, #1, #1, FLAGS                            ;
                                          0000' CF  9F 00034                   PUSHAB  P.ABG                                        ; 0322
                                 63             01  FB 00038                   CALLS   #1, CLI$PRESENT                              ;
   04   AE             01       02             50  F0 0003B                   INSV    R0, #2, #1, FLAGS                            ;
                                          0000' CF  9F 00041                   PUSHAB  P.ABI                                        ; 0323
                                 63             01  FB 00045                   CALLS   #1, CLI$PRESENT                              ;
   04   AE             01       03             50  F0 00048                   INSV    R0, #3, #1, FLAGS                            ;
                       0E    04  AE             01  E0 0004E                   BBS     #1, FLAGS, 1$                                ; 0324
                       09    04  AE             02  E0 00053                   BBS     #2, FLAGS, 1$                                ; 0325
                       04    04  AE             03  E0 00058                   BBS     #3, FLAGS, 1$                                ; 0326
                             04  AE             01  88 0005D                   BISB2   #1, FLAGS                                    ; 0327
                 50 00000000G  00                 06  78 00061 1$:            ASHL    #6, SCH$GL_MAXPIX, R0                        ; 0337
```

```
                                6E      0600    C0  9E 00069          MOVAB   1536(R0), SIZE                          ;
                                          08    AE  9F 0006E          PUSHAB  DATA                                    ; 0339
                                          04    AE  9F 00071          PUSHAB  SIZE                                    ;
                        00000000G       00      02  FB 00074          CALLS   #2, LIB$GET_VM                          ;
                                52              50  D0 0007B          MOVL    R0, STATUS                              ;
                                0D              52  E8 0007E          BLBS    STATUS, 2$                              ;
                                                52  DD 00081          PUSHL   STATUS                                  ; 0341
                                                7E  D4 00083          CLRL    -(SP)                                   ;
                                    007812F2    8F  DD 00085          PUSHL   #7869170                                ;
                                64              03  FB 0008B          CALLS   #3, LIB$STOP                            ;
                        50          08  AE      6E  C1 0008E 2$:      ADDL3   SIZE, DATA, R0                          ; 0342
                                    0C  AE  FF  A0  9E 00093          MOVAB   -1(R0), DATA+4                          ;
                                    10  AE  0000V CF  9E 00098          MOVAB   LOCK_START, DESC                        ; 0348
                                    14  AE  0000V CF  9E 0009E          MOVAB   LOCK_END, DESC+4                        ; 0349
                                                7E  7C 000A4          CLRQ    -(SP)                                   ; 0350
                                          18    AE  9F 000A6          PUSHAB  DESC                                    ;
                                65              03  FB 000A9          CALLS   #3, SYS$LKWSET                          ;
                                52              50  D0 000AC          MOVL    R0, STATUS                              ;
                                05              52  E8 000AF          BLBS    STATUS, 3$                              ;
                                                52  DD 000B2          PUSHL   STATUS                                  ; 0351
                                C4              01  FB 000B4          CALLS   #1, LIB$STOP                            ;
                                    10  AE      08  AE  D0 000B7 3$:      MOVL    DATA, DESC                          ; 0353
                            14  AE      08  AE  00000040    8F  C1 000BC          ADDL3   #64, DATA, DESC+4               ; 0354
                                                7E  7C 000C6          CLRQ    -(SP)                                   ; 0355
                                          18    AE  9F 000C8          PUSHAB  DESC                                    ;
                                65              03  FB 000CB          CALLS   #3, SYS$LKWSET                          ;
                                52              50  D0 000CE          MOVL    R0, STATUS                              ;
                                05              52  E8 000D1          BLBS    STATUS, 4$                              ;
                                                52  DD 000D4          PUSHL   STATUS                                  ; 0356
                                64              01  FB 000D6          CALLS   #1, LIB$STOP                            ;
                                    10  AE          02  D0 000D9 4$:      MOVL    #2, DESC                            ; 0362
                                    14  AE      08  AE  9E 000DD          MOVAB   DATA, DESC+4                        ; 0363
                                        6D      04  AE  9E 000E2          MOVAB   FLAGS, DESC+8                       ; 0364
                                          10    AE  9F 000E6          PUSHAB  DESC                                    ; 0366
                                    0000V   CF      9F 000E9          PUSHAB  GET_DATA                                ;
                        00000000G       00      02  FB 000ED          CALLS   #2, SYS$CMKRNL                          ;
                                52              50  D0 000F4          MOVL    R0, STATUS                              ;
                                0A              52  E8 000F7          BLBS    STATUS, 5$                              ;
                                                52  DD 000FA          PUSHL   STATUS                                  ; 0369
                        00000000G       00      01  FB 000FC          CALLS   #1, LIB$SIGNAL                          ;
                                                04  00103            RET                                             ; 0368
                                          04    AE  9F 00104 5$:      PUSHAB  FLAGS                                   ; 0376
                                          0C    AE  9F 00107          PUSHAB  DATA                                    ;
                                0000V   CF      02  FB 0010A          CALLS   #2, PRINT_DATA                          ;
                                                04  0010F            RET                                             ; 0379

