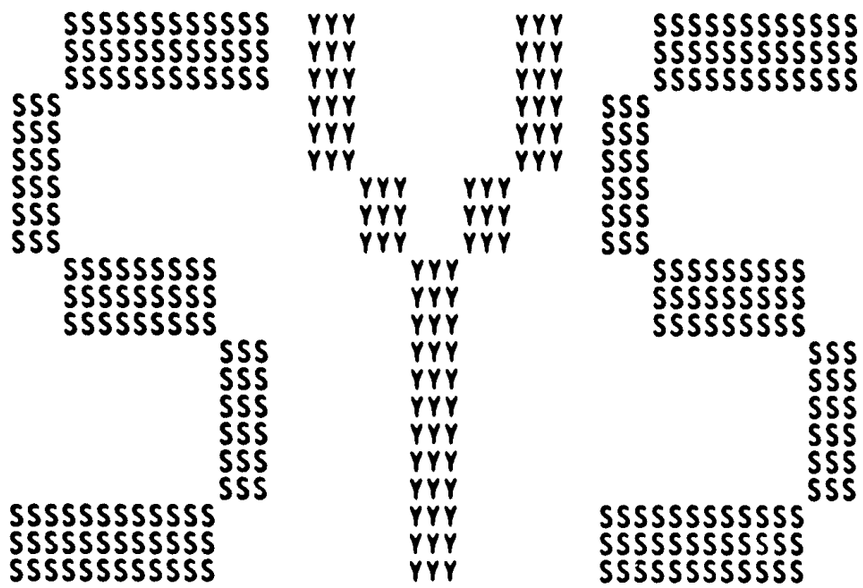

FILEID**SYSLNM

```
  SSSSSSSS  YY      YY    SSSSSSSS  LL          NN      NN  MM      MM
  SSSSSSSS  YY      YY    SSSSSSSS  LL          NN      NN  MM      MM
SS          YY      YY  SS          LL          NN      NN  MMMM  MMMM
SS          YY      YY  SS          LL          NN      NN  MMMM  MMMM
SS            YY  YY    SS          LL          NNNN    NN  MM  MM  MM
SS            YY  YY    SS          LL          NNNN    NN  MM  MM  MM
  SSSSSS        YY        SSSSSS    LL          NN  NN  NN  MM      MM
  SSSSSS        YY        SSSSSS    LL          NN  NN  NN  MM      MM
        SS      YY              SS  LL          NN    NNNN  MM      MM
        SS      YY              SS  LL          NN    NNNN  MM      MM
        SS      YY              SS  LL          NN      NN  MM      MM      ....
        SS      YY              SS  LL          NN      NN  MM      MM      ....
SSSSSSSS        YY      SSSSSSSS    LLLLLLLLLL  NN      NN  MM      MM      ....
SSSSSSSS        YY      SSSSSSSS    LLLLLLLLLL  NN      NN  MM      MM      ....

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
```

Table of contents

```
0000     1          .TITLE  SYSLNM - SYSTEM SERVICES TO MANIPULATE LOGICAL NAMES AND TABLES.
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;
0000     5  ;****************************************************************************
0000     6  ;*                                                                          *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000     9  ;*  ALL RIGHTS RESERVED.                                                    *
0000    10  ;*                                                                          *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16  ;*  TRANSFERRED.                                                            *
0000    17  ;*                                                                          *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                            *
0000    21  ;*                                                                          *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    24  ;*                                                                          *
0000    25  ;*                                                                          *
0000    26  ;****************************************************************************
0000    27  ;
0000    28  ; TODD M. KATZ  01-APR-83
0000    29  ;
0000    30  ; SYSTEM SERVICES TO MANIPULATE LOGICAL NAMES
0000    31  ;
0000    32  ;       CREATE LOGICAL NAME TABLES
0000    33  ;       CREATE LOGICAL NAME
0000    34  ;       DELETE LOGICAL NAME
0000    35  ;       TRANSLATE LOGICAL NAME
0000    36  ;
0000    37  ; MODIFICATION HISTORY:
0000    38  ;
0000    39  ;       V03-027 RAS0327         Ron Schaefer            24-Jul-1984
0000    40  ;               Back out RAS0322 and make the LNM$_LENGTH itemcode
0000    41  ;               be a longword value.
0000    42  ;
0000    43  ;       V03-026 RAS0322         Ron Schaefer            10-Jul-1984
0000    44  ;               Fix $TRNLNM item LNM$_LENGTH to only write a word field.
0000    45  ;
0000    46  ;       V03-025 RAS0312         Ron Schaefer            21-Jun-1984
0000    47  ;               Fix ORB alignment within logical name table.
0000    48  ;
0000    49  ;       V03-024 TMK0014         Todd M. Katz            21-Apr-1984
0000    50  ;               The interface to the internal logical name routine
0000    51  ;               LNM$DELETE_LNMB has been changed. Update $DELLNM, which calls
0000    52  ;               this routine, to reflect this new interface.
0000    53  ;
0000    54  ;               The performance measurement cell used to monitor the rate of
0000    55  ;               logical name translations is currently located within the
0000    56  ;               internal logical name routine LNM$SEARCHLOG. Unfortunately,
0000    57  ;               because of its current placement, any attempts to delete
```

```
0000     58 ;                  specific logical names will also increment this counter. This
0000     59 ;                  is because the system service $DELLNM will call the routine
0000     60 ;                  LNM$SEARCHLOG in such a situation. Therefore, in order to be
0000     61 ;                  able to make a more accurate measurement of the overall rate
0000     62 ;                  of logical name translations, I have decided to move this
0000     63 ;                  performance measurement cell from its current single location
0000     64 ;                  to several more appropriate locations. One of these new
0000     65 ;                  locations is within the system service $TRNLNM, just before the
0000     66 ;                  call to LNM$SEARCHLOG.
0000     67 ;
0000     68 ;          V03-023 TMK0013         Todd M. Katz            29-Mar-1984
0000     69 ;                  Modify the logical name system services to make use of the
0000     70 ;                  updated internal protection checking mechanisms. What this
0000     71 ;                  requires is a modification to SYS$CRELNT so that all shareable
0000     72 ;                  logical name tables are created with a quad-word aligned Object
0000     73 ;                  Rights Block in place of a un-aligned CHIP protection template.
0000     74 ;
0000     75 ;                  Restrict all names which appear with a directory table to 31
0000     76 ;                  characters consisting of the DEC multi-national alphanumeric
0000     77 ;                  character set plus $ and _. This restriction applies to both
0000     78 ;                  logical names and logical name table names. The reason for this
0000     79 ;                  restriction is that in the future we might want to support
0000     80 ;                  hierarchial name spaces. If we decide to do so we would have to
0000     81 ;                  invent and then impose a structure on logical name table names.
0000     82 ;                  This restriction gives us sufficient leeway (and more
0000     83 ;                  importantly sufficient available characters) to be able to
0000     84 ;                  define a hierarchial name space structure in the future.
0000     85 ;
0000     86 ;          V03-022 TMK0012         Todd M. Katz            07-Mar-1984
0000     87 ;                  The fixed portion of each and every translation block has been
0000     88 ;                  increased by a word, LNMX$W_HASH, in order to potentially hold
0000     89 ;                  the translation's hash code value. Furthermore, SYS$CRELNT and
0000     90 ;                  SYS$CRELNM have been modified so that this field within each
0000     91 ;                  and every translation block is initialized to 0. This hash code
0000     92 ;                  field will be used in an optimization of logical name table name
0000     93 ;                  processing. As such, only certain logical names, those that are
0000     94 ;                  contained within the process or system directory and may be used
0000     95 ;                  in logical name table processing, need to have the hash code
0000     96 ;                  fields of each of their translations initialized to their
0000     97 ;                  equivalence string's hash code value. This initialization
0000     98 ;                  actually takes place within the routine LNM$INSLOGTAB in the
0000     99 ;                  module LNMSUB.
0000    100 ;
0000    101 ;          V03-021 TMK0011         Todd M. Katz            13-Feb-1984
0000    102 ;                  Add SS$_NOLOGTAB to the list of errors than can be returned
0000    103 ;                  by SYS$CRELNT, SYS$CRELNM, and SYS$DELLNM.
0000    104 ;
0000    105 ;                  Fix a bug in $TRNLNM. At the present time the very first
0000    106 ;                  longword in the item list is not being probed. Obviously this
0000    107 ;                  could result in a disasterous kernel mode ACCVIO.
0000    108 ;
0000    109 ;                  Fix up $TRNLNM's processing of the PARENT item. The most
0000    110 ;                  important change I have made here, besides a general fixup
0000    111 ;                  of the code, is to return 0 bytes in the parent item if the LNMB
0000    112 ;                  being looked at is not for a logical name table. This is because
0000    113 ;                  logical names, with the exception of logical name tables, do not
0000    114 ;                  have parents.
```

```
0000    115 ;
0000    116 ;               Optimize the processing of $TRNLNM's item list. This is done
0000    117 ;               by having two separate code segments - one to probe and return
0000    118 ;               variable length character string items (TABLE, STRING, PARENT),
0000    119 ;               and the other to probe and return longword length items (LENGTH,
0000    120 ;               ATTRIBUTES, MAX_INDEX). In addition, since ACMODEs are only a
0000    121 ;               byte, ACMODE items are now being probed and filled in in-line.
0000    122 ;               Also, optimize a few instructions within this system service.
0000    123 ;
0000    124 ;       V03-020 TMK0010         Todd M. Katz            29-Dec-1983
0000    125 ;               Re-write SYS$DELLNM for the case when the name of the logical
0000    126 ;               name table entry to be deleted has been specified. What was
0000    127 ;               being done was incorrect. It allowed the possibility of deleting
0000    128 ;               several logical names from different tables. The correct way
0000    129 ;               to implement this is as follows:
0000    130 ;
0000    131 ;                  1. Search for the first instance of a logical name table entry
0000    132 ;                     in one of the specified logical name tables which possesses
0000    133 ;                     the specified access mode.
0000    134 ;
0000    135 ;                  2. Check to make sure that the caller has write access to the
0000    136 ;                     containing logical name table.
0000    137 ;
0000    138 ;                  3. Delete the logical name table entry and all outermode
0000    139 ;                     aliases within the same logical name table.
0000    140 ;
0000    141 ;               The process and system directory logical name tables are now
0000    142 ;               being created with the LNM$V_NODELETE bit set within their
0000    143 ;               LNMB$B_FLAGS fields to indicate that these tables should
0000    144 ;               never be deleted. Therefore, it is no longer necessary for
0000    145 ;               SYS$CRELNT to set this bit within the appropriate directory
0000    146 ;               table to protect it from being deleted during the insertion of
0000    147 ;               the table entry for the new logical name table, and conversely
0000    148 ;               SYS$CRELNT must never clear this bit within a directory table
0000    149 ;               after the attempt at insertion has been made.
0000    150 ;
0000    151 ;               Fix a bug in the creation of logical names without translations
0000    152 ;               by SYS$CRELNM. This used to work, but has broken mysteriously.
0000    153 ;               The problem is that during the first processing of the item
0000    154 ;               list, the number of translations specified by the list and
0000    155 ;               the cumulated size of their translation strings is saved on the
0000    156 ;               stack for use in computing the amount of storage required to be
0000    157 ;               allocated for the new logical name block. These stack associated
0000    158 ;               counters were not being appropriately initialized to zero in the
0000    159 ;               case when no item list (and thus no translations) was present.
0000    160 ;               This resulted in the size of the table name and the address of
0000    161 ;               the table name buffer being popped off the stack and used as
0000    162 ;               the values of these counters. As might be expected this would
0000    163 ;               cause the system service to fail in a variety of interesting
0000    164 ;               ways. Often, since the saved logical name table name descriptor
0000    165 ;               had been removed from the stack but the pointer to the saved
0000    166 ;               descriptor in R9 had not changed, this would lead to a failure
0000    167 ;               to find a logical name table and termination of the system
0000    168 ;               service. Also, since the values used in computing the amount
0000    169 ;               of storage required for the new logical name were random, the
0000    170 ;               value computed could be so large as to cause the storage
0000    171 ;               allocation attempt to fail.
```

```
0000   172 ;
0000   173 ;       V03-019 TMK0009         Todd M. Katz            18-Dec-1983
0000   174 ;               Remove all mention of LNM$V_SYSTEM and LNM$V_GROUP from
0000   175 ;               SYS$CRELNT. The system table and all group tables will be
0000   176 ;               handcrafted, and there will not be a way to create such a
0000   177 ;               table by means of the system service interface.
0000   178 ;
0000   179 ;               Also, change some PUSHRs into PUSHLs (or MOVQs) and POPRs into
0000   180 ;               POPLs (or MOVQs) where appropriate for performance reasons.
0000   181 ;
0000   182 ;       V03-018 TMK0008         Todd M. Katz            26-Oct-1983
0000   183 ;               Quota checking when logical name tables are being created
0000   184 ;               is presently incorrect. Currently, when a logical name
0000   185 ;               table is created only one type of quota check is made. That
0000   186 ;               check makes sure that the quota holder of the parent logical
0000   187 ;               name table has sufficient quota for both the logical name table
0000   188 ;               itself and any quota which will be specifically allocated to it.
0000   189 ;               Actually, two quota checks should be made. The quota holder of
0000   190 ;               the parent logical name table should have sufficient quota for
0000   191 ;               the quota that will be specifically allocated to the new table;
0000   192 ;               but in addition, the containing table (the system or process
0000   193 ;               directory table) must have sufficient quota for the logical
0000   194 ;               name table itself. This is consistant with how quota deductions
0000   195 ;               are made for logical names, and it is consistant with the
0000   196 ;               philosophy that logical name tables are just logical names with
0000   197 ;               a special type of translation.
0000   198 ;
0000   199 ;       V03-017 TMK0007         Todd M. Katz            26-Sep-1983
0000   200 ;               Change the default protection that is assigned to new shareable
0000   201 ;               logical name tables to SYSTEM:RWED OWNER:RWED GROUP: WORLD:
0000   202 ;               so that by default, the system can access and modify any
0000   203 ;               shareable logical name table.
0000   204 ;
0000   205 ;       V03-016 TMK0006         Todd M. Katz            16-Sep-1983
0000   206 ;               Fix a branch in EXE$CRELNT. If no attributes were specified,
0000   207 ;               this service should branch to check out the table name
0000   208 ;               parameter; however, it is branching into the wrong place. This
0000   209 ;               results in an inability to create a logical name table if
0000   210 ;               attributes are not specified. This fixes the problem.
0000   211 ;
0000   212 ;       V03-015 TMK0005         Todd M. Katz            08-Aug-1983
0000   213 ;               Make several modifications to each of the logical name system
0000   214 ;               services.
0000   215 ;
0000   216 ;               Changes to EXE$CRELNT:
0000   217 ;
0000   218 ;               1. Change access mode processing. The new logical name table
0000   219 ;                  may not be created in an access mode inner to that of the
0000   220 ;                  caller unless the user has SYSNAM privilege and has
0000   221 ;                  specified an access mode as an optional system service
0000   222 ;                  arguement. A new logical name table may still be created
0000   223 ;                  with an access mode outer to the mode of the caller, even if
0000   224 ;                  the caller does not have SYSNAM privilege, if the caller
0000   225 ;                  explicitely specifies an access mode as an optional system
0000   226 ;                  service parameter.
0000   227 ;
0000   228 ;               2. A new logical name table maybe marked within the table
```

```
0000    229 ;                   header as either the system table or a group table (but not
0000    230 ;                   both) by setting one of the new attribute bits (reserved to
0000    231 ;                   DIGITAL) LNM$V_SYSTEM or LNM$V_GROUP respectively. The
0000    232 ;                   caller must be in executive or kernel mode to do so.
0000    233 ;
0000    234 ;               3.  It is necessary to have quota access to the parent logical
0000    235 ;                   name table in order to be able create a subtable. In
0000    236 ;                   addition if the table being creating is shareable, and the
0000    237 ;                   caller is specifying a name, then the caller must have write
0000    238 ;                   access to the system directory table. Previously, read and
0000    239 ;                   write access to the parent logical name table was what was
0000    240 ;                   being checked. This is not only insufficient but wrong. I
0000    241 ;                   have also moved these access checks so that they are
0000    242 ;                   performed immediately after the parent logical name table is
000C    243 ;                   located.
0000    244 ;
0000    245 ;               4.  SYSNAM was formerly required to specify the name of a
0000    246 ;                   shareable logical name table and not let the system default
0000    247 ;                   name it (ie - insert the name into the system directory
0000    248 ;                   table). The meaning of SYSNAM has been restricted to refer
0000    249 ;                   only to the access of the system logical name table,
0000    250 ;                   LNM$SYSTEM_TABLE, and to the creation and deletion of inner
0000    251 ;                   access mode logical names and tables.
0000    252 ;
0000    253 ;               5.  If no protection mask was specified, then default the
0000    254 ;                   protection to O:RWED.
0000    255 ;
0000    256 ;               6.  It is never necessary to do any protection checking when
0000    257 ;                   process-private logical name tables are involved. Therefore,
0000    258 ;                   do not allocate or fill in the CHIP protection structure
0000    259 ;                   template when creating a process-private logical name table,
0000    260 ;                   and do not perform any protection checking during the
0000    261 ;                   creation of such tables.
0000    262 ;
0000    263 ;           Changes to EXE$CRELNM:
0000    264 ;
0000    265 ;               1.  Change access mode processing. The new logical name may not
0000    266 ;                   be created in an access mode inner to that of the caller
0000    267 ;                   unless the user has SYSNAM privilege and has specified an
0000    268 ;                   access mode as an optional system service arguement. A new
0000    269 ;                   logical name may still be created with an access mode outer
0000    270 ;                   to the mode of the caller, even if the caller does not have
0000    271 ;                   SYSNAM privilege, if the caller explicitely specifies an
0000    272 ;                   access mode as an optional system service parameter.
0000    273 ;
0000    274 ;               2.  It is only necessary to have write access to the logical
0000    275 ;                   name table in order to be able to insert a logical name
0000    276 ;                   within it. Previously, both read and write access was
0000    277 ;                   required.
0000    278 ;
0000    279 ;               3.  In order to be able to insert a logical name into the system
0000    280 ;                   directory table a caller must be able to write access the
0000    281 ;                   table itself. Previously, the caller could perform such an
0000    282 ;                   insertion if they had SYSNAM privilege. The meaning of
0000    283 ;                   SYSNAM has been restricted to refer only to the access of
0000    284 ;                   the system logical name table, LNM$SYSTEM_TABLE, and to the
0000    285 ;                   creation and deletion of inner access mode logical names and
```

```
0000   286 ;                   tables.
0000   287 ;
0000   288 ;               4. It is never necessary to do any protection checking when
0000   289 ;                  process-private logical name tables are involved. Therefore,
0000   290 ;                  do not perform any protection checking prior to the insertion
0000   291 ;                  of new logical names into such tables.
0000   292 ;
0000   293 ;               Changes to EXE$DELLNM:
0000   294 ;
0000   295 ;               1. Change access mode processing. Logical names or logical
0000   296 ;                  name tables in access modes inner to that of the caller
0000   297 ;                  may not be deleted unless the user has SYSNAM privilege and
0000   298 ;                  has specified an access mode as an optional system service
0000   299 ;                  arguement. Logical names or logical name tables in access
0000   300 ;                  modes outer to the mode of the caller may still be deleted
0000   301 ;                  if the caller explicitely specifies an access mode as an
0000   302 ;                  optional system service parameter, even if the caller does
0000   303 ;                  not have SYSNAM privilege.
0000   304 ;
0000   305 ;               2. When a single logical name table entry was to be deleted
0000   306 ;                  this system service was requiring delete access to the
0000   307 ;                  containing logical name table before allowing the deletion
0000   308 ;                  to proceed. When all logical name table entries within a
0000   309 ;                  logical name table were to be deleted, no access checking
0000   310 ;                  was being done. This is incorrect. The access requirements
0000   311 ;                  for the deletion of logical name tables and logical names
0000   312 ;                  are as follows:
0000   313 ;
0000   314 ;                     a. To delete a logical name, write access to the containing
0000   315 ;                        logical name table is required.
0000   316 ;                     b. To delete a logical name table, either delete access to
0000   317 ;                        the table itself or write access to its containing
0000   318 ;                        logical name table (the system directory table) is
0000   319 ;                        required.
0000   320 ;
0000   321 ;               3. It is never necessary to do any protection checking when
0000   322 ;                  process-private logical name tables are involved. Therefore,
0000   323 ;                  do not perform any protection checking prior to the deletion
0000   324 ;                  of any process-private logical name table entry.
0000   325 ;
0000   326 ;               Changes to EXE$TRNLNM:
0000   327 ;
0000   328 ;               1. This system service would accept an invalid access mode as
0000   329 ;                  an optional system service parameter without returning an
0000   330 ;                  error. Add access mode error checking.
0000   331 ;
0000   332 ;               2. It is never necessary to do any protection checking when
0000   333 ;                  process-private logical name tables are involved. Therefore,
0000   334 ;                  do not perform any protection checking if the target logical
0000   335 ;                  name table entry is process-private.
0000   336 ;
0000   337 ;       V03-014 RAS0166         Ron Schaefer            5-Jul-1983
0000   338 ;               Reverse polarity of branch in positioning to a translation
0000   339 ;               index within a $TRNLNM itmlst.  Also optimize #-1s.
0000   340 ;
0000   341 ;       V03-013 WMC0001         Wayne Cardoza           23-Jun-1983
0000   342 ;               Add chained item lists, parent item code.
```

```
0000    343 ;
0000    344 ;       V03-012 RAS0160         Ron Schaefer            16-Jun-1983
0000    345 ;               Charge all arguments to be by-reference. This affects
0000    346 ;                   ACMODE      all services
0000    347 ;                   ATTR        $CRELNM, $CRELNT and $TRNLNM
0000    348 ;                   PROMSK      $CRELNT
0000    349 ;                   QUOTA       $CRELNT
0000    350 ;
0000    351 ;       V03-011 DMW4046         DMWalp                  9-Jun-1983
0000    352 ;               Post intergration of new logical name structures
0000    353 ;                   1.)  MAX_INDEX in TRNLNM returns the maximum index
0000    354 ;                        if non-negative, else -1
0000    355 ;                   2.)  Clean up the setting default index value of
0000    356 ;                        zero in TRNLNM
0000    357 ;                   3.)  Replace ^XFFFF with LNM$_LNMB_ADDR as item value
0000    358 ;
0000    359 ;       V03-010 DMW4038         DMWalp                  25-May-1983
0000    360 ;               Fix check for LNM$_LNMB_ADDR item
0000    361 ;
0000    362 ;       V03-009 RAS0158         Ron Schaefer            25-May-1983
0000    363 ;               Add protection structure and checking to logical
0000    364 ;               name tables.   Currenly, only SOGW protection is implemented.
