```
   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
```

FILEID**SHODEVCLU

N 7

```
  SSSSSSSS  HH      HH    OOOOOO    DDDDDDDD    EEEEEEEEEE  VV      VV    CCCCCCCC  LL          UU      UU
  SSSSSSSS  HH      HH    OOOOOO    DDDDDDDD    EEEEEEEEEE  VV      VV    CCCCCCCC  LL          UU      UU
SS          HH      HH  OO      OO  DD      DD  EE          VV      VV  CC          LL          UU      UU
SS          HH      HH  OO      OO  DD      DD  EE          VV      VV  CC          LL          UU      UU
SS          HH      HH  OO      OO  DD      DD  EE          VV      VV  CC          LL          UU      UU
SS          HH      HH  OO      OO  DD      DD  EE          VV      VV  CC          LL          UU      UU
  SSSSSS    HHHHHHHHHH  OO      OO  DD      DD  EEEEEEEE    VV      VV  CC          LL          UU      UU
  SSSSSS    HHHHHHHHHH  OO      OO  DD      DD  EEEEEEEE    VV      VV  CC          LL          UU      UU
        SS  HH      HH  OO      OO  DD      DD  EE          VV      VV  CC          LL          UU      UU
        SS  HH      HH  OO      OO  DD      DD  EE          VV      VV  CC          LL          UU      UU
        SS  HH      HH  OO      OO  DD      DD  EE            VV  VV    CC          LL          UU      UU      ....
        SS  HH      HH  OO      OO  DD      DD  EE            VV  VV    CC          LL          UU      UU      ....
SSSSSSSS    HH      HH    OOOOOO    DDDDDDDD    EEEEEEEEEE      VV      CCCCCCCC  LLLLLLLLLL  UUUUUUUUUU      ....
SSSSSSSS    HH      HH    OOOOOO    DDDDDDDD    EEEEEEEEEE      VV      CCCCCCCC  LLLLLLLLLL  UUUUUUUUUU      ....

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 shodevclu (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 finding cluster-wide
;    37          0037  1 !       information about devices by chasing through the lock structures.
;    38          0038  1 !
;    39          0039  1 ! ENVIRONMENT:
;    40          0040  1 !       VAX native, user mode.
;    41          0041  1 !
;    42          0042  1 ! AUTHOR:  CW Hobbs              CREATION DATE:  19-Mar-1984
;    43          0043  1 !
;    44          0044  1 ! MODIFIED BY:
;    45          0045  1 !
;    46          0046  1 !       V03-005 CWH3005                 CW Hobbs                4-May-1984
;    47          0047  1 !               Exclude all null locks from consideration, since the are
;    48          0048  1 !               due to other SHOW DEVICE commands rather than something
;    49          0049  1 !               interesting.
;    50          0050  1 !
;    51          0051  1 !       V03-004 CWH3004                 CW Hobbs                13-Apr-1984
;    52          0052  1 !               Remove declaration for a debugging routine.
;    53          0053  1 !
;    54          0054  1 !       V03-003 CWH3003                 CW Hobbs                13-Apr-1984
;    55          0055  1 !               Change name for LKI$L_REMSYSTEM to LKI$L_REMSYSID and LKI$B_STATE
;    56          0056  1 !               to LKI$B_QUEUE because the definitions changed.
;    57          0057  1 !
```

```
;    58          0058  1 !       V03-002 CWH3002                 CW Hobbs                    12-Apr-1984
;    59          0059  1 !               Complete work now that a full service $GETLKI is available
;    60          0060  1 !
;    61          0061  1 !
;    62          0062  1 !--
```

```
;    64          0063  1 !
;    65          0064  1 ! Include files
;    66          0065  1 !
;    67          0066  1
;    68          0067  1 LIBRARY 'SYS$LIBRARY:LIB';                  ! VAX/VMS system definitions
;    69          0068  1 REQUIRE 'SRC$:SHOWDEF';                     ! SHOW common definitions
;    70          0167  1 REQUIRE 'SRC$:SHODEVDEF';                   ! SHOW DEVICES common definitions
;    71          0458  1
;    72          0459  1 !
;    73          0460  1 ! Table of contents
;    74          0461  1 !
;    75          0462  1 FORWARD ROUTINE
;    76          0463  1     scan_cluster_locks : NOVALUE,               ! User-mode jacket
;    77          0464  1     get_lock_info,                              ! Kernel routine to follow locks for device
;    78          0465  1     get_lock_info_handler;                      ! Handler to keep get_lock_info out of trouble
;    79          0466  1
;    80          0467  1 EXTERNAL ROUTINE
;    81          0468  1     show$write_line;
;    82          0469  1
;    83          0470  1 EXTERNAL LITERAL
;    84          0471  1     show$_lockerr;                              ! Error chasing locks
;    85          0472  1
;    86          0473  1 EXTERNAL
;    87          0474  1     lck$gl_maxid,
;    88          0475  1     lck$gl_idtbl : REF VECTOR [, LONG],
;    89          0476  1     kernel_accvio : VECTOR [4, LONG];
;    90          0477  1
;    91          0478  1 !
;    92          0479  1 ! Define a structure for a local buffer used to pass various items from
;    93          0480  1 ! the kernel-mode routine back to the user-mode routine
;    94          0481  1 !