; Routine Size:  272 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  287          0380  1 OWN lock_start : VECTOR[0] PSECT ($CODE$);      ! Beginning of locked code
;  288          0381  1 ROUTINE get_data (data, flags) =
;  289          0382  2 BEGIN
;  290          0383  2
;  291          0384  2 !---
;  292          0385  2 !
;  293          0386  2 ! This routine executes in KERNEL mode.  It scans all the processes in the
;  294          0387  2 ! system, gathering information on them.
;  295          0388  2 !
;  296          0389  2 ! Inputs
;  297          0390  2 !       DATA  -- address of the scratch area
;  298          0391  2 !       FLAGS -- options longword, to tell what kind of processes are desired
;  299          0392  2 !
;  300          0393  2 ! Outputs
;  301          0394  2 !       DATA -- will contain information on the processes
;  302          0395  2 !
;  303          0396  2 !--
;  304          0397  2
;  305          0398  2 MAP
;  306          0399  2     data : REF VECTOR,
;  307          0400  2     flags : REF $BBLOCK;
;  308          0401  2
;  309          0402  2 REGISTER
;  310          0403  2     locked : REF $BBLOCK,                      ! Pointer to locked page
;  311          0404  2     scratch : REF $BBLOCK,                     ! Pointer to scratch area
;  312          0405  2     pcb : REF $BBLOCK,                         ! Pointer to PCB
;  313          0406  2     null,                                      ! Null process PCB address
;  314          0407  2     pix;                                       ! Process index
;  315          0408  2
;  316          0409  2 !
;  317          0410  2 ! The first page of the scratch area is locked, so that it can be accessed at
;  318          0411  2 ! elevated IPL.  This locked portion will be a temporary storage place for
;  319          0412  2 ! information about one process at a time.  The remainder of the scratch area
;  320          0413  2 ! will contain information on the processes which are to be displayed.
;  321          0414  2 ! Set up these areas so that they can be addressed easily.
;  322          0415  2 !
;  323          0416  2 locked = .data[0];                              ! Point to locked area
;  324          0417  2 scratch = .data[0] + d$k_length;                ! Scratch area is just beyond
;  325          0418  2                                                 ! the locked data
;  326          0419  2
;  327          0420  2 null = pcb = .sch$gl_pcbvec[0];                 ! Save address of NULL PCB
;  328          0421  2
;  329          0422  2 INCR pix FROM 0 TO .sch$gl_maxpix
;  330          0423  2 DO
;  331          0424  3     BEGIN
;  332          0425  3     SET_IPL(IPL$_SYNCH);                       ! Raise IPL
;  333          0426  3     IF .pix EQL 0
;  334          0427  3     OR (pcb = .sch$gl_pcbvec[.pix]) NEQ .null
;  335          0428  3     THEN
;  336          0429  4         BEGIN
;  337          0430  4         locked[d$l_pid] = .pcb[pcb$l_epid];    ! Use the extended pid
;  338          0431  4         locked[d$l_owner] = .pcb[pcb$l_owner];
;  339          0432  4         locked[d$l_uic] = .pcb[pcb$l_uic];
;  340          0433  4         locked[d$l_state] = .pcb[pcb$w_state];
;  341          0434  4         locked[d$l_pri] = .pcb[pcb$b_pri];
;  342          0435  4         locked[d$l_pgcnt] = .pcb[pcb$w_ppgcnt] + .pcb[pcb$w_gpgcrt];
;  343          0436  4         locked[d$l_lef] = .pcb[pcb$l_efwm];
```

```
;  344     0437  5          IF (locked[d$l_sts] = .pcb[pcb$l_sts])
;  345     0438  4          THEN
;  346     0439  5              BEGIN
;  347     0440  5              locked[d$l_iocnt] = .$BBLOCK[.pcb[pcb$l_phd], phd$l_diocnt] +
;  348     0441  5                                  .$BBLOCK[.pcb[pcb$l_phd], phd$l_biocnt];
;  349     0442  5              locked[d$l_pflts] = .$BBLOCK[.pcb[pcb$l_phd], phd$l_pageflts];
;  350     0443  5              locked[d$l_cputim] = .$BBLOCK[.pcb[pcb$l_phd], phd$l_cputim];
;  351     0444  4              END;
;  352     0445  4          CH$MOVE(pcb$s_lname,
;  353     0446  4                  pcb[pcb$t_lname],
;  354     0447  4                  locked[d$t_name]);
;  355     0448  4          SET_IPL(0);
;  356     0449  4
;  357     0450  4          IF .flags[sys$v_proc]
;  358     0451  5          OR (.flags[sys$v_subp] AND .locked[d$l_owner] NEQ 0)
;  359     0452  5          OR (.flags[sys$v_batch] AND .$BBLOCK[locked[d$l_sts], d$v_batch])
;  360     0453  5          OR (.flags[sys$v_net] AND .$BBLOCK[locked[d$l_sts], d$v_netwrk])
;  361     0454  4          THEN scratch = CH$MOVE(d$k_length, .locked, .scratch);
;  362     0455  3          END;
;  363     0456  3
;  364     0457  3      SET_IPL(0);
;  365     0458  2      END;
;  366     0459  2
;  367     0460  2 RETURN 1;                                   ! Return
;  368     0461  1 END;                                ! End of GET_DATA


                                00110 LOCK_START:
                                      .BLKB   0


                                OFFC 00000 GET_DATA:
                                      .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11  ; 0381
                    56      04    BC  D0 00002      MOVL    @DATA, LOCKED           ; 0416
          57        04    BC 00000040   8F  C1 00006      ADDL3   #64, @DATA, SCRATCH     ; 0417
                    5B 00000000G  00  D0 0000F      MOVL    SCH$GL_PCBVEC, R11      ; 0420
                    58            6B  D0 00016      MOVL    (R11), PCB              ;
                    59            58  D0 00019      MOVL    PCB, NULL               ;
                    5A            01  CE 0001C      MNEGL   #1, PIX                 ; 0422
                              0092  31  0001F      BRW     8$                      ;
                    12            08  DA 00022 1$:  MTPR    #8, #18                 ; 0425
                                  5A  D5 00025      TSTL    PIX                     ; 0426
                                  09  13 00027      BEQL    2$                      ;
                    58          6B4A  D0 00029      MOVL    (R11)[PIX], PCB         ; 0427
                    59            58  D1 0002D      CMPL    PCB, NULL               ;
                                  7F  13 00030      BEQL    7$                      ;
                    66      64    A8  D0 00032 2$:  MOVL    100(PCB), (LOCKED)      ; 0430
              04    A6      1C    A8  D0 00036      MOVL    28(PCB), 4(LOCKED)      ; 0431
              28    A6    00BC    C8  D0 0003B      MOVL    188(PCB), 40(LOCKED)    ; 0432
              0C    A6      2C    A8  3C 00041      MOVZWL  44(PCB), 12(LOCKED)     ; 0433
              10    A6      0B    A8  9A 00046      MOVZBL  11(PCB), 16(LOCKED)     ; 0434
                    50      36    A8  3C 0004B      MOVZWL  54(PCB), R0             ; 0435
                    51      34    A8  3C 0004F      MOVZWL  52(PCB), R1             ;
    20    A6              50          51  C1 00053      ADDL3   R1, R0, 32(LOCKED)      ;
              2C    A6      4C    A8  D0 00058      MOVL    76(PCB), 44(LOCKED)     ; 0436
```