0000    365 ;               Correct item list buffer probing in $TRNLNM to make
0000    366 ;               it tolerate longer than necessary buffers.
0000    367 ;
0000    368 ;       V03-008 DMW4029         DMWalp                  25-May-1983
0000    369 ;               Added code to CRELNM to allow MTL and mailbox UCB to be
0000    370 ;               intergrated with the new logical name structures. This
0000    371 ;               includes the addition of LNM$_LNMB_ADDR and work to allow
0000    372 ;               LNM$C_BACKPTR to be accpeted thru system service interface.
0000    373 ;
0000    374 ;       V03-007 TMK0004         Todd M. Katz            30-Apr-1983
0000    375 ;               When VMS must default name a new logical name table, it is
0000    376 ;               given a name of the form LNM$xxxxxxxx instead of a name of
0000    377 ;               the form LNT$xxxxxxxx.
0000    378 ;
0000    379 ;       V03-006 TMK0003         Todd M. Katz            25-Apr-1983
0000    380 ;               Make multiple to changes to the routines within this module:
0000    381 ;
0000    382 ;               EXE$CRELNT:
0000    383 ;
0000    384 ;               1. Un-comment the SETIPL that was commented out for debugging
0000    385 ;                  purposes.
0000    386 ;
0000    387 ;               2. Change several instructions to conform to VMS coding
0000    388 ;                  requirements.
0000    389 ;
0000    390 ;               3. If a process-private logical name table is to be created,
0000    391 ;                  the size of the logical name block is saved on the stack,
0000    392 ;                  the system pool logical name block is deallocated, the size
0000    393 ;                  is retrieved, and process-private P1 space is allocated. I
0000    394 ;                  was saving this size as a word on the stack, and this was
0000    395 ;                  creating problems allocating P1 space, after the size was
0000    396 ;                  popped off the stack because I neglected to zero out the
0000    397 ;                  high order word. To fix this problem, size is now saved as
0000    398 ;                  a longword on the stack.
0000    399 ;
```

```
0000    400 ;                   4. R2 is now saved before calling LNM$LOCKW (which destroys it)
0000    401 ;                      instead of after.
0000    402 ;
0000    403 ;                   5. Instead of picking up the QUOTA arguement twice from the
0000    404 ;                      user's parameter list, it is picked up and saved the first
0000    405 ;                      time it is needed.
0000    406 ;
0000    407 ;                   6. Pickup ACMODE as a byte instead of a longword.
0000    408 ;
0000    409 ;                   7. Change the name of TABNAM to RESNAM, TABLEN to RESLEN, and
0000    410 ;                      LOGNAM to TABNAM.
0000    411 ;
0000    412 ;                   8. If the new table is mapped to an existing table, then save
0000    413 ;                      the address of the existing LNMB on the stack before
0000    414 ;                      deleting the new LNMB so that the table's name maybe
0000    415 ;                      returned if required.
0000    416 ;
0000    417 ;                   EXE$CRELNM:
0000    418 ;
0000    419 ;                   1. Fix an incorrect probing of the TABNAM parameter.
0000    420 ;
0000    421 ;                   2. Un-comment the SETIPL that was commented out for debugging
0000    422 ;                      purposes.
0000    423 ;
0000    424 ;                   3. Do not perform any validation of the item list (except for
0000    425 ;                      the STRING item) during the first pass.
0000    426 ;
0000    427 ;                   4. Return SS$_NOGRPNAM instead of SS$_NOPRIV is the user does
0000    428 ;                      not have sufficient privilege to create a shareable logical
0000    429 ;                      name.
0000    430 ;
0000    431 ;                   5. If a process-private logical name is to be created, the size
0000    432 ;                      of the logical name block is saved on the stack, the system
0000    433 ;                      pool logical name block is deallocated, the size is
0000    434 ;                      retrieved, and process-private P1 space is allocated. I was
0000    435 ;                      saving this size as a word on the stack, and this was
0000    436 ;                      creating problems allocating P1 space, after the size was
0000    437 ;                      popped off the stack because I neglected to zero out the
0000    438 ;                      high order word. To fix this problem, size is now saved as
0000    439 ;                      a longword on the stack.
0000    440 ;
0000    441 ;                   6. Pickup the address of all item buffers before probing. After
0000    442 ;                      probing refer only to system service copies of these
0000    443 ;                      addresses, and not to user copies.
0000    444 ;
0000    445 ;                   7. As each translation block is created, check to make sure that
0000    446 ;                      creation of this block would not result in the exceeding of
0000    447 ;                      the space allocated for the logical name block.
0000    448 ;
0000    449 ;                   8. Change several instructions to conform to VMS coding
0000    450 ;                      requirements.
0000    451 ;
0000    452 ;                   9. Pickup ACMODE as a byte instead of a longword.
0000    453 ;
0000    454 ;                   EXE$DELLNM:
0000    455 ;
0000    456 ;                   1. Un-comment the SETIPL that was commented out for debugging
```

```
0000   457 ;                purposes.
0000   458 ;
0000   459 ;            2. When the deletion of a specific logical name in an access
0000   460 ;               mode other than user has been requested, all outer access
0000   461 ;               mode logical names with the same name and in the same table
0000   462 ;               are also deleted.
0000   463 ;
0000   464 ;            3. Pickup ACMODE as a byte instead of a longword.
0000   465 ;
0000   466 ;            EXE$TRNLNM:
0000   467 ;
0000   468 ;            1. Fix an incorrect probing of the TABNAM parameter.
0000   469 ;
0000   470 ;            2. Un-comment the SETIPL that was commented out for debugging
0000   471 ;               purposes.
0000   472 ;
0000   473 ;            3. Pickup the address of all item buffers before probing. After
0000   474 ;               probing refer only to system service copies of these
0000   475 ;               addresses, and not to user copies.
0000   476 ;
0000   477 ;            4. Change how the index of INDEX items are validated.
0000   478 ;
0000   479 ;            5. Change several instructions to conform to VMS coding
0000   480 ;               requirements.
0000   481 ;
0000   482 ;            6. Return the attribute bit LNM$V_SHAREABLE in the ATTRIBUTES
0000   483 ;               item whenever the name being translated is shareable.
0000   484 ;
0000   485 ;            7. Pickup ACMODE as a byte instead of a longword.
0000   486 ;
0000   487 ;            8. Change how the ACMODE arguement is interrupted. If the
0000   488 ;               arguement is 0, maximize with the mode of the caller;
0000   489 ;               otherwise, the specified mode is used to qualify the
0000   490 ;               search.
0000   491 ;
0000   492 ;            9. Change the attribute bit LNM$V_NOT_EXIST to LNM$V_EXISTS.
0000   493 ;               This bit will be set in the ATTRIBUTES item buffers of those
0000   494 ;               indexes that do exist, and clear in the ATTRIBUTES item
0000   495 ;               buffers of those indexes that do not exist.
0000   496 ;
0000   497 ;        V03-005 BLS0219         Benn Schreiber          19-Apr-1983
0000   498 ;                Make some BSBW's into JSB's with longword displacements.
0000   499 ;
0000   500 ;        V03-004 TMK0002         Todd M. Katz            18-Apr-1983
0000   501 ;                Fix a broken ASSUME statement. This ASSUME broke when I changed
0000   502 ;                the structure of a table header by removing the field
0000   503 ;                LNMTH$L_LOGNAM. Also, no longer fill in this "non-existant"
0000   504 ;                field when creating a logical name table.
0000   505 ;
0000   506 ;        V03-003 TMK0001         Todd M. Katz            18-Mar-1983
0000   507 ;                Re-write SYS$CRELNT. Add SYS$CRELNM, SYS$DELLNM, SYS$TRNLNM.
0000   508 ;
0000   509 ;--
```

```
                     0000    511          .SBTTL  DECLARATIONS
                     0000    512
                     0000    513 ;
                     0000    514 ; MACRO LIBRARY CALLS
                     0000    515 ;
                     0000    516
                     0000    517          $ARMDEF                         ;DEFINE ACCESS RIGHTS MASK
                     0000    518          $CADEF                          ;DEFINE CONDITIONAL ASSEMBLY SWITCHES
                     0000    519          $DYNDEF                         ;DEFINE DATA STRUCTURE TYPE CODES
                     0000    520          $IPLDEF                         ;DEFINE INTERRUPT PRIORITY LEVELS
                     0000    521          $LNMDEF                         ;DEFINE LOGICAL NAME ATTRIBUTES
                     0000    522          $LNMSTRDEF                      ;DEFINE LOGICAL NAME STRUCTURES OFFSETS
                     0000    523          $ORBDEF                         ;DEFINE OBJECT RIGHTS BLOCK OFFSETS
                     0000    524          $PCBDEF                         ;DEFINE PCB OFFSETS
                     0000    525          $PRDEF                          ;DEFINE PROCESSOR REGISTER NUMBERS
                     0000    526          $PRVDEF                         ;DEFINE PRIVILEGE BITS
                     0000    527          $PSLDEF                         ;DEFINE PROCESSOR STATUS FIELDS
                     0000    528          $SSDEF                          ;DEFINE SYSTEM STATUS VALUES
                     0000    529
                     0000    530 ;
                     0000    531 ; LOCAL SYMBOLS
                     0000    532 ;
                     0000    533 ; ARGUMENT LIST OFFSET DEFINITIONS FOR CREATE LOGICAL NAME TABLE.
                     0000    534 ;
                     0000    535
00000004             0000    536 CTATTR=4                                 ;ADDRESS OF TABLE ATTRIBUTES
00000008             0000    537 CTRESNAM=8                               ;ADDRESS OF TABLE NAME STRING DESCRIPTOR
0000000C             0000    538 CTRESLEN=12                              ;ADDRESS OF WORD TO RECEIVE LENGTH OF TABLE NAME
00000010             0000    539 CTQUOTA=16                               ;ADDRESS OF TABLE QUOTA
00000014             0000    540 CTPROT=20                                ;ADDRESS OF PROTECTION MASK
00000018             0000    541 CTTABNAM=24                              ;ADDRESS OF TABLE NAME STRING DESCRIPTOR
0000001C             0000    542 CTPARTAB=28                              ;PARENT TABLE NAME DESCRIPTOR
00000020             0000    543 CTACMODE=32                              ;ADDRESS OF ACCESS MODE
                     0000    544
                     0000    545 ;
                     0000    546 ; ARGUMENT LIST OFFSET DEFINITIONS FOR CREATE LOGICAL NAME.
                     0000    547 ;
                     0000    548
00000004             0000    549 CNATTR=4                                 ;ADDRESS OF ATTRIBUTES
00000008             0000    550 CNTABNAM=8                               ;ADDRESS OF TABLE NAME STRING DESCRIPTOR
0000000C             0000    551 CNLOGNAM=12                              ;ADDRESS OF LOGICAL NAME STRING DESCRIPTOR
00000010             0000    552 CNACMODE=16                              ;ADDRESS OF ACCESS MODE
00000014             0000    553 CNITMLST=20                              ;ADDRESS OF ITEM LIST
                     0000    554
                     0000    555 ;
                     0000    556 ; ARGUMENT LIST OFFSET DEFINITIONS FOR DELETE LOGICAL NAME.
                     0000    557 ;
                     0000    558
00000004             0000    559 DNTABNAM=4                               ;ADDRESS OF TABLE NAME STRING DESCRIPTOR
00000008             0000    560 DNLOGNAM=8                               ;ADDRESS OF LOGICAL NAME STRING DESCRIPTOR
0000000C             0000    561 DNACMODE=12                              ;ADDRESS OF ACCESS MODE
                     0000    562
                     0000    563 ;
                     0000    564 ; ARGUMENT LIST OFFSET DEFINITIONS FOR TRANSLATE LOGICAL NAME.
                     0000    565 ;
                     0000    566
00000004             0000    567 TRATTR=4                                 ;ADDRESS OF TRANSLATION FLAGS
```

```
00000008  0000    568 TRTABNAM=8                      ;ADDRESS OF TABLE NAME STRING DESCRIPTOR
0000000C  0000    569 TRLOGNAM=12                     ;ADDRESS OF LOGICAL NAME STRING DESCRIPTOR
00000010  0000    570 TRACMODE=16                     ;ADDRESS OF ACCESS MODE
00000014  0000    571 TRITMLST=20                     ;ADDRESS OF ITEM LIST
          0000    572
          0000    573 ;
          0000    574 ; PROTECTION MASK CONSTANTS FOR CHECKING.
          0000    575 ;
          0000    576
00000001  0000    577 READ_ACCESS     = ARM$M_READ
00000002  0000    578 WRITE_ACCESS    = ARM$M_WRITE
00000004  0000    579 QUOTA_ACCESS    = ARM$M_EXECUTE
00000008  0000    580 DELETE_ACCESS   = ARM$M_DELETE
          0000    581
          0000    582 ;
          0000    583 ; ASSUME STATEMENTS FOR LOGICAL NAME BLOCK, LOGICAL NAME TRANSLATION, AND
          0000    584 ; LOGICAL NAME TABLE HEADER.
          0000    585 ;
          0000    586
          0000    587         ASSUME  LNMB$B_TYPE+1,      EQ,  LNMB$B_ACMODE
          0000    588         ASSUME  LNMB$B_ACMODE+1,    EQ,  LNMB$L_TABLE
          0000    589         ASSUME  LNMB$L_TABLE+4,     EQ,  LNMB$B_FLAGS
          0000    590         ASSUME  LNMB$B_FLAGS+1,     EQ,  LNMB$T_NAME
          0000    591
          0000    592         ASSUME  LNMX$B_FLAGS,       EQ,  0
          0000    593         ASSUME  LNMX$B_FLAGS+1,     EQ,  LNMX$B_INDEX
          0000    594         ASSUME  LNMX$B_INDEX+1,     EQ,  LNMX$W_HASH
          0000    595         ASSUME  LNMX$W_HASH+2,      EQ,  LNMX$T_XLATION
          0000    596
          0000    597         ASSUME  LNMTH$B_FLAGS,      EQ,  0
          0000    598         ASSUME  LNMTH$B_FLAGS+1,    EQ,  LNMTH$L_HASH
          0000    599         ASSUME  LNMTH$L_HASH+4,     EQ,  LNMTH$L_ORB
          0000    600         ASSUME  LNMTH$L_ORB+4,      EQ,  LNMTH$L_NAME
          0000    601         ASSUME  LNMTH$L_NAME+4,     EQ,  LNMTH$L_PARENT
          0000    602         ASSUME  LNMTH$L_PARENT+4,   EQ,  LNMTH$L_CHILD
          0000    603         ASSUME  LNMTH$L_CHILD+4,    EQ,  LNMTH$L_SIBLING
          0000    604         ASSUME  LNMTH$L_SIBLING+4,  EQ,  LNMTH$L_QTABLE
          0000    605         ASSUME  LNMTH$L_QTABLE+4,   EQ,  LNMTH$L_BYTESLM
          0000    606         ASSUME  LNMTH$L_BYTESLM+4,  EQ,  LNMTH$L_BYTES
          0000    607
      00000000    608         .PSECT  YF$$LNM
          0000    609
          0000    610 ;
          000C    611 ; LOCAL DATA
          0000    61_ ;
          0000    613
      00000014    614 LNM_HEX_TAB_LEN = 4 + 8 + 8             ;SIZE OF "LNM$" + pid + addr
          0000    615 HEXDIGITS:
42 41 39 38 37 36 35 34 33 32 31 30
                46 45 44 43 000C    616         .ASCII  /0123456789ABCDEF/      ;TABLE TO CONVERT INTEGER TO HEX DIGIT
          0010    617
          0010    618 ;       .PAGE
```

```
0010   620          .SBTTL  EXE$CRELNT      - CREATE LOGICAL NAME TABLE
0010   621
0010   622 ;+
0010   623 ; EXE$CRELNT - CREATE LOGICAL NAME TABLE
0010   624 ;
0010   625 ; THIS SERVICE PROVIDES THE CAPABILITY TO CREATE A LOGICAL NAME TABLE.
0010   626 ;
0010   627 ; INPUTS:
0010   628 ;
0010   629 ;       CTATTR(AP)      = ADDRESS OF TABLE ATTRIBUTES.
0010   630 ;       CTRESNAM(AP)    = ADDRESS OF DESCRIPTOR TO RECEIVE TABLE NAME STRING.
0010   631 ;       CTRESLEN(AP)    = ADDRESS OF WORD TO RECEIVE LENGTH OF TABLE NAME.
0010   632 ;       CTQUOTA(AP)     = ADDRESS OF BYTE QUOTA FOR TABLE AND NAMES CONTAINED THEREI
0010   633 ;       CTPROT(AP)      = ADDRESS OF SOGW PROTECTION MASK.
0010   634 ;       CTTABNAM(AP)    = ADDRESS OF TABLE NAME STRING DESCRIPTOR.
0010   635 ;       CTPARTAB(AP)    = ADDRESS OF PARENT TABLE NAME DESCRIPTOR.
0010   636 ;       CTACMODE(AP)    = ADDRESS OF ACCESS MODE OF LOGICAL NAME TABLE TO BE CREATED
0010   637 ;
0010   638 ;       R4 = CURRENT PROCESS PCB ADDRESS.
0010   639 ;
0010   640 ; OUTPUTS:
0010   641 ;
0010   642 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO CREATE LOGICAL NAME TABLE ENTRY.
0010   643 ;
0010   644 ;               R0 = SS$_ACCVIO - TABLE NAME DESCRIPTOR, LOGICAL NAME
0010   645 ;                       DESCRIPTOR, LOGICAL NAME STRING, PARENT TABLE NAME
0010   646 ;                       DESCRIPTOR, PARENT TABLE NAME STRING CANNOT BE READ BY
0010   647 ;                       CALLING ACCESS MODE. TABLE NAME LENGTH WORD, TABLE NAME
0010   648 ;                       STRING BUFFER CANNOT BE WRITTEN BY CALLING ACCESS MODE.
0010   649 ;
0010   650 ;               R0 = SS$_BADPARAM - INVALID ATTRIBUTE OR ACCESS MODE SPECIFIED.
0010   651 ;                       PARENT LOGICAL NAME TABLE NAME NOT SPECIFIED.
0010   652 ;
0010   653 ;               R0 = SS$_DUPLNAM - ATTEMPT MADE TO SUPERSEDE NON-ALIASABLE
0010   654 ;                       LOGICAL NAME TABLE ENTRY OF THE SAME OR AN INNER ACCESS
0010   655 ;                       MODE.
0010   656 ;
0010   657 ;               R0 = SS$_EXLNMQUOTA - INSUFFICIENT QUOTA AVAILABLE IN THE
0010   658 ;                       PARENT LOGICAL NAME TABLE'S QUOTA TABLE FOR THE
0010   659 ;                       CREATION OF THE NEW LOGICAL NAME TABLE ENTRY.
0010   660 ;
0010   661 ;               R0 = SS$_INSFMEM - SUFFICIENT SYSTEM DYNAMIC MEMORY DOES NOT
0010   662 ;                       EXIST TO ALLOCATE THE NEW LOGICAL NAME TABLE ENTRY AND
0010   663 ;                       IMPLICIT RESOURCE WAIT IS NOT ENABLED.
0010   664 ;
0010   665 ;               R0 = SS$_IVLOGNAM - ZERO OR GREATER THAN MAXIMUM LENGTH
0010   666 ;                       LOGICAL OR PARENT TABLE NAME STRING SPECIFIED.
0010   667 ;
0010   668 ;               R0 = SS$_IVLOGTAB - INVALID PARENT TABLE NAME SPECIFIED OR
0010   669 ;                       INVALID LOGICAL NAME TABLE NAME SPECIFIED.
0010   670 ;
0010   671 ;               R0 = SS$_NOLOGTAB - PARENT TABLE SPECIFIED DOES NOT EXIST.
0010   672 ;
0010   673 ;               R0 = SS$_NOPRIV - PROCESS DOES NOT HAVE PRIVILEGE TO CREATE
0010   674 ;                       SPECIFIED LOGICAL NAME TABLE ENTRY.
0010   675 ;
0010   676 ;               R0 = SS$_PARENT_DEL - INSERTION OF NEW LOGICAL NAME TABLE ENTRY
```

```
                     0010   677 ;                       WOULD HAVE RESULTED IN THE DELETION OF A (GRAND)PARENT
                     0010   678 ;                       LOGICAL NAME TABLE ENTRY.
                     0010   679 ;
                     0010   680 ;               R0 = SS$_RESULTOVF - BUFFER NOT LARGE ENOUGH TO CONTAIN NAME
                     0010   681 ;                       OF NEW LOGICAL NAME TABLE.
                     0010   682 ;
                     0010   683 ;               R0 = SS$_TOOMANYLNM - TOO MANY LEVELS OF RECURSION IN SEARCH
                     0010   684 ;                       FOR PARENT TABLE.
                     0010   685 ;
                     0010   686 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                     0010   687 ;
                     0010   688 ;               R0 = SS$_LNMCREATED - NORMAL COMPLETION, NEW LOGICAL NAME TABLE
                     0010   689 ;                       ENTRY CREATED.
                     0010   690 ;
                     0010   691 ;               R0 = SS$_NORMAL - NORMAL COMPLETION, LNM$V_CREATE_IF SET AND
                     0010   692 ;                       THE LOGICAL NAME TABLE ENTRY ALREADY EXISTED.
                     0010   693 ;
                     0010   694 ;               R0 = SS$_SUPERSEDE - NORMAL COMPLETION, NEW LOGICAL NAME TABLE
                     0010   695 ;                       ENTRY SUPERSEDED A PREVIOUSLY EXISTING ENTRY AT THE SAME
                     0010   696 ;                       OR OUTER ACCESS MODE IN THE SPECIFIED PARENT LOGICAL
                     0010   697 ;                       NAME TABLE.
                     0010   698 ;
                     0010   699 ; SIDE EFFECTS:
                     0010   700 ;
                     0010   701 ;       THIS ROUTINE EXITS AT IPL 2, AND MUST CONTINUE TO DO SO, BECAUSE
                     0010   702 ;       IT IS CALLED AT SYSTEM INITIALIZATION TIME.
