```
   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**SHOWLOG

```
 SSSSSSSS   HH      HH     OOOOOO    WW      WW  LL              OOOOOO      GGGGGGGG
 SSSSSSSS   HH      HH     OOOOOO    WW      WW  LL              OOOOOO      GGGGGGGG
SS          HH      HH   OO      OO  WW      WW  LL            OO      OO  GG
SS          HH      HH   OO      OO  WW      WW  LL            OO      OO  GG
SS          HH      HH   OO      OO  WW      WW  LL            OO      OO  GG
SS          HH      HH   OO      OO  WW      WW  LL            OO      OO  GG
  SSSSSS    HHHHHHHHHH   OO      OO  WW      WW  LL            OO      OO  GG
  SSSSSS    HHHHHHHHHH   OO      OO  WW      WW  LL            OO      OO  GG
        SS  HH      HH   OO      OO  WW  WW  WW  LL            OO      OO  GG  GGGGGG
        SS  HH      HH   OO      OO  WW  WW  WW  LL            OO      OO  GG  GGGGGG
        SS  HH      HH   OO      OO  WWWW  WWWW  LL            OO      OO  GG      GG      ....
        SS  HH      HH   OO      OO  WWWW  WWWW  LL            OO      OO  GG      GG      ....
SSSSSSSS    HH      HH     OOOOOO    WW      WW  LLLLLLLLLL      OOOOOO      GGGGGG        ....
SSSSSSSS    HH      HH     OOOOOO    WW      WW  LLLLLLLLLL      OOOOOO      GGGGGG        ....