```
                                50        24    A8  D0 0005D          MOVL    36(PCB), R0                                 ; 0437
                           24   A6              50  D0 00061          MOVL    R0, 36(LOCKED)                              ;
                                15              50  E9 00065          BLBC    R0, 3$                                      ;
                                50        6C    A8  D0 00068          MOVL    108(PCB), R0                                ; 0440
                      14   A6        54    A0  58  A0 C1 0006C          ADDL3   88(R0), 84(R0), 20(LOCKED)                  ; 0441
                           1C   A6   4C    A0  D0 00073          MOVL    76(R0), 28(LOCKED)                          ; 0442
                           18   A6   38    A0  D0 00078          MOVL    56(R0), 24(LOCKED)                          ; 0443
                      30   A6        70    A8              10  28 0007D 3$:    MOVC3   #16, 112(PCB), 48(LOCKED)           ; 0447
                                12              00  DA 00083          MTPR    #0, #18                                     ; 0448
                                1E        08    BC  E8 00086          BLBS    @FLAGS, 6$                                  ; 0450
                           05        08    BC              01  E1 0008A          BBC     #1, @FLAGS, 4$                      ; 0451
                                          04    A6  D5 0008F          TSTL    4(LOCKED)                                   ;
                                                14  12 00092          BNEQ    6$                                          ;
                           05        08    BC              03  E1 00094 4$:    BBC     #3, @FLAGS, 5$                      ; 0452
                           0A   25   A6              06  E0 00099          BBS     #6, 37(LOCKED), 6$                          ;
                           0E   08   BC              02  E1 0009E 5$:    BBC     #2, @FLAGS, 7$                      ; 0453
                           09   26   A6              05  E1 000A3          BBC     #5, 38(LOCKED), 7$                          ;
                           67             66   0040  8F  28 000A8 6$:    MOVC3   #64, (LOCKED), (SCRATCH)            ; 0454
                                57              53  D0 000AE          MOVL    R3, SCRATCH                                 ;
                                12              00  DA 000B1 7$:    MTPR    #0, #18                                     ; 0457
                FF64            5A        01 00000000G  00  F1 000B4 8$:    ACBL    SCH$GL_MAXPIX, #1, PIX, 1$                  ; 0422
                                50              01  D0 000BE          MOVL    #1, R0                                      ; 0460
                                                04  00GC1          RET                                                 ; 0461

; Routine Size:  194 bytes,    Routine Base:  $CODE$ + 011C

;  369          0462  1 OWN lock_end : VECTOR[0] PSECT ($CODE$);        ! End of locked code
```

```
;  371     0463  1 ROUTINE print_data (data,flags) : NOVALUE =
;  372     0464  2 BEGIN
;  373     0465  2
;  374     0466  2 !---
;  375     0467  2 !
;  376     0468  2 ! This routine prints the data contained in DATA, the scratch area
;  377     0469  2 !
;  378     0470  2 ! Inputs
;  379     0471  2 !       DATA -- scratch area, contains the process information (didn't I
;  380     0472  2 !               just say that?)
;  381     0473  2 !
;  382     0474  2 !       FLAGS -- contains the bits set for the qualifiers.  It is specifically
;  383     0475  2 !               used in this routine to determine if the /FULL qualifier is
;  384     0476  2 !               set.
;  385     0477  2 !
;  386     0478  2 ! Outputs
;  387     0479  2 !       The process data is printed.
;  388     0480  2 !
;  389     0481  2 !--
;  390     0482  2
;  391     0483  2 MAP
;  392     0484  2     data : REF VECTOR,
;  393     0485  2     flags: REF $BBLOCK;
;  394     0486  2
;  395     0487  2 LOCAL
;  396     0488  2     scratch : REF $BBLOCK,                      ! Pointer to scratch area
;  397     0489  2     status,                                     ! General status
;  398     0490  2     time : VECTOR[2],                           ! Place to put system time
;  399     0491  2     desc : VECTOR[2],                           ! Descriptor for $FAOL
;  400     0492  2     proctim : VECTOR[4],                        ! Storage area for process time
;  401     0493  2     arglist : VECTOR[4],                        ! Argument list for $FAOL
;  402     0494  2     sysnodebuf : VECTOR[16,BYTE],               ! 16 byte buffer to receive system node
;  403     0495  2     desc_sysnode: $BBLOCK[DSC$C_S_BLN],         ! String descriptor pointing to
;  404     0496  2                                                 ! sysnodebuf
;  405     0497  2     trnlnmlst : $ITMLST_DECL (ITEMS = 1);       ! Item list for translating system
;  406     0498  2                                                 ! node.  [vms.lib]utldef.b32
;  407     0499  2
;  408     0500  2 ! Set up string descriptor to find the logical name system table
;  409     0501  2
;  410     0502  2 sd ('LNM$SYSTEM');
;  411     0503  2
;  412     0504  2 !
;  413     0505  2 ! Set up the scratch area, which contains all the data about the processes.
;  414     0506  2 ! The data is located beyond the locked segment.
;  415     0507  2 !
;  416     0508  2 scratch = .data[0] + d$k_length;                ! Process data begins here.
;  417     0509  2
;  418     0510  2 !
;  419     0511  2 ! If there is no data in the scratch area, then simply return.
;  420     0512  2 !
;  421     0513  2 IF .scratch[d$l_pid] EQL 0
;  422     0514  2 THEN RETURN;
;  423     0515  2
;  424     0516  2 !
;  425     0517  2 ! Determine the time the system has been up.  This is done using the
;  426     0518  2 ! value of EXE$GL_ABSTIM, multiplying it by the right constant and
;  427     0519  2 ! handing it to $ASCTIM.
```