;    95          0482  1 MACRO
;    96          0483  1     lcl_null_lkid       =  0, 0, 32, 0 %,       ! Id of the null mode lock we declared
;    97          0484  1     lcl_lengths         =  4, 0, 32, 0 %,       ! Longword containing both lengths
;    98          0485  1     lcl_ret_length      =  4, 0, 16, 0 %,       ! Word containing total length of items returned
;    99          0486  1     lcl_itm_length      =  6, 0, 15, 0 %,       ! Length of a single lock item
;   100          0487  1     lcl_enq_status      =  8, 0, 32, 0 %,       ! Status from $ENQ for null lock
;   101          0488  1     lcl_val_block       = 12, 0,  0, 0 %;       ! Value block for the resource
;   102          0489  1
;   103          0490  1 LITERAL
;   104          0491  1     lcl_size = 28;                              ! Total size 12 bytes + 16 byte value block
;   105          0492  1
;   106          0493  1 !
;   107          0494  1 ! We would like to be able to REQUIRE 'SHRLIB$:MOUDEF.B32', but MOUDEF has a bunch of
;   108          0495  1 ! definitions which conflict with our own definitions.  Therefore, we have a copy of the
;   109          0496  1 ! definitions which we need.
;   110          0497  1 !
;   111          0498  1 ! Define fields within the device allocation lock value block.
;   112          0499  1 !
;   113          0500  1
;   114          0501  1 MACRO
;   115          0502  1         DC_FLAGS                = 0,0,16,0 %,
;   116          0503  1         DC_NOTFIRST_MNT         = 0,0,1,0 %,
;   117          0504  1         DC_FOREIGN              = 0,1,1,0 %,
;   118          0505  1         DC_GROUP                = 0,2,1,0 %,
;   119          0506  1         DC_SYSTEM               = 0,3,1,0 %,
;   120          0507  1         DC_WRITE                = 0,4,1,0 %,
```

```
;  121          0508  1          DC_NOQUOTA                = 0,5,1,0 %,
;  122          0509  1          DC_OVR_PROT               = 0,6,1,0 %,
;  123          0510  1          DC_OVR_OWNUIC             = 0,7,1,0 %,
;  124          0511  1          DC_NOINTERLOCK            = 0,8,1,0 %,
;  125          0512  1          DC_PROTECTION             = 2,0,16,0 %,
;  126          0513  1          DC_OWNER_UIC              = 4,0,32,0 %;
```

```
;  128          0514  1 GLOBAL ROUTINE scan_cluster_locks (scratch : REF $BBLOCK, buffer : REF $BBLOCK) : NOVALUE =
;  129          0515  2 BEGIN
;  130          0516  2
;  131          0517  2 !---
;  132          0518  2 !
;  133          0519  2 ! This is a user-mode jacket routine for the lock searches
;  134          0520  2 !
;  135          0521  2 ! Inputs
;  136          0522  2 !       SCRATCH  - address of scratch data for this device
;  137          0523  2 !
;  138          0524  2 ! Outputs
;  139          0525  2 !       SCRATCH  - some values "adjusted" for cluster-wide information
;  140          0526  2 !       BUFFER   - output lock information, for now gets a vector containing
;  141          0527  2 !                  the CSIDs of the remote nodes.  First longword is the count
;  142          0528  2 !                  of the CSIDs, longword CSIDs start at second longword
;  143          0529  2 !
;  144          0530  2 !---
;  145          0531  2
;  146          0532  2 OWN
;  147          0533  2     local_csid : INITIAL (0);                  ! The CSID of the local node
;  148          0534  2
;  149          0535  2 LOCAL
;  150          0536  2     status,
;  151          0537  2     lclbuf : $BBLOCK[lcl_size],                ! Buffer to receive misc items from kernel call
;  152          0538  2     lokbuf : $BBLOCK[1200],                    ! Lock info buffer for kernel routines (set to MAXBUF minimu
;  153          0539  2     arglist : VECTOR[16];                      ! CMKRNL and output argument list
;  154          0540  2
;  155          0541  2 BIND
;  156          0542  2     csid_vector = buffer[0,0,32,0] : VECTOR [, LONG],
;  157          0543  2     csid_count = buffer[0,0,32,0];             ! Treat first longword as the length field
;  158          0544  2
;  159          0545  2 !
;  160          0546  2 ! Zero the csid count field in the users buffer, and remember the local state of the mount bit
;  161          0547  2 !
;  162          0548  2 csid_count = 0;
;  163          0549  2 scratch[d_v_local_mount] = .$BBLOCK[scratch[d_l_devchar], dev$v_mnt];
;  164          0550  2
;  165          0551  2 !
;  166          0552  2 ! Get the CSID of the local node if necessary
;  167          0553  2 !
;  168          0554  2 IF .local_csid EQL 0
;  169          0555  2 THEN
;  170          0556  3     BEGIN
;  171          0557  3     arglist[0] = (syi$_node_csid^16 OR 4);
;  172          0558  3     arglist[1] = local_csid;
;  173          0559  3     arglist[2] = arglist[3] = 0;
;  174          0560  4     IF NOT (status = $getsyi (itmlst=arglist))
;  175          0561  3     THEN
;  176          0562  3         SIGNAL_STOP (.status);
;  177          0563  2     END;
;  178          0564  2
;  179          0565  2 !
;  180          0566  2 ! Call the kernel mode routine, since the device lock is a kernel lock
;  181          0567  2 !