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

```
0 module showlog (ident = 'V04-000',       ! File: SHOWLOG.B32 Edit: RAS0319
0         addressing_mode (external = general))=
1 begin
1 !
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1 !
1 !++
1 !
1 ! FACILITY:  Show logical names
1 !
1 ! ABSTRACT:
1 !       This module implements the "SHOW LOGICAL" DCL command.
1 !
1 ! ENVIRONMENT:
1 !       VAX native, user and kernel mode
1 !
1 ! AUTHOR:  Peter D Gilbert        CREATION DATE:  20-Dec-1983
1 !
1 ! MODIFIED BY:
1 !
1 !       V03-004 RAS0328         Ron Schaefer                    25-Jul-1984
1 !               Make SHOW LOGICAL more friendly in terms of space allocated;
1 !               instead of asking for PAGEDYN, ask for the minimum of
1 !               PAGEDYN and process free-space as determined by $GETJPI.
1 !               Fix possible kernel-mode accvio on error path.
1 !
1 !       V03-003 RAS0319         Ron Schaefer                    2-Jul-1984
1 !               Add additional data structure cell to the RT structure.
1 !               Since RT and NT are actually used in 2 facilities, they
1 !               should really be globally defined.
1 !
1 !       V03-002 PDG0002         Peter D Gilbert                 08-Apr-1984
1 !
1 !               Add LOG_V_EQUAL, which is set in CMP_LOG, and indicates there
```

```
1 !                     are two or more identical logical names in the same table.
1 !                     This flag indicates that, even if no other attributes are
1 !                     displayed, the access mode should be displayed to distinguish
1 !                     between these logical names.
1 !
1 !                     Change DUMP_LOG so a depth of 0 is shown as " ", instead of "0".
1 !
1 !                     For recursive translations, avoid writing a blank line until we
1 !                     know it's needed to separate logical names (when NO names were
1 !                     found, the extra blank line looked very tacky).
1 !
1 !             V03-001 TMK0001         Todd M. Katz                    04-Apr-1984
1 !                     Make the following changes:
1 !
1 !                     1. I have changed this module to use the symbol LNM$C_MAXDEPTH
1 !                        to define the maximum logical name recursion depth instead
1 !                        of the local symbol RT_C_MAXDEP.
1 !
1 !                     2. The name of the offset in the table header that contains the
1 !                        address of the protection structure associated with shareable
1 !                        logical name tables has been changed from LNMTH$L_CHP to
1 !                        LNMTH$L_ORB to reflect the replacement of shareable logical
1 !                        name tables' CHIP protection templates by Object Rights
1 !                        Blocks.
1 !
1 !                     3. Change the routine recursively_translate, so that it
1 !                        immediately exits whenever it is called with a depth equal
1 !                        to LNM$C_MAXDEPTH. Previously, this routine was exiting only
1 !                        when it was called with a depth greater-than LNM$C_MAXDEPTH.
1 !                        However, since recursion depths are numbered from 0 to 9 and
1 !                        not from 1 to 10, this was resulting in names being
1 !                        recursively translated to a depth of eleven instead of to a
1 !                        depth of ten.
1 !
1 !--
1
1 library 'sys$library:lib';                    ! VAX/VMS system definitions
1 library 'sys$library:xport';
1 %if %declared(%quote $descriptor) %then undeclare %quote $descriptor; %fi
1 require 'src$:showdef';                       ! SHOW common definitions
1
1
1 !                           B L I S S   E X T E N S I O N S
1 !
1 macro
1     $check_(x) = (local s: initial(x); if not .s then signal_stop(.s)) %,
1     !
1     ! Macros for offset, position, size, and extension.
1     !
1     base_ = 0,0,0,0 %,
1     b0_ = 0,8,0 %,
1     b1_ = 8,8,0 %,
1     l_ = 0,32,0 %,
1     !
1     ! This macro is similar to the Bliss %fieldexpand lexical function.
1     ! However, it works with either field names, or o,p,s,e macros.
1     !
```

```
;  115      M 0213  1     %fieldexpand_(a,b,c,d,e) =
;  116      M 0214  1         %if %length eql 1 %then %fieldexpand(a) %else
;  117      M 0215  1         %if %length eql 2 %then %fieldexpand(a,b) %else
;  118      M 0216  1         %if %length eql 4 %then a,b,c,d %else
;  119      M 0217  1         %if %length eql 5 %then
;  120      M 0218  1             %if e eql 0 %then a %else
;  121      M 0219  1             %if e eql 1 %then b %else
;  122      M 0220  1             %if e eql 2 %then c %else
;  123      M 0221  1             %if e eql 3 %then d %else
;  124      M 0222  1             %error('Invalid parameter to %fieldexpand_')
;  125      M 0223  1             %fi %fi %fi %fi %else
;  126      M 0224  1         %error('Wrong number of arguments to %fieldexpand_')
;  127        0225  1         %fi %fi %fi %fi %;
```

```
!
! BLOCK TO CONTROL RECURSIVE TABLE NAME TRANSLATION
!
!           31            25 24            16             9 8 7               0
!           +---------------+---------------+---------------+-----------------+
! R5 ->     ! TRIES         ! DEPTH         !            T C! ACCESS MODE     !
!           +---------------+---------------+---------------+-----------------+
!           !               CACHE TABLE ADDRESS OR 0                          !
!           +---------------+---------------+---------------+-----------------+
!           !                                                                 !
!           !      LNM$C_MAXDEPTH LONGWORDS TO SERVE AS STACK                 !
!           !                                                                 !
!           !                                                                 !
!           +---------------+---------------+---------------+-----------------+
!
!           T IS SET TO INHIBIT FURTHER RECURSION
!
literal
    rt_c_maxtries=      255;                ! Max number of tries
$unit_field rt_fields =
    set
    rt_w_r5=            [$bits(16)],        ! Access mode and flags
    $overlay(rt_w_r5)
    rt_b_acmode=        [$bits(8)],         ! Access mode
    rt_b_flags=         [$bits(8)],         ! Flags
    $continue
    $overlay(rt_b_flags)
    rt_v_case=          [$bit],             ! Caseless flag
    rt_v_term=          [$bit],             ! Inhibit recursion flag
    $continue
    rt_b_depth=         [$bits(8)],         ! Recursion depth
    rt_b_tries=         [$bits(8)],         ! Recursion tries
    rt_l_cacheptr=      [$address],         ! Address of cache entry
    rt_a_stack=         [$bits(0)]          ! Start of recursion stack
    tes;
literal rt_k_size       = $field_set_units + lnm$c_maxdepth * %upval;
macro   rt_block        = $unit_block(rt_k_size) field(rt_fields) %;


!
! BLOCK TO CONTROL NAME TRANSLATION
!
!           31            25 24            16 15            8 7               0
!           +---------------+---------------+---------------+-----------------+
! R3 ->     ! HASH FUNCTION VALUE OR 0      !              C! ACCESS MODE     !
!           +---------------+---------------+---------------+-----------------+
!           !                       LENGTH OF NAME                            !
!           +---------------+---------------+---------------+-----------------+
!           !                       ADDRESS OF NAME                           !
!           +---------------+---------------+---------------+-----------------+
!           !               TABLE HEADER ADDRESS OR 0                         !
!           +---------------+---------------+---------------+-----------------+
!           !               NAME BLOCK ADDRESS OR 0                           !
!           +---------------+---------------+---------------+-----------------+
!
!           F IS SET TO INHIBIT FURTHER RECURSION
!
```

```
1 $unit_field nt_fields =
1       set
1       nt_w_r5=             [$bits(16)],      ! Access mode and flags
1       $overlay(nt_w_r5)
1       nt_b_acmode=         [$bits(8)],       ! Access mode
1       nt_b_flags=          [$bits(8)],       ! Flags
1       $continue
1       $overlay(nt_b_flags)
1       nt_v_case=           [$bit],           ! Caseless flag
1       $continue
1       nt_w_hash=           [$bits(16)],      ! Hash function or 0
1       nt_l_namlen=         [$integer],       ! Length of name
1       nt_l_namadr=         [$address],       ! Address of name
1       nt_l_tabid=          [$address],       ! Table header address or 0
1       nt_l_thread=         [$address]        ! Lnmb pointer
1       tes;
1 literal nt_k_size          = $field_set_units;
1 macro   nt_block           = $unit_block(nt_k_size) field(nt_fields) %;
```

```
1 ! Define linkages for the routines to lock and unlock the logical name
1 ! table mutex, as well as the match routine and the protection checking
1 ! routine.
1 !
1 linkage
1     lnk_presearch = jsb (          ! R0 = RETURN STATUS
1         register=1,       ! R1 = HASH TABLE ADDRESS (IF TABLE HEADER ADDRESS IS MISSING)
1         register=3;       ! R3 = ADDRESS OF NAME TRANSLATION (NT) BLOCK
1         register=1       ! R1 = ADDRESS OF LOGICAL NAME BLOCK THAT CONTAINS MATCH.
1         ): nopreserve(0,1,2) preserve(3,4,5) notused(6,7,8,9,10,11);
1 linkage
1     lnk_contsearch = jsb (         ! R0 = RETURN STATUS
1         register=3;       ! R3 = ADDRESS OF NAME TRANSLATION (NT) BLOCK
1         register=1       ! R1 = ADDRESS OF LOGICAL NAME BLOCK THAT CONTAINS MATCH.
1         ): nopreserve(0,1,2) preserve(3,4,5) notused(6,7,8,9,10,11);
1 linkage
1     lnk_check_prot = jsb (         ! R0 = RETURN STATUS
1         register=1,       ! R1 = ADDRESS OF LOGICAL NAME TABLE HEADER.
1         register=2,       ! R2 = ACCESS MASK TO USE FOR THE CHECK.
1         register=4,       ! R4 = CURRENT PCB ADDR
1         register=5       ! R5 = ACCESS MODE OF ACCESSOR
1         ): nopreserve(2) preserve(1,3,4,11) notused(6,7,8,9,10);
1 linkage
1     lnk_hash = jsb (     ! R0 = RETURNED HASH VALUE
1         register=0,       ! R0 = LENGTH OF LOGICAL NAME STRING
1         register=1       ! R1 = ADDRESS OF LOGICAL NAME STRING
1         ): nopreserve(2,3) preserve(4,5) notused(6,7,8,9,10,11);
1 linkage
1     lnk_setup = jsb (
1         register=2,       ! R2 = LENGTH OF TABLE NAME STRING
1         register=3,       ! R3 = ADDRESS OF TABLE NAME STRING
1         register=5;       ! R5 = ADDRESS OF RECURSION TABLE SEARCH CONTROL BLOCK
1         register=1       ! R1 = ADDRESS OF LOGICAL NAME TABLE HEADER
1         ): nopreserve(2,3) preserve(4,5) notused(6,7,8,9,10,11);
1 linkage
1     lnk_table = jsb (
1         register=5;       ! R5 = ADDRESS OF RECURSION TABLE SEARCH CONTROL BLOCK
1         register=1       ! R1 = ADDRESS OF LOGICAL NAME TABLE HEADER
1         ): nopreserve(2,3) preserve(4,5) notused(6,7,8,9,10,11);
1 linkage
1     lnk_lock = jsb (
1         register=4       ! R4 = CURRENT PCB ADDR
1         ): nopreserve(1,2,3) notused(5,6,7,8,9,10,11);
1 linkage
1     lnk_match_name = jsb (
1         register=2,       ! R2 = LENGTH OF STRING TO TRY MATCHING
1         register=3,       ! R3 = ADDRESS OF STRING TO TRY MATCHING
1         register=4,       ! R4 = LENGTH OF WILDCARD STRING
1         register=5       ! R5 = ADDRESS OF WILDCARD STRING
1         ): nopreserve(1,2,3,4,5) notused(6,7,8,9,10,11);
1 linkage
1     lnk_compare = jsb (
1         register=0,
1         register=1
1         ): nopreserve(0,1,2,3) preserve(4,5) notused(6,7,8,9,10,11);
1
1 ! Some linkages that we don't use
```

```
;   1 !
;   1 !linkage
;   1 !     lnk_searchlog = jsb (         ! R0 = RETURN STATUS
;   1 !         register=0,      ! R0 = LENGTH OF LOGICAL NAME STRING
;   1 !         register=1,      ! R1 = ADDRESS OF LOGICAL NAME STRING
;   1 !         register=2,      ! R2 = LENGTH OF TABLE NAME STRING
;   1 !         register=3,      ! R3 = ADDRESS OF TABLE NAME STRING
;   1 !         register=5;      ! R5 = SEARCH ACCESS MODE IN LOW BYTE, CASELESS FLAG IN BIT 8.
;   1 !         register=1       ! R1 = ADDRESS OF LOGICAL NAME BLOCK THAT CONTAINS MATCH.
;   1 !         ): preserve(2,3,4,5) notused(6,7,8,9,10,11);
;   1 !linkage
;   1 !     lnk_firsttab = jsb (          ! R0 = RETURN STATUS
;   1 !         register=1,      ! R1 = SEARCH ACCESS MODE IN LOW BYTE, CASELESS FLAG IN BIT 8.
;   1 !         register=2,      ! R2 = LENGTH OF TABLE NAME STRING
;   1 !         register=3;      ! R3 = ADDRESS OF TABLE NAME STRING
;   1 !         register=1       ! R1 = ADDRESS OF LOGICAL NAME TABLE HEADER
;   1 !         ): nopreserve(2,3) preserve(4,5) notused(6,7,8,9,10,11);
;   1 !linkage
;   1 !     lnk_lookup = jsb (
;   1 !         register=0,      ! R0 = HASH CODE TABLE NAME STRING (OR 0 IF NOT KNOWN)
;   1 !         register=2,      ! R2 = LENGTH OF TABLE NAME STRING
;   1 !         register=3,      ! R3 = ADDRESS OF TABLE NAME STRING
;   1 !         register=5;      ! R5 = ADDRESS OF RECURSION TABLE SEARCH CONTROL BLOCK
;   1 !         register=1       ! R1 = ADDRESS OF LOGICAL NAME BLOCK
;   1 !         ): nopreserve(2,3) preserve(5) notused(4,6,7,8,9,10,11);
```

```
1 ! External declarations
1 !
1 external routine
1     cli$present,
1     cli$get_value,
1     lib$get_vm,
1     lib$free_vm,
1     lib$scopy_dxdx,
1     str$free1_dx,
1     lnm$check_prot:     lnk_check_prot,
1     lnm$contsearch:     lnk_contsearch, ! ???
1     lnm$presearch:      lnk_presearch,  ! ???
1     lnm$setup:          lnk_setup,      ! ???
1     lnm$table:          lnk_table,      ! ???
1     lnm$lockr:          lnk_lock,
1     lnm$unlock:         lnk_lock,
1 !   lnm$searchlog:      lnk_searchlog,
1 !   lnm$firsttab:       lnk_firsttab,
1 !   lnm$hash:           lnk_hash,
1 !   lnm$lookup:         lnk_lookup,     ! ???
1     show$write_line,
1     fmg$match_name:     lnk_match_name;
1
1 external literal
1     show$_notran;
1
1 external
1     sgn$gl_pagedyn,
1     lnm$al_hashtbl:     vector[2],      ! Addrs of sys & proc hash tables
1     lnm$al_dirtbl:      vector[2],      ! Addrs of sys & proc directory lnmbs
1 !   ctl$gl_lnmhash,
1 !   lnm$gl_htblsizp,
1 !   lnm$gl_htblsizs,
1 !   lnm$system_directory,
1     ctl$gl_pcb:         ref $bblock;
```

```
1 !        FLAG_BLOCK
1 !
1 !        This defines various flags that are used.
1 !        Most of the flags are gotten directly from CLI calls.
1 !
1 $unit_field flag_fields =
1      set
1      flag_v_system        = [$bit],        ! True if /SYSTEM  specified
1      flag_v_group         = [$bit],        ! True if /GROUP   specified
1      flag_v_job           = [$bit],        ! True if /JOB     specified
1      flag_v_proc          = [$bit],        ! True if /PROCESS specified
1      flag_v_normal        = [$bit],        ! This means not /STRUCTURE
1      flag_v_descend       = [$bit],        ! True if descendants needed
1      flag_v_full          = [$bit],        ! True if /FULL specified
1      flag_v_recurse       = [$bit],        ! True to show recursive translations
1      flag_v_found         = [$bit]         ! True if any translations found
1      tes;
1 literal flag_k_size       = $field_set_units;
1 macro   flag_block        = $unit_block(flag_k_size) field(flag_fields) %;
1
1
1
1 !        CMDTBL_BLOCK
1 !
1 !        This is used to store a table or a logical name from the command line
1 !        (the roots of these linked lists are cmdtbl_list and cmdnam_list).
1 !        For table names, the cmdtbl_l_table field points to a list of
1 !        table_blocks that are relevant/correspond to the particular table name.
1 !        For logical names, the cmdtbl_l_table field is unused.
1 !
1 $unit_field cmdtbl_fields =
1      set
1      cmdtbl_l_next        = [$address],
1      cmdtbl_l_table       = [$address],
1      cmdtbl_q_name        = [$bytes(dsc$k_d_bln)]
1      tes;
1 literal cmdtbl_k_size     = $field_set_units;
1 macro   cmdtbl_block      = $unit_block(cmdtbl_k_size) field(cmdtbl_fields) %;
1
1
1 !        TABLE_BLOCK
1 !
1 !        This stores the information we will display about a table.
1 !        It also provides a tree structure for the tables (via the table_l_next
1 !        and table_l_child fields).
1 !        The table_l_log field points to a linked list of log_blocks.
1 !
1 $unit_field table_fields =
1      set
1      table_l_next         = [$address],
1      table_l_child        = [$address],
1      table_l_byteslm      = [$integer],    ! Same as lnmth$l_byteslm
1      table_l_bytes        = [$integer],    ! Same as lnmth$l_bytes
1      table_l_log          = [$address],    ! Pointer to log_block
1      table_b_acmode       = [$byte],       ! Same as lnmb$b_acmode
1      table_b_flags        = [$byte],       ! Same as lnmth$b_flags
1      table_t_name         = [$byte]        ! Ascic name of the table
```

```
    tes;
literal table_k_size    = $field_set_units;
macro   table_block     = $unit_block(table_k_size) field(table_fields) %;


!
!       LOG_BLOCK
!
!       This holds the information for one logical name.
!       These blocks are actually variable length --
!       from log_t_name through the end of the log_block, the contents are the
!       same as lnmb$t_name through the end of a lnmb block.
!
$unit_field log_fields =
    set
    log_l_next          = [$address],
    log_l_table         = [$address],   ! Pointer to table_block
    log_b_flags         = [$byte],      ! Same as lnmb$b_flags
    $overlay(log_b_flags)
    log_v_dummy         = [$bits(7)],
    log_v_equal         = [$bit],       ! An identical logical is in this table
    $continue
    log_b_acmode        = [$byte],      ! Same as lnmb$b_acmode
    log_t_name          = [$byte]       ! Same as lnmb$t_name
    tes;
literal log_k_size      = $field_set_units;
macro   log_block       = $unit_block(log_k_size) field(log_fields) %;


!
!       KNOWN_BLOCK
!
!       This block is used to store which logical name tables are currently
!       known (to avoid displaying the table and contained names more than
!       once -- consider /TABLE=(LNM$DCL_LOGICAL,LNM$PROCESS) ).
!
!       A lookup on this structure uses known_l_lnmth as a key, and gives
!       known_l_table as the result.
!
$unit_field known_fields =
    set
    known_l_next        = [$address],
    known_l_lnmth       = [$address],   ! Address of the lnmth block
    known_l_table       = [$address]    ! Pointer to the table_block
    tes;
literal known_k_size    = $field_set_units;
macro   known_block     = $unit_block(known_k_size) field(known_fields) %;


!
!       OWN STORAGE
!
own
    vm_size,                                    ! Remaining size of VM
    vm_addr,                                    ! Starting address of VM
    saved_fp,                                   ! Saved FP from kernel mode
    known_list:         ref known_block,        ! List of "known" tables
    cmdtbl_list:        ref cmdtbl_block,       ! Tables from CLI
    cmdnam_list:        ref cmdtbl_block,       ! Logical names from CLI
    flags:              flag_block,             ! Various flags
```

```
;   1     acmode:              byte;                 ! Intergrity mode
;   1
;   1
;   1 ! Macro for a routine to re-call itself in kernel mode
;   1 ! (if it's not already running in kernel mode).
;   1 !
;   1 ! The sole rationale for this is to make the code more readable.
;   1 !
;   1 ! This macro also saves the FP of the kernel mode code.
;   1 ! The idea is to provide an easy way to return out of kernel mode.
;   1 ! And while we're at it, lock the logical name tables for read access.
;   1 !
;   1 macro
; M 1     runs_in_kernel_(rtn) =
; M 1         begin
; M 1         builtin ap, movpsl, fp;
; M 1         local psl: $bblock[%upval];
; M 1         movpsl(psl);
; M 1         if .psl[psl$v_curmod] eql psl$c_kernel
; M 1         then
; M 1             saved_fp = .fp
; M 1         else
; M 1             begin
; M 1             local status;
; M 1             status = $cmkrnl(routin = rtn, arglst = .ap);
; M 1             saved_fp = 0;
; M 1             return .status;
; M 1             end;
; M 1         lnm$lockr(.ctl$gl_pcb);
;   1         end %;
;   1
;   1 ! Macro to write a blank line
;   1 !
;   1 bind
;   1     blank_line = %ascid'';
;   1 macro
; M 1     write_blank_line(z) =
; M 1         util_output(blank_line)
;   1         %;
;   1
;   1 routine util_output(ctrstr, p1) =
;   2     begin
;   2     return show$write_line(.ctrstr, p1);
;   1     end;


                                        .TITLE  SHOWLOG
                                        .IDENT  \V04-000\

                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                     00000 P.AAB:       .BLKB   0                                   ;
           010E0000  00000 P.AAA:       .LONG   17694720                            ;
           00000000' 00004              .ADDRESS P.AAB                              ;

                                        .PSECT  $OWN$,NOEXE,2
```

```
                                   00000 VM_SIZE:.BLKB    4
                                   00004 VM_ADDR:.BLKB    4
                                   00008 SAVED_FP:
                                                 .BLKB    4
                                   0000C KNOWN_LIST:
                                                 .BLKB    4
                                   00010 CMDTBL_LIST:
                                                 .BLKB    4
                                   00014 CMDNAM_LIST:
                                                 .BLKB    4
                                   00018 FLAGS:  .BLKB    2
                                   0001A ACMODE: .BLKB    1

                                         BLANK_LINE=          P.AAA
                                                 .EXTRN   CLI$PRESENT, CLI$GET_VALUE
                                                 .EXTRN   LIB$GET_VM, LIB$FREE_VM
                                                 .EXTRN   LIB$SCOPY_DXDX, STR$FREE1_DX
                                                 .EXTRN   LNM$CHECK_PROT, LNM$CONTSEARCH
                                                 .EXTRN   LNM$PRESEARCH, LNM$SETUP
                                                 .EXTRN   LNM$TABLE, LNM$LOCKR
                                                 .EXTRN   LNM$UNLOCK, SHOW$WRITE_LINE
                                                 .EXTRN   FMG$MATCH_NAME, SHOW$_NOTRAN
                                                 .EXTRN   SGN$GL_PAGEDYN, LNM$AL_HASHTBL
                                                 .EXTRN   LNM$AL_DIRTBL, CTL$GL_PCB

                                                 .PSECT   $CODE$,NOWRT,2

                              0000 00000 UTIL_OUTPUT:
                                                 .WORD    Save nothing                                  ; 0572
                    08    AC    9F 00002         PUSHAB   P1                                            ; 0574
                    04    AC    DD 00005         PUSHL    CTRSTR                                        ;
      00000000G  00             02    FB 00008         CALLS    #2, SHOW$WRITE_LINE                           ;
                                04 0000F         RET                                                    ; 0575

; Routine Size:  16 bytes,    Routine Base:  $CODE$ + 0000
```

```
;     1 ! Routine to allocate some memory.
;     1 ! A large chunk of memory has already been allocated (by lib$get_vm).
;     1 ! This routine simply grabs a little piece of that large chunk.
;     1 !
;     1 ! Because the amount of memory we allocated is based on sgn$gl_pagedyn, and we
;     1 ! are pretty good about not storing information twice, we should never run out
;     1 ! of this memory.  However, if we do, we're ready for it (see below).
;     1 !
;     1 routine util_allocate(size) =
;     2     begin
;     2
;     2     if (vm_size = .vm_size - .size) geq 0
;     2     then
;     2         return .vm_addr + .vm_size
;     2     else if
;     2         .saved_fp eql 0
;     2     then
;     3         begin
;     3         !
;     3         ! We aren't in kernel mode.  Signal insufficient memory.
;     3         !
;     3         signal(show$_insvirmem);
;     3         return 0;
;     3         end
;     2     else
;     3         begin
;     3         !
;     3         ! We are in kernel mode.  Return a status to the caller, by
;     3         ! setting FP to that of our first kernel mode frame and returning.
;     3         !
;     3         builtin
;     3             fp;
;     3         lnm$unlock(.ctl$gl_pcb);
;     3         set_ipl(0);
;     3         fp = .saved_fp;
;     3         return show$_insvirmem;
;     2         end;
;     1     end;



                                        003C 00000 UTIL_ALLOCATE:
                                                          .WORD   Save R2,R3,R4,R5                    ; 0584
                     55        0000'  CF   9E 00002       MOVAB   VM_SIZE, R5                         ;
                     65          04   AC   C2 00007       SUBL2   SIZE, VM_SIZE                       ; 0587
                                      06   19 0000B       BLSS    1$                                  ;
      50        04   A5               65   C1 0000D       ADDL3   VM_SIZE, VM_ADDR, R0                ; 0589
                                           04 00012       RET                                         ; 0590
                                 08   A5   D5 00013 1$:   TSTL    SAVED_FP                            ; 0591
                                      10   12 00016       BNEQ    2$                                  ;
                          007812F2    8F   DD 00018       PUSHL   #7869170                            ; 0597
          00000000G  00               01   FB 0001E       CALLS   #1, LIB$SIGNAL                      ;
                                      50   D4 00025       CLRL    R0                                  ; 0598
                                           04 00027       RET                                         ;
                     54 00000000G     00   D0 00028 2$:   MOVL    CTL$GL_PCB, R4                      ; 0608
```

```
              00000000G  00  16 0002F          JSB     LNM$UNLOCK                          ;
         12              00  DA 00035          MTPR    #0, #18                             ; 0609
         5D        08    A5  D0 00038          MOVL    SAVED_FP, FP                        ; 0610
         50 007812F2     8F  D0 0003C          MOVL    #7869170, R0                        ; 0611
                             04 00043          RET                                         ; 0613

; Routine Size:  68 bytes,    Routine Base:  $CODE$ + 0010


;  521        0614  1
;  522        0615  1
;  523        0616  1 ! Routine to re-allocate some memory.
;  524        0617  1 ! The reallocation is like removing stuff from a stack.
;  525        0618  1 ! The parameter is the address of where we want to back up to.
;  526        0619  1 !
;  527        0620  1 routine util_reallocate(addr): novalue =
;  528        0621  2     begin
;  529        0622  2     vm_size = .addr - .vm_addr;
;  530        0623  1     end;



                                 0000 00000 UTIL_REALLOCATE:
                                                .WORD   Save nothing                       ; 0620
      0000'  CF      04   AC     0000'  CF  C3 00002   SUBL3   VM_ADDR, ADDR, VM_SIZE      ; 0622
                                            04 0000B   RET                                 ; 0623

; Routine Size:  12 bytes,    Routine Base:  $CODE$ + 0054


;  531        0624  1
;  532        0625  1
;  533        0626  1 ! Routine to manipulate pointers.
;  534        0627  1 ! All the parameters are by reference, except the last which is by value.
;  535        0628  1 ! Slide the rightmost parameter to the left, and so on.  Return the value
;  536        0629  1 ! of the leftmost parameter.
;  537        0630  1 !
;  538        0631  1 routine util_slide =
;  539        0632  2     begin
;  540        0633  2     builtin
;  541        0634  2         actualcount,
;  542        0635  2         actualparameter;
;  543        0636  2     macro
;  544        0637  2         swap_(x,y) = (local t; t = .x; x = .y; y = .t) %;
;  545        0638  2     local
;  546        0639  2         x;
;  547        0640  2     x = actualparameter(actualcount());
;  548        0641  2     decr i from actualcount()-1 to 1 do
;  549        0642  2         swap_(actualparameter(.i), x);
;  550        0643  2     return .x;
;  551        0644  1     end;
```

```
                                   000C 00000 UTIL_SLIDE:
                                                        .WORD   Save R2,R3                                  ; 0631
           50          6C   9A 00002                    MOVZBL  (AP), R0                                    ; 0640
           53        6C40   D0 00005                    MOVL    (AP)[R0], X
           50          6C   9A 00009                    MOVZBL  (AP), I                                     ; 0641
                       11   11 0000C                    BRB     2$
           51        6C40   D0 0000E 1$:                MOVL    (AP)[I], R1                                 ; 0642
           52          61   D0 00012                    MOVL    (R1), T                                     ;
           51        6C40   D0 00015                    MOVL    (AP)[I], R1                                 ;
           61          53   D0 00019                    MOVL    X, (R1)                                     ;
           53          52   D0 0001C                    MOVL    T, X                                        ;
           EC          50   F5 0001F 2$:                SOBGTR  I, 1$                                       ;
           50          53   D0 00022                    MOVL    X, R0                                       ; 0643
                            04 00025                    RET                                                 ; 0644

; Routine Size:  38 bytes,    Routine Base:  $CODE$ + 0060


;  552          0645  1
;  553          0646  1
;  554          0647  1  ! Routine to add another cmdtbl_block to a linked list.
;  555          0648  1  !
;  556          0649  1  routine add_to_cmd_list(
;  557          0650  1          p_root,                 ! Address of the root of the linked list
;  558          0651  1          name:   ref $bblock     ! String to be added to the list
;  559          0652  1          ): novalue =
;  560          0653  2      begin
;  561          0654  2      bind
;  562          0655  2          root = .p_root: ref cmdtbl_block;
;  563          0656  2      local
;  564          0657  2          ptr:    ref cmdtbl_block;
;  565          0658  2      ptr = util_allocate(cmdtbl_k_size);
;  566          0659  2      ch$move(dsc$k_d_bln, name[base_], ptr[cmdtbl_q_name]);
;  567          0660  2      ptr = util_slide(ptr[cmdtbl_l_next], root, ptr[base_]);
;  568          0661  1      end;



                                   007C 00000 ADD_TO_CMD_LIST:
                                                        .WORD   Save R2,R3,R4,R5,R6                         ; 0649
                               10   DD 00002            PUSHL   #16                                         ; 0658
                   82    AF    01   FB 00004            CALLS   #1, UTIL_ALLOCATE                           ;
                         56    50   D0 00008            MOVL    R0, PTR                                     ;
      08    A6     08    BC    08   28 0000B            MOVC3   #8, @NAME, 8(PTR)                           ; 0659
                               56   DD 00011            PUSHL   PTR                                         ; 0660
                         04    AC   DD 00013            PUSHL   P_ROOT                                      ;
                               56   DD 00016            PUSHL   PTR                                         ;
                   BE    AF    03   FB 00018            CALLS   #3, UTIL_SLIDE                              ;
                         56    50   D0 0001C            MOVL    R0, PTR                                     ;
                                    04 0001F            RET                                                 ; 0661