```
;  428          0520  2 !
;  429          0521  2 EMUL(%REF(.exe$gl_abstim), %REF(-10000000), %REF(0), time); ! Get the uptime
;  430          0522  2 desc[0] = d$k_length;                          ! Set up a descriptor pointing
;  431          0523  2 desc[1] = .data[0];                           ! to the locked (scratch) area
;  432        P 0524  3 IF NOT (status = $ASCTIM(TIMADR = time,        ! Convert the uptime
;  433        P 0525  3                          TIMBUF = desc,        ! to ASCII, storing here,
;  434        P 0526  3                          TIMLEN = desc,        ! put length here,
;  435          0527  3                          CVTFLG = 0))          ! and give full date and time
;  436          0528  2 THEN
;  437          0529  3     BEGIN
;  438          0530  3     SIGNAL(.status);
;  439          0531  3     RETURN;
;  440          0532  2     END;
;  441          0533  2
;  442          0534  2 ! Initialize the descriptor for system node.
;  443          0535  2
;  444          0536  2 desc_sysnode = 16;                            ! Address of space containing
;  445          0537  2 desc_sysnode[dsc$a_pointer] = sysnodebuf;      ! system node
;  446          0538  2
;  447          0539  2 ! Initialize item list used to find system node
;  448          0540  2
;  449        P 0541  2 $ITMLST_INIT (ITMLST = trnlnmlst,              ! Pre-declared address for item list
;  450        P 0542  2                      (ITMCOD = LNM$_STRING,    ! Logical name translation string
;  451        P 0543  2                                                ! is to be obtained
;  452        P 0544  2                       BUFADR = sysnodebuf,     ! Address to put system node
;  453        P 0545  2                       BUFSIZ = 16,             ! Length of BUFADR
;  454        P 0546  2                       RETLEN = desc_sysnode)   ! Length of returned node
;  455          0547  2                 );
;  456          0548  2
;  457          0549  2 ! Get system node.  Documented in Specification for VMS Logical Name Extension
;  458          0550  2
;  459        P 0551  3 IF NOT (status = $TRNLNM
;  460        P 0552  3                 (ATTR   = %REF(lnm$m_case_blind), ! Letter case makes no difference
;  461        P 0553  3                  TABNAM = SD_LNM$SYSTEM,          ! Logical name table to be searched
;  462        P 0554  3                  LOGNAM = $DESCRIPTOR('SYS$NODE'),! What to translate
;  463        P 0555  3                  ACMODE = %REF(PSL$C_EXEC),       ! Access mode to use
;  464        P 0556  3                  ITMLST = trnlnmlst               ! Predefined item list
;  465          0557  4                  )
;  466          0558  3           )
;  467          0559  2     THEN desc_sysnode[dsc$w_length] = 0
;  468          0560  2     ELSE
;  469          0561  2
;  470          0562  2 ! Strip leading underscore and trailing colons, if either, from node name
;  471          0563  2
;  472          0564  3     BEGIN
;  473          0565  3       IF .sysnodebuf[0] EQL '_'
;  474          0566  3       THEN
;  475          0567  4       BEGIN
;  476          0568  4         desc_sysnode[dsc$w_length] = .desc_sysnode[dsc$w_length] - 1;
;  477          0569  4         desc_sysnode[dsc$a_pointer] = .desc_sysnode[dsc$a_pointer] + 1;
;  478          0570  3       END;
;  479          0571  3
;  480          0572  3       INCRU I FROM 0 TO 1
;  481          0573  3         DO IF NOT CH$FAIL(CH$FIND_CH(.desc_sysnode[dsc$w_length],
;  482          0574  3                                      .desc_sysnode[dsc$a_pointer], ':'))
;  483          0575  3              THEN desc_sysnode[dsc$w_length] = .desc_sysnode[dsc$w_length] - 1;
;  484          0576  2      END;
```

```
;  485          0577  2 !
;  486          0578  2 ! Set up the $FAOL parameter list, with the addresses of the descriptors of
;  487          0579  2 ! the system version, the system node name, and the uptime.
;  488          0580  2 !
;  489          0581  2
;  490          0582  2 arglist[0] = UPLIT(4, sys$gq_version);          ! Version number is 4 bytes
;  491          0583  2 arglist[1] = desc_sysnode;                     ! System node name string desc
;  492          0584  2 arglist[2] = 0;                                ! Zero to get current date
;  493          0585  2 arglist[3] = desc;                            ! Uptime string desc
;  494          0586  2 desc[0] = .desc[0] - 3;                       ! Get rid of trailing ".00"
;  495          0587  2
;  496          0588  2 !
;  497          0589  2 ! Now format and print the header lines.
;  498          0590  2 !
;  499          0591  2 show$write_line(%ASCID 'VAX/VMS !AS  on node !AS !%D   Uptime !AS',
;  500          0592  2                 arglist,
;  501          0593  2                 %ASCID '  Pid    Process Name    State  Pri      I/O       CPU       Page flts Ph.Mem',
;  502          0594  2                 0);
;  503          0595  2
;  504          0596  2 !
;  505          0597  2 ! Loop thru the scratch area, formatting and outputting the data one process
;  506          0598  2 ! at a time.  The data block for each process is set up in as an ordered
;  507          0599  2 ! sequence of longwords, in the order of the arguments to $FAOL, so that the
;  508          0600  2 ! data block itself can be used as the parameter list to $FAOL.  All that is
;  509          0601  2 ! required is some minor fixup.
;  510          0602  2 !
;  511          0603  2 WHILE .scratch[d$l_pid] NEQ 0                  ! Loop thru all processes
;  512          0604  2 DO
;  513          0605  3     BEGIN
;  514          0606  3 !
;  515          0607  3 ! Get the state.
;  516          0608  3 !
;  517          0609  3     IF .scratch[d$l_state] GEQ sch$c_colpg
;  518          0610  3     AND .scratch[d$l_state] LEQ sch$c_cur
;  519          0611  3     THEN
;  520          0612  4         BEGIN
;  521          0613  4         IF .scratch[d$l_state] EQL sch$c_mwait
;  522          0614  4         AND .scratch[d$l_lef] GEQ 0
;  523          0615  4         THEN
;  524          0616  5             BEGIN
;  525          0617  5             IF .scratch[d$l_lef] GEQ 1
;  526          0618  5             AND .scratch[d$l_lef] LEQ rsn_cnt
;  527          0619  5             THEN scratch[d$l_state] = .rsn_table[.scratch[d$l_lef] - 1]
;  528          0620  5             ELSE scratch[d$l_state] = cstring('RWUNK');
;  529          0621  5             END
;  530          0622  4         ELSE  scratch[d$l_state] = .state_table[.scratch[d$l_state] - 1];
;  531          0623  4         END
;  532          0624  3     ELSE scratch[d$l_state] = cstring('UNK');
;  533          0625  3
;  534          0626  3 !
;  535          0627  3 ! If the owner field is not empty, then the process is a subprocess;
;  536          0628  3 ! otherwise, check the status bits for a network or batch process.
;  537          0629  3 !
;  538          0630  3     status = .scratch[d$l_sts];                  ! Save the status
;  539          0631  3     IF .scratch[d$l_owner] NEQ 0
;  540          0632  3     THEN scratch[d$l_sts] = cstring('S')
;  541          0633  3     ELSE IF .$BBLOCK[scratch[d$l_sts], d$v_netwrk]
```