;  182          0568  2 arglist[0] = 4;                                ! Four arguments
;  183          0569  2 arglist[1] = .scratch;                         ! Device scratch area
;  184          0570  2 arglist[2] = lclbuf;                           ! Local buffer for misc returns
```

```
;  185          0571  2 arglist[3] = %ALLOCATION(lokbuf);        ! Buffer for $GETLKI to place
;  186          0572  2 arglist[4] = lokbuf;                    !  list of lock item blocks
;  187          0573  2
;  188          0574  2 IF NOT (status = $CMKRNL(ROUTIN = get_lock_info, ARGLST = arglist))
;  189          0575  2 THEN
;  190          0576  2     BEGIN
;  191          0577  2     IF .status EQL ss$_accvio
;  192          0578  2     THEN SIGNAL(show$_lockerr, 2, .scratch[d_b_devlen], scratch[d_t_device], .status,
;  193          0579  2                 .kernel_accvio[0], .kernel_accvio[1], .kernel_accvio[2], .kernel_accvio[3], 0)
;  194          0580  2     ELSE SIGNAL(show$_lockerr, 2, .scratch[d_b_devlen], scratch[d_t_device], .status);
;  195          0581  2     RETURN;
;  196          0582  2     END;
;  197          0583  2
;  198          0584  2 !
;  199          0585  2 ! If there are any remote nodes represented, then update the device scratch area and pass the CSID
;  200          0586  2 ! back to the caller.
;  201          0587  2 !
;  202          0588  2 INCR k FROM 0 TO .lclbuf[lcl_ret_length]-1 BY lki$k_length
;  203          0589  2 DO
;  204          0590  2     BEGIN
;  205          0591  2     BIND
;  206          0592  2         lki = lokbuf[.k,0,32,0] : $BBLOCK;
;  207          0593  2         !
;  208          0594  2         ! Only look at non-null locks owned by remote nodes.
;  209          0595  2         !
;  210          0596  2         !       null locks      -  not interesting, most likely just other show device commands
;  211          0597  2         !       local locks     -  we can find far more info from the ucb than from the lock
;  212          0598  2         !
;  213          0599  2         IF  .lki[lki$l_remsysid] NEQ .local_csid
;  214          0600  3           AND
;  215          0601  3             .lki[lki$b_grmode] NEQ lck$k_nlmode
;  216          0602  3         THEN
;  217          0603  4             BEGIN
;  218          0604  4             csid_count = .csid_count + 1;                           ! Bump the count of systems
;  219          0605  4             csid_vector[.csid_count] = .lki[lki$l_remsysid];        ! Copy the CSID into the vector
;  220          0606  4             IF .$BBLOCK[lclbuf[lcl_val_block],dc_notfirst_mnt]      ! If mounted on the remote node
;  221          0607  4             THEN
;  222          0608  5                 BEGIN
;  223          0609  5                 scratch[d_v_remote_mounts] = 1;                     ! Remember that it is mounted elsewhere
;  224          0610  5                 $BBLOCK[scratch[d_l_devchar],dev$v_mnt] = 1;        ! Force the MNT bit on
;  225          0611  5                 scratch[d_w_mcount] = .scratch[d_w_mcount] + 1;     ! Bump the mount count
;  226          0612  5                 $BBLOCK[scratch[d_l_devchar],dev$v_for] =           ! Set the foreign bit if mounted foreign
;  227          0613  5                                 .$BBLOCK[lclbuf[lcl_val_block],dc_foreign];
;  228          0614  5                 $BBLOCK[scratch[d_l_devchar],dev$v_swl] =           ! Set the write-locked bit
;  229          0615  5                                 NOT .$BBLOCK[lclbuf[lcl_val_block],dc_write];
;  230          0616  5                 IF NOT .scratch[d_v_local_mount]                    ! If not mounted locally, then set
;  231          0617  5                 THEN                                                !  a dummy volume name
;  232          0618  5                     CH$MOVE (12, UPLIT BYTE ('(remote mnt)'), scratch[d_t_volnam]);
;  233          0619  4                 END;
;  234          0620  4             IF .lki[lki$b_grmode] EQL lck$k_exmode                  ! If lock mode is exclusive, then the
;  235          0621  4             THEN                                                    !  device is also allocated
;  236          0622  5                 BEGIN
;  237          0623  5                 $BBLOCK[scratch[d_l_devchar],dev$v_all] = 1;        ! Force the ALL bit on
;  238          0624  5                 scratch[d_v_remote_all] = 1;                        ! Set flag for the print routines
;  239          0625  4                 END;
;  240          0626  3             END;
;  241          0627  2     END;
```

```
;  242          0628  2
;  243          0629  2 !
;  244          0630  2 ! Conditionally compile some debugging code.  If compiled with /VARIANT<>0, then if the
;  245          0631  2 ! logical name SHOW$DEBUG is defined we will dump the lock state
;  246          0632  2 !
;  247        L 0633  2 %IF %VARIANT NEQ 0
;  248        U 0634  2 %THEN
;  249        U 0635  2     !
;  250        U 0636  2     ! If the logical name SHOW$DEBUG is defined, dump everything we found
;  251        U 0637  2     !