; Routine Size:  32 bytes,    Routine Base:  $CODE$ + 0086
```

```
;  1
;  1
;  1 ! This clever little routine reverses a linked list.
;  1 ! This simplifies building the list, and traversing the list.
;  1 !
;  1 routine reverse_list(
;  1         p_root,          ! Address of the root of the linked list
;  1         next             ! Byte offset of the 'next' field in a node
;  1         ): novalue =
;  2     begin
;  2     bind
;  2         root = .p_root: ref $bblock;
;  2     local
;  2         ptr:     ref $bblock initial(0);
;  2     while .root neq 0 do
;  2         ptr = util_slide(root, root[.next,l_], ptr[base_]);
;  2     root = ptr[base_];
;  1     end;



                                  0000 00000 REVERSE_LIST:
                                                        .WORD   Save nothing                  ; 0667
                                50   D4 00002           CLRL    PTR                           ; 0673
                         04     BC   D5 00004 1$:       TSTL    @P_ROOT                       ; 0676
                                11   13 00007           BEQL    2$                            ;
                                50   DD 00009           PUSHL   PTR                           ; 0677
        7E       04    BC    08 AC   C1 0000B           ADDL3   NEXT, @P_ROOT, -(SP)          ;
                             04 AC   DD 00011           PUSHL   P_ROOT                        ;
                 A2    AF    03      FB 00014           CALLS   #3, UTIL_SLIDE                ;
                                EA   11 00018           BRB     1$                            ;
                 04    BC    50      D0 0001A 2$:       MOVL    PTR, @P_ROOT                  ; 0678
                                     04 0001E           RET                                   ; 0679

; Routine Size:  31 bytes,    Routine Base:  $CODE$ + 00A6


;  1
;  1 routine sort_list(        ! list[cmdtbl_l_table], %fieldexpand (table_l_next,0));
;  1         p_root,           ! Address of the root of the linked list
;  1         next,             ! Byte offset of the 'next' field in a node
;  1         compare           ! Address of a routine to compare nodes
;  1         ): novalue =
;  2     begin
;  2     !
;  2     ! Alphabetize the list pointed to by p_root
;  2     !
;  2     bind
;  2         root = .p_root: ref $bblock;
;  2     bind routine
;  2         compare_rtn = .compare: lnk_compare;
;  2     local
;  2         sort:   ref $bblock initial(0);
;  2     while root[base_] neq 0 do
;  3         begin
```

```
;  605        0698  3          !
;  606        0699  3          ! Grab one item from root list, and insert in it sort list
;  607        0700  3          !
;  608        0701  3          local
;  609        0702  3              r:  ref $bblock,
;  610        0703  3              p:  ref vector[1];
;  611        0704  3          r = sort - .next;
;  612        0705  3          while (p = r[.next,l_]; r = .p[0]) neq 0 do
;  613        0706  4              begin
;  614        0707  4              if compare_rtn( r[base_], root[base_] ) geq 0
;  615        0708  4              then
;  616        0709  4                  exitloop;
;  617        0710  3              end;
;  618        0711  3          r = util_slide(root, root[.next,l_], 0);
;  619        0712  3          r = util_slide(r[.next,l_], p[0], r[base_]);
;  620        0713  2          end;
;  621        0714  2      root = sort[base_];
;  622        0715  1      end;



                                  00FC 00000 SORT_LIST:
                                                      .WORD   Save R2,R3,R4,R5,R6,R7                    ; 0681
                                   7E  D4 00002       CLRL    SORT                                      ; 0693
                  57               6E  9E 00004       MOVAB   SORT, R7                                  ; 0704
                  57         08    AC  C2 00007       SUBL2   NEXT, R7
                  55         04    BC  D0 0000B 1$:   MOVL    @P_ROOT, R5                               ; 0696
                                   3A  13 0000F       BEQL    4$
                  54               57  D0 00011       MOVL    R7, R                                     ; 0704
        56        54         08    AC  C1 00014 2$:   ADDL3   NEXT, R, P                                ; 0705
                  54               66  D0 00019       MOVL    (P), R                                    ;
                                   0A  13 0001C       BEQL    3$                                        ;
                  50               54  7D 0001E       MOVQ    R, R0                                     ; 0707
                             0C    BC  16 00021       JSB     @COMPARE                                  ;
                                   50  D5 00024       TSTL    R0                                        ;
                                   EC  19 00026       BLSS    2$                                        ;
                                   7E  D4 00028 3$:   CLRL    -(SP)                                     ; 0711
                             08 BC45   9F 0002A       PUSHAB  @NEXT[R5]                                 ;
                             04    AC  DD 0002E       PUSHL   P_ROOT                                    ;
              FF65    CF           03  FB 00031       CALLS   #3, UTIL_SLIDE                            ;
                      54           50  D0 00036       MOVL    R0, R                                     ;
                                   54  DD 00039       PUSHL   R                                         ; 0712
                                   56  DD 0003B       PUSHL   P                                         ;
                             08 BC44   9F 0003D       PUSHAB  @NEXT[R]                                  ;
              FF55    CF           03  FB 00041       CALLS   #3, UTIL_SLIDE                            ;
                      54           50  D0 00046       MOVL    R0, R                                     ;
                                   C0  11 00049       BRB     1$                                        ; 0696
           04         BC           6E  D0 0004B 4$:   MOVL    SORT, @P_ROOT                             ; 0714
                                       04 0004F       RET                                               ; 0715

; Routine Size:  80 bytes,    Routine Base:  $CODE$ + 00C5


;  623        0716  1
;  624        0717  1
```

```
1 ! This recursive routine does a preorder tree traversal
1 ! (visit node, visit left subtree, visit right subtree).
1 !
1 ! To reduce the amount of stack space required, the traversal
1 ! is done partly by recursion, partly by iteration.
1 !
1 routine visit_tree(
1         root:   ref $bblock,        ! Root of the tree
1         loffset,                    ! Byte offset of the 'left' field in a node
1         roffset,                    ! Byte offset of the 'right' field in a node
1         action,                     ! Address of an action routine
1         depth                       ! Current depth of the traversal
1         ): novalue =
2     begin
2     local
2         ptr:    ref $bblock;
2     ptr = root - .roffset;
2     while (ptr = .ptr[.roffset,l_]) neq 0 do
3         begin
3         (.action)(ptr[base_], .depth);
3         visit_tree(.ptr[.loffset,l_], .loffset, .roffset, .action, .depth+1);
2         end;
1     end;
```

```
                                    000C 00000 VISIT_TREE:
                                                      .WORD   Save R2,R3                       ; 0724
             50     04  AC  9E 00002                  MOVAB   ROOT, R0                         ; 0734
52           50     0C  AC  C3 00006                  SUBL3   ROFFSET, R0, PTR                 ;
53       14  AC         01  C1 0000B                  ADDL3   #1, DEPTH, R3                    ; 0738
50           52     0C  AC  C1 00010 1$:              ADDL3   ROFFSET, PTR, R0                 ; 0735
             52         60  D0 00015                  MOVL    (R0), PTR                        ;
                        1F  13 00018                  BEQL    2$                               ;
                    14  AC  DD 0001A                  PUSHL   DEPTH                            ; 0737
                        52  DD 0001D                  PUSHL   PTR                              ;
         10  BC             02  FB 0001F              CALLS   #2, @ACTION                      ;
                        53  DD 00023                  PUSHL   R3                               ; 0738
             7E     0C  AC  7D 00025                  MOVQ    ROFFSET, -(SP)                   ;
                    08  AC  DD 00029                  PUSHL   LOFFSET                          ;
50           52     08  AC  C1 0002C                  ADDL3   LOFFSET, PTR, R0                 ;
                        60  DD 00031                  PUSHL   (R0)                             ;
         C9  AF         05  FB 00033                  CALLS   #5, VISIT_TREE                   ;
                        D7  11 00037                  BRB     1$                               ; 0735
                            04 00039 2$:              RET                                      ; 0740

; Routine Size:  58 bytes,    Routine Base:  $CODE$ + 0115
```

```
1 ! This routine calls the CLI and builds a list of cmdtbl_blocks to hold
1 ! the table names specified by the user.
1 !
1 routine get_tables =
2     begin
2     local
2         desc:   $bblock[dsc$c_d_bln],
2         root:   ref cmdtbl_block initial(0);
2     macro
2         add_(x) = add_to_cmd_list( root, x ) %;
2
2     if not .flags[flag_v_normal]
2     then
3         begin
3         !
3         ! For /STRUCTURE, just specify the directories,
3         ! and ask for descendants.
3         !
3         ! Note that the /STRUCTURE qualifier should not be used with
3         ! any of the /TABLE, /PROCESS, /JOB, /GROUP or /SYSTEM qualifiers.
3         !
3         add_( %ascid'LNM$PROCESS_DIRECTORY' );
3         add_( %ascid'LNM$SYSTEM_DIRECTORY' );
3         flags[flag_v_descend] = true;
3         end
2     else
3         begin
3         !
3         ! Get all the relevant table names, in the following order:
3         !       names from the /TABLE qualifier
3         !       names from the /PROCESS, /JOB, /GROUP and /SYSTEM qualifiers
3         !       if still no names, use LNM$DCL_LOGICAL
3         !
3         while
4             begin
4             $init_dyndesc(desc);
4             cli$get_value(%ascid'table', desc[base_])
4             end
3         do
3             add_(desc[base_]);
3         if .flags[flag_v_proc]  then add_( %ascid 'LNM$PROCESS' );
3         if .flags[flag_v_job]   then add_( %ascid 'LNM$JOB' );
3         if .flags[flag_v_group] then add_( %ascid 'LNM$GROUP' );
3         if .flags[flag_v_system]then add_( %ascid 'LNM$SYSTEM' );
3         if .root eql 0          then add_( %ascid 'LNM$DCL_LOGICAL' );
2         end;
2
2     reverse_list( root, %fieldexpand_(cmdtbl_l_next,0) );
2
2     return .root;
1     end;


                                                            .PSECT  $PLIT$,NOWRT,NOEXE,2

52 49 44 5F 53 53 45 43 4F 52 50 24 4D 4E 4C  00008 P.AAD:  .ASCII  \LNM$PROCESS_DIRECTORY\<0><0><0>      ;
```

```
                        00  00  00  59  52  4F  54  43  45  00017                                              ;
                                              010E0015  00020 P.AAC:   .LONG   17694741                        ;
                                             00000000'  00024          .ADDRESS P.AAD                          ;
45  52  49  44  5F  4D  45  54  53  59  53  24  4D  4E  4C  00028 P.AAF:   .ASCII  \LNM$SYSTEM_DIRECTORY\       ;
                                    59  52  4F  54  43  00037                                                  ;
                                              010E0014  0003C P.AAE:   .LONG   17694740                        ;
                                             00000000'  00040          .ADDRESS P.AAF                          ;
                        00  00  00  65  6C  62  61  74  00044 P.AAH:   .ASCII  \table\<0><0><0>                ;
                                              010E0005  0004C P.AAG:   .LONG   17694725                        ;
                                             00000000'  00050          .ADDRESS P.AAH                          ;
            00  53  53  45  43  4F  52  50  24  4D  4E  4C  00054 P.AAJ:   .ASCII  \LNM$PROCESS\<0>            ;
                                              010E000B  00060 P.AAI:   .LONG   17694731                        ;
                                             00000000'  00064          .ADDRESS P.AAJ                          ;
                        00  42  4F  4A  24  4D  4E  4C  00068 P.AAL:   .ASCII  \LNM$JOB\<0>                    ;
                                              010E0007  00070 P.AAK:   .LONG   17694727                        ;
                                             00000000'  00074          .ADDRESS P.AAL                          ;
            00  00  00  50  55  4F  52  47  24  4D  4E  4C  00078 P.AAN:   .ASCII  \LNM$GROUP\<0><0><0>        ;
                                              010E0009  00084 P.AAM:   .LONG   17694729                        ;
                                             00000000'  00088          .ADDRESS P.AAN                          ;
            00  00  4D  45  54  53  59  53  24  4D  4E  4C  0008C P.AAP:   .ASCII  \LNM$SYSTEM\<0><0>          ;
                                              010E000A  00098 P.AAO:   .LONG   17694730                        ;
                                             00000000'  0009C          .ADDRESS P.AAP                          ;
4C  41  43  49  47  4F  4C  5F  4C  43  44  24  4D  4E  4C  000A0 P.AAR:   .ASCII  \LNM$DCL_LOGICAL\<0>        ;
                                                    00  000AF                                                  ;
                                              010E000F  000B0 P.AAQ:   .LONG   17694735                        ;
                                             00000000'  000B4          .ADDRESS P.AAR                          ;


                                                               .PSECT  $CODE$,NOWRT,2

                                            001C 00000 GET_TABLES:
                                                               .WORD   Save R2,R3,R4                   ; 0744
                        54      0000'   CF  9E 00002           MOVAB   FLAGS, R4                       ;
                        53      0000'   CF  9E 00007           MOVAB   P.AAC, R3                       ;
                        52      FF27    CF  9E 0000C           MOVAB   ADD_TO_CMD_LIST, R2             ;
                        5E              08  C2 00011           SUBL2   #8, SP                          ;
                                        7E  D4 00014           CLRL    ROOT                            ; 0745
          16            64              04  E0 00016           BBS     #4, FLAGS, 1$                   ; 0752
                                        53  DD 0001A           PUSHL   R3                              ; 0762
                                04      AE  9F 0001C           PUSHAB  ROOT                            ;
                        62              02  FB 0001F           CALLS   #2, ADD_TO_CMD_LIST             ;
                                1C      A3  9F 00022           PUSHAB  P.AAE                           ; 0763
                                04      AE  9F 00025           PUSHAB  ROOT                            ;
                        62              02  FB 00028           CALLS   #2, ADD_TO_CMD_LIST             ;
                        64              20  88 0002B           BISB2   #32, FLAGS                      ; 0764
                                        67  11 0002E           BRB     7$                              ; 0752
                   04   AE  020E0000    8F  D0 00030 1$:       MOVL    #34471936, DESC                 ; 0776
                                08      AE  D4 00038           CLRL    DESC+4                          ;
                                04      AE  9F 0003B           PUSHAB  DESC                            ; 0777
                                2C      A3  9F 0003E           PUSHAB  P.AAG                           ;
              00000000G 00              02  FB 00041           CALLS   #2, CLI$GET_VALUE               ;
                        0B              50  E9 00048           BLBC    R0, 2$                          ;
                                04      AE  9F 0004B           PUSHAB  DESC                            ; 0780
                                04      AE  9F 0004E           PUSHAB  ROOT                            ;
                        62              02  FB 00051           CALLS   #2, ADD_TO_CMD_LIST             ;
                                        DA  11 00054           BRB     1$                              ;
```

```
09        64                 03  E1 00056 2$:    BBC      #3, FLAGS, 3$                   ; 0781
                       40    A3  9F 0005A        PUSHAB   P.AAI                           ;
                       04    AE  9F 0005D        PUSHAB   ROOT                            ;
          62                 02  FB 00060        CALLS    #2, ADD_TO_CMD_LIST             ;
09        64                 02  E1 00063 3$:    BBC      #2, FLAGS, 4$                   ; 0782
                       50    A3  9F 00067        PUSHAB   P.AAK                           ;
                       04    AE  9F 0006A        PUSHAB   ROOT                            ;
          62                 02  FB 0006D        CALLS    #2, ADD_TO_CMD_LIST             ;
09        64                 01  E1 00070 4$:    BBC      #1, FLAGS, 5$                   ; 0783
                       64    A3  9F 00074        PUSHAB   P.AAM                           ;
                       04    AE  9F 00077        PUSHAB   ROOT                            ;
          62                 02  FB 0007A        CALLS    #2, ADD_TO_CMD_LIST             ;
          09                 64  E9 0007D 5$:    BLBC     FLAGS, 6$                       ; 0784
                       78    A3  9F 00080        PUSHAB   P.AAO                           ;
                       04    AE  9F 00083        PUSHAB   ROOT                            ;
          62                 02  FB 00086        CALLS    #2, ADD_TO_CMD_LIST             ;
                             6E  D5 00089 6$:    TSTL     ROOT                            ; 0785
                             0A  12 0008B        BNEQ     7$                              ;
                     0090    C3  9F 0008D        PUSHAB   P.AAQ                           ;
                       04    AE  9F 00091        PUSHAB   ROOT                            ;
          62                 02  FB 00094        CALLS    #2, ADD_TO_CMD_LIST             ;
                             7E  D4 00097 7$:    CLRL     -(SP)                           ; 0788
                       04    AE  9F 00099        PUSHAB   ROOT                            ;
     20   A2                 02  FB 0009C        CALLS    #2, REVERSE_LIST                ;
          50                 6E  D0 000A0        MOVL     ROOT, R0                        ; 0790
                                 04 000A3        RET                                      ; 0791
```

```
; Routine Size:  164 bytes,    Routine Base:  $CODE$ + 014F
```

```
1
1
1 ! This routine calls the CLI and builds a list of cmdtbl_blocks to hold
1 ! the logical names specified by the user.
1 !
1 routine get_lognam =
2     begin
2     local
2         desc:    $bblock[dsc$c_d_bln],
2         root:    ref cmdtbl_block initial(0);
2
2     macro
2         add_(x) = add_to_cmd_list( root, x ) %;
2     while
3         begin
3         $init_dyndesc(desc);
3         cli$get_value(%ascid'logname', desc[base_])
3         end
2     do
3         begin
3         !
3         ! If the last character is a colon, remove it.
3         !
3         if ch$rchar(.desc[dsc$w_length]+.desc[dsc$a_pointer]) eql ':'
3         then
4             begin
4             local
```

```
;  727          0819  4                tmp:     vector[2];
;  728          0820  4            tmp[0] = .desc[dsc$w_length] - 1;
;  729          0821  4            tmp[1] = .desc[dsc$a_pointer];
;  730          0822  4            lib$scopy_dxdx(tmp[0], desc[base_]);
;  731          0823  3            end;
;  732          0824  3        add_(desc[base_]);
;  733          0825  2        end;
;  734          0826  2    if .root eql 0              then add_( %ascid '*' );
;  735          0827  2    reverse_list( root, %fieldexpand_(cmdtbl_l_next,0) );
;  736          0828  2
;  737          0829  2    return .root;
;  738          0830  1    end;



                                                                  .PSECT  $PLIT$,NOWRT,NOEXE,2

                  00  65  6D  61  6E  67  6F  6C  000B8 P.AAT:    .ASCII  \logname\<0>                            ;
                                          010E0007  000C0 P.AAS:    .LONG   17694727                                ;
                                         00000000'  000C4           .ADDRESS P.AAT                                  ;
                                  00  00  00  2A  000C8 P.AAV:    .ASCII  \*\<0><0><0>                            ;
                                          010E0001  000CC P.AAU:    .LONG   17694721                                ;
                                         00000000'  000D0           .ADDRESS P.AAV                                  ;


                                                                  .PSECT  $CODE$,NOWRT,2

                                            0000 00000 GET_LOGNAM:
                                                                  .WORD   Save nothing                            ; 0797
              5E                   10  C2 00002                   SUBL2   #16, SP                                 ;
                                   7E  D4 00005                   CLRL    ROOT                                    ; 0798
     0C   AE 020E0000              8F  D0 00007 1$:               MOVL    #34471936, DESC                         ; 0807
                        10         AE  D4 0000F                   CLRL    DESC+4                                  ;
                        0C         AE  9F 00012                   PUSHAB  DESC                                    ; 0808
                     0000'         CF  9F 00015                   PUSHAB  P.AAS                                   ;
  00000000G   00                   02  FB 00019                   CALLS   #2, CLI$GET_VALUE                       ;
              34                   50  E9 00020                   BLBC    R0, 3$                                  ;
              50        0C         AE  3C 00023                   MOVZWL  DESC, R0                                ; 0815
              50        10         AE  C0 00027                   ADDL2   DESC+4, R0                              ;
              3A                   60  91 0002B                   CMPB    (R0), #58                               ;
                                   1A  12 0002E                   BNEQ    2$                                      ;
     04   AE            0C         AE  3C 00030                   MOVZWL  DESC, TMP                               ; 0820
                        04         AE  D7 00035                   DECL    TMP                                     ;
     08   AE            10         AE  D0 00038                   MOVL    DESC+4, TMP+4                           ; 0821
                        0C         AE  9F 0003D                   PUSHAB  DESC                                    ; 0822
                        08         AE  9F 00040                   PUSHAB  TMP                                     ;
  00000000G   00                   02  FB 00043                   CALLS   #2, LIB$SCOPY_DXDX                      ;
                        0C         AE  9F 0004A 2$:               PUSHAB  DESC                                    ; 0824
                        04         AE  9F 0004D                   PUSHAB  ROOT                                    ;
     FE3E   CF                     02  FB 00050                   CALLS   #2, ADD_TO_CMD_LIST                     ;
                                   B0  11 00055                   BRB     1$                                      ; 0805
                                   6E  D5 00057 3$:               TSTL    ROOT                                    ; 0826
                                   0C  12 00059                   BNEQ    4$                                      ;
                     0000'         CF  9F 0005B                   PUSHAB  P.AAU                                   ;
                        04         AE  9F 0005F                   PUSHAB  ROOT                                    ;
     FE2C   CF                     02  FB 00062                   CALLS   #2, ADD_TO_CMD_LIST                     ;
```

```
                            7E  D4 00067 4$:    CLRL    -(SP)                          ; 0827
                   04       AE  9F 00069        PUSHAB  ROOT                           ;
          FE42  CF              02  FB 0006C    CALLS   #2, REVERSE_LIST               ;
                50              6E  D0 00071    MOVL    ROOT, R0                       ; 0829
                                    04 00074    RET                                    ; 0830

; Routine Size:  117 bytes,    Routine Base:  $CODE$ + 01F3
```

```
1 ! Little routine to check the protection on a lnmth block
1 !
1 routine check_prot(x: ref $bblock) =
2     begin
2     literal
2         pro_mask = arm$m_read;                     ! Protection mask
2     if not .x[lnmth$v_shareable] then return true
2     else if .x[lnmth$v_directory] then return true       ! ???
2     else if .x[lnmth$l_orb] eql 0 then return true
2     else return lnm$check_prot(x[base_], pro_mask, .ctl$gl_pcb, 0);
1     end;



                                   083C 00000 CHECK_PROT:
                                                      .WORD   Save R2,R3,R4,R5,R11            ; 0833
              51        04   AC  D0 00002             MOVL    X, R1                           ; 0837
              09             61  E9 00006             BLBC    (R1), 1$
      05      61             01  E0 00009             BBS     #1, (R1), 1$                    ; 0838
                        05   A1  D5 0000D             TSTL    5(R1)                           ; 0839
                             04  12 00010             BNEQ    2$                              ;
              50             01  D0 00012 1$:         MOVL    #1, R0                          ;
                                 04 00015             RET                                     ;
                             55  D4 00016 2$:         CLRL    R5                              ; 0840
              54 00000000G   00  D0 00018             MOVL    CTL$GL_PCB, R4                  ;
              52             01  D0 0001F             MOVL    #1, R2                          ;
                 00000000G   00  16 00022             JSB     LNM$CHECK_PROT                  ;
                                 04 00028             RET                                     ; 0841

; Routine Size:  41 bytes,    Routine Base:  $CODE$ + 0268


1
1
1 routine recheck_tables: novalue =
2     begin
2     !
2     ! When the code reenters kernel mode, there is always the possibility
2     ! that some logical name table found by get_relevant_tables got deleted,
2     ! or is no longer accessible (which is okay per se, since we really don't
2     ! need to find the lnmth block again).
2     !
2     ! However, it is possible that one of the lnmth blocks got deleted and
2     ! reused as another lnmth block.  This means that we may be displaying
2     ! logical names that the user has no business seeing!
2     !
2     ! This routine prevents the above security hole, by the following method:
2     ! For each table in known_list, we find the table again (by name or by
2     ! calling for_each_lnmth and checking the known_l_lnmth value), and verify
2     ! that the user has read access to the table (this prevents the security
2     ! hole; we should also check that the tables have the same name and
2     ! attributes).  If it doesn't check out, the known_l_lnmth value is zeroed;
2     ! since lnmb$l_table is never zero, we'll not find any more logical names
2     ! for this table.
2     !
```

```
! This is a pain.
!
local
    known:          ref known_block,
    an_rt_block:    rt_block,
    status;
register
    rt = 5: ref rt_block initial(an_rt_block[base_]);

! Initialize the rt_block for the table name
!
rt[rt_b_acmode] = psl$c_user;          ! Acceptable integrity mode
rt[rt_b_flags] = 0;                    ! Initialize flags
rt[rt_v_case] = true;                  ! Case-insensitive

known = known_list - %fieldexpand_(known_l_next,0);
while (known = .known[known_l_next]) neq 0 do
    if .known[known_l_lnmth] neq 0 then      ! Have we already trashed this?
    begin
    local
        table:      ref table_block,
        lnmth:      ref $bblock;

    table = .known[known_l_table];
    status = lnm$setup(.table[table_t_name], 1+table[table_t_name],
        rt[base_]; lnmth);
    while true do
        begin
        if not .status
        then
            begin
            if .status eql ss$_toomanylnam
            then
                rt[rt_b_depth] = .rt[rt_b_depth] - 1        ! Backup depth
            else
                begin
                known[known_l_lnmth] = 0;
                exitloop            ! Presumably no more names
                end;
            end
        else if
            lnmth[base_] eql .known[known_l_lnmth]
        then
            begin
            !
            ! We found what may be this table (it has the same address).
            ! Check the protection.
            ! We should also check for the same name and attributes, but
            ! this prevents any security violations, even though we may
            ! display incorrect information.
            !
            if not check_prot(lnmth[base_]) then known[known_l_lnmth] = 0;
            exitloop;
            end;

        rt[rt_b_tries] = rt_c_maxtries;     ! Reinitialize number of tries
        status = lnm$table(rt[base_]; lnmth);
```

```
;  831          0922  3                 end;
;  832          0923  2             end;
;  833          0924  1         end;



                                    00FC 00000 RECHECK_TABLES:
                                                          .WORD   Save R2,R3,R4,R5,R6,R7                    ; 0844
               5E            30  C2 00002                 SUBL2   #48, SP                                   ; 0872
               55            6E  9E 00005                 MOVAB   AN_RT_BLOCK, RT                           ; 0872
               65            03  B0 00008                 MOVW    #3, (RT)                                  ; 0876
        01     A5            01  88 0000B                 BISB2   #1, 1(RT)                                 ; 0878
               54     0000'  CF  9E 0000F                 MOVAB   KNOWN_LIST, KNOWN                         ; 0880
               54            64  D0 00014 1$:             MOVL    (KNOWN), KNOWN                            ; 0881
                             4E  13 00017                 BEQL    6$                                        ;
                      04     A4  D5 00019                 TSTL    4(KNOWN)                                  ; 0882
                             F6  13 0001C                 BEQL    1$                                        ;
               50     08     A4  D0 0001E                 MOVL    8(KNOWN), TABLE                           ; 0888
               53     17     A0  9E 00022                 MOVAB   23(TABLE), R3                             ; 0889
               52     16     A0  9A 00026                 MOVZBL  22(TABLE), R2                             ; 0890
                  00000000G  00  16 0002A                 JSB     LNM$SETUP                                 ;