```
;  542          0634  3     THEN scratch[d$l_sts] = cstring('N')
;  543          0635  3     ELSE IF .$BBLOCK[scratch[d$l_sts], d$v_batch]
;  544          0636  3     THEN scratch[d$l_sts] = cstring('B')
;  545          0637  3     ELSE scratch[d$l_sts] = cstring(' ');
;  546          0638  3
;  547          0639  3 !
;  548          0640  3 ! Convert the priority from the internal format to the external format.
;  549          0641  3 !
;  550          0642  3     scratch[d$l_pri] = 31 - .scratch[d$l_pri];
;  551          0643  3
;  552          0644  3 !
;  553          0645  3 ! The process name string should be converted from the counted string
;  554          0646  3 ! format used in the PCB to a length/address descriptor, so that $FAO
;  555          0647  3 ! can print non-alphanumerics as periods.  To do this, use the OWNER
;  556          0648  3 ! field in the scratch area as the count, and point to the beginning
;  557          0649  3 ! of the saved string, instead of the count.
;  558          0650  3 !
;  559          0651  3     scratch[d$l_owner] = .scratch[d$t_name];
;  560          0652  3     scratch[d$a_name] = scratch[d$t_name] + 1;
;  561          0653  3
;  562          0654  3 !
;  563          0655  3 ! Multiply the CPU time for the process by -100000, setting it up for
;  564          0656  3 ! conversion into ASCII form.
;  565          0657  3 !
;  566          0658  3     IF .status
;  567          0659  3     THEN
;  568          0660  4         BEGIN
;  569          0661  4         EMUL(scratch[d$l_cputim], %REF(-100000), %REF(0), time);
;  570          0662  4         desc[0] = 16;
;  571          0663  4         desc[1] = proctim;                          ! Desc will contain the address
;  572          0664  4                                                     ! of where the actual time is.
;  573        P 0665  5         IF NOT (status = $ASCTIM(TIMADR = time,           ! Convert the uptime
;  574        P 0666  5                                 TIMBUF = desc,      ! to ASCII, storing here,
;  575        P 0667  5                                 TIMLEN = desc,      ! put length here,
;  576          0668  5                                 CVTFLG = 0))        ! and give full date and time
;  577          0669  4         THEN
;  578          0670  5         BEGIN
;  579          0671  5           SIGNAL(.status);
;  580          0672  5           RETURN;
;  581          0673  5         END
;  582          0674  4         ELSE
;  583          0675  4           scratch[d$l_cputim] = desc;
;  584          0676  3         END;
;  585          0677  3 !
;  586          0678  3 ! Now produce the line of text and output it.
;  587          0679  3 !
;  588          0680  3     IF .status
;  589          0681  3     THEN
;  590          0682  4     BEGIN
;  591          0683  4       IF .flags[sys$v_full]
;  592          0684  4       THEN show$write_line(%ASCID '!8XL !15AF !5AC  !3UB!9UL!AS !9UL  !5UW  !AC!/         !%I',
;  593          0685  4            .scratch)
;  594          0686  4       ELSE show$write_line(%ASCID '!8XL !15AF !5AC  !3UB!9UL!AS !9UL  !5UW  !AC',
;  595          0687  4            .scratch)
;  596          0688  4     END
;  597          0689  3        ELSE
;  598          0690  4        BEGIN
```

```
;  599          0691  4          IF .flags[sys$v_full]
;  600          0692  4          THEN show$write_line(%ASCID '!8XL !15AF !5AC  !3UB!3(+)       -- swapped out --          !5UW  !
;  601          0693  4               .scratch)
;  602          0694  4          ELSE show$write_line(%ASCID '!8XL !15AF !5AC  !3UB!3(+)       -- swapped out --          !5UW  !
;  603          0695  4               .scratch)
;  604          0696  3        END;
;  605          0697  3
;  606          0698  3    ! Adjust the scratch pointer to point to the next block of process info.
;  607          0699  3    !
;  608          0700  3        scratch = .scratch + d$k_length;
;  609          0701  2        END;
;  610          0702  2
;  611          0703  2    RETURN;
;  612          0704  1    END;                                    ! End of PRINT_DATA


                                                        001D2                .BLKB   2
                                                        001D4 LOCK_END:
                                                                             .BLKB   0