;  252        U 0638  2     IF (BEGIN
;  253        U 0639  2         LOCAL
;  254        U 0640  2             lcl_buf : VECTOR [256, BYTE],
;  255        U 0641  2             out_dsc : VECTOR [2, LONG];
;  256        U 0642  2         out_dsc [0] = 256;
;  257        U 0643  2         out_dsc [1] = lcl_buf;
;  258        U 0644  2         ($trnlog (lognam=%ASCID 'SHOW$DEBUG', rsllen=out_dsc, rslbuf=out_dsc) EQL ss$_normal)
;  259        U 0645  2         END)
;  260        U 0646  2     THEN
;  261        U 0647  2         BEGIN
;  262        U 0648  2         IF NOT .lclbuf[lcl_enq_status]
;  263        U 0649  2         THEN
;  264        U 0650  2             SIGNAL(show$_lockerr, 2, .scratch[d_b_devlen], scratch[d_t_device], .lclbuf[lcl_enq_status]);
;  265        U 0651  2         show$write_line (%ASCID '    - Lock !XL,  length of items !XL,  $enq status !XL, value block !XL !XL
;  266        U 0652  2                                                 lclbuf[0,0,32,0]);
;  267        U 0653  2         show$write_line (%ASCID '    - Mounted on nodes:  !AF', arglist);
;  268        U 0654  2         show$write_line (%ASCID '    -  Lock id     PID       CSID              Remlkid   Remcsid
;  269        U 0655  2                                 arglist);
;  270        U 0656  2         incr k from 0 to .lclbuf[lcl_ret_length]-1 by lki$k_length
;  271        U 0657  2         do
;  272        U 0658  2             begin
;  273        U 0659  2             local
;  274        U 0660  2                 nodename : VECTOR [16, BYTE],
;  275        U 0661  2                 nodename2 : VECTOR [16, BYTE];
;  276        U 0662  2             bind
;  277        U 0663  2                 lki = lokbuf[.k,0,32,0] : $BBLOCK;
;  278        U 0664  2             arglist[0] = .lki[lki$l_lockid];
;  279        U 0665  2             arglist[1] = .lki[lki$l_pid];
;  280        U 0666  2             arglist[2] = .lki[lki$l_sysid];
;  281        U 0667  2             arglist[3] = 0;
;  282        U 0668  2             arglist[4] = nodename;
;  283        U 0669  2             arglist[5] = .lki[lki$l_remlkid];
;  284        U 0670  2             arglist[6] = .lki[lki$l_remsysid];
;  285        U 0671  2             arglist[7] = 0;
;  286        U 0672  2             arglist[8] = nodename2;
;  287        U 0673  2             arglist[9] = .lki[lki$b_rqmode];
;  288        U 0674  2             arglist[10] = .lki[lki$b_grmode];
;  289        U 0675  2             arglist[11] = .lki[lki$b_queue];
;  290        U 0676  2             IF .lki[lki$l_sysid] NEQ 0
;  291        U 0677  2             THEN
;  292        U 0678  2                 BEGIN
;  293        U 0679  2                 LOCAL
;  294        U 0680  2                     itemlist : VECTOR [4, LONG];
;  295        U 0681  2                 itemlist[0] = (syi$_nodename^16 OR 16);
;  296        U 0682  2                 itemlist[1] = nodename;
;  297        U 0683  2                 itemlist[2] = arglist[3];
;  298        U 0684  2                 itemlist[3] = 0;
```

```
;  299        U 0685  2              $getsyi (csidadr=lki[lki$l_sysid],itmlst=itemlist);
;  300        U 0686  2              END;
;  301        U 0687  2          IF .lki[lki$l_remsysid] NEQ 0
;  302        U 0688  2          THEN
;  303        U 0689  2              BEGIN
;  304        U 0690  2              LOCAL
;  305        U 0691  2                  itemlist : VECTOR [4, LONG];
;  306        U 0692  2              itemlist[0] = (syi$_nodename^16 OR 16);
;  307        U 0693  2              itemlist[1] = nodename2;
;  308        U 0694  2              itemlist[2] = arglist[7];
;  309        U 0695  2              itemlist[3] = 0;
;  310        U 0696  2              $getsyi (csidadr=lki[lki$l_remsysid],itmlst=itemlist);
;  311        U 0697  2              END;
;  312        U 0698  2          show$write_line (
;  313        U 0699  2              %ASCID'    -  !XL  !XL  !XL !8<(!AF)!>  !XL  !XL !8<(!AF)!>    !XB      !XB      !XB', arglis
;  314        U 0700  2          end;
;  315        U 0701  2      !
;  316        U 0702  2      ! Format the buffer, 32 bytes at a time
;  317        U 0703  2      !
;  318        U 0704  2      show$write_line (%ASCID '   - Formatted dump of LKI$_LOCKS buffer:', arglist);
;  319        U 0705  2      incr k from 0 to .lclbuf[lcl_ret_length]-1 by 32
;  320        U 0706  2      do
;  321        U 0707  2          begin
;  322        U 0708  2          !
;  323        U 0709  2          ! Move the next chunk of data to the intermediate buffer
;  324        U 0710  2          !