                     0010   703 ;
                     0010   704 ;-
                     0010   705
                00000000    706          .PSECT  Y$EXEPAGED
              0FFC  0000    707          .ENTRY  EXE$CRELNT,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
001F'           31  0002    708          BRW     EXECRELNT
                    0005    709
                00000010    710          .PSECT  YF$$LNM
                    0010    711          .ENABLE LSB
          50    0C  3C  0010    712 900$:    MOVZWL  #SS$_ACCVIO,R0          ;ACCESS VIOLATION
                    04  0013    713          RET
          50    14  3C  0014    714 910$:    MOVZWL  #SS$_BADPARAM,R0        ;BAD SYSTEM SERVICE PARAMETER
                    04  0017    715          RET
    50    0124  8F  3C  0018    716 920$:    MOVZWL  #SS$_INSFMEM,R0         ;NO PRIVILEGE TO CREATE TABLE ENTRY
                    04  001D    717 930$:    RET
    50    015C  8F  3C  001E    718 940$:    MOVZWL  #SS$_IVLOGTAB,R0        ;INVALID LOGICAL NAME TABLE NAME
                    04  0023    719          RET
                    0024    720
                    0024    721 ;
                    0024    722 ; VALIDATE AND COPY PARAMETERS AS NECESSARY. REGISTER ASSIGNMENT FOR THE
                    0024    723 ; PARAMETERS ARE AS FOLLOWS:
                    0024    724 ;
                    0024    725 ;       R5  = ACCESS MODE.
                    0024    726 ;       R6  = PROTECTION MASK.
                    0024    727 ;       R7  = ATTRIBUTE BITS.
                    0024    728 ;       R8  = ADDRESS OF PROBED AND COPIED PARENT TABLE NAME DESCRIPTOR.
                    0024    729 ;       R9  = ADDRESS OF PROBED AND COPIED TABLE NAME DESCRIPTOR.
                    0024    730 ;       R10 = ADDRESS OF PROBED AND COPIED LOGICAL NAME DESCRIPTOR.
                    0024    731 ;       R11 = ADDRESS OF WORD TO RECEIVE TABLE NAME LENGTH.
                    0024    732 ;
                    0024    733
```

```
                     0024   734 EXECRELNT:
                     0024   735         MOVL    CTRESLEN(AP),R11        ;ADDRESS OF TABLE NAME LENGTH WORD
                     0028   736         BEQL    10$                     ;BRANCH IF NOT PRESENT
                     002A   737         IFNOWRT #2,(R11),900$           ;CHECK WRITE ACCESS TO WORD
                     0030   738 10$:
                     0030   739         MOVL    CTPROT(AP),R6           ;ADDRESS OF PROTECTION MASK WORD
                     0034   740         BNEQ    15$                     ;BRANCH IF ONE PRESENT
                     0036   741         MOVZWL  #^XFF00,R6              ;DEFAULT THE PROTECTION TO S:RWED O:RWED
                     003B   742         BRB     20$
                     003D   743 15$:     IFNORD  #2,(R6),900$            ;CHECK READ ACCESS TO WORD
                     0043   744         MOVZWL  (R6),R6                 ;GET VALUE
                     0046   745
                     0046   746 20$:     MOVL    CTRESNAM(AP),R9         ;ADDRESS OF TABLE NAME DESCRIPTOR
                     004A   747         BEQL    30$                     ;BRANCH IF NOT PRESENT
                     004C   748         MOVL    R9,R1                   ;SET UP CALL
                     004F   749         JSB     L^EXE$PROBEW_DSC        ;PROBE DESCRIPTOR AND WRITE PROBE BUFFER
                     0055   750         BLBC    R0,930$                 ;CAN'T WRITE BUFFER OR READ DESCRIPTOR
                     0058   751         MOVQ    R1,-(SP)                ;SAVE TABLE NAME DESCRIPTOR
                     005B   752         MOVL    SP,R9                   ;ADDRESS OF TABLE NAME DESCRIPTOR
                     005E   753 30$:
                     005E   754         MOVL    CTATTR(AP),R7           ;TABLE ATTRIBUTES
                     0062   755         BEQL    35$                     ;BRANCH IF NOT PRESENT
                     0064   756         IFNORD  #4,(R7),900$            ;CHECK READ ACCESS TO VALUE
                     006A   757         MOVL    (R7),R7                 ;GET VALUE
                     006D   758         BITL    #^C< -
                     0074   759                 LNM$M_CONFINE! -        ;CONFINE TO PROCESS
                     0074   760                 LNM$M_CREATE_IF! -      ;SUPERCEDE VS. MAP
                     0074   761                 LNM$M_NO_ALIAS -        ;DO NOT ALLOW ALIASES
                     0074   762                 >,R7
                     0074   763         BNEQ    910$                    ;INVALID TABLE ATTRIBUTES
                     0076   764
                     0076   765         ASSUME  LNM$C_TABNAMLEN,LE,512
                     0076   766 35$:
                     0076   767         MOVL    CTTABNAM(AP),R10        ;ADDRESS OF TABLE NAME DESCRIPTOR
                     007A   768         BEQL    40$                     ;BRANCH IF NOT PRESENT
                     007C   769         IFNORD  #8,(R10),900$           ;CHECK ACCESS TO DESCRIPTOR
                     0082   770         MOVQ    (R10),R1                ;RETRIEVE TABLE NAME DESCRIPTOR
                     0085   771         MOVZWL  R1,R1                   ;ZERO HIGH ORDER WORD OF LENGTH
                     0088   772         BEQL    940$                    ;ERROR IF ZERO LENGTH TABLE NAME
                     008A   773         CMPW    R1,#LNM$C_TABNAMLEN     ;CHECK SIZE OF TABLE NAME STRING?
                     008D   774         BGTRU   940$                    ;ERROR IF SPECIFIED SIZE EXCEEDS MAXIMUM
                     008F   775         IFNORD  R1,(R2),44$             ;CHECK ACCESS TO TABLE NAME BUFFER
                     0095   776         MOVQ    R1,-(SP)                ;SAVE TABLE NAME DESCRIPTOR
                     0098   777         MOVL    SP,R10                  ;ADDRESS OF TABLE NAME DESCRIPTOR
                     009B   778 40$:
                     009B   779         MOVL    CTPARTAB(AP),R8         ;ADDRESS OF PARENT TABLE DESCRIPTOR
                     009F   780         BEQL    42$                     ;ERROR IF NOT PRESENT
                     00A1   781         IFNORD  #8,(R8),44$             ;CHECK ACCESS TO DESCRIPTOR
                     00A7   782         MOVAQ   (R8),R0                 ;ADDRESS OF DESCRIPTOR
                     00AA   783         BSBW    LNM$PROBER              ;PROBE PARENT TABLE NAME STRING
                     00AD   784         BLBC    R0,43$                  ;BRANCH IF CAN'T READ OR NOT PRESENT
                     00B0   785         MOVQ    R1,-(SP)                ;SAVE PARENT TABLE NAME DESCRIPTOR
                     00B3   786         MOVL    SP,R8                   ;ADDRESS OF PARENT TABLE NAME DESCRIPTOR
                     00B6   787
                     00B6   788         MOVL    CTACMODE(AP),R0         ;GET SPECIFIED ACCESS MODE
                     00BA   789         BEQL    50$                     ;BRANCH IF NOT PRESENT
                     00BC   790         IFNORD  #1,(R0),44$             ;CHECK READ ACCESS TO BYTE
```

```
                50   60   9A  00C2    791          MOVZBL  (R0),R0                    ;GET VALUE
                50   03   D1  00C5    792          CMPL    #PSL$C_USER,R0            ;CHECK FOR VALID ACCESS MODE
                     09   1E  00C8    793          BGEQU   45$                       ;OKAY
                   FF47   31  00CA    794 42$:     BRW     910$                      ;INVALID ACCESS MODE
                   FF4D   31  00CD    795 43$:     BRW     930$                      ;RETURN
                   FF3D   31  00D0    796 44$:     BRW     900$                      ;ACCESS VIOLATION
                          00D3    797
                          00D3    798 ;
                          00D3    799 ; IF THE ACCESS MODE OF THE NEW LOGICAL NAME TABLE WAS EXPLICITELY SPECIFIED
                          00D3    800 ; AND THE CALLER HAS THE SYSNAM PRIVILEGE, THEN THE NEW LOGICAL NAME TABLE IS
                          00D3    801 ; CREATED WITH THE SPECIFIED ACCESS MODE. OTHERWISE, THE ACCESS MODE OF THE
                          00D3    802 ; CALLER IS MAXIMIZED WITH ANY EXPLICTELY SPECIFIED ACCESS MODE AND USED TO
                          00D3    803 ; CREATE THE NEW LOGICAL NAME TABLE.
                          00D3    804 ;
                          00D3    805
                          00D3    806 45$:     IFPRIV  SYSNAM,60$                ;SYSNAM REQUIRED TO SPECIFY INNER MODE
00000000'GF       16      00D9    807 50$:     JSB     G^EXE$MAXACMODE           ;MAXIMIZE SPECIFIED MODE WITH CALLER'S
                55   50   D0  00DF    808 60$:     MOVL    R0,R5                     ;MODE TO DETERMINE MODE OF NEW TABLE
                          00E2    809
                          00E2    810 ;
                          00E2    811 ; RAISE IPL TO AST DELIVERY LEVEL SO THAT THERE ARE NO INTERRUPTIONS WHILE
                          00E2    812 ; THE LOGICAL NAME TABLE ENTRY IS BEING CREATED.
                          00E2    813 ;
                          00E2    814
                          00E2    815          SETIPL  S^#IPL$_ASTDEL            ;RAISE TO AST DELIVERY LEVEL
                     54   DD  00E5    816          PUSHL   R4                        ;SAVE THE PCB ADDRESS
                          00E7    817
                          00E7    818 ;
                          00E7    819 ; ALLOCATE A LOGICAL NAME BLOCK FOR THE NEW LOGICAL NAME TABLE ENTRY. ASSUME
                          00E7    820 ; THAT THE NEW LOGICAL NAME TABLE WILL BE SHAREABLE, SO ALLOCATE THE SPACE
                          00E7    821 ; FROM SYSTEM PAGED POOL. UNTIL THE PARENT LOGICAL TABLE BLOCK IS LOCATED,
                          00E7    822 ; WHETHER THE NEW LOGICAL NAME TABLE WILL BE IN SYSTEM POOL AND SHAREABLE
                          00E7    823 ; OR IN P1 SPACE AND NOT SHAREABLE IS NOT KNOWN. HOWEVER, THE LOGICAL NAME
                          00E7    824 ; MUTEX MUST BE LOCKED BEFORE A SEARCH CAN BE MADE FOR THE PARENT LOGICAL NAME
                          00E7    825 ; TABLE, AND ONCE THE MUTEX IS LOCKED, SYSTEM POOL CAN NOT ALTHOUGH P1 POOL CAN
                          00E7    826 ; BE ALLOCATED. THEREFORE, THE NEW LOGICAL NAME TABLE ENTRY IS ASSUMED TO BE
                          00E7    827 ; SHAREABLE, AND IF THE PARENT TABLE IS FOUND TO RESIDE IN SYSTEM POOL, THEN THE
                          00E7    828 ; SYSTEM POOL ALLOCATED FOR THE NEW ENTRY IS DEALLOCATED, AND PROCESS-PRIVATE
                          00E7    829 ; POOL IS ALLOCATED IN ITS PLACE.
                          00E7    830 ;
                          00E7    831
                     5A   D5  00E7    832          TSTL    R10                       ;LOGICAL NAME EXPLICITELY SPECIFIED?
                     05   13  00E9    833          BEQL    70$                       ;NO  - GO GET FIXED SIZE TO USE
                51   6A   D0  00EB    834          MOVL    (R10),R1                  ;YES - LOGICAL NAME SIZE IS USED AS
                     03   11  00EE    835          BRB     75$                       ;      SIZE OF LNMB NAME
                51   14   D0  00F0    836 70$:     MOVL    #LNM_HEX_TAB_LEN,R1       ;FIXED SIZE QUANTITY
                          00F3    837
                          00F3    838 75$:     ADDL2   #<-                       ;LENGTH OF ENTRY =
                          00F4    839                  LNMB$T_NAME+1+ -          ;SIZE OF LNMB + NAME SIZE COUNT FIELD
                          00F4    840                  LNMX$T_XLATION+1+ -       ;SIZE OF LNMX + HEADER SIZE COUNT FIELD
                          00F4    841                  LNMTH$K_LENGTH+ -         ;SIZE OF LNMTH
                          00F4    842                  ORB$C_LENGTH+ -           ;ORB SIZE
                          00F4    843                  1+ -                      ;TRAILER BYTE
                          00F4    844                  ^X07 -                    ;(ROUND LNMB TO NEXT QUADWORD BOUNDARY)
51   0000009C   8F       00F4    845                  >,R1                      ;SIZE OF LNMB NAME
                51   07   CA  00FA    846          BICL2   #^X07,R1                  ;TRUNCATE LNMB SIZE TO QUADWORD MULTIPLE
                00000000'GF       16  00FD    847          JSB     G^EXE$ALOPAGED            ;ALLOCATE THE LOGICAL NAME TABLE ENTRY
```

```
                                                                ;BRANCH IF SUFFICIENT MEMORY
                                                                ;OTHERWISE RETURN ERROR
                                                                ;SAVE NEW BLOCK SIZE WITHIN THE BLOCK

                                                                ;
                                                                ; WRITE LOCK THE LOGICAL NAME MUTEX AND THEN POSITION TO THE PARENT LOGICAL
                                                                ; NAME TABLE. DEALLOCATE THE NEW LNMB IF UNABLE TO POSITION TO THE PARENT
                                                                ; TABLE ENTRY.
                                                                ;

                                                                ;SAVE LOGICAL NAME BLOCK ADDRESS
                                                                ;LOCK TABLES FOR WRITING
                                                                ;ACCESS MODE
                                                                ;TABLE NAME DESCRIPTOR
                                                                ;SEARCH FOR SPECIFIED PARENT TABLE
                                                                ;RESTORE LOGICAL NAME BLOCK ADDRESS
                                                                ;BRANCH ON FAILURE

                                                                ;
                                                                ; IF THE PARENT LOGICAL NAME TABLE IS SHAREABLE THEN:
                                                                ;
                                                                ; 1. MARK THE NEW TABLE AS SHAREABLE.
                                                                ; 2. MAKE SURE THE NEW TABLE IS NOT MARKED AS CONFINED.
                                                                ; 3. MAKE SURE THE CALLER HAS QUOTA ACCESS TO THE PARENT LOGICAL NAME TABLE.
                                                                ; 3. MAKE SURE THAT THE CALLER HAS WRITE ACCESS TO THE SYSTEM DIRECTORY TABLE
                                                                ;    PROVIDED THE USER IS SPECIFYING THE NAME OF THE SHAREABLE TABLE AND NOT
                                                                ;    LETTING VMS DEFAULT NAME IT.
                                                                ;

00 E1 0124 878 80$: BBC #LNMTH$V_SHAREABLE,- ;BRANCH IF THE PARENT TABLE ENTRY IS IN
3B 61 0126 879 LNMTH$B_FLAGS(R1),90$ ;PROCESS-PRIVATE SPACE
57 02 CA 0128 880 BICL2 #LNM$M_CONFINE,R7 ;SHAREABLE TABLES CAN'T BE CONFINED
00 57 10 E3 012B 881 BBCS #LNM$V_SHAREABLE,R7,83$ ;MARK NEW TABLE AS SHAREABLE
012F 882
52 04 9A 012F 883 83$: MOVZBL #QUOTA_ACCESS,R2 ;CODE FOR ACCESS CHECK
54 DD 0132 884 PUSHL R4 ;SAVE LNMB PTR
54 04 AE D0 0134 885 MOVL 4(SP),R4 ;RESTORE PCB
FEC5' 30 0138 886 BSBW LNM$CHECK_PROT ;CHECK THE PROTECTION
54 8E D0 013B 887 MOVL (SP)+,R4 ;RESTORE LNMB
03 50 E8 013E 888 BLBS R0,85$ ;EXIT ON FAILURE
018C 31 0141 889 BRW 160$
0144 890
5A D5 0144 891 85$: TSTL R10 ;WAS A NAME SPECIFIED FOR THE NEW TABLE
59 13 0146 892 BEQL 100$ ;BRANCH IF ONE WASN'T
0148 893
52 02 9A 0148 894 MOVZBL #WRITE_ACCESS,R2 ;CODE FOR ACCESS CHECK
12 BB 014B 895 PUSHR #^M<R1,R4> ;SAVE LNMB AND PARENT TABLE HEADER ADDR
51 09 A1 D0 014D 896 MOVL LNMTH$L_NAME(R1),R1 ;RETRIEVE ADDRESS OF PARENT LNMB
51 0C A1 D0 0151 897 MOVL LNMB$L_TABLE(R1),R1 ;RETRIEVE SYSTEM DIRECTORY TABLE HEADER
54 08 AE D0 0155 898 MOVL 8(SP),R4 ;RESTORE PCB
FEA4' 30 0159 899 BSBW LNM$CHECK_PROT ;CHECK THE PROTECTION
12 BA 015C 900 POPR #^M<R1,R4> ;RESTORE LNMB AND PARENT TABLE HEADER
70 50 E9 015E 901 BLBC R0,105$ ;EXIT ON FAILURE
3E 11 0161 902 BRB 100$
0163 903
0163 904 ;
```

```
                                0163   905 ; IF THE PARENT TABLE IS DISCOVERED TO EXIST IN PROCESS-PRIVATE SPACE, THEN
                                0163   906 ; THIS IS ALSO WHERE THE NEW LOGICAL NAME TABLE ENTRY SHOULD GO. DEALLOCATE
                                0163   907 ; THE SYSTEM SPACE LNMB, AND ALLOCATE A SUFFICIENTLY SIZED BLOCK FROM P1
                                0163   908 ; SPACE.
                                0163   909 ;
                                0163   910
                      51    DD  0163   911 90$:     PUSHL   R1                          ;SAVE ADDRESS OF PARENT'S TABLE HEADER
                   7E   08 A4   3C  0165   912             MOVZWL  LNMB$W_SIZE(R4),-(SP)       ;SAVE SIZE OF LNMB AND COMPUTE SIZE
                6E 00000058 8F   C2  0169   913             SUBL2   #ORB$K_LENGTH,(SP)          ;OF BLOCK TO BE ALLOCATED FROM P1 POOL
                      50    54   D0  0170   914             MOVL    R4,R0                       ;ADDRESS OF SYSTEM SPACE LNMB
                         FE8A'  30  0173   915             BSBW    LNM$DELBLK                  ;DEALLOCATE SYSTEM SPACE LNMB
                                0176   916
                      51 8ED0   D0  0176   917             POPL    R1                          ;SIZE OF STORAGE TO ALLOCATE
                00000000'GF     16  0179   918             JSB     G^EXE$ALOP1PROC             ;ALLOCATE NEW LNMB FROM P1 SPACE
                         08 50  E8  017F   919             BLBS    R0,95$                      ;BRANCH IF SUCCESSFUL
                6E    0124 8F   3C  0182   920             MOVZWL  #SS$_INSFMEM,(SP)           ;OTHERWISE RETURN INSUFFICIENT MEMORY
                        0186    31  0187   921             BRW     190$                        ;ERROR AFTER RELEASING MUTEX
                                018A   922
                      54    52   D0  018A   923 95$:     MOVL    R2,R4                       ;ADDRESS OF NEW LNMB
                   08 A4    51   B0  018D   924             MOVW    R1,LNMB$W_SIZE(R4)          ;SAVE SIZE WITHIN BLOCK
                      51 8ED0   D0  0191   925             POPL    R1                          ;ADDRESS OF PARENT'S TABLE HEADER
                   52   09 A1   D0  0194   926             MOVL    LNMTH$L_NAME(R1),R2         ;ADDRESS OF PARENT'S LNMB
                      01    E1   0198   927             BBC     #LNMB$V_CONFINE,-           ;IS THE PARENT ENTRY MARKED CONFINE?
                   04 10 A2        019A   928                     LNMB$B_FLAGS(R2),100$       ;BRANCH IF NOT
                   00 57    01   E3  019D   929             BBCS    #LNM$V_CONFINE,R7,100$      ;MAKE SURE NEW ENTRY IS IF SO
                                01A1   930
                                01A1   931 ;
                                01A1   932 ; PERFORM QUOTA CHECKS. TWO SUCH QUOTA CHECKS ARE MADE:
                                01A1   933 ;
                                01A1   934 ; 1. THE CONTAINING TABLE (SYSTEM OR PROCESS DIRECTORY TABLE) IS CHECKED TO
                                01A1   935 ;    MAKE SURE IT HAS SUFFICIENT QUOTA TO CONTAIN THE NEW LOGICAL NAME TABLE
                                01A1   936 ;    ENTRY. NOTE THAT AN ASSUMPTION IS MADE THAT THE DIRECTORY TABLES ARE
                                01A1   937 ;    THEIR OWN QUOTA HOLDERS.
                                01A1   938 ;
                                01A1   939 ; 2. THE PARENT LOGICAL NAME TABLE'S QUOTA HOLDER IS CHECKED TO MAKE SURE IT HAS
                                01A1   940 ;    SUFFICIENT QUOTA FOR THE QUOTA WHICH WILL BE DEDUCTED FROM IT AND ALLOCATED
                                01A1   941 ;    SPECIFICALLY TO THE NEW TABLE.