               57            50  D0 00030 2$:             MOVL    R0, STATUS                                ;
               56            51  D0 00033                 MOVL    R1, R6                                    ;
               0E            57  E8 00036                 BLBS    STATUS, 3$                                ; 0893
    00000374   8F            57  D1 00039                 CMPL    STATUS, #884                              ; 0896
               14            12  00040                    BNEQ    4$                                        ;
                      02     A5  97 00042                 DECB    2(RT)                                     ; 0898
                             14  11 00045                 BRB     5$                                        ;
        04     A4            56  D1 00047 3$:             CMPL    LNMTH, 4(KNOWN)                           ; 0906
                             0E  12 0004B                 BNEQ    5$                                        ;
                             56  DD 0004D                 PUSHL   LNMTH                                     ; 0916
        84     AF            01  FB 0004F                 CALLS   #1, CHECK_PROT                            ;
               BE            50  E8 00053                 BLBS    R0, 1$                                    ;
                      04     A4  D4 00056 4$:             CLRL    4(KNOWN)                                  ;
                             B9  11 00059                 BRB     1$                                        ; 0908
        03     A5            01  8E 0005B 5$:             MNEGB   #1, 3(RT)                                 ; 0920
                  00000000G  00  16 0005F                 JSB     LNM$TABLE                                 ; 0921
                             C9  11 00065                 BRB     2$                                        ;
                                 04 00067 6$:             RET                                               ; 0924

; Routine Size:  104 bytes,    Routine Base:  $CODE$ + 0291
```

```
;    835          0925  1  ! This routine does a lookup on the known_list.
;    836          0926  1  ! It looks up the address of a lnmth block:
;    837          0927  1  !        if found, it returns the address of the table_block,
;    838          0928  1  !        if not found, it returns zero.
;    839          0929  1  !
;    840          0930  1  routine lookup_table(lnmth_address) =
;    841          0931  2      begin
;    842          0932  2      local
;    843          0933  2          known:  ref known_block;
;    844          0934  2      known = known_list - %fieldexpand_(known_l_next,0);
;    845          0935  2      while (known = .known[known_l_next]) neq 0 do
;    846          0936  2          if .known[known_l_lnmth] eql .lnmth_address
;    847          0937  2          then
;    848          0938  2              return .known[known_l_table];
;    849          0939  2      return 0;
;    850          0940  1      end;


                                          0000 00000 LOOKUP_TABLE:
                                                           .WORD   Save nothing                          ; 0930
                 50       0000'   CF   9E 00002            MOVAB   KNOWN_LIST, KNOWN                     ; 0934
                 50               60   D0 00007 1$:        MOVL    (KNOWN), KNOWN                        ; 0935
                                  0C   13 0000A            BEQL    2$                                    ;
          04     AC         04    A0   D1 0000C            CMPL    4(KNOWN), LNMTH_ADDRESS               ; 0936
                                  F4   12 00011            BNEQ    1$                                    ;
                 50         08    A0   D0 00013            MOVL    8(KNOWN), R0                          ; 0938
                                       04 00017            RET                                           ;
                                  50   D4 00018 2$:        CLRL    R0                                    ; 0939
                                       04 0001A            RET                                           ; 0940

; Routine Size:  27 bytes,    Routine Base:  $CODE$ + 02F9


;    851          0941  1
;    852          0942  1
;    853          0943  1  ! This routine adds a lnmth block to the list of known tables.
;    854          0944  1  !
;    855          0945  1  routine add_to_known(
;    856          0946  1          lnmth:  ref $bblock
;    857          0947  1          ) =
;    858          0948  2      begin
;    859          0949  2      !
;    860          0950  2      ! Find whether we already know about this lnmth entry
;    861          0951  2      ! If we already know about it (or don't want the user to know about it,
;    862          0952  2      ! because of protection) then return 0,
;    863          0953  2      ! Otherwise, return a pointer to the table_block
;    864          0954  2      !
;    865          0955  2      local
;    866          0956  2          known:  ref known_block;
;    867          0957  2
;    868          0958  2
;    869          0959  2      ! Check protection on the table
;    870          0960  2      !
;    871          0961  2      if not check_prot(lnmth[base_]) then return 0;
```

```
;     2
;     2
;     2   ! See if it's already on the known list
;     2   ! If so, return a zero.
;     2   !
;     2   if lookup_table(lnmth[base_]) neq 0 then return 0;
;     2
;     2
;     3   begin
;     3   bind lnmb = .lnmth[lnmth$l_name]: $bblock;
;     3   local table: ref table_block;
;     3   !
;     3   ! Check the access mode on the table
;     3   !
;     3   if .lnmb[lnmb$b_acmode] gtru .acmode then return 0;
;     3   !
;     3   ! Create and initialize a table_block
;     3   !
;     3   table = util_allocate(table_k_size + .lnmb[lnmb$t_name]);
;     3   ch$move(1+.lnmb[lnmb$t_name], lnmb[lnmb$t_name], table[table_t_name]);
;     3   table[table_b_acmode] = .lnmb[lnmb$b_acmode];
;     3   table[table_b_flags] = .lnmth[lnmth$b_flags];
;     3   table[table_l_bytes] = .lnmth[lnmth$l_bytes];
;     3   table[table_l_byteslm] = .lnmth[lnmth$l_byteslm];
;     3   !
;     3   ! Create and initialize a known_block
;     3   ! Add the known_block to the list of known tables
;     3   !
;     3   known = util_allocate(known_k_size);
;     3   known[known_l_lnmth] = lnmth[base_];
;     3   known[known_l_table] = table[base_];
;     3   util_slide(known[known_l_next], known_list, known[base_]);
;     3   return table[base_];
;     2   end;
;     2
;     1   end;


                                 01FC 00000 ADD_TO_KNOWN:
                                            .WORD    Save R2,R3,R4,R5,R6,R7,R8          ; 0945
             58       04   AC   D0 00002    MOVL     LNMTH, R8                          ; 0961
                           58   DD 00006    PUSHL    R8                                 ;
    FF47     CF            01   FB 00008    CALLS    #1, CHECK_PROT                     ;
             5F            50   E9 0000D    BLBC     R0, 1$                             ;
                           58   DD 00010    PUSHL    R8                                 ; 0967
      CF     AF            01   FB 00012    CALLS    #1, LOOKUP_TABLE                   ;
                           50   D5 00016    TSTL     R0                                 ;
                           55   12 00018    BNEQ     1$                                 ;
             56       09   A8   D0 0001A    MOVL     9(R8), R6                          ; 0971
   0000'     CF       0B   A6   91 0001E    CMPB     11(R6), ACMODE                     ; 0976
                           49   1A 00024    BGTRU    1$                                 ;
             7E       11   A6   9A 00026    MOVZBL   17(R6), -(SP)                      ; 0980
             6E            17   C0 0002A    ADDL2    #23, (SP)                          ;
    FCCA     CF            01   FB 0002D    CALLS    #1, UTIL_ALLOCATE                  ;
```

```
                    57              50 D0 00032          MOVL    R0, TABLE                          ;
                    50        11    A6 9A 00035          MOVZBL  17(R6), R0                         ; 0981
                                    50 D6 00039          INCL    R0                                 ;
          16   A7   11  A6          50 28 0003B          MOVC3   R0, 17(R6), 22(TABLE)              ;
                    14  A7    0B    A6 90 00041          MOVB    11(R6), 20(TABLE)                  ; 0982
                    15  A7          68 90 00046          MOVB    (R8), 21(TABLE)                    ; 0983
                    08  A7    1D    A8 7D 0004A          MOVQ    29(R8), 8(TABLE)                   ; 0985
                                    0C DD 0004F          PUSHL   #12                                ; 0990
                  FCA6  CF          01 FB 00051          CALLS   #1, UTIL_ALLOCATE                  ;
                    04  A0          58 D0 00056          MOVL    R8, 4(KNOWN)                       ; 0991
                    08  A0          57 D0 0005A          MOVL    TABLE, 8(KNOWN)                    ; 0992
                                    50 DD 0005E          PUSHL   KNOWN                              ; 0993
                            0000'   CF 9F 00060          PUSHAB  KNOWN_LIST                         ;
                                    50 DD 00064          PUSHL   KNOWN                              ;
                  FCE1  CF          03 FB 00066          CALLS   #3, UTIL_SLIDE                     ;
                        50          57 D0 0006B          MOVL    TABLE, R0                          ; 0994
                                       04 0006E          RET                                        ;
                                    50 D4 0006F 1$:      CLRL    R0                                 ; 0997
                                       04 00071          RET                                        ;

; Routine Size:  114 bytes,    Routine Base:  $CODE$ + 0314


;   908      0998  1
;   909      0999  1
;   910      1000  1 ! This little routine determines whether a lnmth table name
;   911      1001  1 ! matches any of the table names specified by the user.
;   912      1002  1 ! It is called via the routine for_each_lnmth.
;   913      1003  1 !
;   914      1004  1 routine match_lnmth( lnmth: ref $bblock, list: ref cmdtbl_block ): novalue =
;   915      1005  2     begin
;   916      1006  2     bind name = list[cmdtbl_q_name]: $bblock[dsc$c_d_bln];
;   917      1007  2     bind lnmb = .lnmth[lnmth$l_name]: $bblock;
;   918      1008  2     local table: ref table_block;
;   919      1009  2     if fmg$match_name(.lnmb[lnmb$t_name], 1+lnmb[lnmb$t_name],
;   920      1010  2         .name[dsc$w_length], .name[dsc$a_pointer]) then
;   921      1011  2     if (table = add_to_known(lnmth[base_])) neq 0
;   922      1012  2     then
;   923      1013  2         util_slide(table[table_l_next], list[cmdtbl_l_table], table[base_]);
;   924      1014  1     end;



                                  007C 00000 MATCH_LNMTH:
                                             .WORD   Save R2,R3,R4,R5,R6                ; 1004
          51        08  AC          08 C1 00002          ADDL3   #8, LIST, R1                       ; 1006
                        56    04    AC D0 00007          MOVL    LNMTH, R6                          ; 1007
                        50    09    A6 D0 0000B          MOVL    9(R6), R0                          ;
                        53    12    A0 9E 0000F          MOVAB   18(R0), R3                         ; 1009
                        55    04    A1 D0 00013          MOVL    4(R1), R5                          ;
                        54          61 3C 00017          MOVZWL  (R1), R4                           ;
                        52    11    A0 9A 0001A          MOVZBL  17(R0), R2                         ;
                         00000000G  00 16 0001E          JSB     FMG$MATCH_NAME                     ;
                        19          50 E9 00024          BLBC    R0, 1$                             ;
                                    56 DD 00027          PUSHL   R6                                 ; 1011
```

```
                 FF60   CF      01  FB 00029          CALLS   #1, ADD_TO_KNOWN                 ;
                                50  D5 0002E          TSTL    TABLE                            ;
                                0E  13 00030          BEQL    1$                               ;
                                50  DD 00032          PUSHL   TABLE                            ; 1013
            7E     08   AC      04  C1 00034          ADDL3   #4, LIST, -(SP)                  ;
                                50  DD 00039          PUSHL   TABLE                            ;
                 FC9A   CF      03  FB 0003B          CALLS   #3, UTIL_SLIDE                   ;
                                    04 00040 1$:      RET                                      ; 1014

; Routine Size:  65 bytes,    Routine Base:  $CODE$ + 0386

;    1
;    1 ! This comparison routine is used for sorting table_blocks.
;    1 ! It is called via the routine sort_list.
;    1 !
;    1 routine cmp_table( t1: ref table_block, t2: ref table_block ): lnk_compare =
;    2     begin
;    2     return ch$compare(
;    2         .t1[table_t_name], 1+t1[table_t_name],
;    2         .t2[table_t_name], 1+t2[table_t_name]);
;    1     end;



                                54  DD 00000 CMP_TABLE:
                                                      PUSHL   R4                               ; 1020
                      53    16  A0  9A 00002          MOVZBL  22(T1), R3                       ; 1023
                      52    16  A1  9A 00006          MOVZBL  22(T2), R2                       ; 1024
                      54        01  D0 0000A          MOVL    #1, R4                           ; 1022
      52       00  17   A0      53  2D 0000D          CMPC5   R3, 23(T1), #0, R2, 23(T2)       ;
                            17  A1     00013                                                   ;
                                03  1A 00015          BGTRU   1$                               ;
                      54        01  D9 00017          SBWC    #1, R4                           ;
                      50        54  D0 0001A 1$:      MOVL    R4, R0                           ;
                                10  BA 0001D          POPR    #^M<R4>                          ; 1025
                                    05 0001F          RSB                                      ;

; Routine Size:  32 bytes,    Routine Base:  $CODE$ + 03C7

;    1
;    1 ! This comparison routine is used for sorting log_blocks.
;    1 ! It is called via the routine sort_list.
;    1 !
;    1 routine cmp_log( t1: ref log_block, t2: ref log_block ): lnk_compare =
;    2     begin
;    2     local
;    2         s;
;    2     s = ch$compare(
;    2         .t1[log_t_name], 1+t1[log_t_name],
;    2         .t2[log_t_name], 1+t2[log_t_name]);
;    2     if .s eql 0
;    2     then
```

```
;  949      1039 3         begin
;  950      1040 3         t1[log_v_equal] = true;
;  951      1041 3         t2[log_v_equal] = true;
;  952      1042 2         end;
;  953      1043 2     return .s;
;  954      1044 1     end;


                              30 BB 00000 CMP_LOG:PUSHR   #^M<R4,R5>                        ; 1030
                    55        51 D0 00002         MOVL    R1, R5                            ;
                              50 DD 00005         PUSHL   R0                                ;
              50    6E        0A C1 00007         ADDL3   #10, T1, R0                       ; 1035
                    51        60 9A 0000B         MOVZBL  (R0), R1                          ;
                    50    0A  A5 9A 0000E         MOVZBL  10(T2), R0                        ; 1036
                    54        01 D0 00012         MOVL    #1, R4                            ; 1034
              7E    6E        0B C1 00015         ADDL3   #11, T1, -(SP)                    ;
      50      00    9E        51 2D 00019         CMPC5   R1, @(SP)+, #0, R0, 11(T2)        ;
                          0B  A5    0001E                                                   ;
                              03 1A 00020         BGTRU   1$                                ;
                    54        01 D9 00022         SBWC    #1, R4                            ;
                    50        54 D0 00025 1$:     MOVL    R4, S                             ;
                              0D 12 00028         BNEQ    2$                                ; 1037
              51    6E        08 C1 0002A         ADDL3   #8, T1, R1                        ; 1040
                    61    80  8F 88 0002E         BISB2   #128, (R1)                        ;
              08    A5    80  8F 88 00032         BISB2   #128, 8(T2)                       ; 1041
                    5E        04 C0 00037 2$:     ADDL2   #4, SP                            ; 1044
                              30 BA 0003A         POPR    #^M<R4,R5>                        ;
                              05 0003C            RSB                                       ;

; Routine Size:  61 bytes,    Routine Base:  $CODE$ + 03E7


;  955      1045 1
;  956      1046 1
;  957      1047 1 ! This routine creates a log_block to hold the information about a logical name
;  958      1048 1 ! and links it to a table_block.
;  959      1049 1 !
;  960      1050 1 routine new_lnmb(lnmb: ref $bblock, table: ref table_block) =
;  961      1051 2     begin
;  962      1052 2     local
;  963      1053 2         size,
;  964      1054 2         log:    ref log_block,
;  965      1055 2         lnmx:   ref $bblock,
;  966      1056 2         funny:  initial(true);
;  967      1057 2
;  968      1058 2     ! Do we really want this logical name?
;  969      1059 2     !
;  970      1060 2     if .lnmb[lnmb$b_acmode] gtru .acmode then return 0;
;  971      1061 2
;  972      1062 2     ! Find the end of the relevant information
;  973      1063 2     ! Also, make sure we find at least one logical name with a positive index.
;  974      1064 2     !
;  975      1065 2     lnmx = .lnmb[lnmb$t_name] + 1+lnmb[lnmb$t_name];
;  976      1066 2     while not .lnmx[lnmx$v_xend] do
```

```
;  977      1067  3          begin
;  978      1068  3          if .lnmx[lnmx$b_index] geq 0 then funny = false;
;  979      1069  3          lnmx = .lnmx[lnmx$t_xlation] + 1+lnmx[lnmx$t_xlation];
;  980      1070  2          end;
;  981      1071  2      if .funny and not .lnmb[lnmb$v_table] then return 0;
;  982      1072  2
;  983      1073  2      !
;  984      1074  2      ! Now that we know the beginning and end of the relevant information,
;  985      1075  2      ! allocate a log_block to hold it.
;  986      1076  2      !
;  987      1077  2      size = lnmx[base_] - lnmb[lnmb$t_name] + 1+%fieldexpand_(lnmx$v_xend,0);
;  988      1078  2      log = util_allocate(log_k_size + .size);
;  989      1079  2      ch$move(.size, lnmb[lnmb$t_name], log[log_t_name]);
;  990      1080  2      util_slide(log[log_l_next], table[table_l_log], log[base_]);
;  991      1081  2      log[log_l_table] = table[base_];
;  992      1082  2      log[log_b_flags] = .lnmb[lnmb$b_flags];
;  993      1083  2      log[log_b_acmode] = .lnmb[lnmb$b_acmode];
;  994      1084  2      log[log_v_equal] = false;
;  995      1085  2      return log[base_];
;  996      1086  1      end;



                                        00FC 00000 NEW_LNMB:
                                                         .WORD    Save R2,R3,R4,R5,R6,R7                 ; 1050
                  52              01  D0 00002           MOVL     #1, FUNNY                              ; 1051
                  57        04    AC  D0 00005           MOVL     LNMB, R7                               ; 1060
          0000'   CF        0B    A7  91 00009           CMPB     11(R7), ACMODE                         ;
                                  69  1A 0000F           BGTRU    5$                                     ;
                  50        11    A7  9A 00011           MOVZBL   17(R7), R0                             ; 1065
                  51        12 A740   9E 00015           MOVAB    18(R7)[R0], LNMX                       ;
        12        61              02  E0 0001A 1$:       BBS      #2, (LNMX), 3$                         ; 1066
                            01    A1  95 0001E           TSTB     1(LNMX)                                ; 1068
                                  02  19 00021           BLSS     2$                                     ;
                                  52  D4 00023           CLRL     FUNNY                                  ;
                  50        04    A1  9A 00025 2$:       MOVZBL   4(LNMX), R0                            ; 1069
                  51        05 A140   9E 00029           MOVAB    5(LNMX)[R0], LNMX                      ;
                                  EA  11 0002E           BRB      1$                                     ; 1066
                  05              52  E9 00030 3$:       BLBC     FUNNY, 4$                              ; 1071
        42        10    A7        03  E1 00033           BBC      #3, 16(R7), 5$                         ;
                  52        11    A7  9E 00038 4$:       MOVAB    17(R7), R2                             ; 1077
                  52              51  C2 0003C           SUBL2    LNMX, R2                               ;
        52        01              52  C3 0003F           SUBL3    R2, #1, SIZE                           ;
                            0B    A2  9F 00043           PUSHAB   11(SIZE)                               ; 1078
          FBA1    CF              01  FB 00046           CALLS    #1, UTIL_ALLOCATE                      ;
                  56              50  D0 0004B           MOVL     R0, LOG                                ;
   0A     A6      11    A7        52  28 0004E           MOVC3    SIZE, 17(R7), 10(LOG)                  ; 1079
                                  56  DD 00054           PUSHL    LOG                                    ; 1080
        7E        08    AC        10  C1 00056           ADDL3    #16, TABLE, -(SP)                      ;
                                  56  DD 0005B           PUSHL    LOG                                    ;
          FBDA    CF              03  FB 0005D           CALLS    #3, UTIL_SLIDE                         ;
            04    A6        08    AC  D0 00062           MOVL     TABLE, 4(LOG)                          ; 1081
            08    A6        10    A7  90 00067           MOVB     16(R7), 8(LOG)                         ; 1082
            09    A6        0B    A7  90 0006C           MOVB     11(R7), 9(LOG)                         ; 1083
            08    A6        80    8F  8A 00071           BICB2    #128, 8(LOG)                           ; 1084
```

```
                          50              56  D0 00076            MOVL    LOG, R0                                  ; 1085
                                              04 00079            RET                                              ;
                                          50  D4 0007A 5$:        CLRL    R0                                       ; 1086
                                              04 0007C            RET                                              ;

; Routine Size:  125 bytes,    Routine Base:  $CODE$ + 0424

;  997          1087  1
;  998          1088  1
;  999          1089  1 ! This routine determines whether a lnmb block is relevant, and (if so) creates
; 1000          1090  1 ! a log_block to hold the information, and links it to a table_block.
; 1001          1091  1 ! It is called via the routine for_each_lnmb.
; 1002          1092  1 !
; 1003          1093  1 routine match_lnmb( lnmb: ref $bblock, dummy ): novalue =
; 1004          1094  2     begin
; 1005          1095  2     local
; 1006          1096  2         table:  ref table_block;
; 1007          1097  2
; 1008          1098  2     ! Verify that this is one of the desired logical names
; 1009          1099  2     !    It must be in an acceptable integrity mode
; 1010          1100  2     !    It must be in one of the specified tables
; 1011          1101  2     !    It must match one of the logical names from the command line
; 1012          1102  2     !
; 1013          1103  3     begin
; 1014          1104  3     local
; 1015          1105  3         list:   ref cmdtbl_block;
; 1016          1106  3
; 1017          1107  3     if .lnmb[lnmb$b_acmode] gtru .acmode then return;
; 1018          1108  3     if (table = lookup_table(.lnmb[lnmb$l_table])) eql 0 then return;
; 1019          1109  3
; 1020          1110  3     list = cmdnam_list - %fieldexpand_(cmdtbl_l_next,0);
; 1021          1111  4     if ( while (list = .list[cmdtbl_l_next]) neq 0 do
; 1022          1112  5         begin
; 1023          1113  5         bind name = list[cmdtbl_q_name]: $bblock[dsc$c_d_bln];
; 1024          1114  5         if fmg$match_name(.lnmb[lnmb$t_name], 1+lnmb[lnmb$t_name],
; 1025          1115  5             .name[dsc$w_length], .name[dsc$a_pointer])
; 1026          1116  5         then
; 1027          1117  5             exitloop false;
; 1028          1118  4         end )
; 1029          1119  3     then
; 1030          1120  3         return;
; 1031          1121  2     end;
; 1032          1122  2
; 1033          1123  2     ! Link the new lnmb block to the table
; 1034          1124  2     !
; 1035          1125  2     new_lnmb( lnmb[base_], table[base_] );
; 1036          1126  2
; 1037          1127  1     end;



                                         03FC 00000 MATCH_LNMB:
                                                             .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9                  ; 1093
                          57         04   AC  D0 00002        MOVL    LNMB, R7                                      ; 1107
```

```
        0000'  CF        0B   A7  91 00006        CMPB    11(R7), ACMODE                     ;
                              3F  1A 0000C        BGTRU   2$                                 ;
                         0C   A7  DD 0000E        PUSHL   12(R7)                             ; 1108
        FE42   CF             01  FB 00011        CALLS   #1, LOOKUP_TABLE                   ;
               59             50  D0 00016        MOVL    R0, TABLE                          ;
               32             13     00019        BEQL    2$                                 ;
               56      0000'  CF  9E 0001B        MOVAB   CMDNAM_LIST, LIST                  ; 1110
               58        12   A7  9E 00020        MOVAB   18(R7), R8                         ; 1114
               56             66  D0 00024 1$:    MOVL    (LIST), LIST                       ; 1111
                              24  13 00027        BEQL    2$                                 ;
               50        08   A6  9E 00029        MOVAB   8(LIST), R0                        ; 1113
               55        04   A0  D0 0002D        MOVL    4(R0), R5                          ; 1114
               54             60  3C 00031        MOVZWL  (R0), R4                           ;
               53             58  D0 00034        MOVL    R8, R3                             ;
               52        11   A7  9A 00037        MOVZBL  17(R7), R2                         ;
                     00000000G    00  16 0003B        JSB     FMG$MATCH_NAME                     ;
               E0             50  E9 00041        BLBC    R0, 1$                             ;
                       0280   8F  BB 00044        PUSHR   #^M<R7,R9>                         ; 1125
        FF36   CF             02  FB 00048        CALLS   #2, NEW_LNMB                       ;
                                  04 0004D 2$:    RET                                        ; 1127

; Routine Size:  78 bytes,    Routine Base:  $CODE$ + 04A1


; 1038          1128  1
; 1039          1129  1
; 1040          1130  1
; 1041          1131  1 ! Get the descendants of the lnmth block, and add them to the table block
; 1042          1132  1 !
; 1043          1133  1 routine get_descend(
; 1044          1134  1         table:  ref table_block,
; 1045          1135  1         lnmth:  ref $bblock
; 1046          1136  1         ): novalue =
; 1047          1137  2     begin
; 1048          1138  2     local
; 1049          1139  2         sublnmth:       ref $bblock;
; 1050          1140  2     sublnmth = lnmth[lnmth$l_child] - %fieldexpand_(lnmth$l_sibling,0);
; 1051          1141  2     while (sublnmth = .sublnmth[lnmth$l_sibling]) neq 0 do
; 1052          1142  3         begin
; 1053          1143  3         local
; 1054          1144  3             subtable:   ref table_block;
; 1055          1145  3         if (subtable = add_to_known(sublnmth[base_])) neq 0
; 1056          1146  3         then
; 1057          1147  4             begin
; 1058          1148  4             util_slide(subtable[table_l_next],
; 1059          1149  4                 table[table_l_child], subtable[base_]);
; 1060          1150  4             get_descend(subtable[base_], sublnmth[base_]);
; 1061          1151  4             end
; 1062          1152  3         else
; 1063          1153  4             begin
; 1064          1154  4             !
; 1065          1155  4             ! Note that we keep looking for descendant tables,
; 1066          1156  4             ! even if we've run across one that we've either seen before,
; 1067          1157  4             ! or have no privilege to read.
; 1068          1158  4             ! Note: these get linked to the table, rather than the subtable.
; 1069          1159  4             !
; 1070          1160  4             get_descend(table[base_], sublnmth[base_]);
```

```
; 1071          1161  3                end;
; 1072          1162  2            end;
; 1073          1163  2        sort_list( table[table_l_child], %fieldexpand_(table_l_next,0),
; 1074          1164  2            cmp_table );
; 1075          1165  1        end;



                                                       000C 00000 GET_DESCEND:
                                                                          .WORD   Save R2,R3                              ; 1133
        52         08   AC                 04   C3 00002                  SUBL3   #4, LNMTH, SUBLNMTH                     ; 1140
                        52         15      A2   D0 00007 1$:              MOVL    21(SUBLNMTH), SUBLNMTH                  ; 1141
                                           2B   13 0000B                  BEQL    4$                                      ;
                                           52   DD 0000D                  PUSHL   SUBLNMTH                                ; 1145
                 FE11   CF                 01   FB 0000F                  CALLS   #1, ADD_TO_KNOWN                        ;
                        53                 50   D0 00014                  MOVL    R0, SUBTABLE                            ;
                                           14   13 00017                  BEQL    2$                                      ;
                                           53   DD 00019                  PUSHL   SUBTABLE                                ; 1149
        7E         04   AC                 04   C1 0001B                  ADDL3   #4, TABLE, -(SP)                        ;
                                           53   DD 00020                  PUSHL   SUBTABLE                                ;
                 FB4A   CF                 03   FB 00022                  CALLS   #3, UTIL_SLIDE                          ;
                                           52   DD 00027                  PUSHL   SUBLNMTH                                ; 1150
                                           53   DD 00029                  PUSHL   SUBTABLE                                ;
                                           05   11 0002B                  BRB     3$                                      ;
                                           52   DD 0002D 2$:              PUSHL   SUBLNMTH                                ; 1160