                                                                             .PSECT  $PLIT$,NOWRT,NOEXE,2

                4D  45  54  53  59  53  24  4D  4E  4C  000D8 P.ABL:   .ASCII  \LNM$SYSTEM\                                  ;
                                                        000E2            .BLKB   2                                              ;
                                    0000000A            000E4 P.ABK:   .LONG   10                                             ;
                                    00000000'           000E8          .ADDRESS P.ABL                                         ;
                        45  44  4F  4E  24  53  59  53  000EC P.ABN:   .ASCII  \SYS$NODE\                                    ;
                                    00000008            000F4 P.ABM:   .LONG   8                                              ;
                                    00000000'           000F8          .ADDRESS P.ABN                                         ;
                                    00000004            000FC P.ABO:   .LONG   4                                              ;
                                    00000000G           00100          .ADDRESS SYS$GQ_VERSION                                ;
6E  6F  20  20  53  41  21  20  53  4D  56  2F  58  41  56  00104 P.ABQ:   .ASCII  \VAX/VMS !AS  on node !AS !%D   Uptime !A\    ;
20  20  44  25  21  20  53  41  21  20  65  64  6F  6E  20  00113
                    41  21  20  65  6D  69  74  70  55  20  00122
                                        00  00  00  53  0012C          .ASCII  \S\<0><0><0>                                   ;
                                    010E0029            00130 P.ABP:   .LONG   17694761                                       ;
                                    00000000'           00134          .ADDRESS P.ABQ                                         ;
73  65  63  6F  72  50  20  20  20  20  64  69  50  20  20  00138 P.ABS:   .ASCII  \  Pid    Process Name    State  Pri     \    ;
65  74  61  74  53  20  20  20  20  65  6D  61  4E  20  73  00147
                    20  20  20  20  20  69  72  50  20  20  00156
20  55  50  43  20  20  20  20  20  20  20  20  4F  2F  49  20  00160          .ASCII  \ I/O       CPU       Page flts Ph.Mem\<0>  ;
73  74  6C  66  20  65  67  61  50  20  20  20  20  20  20  20  0016F
                            00  6D  65  4D  2E  68  50  20  00  0017E
                                                    00  00  00186          .ASCII  <0><0>                                         ;
                                    010E004D            00188 P.ABR:   .LONG   17694797                                       ;
                                    00000000'           0018C          .ADDRESS P.ABS                                         ;
                                            4B  4E  55  57  52  05  00190 P.ABT:   .ASCII  <5>\RWUNK\                           ;
                                                4B  4E  55  03  00196 P.ABU:   .ASCII  <3>\UNK\                             ;
                                                        53  01  0019A P.ABV:   .ASCII  <1>\S\                               ;
                                                        4E  01  0019C P.ABW:   .ASCII  <1>\N\                               ;
                                                        42  01  0019E P.ABX:   .ASCII  <1>\B\                               ;
                                                        20  01  001A0 P.ABY:   .ASCII  <1>\ \                               ;
                                                                001A2          .BLKB   2                                              ;
43  41  35  21  20  46  41  35  31  21  20  4C  58  38  21  001A4 P.ACA:   .ASCII  \!8XL !15AF !5AC  !3UB!9UL!AS !9UL  !5UW \    ;
21  20  53  41  21  4C  55  39  21  42  55  33  21  20  20  001B3
                    20  57  55  35  21  20  20  4C  55  39  001C2
```

```
20 20 20 20 20 20 20 20 20 2F 21 43 41 21 20 001CC          .ASCII  \ !AC!/          !%I\<0><0>              ;
                              00 00 49 25 21 001DB                                                          ;
                                    010E003A 001E0 P.ABZ:   .LONG   17694778                                ;
                                    00000000' 001E4         .ADDRESS P.ACA                                  ;
43 41 35 21 20 46 41 35 31 21 20 4C 58 38 21 001E8 P.ACC:   .ASCII  \!8XL !15AF !5AC  !3UB!9UL!AS !9UL  !5UW \ ;
21 20 53 41 21 4C 55 39 21 42 55 33 21 20 20 001F7                                                          ;
               20 57 55 35 21 20 20 4C 55 39 00206                                                          ;
                              43 41 21 20 00210         .ASCII  \ !AC\                                      ;
                                    010E002C 00214 P.ACB:   .LONG   17694764                                ;
                                    00000000' 00218         .ADDRESS P.ACC                                  ;
43 41 35 21 20 46 41 35 31 21 20 4C 58 38 21 0021C P.ACE:   .ASCII  \!8XL !15AF !5AC  !3UB!3(+)       -- swa\ ;
20 20 20 20 29 2B 28 33 21 42 55 33 21 20 20 0022B                                                          ;
               61 77 73 20 20 2D 2D 20 20 20 0023A                                                          ;
20 20 2D 2D 20 20 74 75 6F 20 20 64 65 70 70 00244         .ASCII  \pped  out  --          !5UW  !AC!/      \ ;
21 20 20 57 55 35 21 20 20 20 20 20 20 20 20 00253                                                          ;
               20 20 20 20 20 20 2F 21 43 41 00262                                                          ;
                  00 00 49 25 21 20 20 20 0026C         .ASCII  \    !%I\<0><0>                              ;
                                    010E0056 00274 P.ACD:   .LONG   17694806                                ;
                                    00000000' 00278         .ADDRESS P.ACE                                  ;
43 41 35 21 20 46 41 35 31 21 20 4C 58 38 21 0027C P.ACG:   .ASCII  \!8XL !15AF !5AC  !3UB!3(+)       -- swa\ ;
20 20 20 20 29 2B 28 33 21 42 55 33 21 20 20 0028B                                                          ;
               61 77 73 20 20 2D 2D 20 20 20 0029A                                                          ;
20 20 2D 2D 20 20 74 75 6F 20 20 64 65 70 70 002A4         .ASCII  \pped  out  --          !5UW  !AC\        ;
21 20 20 57 55 35 21 20 20 20 20 20 20 20 20 002B3                                                          ;
                                       43 41 002C2                                                          ;
                                    010E0048 002C4 P.ACF:   .LONG   17694792                                ;
                                    00000000' 002C8         .ADDRESS P.ACG                                  ;