                                01A1   942 ;
                                01A1   943
                   52   09 A1   D0  01A1   944 100$:   MOVL    LNMTH$L_NAME(R1),R2         ;RETRIEVE PARENT'S TABLE HEADER ADDRESS
                   52   0C A2   D0  01A5   945             MOVL    LNMB$L_TABLE(R2),R2         ;RETRIEVE CONTAINING TABLE'S ADDRESS
                   50   08 A4   3C  01A9   946             MOVZWL  LNMB$W_SIZE(R4),R0          ;SIZE OF NEW TABLE ENTRY
                   21 A2    50   D1  01AD   947             CMPL    R0,LNMTH$L_BYTES(R2)        ;IS THERE SUFFICIENT QUOTA?
                            19   1A  01B1   948             BGTRU   103$                        ;NO  - DEALLOCATE LNMB AND RETURN ERROR
                                01B3   949
                   58   10 AC   D0  01B3   950             MOVL    CTQUOTA(AP),R8              ;RETRIEVE NEW TABLE'S QUOTA
                            20   13  01B7   951             BEQL    110$                        ;SKIP SECOND QUOTA CHECK IF NOT PRESENT
                                01B9   952             IFNORD  #4,(R8),107$                ;CHECK READ ACCESS TO VALUE
                      58   68   D0  01BF   953             MOVL    (R8),R8                     ;GET VALUE
                                01C2   954
                   52   19 A1   D0  01C2   955 101$:   MOVL    LNMTH$L_QTABLE(R1),R2       ;RETRIEVE PARENT QUOTA HOLDER'S ADDRESS
                   21 A2    58   D1  01C6   956             CMPL    R8,LNMTH$L_BYTES(R2)        ;IS THERE SUFFICIENT QUOTA?
                            0D   1B  01CA   957             BLEQU   110$                        ;YES - THEN CONTINUE
                50  224C 8F   3C  01CC   958 103$:   MOVZWL  #SS$_EXLNMQUOTA,R0          ;NO  - THEN DEALLOCATE LNMB AND RETURN
                         00FC   31  01D1   959 105$:   BRW     160$                        ;      EXCEEDED QUOTA ERROR
                                01D4   960
                      50   0C   3C  01D4   961 107$:   MOVZWL  #SS$_ACCVIO,R0              ;NO ACCESS
```

```
                F8   11   01D7   962          BRB     105$                    ;EXIT
                          01D9   963
                          01D9   964 ;
                          01D9   965 ; FILL IN THE LNMB PORTION OF THE LOGICAL NAME BLOCK FOR THE NEW LOGICAL NAME
                          01D9   966 ; TABLE ENTRY.
                          01D9   967 ;
                          01D9   968 ; REGISTER USAGE IS AS FOLLOWS:
                          01D9   969 ;
                          01D9   970 ;       R1  = ADDRESS OF TABLE HEADER FOR PARENT LOGICAL NAME TABLE.
                          01D9   971 ;       R3  = SIZE OF NEW LOGICAL NAME TABLE ENTRY'S NAME.
                          01D9   972 ;       R4  = ADDRESS OF LOGICAL NAME BLOCK FOR NEW LOGICAL NAME TABLE.
                          01D9   973 ;       R5  = ACCESS MODE.
                          01D9   974 ;       R6  = ADDRESS OF PROTECTION MASK.
                          01D9   975 ;       R7  = ATTRIBUTE BITS.
                          01D9   976 ;       R8  = QUOTA.
                          01D9   977 ;       R9  = ADDRESS OF PROBED AND COPIED TABLE NAME DESCRIPTOR.
                          01D9   978 ;       R10 = ADDRESS OF PROBED AND COPIED LOGICAL NAME DESCRIPTOR.
                          01D9   979 ;       R11 = ADDRESS OF WORD TO RECEIVE TABLE NAME LENGTH.
                          01D9   980 ;
                          01D9   981
          50    0A A4    9E   01D9   982 110$:   MOVAB   LNMB$B_TYPE(R4),R0      ;POSITION TO BLOCK TYPE
          80    40 8F    90   01DD   983         MOVB    #DYN$C_LNM,(R0)+        ;SET DATA STRUCTURE TYPE
                80  55    90   01E1   984         MOVB    R5,(R0)+                ;SET OWNER ACCESS MODE
                          01E4   985
          52    09 A1    D0   01E4   986         MOVL    LNMTH$L_NAME(R1),R2     ;ADDRESS OF PARENT'S LOGICAL NAME BLOCK
          80    0C A2    D0   01E8   987         MOVL    LNMB$L_TABLE(R2),(R0)+  ;STORE DIRECTORY TABLE HEADER ADDRESS
                          01EC   988
        80  08    57    89   01EC   989         BISB3   R7,#LNMB$M_TABLE,(R0)+  ;STORE NAME ATTRIBUTES
                          01F0   990
                          01F0   991 ;
                          01F0   992 ; IF THE CALLER HAS SPECIFIED A LOGICAL NAME STRING THEN THAT STRING BECOMES
                          01F0   993 ; THE NAME OF THE LNMB FOR THE NEW LOGICAL NAME TABLE ENTRY BY MOVING IT INTO
                          01F0   994 ; THE LOGICAL NAME BLOCK. OTHERWISE, THE UNIQUE NAME OF THE LNMB IS A
                          01F0   995 ; CONSTRUCTED QUANTITY OF FIXED (12 BYTES) SIZE.
                          01F0   996 ;
                          01F0   997
                5A    D5   01F0   998         TSTL    R10                     ;LOGICAL NAME STRING SPECIFIED?
                2A    13   01F2   999         BEQL    120$                    ;NO  - BRANCH TO CONSTRUCT UNIQUE NAME
          80    6A    90   01F4  1000         MOVB    (R10),(R0)+             ;STORE LENGTH OF NAME
          7E    54    7D   01F7  1001         MOVQ    R4,-(SP)                ;SAVE NEEDED REGISTERS OVER MOVTUC
                51    DD   01FA  1002         PUSHL   R1
    00    04 BA    6A    2F   01FC  1003         MOVTUC  (R10),@4(R10),#0,-      ;CHECK FOR PROPER TABLE NAME SYNTAX AND
        00000000'EF               0201  1004                 EXE$LNM_SYNTAX_DAT,-    ;AT THE SAME TIME MOVE THE TABLE NAME
          60    6A                0206  1005                 (R10),(R0)              ;INTO THE LOGICAL NAME TABLE ENTRY
                0B    1D   0208  1006         BVS     115$                    ;BRANCH IF TABLE NAME HAS INVALID CHARS
          50    55    D0   020A  1007         MOVL    R5,R0                   ;POSITION PAST LNMB NAME STRING
                51  8ED0   020D  1008         POPL    R1                      ;RESTORE SAVED REGISTERS
          54    8E    7D   0210  1009         MOVQ    (SP)+,R4
                46    11   0213  1010         BRB     130$                    ;BRANCH AROUND UNIQUE NAME CONSTRUCTION
                          0215  1011
                32    BA   0215  1012 115$:   POPR    #^M<R1,R4,R5>           ;RESTORE REGISTERS
      50    015C 8F    3C   0217  1013         MOVZWL  #SS$_IVLOGTAB,R0        ;INVALID TABLE NAME STRING SPECIFIED
                B3    11   021C  1014         BRB     105$                    ;SO GO RETURN THE APPROPRIATE ERROR
                          021E  1015
          80    14    90   021E  1016 120$:   MOVB    #LNM_HEX_TAB_LEN,(R0)+  ;STORE LENGTH OF NAME
    80  244D4E4C 8F    D0   0221  1017         MOVL    #^A/LNM$7,(R0)+         ;STORE BEGINNING OF NAME
          53    1C    D0   0228  1018         MOVL    #<8-1>*4,R3             ;STARTING NIBBLE POSITION
```

```
                52 54 04 53   EF  022B  1019 125$:   EXTZV   R3,#4,R4,R2              ;EXTRACT HEX DIGIT
                80 FDCB CF42  90  0230  1020         MOVB    HEXDIGITS[R2],(R0)+     ;STORE ASCII EQUIVALENT
                   53 04      C2  0236  1021         SUBL    #4,R3                   ;ADJUST POSITION TO SUCCEEDING NIBBLE
                      F0      18  0239  1022         BGEQ    125$                    ;ITERATE OVER 8 DIGITS
                      54      DD  023B  1023         PUSHL   R4                      ;SAVE ADDR OF LNMB
                54 04 AE      D0  023D  1024         MOVL    4(SP),R4                ;GET PCB ADDR
                54 64 A4      D0  0241  1025         MOVL    PCB$L_EPID(R4),R4       ;GET PID
                   53 1C      D0  0245  1026         MOVL    #<8-1>*4,R3             ;STARTING NIBBLE POSITION
                52 54 04 53   EF  0248  1027 126$:   EXTZV   R3,#4,R4,R2              ;EXTRACT HEX DIGIT
                80 FDAE CF42  90  024D  1028         MOVB    HEXDIGITS[R2],(R0)+     ;STORE ASCII EQUIVALENT
                   53 04      C2  0253  1029         SUBL    #4,R3                   ;ADJUST POSITION TO SUCCEEDING NIBBLE
                      F0      18  0256  1030         BGEQ    126$                    ;ITERATE OVER 8 DIGITS
                      54 8E   D0  0258  1031         MOVL    (SP)+,R4                ;RESTORE LNMB ADDR
                              0258  1032 ;
                              0258  1033 ; FILL IN THE LNMX PORTION OF THE LOGICAL NAME BLOCK FOR THE NEW LOGICAL NAME
                              0258  1034 ; TABLE ENTRY. FOR LOGICAL NAME TABLES THE SOLE TRANSLATION CONSISTS OF THE
                              0258  1035 ; TABLE HEADER.
                              0258  1036 ;
                              0258  1037
                   80 02      90  0258  1038 130$:   MOVB    #LNMX$M_TERMINAL,(R0)+  ;STORE TRANSLATION ATTRIBUTES
                80 82 8F      90  025E  1039         MOVB    #LNMX$C_TABLE,(R0)+     ;STORE TRANSLATION INDEX (SPECIAL)
                      80      B4  0262  1040         CLRW    (R0)+                   ;INITIALIZE HASH CODE LOCATION TO 0
                   80 25      90  0264  1041         MOVB    #LNMTH$K_LENGTH,(R0)+   ;STORE LENGTH OF TABLE HEADER
                              0267  1042
                              0267  1043 ;
                              0267  1044 ; FILL IN THE LNMTH PORTION OF THE LOGICAL NAME BLOCK FOR THE NEW LOGICAL NAME
                              0267  1045 ; TABLE ENTRY. THE TABLE HEADER OF THE NEW ENTRY CONSTIUTES THE SOLE
                              0267  1046 ; TRANSLATION OF THE NEW LOGICAL NAME TABLE ENTRY.
                              0267  1047 ;
                              0267  1048
                   52 50      D0  0267  1049         MOVL    R0,R2                   ;SAVE ADDRESS OF TABLE HEADER
                53 57 10      9C  026A  1050         ROTL    #16,R7,R3               ;TABLE ATTRIBUTES TO LOW BYTE OF R3
                   80 53      90  026E  1051         MOVB    R3,(R0)+                ;STORE TABLE HEADER ATTRIBUTES
                              0271  1052
                80 01 A1      D0  0271  1053         MOVL    LNMTH$L_HASH(R1),(R0)+  ;HASH TABLE ADDRESS IS SAME AS PARENTS
                              0275  1054
                      80      D4  0275  1055         CLRL    (R0)+                   ;ZERO OBJECT RIGHTS BLOCK ADDR FOR NOW
                   80 64      9E  0277  1056         MOVAB   (R4),(R0)+              ;STORE CONTAINING LNMB BLOCK ADDRESS
                   80 51      D0  027A  1057         MOVL    R1,(R0)+                ;STORE PARENT TABLE HEADER ADDRESS
                      80      7C  027D  1058         CLRQ    (R0)+                   ;STORE CHILD TABLE HEADER ADDRESS
                              027F  1059                                             ;STORE SIBLING TABLE HEADER ADDRESS
                              027F  1060
                80 19 A1      D0  027F  1061         MOVL    LNMTH$L_QTABLE(R1),(R0)+;ASSUME QUOTA HOLDER IS SAME AS PARENTS
                   60 58      D0  0283  1062         MOVL    R8,(R0)                 ;STORE TABLE QUOTA
                      04      13  0286  1063         BEQL    135$                    ;BRANCH IF POOLED QUOTA
                19 A2 52      D0  0288  1064         MOVL    R2,LNMTH$L_QTABLE(R2)   ;NEW ENTRY IS A QUOTA TABLE
                   80 80      D0  028C  1065 135$:   MOVL    (R0)+,(R0)+             ;STORE QUOTA BYTES REMAINING (= QUOTA)
                              028F  1066
                              028F  1067 ;
                              028F  1068 ; FILL IN THE NEXT LNMX PORTION OF THE LOGICAL NAME BLOCK FOR THE NEW LOGICAL
                              028F  1069 ; NAME TABLE ENTRY. THE LAST TRANSLATION BLOCK CONSISTS SOLELY OF A FLAGS
                              028F  1070 ; FIELD, AND IS MARKED WITHIN THIS FLAGS FIELD AS THE LAST LNMX.
                              028F  1071 ;
                              028F  1072
                   80 04      90  028F  1073         MOVB    #LNMX$M_XEND,(R0)+      ;STORE END FLAG
                              0292  1074
                              0292  1075 ;
```

```
                     0292  1076 ; INITIALIZE THE OBJECT RIGHTS BLOCK WHICH EXISTS FOR ALL SHAREABLE LOGICAL NAME
                     0292  1077 ; TABLES. PROCESS-PRIVATE LOGICAL NAME TABLES DO NOT REQUIRE SUCH A ORB BECAUSE
                     0292  1078 ; BECAUSE PROTECTION CHECKING IS NOT REQUIRED, AND IS NEVER PERFORMED ON THEM.
                     0292  1079 ;
                     0292  1080
          00   E1    0292  1081          BBC     #LNMTH$V_SHAREABLE,-    ;IS PARENT LOGICAL NAME TABLE SHAREABLE?
        03 61          0294  1082                  LNMTH$B_FLAGS(R1),137$  ;IF NOT, SKIP INITIALIZATION OF ORB
         FD67'   30    0296  1083          BSBW    LNM$INIT_PROT           ;INITIALIZE TABLE'S OBJECT RIGHTS BLOCK
    56   52    D0      0299  1084 137$:    MOVL    R2,R6                   ;SAVE TABLE HEADER ADDRESS
                     029C  1085
                     029C  1086 ;
                     029C  1087 ; SET THE TEMPORARY BIT LNMB$V_NODELETE IN THE FLAGS BYTE OF THE LNMB BLOCK OF
                     029C  1088 ; THE NEW LOGICAL NAME TABLE ENTRY'S PARENT, GRANDPARENT ETC... THIS WILL
                     029C  1089 ; PREVENT THE NEW ENTRY FROM BEING INSERTED IF ITS INSERTION WOULD RESULT IN
                     029C  1090 ; THE DELETION OF ANY LOGICAL NAME TABLE ENTRY IN ITS DIRECT LINE OF DESCENT;
                     029C  1091 ; AND THUS, THE DELETION OF ITS PARENT TABLE.
                     029C  1092 ;
                     029C  1093
          01   E0    029C  1094 140$:    BBS     #LNMTH$V_DIRECTORY,-    ;SYSTEM/PROCESS DIRECTORY TABLE?
        0E 61          029E  1095                  LNMTH$B_FLAGS(R1),150$  ;YES - GO INSERT NEW TABLE ENTRY
    52  09 A1    D0    02A0  1096          MOVL    LNMTH$L_NAME(R1),R2     ;ADDRESS OF PARENT'S LNMB BLOCK
          10   88    02A4  1097          BISB2   #LNMB$M_NODELETE,-      ;SET THE TEMPORARY BIT BLOCKING DELETION
        10 A2          02A6  1098                  LNMB$B_FLAGS(R2)        ;OF THIS LOGICAL NAME TABLE ENTRY
    51  0D A1    D0    02A8  1099          MOVL    LNMTH$L_PARENT(R1),R1   ;NO  - RETRIEVE PARENT'S TABLE HEADER
          EE   11    02AC  1100          BRB     140$                    ;      ADDRESS AND CONTINUE
                     02AE  1101
                     02AE  1102 ;
                     02AE  1103 ; AT THIS POINT ALL CHECKS HAVE BEEN MADE, AND THE TABLE STRUCTURES MAY NOW
                     02AE  1104 ; BE MODIFIED FOR THE FIRST TIME. INSERT THE NEW LOGICAL NAME TABLE ENTRY AND
                     02AE  1105 ; THEN CLEAR THE TEMPORARY DO NOT DELETE BIT, LNMB$V_NODELETE REGARDLESS OF THE
                     02AE  1106 ; OUTCOME OF THE INSERTION.
                     02AE  1107 ;
                     02AE  1108 ; THIS INSERTION MAY TAKE ONE OF THREE FORMS:
                     02AE  1109 ;
                     02AE  1110 ; 1. THE NEW LOGICAL NAME TABLE ENTRY MAYBE INSERTED AS A NEW LOGICAL NAME
                     02AE  1111 ;    TABLE. FOR THIS CASE TO OCCUR THE FOLLOWING CONDITIONS MUST HOLD TRUE.
                     02AE  1112 ;
                     02AE  1113 ;       A. THERE IS NO EXISTING LOGICAL NAME TABLE ENTRY IN THE SAME DIRECTORY
                     02AE  1114 ;          (PROCESS OT SYSTEM) WITH THE SAME NAME AND ACCESS MODE.
                     02AE  1115 ;
                     02AE  1116 ;       B. THERE IS NO EXISTING LOGICAL NAME TABLE ENTRY IN THE SAME DIRECTORY
                     02AE  1117 ;          (PROCESS OR SYSTEM) WITH THE SAME NAME AND AN INNER ACCESS MODE THAT
                     02AE  1118 ;          DOES NOT ALLOW ALIASES.
                     02AE  1119 ;
                     02AE  1120 ; 2. THE LOGICAL NAME TABLE ENTRY MAYBE ''MAPPED'' INTO AN EXISTING LOGICAL NAME
                     02AE  1121 ;    TABLE ENTRY. FOR THIS CASE TO OCCUR THE FOLLOWING CONDITIONS MUST BE MET.
                     02AE  1122 ;
                     02AE  1123 ;       A. THE LNM$V_CREATE_IF BIT MUST HAVE BEEN SET IN THE ATTRIBUTES FIELD.
                     02AE  1124 ;
                     02AE  1125 ;       B. THERE MUST EXIST A LOGICAL NAME TABLE ENTRY WITH THE SAME NAME AND
                     02AE  1126 ;          ACCESS MODE WITHIN THE SAME DIRECTORY (PROCESS OR SYSTEM).
                     02AE  1127 ;
                     02AE  1128 ; 3. THE LOGICAL NAME TABLE ENTRY MAYBE INSERTED SUPERSEDING AN EXISTING ENTRY
                     02AE  1129 ;    IN THE SAME DIRECTORY (PROCESS OR SYSTEM) WITH THE SAME NAME AND ACCESS
                     02AE  1130 ;    MODE. THE OLD LOGICAL NAME TABLE ENTRY TOGETHER WITH ITS ENTIRE HIERARCHY
                     02AE  1131 ;    OF LOGICAL NAMES AND LOGICAL NAME TABLE ENTRIES IS DELETED, PROVIDED OF
                     02AE  1132 ;    COURSE, THAT THE OLD ENTRY WAS FOR A LOGICAL NAME TABLE. THE CONDITIONS
```

```
                     02AE  1133 ;    WHICH MUST BE MET FOR THIS CASE TO OCCUR ARE:
                     02AE  1134 ;
                     02AE  1135 ;    A. THE LNM$V_CREATE_IF BIT MUST NOT HAVE BEEN SET IN THE ATTRIBUTES FIELD.
                     02AE  1136 ;
                     02AE  1137 ;    B. THERE MUST EXIST A LOGICAL NAME TABLE ENTRY WITH THE SAME NAME AND
                     02AE  1138 ;       ACCESS MODE WITHIN THE SAME DIRECTORY (PROCESS OR SYSTEM).
                     02AE  1139 ;
                     02AE  1140 ; IF A CASE 1 OR 3 INSERTION IS PERFORMED, AND THE NEW LOGICAL NAME TABLE ENTRY
                     02AE  1141 ; DOES NOT ALLOW ALIASES, THEN ANY LOGICAL NAME TABLE ENTRIES IN THE SAME
                     02AE  1142 ; DIRECTORY (PROCESS OR SYSTEM) WITH THE SAME NAME BUT AT AN OUTER ACCESS MODES
                     02AE  1143 ; ARE DELETED ALONG WITH THEIR HIERARCHY OF LOGICAL NAMES AND LOGICAL NAME
                     02AE  1144 ; TABLE ENTRIES, PROVIDED THEY TOO ARE LOGICAL NAME TABLES.
                     02AE  1145 ;
                     02AE  1146
   51   54   D0      02AE  1147 150$:   MOVL    R4,R1                       ;ADDRESS OF NEW LOGICAL NAME TABLE ENTRY
   52   57   D0      02B1  1148         MOVL    R7,R2                       ;ATTRIBUTES
      FD49'  30      02B4  1149         BSBW    LNM$INSLOGTAB               ;INSERT NEW LOGICAL NAME TABLE ENTRY
   53   51   D0      02B7  1150         MOVL    R1,R3                       ;SAVE LNMB ADDRESS OF MAPPED-TO ENTRY
                     02BA  1151                                             ;IF SUCH A MAPPING HAS TAKEN PLACE
                     02BA  1152
51   0D A6   D0      02BA  1153         MOVL    LNMTH$L_PARENT(R6),R1       ;RETRIEVE PARENT'S TABLE HEADER
        01   E0      02BE  1154 155$:   BBS     #LNMTH$V_DIRECTORY,-        ;SYSTEM/PROCESS DIRECTORY TABLE?
     0E 61           02C0  1155         LNMTH$B_FLAGS(R1),160$      ;YES - GO CHECK INSERTION
52   09 A1   D0      02C2  1156         MOVL    LNMTH$L_NAME(R1),R2         ;ADDRESS OF PARENT'S LNMB BLOCK
        10   8A      02C6  1157         BICB2   #LNMB$M_NODELETE,-          ;CLEAR THE TEMPORARY BIT BLOCKING
     10 A2           02C8  1158         LNMB$B_FLAGS(R2)            ;DELETION OF THIS LNMB
51   0D A1   D0      02CA  1159         MOVL    LNMTH$L_PARENT(R1),R1       ;NO  - RETRIEVE PARENT'S TABLE HEADER
        EE   11      02CE  1160         BRB     155$                        ;      ADDRESS AND CONTINUE
                     02D0  1161
        50   DD      02D0  1162 160$:   PUSHL   R0                          ;SAVE STATUS
     08 50   E8      02D2  1163         BLBS    R0,170$                     ;BRANCH ON SUCCESS
   50   54   D0      02D5  1164         MOVL    R4,R0                       ;LOGICAL NAME BLOCK FOR NEW TABLE ENTRY
      FD25'  30      02D8  1165         BSBW    LNM$DELBLK                  ;DELETE BLOCK CONTAINING NEW TABLE ENTRY
        33   11      02DB  1166         BRB     190$                        ;JOIN MAIN EXIT - STATUS ON STACK
                     02DD  1167
                     02DD  1168 ;
                     02DD  1169 ; AT THIS POINT THE NEW LOGICAL NAME TABLE HAS SUCCESSFULLY BEEN INSERTED.