                                   04      AC   DD 0002F                  PUSHL   TABLE                                   ;
                   CA   AF                 02   FB 00032 3$:              CALLS   #2, GET_DESCEND                         ;
                                           CF   11 00036                  BRB     1$                                      ; 1141
                                 FE9C      CF   9F 00038 4$:              PUSHAB  CMP_TABLE                               ; 1163
                                           7E   D4 0003C                  CLRL    -(SP)                                   ;
        7E         04   AC                 04   C1 0003E                  ADDL3   #4, TABLE, -(SP)                        ;
                 FB8E   CF                 03   FB 00043                  CALLS   #3, SORT_LIST                           ;
                                                04 00048                  RET                                             ; 1165

; Routine Size:  73 bytes,    Routine Base:  $CODE$ + 04EF


; 1076          1166  1
; 1077          1167  1
; 1078          1168  1  ! Indices for getting to the process and directory tables via lnm$al_dirtbl.
; 1079          1169  1  !
; 1080          1170  1  literal
; 1081          1171  1      k_process_directory = 1,
; 1082          1172  1      k_system_directory  = 0;
; 1083          1173  1
; 1084          1174  1  ! A little routine used by for_each_lnmth, below.
; 1085          1175  1  !
; 1086          1176  1  routine fe_lnmth(lnmth: ref $bblock, rtn, prm): novalue =
; 1087          1177  2      begin
; 1088          1178  2      local
; 1089          1179  2          p:       ref $bblock;
; 1090          1180  2      p = lnmth - %fieldexpand_(lnmth$l_sibling,0);
; 1091          1181  2      while (p = .p[lnmth$l_sibling]) neq 0 do
; 1092          1182  3          begin
; 1093          1183  3          (.rtn)(p[base_], .prm);
```

```
; 1094          1184  3             fe_lnmth(.p[lnmth$l_child], .rtn, .prm);
; 1095          1185  2             end;
; 1096          1186  1         end;



                                       0004 00000 FE_LNMTH:
                                                             .WORD    Save R2                             ; 1176
                     52        EF    AC   9E 00002           MOVAB    LNMTH-21, P                         ; 1180
                     52        15    A2   D0 00006 1$:       MOVL     21(P), P                            ; 1181
                               16    13   0000A              BEQL     2$                                  ;
                               0C    AC   DD 0000C           PUSHL    PRM                                 ; 1183
                                     52   DD 0000F           PUSHL    P                                   ;
                08     BC            02   FB 00011           CALLS    #2, @RTN                            ;
                       7E      08    AC   7D 00015           MOVQ     RTN, -(SP)                          ; 1184
                               11    A2   DD 00019           PUSHL    17(P)                               ;
                E0     AF            03   FB 0001C           CALLS    #3, FE_LNMTH                        ;
                                     E4   11 00020           BRB      1$                                  ; 1181
                                          04 00022 2$:       RET                                          ; 1186

; Routine Size:  35 bytes,    Routine Base:  $CODE$ + 0538

; 1097          1187  1
; 1098          1188  1 ! This routine loops over all lnmth blocks,
; 1099          1189  1 ! and calls a supplied routine for each.
; 1100          1190  1 !
; 1101          1191  1 routine for_each_lnmth(rtn,prm): novalue =
; 1102          1192  2     begin
; 1103          1193  2     decr i from k_process_directory to k_system_directory do
; 1104          1194  3         begin
; 1105          1195  3         local
; 1106          1196  3             lnmth:      ref $bblock,
; 1107          1197  3             lnmx:       ref $bblock,
; 1108          1198  3             lnmb:       ref $bblock;
; 1109          1199  3         lnmb = ..lnm$al_dirtbl[.i];
; 1110          1200  3         lnmx = .lnmb[lnmb$t_name] + 1+lnmb[lnmb$t_name];
; 1111          1201  3         lnmth = 1+lnmx[lnmx$t_xlation];
; 1112          1202  3         fe_lnmth(lnmth[base_], .rtn, .prm);
; 1113          1203  2         end;
; 1114          1204  1     end;



                                       0004 00000 FOR_EACH_LNMTH:
                                                             .WORD    Save R2                             ; 1191
                     52              01   D0 00002           MOVL     #1, I                               ; 1202
                     50  00000000G0042    D0 00005 1$:       MOVL     LNM$AL_DIRTBL[I], R0                ; 1199
                     51              60   D0 0000D           MOVL     (R0), LNMB                          ;
                     50        11    A1   9A 00010           MOVZBL   17(LNMB), R0                        ; 1200
                     50        12  A140   9E 00014           MOVAB    18(LNMB)[R0], LNMX                  ;
                     50              05   C0 00019           ADDL2    #5, LNMTH                           ; 1201
                     7E        04    AC   7D 0001C           MOVQ     RTN, -(SP)                          ; 1202
```

```
                                        50  DD 00020          PUSHL   LNMTH                                   ;
                   B7     AF            03  FB 00022          CALLS   #3, FE_LNMTH                            ;
                          DC            52  F4 00026          SOBGEQ  I, 1$                                   ; 1193
                                            04 00029          RET                                             ; 1204

; Routine Size:  42 bytes,    Routine Base:  $CODE$ + 055B

; 1115          1205   1
; 1116          1206   1
; 1117          1207   1  ! This routine loops over all lnmb blocks,
; 1118          1208   1  ! and calls a supplied routine for each.
; 1119          1209   1  !
; 1120          1210   1  routine for_each_lnmb(rtn,prm): novalue =
; 1121          1211   2      begin
; 1122          1212   2
; 1123          1213   2      runs_in_kernel_(for_each_lnmb);
; 1124          1214   2      recheck_tables();
; 1125          1215   2
; 1126          1216   2      decr i from k_process_directory to k_system_directory do
; 1127          1217   3          begin
; 1128          1218   3          local
; 1129          1219   3              bucket:     ref vector,
; 1130          1220   3              lnmhsh:     ref $bblock;
; 1131          1221   3
; 1132          1222   3          lnmhsh = ..lnm$al_hashtbl[.i];
; 1133          1223   3          bucket = lnmhsh[base_] + lnmhsh$k_bucket;
; 1134          1224   3          decr i from not .lnmhsh[lnmhsh$l_mask] to 0 do
; 1135          1225   4              begin
; 1136          1226   4              local
; 1137          1227   4                  lnmb:   ref $bblock;
; 1138          1228   4              lnmb = bucket[.i] - %fieldexpand_(lnmb$l_flink,0);
; 1139          1229   4              while (lnmb = .lnmb[lnmb$l_flink]) neq 0 do
; 1140          1230   5                  begin
; 1141          1231   5                  (.rtn)(lnmb[base_], .prm);
; 1142          1232   4                  end;
; 1143          1233   3              end;
; 1144          1234   2          end;
; 1145          1235   2
; 1146          1236   2      lnm$unlock(.ctl$gl_pcb);
; 1147          1237   2      set_ipl(0);
; 1148          1238   2
; 1149          1239   1      end;


                                                              .EXTRN  SYS$CMKRNL

                                        007C 00000 FOR_EACH_LNMB:
                                                              .WORD   Save R2,R3,R4,R5,R6                     ; 1210
                          56 00000000G  00  9E 00002          MOVAB   CTL$GL_PCB, R6                          ;
                                        50  DC 00009          MOVPSL  PSL                                     ; 1213
         00          50           02    18  ED 0000B          CMPZV   #24, #2, PSL, #0                        ;
                                        07  12 00010          BNEQ    1$                                      ;
                     0000'  CF          5D  D0 00012          MOVL    FP, SAVED_FP                            ;
                                        11  11 00017          BRB     2$                                      ;
                                        5C  DD 00019 1$:      PUSHL   AP                                      ;
```

```
                          E2   AF  9F 0001B         PUSHAB  FOR_EACH_LNMB                             ;
       00000000G   00          02  FB 0001E         CALLS   #2, SYS$CMKRNL                            ;
                     0000'     CF  D4 00025         CLRL    SAVED_FP                                  ;
                               04     00029         RET                                               ;
              54               66  D0 0002A 2$:     MOVL    CTL$GL_PCB, R4                            ;
                     00000000G 00  16 0002D         JSB     LNM$LOCKR                                 ;
      FCD4   CF                00  FB 00033         CALLS   #0, RECHECK_TABLES                        ; 1214
              55               01  D0 00038         MOVL    #1, I                                     ; 1231
              50  00000000G0045    D0 0003B 3$:     MOVL    LNM$AL_HASHTBL[I], R0                     ; 1222
              50               60  D0 00043         MOVL    (R0), LNMHSH                              ;
              52          0C   A0  9E 00046         MOVAB   12(R0), BUCKET                            ; 1223
              53               60  D2 0004A         MCOML   (LNMHSH), R3                              ; 1224
                               53  D6 0004D         INCL    I                                         ; 1228
                               14  11 0004F         BRB     6$                                        ;
              54             6243  DE 00051 4$:     MOVAL   (BUCKET)[I], LNMB                         ;
              54               64  D0 00055 5$:     MOVL    (LNMB), LNMB                              ; 1229
                               0B  13 00058         BEQL    6$                                        ;
                          08   AC  DD 0005A         PUSHL   PRM                                       ; 1231
                               54  DD 0005D         PUSHL   LNMB                                      ;
        04   BC                02  FB 0005F         CALLS   #2, @RTN                                  ;
                               F0  11 00063         BRB     5$                                        ; 1229
              E9               53  F4 00065 6$:     SOBGEQ  I, 4$                                     ; 1224
              D0               55  F4 00068         SOBGEQ  I, 3$                                     ; 1216
              54               66  D0 0006B         MOVL    CTL$GL_PCB, R4                            ; 1236
                     00000000G 00  16 0006E         JSB     LNM$UNLOCK                                ;
              12               00  DA 00074         MTPR    #0, #18                                   ; 1237
                               04     00077         RET                                               ; 1239

; Routine Size:  120 bytes,    Routine Base:  $CODE$ + 0585
```

```
; 1151          1240  1  ! Routines to dump the information about tables and logical names.
; 1152          1241  1  !
; 1153          1242  1  ! Because of the multiplicity of formats, the following technique is used
; 1154          1243  1  ! to produce an fao string and its parameters:
; 1155          1244  1  !       Two buffers are used, the fao control string is built in one,
; 1156          1245  1  !       and (in parallel) the fao parameters are built in the other.
; 1157          1246  1  !
; 1158          1247  1  ! The following macros are used to append stuff to these buffers.
; 1159          1248  1  !
; 1160          1249  1  macro
; 1161          1250  1      ac_(x) = uplit byte(%ascic x) %,
; 1162          1251  1      appstr_(y) = append_str(args[0], ac_(y)) %,
; 1163          1252  1      apparg_(y) = append_arg(args[0], y) %;
; 1164          1253  1
; 1165          1254  1  routine append_str(args: ref vector, ascic: ref vector[,byte]): novalue =
; 1166          1255  2      begin
; 1167          1256  2      bind desc = .args[1]: vector[2];
; 1168          1257  2      ch$move(.ascic[0], ascic[1], .desc[0]+.desc[1]);
; 1169          1258  2      desc[0] = .desc[0] + .ascic[0];
; 1170          1259  1      end;


                                        00FC 00000 APPEND_STR:
                                                       .WORD   Save R2,R3,R4,R5,R6,R7            ; 1254
                         50     04  AC  D0 00002       MOVL    ARGS, R0                          ; 1256
                         56     04  A0  D0 00006       MOVL    4(R0), R6                         ;
                         57     08  AC  D0 0000A       MOVL    ASCIC, R7                         ; 1257
                         51         67  9A 0000E       MOVZBL  (R7), R1                          ;
              50         66     04  A6  C1 00011       ADDL3   4(R6), (R6), R0                   ;
              60     01  A7         51  28 00016       MOVC3   R1, 1(R7), (R0)                   ;
                         50         67  9A 0001B       MOVZBL  (R7), R0                          ; 1258
                         66         50  C0 0001E       ADDL2   R0, (R6)                          ;
                                        04 00021       RET                                       ; 1259

; Routine Size:  34 bytes,    Routine Base:  $CODE$ + 05FD


; 1171          1260  1
; 1172          1261  1  routine append_arg(args: ref vector, y): novalue =
; 1173          1262  2      begin
; 1174          1263  2      args[0] = .args[0] + 1;
; 1175          1264  2      args[.args[0]] = .y;
; 1176          1265  1      end;


                                        0000 00000 APPEND_ARG:
                                                       .WORD   Save nothing                      ; 1261
                                04  BC  D6 00002       INCL    @ARGS                             ; 1263
                         50     04  BC  D0 00005       MOVL    @ARGS, R0                         ; 1264
                    04 BC40     08  AC  D0 00009       MOVL    Y, @ARGS[R0]                      ;
                                        04 0000F       RET                                       ; 1265
```

```
; Routine Size:  16 bytes,    Routine Base:  $CODE$ + 061F

; 1177          1266  1
; 1178          1267  1 ! Dump the quota for a logical name table
; 1179          1268  1 !
; 1180          1269  1 routine dump_table_quot(table: ref table_block, args: ref vector): novalue =
; 1181          1270  2     begin
; 1182          1271  2     !
; 1183          1272  2     ! Don't dump the quota if this is a directory table,
; 1184          1273  2     ! or if there is no limit on the quota.
; 1185          1274  2     !
; 1186          1275  2     if  (.table[table_b_flags] and lnmth$m_directory) eql 0 and
; 1187          1276  2         .table[table_l_byteslm] neq 0
; 1188          1277  2     then
; 1189          1278  3         begin
; 1190          1279  3         appstr_('  [Quota=(!UL,!UL)]');
; 1191          1280  3         apparg_(.table[table_l_bytes]);
; 1192          1281  3         apparg_(.table[table_l_byteslm]);
; 1193          1282  2         end;
; 1194          1283  1     end;


                                                              .PSECT  $PLIT$,NOWRT,NOEXE,2

2C  4C  55  21  28  3D  61  74  6F  75  51  5B  20  20  13  000D4 P.AAW:  .ASCII  <19>\  [Quota=(!UL,!UL)]\              ;
                                        5D  29  4C  55  21  000E3                                                          ;


                                                              .PSECT  $CODE$,NOWRT,2

                                            0004 00000 DUMP_TABLE_QUOT:
                                                              .WORD   Save R2                                    ; 1269
                          52        04  AC  D0 00002          MOVL    TABLE, R2                                  ; 1275
              24     15   A2            01  E0 00006          BBS     #1, 21(R2), 1$                             ;
                                    08  A2  D5 0000B          TSTL    8(R2)                                      ; 1276
                                        1F  13 0000E          BEQL    1$                                         ;
                               0000'    CF  9F 00010          PUSHAB  P.AAW                                      ; 1279
                                    08  AC  DD 00014          PUSHL   ARGS                                       ;
                     B3   AF            02  FB 00017          CALLS   #2, APPEND_STR                             ;
                                    0C  A2  DD 0001B          PUSHL   12(R2)                                     ; 1280
                                    08  AC  DD 0001E          PUSHL   ARGS                                       ;
                     CB   AF            02  FB 00021          CALLS   #2, APPEND_ARG                             ;
                                    08  A2  DD 00025          PUSHL   8(R2)                                      ; 1281
                                    08  AC  DD 00028          PUSHL   ARGS                                       ;
                     C1   AF            02  FB 0002B          CALLS   #2, APPEND_ARG                             ;
                                            04 0002F 1$:      RET                                                ; 1283

; Routine Size:  48 bytes,    Routine Base:  $CODE$ + 062F

; 1195          1284  1
; 1196          1285  1 ! Dump logical name table attributes
; 1197          1286  1 !
```

```
; 1198      1287 1 routine dump_table_attr(table: ref table_block, args: ref vector): novalue =
; 1199      1288 2     begin
; 1200      1289 2     macro
; 1201      1290 2         f_(x) = (.table[table_b_flags] and %name('lnmth$m_',x)) neq 0 %;
; 1202      1291 2     local
; 1203      1292 2         cnt;
; 1204      1293 2
; 1205      1294 2     ! Save the current number of arguments
; 1206      1295 2     !
; 1207      1296 2     cnt = .args[0];
; 1208      1297 2
; 1209      1298 2     ! Append ascic strings for the various attributes
; 1210      1299 2     !
; 1211      1300 2     ! The following order was chosen, so that they form readable phrases.
; 1212      1301 2     !
; 1213      1302 2     if f_('shareable')  then apparg_(ac_('shareable'));
; 1214      1303 2     if f_('group')      then apparg_(ac_('group'));
; 1215      1304 2     if f_('system')     then apparg_(ac_('system'));
; 1216      1305 2     if f_('directory')  then apparg_(ac_('directory'));
; 1217      1306 2
; 1218      1307 2     ! Determine how many attributes we'll be showing (if any), and
; 1219      1308 2     ! append sufficient fao control strings for all of them.
; 1220      1309 2     !
; 1221      1310 2     if (cnt = .args[0] - .cnt) neq 0
; 1222      1311 2     then
; 1223      1312 3         begin
; 1224      1313 3         appstr_('  [!AC');
; 1225      1314 3         decr i from .cnt-1 to 1 do appstr_(',!AC');
; 1226      1315 3         appstr_(']');
; 1227      1316 2         end;
; 1228      1317 1     end;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

    65 6C 62 61 65 72 61 68 73 09  000E8 P.AAX:  .ASCII  <9>\shareable\          ;
                   70 75 6F 72 67 05  000F2 P.AAY:  .ASCII  <5>\group\              ;
                6D 65 74 73 79 73 06  000F8 P.AAZ:  .ASCII  <6>\system\             ;
    79 72 6F 74 63 65 72 69 64 09  000FF P.ABA:  .ASCII  <9>\directory\          ;
             43 41 21 5B 20 20 06  00109 P.ABB:  .ASCII  <6>\  [!AC\             ;
                      43 41 21 2C 04  00110 P.ABC:  .ASCII  <4>\,!AC\               ;
                               5D 01  00115 P.ABD:  .ASCII  <1>\]\                  ;


                                                        .PSECT  $CODE$,NOWRT,2

                             007C 00000 DUMP_TABLE_ATTR:
                                                .WORD   Save R2,R3,R4,R5,R6          ; 1287
              56      BB   AF  9E 00002         MOVAB   APPEND_ARG, R6               ;
              55   0000'   CF  9E 00006         MOVAB   P.AAX, R5                    ;
              53      04   AC  7D 0000B         MOVQ    TABLE, R3                    ; 1302
              52           64  D0 0000F         MOVL    (R4), CNT                    ; 1296
              05      15   A3  E9 00012         BLBC    21(R3), 1$                   ; 1302
                           30  BB 00016         PUSHR   #^M<R4,R5>                   ;
              66           02  FB 00018         CALLS   #2, APPEND_ARG               ;
```

```
                08        15    A3               02  E1 0001B 1$:     BBC     #2, 21(R3), 2$                      ; 1303
                                           0A    A5  9F 00020         PUSHAB  P.AAY                               ;
                                                 54  DD 00023         PUSHL   R4                                  ;
                                66               02  FB 00025         CALLS   #2, APPEND_ARG                      ;
                08        15    A3               03  E1 00028 2$:     BBC     #3, 21(R3), 3$                      ; 1304
                                           10    A5  9F 0002D         PUSHAB  P.AAZ                               ;
                                                 54  DD 00030         PUSHL   R4                                  ;
                                66               02  FB 00032         CALLS   #2, APPEND_ARG                      ;
                08        15    A3               01  E1 00035 3$:     BBC     #1, 21(R3), 4$                      ; 1305
                                           17    A5  9F 0003A         PUSHAB  P.ABA                               ;
                                                 54  DD 0003D         PUSHL   R4                                  ;
                                66               02  FB 0003F         CALLS   #2, APPEND_ARG                      ;
                52              64               52  C3 00042 4$:     SUBL3   CNT, (R4), CNT                      ; 1310
                                                 1A  13 00046         BEQL    7$                                  ;
                                           21    A5  9F 00048         PUSHAB  P.ABB                               ; 1313
                                                 03  11 0004B         BRB     6$                                  ;
                                           28    A5  9F 0004D 5$:     PUSHAB  P.ABC                               ; 1314
                                                 54  DD 00050 6$:     PUSHL   R4                                  ;
                          DE    A6               02  FB 00052         CALLS   #2, APPEND_STR                      ;
                                F4               52  F5 00056         SOBGTR  I, 5$                               ;
                                           2D    A5  9F 00059         PUSHAB  P.ABD                               ; 1315
                                                 54  DD 0005C         PUSHL   R4                                  ;
                          DE    A6               02  FB 0005E         CALLS   #2, APPEND_STR                      ;
                                                     04 00062 7$:     RET                                         ; 1317

; Routine Size:  99 bytes,    Routine Base:  $CODE$ + 065F


; 1229            1318  1
; 1230            1319  1
; 1231            1320  1 ! Return an ascic string for an access/integrity mode
; 1232            1321  1 !
; 1233            1322  1 routine ac_acmode(mode) =
; 1234            1323  2     begin
; 1235            1324  2     own
; 1236            1325  2         mode_str: vector[4] psect($plit$) preset(
; 1237            1326  2                 [psl$c_kernel]= ac_('kernel'),
; 1238            1327  2                 [psl$c_exec]=   ac_('exec'),
; 1239            1328  2                 [psl$c_super]=  ac_('super'),
; 1240            1329  2                 [psl$c_user]=   ac_('user'));
; 1241            1330  2     if .mode lssu 4
; 1242            1331  2     then
; 1243            1332  2         return .mode_str[.mode]
; 1244            1333  2     else
; 1245            1334  2         return ac_('?');
; 1246            1335  1     end;


                                                                      .PSECT  $PLIT$,NOWRT,NOEXE,2

                     6C  65  6E  72  65  6B  06  00117 P.ABE:  .ASCII  <6>\kernel\                            ;
                             63  65  78  65  04  0011E P.ABF:  .ASCII  <4>\exec\                              ;
                         72  65  70  75  73  05  00123 P.ABG:  .ASCII  <5>\super\                             ;
                             72  65  73  75  04  00129 P.ABH:  .ASCII  <4>\user\                              ;
                                                     0012E         .BLKB   2
          00000000' 00000000' 00000000' 00000000'  00130 MODE_STR:
```

```
                                                        .ADDRESS P.ABE, P.ABF, P.ABG, P.ABH                 ;
                               3F  01  00140 P.ABI:     .ASCII  <1>\?\                                      ;

                                                        .PSECT  $CODE$,NOWRT,2

                                 0000 00000 AC_ACMODE:
                                                        .WORD   Save nothing                                ; 1322
               50      04  AC  D0 00002                 MOVL    MODE, R0                                    ; 1330
               04          50  D1 00006                 CMPL    R0, #4                                      ;
                           07  1E 00009                 BGEQU   1$                                          ;
               50    0000'CF40 D0 0000B                 MOVL    MODE_STR[R0], R0                            ; 1334
                               04 00011                 RET                                                 ;
               50    0000'  CF 9E 00012 1$:             MOVAB   P.ABI, R0                                   ;
                               04 00017                 RET                                                 ; 1335

; Routine Size:  24 bytes,    Routine Base:  $CODE$ + 06C2


; 1247          1336  1
; 1248          1337  1 ! Dump the access mode for a logical name table
; 1249          1338  1 !
; 1250          1339  1 routine dump_table_mode(table: ref table_block, args: ref vector): novalue =
; 1251          1340  2     begin
; 1252          1341  2 !   if .table[table_b_acmode] eql psl$c_super then return;
; 1253          1342  2     appstr_('!_[!AC]');
; 1254          1343  2     apparg_(ac_acmode(.table[table_b_acmode]));
; 1255          1344  1     end;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

        5D  43  41  21  5B  5F  21  07  00142 P.ABJ:    .ASCII  <7>\!_[!AC]\                                ;


                                                        .PSECT  $CODE$,NOWRT,2

                                 0000 00000 DUMP_TABLE_MODE:
                                                        .WORD   Save nothing                                ; 1339
                     0000'  CF 9F 00002                 PUSHAB  P.ABJ                                       ; 1342
                        08  AC DD 00006                 PUSHL   ARGS                                        ;
          FF15   CF            02 FB 00009              CALLS   #2, APPEND_STR                              ;
                 50     04  AC D0 0000E                 MOVL    TABLE, R0                                   ; 1343
                 7E     14  A0 9A 00012                 MOVZBL  20(R0), -(SP)                               ;
            CE   AF            01 FB 00016              CALLS   #1, AC_ACMODE                               ;
                               50 DD 0001A              PUSHL   R0                                          ;
                        08  AC DD 0001C                 PUSHL   ARGS                                        ;
          FF21   CF            02 FB 0001F              CALLS   #2, APPEND_ARG                              ;
                               04 00024                 RET                                                 ; 1344

; Routine Size:  37 bytes,    Routine Base:  $CODE$ + 06DA


; 1256          1345  1
```

```
; 1257      1346  1 ! Dump the attributes for a logical name
; 1258      1347  1 !
; 1259      1348  1 routine dump_log_attr(log: ref log_block, args: ref vector): novalue =
; 1260      1349  2     begin
; 1261      1350  2     macro
; 1262      1351  2         f_(x) = (.log[log_b_flags] and %name('lnmb$m_',x)) neq 0 %;
; 1263      1352  2     local
; 1264      1353  2         cnt;
; 1265      1354  2
; 1266      1355  2     cnt = .args[0];
; 1267      1356  2
; 1268      1357  2 !     if .log[log_acmode] neq psl$c_super
; 1269      1358  2 !     then
; 1270      1359  3         begin
; 1271      1360  3         apparg_(ac_acmode(.log[log_b_acmode]));
; 1272      1361  2         end;
; 1273      1362  2
; 1274      1363  2     ! The following are in alphabetical order, for lack of anything better.
; 1275      1364  2     !
; 1276      1365  2     if f_('confine')    then apparg_(ac_('confine'));
; 1277      1366  2     if f_('crelog')     then apparg_(ac_('crelog'));
; 1278      1367  2     if f_('no_alias')   then apparg_(ac_('no_alias'));
; 1279      1368  2     if f_('nodelete')   then apparg_(ac_('nodelete'));
; 1280      1369  2     if f_('table')      then apparg_(ac_('table'));
; 1281      1370  2
; 1282      1371  2     if (cnt = .args[0] - .cnt) neq 0
; 1283      1372  2     then
; 1284      1373  3         begin
; 1285      1374  3         appstr_(' [!AC');
; 1286      1375  3         decr i from .cnt-1 to 1 do appstr_(',!AC');
; 1287      1376  3         appstr_(']');
; 1288      1377  2         end;
; 1289      1378  1     end;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

          65 6E 69 66 6E 6F 63 07  0014A P.ABK:   .ASCII  <7>\confine\            ;
             67 6F 6C 65 72 63 06  00152 P.ABL:   .ASCII  <6>\crelog\             ;
    73 61 69 6C 61 5F 6F 6E 08     00159 P.ABM:   .ASCII  <8>\no_alias\           ;
    65 74 65 6C 65 64 6F 6E 08     00162 P.ABN:   .ASCII  <8>\nodelete\           ;
                65 6C 62 61 74 05  0016B P.ABO:   .ASCII  <5>\table\              ;
                43 41 21 5B 20 05  00171 P.ABP:   .ASCII  <5>\ [!AC\              ;
                   43 41 21 2C 04  00177 P.ABQ:   .ASCII  <4>\,!AC\               ;
                            5D 01  0017C P.ABR:   .ASCII  <1>\]\                  ;


                                                        .PSECT  $CODE$,NOWRT,2

                              007C 00000 DUMP_LOG_ATTR:
                                                  .WORD   Save R2,R3,R4,R5,R6     ; 1348
          56     0000'  CF  9E 00002              MOVAB   P.ABK, R6               ;
          55     FF15   CF  9E 00007              MOVAB   APPEND_ARG, R5          ;
          53       04   AC  7D 0000C              MOVQ    LOG, R3                 ; 1360
          52            64  D0 00010              MOVL    (R4), CNT               ; 1355
```

```
                 7E      09   A3  9A 00013        MOVZBL  9(R3), -(SP)                    ; 1360