;  325        U 0711  2          arglist[0]=.lokbuf[.k+28,0,32,0];
;  326        U 0712  2          arglist[1]=.lokbuf[.k+24,0,32,0];
;  327        U 0713  2          arglist[2]=.lokbuf[.k+20,0,32,0];
;  328        U 0714  2          arglist[3]=.lokbuf[.k+16,0,32,0];
;  329        U 0715  2          arglist[4]=.lokbuf[.k+12,0,32,0];
;  330        U 0716  2          arglist[5]=.lokbuf[.k+8,0,32,0];
;  331        U 0717  2          arglist[6]=.lokbuf[.k+4,0,32,0];
;  332        U 0718  2          arglist[7]=.lokbuf[.k,0,32,0];
;  333        U 0719  2          arglist[8]=32;
;  334        U 0720  2          arglist[9]=lokbuf[.k,0,32,0];
;  335        U 0721  2          arglist[10]=.k;
;  336        U 0722  2          show$write_line (%ASCID '   -  !8(9XL) !32AF !XW', arglist);
;  337        U 0723  2          end;
;  338        U 0724  2      END;
;  339          0725  2 %FI          ! End of variant for debug listing
;  340          0726  2
;  341          0727  2 RETURN;
;  342          0728  1 END;


                                                        .TITLE  SHODEVCLU
                                                        .IDENT  \V04-000\

                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

          29  74  6E  6D  20  65  74  6F  6D  65  72  28  00000 P.AAA:  .ASCII  \(remote mnt)\                    ;

                                                        .PSECT  $OWN$,NOEXE,2

                                              00000000  00000 LOCAL_CSID:
                                                        .LONG   0                                               ;
```

```
                                                        .EXTRN  SHOW$WRITE_LINE
                                                        .EXTRN  SHOW$_LOCKERR, LCK$GL_MAXID
                                                        .EXTRN  LCK$GL_IDTBL, KERNEL_ACCVIO
                                                        .EXTRN  SYS$GETSYI, SYS$CMKRNL

                                                        .PSECT  $CODE$,NOWRT,2

                                        0FFC 00000      .ENTRY  SCAN_CLUSTER_LOCKS, Save R2,R3,R4,R5,R6,R7,-; 0514
                                                                R8,R9,R10,R11
                                5E      FAF4   CE  9E 00002     MOVAB   -1292(SP), SP
                                5A        08   AC  D0 00007     MOVL    BUFFER, R10                           ; 0542
                                             6A  D4 0000B       CLRL    (R10)                                 ; 0548
                                57        04   AC  D0 0000D     MOVL    SCRATCH, R7                           ; 0549
                                59        70   A7  9E 00011     MOVAB   112(R7), R9
        50              69              01      13  EF 00015     EXTZV   #19, #1, (R9), R0
    04  A7              01              06      50  F0 0001A     INSV    R0, #6, #1, 4(R7)
                                      0000'     CF  D5 00020     TSTL    LOCAL_CSID                            ; 0554
                                                31  12 00024     BNEQ    1$
                                6E 10D00004     8F  D0 00026     MOVL    #282066948, ARGLIST                   ; 0557
                          04    AE      0000'   CF  9E 0002D     MOVAB   LOCAL_CSID, ARGLIST+4                 ; 0558
                                          08    AE  7C 00033     CLRQ    ARGLIST+8                             ; 0559
                                                7E  7C 00036     CLRQ    -(SP)                                 ; 0560
                                                7E  D4 00038     CLRL    -(SP)
                                          0C    AE  9F 0003A     PUSHAB  ARGLIST
                                                7E  7C 0003D     CLRQ    -(SP)
                                                7E  D4 0003F     CLRL    -(SP)
                    00000000G   00              07  FB 00041     CALLS   #7, SYS$GETSYI
                                52              50  D0 00048     MOVL    R0, STATUS
                                09              52  E8 0004B     BLBS    STATUS, 1$
                                                52  DD 0004E     PUSHL   STATUS                                ; 0562
                    00000000G   00              01  FB 00050     CALLS   #1, LIB$STOP
                                6E              04  D0 00057 1$: MOVL    #4, ARGLIST                           ; 0568
                          04    AE              57  D0 0005A     MOVL    R7, ARGLIST+4                         ; 0569
                          08    AE        E4    AD  9E 0005E     MOVAB   LCLBUF, ARGLIST+8                     ; 0570
                          0C    AE      04B0    8F  3C 00063     MOVZWL  #1200, ARGLIST+12                     ; 0571
                          10    AE        40    AE  9E 00069     MOVAB   LOKBUF, ARGLIST+16                    ; 0572
                                                5E  DD 0006E     PUSHL   SP                                    ; 0574
                                      0000V     CF  9F 00070     PUSHAB  GET_LOCK_INFO
                    00000000G   00              02  FB 00074     CALLS   #2, SYS$CMKRNL
                                52              50  D0 0007B     MOVL    R0, STATUS
                                45              52  E8 0007E     BLBS    STATUS, 3$
                                50        08    A7  9E 00081     MOVAB   8(R7), R0                             ; 0578
                                0C              52  D1 00085     CMPL    STATUS, #12                           ; 0577
                                                26  12 00088     BNEQ    2$
                                                7E  D4 0008A     CLRL    -(SP)                                 ; 0578
                                7E  00000000G   00  7D 0008C     MOVQ    KERNEL_ACCVIO+8, -(SP)                ; 0579
                                7E  00000000G   00  7D 00093     MOVQ    KERNEL_ACCVIO, -(SP)
                                                05  BB 0009A     PUSHR   #^M<R0,R2>                            ; 0578