                     02DD  1170 ; FILL IN ANY SPECIFIED OUTPUT PARAMETERS (TABLEN AND/OR TABNAM) BEFORE
                     02DD  1171 ; RETURNING THE STATUS TO THE CALLER OF THE SYSTEM SERVICE. IF THE NEW TABLE
                     02DD  1172 ; ENTRY WAS JUST MAPPED TO AN EXISTING TABLE ENTRY, THEN THE LOGICAL NAME BLOCK
                     02DD  1173 ; CONTAINING THE NEW TABLE ENTRY IS DELETED BEFORE RETURNING.
                     02DD  1174 ;
                     02DD  1175
   50   01   D1      02DD  1176 170$:   CMPL    #SS$_NORMAL,R0              ;WAS NEW ENTRY MAPPED TO EXISTING ONE?
        0B   12      02E0  1177         BNEQ    175$                        ;BRANCH IF NOT
   50   54   D0      02E2  1178         MOVL    R4,R0                       ;ADDRESS OF NEW ENTRY
        53   DD      02E5  1179         PUSHL   R3                          ;SAVE ADDRESS OF ENTRY MAPPED TO
      FD16'  30      02E7  1180         BSBW    LNM$DELBLK                  ;DELETE NEW ENTRY
        53 8ED0      02EA  1181         POPL    R3                          ;ADDRESS OF ENTRY MAPPED TO
                     02ED  1182
51   11 A4   9E      02ED  1183 175$:   MOVAB   LNMB$T_NAME(R4),R1          ;ADDRESS OF TABLE NAME COUNTED STRING
   50   81   9A      02F1  1184         MOVZBL  (R1)+,R0                    ;LENGTH OF NAME STRING
                     02F4  1185
        5B   D5      02F4  1186         TSTL    R11                         ;RETURN LENGTH?
        03   13      02F6  1187         BEQL    180$                        ;BRANCH IF NO
   6B   50   B0      02F8  1188         MOVW    R0,(R11)                    ;RETURN LENGTH
                     02FB  1189
```

```
                                                B 7
                59   D5  02FB  1190 180$:   TSTL    R9                      ;WANT LOGICAL NAME TABLE NAME RETURNED?
                11   13  02FD  1191         BEQL    190$                    ;BRANCH IF NO
             69 50   B1  02FF  1192         CMPW    R0,(R9)                 ;ENOUGH ROOM?
                07   14  0302  1193         BGTR    185$                    ;BRANCH IF NOT ENOUGH ROOM
 04 B9      61 50   28  0304  1194         MOVC3   R0,(R1),@4(R9)          ;RETURN NAME STRING
                05   11  0309  1195         BRB     190$
    6E    0214 8F   3C  030B  1196 185$:   MOVZWL  #SS$_RESULTOVF,(SP)     ;NOT ENOUGH ROOM FOR NAME
                     0310  1197
       54   04 AE   D0  0310  1198 190$:   MOVL    4(SP),R4                ;RETRIEVE PCB ADDRESS
             FCE9'   30  0314  1199         BSBW    LNM$UNLOCK              ;UNLOCK TABLES
                50 8ED0  0317  1200         POPL    R0                      ;RESTORE STATUS
                   04  031A  1201         RET                             ;RETURN TO CALLER
                       031B  1202
                       031B  1203         .DISABLE        LSB
                       031B  1204 ;       .PAGE
```

```
                        .SBTTL  EXE$CRELNM      - CREATE LOGICAL NAME
;+
; EXE$CRELNM - CREATE LOGICAL NAME
;
; THIS SERVICE PROVIDES THE CAPABILITY TO CREATE A LOGICAL NAME ENTRY WITH
; SOME NUMBER OF EQUIVALENCE STRINGS.
;
; INPUTS:
;
;       CNATTR(AP)      = ADDRESS OF LOGICAL NAME ATTRIBUTES.
;       CNTABNAM(AP)    = ADDRESS OF TABLE NAME STRING DESCRIPTOR.
;       CNLOGNAM(AP)    = ADDRESS OF LOGICAL NAME STRING DESCRIPTOR.
;       CNACMODE(AP)    = ADDRESS OF ACCESS MODE OF LOGICAL NAME TO BE CREATED.
;       CNITMLST(AP)    = ADDRESS OF ITEM LIST DEFINING TRANSLATIONS.
;
;       R4 = CURRENT PROCESS PCB ADDRESS.
;
; OUTPUTS:
;
;       R0 LOW BIT CLEAR INDICATES FAILURE TO CREATE LOGICAL NAME TABLE ENTRY.
;
;               R0 = SS$_ACCVIO - LOGICAL NAME DESCRIPTOR, LOGICAL NAME STRING,
;                       TABLE NAME DESCRIPTOR, TABLE NAME STRING, AN ITEM IN THE
;                       ITEM LIST, AN INDEX ITEM BUFFER, AN ATTRIBUTES ITEM
;                       BUFFER, A STRING ITEM BUFFER, CANNOT BE READ BY CALLING
;                       ACCESS MODE. TABLE ITEM BUFFER, TABLE ITEM SIZE BUFFER
;                       CANNOT BE WRITTEN BY CALLING ACCESS MODE.
;
;               R0 = SS$_BADPARAM - INVALID ATTRIBUTE, ACCESS MODE, ITEM TYPE,
;                       ITEM LENGTH, ITEM SPECIFIED. LOGICAL NAME DESCRIPTOR,
;                       TABLE NAME DESCRIPTOR NOT SPECIFIED.
;
;               R0 = SS$_DUPLNAM - ATTEMPT MADE TO SUPERSEDE NON-ALIASABLE
;                       LOGICAL NAME TABLE ENTRY.
;
;               R0 = SS$_EXLNMQUOTA - INSUFFICIENT QUOTA AVAILABLE IN THE QUOTA
;                       TABLE FOR THE CREATION OF THE NEW LOGICAL NAME TABLE
;                       ENTRY.
;
;               R0 = SS$_INSFMEM - SUFFICIENT SYSTEM DYNAMIC MEMORY DOES NOT
;                       EXIST TO ALLOCATE LOGICAL NAME TABLE ENTRY AND
;                       IMPLICIT RESOURCE WAIT IS NOT ENABLED.
;
;               R0 = SS$_IVLOGNAM - ZERO OR GREATER THAN MAXIMUM LENGTH
;                       LOGICAL NAME STRING, TABLE NAME STRING, OR EQUIVALENCE
;                       STRING SPECIFIED, OR THE LOGICAL NAME IS TO BE CONTAINED
;                       WITHIN A DIRECTORY TABLE AND IS EITHER GREATER THAN 31
;                       CHARACTERS IN SIZE OR HAS INVALID CHARACTERS FOR SUCH
;                       NAMES.
;
;               R0 = SS$_IVLOGTAB - INVALID LOGICAL NAME TABLE NAME SPECIFIED.
;
;               R0 = SS$_NOLOGTAB - LOGICAL NAME TABLE SPECIFIED DOES NOT EXIST.
;
;               R0 = SS$_TOOMANYLNM - TOO MANY LEVELS OF RECURSION IN SEARCH
;                       FOR LOGICAL NAME TABLE.
;
```

```
                                ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
                                ;
                                ;               R0 = SS$_BUFFEROVF - REQUEST SUCCESSFULLY COMPLETED. AN ITEM
                                ;                    BUFFER IS NOT LARGE ENOUGH TO HOLD REQUESTED DATA.
                                ;
                                ;               R0 = SS$_NORMAL - NORMAL COMPLETION, NEW ENTRY ENTERED IN
                                ;                    SPECIFIED LOGICAL NAME TABLE.
                                ;
                                ;               R0 = SS$_SUPERSEDE - NORMAL COMPLETION, NEW ENTRY SUPERSEDED
                                ;                    PREVIOUS ENTRY IN SPECIFIED LOGICAL NAME TABLE.
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       THIS ROUTINE EXITS AT IPL 2, AND MUST CONTINUE TO DO SO, BECAUSE
                                ;       IT IS CALLED AT SYSTEM INITIALIZATION TIME.
                                ;
                                ;-
                                
                                        .PSECT  Y$EXEPAGED
                            0FFC        .ENTRY  EXE$CRELNM,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                    031F'   31          BRW     EXECRELNM
                                
                                        .PSECT  YF$$LNM
                                        .ENABLE LSB
                                
                                ;
                                ; ERROR MESSAGES AND RETURNS
                                ;
                                
                    50  0C  3C  9000$:  MOVZWL  #SS$_ACCVIO,R0          ;ACCESS VIOLATION
                            04          RET
                    50  14  3C  9010$:  MOVZWL  #SS$_BADPARAM,R0        ;BAD SYSTEM SERVICE PARAMETER
                            04          RET
                50  0124 8F  3C  9020$:  MOVZWL  #SS$_INSFMEM,R0         ;INSUFFICIENT MEMORY
                            04  9030$:  RET
                                
                                EXECRELNM:
                                
                                ;
                                ; VALIDATE AND COPY PARAMETERS AS NECESSARY. REGISTER ASSIGNMENT FOR THE
                                ; PARAMETERS ARE AS FOLLOWS:
                                ;
                                ;       R5  = ACCESS MODE.
                                ;       R7  = ATTRIBUTE BITS.
                                ;       R9  = ADDRESS OF PROBED AND COPIED TABLE NAME DESCRIPTOR.
                                ;       R10 = ADDRESS OF PROBED AND COPIED LOGICAL NAME DESCRIPTOR.
                                ;       R11 = ADDRESS OF START OF ITEM LIST.
                                ;
                                ;---
                                
                  50  10 AC    D0          MOVL    CNACMODE(AP),R0         ;GET SPECIFIED ACCESS MODE
                         14     13          BEQL    50$                     ;BRANCH IF NOT PRESENT
                                            IFNORD  #1,(R0),9000$           ;CHECK READ ACCESS TO BYTE
                      50 60     9A          MOVZBL  (R0),R0                 ;GET VALUE
                      50 03     D1          CMPL    #PSL$C_USER,R0          ;CHECK FOR VALID ACCESS MODE
                         E2     1F          BLSSU   9010$                   ;INVALID ACCESS MODE
```

```
                                033D  1320 ;
                                033D  1321 ; IF THE ACCESS MODE OF THE NEW LOGICAL NAME WAS EXPLICITELY SPECIFIED AND THE
                                033D  1322 ; CALLER HAS THE SYSNAM PRIVILEGE, THEN THE NEW LOGICAL NAME IS CREATED WITH
                                033D  1323 ; THE SPECIFIED ACCESS MODE. OTHERWISE, THE ACCESS MODE OF THE CALLER IS
                                033D  1324 ; MAXIMIZED WITH ANY EXPLICTELY SPECIFIED ACCESS MODE AND USED TO CREATE THE
                                033D  1325 ; NEW LOGICAL NAME.
                                033D  1326 ;
                                033D  1327
                                033D  1328          IFPRIV  SYSNAM,60$              ;PRIVILEGE
        00000000'GF   16        0343  1329 50$:     JSB     G^EXE$MAXACMODE         ;MAXIMIZED WITH MODE OF CALLER TO
             55   50    D0      0349  1330 60$:     MOVL    R0,R5                   ;DETERMINE ACCESS MODE OF NEW TABLE
                                034C  1331
             57   04 AC    D0    034C  1332          MOVL    CNATTR(AP),R7           ;TABLE ATTRIBUTES
                     12    13    0350  1333          BEQL    200$                    ;BRANCH IF NOT PRESENT
                                0352  1334          IFNORD  #4,(R7),9000$           ;CHECK READ ACCESS
                57   67    D0    0358  1335          MOVL    (R7),R7                 ;GET VALUE
     57 FFFFFFF8 8F    D3      035B  1336          BITL    #^C< -
                                0362  1337                  LNM$M_CONFINE! -        ;CONFINE TO PROCESS
                                0362  1338                  LNM$M_CRELOG! -         ;$CRELOG USED TO CREATE NAME
                                0362  1339                  LNM$M_NO_ALIAS -        ;DO NOT ALLOW ALIASES
                                0362  1340                  >,R7
                     BB   12    0362  1341          BNEQ    9010$                   ;INVALID TABLE ATTRIBUTES
                                0364  1342
             5A   0C AC    D0    0364  1343 200$:    MOVL    CNLOGNAM(AP),R10        ;ADDRESS OF LOGICAL NAME DESCRIPTOR
                     B5    13    0368  1344          BEQL    9010$                   ;ERROR IF NOT PRESENT
                                036A  1345          IFNORD  #8,(R10),9000$          ;CHECK ACCESS TO DESCRIPTOR
                50   6A    7E    0370  1346          MOVAQ   (R10),R0                ;ADDRESS OF DESCRIPTOR
                   FC8A'   30    0373  1347          BSBW    LNM$PROBER              ;PROBE LOGICAL NAME STRING
                  AF   50  E9    0376  1348          BLBC    R0,9030$                ;BRANCH IF CAN'T READ OR NOT PRESENT
                7E     51  7D    0379  1349          MOVQ    R1,-(SP)                ;SAVE LOGICAL NAME DESCRIPTOR
                5A     5E  D0    037C  1350          MOVL    SP,R10                  ;ADDRESS OF LOGICAL NAME DESCRIPTOR
                                037F  1351
             59   08 AC    D0    037F  1352          MOVL    CNTABNAM(AP),R9         ;ADDRESS OF TABLE NAME DESCRIPTOR
                     9A    13    0383  1353          BEQL    9010$                   ;ERROR IF NOT PRESENT
                                0385  1354          IFNORD  #8,(R9),9000$           ;CHECK ACCESS TO DESCRIPTOR
                50   69    7E    038B  1355          MOVAQ   (R9),R0                 ;SET UP CALL
                   FC6F'   30    038E  1356          BSBW    LNM$PROBER              ;PROBE TABLE NAME STRING
                  94   50  E9    0391  1357          BLBC    R0,9030$                ;BRANCH IF CAN'T READ OR NOT PRESENT
                7E     51  7D    0394  1358          MOVQ    R1,-(SP)                ;SAVE TABLE NAME DESCRIPTOR
                59     5E  D0    0397  1359          MOVL    SP,R9                   ;ADDRESS OF TABLE NAME DESCRIPTOR
                                039A  1360
                                039A  1361 ;
                                039A  1362 ; PROBE AND VERIFY THE ITEM LIST. THIS IS THE FIRST OF TWO PASSES DOWN
                                039A  1363 ; THE ITEM LIST.  THIS FIRST PASS IS BEING DONE TO ACCUMULATE
                                039A  1364 ; INFORMATION AS TO THE NUMBER OF TRANSLATION STRINGS BEING DEFINED AND
                                039A  1365 ; THEIR COLLECTIVE SIZE FOR USE IN DETERMINING THE AMOUNT OF SPACE TO
                                039A  1366 ; ALLOCATE FOR THE NEW LOGICAL NAME TABLE ENTRY.  VALID ITEM TYPES FOR
                                039A  1367 ; THIS SYSTEM SERVICE ARE:
                                039A  1368 ;
                                039A  1369 ;       ATTRIBUTES  - ATTRIBUTES FOR NEXT TRANSLATION STRING
                                039A  1370 ;       INDEX       - INDEX TO BE ASSIGNED TO NEXT TRANSLATION STRING
                                039A  1371 ;       STRING      - EQUIVALENCE STRING
                                039A  1372 ;       TABLE       - LOGICAL NAME TABLE NAME   (OUTPUT ITEM)
                                039A  1373 ;       LNMBADDR    - LOGICAL NAME LNMB ADDRESS (OUTPUT ITEM)
                                039A  1374 ;
                                039A  1375
                     7E   7C    039A  1376          CLRQ    -(SP)                   ;CLEAR COUNTERS
```

```
                5B   14 AC    D0  039C  1377          MOVL    CNITMLST(AP),R11        ;ADDRESS OF ITEM LIST
                       5D    13  03A0  1378          BEQL    2010$                   ;BRANCH IF NONE
                       50    D4  03A2  1379          CLRL    R0                      ;CHAINED ITEM LIST COUNTER
                   58  5B    D0  03A4  1380          MOVL    R11,R8                  ;USER R8 TO VERIFY ITEM LIST
                                03A7  1381 1005$:    IFNORD  #4,(R8),1021$           ;CHECK IF FIRST LONGWORD READABLE
                                03AD  1382 1010$:
                52   02 A8    3C  03AD  1383          MOVZWL  2(R8),R2                ;GET ITEM TYPE
                       4C    13  03B1  1384          BEQL    2010$                   ;DONE IF ITEM TYPE IS ZERO
                                03B3  1385          IFNORD  #12,4(R8),1021$         ;CHECK REST OF THIS DESCRIPTOR
                                03BA  1386                                          ;PLUS FIRST LONGWORD OF NEXT ONE
                                03BA  1387
                                03BA  1388          CASE    R2,<-                   ;HANDLE EACH ITEM TYPE SEPARATELY
                                03BA  1389                      1040$, -            ;INDEX ITEM
                                03BA  1390                      1030$, -            ;STRING ITEM
                                03BA  1391                      1040$, -            ;ATTRIBUTES ITEM
                                03BA  1392                      1040$, -            ;TABLE ITEM
                                03BA  1393                      >,W,#1
                   52   09    B1  03C6  1394          CMPW    #LNM$_LNMB_ADDR, R2     ;TEST FOR LNMB_ADDR
                         1D    13  03C9  1395          BEQL    1040$
                52   FFFF 8F    B1  03CB  1396          CMPW    #LNM$_CHAIN,R2          ;TEST FOR CHAINED LIST
                         1C    13  03D0  1397          BEQL    1050$
                      FF4A     31  03D2  1398 1015$:    BRW     9010$                   ;ILLEGAL ITEM TYPE
                               04  03D5  1399 1020$:    RET                             ;RETURN ANY ERRORS
                                03D6  1400
                      FF42     31  03D6  1401 1021$:    BRW     9000$                   ;ACCESS VIOLATION
                                03D9  1402
                                03D9  1403 ;
                                03D9  1404 ; STRING ITEM.
                                03D9  1405 ;
                                03D9  1406 ; VALIDATE THE ITEM.  IF THE STRING SPECIFIED IS A VALID EQUIVALENCE
                                03D9  1407 ; STRING THEN INCREMENT THE TRANSLATION STRING COUNTER, AND ADD THE
                                03D9  1408 ; SIZE OF THIS STRING TO THE TOTAL STRING SIZE BEING ACCUMULATED.
                                03D9  1409 ;
                                03D9  1410
                   50   68    7E  03D9  1411 1030$:    MOVAQ   (R8),R0                 ;ADDRESS OF STRING ITEM DESCRIPTOR
                      FC21'     30  03DC  1412          BSBW    LNM$PROBER              ;PROBE TRANSLATION STRING
                        F3 50   E9  03DF  1413          BLBC    R0,1020$                ;BRANCH IF CAN'T READ OR NOT PRESENT
                         6E     D6  03E2  1414          INCL    (SP)                    ;INCREMENT EQUIVALENCE STRING COUNTER
                04 AE    51     C0  03E4  1415          ADDL2   R1,4(SP)                ;ADD SIZE OF STRING TO TOTAL SIZE
                                03E8  1416
                                03E8  1417 ;
                                03E8  1418 ; POSITION TO THE NEXT ITEM IN THE ITEM LIST.
                                03E8  1419 ;
                                03E8  1420
                   58   0C    C0  03E8  1421 1040$:    ADDL2   #12,R8                  ;POSITION TO NEXT ITEM
                      FFBF     31  03EB  1422          BRW     1010$                   ;AND CONTINUE ITEM LIST VERIFICATION
                                03EE  1423
                                03EE  1424 ;
                                03EE  1425 ; POSITION TO THE NEXT CHAIN OF THE CHAINED ITEM LIST.
                                03EE  1426 ;
                                03EE  1427
                         50     D6  03EE  1428 1050$:    INCL    R0                      ;COUNT ONE MORE
                0400 8F  50     B1  03F0  1429          CMPW    R0,#1024
                         DB     14  03F5  1430          BGTR    1015$                   ;TOO MANY - ASSUME A LOOP
                   58   04 A8   D0  03F7  1431          MOVL    4(R8),R8                ;GET POINTER TO NEXT LIST
                         02     13  03FB  1432          BEQL    2010$                   ;LAST ITEM
                         A8     11  03FD  1433          BRB     1005$                   ;PROCESS THIS ONE
```

```
                        03FF  1434
                        03FF  1435 ;
                        03FF  1436 ; RAISE  IPL  TO AST DELIVERY LEVEL SO THAT THERE ARE NO
                        03FF  1437 ; INTERRUPTIONS WHILE THE NEW LOGICAL NAME ENTRY IS BEING CREATED.
                        03FF  1438 ;
                        03FF  1439
                        03FF  1440 2010$:  SETIPL  S^#IPL$_ASTDEL          ;RAISE TO AST DELIVERY LEVEL
                        0402  1441
                        0402  1442 ;
                        0402  1443 ; ALLOCATE A LOGICAL NAME BLOCK FOR THE NEW LOGICAL NAME TABLE ENTRY.
                        0402  1444 ; ASSUME THAT THE NEW LOGICAL NAME TABLE WILL BE SHAREABLE, SO ALLOCATE
                        0402  1445 ; THE SPACE FROM SYSTEM PAGED POOL.  UNTIL THE LOGICAL NAME TABLE IN
                        0402  1446 ; WHICH THE NEW ENTRY WILL BE LOCATED IS FOUND, WHETHER THE NEW
                        0402  1447 ; LOGICAL NAME TABLE ENTRY WILL BE IN SYSTEM POOL AND SHAREABLE OR IN
                        0402  1448 ; P1 SPACE AND NOT SHAREABLE IS NOT KNOWN.  HOWEVER, THE LOGICAL NAME
                        0402  1449 ; MUTEX MUST BE LOCKED BEFORE A SEARCH CAN BE MADE FOR THE LOGICAL
                        0402  1450 ; NAME TABLE, AND ONCE THE MUTEX IS LOCKED, SYSTEM POOL CAN NOT
                        0402  1451 ; ALTHOUGH P1 POOL CAN BE ALLOCATED.  THEREFORE, THE NEW LOGICAL NAME
                        0402  1452 ; TABLE ENTRY IS ASSUMED TO BE SHAREABLE, AND IF THE LOGICAL NAME
                        0402  1453 ; TABLE WHICH TO TO CONTAIN THE NEW ENTRY IS FOUND TO RESIDE IN
                        0402  1454 ; SYSTEM POOL, THEN THE SYSTEM POOL ALLOCATED FOR THE NEW ENTRY IS
                        0402  1455 ; DEALLOCATED, AND PROCESS-PRIVATE POOL IS ALLOCATED IN ITS PLACE.