           00A3  C5           01  FB 00017        CALLS   #1, AC_ACMODE                   ;
                              50  DD 0001C        PUSHL   R0                              ;
                              54  DD 0001E        PUSHL   R4                              ;
                 65           02  FB 00020        CALLS   #2, APPEND_ARG                  ;
                 53      08   A3  9A 00023        MOVZBL  8(R3), R3                       ; 1365
       07        53           01  E1 00027        BBC     #1, R3, 1$                      ;
                       0050   8F  BB 0002B        PUSHR   #^M<R4,R6>                      ;
                 65           02  FB 0002F        CALLS   #2, APPEND_ARG                  ;
       08        53           02  E1 00032 1$:    BBC     #2, R3, 2$                      ; 1366
                         08   A6  9F 00036        PUSHAB  P.ABL                           ;
                              54  DD 00039        PUSHL   R4                              ;
                 65           02  FB 0003B        CALLS   #2, APPEND_ARG                  ;
                 08           53  E9 0003E 2$:    BLBC    R3, 3$                          ; 1367
                         0F   A6  9F 00041        PUSHAB  P.ABM                           ;
                              54  DD 00044        PUSHL   R4                              ;
                 65           02  FB 00046        CALLS   #2, APPEND_ARG                  ;
       08        53           04  E1 00049 3$:    BBC     #4, R3, 4$                      ; 1368
                         18   A6  9F 0004D        PUSHAB  P.ABN                           ;
                              54  DD 00050        PUSHL   R4                              ;
                 65           02  FB 00052        CALLS   #2, APPEND_ARG                  ;
       08        53           03  E1 00055 4$:    BBC     #3, R3, 5$                      ; 1369
                         21   A6  9F 00059        PUSHAB  P.ABO                           ;
                              54  DD 0005C        PUSHL   R4                              ;
                 65           02  FB 0005E        CALLS   #2, APPEND_ARG                  ;
       52        64           52  C3 00061 5$:    SUBL3   CNT, (R4), CNT                  ; 1371
                              1A  13 00065        BEQL    8$                              ;
                         27   A6  9F 00067        PUSHAB  P.ABP                           ; 1374
                              03  11 0006A        BRB     7$                              ;
                         2D   A6  9F 0006C 6$:    PUSHAB  P.ABQ                           ; 1375
                              54  DD 0006F 7$:    PUSHL   R4                              ;
             DE  A5           02  FB 00071        CALLS   #2, APPEND_STR                  ;
                 F4           52  F5 00075        SOBGTR  I, 6$                           ;
                         32   A6  9F 00078        PUSHAB  P.ABR                           ; 1376
                              54  DD 0007B        PUSHL   R4                              ;
             DE  A5           02  FB 0007D        CALLS   #2, APPEND_STR                  ;
                                  04 00081 8$:    RET                                     ; 1378

; Routine Size:  130 bytes,    Routine Base:  $CODE$ + 06FF


; 1290          1379  1
; 1291          1380  1
; 1292          1381  1  ! Dump translation attributes
; 1293          1382  1  !
; 1294          1383  1  routine dump_lnmx_attr(lnmx: ref $bblock, args: ref vector): novalue =
; 1295          1384  2      begin
; 1296          1385  2      macro
; 1297          1386  2          f_(x) = .lnmx[%name('lnmx$v_',x)] %;
; 1298          1387  2      local
; 1299          1388  2          cnt;
; 1300          1389  2      cnt = .args[0];
; 1301          1390  2
; 1302          1391  2      ! The following are in alphabetical order, for lack of anything better.
; 1303          1392  2      !
; 1304          1393  2      if f_('concealed')  then apparg_(ac_('concealed'));
; 1305          1394  2      if f_('terminal')   then apparg_(ac_('terminal'));
```

```
; 1306        1395 2      if (cnt = .args[0] - .cnt) neq 0
; 1307        1396 2      then
; 1308        1397 3          begin
; 1309        1398 3          appstr_(' [!AC');
; 1310        1399 3          decr i from .cnt-1 to 1 do appstr_(',!AC');
; 1311        1400 3          appstr_(']');
; 1312        1401 2          end;
; 1313        1402 1      end;


                                                              .PSECT  $PLIT$,NOWRT,NOEXE,2

         64 65 6C 61 65 63 6E 6F 63 09 0017E P.ABS:  .ASCII  <9>\concealed\              ;
            6C 61 6E 69 6D 72 65 74 08 00188 P.ABT:  .ASCII  <8>\terminal\               ;
                        43 41 21 5B 20 05 00191 P.ABU:  .ASCII  <5>\ [!AC\                ;
                           43 41 21 2C 04 00197 P.ABV:  .ASCII  <4>\,!AC\                 ;
                                    5D 01 0019C P.ABW:  .ASCII  <1>\]\                    ;


                                                              .PSECT  $CODE$,NOWRT,2

                                   003C 00000 DUMP_LNMX_ATTR:
                                                              .WORD   Save R2,R3,R4,R5          ; 1383
                   55     FE98  CF  9E 00002              MOVAB   APPEND_ARG, R5            ;
                   54     0000' CF  9E 00007              MOVAB   P.ABS, R4                 ;
                   53       08  AC  D0 0000C              MOVL    ARGS, R3                  ; 1389
                   52           63  D0 00010              MOVL    (R3), CNT                 ;
                   05       04  BC  E9 00013              BLBC    @LNMX, 1$                 ; 1393
                                18  BB 00017              PUSHR   #^M<R3,R4>                ;
                   65           02  FB 00019              CALLS   #2, APPEND_ARG            ;
         08     04 BC           01  E1 0001C 1$:          BBC     #1, @LNMX, 2$             ; 1394
                            0A  A4  9F 00021              PUSHAB  P.ABT                     ;
                                53  DD 00024              PUSHL   R3                        ;
                   65           02  FB 00026              CALLS   #2, APPEND_ARG            ;
         52        63           52  C3 00029 2$:          SUBL3   CNT, (R3), CNT            ; 1395
                                1A  13 0002D              BEQL    5$                        ;
                            13  A4  9F 0002F              PUSHAB  P.ABU                     ; 1398
                                03  11 00032              BRB     4$                        ;
                            19  A4  9F 00034 3$:          PUSHAB  P.ABV                     ; 1399
                                53  DD 00037 4$:          PUSHL   R3                        ;
                DE A5           02  FB 00039              CALLS   #2, APPEND_STR            ;
                   F4           52  F5 0003D              SOBGTR  I, 3$                     ;
                            1E  A4  9F 00040              PUSHAB  P.ABW                     ; 1400
                                53  DD 00043              PUSHL   R3                        ;
                DE A5           02  FB 00045              CALLS   #2, APPEND_STR            ;
                                04     00049 5$:          RET                               ; 1402

; Routine Size:  74 bytes,    Routine Base:  $CODE$ + 0781

; 1314        1403 1
; 1315        1404 1
; 1316        1405 1 ! Dump translation strings and their attributes
; 1317        1406 1 !
; 1318        1407 1 routine dump_log_lnmx(log: ref log_block, args: ref vector): novalue =
```

```
begin
local
    first:  initial(true),
    lnmx:   ref $bblock;
builtin
    ap,
    callg;
lnmx = .log[log_t_name] + 1+log[log_t_name];
while not .lnmx[lnmx$v_xend] do
    begin
    !
    ! Don't dump logical names that have "funny" indices.
    !
    if .lnmx[lnmx$b_index] geq 0 or .lnmx[lnmx$b_index] eql lnmx$c_table
    then
        begin
        local len,adr;
        if .first
            then appstr_(' = "!AF"')
            else appstr_('!_ = "!AF"');
        len = .lnmx[lnmx$t_xlation];
        adr = 1+lnmx[lnmx$t_xlation];
        if .lnmx[lnmx$b_index] eql lnmx$c_table
        then
            len = 0;
        if .len gequ 4 then if .(.adr)<0,16,0> eql %x'001b'
        then
            (len = .len - 4; adr = .adr + 4);
        apparg_(.len);
        apparg_(.adr);
        if .flags[flag_v_full]
        then
            dump_lnmx_attr(lnmx[base_], args[0]);
        if .first and .flags[flag_v_recurse]
        then
            begin
            bind
                table = .log[log_l_table]: table_block;
            appstr_(' (!AC)');
            apparg_(table[table_t_name]);
            end;
        callg(args[0], util_output);

        ! Reinitialize the argument list
        !
        begin
        bind desc = .args[1]: vector[2];
        args[0] = 0;
        apparg_(desc[0]);
        desc[0] = 0;
        end;

        first = false;      ! No longer the first time here
        end;
    lnmx = .lnmx[lnmx$t_xlation] + 1+lnmx[lnmx$t_xlation];
    end;
if not .first then flags[flag_v_found] = true;
```

```
; 1376          1465  1      end;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

          22  46  41  21  22  20  3D  20  08  0019E P.ABX:  .ASCII  <8>\ = "!AF"\                        ;
      22  46  41  21  22  20  3D  5F  21  09  001A7 P.ABY:  .ASCII  <9>\!_= "!AF"\                       ;
              29  43  41  21  28  20  06  001B1 P.ABZ:  .ASCII  <6>\ (!AC)\                              ;


                                                        .PSECT  $CODE$,NOWRT,2

                                   00FC 00000 DUMP_LOG_LNMX:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7             ; 1407
                      57   FE4E   CF   9E 00002          MOVAB   APPEND_ARG, R7                     ;
                      56           01   D0 00007          MOVL    #1, FIRST                          ; 1408
                      51     04    AC   D0 0000A          MOVL    LOG, R1                            ; 1415
                      50     0A    A1   9A 0000E          MOVZBL  10(R1), R0                         ;
                      52     0B  A140   9E 00012          MOVAB   11(R1)[R0], LNMX                   ;
            03        62           02   E1 00017 1$:      BBC     #2, (LNMX), 2$                     ; 1416
                                 00A6   31 0001B          BRW     11$                                ;
                      54     01    A2   98 0001E 2$:      CVTBL   1(LNMX), R4                        ; 1421
                                   09   18 00022          BGEQ    3$                                 ;
                 82   8F           54   91 00024          CMPB    R4, #-126                          ;
                                   03   13 00028          BEQL    3$                                 ;
                                 008B   31 0002A          BRW     10$                                ;
                      55     08    AC   D0 0002D 3$:      MOVL    ARGS, R5                           ; 1426
                      06           56   E9 00031          BLBC    FIRST, 4$                          ; 1425
                          0000'    CF   9F 00034          PUSHAB  P.ABX                              ; 1426
                                   04   11 00038          BRB     5$                                 ;
                          0000'    CF   9F 0003A 4$:      PUSHAB  P.ABY                              ; 1427
                                   55   DD 0003E 5$:      PUSHL   R5                                 ;
                 DE   A7           02   FB 00040          CALLS   #2, APPEND_STR                     ;
                      50     04    A2   9A 00044          MOVZBL  4(LNMX), LEN                       ; 1428
                      53     05    A2   9E 00048          MOVAB   5(R2), ADR                         ; 1429
                 82   8F           54   91 0004C          CMPB    R4, #-126                          ; 1430
                                   02   12 00050          BNEQ    6$                                 ;
                                   50   D4 00052          CLRL    LEN                                ; 1432
                      04           50   D1 00054 6$:      CMPL    LEN, #4                            ; 1433
                                   0B   1F 00057          BLSSU   7$                                 ;
                      1B           63   B1 00059          CMPW    (ADR), #27                         ;
                                   06   12 0005C          BNEQ    7$                                 ;
                      50           04   C2 0005E          SUBL2   #4, LEN                            ; 1435
                      53           04   C0 00061          ADDL2   #4, ADR                            ;
                                   50   DD 00064 7$:      PUSHL   LEN                                ; 1436
                                   55   DD 00066          PUSHL   R5                                 ;
                      67           02   FB 00068          CALLS   #2, APPEND_ARG                     ;
                                   53   DD 0006B          PUSHL   ADR                                ; 1437
                                   55   DD 0006D          PUSHL   R5                                 ;
                      67           02   FB 0006F          CALLS   #2, APPEND_ARG                     ;
            07   0000'    CF       06   E1 00072          BBC     #6, FLAGS, 8$                      ; 1438
                                   24   BB 00078          PUSHR   #^M<R2,R5>                         ; 1440
                 0162     C7       02   FB 0007A          CALLS   #2, DUMP_LNMX_ATTR                 ;
                      1E           56   E9 0007F 8$:      BLBC    FIRST, 9$                          ; 1441
                          0000'    CF   95 00082          TSTB    FLAGS                              ;
```

```
                                  18  18 00086          BGEQ    9$                                ;
             53           04  AC  D0 00088          MOVL    LOG, R3                           ; 1445
                       0000'  CF  9F 0008C          PUSHAB  P.ABZ                             ; 1446
                                  55  DD 00090          PUSHL   R5                                ;
                   DE  A7         02  FB 00092          CALLS   #2, APPEND_STR                    ;
        7E         04  A3         16  C1 00096          ADDL3   #22, 4(R3), -(SP)                 ; 1447
                                  55  DD 0009B          PUSHL   R5                                ;
                       67         02  FB 0009D          CALLS   #2, APPEND_ARG                    ;
                 F9E1  C7         65  FA 000A0 9$:      CALLG   (R5), UTIL_OUTPUT                 ; 1449
                       50     08  AC  D0 000A5          MOVL    ARGS, R0                          ; 1454
                       53     04  A0  D0 000A9          MOVL    4(R0), R3                         ;
                                  60  D4 000AD          CLRL    (R0)                              ; 1455
                                  09  BB 000AF          PUSHR   #^M<R0,R3>                        ; 1456
                       67         02  FB 000B1          CALLS   #2, APPEND_ARG                    ;
                                  63  D4 000B4          CLRL    (R3)                              ; 1457
                                  56  D4 000B6          CLRL    FIRST                             ; 1460
                       50     04  A2  9A 000B8 10$:     MOVZBL  4(LNMX), R0                       ; 1462
                       52  05 A240  9E 000BC          MOVAB   5(LNMX)[R0], LNMX                 ;
                              FF53  31 000C1          BRW     1$                                ; 1416
                       05         56  E8 000C4 11$:     BLBS    FIRST, 12$                        ; 1464
                 0000'  CF        01  88 000C7          BISB2   #1, FLAGS+1                       ;
                                  04 000CC 12$:         RET                                       ; 1465

; Routine Size:  205 bytes,    Routine Base:  $CODE$ + 07CB

; 1377         1466  1
; 1378         1467  1  ! Dump a logical name and its attributes, and translations and their attributes
; 1379         1468  1  !
; 1380         1469  1  routine dump_log(
; 1381         1470  1          log:    ref log_block,
; 1382         1471  1          depth
; 1383         1472  1          ): novalue =
; 1384         1473  2      begin
; 1385         1474  2      local
; 1386         1475  2          buf:    vector[80,byte],
; 1387         1476  2          desc:   vector[2],
; 1388         1477  2          args:   vector[20];
; 1389         1478  2      macro
; 1390         1479  2          f_(x) = (.log[log_b_flags] and %name('lnmb$m_',x)) neq 0 %;
; 1391         1480  2      builtin
; 1392         1481  2          actualcount,
; 1393         1482  2          testbitsc,
; 1394         1483  2          callg;
; 1395         1484  2      args[0] = 0;
; 1396         1485  2      apparg_(desc[0]);
; 1397         1486  2      desc[0] = 0;
; 1398         1487  2      desc[1] = buf[0];
; 1399         1488  2      if actualcount() gequ 2
; 1400         1489  2      then
; 1401         1490  3          begin
; 1402         1491  3          apparg_(.depth);
; 1403         1492  3          if .depth gtr 0
; 1404         1493  3              then appstr_('!UL  "!AC"')
; 1405         1494  3              else appstr_('!+   "!AC"');
; 1406         1495  3          end
; 1407         1496  2      else
```

```
2          appstr_('  "!AC"' );
2      apparg_(log[log_t_name]);
2      if .flags[flag_v_full]
2      then
3          begin
3          !
3          ! Dump all of the attributes
3          !
3          dump_log_attr(log[base_], args[0]);
3          end
2      else if f_('table')
2      then
3          begin
3          !
3          ! Let the user know that this is a table
3          !
3          appstr_(' [table]');
3          end
2      else if .log[log_v_equal]
2      then
3          begin
3          !
3          ! Display the access mode, so the logicals are distinguishable
3          !
3          apparg_(ac_acmode(.log[log_b_acmode]));
3          appstr_(' [!AC]');
2          end;
2      dump_log_lnmx(log[base_], args[0]);
1      end;


                                                 .PSECT  $PLIT$,NOWRT,NOEXE,2

22 43 41 21 22 20 20 4C 55 21 0A  001B8 P.ACA:  .ASCII  <10>\!UL  "!AC"\              ;
22 43 41 21 22 20 20 20 2B 21 0A  001C3 P.ACB:  .ASCII  <10>\!+   "!AC"\              ;
         22 43 41 21 22 20 20 07  001CE P.ACC:  .ASCII  <7>\  "!AC"\                  ;
      5D 65 6C 62 61 74 5B 20 08  001D6 P.ACD:  .ASCII  <8>\ [table]\                 ;
            5D 43 41 21 5B 20 06  001DF P.ACE:  .ASCII  <6>\ [!AC]\                   ;


                                                 .PSECT  $CODE$,NOWRT,2

                       001C 00000 DUMP_LOG:
                                                 .WORD   Save R2,R3,R4                ; 1469
          54   0000'  CF  9E 00002               MOVAB   P.ACA, R4                    ;
          53   FD7C   CF  9E 00007               MOVAB   APPEND_ARG, R3               ;
          5E   FF5C   CE  9E 0000C               MOVAB   -164(SP), SP                 ;
                      7E  D4 00011               CLRL    ARGS                         ; 1484
                 50   AE  9F 00013               PUSHAB  DESC                         ; 1485
                 04   AE  9F 00016               PUSHAB  ARGS                         ;
          63          02  FB 00019               CALLS   #2, APPEND_ARG               ;
                 50   AE  D4 0001C               CLRL    DESC                         ; 1486
       54 AE     58   AE  9E 0001F               MOVAB   BUF, DESC+4                  ; 1487
          02          6C  91 00024               CMPB    (AP), #2                     ; 1488
                      17  1F 00027               BLSSU   2$                           ;
```

```
                         08   AC  DD 00029         PUSHL    DEPTH                          ; 1491
                         04   AE  9F 0002C         PUSHAB   ARGS                           ;
                   63         02  FB 0002F         CALLS    #2, APPEND_ARG                 ;
                         08   AC  D5 00032         TSTL     DEPTH                          ; 1492
                              04  15 00035         BLEQ     1$                             ;
                              54  DD 00037         PUSHL    R4                             ; 1493
                              08  11 00039         BRB      3$                             ;
                         0B   A4  9F 0003B 1$:     PUSHAB   P.ACB                          ; 1494
                              03  11 0003E         BRB      3$                             ;
                         16   A4  9F 00040 2$:     PUSHAB   P.ACC                          ; 1497
                         04   AE  9F 00043 3$:     PUSHAB   ARGS                           ;
              DE   A3         02  FB 00046         CALLS    #2, APPEND_STR                 ;
                   52    04   AC  D0 0004A         MOVL     LOG, R2                        ; 1498
                         0A   A2  9F 0004E         PUSHAB   10(R2)                         ;
                         04   AE  9F 00051         PUSHAB   ARGS                           ;
                   63         02  FB 00054         CALLS    #2, APPEND_ARG                 ;
      0B     0000'   CF       06  E1 00057         BBC      #6, FLAGS, 4$                  ; 1499
                       4004   8F  BB 0005D         PUSHR    #^M<R2,SP>                     ; 1505
            00E0     C3       02  FB 00061         CALLS    #2, DUMP_LOG_ATTR              ;
                              2A  11 00066         BRB      7$                             ; 1499
      05       08    A2       03  E1 00068 4$:     BBC      #3, 8(R2), 5$                  ; 1507
                         1E   A4  9F 0006D         PUSHAB   P.ACD                          ; 1513
                              19  11 00070         BRB      6$                             ;
                         08   A2  95 00072 5$:     TSTB     8(R2)                          ; 1515
                              1B  18 00075         BGEQ     7$                             ;
                   7E    09   A2  9A 00077         MOVZBL   9(R2), -(SP)                   ; 1521
            00A3     C3       01  FB 0007B         CALLS    #1, AC_ACMODE                  ;
                              50  DD 00080         PUSHL    R0                             ;
                         04   AE  9F 00082         PUSHAB   ARGS                           ;
                   63         02  FB 00085         CALLS    #2, APPEND_ARG                 ;
                         27   A4  9F 00088         PUSHAB   P.ACE                          ; 1522
                         04   AE  9F 0008B 6$:     PUSHAB   ARGS                           ;
              DE   A3         02  FB 0008E         CALLS    #2, APPEND_STR                 ;
                       4004   8F  BB 00092 7$:     PUSHR    #^M<R2,SP>                     ; 1524
            01AC     C3       02  FB 00096         CALLS    #2, DUMP_LOG_LNMX              ;
                                  04 0009B         RET                                     ; 1525

; Routine Size:  156 bytes,    Routine Base:  $CODE$ + 0898


; 1437          1526  1
; 1438          1527  1  ! Dump a logical name table and its attributes, and the contained logical names
; 1439          1528  1  !
; 1440          1529  1  routine dump_table(
; 1441          1530  1          table:  ref table_block,
; 1442          1531  1          depth
; 1443          1532  1          ): novalue =
; 1444          1533  2      begin
; 1445          1534  2      local
; 1446          1535  2          buf:    vector[80,byte],
; 1447          1536  2          desc:   vector[2],
; 1448          1537  2          args:   vector[20];
; 1449          1538  2      builtin
; 1450          1539  2          callg;
; 1451          1540  2
; 1452          1541  2      ! Note that we write the name of the table before checking whether there
; 1453          1542  2      ! are actually any logical names to be displayed in that table.
```

```
2          ! The idea for this is that it provides useful feedback about which tables
2          ! were searched.
2          ! This can be changed by including:
2          !
2     !!!!if .flags[flag_v_normal] and .table[table_l_log] eql 0 then return;
2          !
2          ! Display the table name and attributes
2          !
2          if .flags[flag_v_normal] then write_blank_line();
2          args[0] = 0;
2          apparg_(desc[0]);
2          desc[0] = 0;
2          desc[1] = buf[0];
2          if not .flags[flag_v_normal] then appstr_('!#* ');
2          appstr_('(!AC)' );
2          if not .flags[flag_v_normal] then apparg_(4*min(.depth,10));
2          apparg_(table[table_t_name]);
2
2          ! Dump attributes, if needed
2          !
2          if .flags[flag_v_full]
2          then
3              begin
3              dump_table_mode(table[base_], args[0]);
3              dump_table_attr(table[base_], args[0]);
3              dump_table_quot(table[base_], args[0]);
2              end;
2          callg(args[0], util_output);
2
2          ! If there are logical names in this table, display them.
2          !
2          if .table[table_l_log] neq 0
2          then
3              begin
3              local
3                  log:        ref log_block;
3              write_blank_line();
3              sort_list( table[table_l_log], %fieldexpand_(log_l_next,0), cmp_log );
3              log = table[table_l_log] - %fieldexpand_(log_l_next,0);
3              while (log = .log[log_l_next]) neq 0 do
3                  dump_log(log[base_]);
2              end;
1          end;


                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

          20  2A  23  21  04  001E6 P.ACF:  .ASCII  <4>\!#* \                          ;
      29  43  41  21  28  05  001EB P.ACG:  .ASCII  <5>\(!AC)\                         ;


                                        .PSECT  $CODE$,NOWRT,2

                        001C 00000 DUMP_TABLE:
                                        .WORD   Save R2,R3,R4                          ; 1529
```

```
                54   0000'  CF  9E 00002          MOVAB   FLAGS, R4                 ;
                53   F6C1   CF  9E 00007          MOVAB   UTIL_OUTPUT, R3           ;
                5E   FF58   CE  9E 0000C          MOVAB   -168(SP), SP              ;
    07          64          04  E1 00011          BBC     #4, FLAGS, 1$             ; 1551
                     0000'  CF  9F 00015          PUSHAB  BLANK_LINE                ;
                63          01  FB 00019          CALLS   #1, UTIL_OUTPUT           ;
                            6E  D4 0001C 1$:      CLRL    ARGS                      ; 1552
                     50     AE  9F 0001E          PUSHAB  DESC                      ; 1553
                     04     AE  9F 00021          PUSHAB  ARGS                      ;
          061F  C3          02  FB 00024          CALLS   #2, APPEND_ARG            ;
                     50     AE  D4 00029          CLRL    DESC                      ; 1554
            54  AE   58     AE  9E 0002C          MOVAB   BUF, DESC+4               ; 1555
    0C          64          04  E0 00031          BBS     #4, FLAGS, 2$             ; 1556
                     0000'  CF  9F 00035          PUSHAB  P.ACF                     ;
                     04     AE  9F 00039          PUSHAB  ARGS                      ;
          05FD  C3          02  FB 0003C          CALLS   #2, APPEND_STR            ;
                     0000'  CF  9F 00041 2$:      PUSHAB  P.ACG                     ; 1557
                     04     AE  9F 00045          PUSHAB  ARGS                      ;
          05FD  C3          02  FB 00048          CALLS   #2, APPEND_STR            ;
    18          64          04  E0 0004D          BBS     #4, FLAGS, 4$             ; 1558
                50   08     AC  D0 00051          MOVL    DEPTH, R0                 ;
                0A          50  D1 00055          CMPL    R0, #10                   ;
                            03  15 00058          BLEQ    3$                        ;
                50          0A  D0 0005A          MOVL    #10, R0                   ;
    7E          50          02  78 0005D 3$:      ASHL    #2, R0, -(SP)             ;
                     04     AE  9F 00061          PUSHAB  ARGS                      ;
          061F  C3          02  FB 00064          CALLS   #2, APPEND_ARG            ;
    7E      04  AC          16  C1 00069 4$:      ADDL3   #22, TABLE, -(SP)         ; 1559
                     04     AE  9F 0006E          PUSHAB  ARGS                      ;
          061F  C3          02  FB 00071          CALLS   #2, APPEND_ARG            ;
    1E          64          06  E1 00076          BBC     #6, FLAGS, 5$             ; 1563
                            5E  DD 0007A          PUSHL   SP                        ; 1566
                     04     AC  DD 0007C          PUSHL   TABLE                     ;
          06DA  C3          02  FB 0007F          CALLS   #2, DUMP_TABLE_MODE       ;
                            5E  DD 00084          PUSHL   SP                        ; 1567
                     04     AC  DD 00086          PUSHL   TABLE                     ;
          065F  C3          02  FB 00089          CALLS   #2, DUMP_TABLE_ATTR       ;
                            5E  DD 0008E          PUSHL   SP                        ; 1568
                     04     AC  DD 00090          PUSHL   TABLE                     ;
          062F  C3          02  FB 00093          CALLS   #2, DUMP_TABLE_QUOT       ;
                63          6E  FA 00098 5$:      CALLG   ARGS, UTIL_OUTPUT         ; 1570
                50   04     AC  D0 0009B          MOVL    TABLE, R0                 ; 1574
                52   10     A0  9E 0009F          MOVAB   16(R0), R2                ;
                            62  D5 000A3          TSTL    (R2)                      ;
                            22  13 000A5          BEQL    7$                        ;
                     0000'  CF  9F 000A7          PUSHAB  BLANK_LINE                ; 1579
                63          01  FB 000AB          CALLS   #1, UTIL_OUTPUT           ;
                     03E7   C3  9F 000AE          PUSHAB  CMP_LOG                   ; 1580
                            7E  D4 000B2          CLRL    -(SP)                     ;
                            52  DD 000B4          PUSHL   R2                        ;
          00C5  C3          03  FB 000B6          CALLS   #3, SORT_LIST             ;
                52          62  D0 000BB 6$:      MOVL    (LOG), LOG                ; 1582
                            09  13 000BE          BEQL    7$                        ;
                            52  DD 000C0          PUSHL   LOG                       ; 1583
          0898  C3          01  FB 000C2          CALLS   #1, DUMP_LOG              ;
                            F2  11 000C7          BRB     6$                        ;
                            04  04 000C9 7$:      RET                               ; 1585
```

```
; Routine Size:  202 bytes,    Routine Base:  $CODE$ + 0934
```

```
! Find all the relevant logical name tables specified by the user.
! This routine uses cmdtbl_list, and calls add_to_known to create table_blocks.
!
routine get_relevant_tables =
    begin
    local
        list:   ref cmdtbl_block;

    runs_in_kernel_(get_relevant_tables);

    ! Traverse the list, and process each name as either:
    !    A wildcard string (find all table names that match)