                                7E        06    A7  9A 0009C     MOVZBL  6(R7), -(SP)
                                                02  DD 000A0     PUSHL   #2
                                    00000000G   8F  DD 000A2     PUSHL   #SHOW$_LOCKERR
                    00000000G   00              0A  FB 000A8     CALLS   #10, LIB$SIGNAL
                                                04  000AF     RET
                                                05  BB 000B0 2$: PUSHR   #^M<R0,R2>                            ; 0580
                                7E        06    A7  9A 000B2     MOVZBL  6(R7), -(SP)
                                                02  DD 000B6     PUSHL   #2
```

```
                              00000000G  8F   DD 000B8        PUSHL   #SHOW$_LOCKERR                  ;
                  00000000G  00          05   FB 000BE        CALLS   #5, LIB$SIGNAL                  ;
                                              04 000C5        RET                                     ; 0576
                         5B        E8   AD    3C 000C6 3$:    MOVZWL  LCLBUF+4, R11                   ; 0588
                                        5B    D7 000CA        DECL    R11                             ;
                         58             18    CE 000CC        MNEGL   #24, K                          ; 0599
                                        63    11 000CF        BRB     6$                              ;
                         56        40 AE48    9E 000D1 4$:    MOVAB   LOKBUF[K], R6                   ; 0592
                   0000'  CF        14   A6   D1 000D6        CMPL    20(R6), LOCAL_CSID              ; 0599
                                        56    13 000DC        BEQL    6$                              ;
                                   0D   A6    95 000DE        TSTB    13(R6)                          ; 0601
                                        51    13 000E1        BEQL    6$                              ;
                                        6A    D6 000E3        INCL    (R10)                           ; 0604
                         50             6A    D0 000E5        MOVL    (R10), R0                       ; 0605
                       6A40        14   A6    D0 000E8        MOVL    20(R6), (R10)[R0]               ;
                         34        F0   AD    E9 000ED        BLBC    LCLBUF+12, 5$                   ; 0606
                    04   A7        80   8F    88 000F1        BISB2   #128, 4(R7)                     ; 0609
                    02   A9             08    88 000F6        BISB2   #8, 2(R9)                       ; 0610
                              00CC      C7    B6 000FA        INCW    204(R7)                         ; 0611
      50      F0  AD             01     01    EF 000FE        EXTZV   #1, #1, LCLBUF+12, R0           ; 0613
   03 A9          01             00     50    F0 00104        INSV    R0, #0, #1, 3(R9)               ;
      50      F0  AD             01     04    EF 0010A        EXTZV   #4, #1, LCLBUF+12, R0           ; 0615
                                 50     50    D2 00110        MCOML   R0, R0                          ;
      69          01             19     50    F0 00113        INSV    R0, #25, #1, (R9)               ;
                  08        04   A7     06    E0 00118        BBS     #6, 4(R7), 5$                   ; 0616
           00B8   C7     0000'   CF     0C    28 0011D        MOVC3   #12, P.AAA, 184(R7)             ; 0618
                                 05  0D A6    91 00125 5$:    CMPB    13(R6), #5                      ; 0620
                                        09    12 00129        BNEQ    6$                              ;
                         02   A9   80   8F    88 0012B        BISB2   #128, 2(R9)                     ; 0623
                         05   A7        01    88 00130        BISB2   #1, 5(R7)                       ; 0624
    FF97          58             18     5B    F1 00134 6$:    ACBL    R11, #24, K, 4$                 ; 0588
                                              04 0013A        RET                                     ; 0728

; Routine Size:  315 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  344          0729  1 GLOBAL ROUTINE get_lock_info (scratch : REF $BBLOCK, lclbuf : REF $BBLOCK, lokbuf_size, lokbuf : REF $BBLOCK
;  345          0730  2 BEGIN
;  346          0731  2
;  347          0732  2 !---
;  348          0733  2 !
;  349          0734  2 ! This routine is called in KERNEL mode to scan the device lock data base and
;  350          0735  2 ! determine any cluster-wide information which is available.
;  351          0736  2 !
;  352          0737  2 ! Inputs
;  353          0738  2 !       SCRATCH     - address of scratch data for this device
;  354          0739  2 !       LOKBUF_SIZE - size of lock info buffer
;  355          0740  2 !       LOKBUF      - lock info buffer for the $GETLKI call, passed in so that we
;  356          0741  2 !                     don't have kernel stack restrictions on the size of the buffer
;  357          0742  2 !
;  358          0743  2 ! Outputs
;  359          0744  2 !       SCRATCH  - some values "adjusted" for cluster-wide information
;  360          0745  2 !       LCLBUF   - output lock information for control and debug listings
;  361          0746  2 !       LOKBUF   - lock info vector
;  362          0747  2 !
;  363          0748  2 !---
;  364          0749  2
;  365          0750  2 LOCAL
;  366          0751  2     iosb : $BBLOCK [8],
;  367          0752  2     itemlist : $BBLOCK [16],
;  368          0753  2     lokbuf_len : $BBLOCK [4],
;  369          0754  2     lksb : $BBLOCK [24],                        ! status block + value block
;  370          0755  2     name : VECTOR [20, BYTE],
;  371          0756  2     name_desc : VECTOR [2, LONG],
;  372          0757  2     status;
;  373          0758  2
;  374          0759  2 !
;  375          0760  2 ! Trap anything weird, and turn it into a return
;  376          0761  2 !
;  377          0762  2 ENABLE
;  378          0763  2     get_lock_info_handler;
;  379          0764  2
;  380          0765  2 !
;  381          0766  2 ! Get a null-mode lock on the device name
;  382          0767  2 !