                        0402  1456 ;
                        0402  1457
      51   6E   04   C5  0402  1458          MULL3   #LNMX$T_XLATION,(SP),R1 ;LNMXS OVERHEAD +
           51   8E   C0  0406  1459          ADDL2   (SP)+,R1                ;TRANSLATION STRING COUNT FIELDS +
           51   8E   C0  0409  1460          ADDL2   (SP)+,R1                ;TRANSLATION STRINGS SIZES +
           51   6A   C0  040C  1461          ADDL2   (R10),R1                ;LOGICAL NAME SIZE +
                     C0  040F  1462          ADDL2   #<-                     ;LOGICAL NAME COUNT FIELD +
                         0410  1463                  LNMB$T_NAME+1+ -        ;LNMB OVERHEAD +
                         0410  1464                  1 -                     ;TRAILER BYTE =
           51   13         0410  1465                  >,R1                    ;SIZE OF SYSTEM POOL TO ALLOCATE
                         0412  1466
 00000000'GF    16   0412  1467          JSB     G^EXE$ALOPAGED              ;ALLOCATE THE LOGICAL NAME TABLE ENTRY
       03 50    E8   0418  1468          BLBS    R0,2040$                    ;BRANCH IF SUFFICIENT MEMORY
         FF05   31   041B  1469          BRW     9020$                       ;OTHERWISE RETURN ERROR
 08 A2   51    B0   041E  1470 2040$:   MOVW    R1,LNMB$W_SIZE(R2)          ;SAVE NEW BLOCK SIZE WITHIN THE BLOCK
                     0422  1471
                     0422  1472 ;
                     0422  1473 ; WRITE LOCK THE LOGICAL NAME MUTEX AND THEN POSITION TO THE LOGICAL NAME
                     0422  1474 ; TABLE THAT IS TO CONTAIN THE NEW LOGICAL NAME TABLE ENTRY. DEALLOCATE THE NEW
                     0422  1475 ; LNMB  IF UNABLE TO POSITION TO ANY TABLE ENTRY.
                     0422  1476 ;
                     0422  1477
          14   BB   0422  1478          PUSHR   #^M<R2,R4>                  ;SAVE ADDRESS OF PCB AND NEW LNMB
     FBD9'   30   0424  1479          BSBW    LNM$LOCKW                   ;LOCK TABLES FOR WRITING
   51   55   D0   0427  1480          MOVL    R5,R1                       ;ACCESS MODE
   52   69   7D   042A  1481          MOVQ    (R9),R2                     ;TABLE NAME DESCRIPTOR
     FBD0'   30   042D  1482          BSBW    LNM$FIRSTTAB                ;SEARCH FOR SPECIFIED TABLE
        54 8ED0   0430  1483          POPL    R4                          ;RESTORE ADDRESS OF NEW LNMB
        50   DD   0433  1484          PUSHL   R0                          ;SAVE RETURN STATUS
       03 50   E8   0435  1485          BLBS    R0,2050$                    ;BRANCH ON FAILURE
        021B   31   0438  1486          BRW     4020$
                     043B  1487
                     043B  1488 ;
                     043B  1489 ; IF THE NEW LOGICAL NAME TABLE ENTRY WILL BE SHAREABLE BECAUSE THE LOGICAL
                     043B  1490 ; NAME TABLE WHICH WILL CONTAIN THE NEW ENTRY HAS BEEN FOUND TO BE, THEN MAKE
```

```
                                043B  1491 ; SURE THE NEW LOGICAL NAME TABLE ENTRY WILL NOT BE MARKED AS CONFINED AND
                                043B  1492 ; MAKE SURE THE CALLER HAS WRITE ACCESS TO THE CONTAINING TABLE.
                                043B  1493
                      00    E1  043B  1494 2050$:  BBC     #LNMTH$V_SHAREABLE,-     ;BRANCH IF THE CONTAINING TABLE ENTRY
                    1B 61         043D  1495             LNMTH$B_FLAGS(R1),2070$ ;IS IN PROCESS-PRIVATE SPACE
                57    02    CA  043F  1496             BICL2   #LNM$M_CONFINE,R7       ;SHAREABLE NAMES CAN'T BE CONFINED
                                0442  1497
                52    02    9A  0442  1498             MOVZBL  #WRITE_ACCESS,R2        ;CODE FOR ACCESS CHECK
                      54    DD  0445  1499             PUSHL   R4                      ;SAVE LNMB PTR
            54    08 AE    D0  0447  1500             MOVL    8(SP),R4                ;RESTORE PCB
                    FBB2'   30  044B  1501             BSBW    LNM$CHECK_PROT          ;CHECK THE PROTECTION
                54    8E    D0  044E  1502             MOVL    (SP)+,R4                ;RESTORE LNMB
                  41 50     E8  0451  1503             BLBS    R0,2090$                ;OK
                6E    50    D0  0454  1504             MOVL    R0,(SP)                 ;SAVE STATUS
                    01FC    31  0457  1505             BRW     4020$                   ;AND EXIT
                                045A  1506
                                045A  1507 ;
                                045A  1508 ; IF THE LOGICAL NAME TABLE THAT IS TO CONTAIN THE NEW LOGICAL NAME TABLE ENTRY
                                045A  1509 ; IS DISCOVERED TO EXIST IN PROCESS-PRIVATE SPACE, THEN THIS IS ALSO WHERE THE
                                045A  1510 ; NEW LOGICAL NAME ENTRY SHOULD GO.  DEALLOCATE THE SYSTEM SPACE LNMB, AND
                                045A  1511 ; ALLOCATE THE SAME SIZE BLOCK FROM P1 SPACE.
                                045A  1512 ;
                                045A  1513
                      51    DD  045A  1514 2070$:  PUSHL   R1                      ;SAVE ADDRES OF CONTAINING TABLE HEADER
            7E    08 A4    3C  045C  1515             MOVZWL  LNMB$W_SIZE(R4),-(SP)   ;SAVE SIZE OF LNMB
                50    54    D0  0460  1516             MOVL    R4,R0                   ;ADDRESS OF SYSTEM SPACE LNMB
                    FB9A'   30  0463  1517             BSBW    LNM$DELBLK              ;DEALLOCATE SYSTEM SPACE LNMB
                                0466  1518
                      51 8ED0   0466  1519             POPL    R1                      ;SIZE OF STORAGE TO ALLOCATE
                00000000'GF     16  0469  1520             JSB     G^EXE$ALOP1PROC         ;ALLOCATE NEW LNMB FROM P1 SPACE
                        0C 50   E8  046F  1521             BLBS    R0,2080$                ;BRANCH IF SUCCESSFUL
            04 AE    0124 8F    3C  0472  1522             MOVZWL  #SS$_INSFMEM,4(SP)      ;OTHERWISE RETURN INSUFFICIENT MEMORY
                    5E    04    C0  0478  1523             ADDL2   #4,SP                   ;THROW AWAY TABLE HEADER ADDRESS
                        01DE    31  047B  1524             BRW     4030$                   ;RETURN ERROR AFTER RELEASING MUTEX
                                047E  1525
                      54    52    D0  047E  1526 2080$:  MOVL    R2,R4                   ;ADDRESS OF NEW LNMB
                08 A4    51    B0  0481  1527             MOVW    R1,LNMB$W_SIZE(R4)      ;SAVE NEW BLOCKS SIZE WITHIN BLOCK
                      51 8ED0   0485  1528             POPL    R1                      ;ADDRESS OF CONTAINING TABLE HEADER
                52    09 A1    D0  0488  1529             MOVL    LNMTH$L_NAME(R1),R2     ;ADDRESS OF TABLE'S LNMB
                      01    E1  048C  1530             BBC     #LNMB$V_CONFINE,-       ;IS THE ENTRY MARKED CONFINE?
                  04 10 A2        048E  1531                     LNMB$B_FLAGS(R2),2090$  ;BRANCH IF NOT
                00 57    01    E3  0491  1532             BBCS    #LNM$V_CONFINE,R7,2090$ ;MAKE SURE NEW ENTRY IS IF SO
                                0495  1533
                                0495  1534 ;
                                0495  1535 ; MAKE SURE THAT THE QUOTA HOLDER OF THE LOGICAL NAME TABLE WHICH IS TO CONTAIN
                                0495  1536 ; THE NEW LOGICAL NAME TABLE ENTRY HAS SUFFICIENT QUOTA FOR THE CREATION OF
                                0495  1537 ; THIS NEW LOGICAL NAME TABLE ENTRY.
                                0495  1538 ;
                                0495  1539
                52    19 A1    D0  0495  1540 2090$:  MOVL    LNMTH$L_QTABLE(R1),R2   ;RETRIEVE QUOTA HOLDER'S ADDRESS
                50    08 A4    3C  0499  1541             MOVZWL  LNMB$W_SIZE(R4),R0      ;SIZE OF QUOTA TO BE WITHDRAWN
                                049D  1542
                21 A2    50    D1  049D  1543             CMPL    R0,LNMTH$L_BYTES(R2)    ;IS THEIR SUFFICIENT QUOTA?
                        08    1B  04A1  1544             BLEQU   3000$                   ;YES - THEN CONTINUE
            6E    224C 8F    3C  04A3  1545             MOVZWL  #SS$_EXLNMQUOTA,(SP)    ;NO  - THEN DEALLOCATE LNMB AND RETURN
                    01AB    31  04A8  1546             BRW     4020$                   ;       EXCEEDED QUOTA ERROR
                                04AB  1547
```

```
                                04AB  1548 ;
                                04AB  1549 ; BUILD THE LOGICAL NAME TABLE ENTRY. REGISTER USAGE IS AS FOLLOWS:
                                04AB  1550 ;
                                04AB  1551 ;       R1  = ADDRESS OF THE LOGICAL NAME TABLE'S TABLE HEADER.
                                04AB  1552 ;       R4  = ADDRESS OF LOGICAL NAME BLOCK FOR NEW LOGICAL NAME TABLE ENTRY.
                                04AB  1553 ;       R5  = ACCESS MODE.
                                04AB  1554 ;       R7  = ATTRIBUTE BITS.
                                04AB  1555 ;       R9  = ADDRESS OF PROBED AND COPIED TABLE NAME DESCRIPTOR.
                                04AB  1556 ;       R10 = ADDRESS OF PROBED AND COPIED LOGICAL NAME DESCRIPTOR.
                                04AB  1557 ;       R11 = ADDRESS OF ITEM LIST.
                                04AB  1558 ;
                                04AB  1559
        50    0A A4    9E      04AB  1560 3000$:  MOVAB   LNMB$B_TYPE(R4),R0      ;POSITION TO BLOCK TYPE
        80    40 8F    90      04AF  1561         MOVB    #DYN$C_LNM,(R0)+        ;SET DATA STRUCTURE TYPE
           80    55    90      04B3  1562         MOVB    R5,(R0)+                ;SET OWNER ACCESS MODE
           80    51    D0      04B6  1563         MOVL    R1,(R0)+                ;ADDRESS OF CONTAINING TABLE HEADER
           80    57    90      04B9  1564         MOVB    R7,(R0)+                ;STORE NAME ATTRIBUTES
           80    6A    90      04BC  1565         MOVB    (R10),(R0)+             ;STORE LENGTH OF NAME
                 54    DD      04BF  1566         PUSHL   R4                      ;SAVE LNMB ADDRESS ON STACK
                 01    E0      04C1  1567         BBS     #LNMTH$V_DIRECTORY,-    ;BRANCH IF THIS LOGICAL NAME TO BE
                    07 61        04C3  1568                 LNMTH$B_FLAGS(R1),3005$ ;CONTAINED WITHIN A DIRECTORY HEADER
60      04 BA    6A    28      04C5  1569         MOVC3   (R10),@4(R10),(R0)      ;MOVE LOGICAL NAME INTO LNMB NAME FIELD
                 21    11      04CA  1570         BRB     3010$                   ;JOIN COMMON CODE
                               04CC  1571
           1F    6A    B1      04CC  1572 3005$:  CMPW    (R10),#LNM$C_TABNAMLEN  ;SIZE EXCEED MAX FOR NAME IN DIRECTORY?
                 0B    1B      04CF  1573         BLEQU   3009$                   ;CONTINUE IF IT DOESNT
                 54 8ED0       04D1  1574 3007$:  POPL    R4                      ;OTHERWISE RESTORE LNMB ADDRESS
     6E    0154 8F    3C      04D4  1575         MOVZWL  #SS$_IVLOGNAM,(SP)      ;SET ERROR CODE
              017A    31      04D9  1576         BRW     4020$                   ;GO RETURN ERROR
                               04DC  1577
00      04 BA    6A    2F      04DC  1578 3009$:  MOVTUC  (R10),@4(R10),#0,-      ;CHECK FOR PROPER LOGICAL NAME SYNTAX
        00000000'EF            04E1  1579                 EXE$LNM_SYNTAX_DAT,-     ;AND AT THE SAME TIME MOVE THE LOGICAL
           60    6A            04E6  1580                 (R10),(R0)              ;NAME INTO THE LNMB NAME FIELD
                 E7    1D      04E8  1581         BVS     3007$                   ;GO RETURN ERROR IF IMPROPER SYNTAX
           53    55    D0      04EA  1582         MOVL    R5,R3                   ;ELSE SHIFT LNMX ADDR INTO PROPER REG
                 54 8ED0       04ED  1583 3010$:  POPL    R4                      ;RESTORE LNMB ADDRESS FROM STACK
                               04F0  1584
                               04F0  1585 ;
                               04F0  1586 ; FILL IN THE LNMX PORTION OF THE LOGICAL NAME BLOCK FOR THE NEW LOGICAL NAME
                               04F0  1587 ; TABLE ENTRY.  THIS INVOLVES PROCESSING THE ITEM LIST FOR A SECOND TIME SO
                               04F0  1588 ; THAT THE LOGICAL NAME TRANSLATION BLOCKS MAYBE CONSTRUCTED, AND ANY OUTPUT
                               04F0  1589 ; INFORMATION RETURNED.
                               04F0  1590 ;
                               04F0  1591
              5B    D5      04F0  1592         TSTL    R11                     ;WAS AN ITEM LIST DEFINED?
              03    12      04F2  1593         BNEQ    3030$                   ;YES - GO PROCESS IT
           013C    31      04F4  1594 3020$:  BRW     3950$                   ;NO  - SKIP ITEM LIST PROCESSING
                               04F7  1595
              56    D4      04F7  1596 3030$:  CLRL    R6                      ;0 IS THE DEFAULT INDEX
              58    D4      04F9  1597         CLRL    R8                      ;NO ATTRIBUTES DEFINED AS YET
                               04FB  1598 3035$:  IFNORD  #4,(R11),3060$          ;CHECK IF FIRST LONGWORD READABLE
                               0501  1599 3040$:
52      02 AB    3C      0501  1600         MOVZWL  2(R11),R2               ;GET ITEM TYPE
              ED    13      0505  1601         BEQL    3020$                   ;DONE IF ITEM TYPE IS ZERO
                               0507  1602         IFNORD  #12,4(R11),3060$                ;CHECK REST OF THIS DESCRIPTOR
                               050E  1603                                         ;PLUS FIRST LONGWORD OF NEXT ONE
                               050E  1604
```

```
                     050E  1605          CASE    R2,<-                           ;HANDLE EACH ITEM TYPE SEPARATELY
                     050E  1606                        3200$, -                  ;INDEX ITEM
                     050E  1607                        3300$, -                  ;STRING ITEM
                     050E  1608                        3200$, -                  ;ATTRIBUTES ITEM
                     050E  1609                        3400$, -                  ;TABLE ITEM
                     050E  1610                        >,W,#1
     52   09    B1   051A  1611          CMPW    #LNM$_LNMB_ADDR, R2             ;TEST FOR SPECIAL ITEM
          15    13   051D  1612          BEQL    3100$
52   FFFF 8F    B1   051F  1613          CMPW    #LNM$_CHAIN,R2                  ;TEST FOR CHAINED ITEM LIST
          06    13   0524  1614          BEQL    3070$
        0100    31   0526  1615 3050$:   BRW     3930$                           ;ILLEGAL ITEM
        00F8    31   0529  1616 3060$:   BRW     3920$                           ;REPORT ACCESS VIOLATION
                     052C  1617
                     052C  1618 ;
                     052C  1619 ; POSITION TO THE NEXT CHAIN IN THE CHAINED ITEM LIST.
                     052C  1620 ;
                     052C  1621
5B   04 AB    D0   052C  1622 3070$:   MOVL    4(R11),R11                      ;GET POINTER
          C2    13   0530  1623          BEQL    3020$                           ;LAST ITEM
          C7    11   0532  1624          BRB     3035$                           ;GO PROCESS IT
                     0534  1625
                     0534  1626 ;
                     0534  1627 ; LOGICAL NAME BLOCK ADDRESS
                     0534  1628 ;
                     0534  1629 ; TEST IF THE LOGICAL NAME BLOCK ADDRESS CAN BE WRITTEN IN THE
                     0534  1630 ; REQUESTED PLACE.  ( CURRENT USE IS TO LINK A LOGICAL NAME BLOCK
                     0534  1631 ; WITH MTL OR MAILBOX UCB )
                     0534  1632 ;
                     0534  1633
     6B   04    B1   0534  1634 3100$:   CMPW    #4,(R11)                        ;IS THE ITEM A LONGWORD
          ED    12   0537  1635          BNEQ    3050$                           ;ERROR IF NOT
     50   04 AB    D0   0539  1636          MOVL    4(R11),R0                       ;RETRIEVE ITEM BUFFER ADDRESS
                     053D  1637          IFNOWRT #4,(R0),3060$                   ;ACCVIO IF ITEM BUFFER NOT WRITEABLE
     60   54    D0   0543  1638          MOVL    R4,(R0)                         ;MOVE THE ADDRESS OF LNM TO BUFFER
        00D5    31   0546  1639          BRW     3910$                           ;GO POSITION TO NEXT ITEM
                     0549  1640
                     0549  1641 ;
                     0549  1642 ; INDEX OR ATTRIBUTES ITEM.
                     0549  1643 ;
                     0549  1644 ; VALIDATE THE ITEM.  SAVE THE INDEX OR ATTRIBUTES FOR APPLICATION TO
                     0549  1645 ; THE NEXT  TRANSLATION BLOCK THAT IS TO BE CREATED.
                     0549  1646 ;
                     0549  1647
     6B   04    B1   0549  1648 3200$:   CMPW    #4,(R11)                        ;IS ITEM BUFFER AT LEAST A LONGWORD?
          D8    1A   054C  1649          BGTRU   3050$                           ;ERROR IF NOT
     50   04 AB    D0   054E  1650          MOVL    4(R11),R0                       ;RETRIEVE ITEM BUFFER ADDRESS
                     0552  1651          IFNORD  (R11),(R0),3060$                ;ACCVIO IF ITEM BUFFER NOT READABLE
     50   60    D0   0558  1652          MOVL    (R0),R0                         ;PICK UP THE LONGWORD
                     055B  1653
     52   01    B1   055B  1654          CMPW    #LNM$_INDEX,R2                  ;IS IT AN INDEX ITEM?
          11    13   055E  1655          BEQL    3210$                           ;YES - BRANCH AND VERIFY INDEX ITEM
50   FFFFFCFF 8F    D3   0560  1656          BITL    #^C< -                          ;TEST FOR INVALID ATTRIBUTES
                     0567  1657                  LNM$M_CONCEALED! -              ;TRANSLATION IS CONCEALED
                     0567  1658                  LNM$M_TERMINAL -                ;TRANSLATION IS TERMINAL
                     0567  1659                  >,R0
          BD    12   0567  1660          BNEQ    3050$                           ;INVALID ATTRIBUTES ITEM
58   50   F8 8F    9C   0569  1661          ROTL    #-8,R0,R8                       ;SAVE ATTRIBUTES FOR LNMX CREATION
```

```
                00AD   31  056E  1662          BRW     3910$                   ;GO POSITION TO NEXT ITEM
                       0571  1663
      50 FFFFFF80 8F   D3  0571  1664 3210$:   BITL    #^C127,R0               ;IS INDEX BETWEEN 0 AND 127?
                  16   13  0578  1665          BEQL    3220$                   ;YES - NO PRIVILEGE NEEDED
                       057A  1666
                       057A  1667 ;
                       057A  1668 ; THIS IS HERE SO THAT MTL OR MAILBOX UCB CAN HAVE ITS ADDRESS
                       057A  1669 ; STORED IN THE LOGICAL NAME BLOCK.
                       057A  1670 ;
                       057A  1671
                       057A  1672
                  52   DC  057A  1673          MOVPSL  R2                      ;GET THE PSL
      52 00C00000 8F   D3  057C  1674          BITL    #PSL$M_PRVMOD,R2        ;TEST IF PREVIOUS MODE WAS KERNEL
                  A1   12  0583  1675          BNEQ    3050$                   ;NO - BAD INDEX VALUE
      50 FFFFFF81 8F   D1  0585  1676          CMPL    #LNMX$C_BACKPTR,R0      ;CORRECT INDEX
                  98   12  058C  1677          BNEQ    3050$                   ;NO - BAD INDEX VALUE
                  05   11  058E  1678          BRB     3225$
                56   50   D1  0590  1679 3220$:   CMPL    R0,R6                   ;IS INDEX LESS THAN LAST ONE SEEN?
                  91   19  0593  1680          BLSS    3050$                   ;YES  - INVALID INDEX ITEM
                56   50   D0  0595  1681 3225$:   MOVL    R0,R6                   ;SAVE INDEX FOR LNMX CREATION
                   0083   31  0598  1682          BRW     3910$                   ;GO POSITION TO NEXT ITEM
                       059B  1683
                       059B  1684 ;
                       059B  1685 ; STRING ITEM.