                                                   SD_LNM$SYSTEM=       P.ABK
                                                            .EXTRN  SYS$ASCTIM, SYS$TRNLNM

                                                            .PSECT  $CODE$,NOWRT,2

                                          007C 00000 PRINT_DATA:
                                                            .WORD   Save R2,R3,R4,R5,R6                     ; 0463
                       56 00000000G  00  9E 00002           MOVAB   SHOW$WRITE_LINE, R6                     ;
                       55 00000000G  00  9E 00009           MOVAB   SYS$ASCTIM, R5                          ;
                       5E        A0  AE  9E 00010           MOVAB   -96(SP), SP                             ;
          52       04  BC 00000040  8F  C1 00014           ADDL3   #64, @DATA, SCRATCH                     ; 0508
                                     62  D5 0001D           TSTL    (SCRATCH)                               ; 0513
                                     01  12 0001F           BNEQ    1$                                      ;
                                         04 00021           RET                                             ;
   58   AE        00 FF676980  8F 00000000G  00  7A 00022 1$:  EMUL    EXE$GL_ABSTIM, #-10000000, #0, TIME     ; 0521
                       50  AE        40  8F  9A 00030           MOVZBL  #64, DESC                               ; 0522
                       54  AE        04  BC  D0 00035           MOVL    @DATA, DESC+4                           ; 0523
                                     7E  D4 0003A           CLRL    -(SP)                                   ; 0527
                             5C  AE  9F 0003C           PUSHAB  TIME                                    ;
                             58  AE  9F 0003F           PUSHAB  DESC                                    ;
                             5C  AE  9F 00042           PUSHAB  DESC                                    ;
                       65        04  FB 00045           CALLS   #4, SYS$ASCTIM                          ;
                       54        50  D0 00048           MOVL    R0, STATUS                              ;
                       03        54  E8 0004B           BLBS    STATUS, 2$                              ;
                             014B    31 0004E           BRW     19$                                     ;
                  18  AE        10  D0 00051 2$:        MOVL    #16, DESC_SYSNODE                       ; 0536
                  1C  AE   20   AE  9E 00055           MOVAB   SYSNODEBUF, DESC_SYSNODE+4              ; 0537
                      50   08   AE  9E 0005A           MOVAB   TRNLNMLST, $$ITMBLKPTR                  ; 0547
                  80 00020010   8F  D0 0005E           MOVL    #131088, ($$ITMBLKPTR)+                 ;
```

```
                                80        20    AE   9E 00065          MOVAB   SYSNODEBUF, ($$ITMBLKPTR)+                    ;
                                80        18    AE   9E 00069          MOVAB   DESC_SYSNODE, ($$ITMBLKPTR)+                   ;
                                          80    D4   0006D            CLRL    ($$ITMBLKPTR)+                                 ;
                                          08    AE   9F 0006F          PUSHAB  TRNLNMLST                                      ; 0557
                             08   AE                 01   D0 00072     MOVL    #1, 8(SP)                                      ;
                                          08    AE   9F 00076          PUSHAB  8(SP)                                          ;
                                    0000'       CF   9F 00079          PUSHAB  P.ABM                                          ;
                                    0000'       CF   9F 0007D          PUSHAB  SD_LNM$SYSTEM                                  ;
                             10   AE 02000000        8F   D0 00081     MOVL    #33554432, 16(SP)                              ;
                                          10    AE   9F 00089          PUSHAB  16(SP)                                         ;
                        00000000G         00        05   FB 0008C     CALLS   #5, SYS$TRNLNM                                 ;
                                          54        50   D0 00093     MOVL    R0, STATUS                                     ;
                                          05        54   E8 00096     BLBS    STATUS, 3$                                     ;
                                          18    AE   B4 00099          CLRW    DESC_SYSNODE                                   ; 0559
                                          27        11   0009C        BRB     8$                                             ;
                             5F   8F           20    AE   91 0009E 3$:  CMPB    SYSNODEBUF, #95                                ; 0565
                                          06        12   000A3        BNEQ    4$                                             ;
                                          18    AE   B7 000A5          DECW    DESC_SYSNODE                                   ; 0568
                                          1C    AE   D6 000A8          INCL    DESC_SYSNODE+4                                 ; 0569
                                          53        D4   000AB 4$:     CLRL    I                                              ; 0572
                   1C   BE        18   AE           3A   3A 000AD 5$:  LOCC    #58, DESC_SYSNODE, @DESC_SYSNODE+4             ; 0573
                                          02        12   000B3        BNEQ    6$                                             ;
                                          51        D4   000B5        CLRL    R1                                             ;
                                          51        D5   000B7 6$:     TSTL    R1                                             ; 0574
                                          03        13   000B9        BEQL    7$                                             ;
                                          18    AE   B7 000BB          DECW    DESC_SYSNODE                                   ; 0575
                                          53        D6   000BE 7$:     INCL    I                                              ; 0573
                             01                     53   D1 000C0     CMPL    I, #1                                          ;
                                          E8        1B   000C3        BLEQU   5$                                             ;
                   30   AE      0000'       CF   9E 000C5 8$:  MOVAB   P.ABO, ARGLIST                                         ; 0582
                   34   AE        18        AE   9E 000CB          MOVAB   DESC_SYSNODE, ARGLIST+4                            ; 0583
                                  38        AE   D4 000D0          CLRL    ARGLIST+8                                          ; 0584
                   3C   AE        50        AE   9E 000D3          MOVAB   DESC, ARGLIST+12                                   ; 0585
                   50   AE                        03   C2 000D8     SUBL2   #3, DESC                                           ; 0586
                                          7E        D4   000DC        CLRL    -(SP)                                          ; 0591
                                    0000'       CF   9F 000DE          PUSHAB  P.ABR                                          ; 0592
                                  38        AE   9F 000E2          PUSHAB  ARGLIST                                            ; 0591
                                    0000'       CF   9F 000E5          PUSHAB  P.ABP                                          ;
                             66                     04   FB 000E9     CALLS   #4, SHOW$WRITE_LINE                            ;
                                          62        D5   000EC 9$:     TSTL    (SCRATCH)                                      ; 0603
                                          01        12   000EE        BNEQ    10$                                            ;
                                                    04   000F0        RET                                                    ;
                             51        0C   A2   9E 000F1 10$:  MOVAB   12(SCRATCH), R1                                      ; 0609
                                          61        D5   000F5        TSTL    (R1)                                           ;
                                          35        15   000F7        BLEQ    13$                                            ;
                             0E                     61   D1 000F9     CMPL    (R1), #14                                      ; 0610
                                          30        14   000FC        BGTR    13$                                            ;
                             02                     61   D1 000FE     CMPL    (R1), #2                                       ; 0613
                                          20        12   00101        BNEQ    12$                                            ;
                                  2C        A2   D5 00103          TSTL    44(SCRATCH)                                        ; 0614
                                          1B        19   00106        BLSS    12$                                            ;
                                          12        15   00108        BLEQ    11$                                            ; 0617
                             0E           2C   A2   D1 0010A     CMPL    44(SCRATCH), #14                                   ; 0618
                                          0C        14   0010E        BGTR    11$                                            ;
                             50           2C   A2   D0 00110     MOVL    44(SCRATCH), R0                                    ; 0619
                             61      0000'CF   40   D0 00114     MOVL    RSN_TABLE-4[R0], (R1)                              ;
                                          17        11   0011A        BRB     14$                                            ;
```