;  383          0768  2 (name[0]) = 'SYS$';                            ! Sys prefix on the name
;  384          0769  2 CH$MOVE (.scratch[d_b_devlen], scratch[d_t_device], name[4]);
;  385          0770  2 name_desc[0] = 4 + .scratch[d_b_devlen];
;  386          0771  2 name_desc[1] = name;
;  387        P 0772  2 status = $ENQW (efn=0,
;  388        P 0773  2                 lkmode=LCK$K_NLMODE,
;  389        P 0774  2                 lksb=lksb,
;  390        P 0775  2                 flags=(LCK$M_NOQUEUE OR LCK$M_VALBLK OR LCK$M_SYNCSTS OR LCK$M_SYSTEM),
;  391        P 0776  2                 resnam=name_desc,
;  392          0777  2                 acmode=0);
;  393          0778  2 lclbuf[lcl_null_lkid]=.lksb[4,0,32,0];          ! Stick the null lock id into the buffer
;  394          0779  2 CH$MOVE (16,lksb[8,0,32,0],lclbuf[lcl_val_block]);! Copy the value block
;  395          0780  2 IF .status                                      ! If the enqueue worked then check the stat block
;  396          0781  2 THEN status = .lksb[0,0,16,0];
;  397          0782  2 lclbuf[lcl_enq_status] = .status;              ! Save $enq status
;  398          0783  2 IF NOT .status
;  399          0784  2 THEN
;  400          0785  2   IF .status NEQ ss$_valnotvalid               ! Cope with a value block not being valid
```

```
;  401          0786  2   THEN
;  402          0787  3     BEGIN
;  403          0788  3     $DEQ (lkid=.lksb[4,0,32,0]);                ! Release the lock, just in case one was granted
;  404          0789  3     RETURN .status;                             ! Return with status
;  405          0790  2     END;
;  406          0791  2
;  407          0792  2 !
;  408          0793  2 ! Get lock information, and find out all the other locks on this resource name
;  409          0794  2 !
;  410          0795  2 itemlist[0,0,16,0] = .lokbuf_size;               ! Size of scratch buffer
;  411          0796  2 itemlist[2,0,16,0] = LKI$_LOCKS;                 ! Find out which other locks
;  412          0797  2 itemlist[4,0,32,0] = .lokbuf;                    ! Address of scratch buffer
;  413          0798  2 itemlist[8,0,32,0] = lokbuf_len;                 ! Address for returned length
;  414          0799  2 itemlist[12,0,32,0] = 0;                         ! End of itemlist
;  415          0800  2 lokbuf_len = 0;                                  ! Zero returned length (?? bogus when GETLKI works)
;  416        P 0801  2 status = $GETLKI (efn=0,
;  417        P 0802  2                   lkidadr=lksb[4,0,32,0],
;  418        P 0803  2                   itmlst=itemlist,
;  419          0804  2                   iosb=iosb);
;  420          0805  2 lclbuf[lcl_lengths] = .lokbuf_len[0,0,32,0];     ! Save both length fields
;  421          0806  2 IF .status                                       ! If the getlki worked then check the stat block
;  422          0807  2 THEN status = .iosb[0,0,16,0];
;  423          0808  2
;  424          0809  2 !
;  425          0810  2 ! Release the null lock
;  426          0811  2 !
;  427          0812  2 $DEQ (lkid=.lksb[4,0,32,0]);
;  428          0813  2
;  429          0814  2 RETURN .status;                                  ! Return with status
;  430          0815  1 END;


                                                    .EXTRN  SYS$ENQW, SYS$DEQ
                                                    .EXTRN  SYS$GETLKI

                                        00FC 00000  .ENTRY  GET_LOCK_INFO, Save R2,R3,R4,R5,R6,R7       ; 0729
                              5E      B0 AE  9E 00002  MOVAB  -80(SP), SP                               ;
                              6D    00B3 CF  DE 00006  MOVAL  4$, (FP)                                  ; 0730
                        0C    AE 24535953 8F  D0 0000B  MOVL   #609442131, NAME                         ; 0768
                              56      04 AC  D0 00013  MOVL   SCRATCH, R6                               ; 0769
                              50      06 A6  9A 00017  MOVZBL 6(R6), R0                                 ;
                10    AE      08 A6      50  28 0001B  MOVC3  R0, 8(R6), NAME+4                        ;
                        04    AE      06 A6  9A 00021  MOVZBL 6(R6), NAME_DESC                          ; 0770
                        04    AE         04  C0 00026  ADDL2  #4, NAME_DESC                             ;
                        08    AE      0C AE  9E 0002A  MOVAB  NAME, NAME_DESC+4                         ; 0771
                                         7E  7C 0002F  CLRQ   -(SP)                                     ; 0777
                                         7E  7C 00031  CLRQ   -(SP)                                     ;
                                         7E  7C 00033  CLRQ   -(SP)                                     ;
                                      1C AE  9F 00035  PUSHAB NAME_DESC                                 ;
                                         1D  DD 00038  PUSHL  #29                                       ;
                                      40 AE  9F 0003A  PUSHAB LKSB                                      ;
                                         7E  7C 0003D  CLRQ   -(SP)                                     ;
                00000000G             00   0B  FB 0003F  CALLS  #11, SYS$ENQW                         ;
                              57        50  D0 00046  MOVL   R0, STATUS                                 ;
                              56      08 AC  D0 00049  MOVL   LCLBUF, R6                                ; 0778
                              66      24 AE  D0 0004D  MOVL   LKSB+4, (R6)                              ;
```

```
                0C  A6      28  AE          10  28 00051          MOVC3   #16, LKSB+8, 12(R6)             ; 0779
                                04          57  E9 00057          BLBC    STATUS, 1$                      ; 0780
                                57      20  AE  3C 0005A          MOVZWL  LKSB, STATUS                    ; 0781
                                08      A6  57  D0 0005E 1$:      MOVL    STATUS, 8(R6)                   ; 0782
                                09          57  E8 00062          BLBS    STATUS, 2$                      ; 0783
                          000009F0      8F  57  D1 00065          CMPL    STATUS, #2544                   ; 0785
                                            3D  12 0006C          BNEQ    3$                              ;