                       059B  1686 ;
                       059B  1687 ; VALIDATE THE ITEM, INCLUDING THAT THE STRING SPECIFIED IS A VALID EQUIVALENCE
                       059B  1688 ; STRING.  VERIFY THAT THE CREATION OF THIS TRANSLATION BLOCK WILL NOT EXCEED
                       059B  1689 ; THE AMOUNT OF SPACE ALLOCATED.  FILL IN THE NEXT TRANSLATION BLOCK UTILIZING
                       059B  1690 ; THE  STRING SPECIFIED AS THE TRANSLATION STRING AS WELL AS THE CURRENT INDEX
                       059B  1691 ; AND  ATTRIBUTES SETTINGS. INCREMENT THE CURRENT INDEX.
                       059B  1692 ;
                       059B  1693
                50   6B   7E  059B  1694 3300$:   MOVAQ   (R11),R0               ;ADDRESS OF STRING ITEM DESCRIPTOR
                   FA5F'   30  059E  1695          BSBW    LNM$PROBER              ;PROBE TRANSLATION STRING
                  42 50    E9  05A1  1696          BLBC    R0,3410$                ;BRANCH IF CAN'T READ OR NOT PRESENT
                       05A4  1697
      50        53   54   C3  05A4  1698          SUBL3   R4,R3,R0                ;NUMBER OF BYTES CURRENTLY IN USE
                50   06   C0  05A8  1699          ADDL2   #LNMX$T_XLATION+2,R0    ;ADD OVERHEAD FOR CURRENT TRANSLATION
                50   51   C0  05AB  1700          ADDL2   R1,R0                   ;ADD IN SIZE OF CURRENT TRANSLATION
             08 A4   50   B1  05AE  1701          CMPW    R0,LNMB$W_SIZE(R4)      ;EXCEED LOGICAL NAME BLOCK SIZE?
                     75   1A  05B2  1702          BGTRU   3930$                   ;YES - RETURN AN FRROR
                       05B4  1703
                83   58   90  05B4  1704          MOVB    R8,(R3)+                ;CURRENT ATTRIBUTES BECOME LNMX FLAGS
                83   56   90  05B7  1705          MOVB    R6,(R3)+                ;CURRENT INDEX LEVEL BECOMES LNMX LEVEL
                     83   B4  05BA  1706          CLRW    (R3)+                   ;INITIALIZE HASH CODE LOCATION
                     56   D6  05BC  1707          INCL    R6                      ;INCREMENT INDEX LEVEL BY 1
                83   51   90  05BE  1708          MOVB    R1,(R3)+                ;STORE SIZE OF TRANSLATION STRING
                     54   DD  05C1  1709          PUSHL   R4                      ;SAVE NEEDED REGISTERS OVER MOVC3
      63        62   51   28  05C3  1710          MOVC3   R1,(R2),(R3)            ;MOVE TRANSLATION STRING INTO LNMX
                     54 8ED0  05C7  1711          POPL    R4                      ;RESTORE REGISTERS
                     52   11  05CA  1712          BRB     3910$                   ;GO POSITION TO NEXT ITEM
                       05CC  1713
                       05CC  1714 ;
                       05CC  1715 ; TABLE ITEM.
                       05CC  1716 ;
                       05CC  1717 ; VALIDATE THE ITEM.  STORE THE NAME OF THE LOGICAL NAME TABLE WHICH IS TO
                       05CC  1718 ; CONTAIN THIS NEW LOGICAL NAME TABLE ENTRY IN THE ITEM BUFFER PROVIDED, AND
```

```
                                05CC  1719 ; THE SIZE OF THE NAME STORED IN THE ITEM SIZE BUFFER, IF PROVIDED.
                                05CC  1720 ;
                                05CC  1721
      7E    6B    7D          05CC  1722 3400$:  MOVQ    (R11),-(SP)             ;SAVE DESCRIPTOR OF TABLE ITEM BUFFER
      51    6E    3C          05CF  1723         MOVZWL  (SP),R1                 ;SIZE OF TABLE ITEM BUFFER
      50    04    AE    D0    05D2  1724         MOVL    4(SP),R0                ;ADDRESS OF TABLE ITEM BUFFER
                  53    DD    05D6  1725         PUSHL   R3                      ;SAVE CURRENT LNMX POSITION
                  53    D4    05D8  1726         CLRL    R3                      ;ACCESS MODE
 00000000'GF      16          05DA  1727         JSB     G^EXE$PROBEW            ;PROBE TABLE ITEM BUFFER FOR WRITTING
                  53 8ED0     05E0  1728         POPL    R3                      ;RESTORE CURRENT LNMX POSITION
      51    8E    7D          05E3  1729         MOVQ    (SP)+,R1                ;RESTORE TABLE ITEM BUFFER DESCRIPTOR
        45  50    E9          05E6  1730 3410$:  BLBC    R0,3940$                ;RETURN IF NOT WRITEABLE
                              05E9  1731
      55    0C    A4    D0    05E9  1732         MOVL    LNMB$L_TABLE(R4),R5     ;ADDRESS OF CONTAINING TABLE HEADER
      55    09    A5    D0    05ED  1733         MOVL    LNMTH$L_NAME(R5),R5     ;ADDRESS OF CONTAINING LNMB
      55    11    A5    9E    05F1  1734         MOVAB   LNMB$T_NAME(R5),R5      ;ADDRESS OF COUNT FIELD
      50    85    9A          05F5  1735         MOVZBL  (R5)+,R0                ;SIZE OF CONTAINING TABLE'S NAME
                              05F8  1736
      51    50    B1          05F8  1737         CMPW    R0,R1                   ;ENOUGH ROOM IN BUFFER FOR NAME?
                  08    15    05FB  1738         BLEQ    3420$                   ;BRANCH IF SUFFICIENT ROOM
6E    0601  8F    3C          05FD  1739         MOVZWL  #SS$_BUFFEROVF,(SP)     ;CHANGE RETURN STATUS
      50    51    3C          0602  1740         MOVZWL  R1,R0                   ;RETURN AS MUCH AS POSSIBLE
                              0605  1741
      51    08    AB    D0    0605  1742 3420$:  MOVL    8(R11),R1               ;WAS A RETURN LENGTH BUFFER SPECIFIED
                  09    13    0609  1743         BEQL    3430$                   ;NO  - DON'T RETURN LENGTH
                              060B  1744         IFNOWRT #2,(R1),3920$           ;YES - PROBE RETURN LENGTH BUFFER
      61    50    B0          0611  1745         MOVW    R0,(R1)                 ;      AND RETURN LENGTH
                              0614  1746
      7E    53    7D          0614  1747 3430$:  MOVQ    R3,-(SP)                ;SAVE REGISTERS OVER MOVC3
62    65    50    28          0617  1748         MOVC3   R0,(R5),(R2)            ;RETURN CONTAINING TABLE NAME STRING
      53    8E    7D          061B  1749         MOVQ    (SP)+,R3                ;RESTORE SAVED REGISTERS
                              061E  1750
      5B    0C    C0          061E  1751 3910$:  ADDL2   #12,R11                 ;POSITION TO NEXT ITEM
          FEDD    31          0621  1752         BRW     3040$                   ;AND CONTINUE ITEM LIST VERIFICATION
                              0624  1753
      6E    0C    3C          0624  1754 3920$:  MOVZWL  #SS$_ACCVIO,(SP)        ;ACCESS VIOLATION
            2D    11          0627  1755         BRB     4020$                   ;DEALLOCATE NEW LNMB AND RETURN ERROR
                              0629  1756
      6E    14    3C          0629  1757 3930$:  MOVZWL  #SS$_BADPARAM,(SP)      ;BAD PARAMETER SEEN
            28    11          062C  1758         BRB     4020$                   ;DEALLOCATE NEW LNMB AND RETURN ERROR
                              062E  1759
      6E    50    D0          062E  1760 3940$:  MOVL    R0,(SP)                 ;SAVE RETURN STATUS
            23    11          0631  1761         BRB     4020$                   ;DEALLOCATE NEW LNMB AND RETURN ERROR
                              0633  1762
                              0633  1763 ;
                              0633  1764 ; FILL IN THE LAST LNMX PORTION OF THE LOGICAL NAME BLOCK FOR THE NEW LOGICAL
                              0633  1765 ; NAME TABLE ENTRY. THE LAST TRANSLATION BLOCK CONSISTS SOLELY OF A FLAGS FIELD,
                              0633  1766 ; AND IS MARKED WITHIN THIS FLAGS FIELD AS THE LAST LNMX.
                              0633  1767 ;
                              0633  1768
      83    04    90          0633  1769 3950$:  MOVB    #LNMX$M_XEND,(R3)+      ;STORE END FLAG
                              0636  1770
                              0636  1771 ;
                              0636  1772 ; AT THIS POINT ALL CHECKS HAVE BEEN MADE, AND THE TABLE STRUCTURES MAY NOW
                              0636  1773 ; BE MODIFIED FOR THE FIRST TIME.  INSERT THE NEW LOGICAL NAME TABLE ENTRY.
                              0636  1774 ;
                              0636  1775 ; THIS INSERTION MAY TAKE ONE OF TWO FORMS:
```

```
                     0636  1776 ;
                     0636  1777 ; 1.) THE LOGICAL NAME TABLE ENTRY MAYBE INSERTED AS A NEW LOGICAL NAME. THE
                     0636  1778 ;     FOLLOWING CONDITIONS MUST HOLD TRUE.
                     0636  1779 ;
                     0636  1780 ;       A.) THERE IS NO LOGICAL NAME TABLE ENTRY WITH THE SAME NAME AND ACCESS
                     0636  1781 ;           MODE WITHIN THE SAME CONTAINING LOGICAL NAME TABLE.
                     0636  1782 ;
                     0636  1783 ;       B.) THERE IS NO LOGICAL NAME TABLE ENTRY WITH THE SAME NAME AND AN
                     0636  1784 ;           INNER ACCESS MODE WITHIN THE SAME CONTAINING LOGICAL NAME TABLE
                     0636  1785 ;           THAT DOES NOT ALLOW ALIASES.
                     0636  1786 ;
                     0636  1787 ; 2.) THE ENTRY MAYBE INSERTED SUPERSEDING AN EXISTING ENTRY WITHIN THE SAME
                     0636  1788 ;     CONTAINING LOGICAL NAME TABLE WITH THE SAME NAME AND ACCESS MODE. THE
                     0636  1789 ;     OLD LOGICAL NAME TABLE ENTRY IS DELETED.
                     0636  1790 ;
                     0636  1791 ; IF EITHER CASE OCCURS, AND THE NEW LOGICAL NAME TABLE ENTRY DOES NOT ALLOW
                     0636  1792 ; ALIASES, THEN ANY LOGICAL NAME TABLE ENTRIES IN THE SAME CONTAINING LOGICAL
                     0636  1793 ; NAME TABLE WITH THE SAME NAME BUT AT AN OUTER ACCESS MODE ARE DELETED.
                     0636  1794 ;
                     0636  1795
      51    54   D0  0636  1796          MOVL    R4,R1                   ;ADDRES OF NEW LOGICAL NAME TABLE ENTRY
      52    57   D0  0639  1797          MOVL    R7,R2                   ;ATTRIBUTES
         F9C1'  30  063C  1798          BSBW    LNM$INSLOGTAB           ;INSERT NEW LOGICAL NAME TABLE ENTRY
50    0631  8F  B1  063F  1799          CMPW    #SS$_SUPERSEDE,R0       ;INSERT ION SUCCEED WITH SUPERSEDE?
            0A  12  0644  1800          BNEQ    4010$                   ;BRANCH IF NO
6E    0601  8F  B1  0646  1801          CMPW    #SS$_BUFFEROVF,(SP)     ;BUFFER OVERFLOW SEEN?
            0F  13  064B  1802          BEQL    4030$                   ;YES - THEN RETURN THAT STATUS
      6E    50  D0  064D  1803          MOVL    R0,(SP)                 ;OTHERWISE RETURN SUPERSEDE SUCCESS
         09 50  E8  0650  1804 4010$:   BLBS    R0,4030$                ;BRANCH ON SUCCESS
      6E    50  D0  0653  1805          MOVL    R0,(SP)                 ;OTHRWISE SAVE ERROR STATUS
                  0656  1806
      50    54  D0  0656  1807 4020$:   MOVL    R4,R0                   ;LOGICAL NAME BLOCK FOR NEW TABLE ENTRY
         F9A4'  30  0659  1808          BSBW    LNM$DELBLK              ;DELETE BLOCK CONTAINING NEW TABLE ENTRY
                  065C  1809
54    04 AE   D0  065C  1810 4030$:   MOVL    4(SP),R4                ;RETRIEVE PCB ADDRESS
         F99D'  30  0660  1811          BSBW    LNM$UNLOCK              ;UNLOCK TABLES
            50 8ED0  0663  1812          POPL    R0                      ;RESTORE STATUS
                04  0666  1813          RET                             ;RETURN TO CALLER
                  0667  1814
                  0667  1815          .DISABLE        LSB
                  0667  1816
                  0667  1817 ;        .PAGE
```

```
0667  1819          .SBTTL  EXE$DELLNM      - DELETE LOGICAL NAME
0667  1820 ;+
0667  1821 ; EXE$DELLNM - DELETE LOGICAL NAME
0667  1822 ;
0667  1823 ; THIS SERVICE PROVIDES THE CAPABILITY TO:
0667  1824 ;
0667  1825 ;       1. DELETE A SPECIFIED LOGICAL NAME.
0667  1826 ;       2. DELETE A SPECIFIED LOGICAL NAME TABLE.
0667  1827 ;       3. DELETE ALL THE LOGICAL NAME TABLE ENTRIES CONTAINED WITHIN A
0667  1828 ;          SPECIFIED LOGICAL NAME TABLE.
0667  1829 ;
0667  1830 ; INPUTS:
0667  1831 ;
0667  1832 ;       DNACMODE(AP)    = ADDRESS OF ACCESS MODE OF LOGICAL NAMES(S) TO BE DFLETED.
0667  1833 ;       DNLOGNAM(AP)    = ADDRESS OF LOGICAL NAME STRING DESCRIPTOR.
0667  1834 ;       DNTABNAM(AP)    = ADDRESS OF TABLE NAME STRING DFSCRIPTOR.
0667  1835 ;
0667  1836 ;       R4 = CURRENT PROCESS PCB ADDRESS.
0667  1837 ;
0667  1838 ; OUTPUTS:
0667  1839 ;
0667  1840 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO DELETE LOGICAL NAME(S).
0667  1841 ;
0667  1842 ;               R0 = SS$_ACCVIO - TABLE NAME DESCRIPTOR, TABLE NAME STRING
0667  1843 ;                       LOGICAL NAME DESCRIPTOR, LOGICAL NAME STRING, CANNOT
0667  1844 ;                       BE READ BY CALLING ACCESS MODE.
0667  1845 ;
0667  1846 ;               R0 = SS$_BADPARAM - INVALID ACCESS MODE SPECIFIED. NO TABLE
0667  1847 ;                       NAME DESCRIPTOR SPECIFIED.
0667  1848 ;
0667  1849 ;               R0 = SS$_IVLOGNAM - ZERO OR GREATER THAN MAXIMUM LENGTH
0667  1850 ;                       LOGICAL OR TABLE NAME STRING SPECIFIED.
0667  1851 ;
0667  1852 ;               R0 = SS$_IVLOGTAB - INVALID TABLE NAME SPECIFIED.
0667  1853 ;
0667  1854 ;               R0 = SS$_NOLOGNAM - LOGICAL NAME TABLE OR LOGICAL NAME
0667  1855 ;                       SPECIFIED DOES NOT EXIST.
0667  1856 ;
0667  1857 ;               R0 = SS$_NOLOGTAB - LOGICAL NAME TABLE SPECIFIED DOES NOT EXIST.
0667  1858 ;
0667  1859 ;               R0 = SS$_NOPRIV - PROCESS DOES NOT HAVE PRIVILEGE TO DELETE
0667  1860 ;                       THE SPECIFIED LOGICAL NAME(S).
0667  1861 ;
0667  1862 ;               R0 = SS$_TOOMANYLNM - TOO MANY LEVELS OF RECURSION IN SEARCH
0667  1863 ;                       FOR LOGICAL NAME TABLE OR LOGICAL NAME.
0667  1864 ;
0667  1865 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
0667  1866 ;
0667  1867 ;               R0 = SS$_NORMAL - NORMAL COMPLETION. LOGICAL NAME(S) HAVE BEEN
0667  1868 ;                       DELETED.
0667  1869 ;
0667  1870 ; SIDE EFFFCTS:
0667  1871 ;
0667  1872 ;       THIS ROUTINE EXITS AT IPL 2.
0667  1873 ;
0667  1874 ;-
0667  1875
```

```
                        0000000A  1876          .PSECT  Y$EXEPAGED
                    0FFC  000A  1877          .ENTRY  EXE$DELLNM,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                0660'   31  000C  1878          BRW     EXEDELLNM
                            000F  1879
                        00000667  1880          .PSECT  YF$$LNM
                            0667  1881          .ENABLE LSB
            50    0C    3C  0667  1882 900$:     MOVZWL  #SS$_ACCVIO,R0          ;ACCESS VIOLATION
                        04  066A  1883          RET
            50    14    3C  066B  1884 910$:     MOVZWL  #SS$_BADPARAM,R0        ;BAD SYSTEM SERVICE PARAMETER
                        04  066E  1885 930$:     RET
                            066F  1886
                            066F  1887 ;
                            066F  1888 ; VALIDATE AND COPY PARAMETERS AS NECESSARY. REGISTER ASSIGNMENT FOR THE
                            066F  1889 ; PARAMETERS ARE AS FOLLOWS:
                            066F  1890 ;
                            066F  1891 ;       R5  = ACCESS MODE.
                            066F  1892 ;       R9  = ADDRESS OF PROBED AND COPIED TABLE NAME DESCRIPTOR.
                            066F  1893 ;       R10 = ADDRESS OF PROBED AND COPIED LOGICAL NAME DESCRIPTOR.
                            066F  1894 ;
                            066F  1895
                            066F  1896 EXEDELLNM:
                            066F  1897
        5A    08 AC     D0  066F  1898          MOVL    DNLOGNAM(AP),R10        ;ADDRESS OF LOGICAL NAME DESCRIPTOR
                  15    13  0673  1899          BEQL    10$                     ;BRANCH IF NOT PRESENT
                            0675  1900          IFNORD  #8,(R10),900$           ;CHECK ACCESS TO DESCRIPTOR
            50    6A    7E  067B  1901          MOVAQ   (R10),R0                ;ADDRESS OF DESCRIPTOR
                F97F'   30  067E  1902          BSBW    LNM$PROBER              ;PROBE LOGICAL NAME STRING
                  EA    50  E9  0681  1903          BLBC    R0,930$             ;BRANCH IF CAN'T READ OR NOT PRESENT
            7E    51    7D  0684  1904          MOVQ    R1,-(SP)                ;SAVE LOGICAL NAME DESCRIPTOR
            5A    5E    D0  0687  1905          MOVL    SP,R10                  ;ADDRESS OF LOGICAL NAME DESCRIPTOR
                            068A  1906 10$:
        59    04 AC     D0  068A  1907          MOVL    DNTABNAM(AP),R9         ;ADDRESS OF TABLE NAME DESCRIPTOR
                  DB    13  068E  1908          BEQL    910$                    ;ERROR IF NOT PRESENT
                            0690  1909          IFNORD  #8,(R9),900$            ;CHECK ACCESS TO DESCRIPTOR
            50    69    7E  0696  1910          MOVAQ   (R9),R0                 ;SET UP CALL
                F964'   30  0699  1911          BSBW    LNM$PROBER              ;PROBE TABLE NAME STRING
                  C2    50  E9  069C  1912          BLBC    R0,930$             ;BRANCH IF CAN'T READ OR NOT PRESENT
            7E    51    7D  069F  1913          MOVQ    R1,-(SP)                ;SAVE TABLE NAME DESCRIPTOR
            59    5E    D0  06A2  1914          MOVL    SP,R9                   ;ADDRESS OF TABLE NAME DESCRIPTOR
                            06A5  1915
        50    0C AC     D0  06A5  1916          MOVL    DNACMODE(AP),R0         ;GET SPECIFIED ACCESS MODE
                  14    13  06A9  1917          BEQL    20$                     ;BRANCH IF NOT PRESENT
                            06AB  1918          IFNORD  #1,(R0),900$            ;CHECK ACCESS TO VALUE
            50    60    9A  06B1  1919          MOVZBL  (R0),R0                 ;GET VALUE
            50    03    D1  06B4  1920          CMPL    #PSL$C_USER,R0          ;CHECK FOR VALID ACCESS MODE
                  B2    1F  06B7  1921          BLSSU   910$                    ;INVALID ACCESS MODE
                            06B9  1922
                            06B9  1923 ;
                            06B9  1924 ; IF THE ACCESS MODE OF THE LOGICAL NAME TABLE ENTRY(S) TO BE DELETED WAS
                            06B9  1925 ; EXPLICITELY SPECIFIED AND THE CALLER HAS THE SYSNAM PRIVILEGE, THEN THE
                            06B9  1926 ; SPECIFIED ACCESS MODE IS USED TO QUALIFY THE LOGICAL NAME TABLE ENTRY(S)
                            06B9  1927 ; DELETION. OTHERWISE, THE ACCESS MODE OF THE CALLER IS  MAXIMIZED WITH ANY
                            06B9  1928 ; EXPLICTELY SPECIFIED ACCESS MODE AND USED TO QUALIFY THE LOGICAL NAME TABLE
                            06B9  1929 ; ENTRY(S) DELETION.
                            06B9  1930 ;
                            06B9  1931
                            06B9  1932          IFPRIV  SYSNAM,30$              ;CALLER HAS SYSNAM PRIVILEGE
```

```
                00000000'GF   16  06BF  1933 20$:    JSB     G^EXE$MAXACMODE          ;MAXIMIZED WITH MODE OF CALLER TO
                     55   50   D0  06C5  1934 30$:    MOVL    R0,R5                    ;DETERMINE ACCESS MODE OF TABLE ENTRY(S)
                                  06C8  1935
                                  06C8  1936 ;
                                  06C8  1937 ; RAISE IPL TO AST DELIVERY LEVEL SO THAT THERE ARE NO INTERRUPTIONS WHILE
                                  06C8  1938 ; THE DELETION OF THE LOGICAL NAME TABLE ENTRY(S) IS BEING CARRIED OUT, AND
                                  06C8  1939 ; LOCK THE LOGICAL NAME MUTEX FOR WRITING.