    !    A table name (process that table)
    !    A search list (process all relevant tables)
    !
    list = cmdtbl_list - %fieldexpand_(cmdtbl_l_next,0);
    while (list = .list[cmdtbl_l_next]) neq 0 do
        begin
        bind name = list[cmdtbl_q_name]: $bblock[dsc$c_d_bln];
        if not ch$fail(ch$find_ch(.name[dsc$w_length],.name[dsc$a_pointer],'*'))
        or not ch$fail(ch$find_ch(.name[dsc$w_length],.name[dsc$a_pointer],'%'))
        then
            begin
            !
            ! A wildcard string (find all table names that match)
            !
            for_each_lnmth( match_lnmth, list[base_] );
            sort_list( list[cmdtbl_l_table], %fieldexpand_(table_l_next,0),
                cmp_table );
            end
        else
            begin
            !
            ! A table name (process that table)
            ! A search list (process all relevant tables)
            !
            ! In either case, we simply use lnm$setup and lnm$table.
            !
            ! ??? Note that there is a slight problem with the access mode.
            ! If we set it to .acmode, then SHOW LOGICAL /EXEC causes no tables
            ! to be found, since LNM$DCL_LOGICAL is a supervisor mode name.
            ! Instead, we set it to psl$c_user, and let add_to_known check the
            ! access mode of the resulting table.
            !
            local
                an_rt_block:    rt_block,
                lnmth:          ref $bblock,
                status;
            register
                rt = 5: ref rt_block initial(an_rt_block[base_]);
            rt[rt_b_acmode] = psl$c_user;          ! Acceptable integrity mode
            rt[rt_b_flags] = 0;                    ! Initialize flags
            rt[rt_v_case] = true;                  ! Case-insensitive
            status = lnm$setup(.name[dsc$w_length],.name[dsc$a_pointer],
                rt[base_], lnmth);
            while true do
                begin
```

```
                    if not .status
                    then
                        begin
                        if .status eql ss$_toomanylnam
                        then
                            rt[rt_b_depth] = .rt[rt_b_depth] - 1     ! Backup depth
                        else
                            exitloop                    ! Presumably no more names
                        end
                    else
                        begin
                        local table: ref table_block;
                        if (table = add_to_known(lnmth[base_])) neq 0
                        then
                            util_slide(table[table_l_next],
                                list[cmdtbl_l_table], table[base_]);
                        end;
                    rt[rt_b_tries] = rt_c_maxtries; ! Reinitialize number of tries
                    status = lnm$table(rt[base_]; lnmth);
                    end;
                reverse_list(list[cmdtbl_l_table], %fieldexpand_(table_l_next,0));
                end;
            end;

        ! Reverse the order of known_list, so that traversing the list will
        ! visit the tables in the same order as they were found.

        reverse_list(known_list, %fieldexpand_(known_l_next,0));

        ! Now get the descendants, if requested.
        !
        ! Note that this is done after finding all the tables specified on the
        ! command line.  Thus, the user's requested order is honored as much as
        ! possible.  It also avoids anomolous behaviour for "/table=*/descendants".
        !
        ! ??? However, it gives odd behaviour for the command
        ! "/table=lnm$*_directory/descendants/process/group/job/system"
        ! as compared with saying "/table=*dire*/descendants".

        if .flags[flag_v_descend]
        then
            begin
            !
            ! We traverse the known list in the order in which things were found
            !
            local
                known:      ref known_block;
            known = known_list - %fieldexpand_(known_l_next,0);
            while (known = .known[known_l_next]) neq 0 do
                begin
                local
                    new:    ref known_block;
                get_descend(.known[known_l_table], .known[known_l_lnmth]);
                new = known_list - %fieldexpand_(known_l_next,0);
                while .new[known_l_next] neq known[base_] do
                    new = .new[known_l_next];
                if known_list[base_] neq known[base_]        ! Any new tables?
```

```
; 1612      1700  4           then
; 1613      1701  5               begin
; 1614      1702  5               known_list = util_slide(new[known_l_next], known[known_l_next],
; 1615      1703  5                   known_list[base_]);
; 1616      1704  5               known = new[base_];
; 1617      1705  4               end;
; 1618      1706  3           end;
; 1619      1707  2       end;
; 1620      1708  2
; 1621      1709  2   lnm$unlock(.ctl$gl_pcb);
; 1622      1710  2   set_ipl(0);
; 1623      1711  2
; 1624      1712  2   return ss$_normal;
; 1625      1713  1   end;



                                          07FC 00000 GET_RELEVANT_TABLES:
                                                     .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10              ; 1589
                       5A 00000000G   00  9E 00002   MOVAB   CTL$GL_PCB, R10                               ;
                       59     0000'   CF  9E 00009   MOVAB   KNOWN_LIST, R9                                ;
                       58      F650   CF  9E 0000E   MOVAB   UTIL_SLIDE, R8                                ;
                       5E             30  C2 00013   SUBL2   #48, SP                                       ;
                                      50  DC 00016   MOVPSL  PSL                                           ; 1594
      00         50           02      18  ED 00018   CMPZV   #24, #2, PSL, #0                              ;
                                      06  12 0001D   BNEQ    1$                                            ;
                     FC  A9           5D  D0 0001F   MOVL    FP, SAVED_FP                                  ;
                                      11  11 00023   BRB     2$                                            ;
                                      5C  DD 00025 1$: PUSHL AP                                            ;
                               099E   C8  9F 00027   PUSHAB  GET_RELEVANT_TABLES                           ;
            00000000G  00             02  FB 0002B   CALLS   #2, SYS$CMKRNL                                ;
                                 FC   A9  D4 00032   CLRL    SAVED_FP                                      ;
                                          04 00035   RET                                                   ;
                       54             6A  D0 00036 2$: MOVL  CTL$GL_PCB, R4                                ;
                          00000000G   00  16 00039   JSB     LNM$LOCKR                                     ;
                       54        04   A9  9E 0003F   MOVAB   CMDTBL_LIST, LIST                             ; 1601
                       54             64  D0 00043 3$: MOVL  (LIST), LIST                                  ; 1602
                                      03  12 00046   BNEQ    4$                                            ;
                               0094   31 00048       BRW     14$                                           ;
                       56        08   A4  9E 0004B 4$: MOVAB 8(LIST), R6                                   ; 1604
                       57        04   A4  9E 0004F   MOVAB   4(R4), R7                                     ; 1613
            04   B6    66             2A  3A 00053   LOCC    #42, (R6), @4(R6)                             ; 1605
                                      02  12 00058   BNEQ    5$                                            ;
                                      51  D4 0005A   CLRL    R1                                            ;
                                      51  D5 0005C 5$: TSTL  R1                                            ;
                                      0D  12 0005E   BNEQ    7$                                            ;
            04   B6    66             25  3A 00060   LOCC    #37, (R6), @4(R6)                             ; 1606
                                      02  12 00065   BNEQ    6$                                            ;
                                      51  D4 00067   CLRL    R1                                            ;
                                      51  D5 00069 6$: TSTL  R1                                            ;
                                      19  13 0006B   BEQL    9$                                            ;
                                      54  DD 0006D 7$: PUSHL LIST                                          ; 1612
                               0326   C8  9F 0006F   PUSHAB  MATCH_LNMTH                                   ;
                    04FB  C8          02  FB 00073   CALLS   #2, FOR_EACH_LNMTH                            ;
                               0367   C8  9F 00078   PUSHAB  CMP_TABLE                                     ; 1613
```

```
                              7E  D4 0007C        CLRL    -(SP)                         ;
                              57  DD 0007E        PUSHL   R7                            ;
              65    A8        03  FB 00080        CALLS   #3, SORT_LIST                 ;
                              BD  11 00084 8$:    BRB     3$                            ; 1605
                    55        6E  9E 00086 9$:    MOVAB   AN_RT_BLOCK, RT               ; 1635
                    65        03  B0 00089        MOVW    #3, (RT)                      ; 1636
              01    A5        01  88 0008C        BISB2   #1, 1(RT)                     ; 1638
                    53  04    A6  D0 00090        MOVL    4(R6), R3                     ; 1640
                    52        66  3C 00094        MOVZWL  (R6), R2                      ;
                  00000000G   00  16 00097        JSB     LNM$SETUP                     ;
                    52        50  D0 0009D 10$:   MOVL    R0, STATUS                    ;
                    56        51  D0 000A0        MOVL    R1, R6                        ;
                    0E        52  E8 000A3        BLBS    STATUS, 11$                   ; 1643
        00000374    8F        52  D1 000A6        CMPL    STATUS, #884                  ; 1646
                              26  12 000AD        BNEQ    13$                           ;
                        02    A5  97 000AF        DECB    2(RT)                         ; 1648
                              15  11 000B2        BRB     12$                           ;
                              56  DD 000B4 11$:   PUSHL   LNMTH                         ; 1655
            02B4    C8        01  FB 000B6        CALLS   #1, ADD_TO_KNOWN              ;
                              50  D5 000BB        TSTL    TABLE                         ;
                              0A  13 000BD        BEQL    12$                           ;
                              50  DD 000BF        PUSHL   TABLE                         ; 1658
                        04    A4  9F 000C1        PUSHAB  4(LIST)                       ;
                              50  DD 000C4        PUSHL   TABLE                         ;
                    68        03  FB 000C6        CALLS   #3, UTIL_SLIDE                ;
              03    A5        01  8E 000C9 12$:   MNEGB   #1, 3(RT)                     ; 1660
                  00000000G   00  16 000CD        JSB     LNM$TABLE                     ; 1661
                              C8  11 000D3        BRB     10$                           ;
                              7E  D4 000D5 13$:   CLRL    -(SP)                         ; 1663
                              57  DD 000D7        PUSHL   R7                            ;
              46    A8        02  FB 000D9        CALLS   #2, REVERSE_LIST              ;
                              A5  11 000DD        BRB     8$                            ; 1602
                              7E  D4 000DF 14$:   CLRL    -(SP)                         ; 1670
                              59  DD 000E1        PUSHL   R9                            ;
              46    A8        02  FB 000E3        CALLS   #2, REVERSE_LIST              ;
    36        0C    A9        05  E1 000E7        BBC     #5, FLAGS, 18$                ; 1682
                    52        69  9E 000EC        MOVAB   KNOWN_LIST, KNOWN             ; 1690
                    52        62  D0 000EF 15$:   MOVL    (KNOWN), KNOWN                ; 1691
                              2E  13 000F2        BEQL    18$                           ;
                        04    A2  DD 000F4        PUSHL   4(KNOWN)                      ; 1695
                        08    A2  DD 000F7        PUSHL   8(KNOWN)                      ;
            048F    C8        02  FB 000FA        CALLS   #2, GET_DESCEND               ;
                    53        69  9E 000FF        MOVAB   KNOWN_LIST, NEW               ; 1696
                    52        63  D1 00102 16$:   CMPL    (NEW), KNOWN                  ; 1697
                              05  13 00105        BEQL    17$                           ;
                    53        63  D0 00107        MOVL    (NEW), NEW                    ; 1698
                              F6  11 0010A        BRB     16$                           ;
                    52        69  D1 0010C 17$:   CMPL    KNOWN_LIST, KNOWN             ; 1699
                              DE  13 0010F        BEQL    15$                           ;
                              69  DD 00111        PUSHL   KNOWN_LIST                    ; 1703
                              52  DD 00113        PUSHL   KNOWN                         ;
                              53  DD 00115        PUSHL   NEW                           ;
                    68        03  FB 00117        CALLS   #3, UTIL_SLIDE                ;
                    69        50  D0 0011A        MOVL    R0, KNOWN_LIST                ;
                    52        53  D0 0011D        MOVL    NEW, KNOWN                    ; 1704
                              CD  11 00120        BRB     15$                           ; 1691
                    54        6A  D0 00122 18$:   MOVL    CTL$GL_PCB, R4                ; 1709
```

```
              00000000G  00  16 00125          JSB     LNM$UNLOCK                     ;
        12               00  DA 0012B          MTPR    #0, #18                        ; 1710
        50               01  D0 0012E          MOVL    #1, R0                         ; 1712
                             04 00131          RET                                    ; 1713

; Routine Size:  306 bytes,    Routine Base:  $CODE$ + 09FE
```

```
1 routine translate(
1         name_len,
1         name_adr,
1         p_known           ! Address of pointer to a known_block
1         ) =
2     begin
2     !
2     ! This routine translates a logical name in a particular table, and returns
2     ! a pointer to a log_block (or 0).
2     !
2     local
2         proc_nt:          nt_block,
2         syst_nt:          nt_block,
2         known:            ref known_block,
2         log:              ref log_block,
2         status;
2     local
2         nt:     ref nt_block;                                  ! Usually R3
2     label
2         main;
2
2     runs_in_kernel (translate);
2     recheck_tables();
2
2 main:
3     begin
3     !
3     ! Initialize the values we will return to zero
3     !
3     known = 0;
3     log = 0;
3
3     ! Initialize the nt_block for the logical name
3     !
3     decr i from k_process_directory to k_system_directory do
4         begin
4         local
4             lnmb:         ref $bblock;
4
4         if .i eql k_system_directory
4             then nt = syst_nt[base_]
4             else nt = proc_nt[base_];
4
4         ! ??? Note that we set the access mode to .acmode.
4         ! This should be okay, since lnm$presearch and lnm$contsearch only
4         ! check the access mode of the logical name, not the logical name
4         ! table.
4         !
4         nt[nt_b_acmode] = .acmode;          ! Acceptable integrity mode
4         nt[nt_b_flags] = 0;                 ! Initialize flags
4         nt[nt_v_case] = true;               ! Case-insensitive
4         nt[nt_w_hash] = 0;                  ! Hash function or 0
4         nt[nt_l_namlen] = .name_len;        ! Length of name
4         nt[nt_l_namadr] = .name_adr;        ! Address of name
4         nt[nt_l_tabid] = 0;                 ! Table header address or 0
4         nt[nt_l_thread] = 0;                ! Lnmb pointer
4
```

```
            ! Do a presearch on the logical name
            !
            status = lnm$presearch(..lnm$al_hashtbl[.i], nt[base_]; lnmb);
            if not .status then nt[nt_l_thread] = 0;
            end;

        if .proc_nt[nt_l_thread] eql 0 and .syst_nt[nt_l_thread] eql 0
        then
            leave main;

        ! Loop through, trying to find a known table that holds this logical name
        !
        known = .p_known - %fieldexpand_(known_l_next,0);
        while (known = .known[known_l_next]) neq 0 do
            begin
            !
            ! We simply put the address of the lnmth block into the nt_l_tabid
            ! field and call lnm$contsearch.  This is okay, because recheck_tables
            ! has verified that all the tables are still good (or has cleared the
            ! known_l_lnmth field for those that aren't).
            !
            label
                lookup;
lookup:     begin
            !
            ! See if the name is in that table.
            !
            local
                table:      ref table_block,
                lnmth:      ref $bblock,
                lnmb:       ref $bblock,
                save_thread;
            table = .known[known_l_table];
            lnmth = .known[known_l_lnmth];
            if lnmth[base_] eql 0 then leave lookup;         ! Ignore this table?
            if .lnmth[lnmth$v_shareable]                     ! Should we look in
                then nt = syst_nt[base_]                      !  system hash table, or
                else nt = proc_nt[base_];                     !  process hash table?
            nt[nt_l_tabid] = lnmth[base_];                   ! Stuff it here
            if .nt[nt_l_thread] eql 0 then leave lookup;     ! Any possibilities?
            !
            ! Now go look for it
            !
            save_thread = .nt[nt_l_thread];
            status = lnm$contsearch(nt[base_]; lnmb);
            nt[nt_l_thread] = .save_thread;
            if not .status then leave lookup;                ! Check status
            log = new_lnmb(lnmb[base_], table[base_]);       ! Save the lnmb block
            if log[base_] eql 0 then leave lookup;           ! Verify it's acceptable
            leave main;                                      ! We found one!
            end;    ! of lookup

            ! Now go try another table
            !
            0;
            end;
```

```
2       end;            ! of main
2
2       lnm$unlock(.ctl$gl_pcb);
2       set_ipl(0);
2
2       ! Return the address of the log_block,
2       ! and the known_block of the table.
2       !
2       .p_known = known[base_];
2       return log[base_];
1       end;
```

```
                                         01FC 00000 TRANSLATE:
                                                          .WORD   Save R2,R3,R4,R5,R6,R7,R8          ; 1714
                   58 00000000G   00   9E 00002           MOVAB   CTL$GL_PCB, R8                     ;
                   5E             28   C2 00009           SUBL2   #40, SP                            ;
                                  50   DC 0000C           MOVPSL  PSL                                ; 1735
    00          50          02    18   ED 0000E           CMPZV   #24, #2, PSL, #0                   ;
                                  07   12 00013           BNEQ    1$                                 ;
               0000'   CF         5D   D0 00015           MOVL    FP, SAVED_FP                       ;
                                  11   11 0001A           BRB     2$                                 ;
                                  5C   DD 0001C 1$:       PUSHL   AP                                 ;
                          DF      AF   9F 0001E           PUSHAB  TRANSLATE                          ;
              00000000G   00      02   FB 00021           CALLS   #2, SYS$CMKRNL                     ;
                        0000'     CF   D4 00028           CLRL    SAVED_FP                           ;
                                       04 0002C           RET                                        ;
                   54             68   D0 0002D 2$:       MOVL    CTL$GL_PCB, R4                     ;
                      00000000G   00   16 00030           JSB     LNM$LOCKR                          ;
               F726    CF         00   FB 00036           CALLS   #0, RECHECK_TABLES                 ; 1736
                                  57   D4 0003B           CLRL    LOG                                ; 1744
                   54             01   7D 0003D           MOVQ    #1, I                              ; 1767
                                  54   D5 00040 3$:       TSTL    I                                  ; 1753
                                  05   12 00042           BNEQ    4$                                 ;
                   53             6E   9E 00044           MOVAB   SYST_NT, NT                        ; 1754
                                  04   11 00047           BRB     5$                                 ;
                   53       14    AE   9E 00049 4$:       MOVAB   PROC_NT, NT                        ; 1755
                   63     0000'   CF   9B 0004D 5$:       MOVZBW  ACMODE, (NT)                       ; 1762
               01  A3             01   88 00052           BISB2   #1, 1(NT)                          ; 1764
                            02    A3   B4 00056           CLRW    2(NT)                              ; 1765
               04  A3       04    AC   7D 00059           MOVQ    NAME_LEN, 4(NT)                    ; 1766
                            0C    A3   7C 0005E           CLRQ    12(NT)                             ; 1768
                   50 00000000G0044    D0 00061           MOVL    LNM$AL_HASHTBL[I], R0              ; 1773
                   51             60   D0 00069           MOVL    (R0), R1                           ;
                      00000000G   00   16 0006C           JSB     LNM$PRESEARCH                      ;
                   52             50   D0 00072           MOVL    R0, STATUS                         ;
                   03             52   E8 00075           BLBS    STATUS, 6$                         ; 1774
                            10    A3   D4 00078           CLRL    16(NT)                             ;
                   C2             54   F4 0007B 6$:       SOBGEQ  I, 3$                              ; 1748
                   24             AE   D5 0007E           TSTL    PROC_NT+16                         ; 1777
                                  05   12 00081           BNEQ    7$                                 ;
                            10    AE   D5 00083           TSTL    SYST_NT+16                         ;
                                  4A   13 00086           BEQL    11$                                ;
                   55       0C    AC   D0 00088 7$:       MOVL    P_KNOWN, KNOWN                     ; 1783
```

```
              55           65  D0 0008C 8$:     MOVL    (KNOWN), KNOWN                          ; 1784
                           41  13 0008F         BEQL    11$                                      ;
              56       08  A5  D0 00091         MOVL    8(KNOWN), TABLE                          ; 1803
              50       04  A5  D0 00095         MOVL    4(KNOWN), LNMTH                          ; 1804
                           F1  13 00099         BEQL    8$                                       ; 1805
              05           60  E9 0009B         BLBC    (LNMTH), 9$                              ; 1806
              53           6E  9E 0009E         MOVAB   SYST_NT, NT                              ; 1807
                           04  11 000A1         BRB     10$                                      ;
              53       14  AE  9E 000A3 9$:     MOVAB   PROC_NT, NT                              ; 1808
         0C   A3           50  D0 000A7 10$:    MOVL    LNMTH, 12(NT)                            ; 1809
                       10  A3  D5 000AB         TSTL    16(NT)                                   ; 1810
                           DC  13 000AE         BEQL    8$                                       ;
              54       10  A3  D0 000B0         MOVL    16(NT), SAVE_THREAD                      ; 1814
                00000000G  00  16 000B4         JSB     LNM$CONTSEARCH                           ; 1815
              52           50  D0 000BA         MOVL    R0, STATUS                               ;
         10   A3           54  D0 000BD         MOVL    SAVE_THREAD, 16(NT)                      ; 1816
              C8           52  E9 000C1         BLBC    STATUS, 8$                               ; 1817
                     0042  8F  BB 000C4         PUSHR   #^M<R1,R6>                               ; 1818
       F827   CF           02  FB 000C8         CALLS   #2, NEW_LNMB                             ;
              57           50  D0 000CD         MOVL    R0, LOG                                  ;
                           BA  13 000D0         BEQL    8$                                       ; 1819
              54           68  D0 000D2 11$:    MOVL    CTL$GL_PCB, R4                           ; 1830
                00000000G  00  16 000D5         JSB     LNM$UNLOCK                               ;
              12           00  DA 000DB         MTPR    #0, #18                                  ; 1831
         0C   BC           55  D0 000DE         MOVL    KNOWN, @P_KNOWN                          ; 1836
              50           57  D0 000E2         MOVL    LOG, R0                                  ; 1837
                               04 000E5         RET                                              ; 1838

; Routine Size:  230 bytes,    Routine Base:  $CODE$ + 0B30
```

```
1 ! Routine to recursively translate a logical name.
1 !
1 ! Note that while we are in this routine, we use the virtual memory as a stack
1 ! (via util_allocate and util_reallocate).
1 !
1 routine recursively_translate(
1         name_len,
1         name_adr,
1         depth
1         ): novalue =
2     begin
2     local
2         known:  ref known_block,
2         log:    ref log_block,
2         lnmx:   ref $bblock;
2     own
2         transx: vector[lnm$c_maxdepth];
2
2
2     if .depth gequ lnm$c_maxdepth then return;
2
2     ! Set up for the first known table
2     !
2     known = known_list - %fieldexpand_(known_l_next,0);
2     while (known = .known[known_l_next]) neq 0 do
3         begin
3
3         ! Look for the next translation, in a table starting with known.
3         ! If a found, known is set to the known_block in which it was found.
3         !
3         log = translate(.name_len, .name_adr, known);
3         if log[base_] eql 0 then return;
3
3         ! Avoid redisplaying recursive translations (in the current path)
3         !
3         transx[.depth] = log[base_];
3         decr i from .depth-1 to 0 do
3             if .transx[.i] eql log[base_] then return;
3
3         ! Display this translation
3         !
3         dump_log(log[base_], .depth);
3
3         ! Unlink the log_block from the table.
3         ! This isn't really needed.  It's just good practice, since the pointer
3         ! to the log_block is invalidated by the call to util_reallocate.
3         !
4         begin
4         local table: ref table_block;
4         table = .log[log_l_table];
4         table[table_l_log] = 0;
3         end;
3
3         ! Now loop through, doing recursive translations
3         !
3         lnmx = .log[log_t_name] + 1+log[log_t_name];
3         while not .lnmx[lnmx$v_xend] do
```

```
4              begin
4              if .lnmx[lnmx$b_index] geq 0 and not .lnmx[lnmx$v_terminal]
4              then
5                  begin
5                  local len,adr;
5                  len = .lnmx[lnmx$t_xlation];
5                  adr = 1+lnmx[lnmx$t_xlation];
5                  if .len gequ 4 then if .(.adr)<0,16,0> eql %x'001b'
5                  then
5                      (len = .len - 4; adr = .adr + 4);
5
5                  ! If we have a trailing colon, remove it
5                  ! (unless there are actually two trailing colons).
5                  !
5                  if .len gequ 2 then if
5                      ch$rchar(.adr+.len-2) neq ':' and
5                      ch$rchar(.adr+.len-1) eql ':'
5                  then
5                      len = .len - 1;
5
5                  recursively_translate(.len, .adr, .depth+1);
4                  end;
4              lnmx = .lnmx[lnmx$t_xlation] + 1+lnmx[lnmx$t_xlation];
3              end;
3
3          ! Back up our memory allocator.
3          !
3          util_reallocate(log[base_]);
2          end;
1      end;



                                                .PSECT  $OWN$,NOEXE,2

                              0001B             .BLKB   1
                              0001C TRANSX:     .BLKB   40


                                                .PSECT  $CODE$,NOWRT,2

                         007C 00000 RECURSIVELY_TRANSLATE:
                                                .WORD   Save R2,R3,R4,R5,R6                 ; 1844
       5E             04  C2 00002              SUBL2   #4, SP
       55        0C   AC  D0 00005              MOVL    DEPTH, R5                           ; 1858
       0A             55  D1 00009              CMPL    R5, #10
                      01  1F 0000C              BLSSU   1$                                  ;
                      04     0000E              RET                                         ;
       6E      0000'  CF  9E 0000F 1$:          MOVAB   KNOWN_LIST, KNOWN                   ; 1862
       56        FF   A5  9E 00014              MOVAB   -1(R5), R6                          ; 1875
                      9E  DD 00018 2$:          PUSHL   @KNOWN                              ; 1863
                      22  13 0001A              BEQL    4$                                  ;
                      5E  DD 0001C              PUSHL   SP                                  ; 1869
       7E        04   AC  7D 0001E              MOVQ    NAME_LEN, -(SP)                     ;
  FEF3 CF             03  FB 00022              CALLS   #3, TRANSLATE                       ;
       53             50  D0 00027              MOVL    R0, LOG                             ;
```

```
                            12  13 0002A          BEQL    4$                                   ; 1870
      0000'CF45             53  D0 0002C          MOVL    LOG, TRANSX[R5]                      ; 1874
             50       01    A6  9E 00032          MOVAB   1(R6), I                             ; 1875
                            08  11 00036          BRB     5$
             53     0000'CF40   D1 00038 3$:      CMPL    TRANSX[I], LOG                       ; 1876
                            77  13 0003E 4$:      BEQL    11$
             F5             50  F4 00040 5$:      SOBGEQ  I, 3$
                            28  BB 00043          PUSHR   #^M<R3,R5>                           ; 1880
      FC38   CF             02  FB 00045          CALLS   #2, DUMP_LOG
             50       04    A3  D0 0004A          MOVL    4(LOG), TABLE                        ; 1888
                      10    A0  D4 0004E          CLRL    16(TABLE)                            ; 1889
             50       0A    A3  9A 00051          MOVZBL  10(LOG), R0                          ; 1894
             52       0B  A340  9E 00055          MOVAB   11(LOG)[R0], LNMX
 4F          62             02  E0 0005A 6$:      BBS     #2, (LNMX), 10$                      ; 1895