                                38      AE      AC  B0 0006E 2$:  MOVW    LOKBUF_SIZE, ITEMLIST           ; 0795
                                3A      AE  0208  8F  B0 00073    MOVW    #520, ITEMLIST+2                ; 0796
                                3C      AE    10  AC  D0 00079    MOVL    LOKBUF, ITEMLIST+4              ; 0797
                                40      AE        6E  9E 0007E    MOVAB   LOKBUF_LEN, ITEMLIST+8          ; 0798
                                            44  AE  D4 00082      CLRL    ITEMLIST+12                     ; 0799
                                            6E  D4 00085          CLRL    LOKBUF_LEN                      ; 0800
                                            7E  7C 00087          CLRQ    -(SP)                           ; 0804
                                            7E  D4 00089          CLRL    -(SP)                           ;
                                            54  AE 9F 0008B       PUSHAB  IOSB                            ;
                                            48  AE 9F 0008E       PUSHAB  ITEMLIST                        ;
                                            38  AE 9F 00091       PUSHAB  LKSB+4                          ;
                                            7E  D4 00094          CLRL    -(SP)                           ;
                          00000000G     00      07  FB 00096     CALLS   #7, SYS$GETLKI                  ;
                                        57      50  D0 0009D     MOVL    R0, STATUS                      ;
                                04      A6      6E  D0 000A0     MOVL    LOKBUF_LEN, 4(R6)               ; 0805
                                        04      57  E9 000A4     BLBC    STATUS, 3$                      ; 0806
                                        57  48  AE  3C 000A7     MOVZWL  IOSB, STATUS                    ; 0807
                                            7E  7C 000AB 3$:      CLRQ    -(SP)                           ; 0812
                                            7E  D4 000AD          CLRL    -(SP)                           ;
                                            30  AE  DD 000AF      PUSHL   LKSB+4                          ;
                          00000000G     00      04  FB 000B2     CALLS   #4, SYS$DEQ                     ;
                                        50      57  D0 000B9     MOVL    STATUS, R0                      ; 0814
                                                04  000BC       RET                                     ; 0815
                                              0000  000BD 4$:     .WORD   Save nothing                    ; 0730
                                            7E  D4 000BF          CLRL    -(SP)                           ;
                                            5E  DD 000C1          PUSHL   SP                              ;
                                    7E          04  AC  7D 000C3  MOVQ    4(AP), -(SP)                    ;
                              0000V CF              03  FB 000C7  CALLS   #3, GET_LOCK_INFO_HANDLER       ;
                                                    04  000CC    RET                                     ;

; Routine Size:  205 bytes,    Routine Base:  $CODE$ + 013B
```

```
1 GLOBAL ROUTINE get_lock_info_handler (sig : REF BLOCK[,BYTE], mech : REF BLOCK[,BYTE]) =
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       This routine intercepts kernel mode signals and converts ACCVIOs to returns
2 !
2 ! INPUTS:
2 !
2 !       sig  - signal argument list
2 !       mech - mechanism argument list
2 !
2 ! SIDE EFFECTS:
2 !
2 !       A return is made to user mode code.
2 !--
2
2 EXTERNAL ROUTINE
2     LIB$SIG_TO_RET : ADDRESSING_MODE (GENERAL);
2
2 ! If the signal name is an accvio, then clean up
2 !
2 IF .sig [chf$l_sig_name] EQL ss$_accvio         ! Is it an accvio?
2 THEN
3     BEGIN
3     CH$MOVE (4*4, sig[chf$l_sig_arg1], kernel_accvio[0]);
3     RETURN LIB$SIG_TO_RET (.sig, .mech);         ! Convert signal to return
2     END;
2
2 RETURN ss$_resignal;
1 END;
```

```
                                                           .EXTRN  LIB$SIG_TO_RET

                                          007C 00000       .ENTRY  GET_LOCK_INFO_HANDLER, Save R2,R3,R4,R5,R6   ; 0816
                        56       04   AC   D0 00002       MOVL    SIG, R6                                      ; 0839
                        0C       04   A6   D1 00006       CMPL    4(R6), #12                                   ;
                                      16   12 0000A       BNEQ    1$                                           ;
        00000000G  00       08   A6        10   28 0000C       MOVC3   #16, 8(R6), KERNEL_ACCVIO                    ; 0842
                                 08   AC   DD 00015       PUSHL   MECH                                         ; 0843
                                      56   DD 00018       PUSHL   R6                                           ;
                   00000000G  00      02   FB 0001A       CALLS   #2, LIB$SIG_TO_RET                           ;
                                      04 00021            RET                                                  ;
                        50     0918   8F   3C 00022 1$:   MOVZWL  #2328, R0                                    ; 0846
                                      04 00027            RET                                                  ; 0847

; Routine Size:  40 bytes,    Routine Base:  $CODE$ + 0208
```

```
;  465          0848  1 END
;  466          0849  0 ELUDOM


                                                        .EXTRN  LIB$SIGNAL, LIB$STOP

;                                PSECT SUMMARY
;
;      Name                      Bytes                         Attributes
;
;  $OWN$                           4  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                         12  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                        560  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        40         0        1000        00:01.9


;                                    COMMAND QUALIFIERS

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

; Size:          560 code + 16 data bytes
; Run Time:        00:22.9
; Elapsed Time:    01:13.1
; Lines/CPU Min:    2227
; Lexemes/CPU-Min: 57993
; Memory Used:  161 pages
; Compilation Complete
```

SHODEVPRT
LIS

SHODEVUTL
LIS

SHODEV
LIS

SHODEVCLU
LIS