```
                                61      0000'  CF   9E 0011C 11$:   MOVAB   P.ABT, (R1)                                  ; 0620
                                        10   11 00121             BRB     14$                                            ; 0613
                                50      61   D0 00123 12$:       MOVL    (R1), R0                                       ; 0622
                                61  0000'CF40   D0 00126           MOVL    STATE_TABLE-4[R0], (R1)
                                        05   11 0012C             BRB     14$                                            ; 0609
                                61      0000'  CF   9E 0012E 13$:   MOVAB   P.ABU, (R1)                                  ; 0624
                                50        24   A2   9E 00133 14$:   MOVAB   36(SCRATCH), R0                               ; 0630
                                54             60   D0 00137       MOVL    (R0), STATUS
                                          04   A2   D5 0013A       TSTL    4(SCRATCH)                                     ; 0631
                                          07   13 0013D           BEQL    15$
                                60      0000'  CF   9E 0013F       MOVAB   P.ABV, (R0)                                    ; 0632
                                        1B   11 00144             BRB     18$
                        07              60      15   E1 00146 15$:   BBC     #21, (R0), 16$                                 ; 0633
                                60      0000'  CF   9E 0014A       MOVAB   P.ABW, (R0)                                    ; 0634
                                        10   11 0014F             BRB     18$
                        07              60      0E   E1 00151 16$:   BBC     #14, (R0), 17$                                 ; 0635
                                60      0000'  CF   9E 00155       MOVAB   P.ABX, (R0)                                    ; 0636
                                        05   11 0015A             BRB     18$
                                60      0000'  CF   9E 0015C 17$:   MOVAB   P.ABY, (R0)                                    ; 0637
                10      A2              1F      10   A2   C3 00161 18$:   SUBL3   16(SCRATCH), #31, 16(SCRATCH)             ; 0642
                                  04    A2      30   A2   9A 00167       MOVZBL  48(SCRATCH), 4(SCRATCH)                    ; 0651
                                  08    A2      31   A2   9E 0016C       MOVAB   49(R2), 8(SCRATCH)                         ; 0652
                                        4F             54   E9 00171       BLBC    STATUS, 22$                            ; 0658
        58      AE          00 FFFE7960   8F      18   A2   7A 00174       EMUL    24(SCRATCH), #-100000, #0, TIME        ; 0661
                                  50    AE              10   D0 0017F       MOVL    #16, DESC                              ; 0662
                                  54    AE      40   AE   9E 00183       MOVAB   PROCTIM, DESC+4                            ; 0663
                                        7E   D4 00188             CLRL    -(SP)                                          ; 0668
                                        5C   AE   9F 0018A             PUSHAB  TIME
                                        58   AE   9F 0018D             PUSHAB  DESC
                                        5C   AE   9F 00190             PUSHAB  DESC
                                  65            04   FB 00193         CALLS   #4, SYS$ASCTIM
                                  54            50   D0 00196         MOVL    R0, STATUS
                                  0A            54   E8 00199         BLBS    STATUS, 20$
                                        54   DD 0019C 19$:       PUSHL   STATUS                                         ; 0671
                    00000000G     00            01   FB 0019E         CALLS   #1, LIB$SIGNAL
                                                04   A5 001A5         RET                                            ; 0670
                          18    A2      50   AE   9E 001A6 20$:       MOVAB   DESC, 24(SCRATCH)                          ; 0675
                                15            54   E9 001AB         BLBC    STATUS, 22$                                ; 0680
                08        08    BC            04   E1 001AE         BBC     #4, @FLAGS, 21$                            ; 0683
                                        52   DD 001B3             PUSHL   SCRATCH                                        ; 0685
                                0000'   CF   9F 001B5             PUSHAB  P.ABZ                                          ; 0684
                                        1B   11 001B9             BRB     24$
                                        52   DD 001BB 21$:       PUSHL   SCRATCH                                        ; 0687
                                0000'   CF   9F 001BD             PUSHAB  P.ACB                                          ; 0686
                                        13   11 001C1             BRB     24$
                08        08    BC            04   E1 001C3 22$:       BBC     #4, @FLAGS, 23$                            ; 0691
                                        52   DD 001C8             PUSHL   SCRATCH                                        ; 0693
                                0000'   CF   9F 001CA             PUSHAB  P.ACD                                          ; 0692
                                        06   11 001CE             BRB     24$
                                        52   DD 001D0 23$:       PUSHL   SCRATCH                                        ; 0695
                                0000'   CF   9F 001D2             PUSHAB  P.ACF                                          ; 0694
                                  66            02   FB 001D6 24$:       CALLS   #2, SHOW$WRITE_LINE
                                  52      40    A2   9E 001D9         MOVAB   64(R2), SCRATCH                            ; 0700
                                          FF0C  31   001DD             BRW     9$                                         ; 0603
                                                04   001E0             RET                                            ; 0704

; Routine Size:  481 bytes,    Routine Base:  $CODE$ + 01D4
```

```
;  614          0705  1 END
;  615          0706  0 ELUDOM


                                                        .EXTRN  LIB$SIGNAL, LIB$STOP

;                               PSECT SUMMARY
;
;      Name                     Bytes                          Attributes
;
;  $PLIT$                         716  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $OWN$                          112  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                         949  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        62         0        1000        00:01.9


;                                   COMMAND QUALIFIERS

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

; Size:          947 code + 830 data bytes
; Run Time:          00:22.7
; Elapsed Time:      01:14.4
; Lines/CPU Min:      1862
; Lexemes/CPU-Min:  23509
; Memory Used:   204 pages
; Compilation Complete
```

SHOWTERM
LIS

SHOWMISC
LIS

SHOWPROC
LIS

SHOWSYS
LIS

SHOWMAIN
LIS

SHOWMSCP
LIS

SHOWQUE
LIS

SHOWMSG
LIS