                      01    A2  95 0005E          TSTB    1(LNMX)                              ; 1897
                            3F  19 00061          BLSS    9$
 3B          62             01  E0 00063          BBS     #1, (LNMX), 9$
             54       04    A2  9A 00067          MOVZBL  4(LNMX), LEN                         ; 1901
             50       05    A2  9E 0006B          MOVAB   5(R2), ADR                           ; 1902
             04             54  D1 0006F          CMPL    LEN, #4                              ; 1903
                            0B  1F 00072          BLSSU   7$
             1B             60  B1 00074          CMPW    (ADR), #27
                            06  12 00077          BNEQ    7$
             54             04  C2 00079          SUBL2   #4, LEN                              ; 1905
             50             04  C0 0007C          ADDL2   #4, ADR
             02             54  D1 0007F 7$:      CMPL    LEN, #2                              ; 1910
                            12  1F 00082          BLSSU   8$
 51          50             54  C1 00084          ADDL3   LEN, ADR, R1                         ; 1911
             3A       FE    A1  91 00088          CMPB    -2(R1), #58
                            08  13 0008C          BEQL    8$
             3A       FF    A1  91 0008E          CMPB    -1(R1), #58                          ; 1912
                            02  12 00092          BNEQ    8$
                            54  D7 00094          DECL    LEN                                  ; 1914
                      01    A5  9F 00096 8$:      PUSHAB  1(R5)                                ; 1916
                            50  DD 00099          PUSHL   ADR
                            54  DD 0009B          PUSHL   LEN
      FF5E   CF             03  FB 0009D          CALLS   #3, RECURSIVELY_TRANSLATE
             50       04    A2  9A 000A2 9$:      MOVZBL  4(LNMX), R0                          ; 1918
             52       05  A240  9E 000A6          MOVAB   5(LNMX)[R0], LNMX
                            AD  11 000AB          BRB     6$                                   ; 1895
                            53  DD 000AD 10$:     PUSHL   LOG                                  ; 1923
      F38A   CF             01  FB 000AF          CALLS   #1, UTIL_REALLOCATE
                          FF61  31 000B4          BRW     2$                                   ; 1863
                                04 000B7 11$:     RET                                          ; 1925

; Routine Size:  184 bytes,    Routine Base:  $CODE$ + 0C16
```

```
1  ! This is the main routine
1  !
1  global routine show$logical =
2      begin
2      local
2          list:    ref cmdtbl_block;
2      local
2          free_pages,
2          iosb: vector[2],
2          jpi_list: vector[4] initial(jpi$_freptecnt^16+4,free_pages,0,0);
2
2      macro
2          dsc_lenadr_(x) = .x[dsc$w_length], .x[dsc$a_pointer] %;
2
2      ! Initialize all the flags to off
2      !
2      ch$fill(0, %allocation(flags), flags[base_]);
2      saved_fp = 0;
2
2      ! Allocate virtual memory
2      !
2      $check_($getjpiw(efn=0,iosb=iosb,itmlst=jpi_list));
2      vm_size = minu((.sgn$gl_pagedyn + 1023),((.free_pages-32)*512));
2      $check_(lib$get_vm(vm_size, vm_addr));
2
2  !+
2  ! Collect the qualifiers and logical names to be translated
2  !-
2
2      ! Determine which standard logical name tables to search
2      !    Tables specified by the /TABLE qualifier are searched before
2      !    the tables that correspond to these qualifiers.
2      !    If none of these are specified, all four tables are searched.
2      !
2      flags[flag_v_system] =       cli$present(%ascid 'system');
2      flags[flag_v_group] =        cli$present(%ascid 'group');
2      flags[flag_v_job] =          cli$present(%ascid 'job');
2      flags[flag_v_proc] =         cli$present(%ascid 'process');
2
2      ! Indicate that the heirarchical structure of the tables should be shown
2      !
2      flags[flag_v_normal]=        not cli$present(%ascid 'structure');
2
2      ! Indicates descendants of the specified tables should also be searched
2      !
2      flags[flag_v_descend] =      cli$present(%ascid 'descendants');
2
2      ! Indicates that lots of information should be displayed
2      !
2      flags[flag_v_full] =         cli$present(%ascid 'full');
2
2      ! Determine the qualifying integrity mode
2      !    Only logical names equal or inner to the qualifying mode are shown.
2      !    The integrity mode is also used to qualify which tables are searched,
2      !    and which tables are shown by the /STRUCTURE qualifier.
2      ! Also, turn on inner integrity mode flags
2      ! If none of the mode qualifiers were specified, assume user mode or inner.
```

```
	!
	begin
	own
	    mode_str: vector[4] psect($plit$) preset(
			[psl$c_kernel]= %ascid 'KERNEL_MODE',
			[psl$c_exec]=   %ascid 'EXECUTIVE_MODE',
			[psl$c_super]=  %ascid 'SUPERVISOR_MODE',
			[psl$c_user]=   %ascid 'USER_MODE');
	bind
	    dsc_access_mode = %ascid'access_mode';

	acmode = psl$c_user;
	if cli$present(dsc_access_mode)
	then
	    begin
	    local
		desc:		$bblock[dsc$c_d_bln];
	    $init_dyndesc(desc);
	    cli$get_value(dsc_access_mode, desc[base_]);
	    decr mode from psl$c_user to psl$c_kernel do
		begin
		bind
		    name = .mode_str[.mode]: $bblock[dsc$c_s_bln];
		if .desc[dsc$w_length] lequ .name[dsc$w_length] then
		if ch$eql(  .desc[dsc$w_length], .desc[dsc$a_pointer],
			    .desc[dsc$w_length], .name[dsc$a_pointer] )
		then
		    begin
		    acmode = .mode;
		    exitloop;
		    end;
		end;
	    str$free1_dx(desc[base_]);
	    end;
	end;

!+
! Get the table names specified from the command line (cmdtbl_list)
! Get the logical names specified from the command line (cmdnam_list)
!-
	cmdtbl_list = get_tables();
	cmdnam_list = get_lognam();

	! Determine whether recursive translations are desired.
	!   We will show recursive translations if NONE of the logical names
	!   contain wildcard characters.
	!
	begin
	local list: ref cmdtbl_block;
	list = cmdnam_list - %fieldexpand_(cmdtbl_l_next,0);
	flags[flag_v_recurse] = true;
	while (list = .list[cmdtbl_l_next]) neq 0 do
	    begin
	    bind name = list[cmdtbl_q_name]: $bblock[dsc$c_d_bln];
	    macro contains_(x,y) = not ch$fail(ch$find_ch(dsc_lenadr_(x), y)) %;
	    if contains_(name, '*') or contains_(name, '%')
	    then
```

```
                begin
                flags[flag_v_recurse] = false;
                exitloop;
                end;
            end;
        end;

        ! Find the relevant tables
        !
        $check_(get_relevant_tables());
        if known_list[base_] eql 0
        then
            begin
            !
            ! Complain if we found no logical name tables
            !
            signal(ss$_ivlogtab and not sts$m_severity or sts$k_warning);
            end;

        ! Check for recursive translations
        !
        if .flags[flag_v_recurse]
        then
            begin
            local
                list:           ref cmdtbl_block;
            list = cmdnam_list - %fieldexpand_(cmdtbl_l_next,0);
            while (list = .list[cmdtbl_l_next]) neq 0 do
                begin
                bind name = list[cmdtbl_q_name]: $bblock[dsc$c_d_bln];
                if .flags[flag_v_found] then write_blank_line();
                flags[flag_v_found] = false;
                recursively_translate(dsc_lenadr_(name), 0);
                if not .flags[flag_v_found]
                then
                    signal(show$_notran,1,list[cmdtbl_q_name]);
                end;
            end
        else
            begin

            if .flags[flag_v_normal]
            then
                for_each_lnmb( match_lnmb, 0 );

            ! Now display the tables, and the logical names in the tables
            !
            list = cmdtbl_list - %fieldexpand_(cmdtbl_l_next,0);
            while (list = .list[cmdtbl_l_next]) neq 0 do
                begin
                visit_tree(.list[cmdtbl_l_table],
                    %fieldexpand_(table_l_child,0),
                    %fieldexpand_(table_l_next,0),
                    dump_table,
                    0);
                end;
```

```
3
3         ! If none of the logical names translated, complain about it.
3         !
3         if .flags[flag_v_normal] and not .flags[flag_v_found]
3         then
3             signal(show$_notran,1,cmdnam_list[cmdtbl_q_name]);
3
2         end;
2
2
2     ! Free the dynamic strings that have been holding table and logical names.
2     !
2     decr i from 1 to 0 do
3         begin
3         if .i
3         then list = cmdtbl_list - %fieldexpand_(cmdtbl_l_next,0)
3         else list = cmdnam_list - %fieldexpand_(cmdtbl_l_next,0);
3         while (list = .list[cmdtbl_l_next]) neq 0 do
4             begin
4             bind name = list[cmdtbl_q_name]: $bblock[dsc$c_d_bln];
4             if .name[dsc$b_class] eql dsc$k_class_d
4             then
4                 str$free1_dx(name[base_]);
3             end;
2         end;
2
2
2     ! Deallocate virtual memory
2     !
2     vm_size = minu((.sgn$gl_pagedyn + 1023),((.free_pages-32)*512));
2     lib$free_vm(vm_size, vm_addr);
2
2     return ss$_normal;
1     end;
; INFO#250            L1:1948
; Referenced LOCAL symbol FREE_PAGES is probably not initialized


                                                               .PSECT  $PLIT$,NOWRT,NOEXE,2

                                               001F1          .BLKB   3
                                    04150004   001F4 P.ACH:   .LONG   68485124
              00000000   00000000   00000000   001F8          .LONG   0, 0, 0
            00  00  6D  65  74  73  79  73     00204 P.ACJ:   .ASCII  \system\<0><0>
                                    010E0006   0020C P.ACI:   .LONG   17694726
                                   00000000'   00210          .ADDRESS P.ACJ
            00  00  00  70  75  6F  72  67     00214 P.ACL:   .ASCII  \group\<0><0><0>
                                    010E0005   0021C P.ACK:   .LONG   17694725
                                   00000000'   00220          .ADDRESS P.ACL
                            00  62  6F  6A     00224 P.ACN:   .ASCII  \job\<0>
                                    010E0003   00228 P.ACM:   .LONG   17694723
                                   00000000'   0022C          .ADDRESS P.ACN
            00  73  73  65  63  6F  72  70     00230 P.ACP:   .ASCII  \process\<0>
                                    010E0007   00238 P.ACO:   .LONG   17694727
                                   00000000'   0023C          .ADDRESS P.ACP
00  00  00  65  72  75  74  63  75  72  74  73 00240 P.ACR:   .ASCII  \structure\<0><0><0>
```

```
                                   010E0009  0024C P.ACQ:  .LONG    17694729                        ;
                                  00000000'  00250         .ADDRESS P.ACR                           ;
         00 73 74 6E 61 64 6E 65 63 73 65 64  00254 P.ACT:  .ASCII   \descendants\<0>                 ;
                                   010E000B  00260 P.ACS:  .LONG    17694731                        ;
                                  00000000'  00264         .ADDRESS P.ACT                           ;
                                6C 6C 75 66  00268 P.ACV:  .ASCII   \full\                          ;
                                   010E0004  0026C P.ACU:  .LONG    17694724                        ;
                                  00000000'  00270         .ADDRESS P.ACV                           ;
         00 45 44 4F 4D 5F 4C 45 4E 52 45 4B  00274 P.ACX:  .ASCII   \KERNEL_MODE\<0>                 ;
                                   010E000B  00280 P.ACW:  .LONG    17694731                        ;
                                  00000000'  00284         .ADDRESS P.ACX                           ;
00 45 44 4F 4D 5F 45 56 49 54 55 43 45 58 45  00288 P.ACZ:  .ASCII   \EXECUTIVE_MODE\<0><0>           ;
                                         00  00297                                                  ;
                                   010E000E  00298 P.ACY:  .LONG    17694734                        ;
                                  00000000'  0029C         .ADDRESS P.ACZ                           ;
45 44 4F 4D 5F 52 4F 53 49 56 52 45 50 55 53  002A0 P.ADB:  .ASCII   \SUPERVISOR_MODE\<0>             ;
                                         00  002AF                                                  ;
                                   010E000F  002B0 P.ADA:  .LONG    17694735                        ;
                                  00000000'  002B4         .ADDRESS P.ADB                           ;
         00 00 00 45 44 4F 4D 5F 52 45 53 55  002B8 P.ADD:  .ASCII   \USER_MODE\<0><0><0>             ;
                                   010E0009  002C4 P.ADC:  .LONG    17694729                        ;
                                  00000000'  002C8         .ADDRESS P.ADD                           ;
         00000000' 00000000' 00000000' 00000000'  002CC MODE_STR:                                    ;
                                                           .ADDRESS P.ACW, P.ACY, P.ADA, P.ADC      ;
         00 65 64 6F 6D 5F 73 73 65 63 63 61  002DC P.ADF:  .ASCII   \access_mode\<0>                 ;
                                   010E000B  002E8 P.ADE:  .LONG    17694731                        ;
                                  00000000'  002EC         .ADDRESS P.ADF                           ;

                                                   DSC_ACCESS_MODE=     P.ADE
                                                           .EXTRN   SYS$GETJPIW

                                                           .PSECT   $CODE$,NOWRT,2

                                       0FFC 00000          .ENTRY   SHOW$LOGICAL, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 1928
                                                                    R10,R11                         ;
                    5B 00000000G  00   9E 00002            MOVAB    SGN$GL_PAGEDYN, R11             ;
                    5A 00000000G  00   9E 00009            MOVAB    LIB$SIGNAL, R10                 ;
                    59 00000000G  00   9E 00010            MOVAB    LIB$STOP, R9                    ;
                    58     0000'  CF   9E 00017            MOVAB    DSC_ACCESS_MODE, R8             ;
                    57 00000000G  00   9E 0001C            MOVAB    CLI$PRESENT, R7                 ;
                    56     0000'  CF   9E 00023            MOVAB    FLAGS, R6                       ;
                    5E            24   C2 00028            SUBL2    #36, SP                         ;
        0C    AE     FF0C   C8    10   28 0002B            MOVC3    #16, P.ACH, JPI_LIST            ; 1935
                       10   AE    6E   9E 00032            MOVAB    FREE_PAGES, JPI_LIST+4          ; 1929
                                  66   B4 00036            CLRW     FLAGS                           ; 1942
                            F0    A6   D4 00038            CLRL     SAVED_FP                        ; 1943
                                  7E   7C 0003B            CLRQ     -(SP)                           ; 1947
                            24    AE   9F 0003D            PUSHAB   IOSB                            ;
                            18    AE   9F 00040            PUSHAB   JPI_LIST                        ;
                                  7E   7C 00043            CLRQ     -(SP)                           ;
                                  7E   D4 00045            CLRL     -(SP)                           ;
             00000000G  00        07   FB 00047            CALLS    #7, SYS$GETJPIW                 ;
                        05        50   E8 0004E            BLBS     S, 1$                           ;
                                  50   DD 00051            PUSHL    S                               ;
                        69        01   FB 00053            CALLS    #1, LIB$STOP                    ;
          50            6B 000003FF 8F C1 00056 1$:        ADDL3    #1023, SGN$GL_PAGEDYN, R0       ; 1948
          55            6E        20   C3 0005E            SUBL3    #32, FREE_PAGES, R5             ;
```

```
            55         55          09  78 00062        ASHL    #9, R5, R5                        ;
                       55          50  D1 00066        CMPL    R0, R5                            ;
                                   03  1B 00069        BLEQU   2$                                ;
                       50          55  D0 0006B        MOVL    R5, R0                            ;
                  E8   A6          50  D0 0006E 2$:    MOVL    R0, VM_SIZE                       ;
                            EC     A6  9F 00072        PUSHAB  VM_ADDR                           ; 1949
                            E8     A6  9F 00075        PUSHAB  VM_SIZE                           ;
           00000000G   00          02  FB 00078        CALLS   #2, LIB$GET_VM                    ;
                       05          50  E8 0007F        BLBS    S, 3$                             ;
                                   50  DD 00082        PUSHL   S                                 ;
                       69          01  FB 00084        CALLS   #1, LIB$STOP                      ;
                          FF24     C8  9F 00087 3$:    PUSHAB  P.ACI                             ; 1960
                       67          01  FB 0008B        CALLS   #1, CLI$PRESENT                   ;
    66          01     00          50  F0 0008E        INSV    R0, #0, #1, FLAGS                 ;
                          FF34     C8  9F 00093        PUSHAB  P.ACK                             ; 1961
                       67          01  FB 00097        CALLS   #1, CLI$PRESENT                   ;
    66          01     01          50  F0 0009A        INSV    R0, #1, #1, FLAGS                 ;
                          FF40     C8  9F 0009F        PUSHAB  P.ACM                             ; 1962
                       67          01  FB 000A3        CALLS   #1, CLI$PRESENT                   ;
    66          01     02          50  F0 000A6        INSV    R0, #2, #1, FLAGS                 ;
                          FF50     C8  9F 000AB        PUSHAB  P.ACO                             ; 1963
                       67          01  FB 000AF        CALLS   #1, CLI$PRESENT                   ;
    66          01     03          50  F0 000B2        INSV    R0, #3, #1, FLAGS                 ;
                          FF64     C8  9F 000B7        PUSHAB  P.ACQ                             ; 1967
                       67          01  FB 000BB        CALLS   #1, CLI$PRESENT                   ;
                       51          50  D2 000BE        MCOML   R0, R1                            ;
    66          01     04          51  F0 000C1        INSV    R1, #4, #1, FLAGS                 ;
                          FF78     C8  9F 000C6        PUSHAB  P.ACS                             ; 1971
                       67          01  FB 000CA        CALLS   #1, CLI$PRESENT                   ;
    66          01     05          50  F0 000CD        INSV    R0, #5, #1, FLAGS                 ;
                            84     A8  9F 000D2        PUSHAB  P.ACU                             ; 1975
                       67          01  FB 000D5        CALLS   #1, CLI$PRESENT                   ;
    66          01     06          50  F0 000D8        INSV    R0, #6, #1, FLAGS                 ;
                  02   A6          03  90 000DD        MOVB    #3, ACMODE                        ; 1994
                                   58  DD 000E1        PUSHL   R8                                ; 1995
                       67          01  FB 000E3        CALLS   #1, CLI$PRESENT                   ;
                       41          50  E9 000E6        BLBC    R0, 7$                            ;
                  04   AE 020E0000 8F  D0 000E9        MOVL    #34471936, DESC                   ; 2000
                            08     AE  D4 000F1        CLRL    DESC+4                            ;
                            04     AE  9F 000F4        PUSHAB  DESC                              ; 2001
                                   58  DD 000F7        PUSHL   R8                                ;
           00000000G   00          02  FB 000F9        CALLS   #2, CLI$GET_VALUE                 ;
                       54          03  D0 00100        MOVL    #3, MODE                          ; 2006
                       50      E4 A844 D0 00103 4$:    MOVL    MODE_STR[MODE], R0                ; 2005
                       60      04  AE  B1 00108        CMPW    DESC, (R0)                        ; 2006
                                   0F  1A 0010C        BGTRU   5$                                ;
         04    B0   08   BE    04  AE  29 0010E        CMPC3   DESC, @DESC+4, @4(R0)             ; 2007
                                   06  12 00115        BNEQ    5$                                ;
                  02   A6          54  90 00117        MOVB    MODE, ACMODE                      ; 2011
                                   03  11 0011B        BRB     6$                                ; 2010
                       E3          54  F4 0011D 5$:    SOBGEQ  MODE, 4$                          ; 2002
                            04     AE  9F 00120 6$:    PUSHAB  DESC                              ; 2015
           00000000G   00          01  FB 00123        CALLS   #1, STR$FREE1_DX                  ;
                F352   CF          00  FB 0012A 7$:    CALLS   #0, GET_TABLES                    ; 2023
                  F8   A6          50  D0 0012F        MOVL    R0, CMDTBL_LIST                   ;
                F3ED   CF          00  FB 00133        CALLS   #0, GET_LOGNAM                    ; 2024
                  FC   A6          50  D0 00138        MOVL    R0, CMDNAM_LIST                   ;
```

```
                      53        FC    A6  9E 0013C        MOVAB   CMDNAM_LIST, LIST                   ; 2032
                      66        80    8F  88 00140        BISB2   #128, FLAGS                         ; 2033
                      53              63  D0 00144 8$:    MOVL    (LIST), LIST                        ; 2034
                                      22  13 00147        BEQL    12$                                 ;
                      52        08    A3  9E 00149        MOVAB   8(LIST), R2                         ; 2036
        04    B2      62              2A  3A 0014D        LOCC    #42, (R2), @4(R2)                   ; 2038
                                      02  12 00152        BNEQ    9$                                  ;
                                      51  D4 00154        CLRL    R1                                  ;
                                      51  D5 00156 9$:    TSTL    R1                                  ;
                                      0D  12 00158        BNEQ    11$                                 ;
        04    B2      62              25  3A 0015A        LOCC    #37, (R2), @4(R2)                   ;
                                      02  12 0015F        BNEQ    10$                                 ;
                                      51  D4 00161        CLRL    R1                                  ;
                                      51  D5 00163 10$:   TSTL    R1                                  ;
                                      DD  13 00165        BEQL    8$                                  ;
                      66        80    8F  8A 00167 11$:   BICB2   #128, FLAGS                         ; 2041
               FBC0   CF              00  FB 0016B 12$:   CALLS   #0, GET_RELEVANT_TABLES             ; 2050
                      05              50  E8 00170        BLBS    S, 13$                              ;
                                      50  DD 00173        PUSHL   S                                   ;
                      69              01  FB 00175        CALLS   #1, LIB$STOP                        ;
                                F4    A6  D5 00178 13$:   TSTL    KNOWN_LIST                          ; 2051
                                      08  12 0017B        BNEQ    14$                                 ;
                      7E      0158    8F  3C 0017D        MOVZWL  #344, -(SP)                         ; 2057
                      6A              01  FB 00182        CALLS   #1, LIB$SIGNAL                      ;
                                      66  95 00185 14$:   TSTB    FLAGS                               ; 2062
                                      3F  18 00187        BGEQ    17$                                 ;
                      52        FC    A6  9E 00189        MOVAB   CMDNAM_LIST, LIST                   ; 2067
                      52              62  D0 0018D 15$:   MOVL    (LIST), LIST                        ; 2068
                                      79  13 00190        BEQL    21$                                 ;
                      53        08    A2  9E 00192        MOVAB   8(LIST), R3                         ; 2070
                      09        01    A6  E9 00196        BLBC    FLAGS+1, 16$                        ; 2071
                              FD18    C8  9F 0019A        PUSHAB  BLANK_LINE                          ;
               F18F   CF              01  FB 0019E        CALLS   #1, UTIL_OUTPUT                     ;
                 01   A6              01  8A 001A3 16$:   BICB2   #1, FLAGS+1                         ; 2072
                                      7E  D4 001A7        CLRL    -(SP)                               ; 2073
                                04    A3  DD 001A9        PUSHL   4(R3)                               ;
                      7E              63  3C 001AC        MOVZWL  (R3), -(SP)                         ;
               FD94   CF              03  FB 001AF        CALLS   #3, RECURSIVELY_TRANSLATE           ;
                      D5        01    A6  E8 001B4        BLBS    FLAGS+1, 15$                        ; 2074
                                08    A2  9F 001B8        PUSHAB  8(LIST)                             ; 2076
                                      01  DD 001BB        PUSHL   #1                                  ;
                         00000000G    8F  DD 001BD        PUSHL   #SHOW$_NOTRAN                       ;
                      6A              03  FB 001C3        CALLS   #3, LIB$SIGNAL                      ;
                                      C5  11 001C6        BRB     15$                                 ; 2068
              0B      66              04  E1 001C8 17$:   BBC     #4, FLAGS, 18$                      ; 2082
                                      7E  D4 001CC        CLRL    -(SP)                               ; 2084
                              F601    CF  9F 001CE        PUSHAB  MATCH_LNMB                          ;
               F6E0   CF              02  FB 001D2        CALLS   #2, FOR_EACH_LNMB                   ;
                      52        F8    A6  9E 001D7 18$:   MOVAB   CMDTBL_LIST, LIST                   ; 2088
                      52              62  D0 001DB 19$:   MOVL    (LIST), LIST                        ; 2089
                                      13  13 001DE        BEQL    20$                                 ;
                                      7E  D4 001E0        CLRL    -(SP)                               ; 2091
                              FA80    CF  9F 001E2        PUSHAB  DUMP_TABLE                          ;
                      7E              04  7D 001E6        MOVQ    #4, -(SP)                           ;
                                04    A2  DD 001E9        PUSHL   4(LIST)                             ;
               F256   CF              05  FB 001EC        CALLS   #5, VISIT_TREE                      ;
                                      E8  11 001F1        BRB     19$                                 ; 2089
```

```
14              66              04 E1 001F3 20$:    BBC     #4, FLAGS, 21$                    ; 2100
                10        01    A6 E8 001F7         BLBS    FLAGS+1, 21$                      ;
7E        FC    A6              08 C1 001FB         ADDL3   #8, CMDNAM_LIST, -(SP)            ; 2102
                                01 DD 00200         PUSHL   #1                                ;
                   00000000G    8F DD 00202         PUSHL   #SHOW$_NOTRAN                     ;
                6A              03 FB 00208         CALLS   #3, LIB$SIGNAL                    ;
                53              01 D0 0020B 21$:    MOVL    #1, I                             ; 2109
                06              53 E9 0020E 22$:    BLBC    I, 23$                            ; 2111
                52        F8    A6 9E 00211         MOVAB   CMDTBL_LIST, LIST                 ; 2112
                                04 11 00215         BRB     24$                               ;
                52        FC    A6 9E 00217 23$:    MOVAB   CMDNAM_LIST, LIST                 ; 2113
                52              62 D0 0021B 24$:    MOVL    (LIST), LIST                      ; 2114
                                15 13 0021E         BEQL    25$                               ;
                50        08    A2 9E 00220         MOVAB   8(LIST), R0                       ; 2116
                02        03    A0 91 00224         CMPB    3(R0), #2                         ; 2117
                                F1 12 00228         BNEQ    24$                               ;
                                50 DD 0022A         PUSHL   R0                                ; 2119
      00000000G 00              01 FB 0022C         CALLS   #1, STR$FREE1_DX                  ;
                                E6 11 00233         BRB     24$                               ; 2114
                D6              53 F4 00235 25$:    SOBGEQ  I, 22$                            ; 2109
50              6B  000003FF    8F C1 00238         ADDL3   #1023, SGN$GL_PAGEDYN, R0         ; 2126
                55              50 D1 00240         CMPL    R0, R5                            ;
                                03 1B 00243         BLEQU   26$                               ;
                50              55 D0 00245         MOVL    R5, R0                            ;
          E8    A6              50 D0 00248 26$:    MOVL    R0, VM_SIZE                       ;
                          EC    A6 9F 0024C         PUSHAB  VM_ADDR                           ; 2127
                          E8    A6 9F 0024F         PUSHAB  VM_SIZE                           ;
      00000000G 00              02 FB 00252         CALLS   #2, LIB$FREE_VM                   ;
                50              01 D0 00259         MOVL    #1, R0                            ; 2129
                                   04 0025C         RET                                       ; 2130

; Routine Size:  605 bytes,    Routine Base:  $CODE$ + 0CCE


; 2046          2131   1
; 2047          2132   1 end
; 2048          2133   0 eludom



                                                    .EXTRN  LIB$SIGNAL, LIB$STOP

;                              PSECT SUMMARY
;
;       Name                     Bytes                      Attributes
;
;  $OWN$                            68  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $SPLIT$                         752  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                         3883  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      71         0       1000      00:01.9
;  _$255$DUA28:[SYSLIB]XPORT.L32;1              590      29         4        252      00:01.0



; Information:  1
; Warnings:     0
; Errors:       0


;                                COMMAND QUALIFIERS

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

; Size:          3883 code + 820 data bytes
; Run Time:         01:22.3
; Elapsed Time:     04:01.5
; Lines/CPU Min:     1555
; Lexemes/CPU-Min: 35981
; Memory Used:  272 pages
; Compilation Complete
```

0056 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SHOMSGUTL
LIS

SHONET
LIS

SHOWAUDIT
LIS

SHOWIO
LIS

SHOWLOG
LIS

SHOWERROR
LIS

SHOWFILES
LIS

SHOMEMORY
LIS

SHOWTERM
LIS

SHOWMISC
LIS

SHOWPROC
LIS

SHOWSYS
LIS

SHOWMAIN
LIS

SHOWMSCP
LIS

SHOWQUE
LIS

SHOWMSG
LIS