                                  06C8  1940 ;
                                  06C8  1941
                                  06C_  1942          SETIPL  S^#IPL$_ASTDEL           ;RAISE TO AST DELIVERY LEVEL
                         F932'    30  06CB  1943          BSBW    LNM$LOCKW                ;LOCK TABLES FOR WRITING
                                  06CE  1944
                                  06CE  1945 ;
                                  06CE  1946 ; IF A LOGICAL NAME WAS SPECIFIED, THEN IT REPRESENTS THE LOGICAL NAME TABLE
                                  06CE  1947 ; ENTRY WITH THE SPECIFIED ACCESS MODE WHICH IS TO BE DELETED FROM THE LOGICAL
                                  06CE  1948 ; NAME TABLE WITH THE SPECIFIED LOGICAL NAME TABLE NAME. FIND THE FIRST SUCH
                                  06CE  1949 ; LOGICAL NAME TABLE ENTRY AND DELETE IT. IF THE ACCESS MODE REQUESTED IS OTHER
                                  06CE  1950 ; THAN USER, THAN ALL OUTER ACCESS MODE ALIASES IN THIS TABLE ARE ALSO DELETED.
                                  06CE  1951 ;
                                  06CE  1952 ; NOTE THAT IF THE FIRST SUCH LOGICAL NAME TABLE ENTRY IS THAT OF A LOGICAL
                                  06CE  1953 ; NAME TABLE, THEN ALL LOGICAL NAMES CONTAINED WITHIN THE TABLE TOGETHER WITH
                                  06CE  1954 ; ALL ITS SUBTABLES AND THE LOGICAL NAMES CONTAINED WITHIN THEM ARE DELETED.
                                  06CE  1955 ;
                                  06CE  1956
                            5A    D5  06CE  1957          TSTL    R10                      ;IS A SPECIFIC ENTRY TO BE DELETED?
                            54    13  06D0  1958          BEQL    60$                      ;BRANCH IF SUPPOSED TO DELETE THEM ALL
                                  06D2  1959
                         50  6A    7D  06D2  1960          MOVQ    (R10),R0                 ;LOGICAL NAME DESCRIPTOR
                         52  69    7D  06D5  1961          MOVQ    (R9),R2                  ;TABLE NAME DESCRIPTOR
                            F925'  30  06D8  1962          BSBW    LNM$SEARCHLOG            ;SEARCH FOR JUST SUCH A LOGICAL NAME
                           67 50   E9  06DB  1963          BLBC    R0,80$                   ;EXIT IF NO NAME FOUND
                                  06DE  1964
                      0B A1        55  91  06DE  1965          CMPB    R5,LNMB$B_ACMODE(R1)     ;DO THE ACCESS MODES MATCH
                                  07  13  06E2  1966          BEQLU   40$                      ;IF SO THEN HAVE FOUND ENTRY TO DELETE
                   50    01BC     8F  3C  06E4  1967          MOVZWL  #SS$_NOLOGNAM,R0         ;OTHERWISE GO RETURN ERROR
                                  5A  11  06E9  1968          BRB     80$
                                  06EB  1969
                                  06EB  1970 ;
                                  06EB  1971 ; DETERMINE WHETHER THE CALLER CAN DELETE THE LOGICAL NAME TABLE ENTRY, AND
                                  06EB  1972 ; THEN DELETE IS IF SUCH ACCESS IS ALLOWED. THE ACCESS REQUIREMENTS FOR DELETION
                                  06EB  1973 ; ARE AS FOLLOWS:
                                  06EB  1974 ;
                                  06EB  1975 ; LOGICAL NAME:       THE CALLER NEEDS WRITE ACCESS TO THE CONTAINING LOGICAL
                                  06EB  1976 ;                     NAME TABLE.
                                  06EB  1977 ;
                                  06EB  1978 ; LOGICAL NAME TABLE: THE CALLER NEEDS EITHER WRITE ACCESS TO THE CONTAINING
                                  06EB  1979 ;                     LOGICAL NAME TABLE (THE SYSTEM OR PROCESS DIRECTORY
                                  06EB  1980 ;                     TABLE), OR DELETE ACCESS TO THE LOGICAL NAME TABLE
                                  06EB  1981 ;                     ITSELF.
                                  06EB  1982 ;
                                  06EB  1983 ; NOTE THAT IF THE LOGICAL NAME TABLE ENTRY IS PROCESS-PRIVATE THEN NO
                                  06EB  1984 ; PROTECTION CHECKING IS PERFORMED, AND THE CALLER CAN ALWAYS DELETE THE ENTRY.
                                  06EB  1985 ;
                                  06EB  1986
                      32 51        1F  E1  06EB  1987 40$:     BBC     #31,R1,50$               ;SKIP CHECK IF PROCESS-PRIVATE ENTRY
                                  52  02  9A  06EF  1988          MOVZBL  #WRITE_ACCESS,R2         ;CODE FOR ACCESS CHECK
                                  51  DD  06F2  1989          PUSHL   R1                       ;SAVE LNMB ADDRESS
```

```
                51    0C A1   D0  06F4  1990          MOVL    LNMB$L_TABLE(R1),R1     ;GET TABLE HEADER ADDRESS
                  F905'   30    06F8  1991          BSBW    LNM$CHECK_PROT          ;CHECK THE PROTECTION
                     51 8ED0    06FB  1992          POPL    R1                      ;RESTORE LNMB
                  20 50   E8    06FE  1993          BLBS    R0,50$                  ;DELETE THE ENTRY ON SUCCESS
                     03   E1    0701  1994          BBC     #LNMB$V_TABLE,-         ;RETURN AN ERROR ON FAILURE IF THE
              3F 10 A1          0703  1995                  LNMB$B_FLAGS(R1),80$    ;ENTRY TO BE DELETED IS NOT A TABLE
                                0706  1996
                52    08   9A   0706  1997          MOVZBL  #DELETE_ACCESS,R2       ;CODE FOR ACCESS CHECK
                      51   DD   0709  1998          PUSHL   R1                      ;SAVE LNMB ADDRESS
             50    11 A1   9A   070B  1999          MOVZBL  LNMB$T_NAME(R1),R0      ;RETRIEVE SIZE OF NAME STRING
          51    12 A140   9E   070F  2000          MOVAB   LNMB$T_NAME+1(R1)[R0],R1;POSITION TO LNMX
             51    05 A1   9E   0714  2001          MOVAB   LNMX$T_XLATION+1(R1),R1 ;POSITION TO TABLE HEADER
                  F8E5'   30    0718  2002          BSBW    LNM$CHECK_PROT          ;CHECK THE PROTECTION
                     51 8ED0    071B  2003          POPL    R1                      ;RESTORE LNMB
                  24 50   E9    071E  2004          BLBC    R0,80$                  ;QUIT ON FAILURE
                                0721  2005
                  F8DC'   30    0721  2006  50$:     BSBW    LNM$DELETE_LNMB         ;DELETE ALL APPROPRIATE LOGICAL NAMES
                    1F    11    0724  2007          BRB     80$                     ;AND RETURN STATUS OF DELETION
                                0726  2008
                                0726  2009  ;
                                0726  2010  ; IF NO LOGICAL NAME WAS SPECIFIED, THEN ALL THE LOGICAL NAME TABLE ENTRIES
                                0726  2011  ; CONTAINED WITHIN THE SPECIFIED LOGICAL NAME TABLE AND POSSESSING AN ACCESS
                                0726  2012  ; MODE EQUAL TO OR GREATER THEN THE DESIGNATED ACCESS MODE ARE TO BE DELETED.
                                0726  2013  ; POSITION TO THE SPECIFIED LOGICAL NAME TABLE, AND PERFORM THE ACTUAL DELETION
                                0726  2014  ; OF THE NAMES CONTAINED WITHIN THE TABLE.
                                0726  2015  ;
                                0726  2016  ; TWO POSSIBLE CASES EXIST:
                                0726  2017  ;
                                0726  2018  ; CASE 1: THE SPECIFIED LOGICAL NAME TABLE NAME IS THAT OF A LOGICAL NAME TABLE
                                0726  2019  ;          OTHER THAN THE SYSTEM/PROCESS DIRECTORY TABLE. IN THIS CASE ALL THE
                                0726  2020  ;          LOGICAL NAMES, BUT NOT THE LOGICAL NAME TABLE SUB-STRUCTURE, THAT
                                0726  2021  ;          POSSESS AN ACCESS MODE EQUAL TO OR GREATER THAN THE DESIGNATED ACCESS
                                0726  2022  ;          MODE ARE DELETED. THE LOGICAL NAME TABLE SUB-STRUCTURE IS NOT DELETED
                                0726  2023  ;          BECAUSE ALL LOGICAL NAME TABLES ARE CONTAINED WITHIN THE SYSTEM OR
                                0726  2024  ;          PROCESS DIRECTORY TABLES AND NOT WITHIN INDIVIDUAL LOGICAL NAME
                                0726  2025  ;          TABLES.
                                0726  2026  ;
                                0726  2027  ; CASE 2: THE SPECIFIED LOGICAL NAME TABLE NAME IS THAT OF THE SYSTEM/PROCESS
                                0726  2028  ;          DIRECTORY TABLE. IN THIS CASE ALL OF THE LOGICAL NAME TABLE ENTRIES
                                0726  2029  ;          (INCLUDING ALL LOGICAL NAME TABLES) CONTAINED WITHIN THE SPECIFIED
                                0726  2030  ;          DIRECTORY TABLE AND POSSESSING AN ACCESS MODE EQUAL TO OR GREATER
                                0726  2031  ;          THAN THE DESIGNATED ACCESS MODE ARE DELETED.
                                0726  2032  ;
                                0726  2033  ; NOTE THAT IF THE LOGICAL NAME TABLE IS PROCESS-PRIVATE THEN NO PROTECTION
                                0726  2034  ; CHECKING IS PERFORMED, AND THE CALLER CAN ALWAYS DELETE THE ENTRY(S) IN THE
                                0726  2035  ; TABLE.
                                0726  2036  ;
                                0726  2037
          51    55   D0   0726  2038  60$:     MOVL    R5,R1                   ;ACCESS MODE
          52    69   7D   0729  2039          MOVQ    (R9),R2                 ;TABLE NAME DESCRIPTOR
             F8D1'   30    072C  2040          BSBW    LNM$FIRSTTAB            ;SEARCH FOR SPECIFIED LOGICAL NAME TABLE
               13 50   E9   072F  2041          BLBC    R0,80$                  ;BRANCH ON FAILURE
                                0732  2042
                                0732  2043  ;
                                0732  2044  ; MAKE SURE THAT THE CALLER HAS WRITE ACCESS TO THE LOGICAL NAME TABLE BEFORE
                                0732  2045  ; DELETING ALL THE ENTRIES THAT IT CONTAINS.
                                0732  2046  ;
```

```
                     0732  2047
      00    E1       0732  2048          BBC     #LNMTH$V_SHAREABLE,-    ;IS THE TABLE PROCESS-PRIVATE?
   09 61             0734  2049                  LNMTH$B_FLAGS(R1),70$   ;IF SO THEN SKIP ACCESS CHECK
52    02    9A       0736  2050          MOVZBL  #WRITE_ACCESS,R2        ;CODE FOR ACCESS CHECK
   F8C4'    30       0739  2051          BSBW    LNM$CHECK_PROT          ;CHECK THE PROTECTION
   06 50    E9       073C  2052          BLBC    R0,80$                  ;QUIT ON FAILURE
                     073F  2053
52    55    D0       073F  2054 70$:     MOVL    R5,R2                   ;ACCESS MODE
   F8BB'    30       0742  2055          BSBW    LNM$DELETE_TAB          ;DELETE ALL ENTIRES IN SPECIFIED TABLE
                     0745  2056
                     0745  2057 ;
                     0745  2058 ; UNLOCK THE LOGICAL NAME TABLE MUTEX AND RETURN STATUS TO CALLER.
                     0745  2059 ;
                     0745  2060
      50    DD       0745  2061 80$:     PUSHL   R0                      ;SAVE RETURN CODE OVER UNLOCKING
   F8B6'    30       0747  2062          BSBW    LNM$UNLOCK              ;UNLOCK TABLES
      50 8ED0       074A  2063          POPL    R0                      ;RESTORE RETURN CODE
            04       074D  2064          RET                             ;RETURN TO CALLER
                     074E  2065
                     074E  2066          .DISABLE        LSB
                     074E  2067
                     074E  2068 ;        .PAGE
```

```
074E  2070          .SBTTL  EXE$TRNLNM      - TRANSLATE LOGICAL NAME
074E  2071 ;+
074E  2072 ; EXE$TRNLNM - TRANSLATE LOGICAL NAME
074E  2073 ;
074E  2074 ; THIS SERVICE PROVIDES THE CAPABILITY TO LOOKUP A LOGICAL NAME IN THE SPECIFIED
074E  2075 ; LOGICAL NAME TABLE, AND TO RETURN INFORMATION ABOUT IT IN THE CALLER SPECIFIED
074E  2076 ; ITEM LIST.
074E  2077 ;
074E  2078 ; INPUTS:
074E  2079 ;
074E  2080 ;       TRATTR(AP)      = ADDRESS OF LOGICAL NAME TRANSLATION ATTRIBUTES.
074E  2081 ;       TRTABNAM(AP)    = ADDRESS OF TABLE NAME STRING DESCRIPTOR.
074E  2082 ;       TRLOGNAM(AP)    = ADDRESS OF LOGICAL NAME STRING DESCRIPTOR.
074E  2083 ;       TRACMODE(AP)    = ADDRESS OF ACCESS MODE.
074E  2084 ;       TRITMLST(AP)    = ADDRESS OF ITEM LIST.
074E  2085 ;
074E  2086 ;       R4 = CURRENT PROCESS PCB ADDRESS.
074E  2087 ;
074E  2088 ; OUTPUTS:
074E  2089 ;
074E  2090 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO TRANSLATE LOGICAL NAME.
074E  2091 ;
074E  2092 ;               R0 = SS$_ACCVIO - LOGICAL NAME DESCRIPTOR, LOGICAL NAME STRING,
074E  2093 ;                   TABLE NAME DESCRIPTOR, TABLE NAME STRING, AN ITEM IN THE
074E  2094 ;                   ITEM LIST, AN INDEX ITEM BUFFER CANNOT BE READ BY
074E  2095 ;                   CALLING ACCESS MODE. A TABLE ITEM BUFFER, A TABLE ITEM
074E  2096 ;                   SIZE BUFFER, AN ATTRIBUTES ITEM BUFFER, AN ATTRIBUTES
074E  2097 ;                   ITEM SIZE BUFFER, A STRING ITEM BUFFER, A STRING ITEM
074E  2098 ;                   SIZE BUFFER, A LENGTH ITEM BUFFER, A LENGTH ITEM SIZE
074E  2099 ;                   BUFFER, AN ACMODE ITEM BUFFER, AN ACMODE ITEM SIZE
074E  2100 ;                   BUFFER CANNOT BE WRITTEN BY CALLING ACCESS MODE.
074E  2101 ;
074E  2102 ;               R0 = SS$_BADPARAM - INVALID ATTRIBUTE, ACCESS MODE, ITEM TYPE,
074E  2103 ;                   ITEM LENGTH, SPECIFIED. LOGICAL NAME DESCRIPTOR OR
074E  2104 ;                   TABLE NAME DESCRIPTOR NOT SPECIFIED.
074E  2105 ;
074E  2106 ;               R0 = SS$_IVLOGNAM - ZERO OR GREATER THAN MAXIMUM LENGTH
074E  2107 ;                   LOGICAL OR TABLE NAME STRING SPECIFIED.
074E  2108 ;
074E  2109 ;               R0 = SS$_IVLOGTAB - INVALID TABLE NAME SPECIFIED.
074E  2110 ;
074E  2111 ;               R0 = SS$_NOLOGNAM - LOGICAL TABLE NAME ENTRY SPECIFIED DOES NOT
074E  2112 ;                   EXIST.
074E  2113 ;
074E  2114 ;               R0 = SS$_NOPRIV - PROCESS DOES NOT HAVE PRIVILEGE TO ACCESS
074E  2115 ;                   THE SPECIFIED LOGICAL NAME TABLE ENTRY.
074E  2116 ;
074E  2117 ;               R0 = SS$_TOOMANYLNM - TOO MANY LEVELS OF RECURSION IN SEARCH
074E  2118 ;                   FOR LOGICAL NAME TABLE.
074E  2119 ;
074E  2120 ;       R0 LOW BIT SET INDICATES SUCCESSFUL COMPLETION.
074E  2121 ;
074E  2122 ;               R0 = SS$_BUFFEROVF - REQUEST SUCCESSFULLY COMPLETED. AN ITEM
074E  2123 ;                   BUFFER IS NOT LARGE ENOUGH TO HOLD REQUESTED DATA.
074E  2124 ;
074E  2125 ;               R0 = SS$_NORMAL - NORMAL COMPLETION.
074E  2126 ;
```

```
                                074E  2127 ; SIDE EFFECTS:
                                074E  2128 ;
                                074E  2129 ;       THIS ROUTINE EXITS AT IPL 2.
                                074E  2130 ;
                                074E  2131 ;-
                                074E  2132
                          0000000F  2133         .PSECT  Y$EXEPAGED
                              0FFC  000F  2134         .ENTRY  EXE$TRNLNM,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                     0742'     31  0011  2135         BRW     EXETRNLNM
                                   0014  2136
                          0000074E  2137         .PSECT  YF$$LNM
                                074E  2138         .ENABLE LSB
                        50    0C  3C  074E  2139 900$:   MOVZWL  #SS$_ACCVIO,R0          ;ACCESS VIOLATION
                                  04  0751  2140        RET
                        50    14  3C  0752  2141 910$:   MOVZWL  #SS$_BADPARAM,R0        ;BAD SYSTEM SERVICE PARAMETER
                                  04  0755  2142 920$:   RET
                                      0756  2143
                                      0756  2144 EXETRNLNM:
                                      0756  2145
                                      0756  2146 ;
                                      0756  2147 ; VALIDATE AND COPY PARAMETERS AS NECESSARY. REGISTER ASSIGNMENT FOR THE
                                      0756  2148 ; PARAMETERS ARE AS FOLLOWS:
                                      0756  2149 ;
                                      0756  2150 ;       R5  = ACCESS MODE.
                                      0756  2151 ;       R7  = ATTRIBUTE BITS.
                                      0756  2152 ;       R9  = ADDRESS OF PROBED AND COPIED TABLE NAME DESCRIPTOR.
                                      0756  2153 ;       R10 = ADDRESS OF PROBED AND COPIED LOGICAL NAME DESCRIPTOR.
                                      0756  2154 ;       R11 = ADDRESS OF START OF THE ITEM LIST.
                                      0756  2155 ;
                                      0756  2156 ;---
                                      0756  2157
                                7E    D4  0756  2158         CLRL    -(SP)                   ;THIS WILL EVENTUALLY BE CHAIN COUNTER
                                      0758  2159
                        57    04  AC    D0  0758  2160         MOVL    TRATTR(AP),R7           ;TRANSLATION ATTRIBUTES
                                    12  13  075C  2161         BEQL    10$                     ;BRANCH IF NOT PRESENT
                                        075E  2162         IFNORD  #4,(R7),900$            ;CHECK ACCESS
                                57    67  D0  0764  2163         MOVL    (R7),R7                 ;GET VALUE
                        57  FDFFFFFF  8F  D3  0767  2164         BITL    #^C< -
                                        076E  2165                 LNM$M_CASE_BLIND -      ;CASE SENSITIVE VS. CASE INSENSITIVE
                                        076E  2166                 >,R7
                                    E2  12  076E  2167         BNEQ    910$                    ;INVALID TRANSLATION ATTRIBUTES
                                        0770  2168
                        5A    0C  AC    D0  0770  2169 10$:    MOVL    TRLOGNAM(AP),R10        ;ADDRESS OF LOGICAL NAME DESCRIPTOR
                                    DC  13  0774  2170         BEQL    910$                    ;ERROR IF NOT PRESENT
                                        0776  2171         IFNORD  #8,(R10),900$           ;CHECK ACCESS TO DESCRIPTOR
                            50    6A  7E  077C  2172         MOVAQ   (R10),R0                ;ADDRESS OF DESCRIPTOR
                               F87E' 30  077F  2173         BSBW    LNM$PROBER              ;PROBE LOGICAL NAME STRING
                              D0 50  E9  0782  2174         BLBC    R0,920$                 ;BRANCH IF CAN'T READ OR NOT PRESENT
                            7E    51  7D  0785  2175         MOVQ    R1,-(SP)                ;SAVE LOGICAL NAME DESCRIPTOR
                            5A    5E  D0  0788  2176         MOVL    SP,R10                  ;ADDRESS OF LOGICAL NAME DESCRIPTOR
                                      078B  2177
                        59    08  AC    D0  078B  2178         MOVL    TRTABNAM(AP),R9         ;ADDRESS OF TABLE NAME DESCRIPTOR
                                    C1  13  078F  2179         BEQL    910$                    ;ERROR IF NOT PRESENT
                                        0791  2180         IFNORD  #8,(R9),900$            ;CHECK ACCESS TO DESCRIPTOR
                            50    69  7E  0797  2181         MOVAQ   (R9),R0                 ;ADDRESS OF DESCRIPTOR
                               F863' 30  079A  2182         BSBW    LNM$PROBER              ;PROBE TABLE NAME STRING
                              B5 50  E9  079D  2183         BLBC    R0,920$                 ;BRANCH IF CAN'T READ OR NOT PRESENT
```

```
                                                H 8
                     7E    51    7D  07A0  2184          MOVQ    R1,-(SP)                ;SAVE TABLE NAME DESCRIPTOR
                     59    5E    D0  07A3  2185          MOVL    SP,R9                   ;ADDRESS OF TABLE NAME DESCRIPTOR
                                     07A6  2186
                     55    03    9A  07A6  2187          MOVZBL  #PSL$C_USER,R5          ;DEFAULT IS USER MODE(ALL ACCESS)
                50     10 AC    D0  07A9  2188          MOVL    TRACMODE(AP),R0         ;GET SPECIFIED ACCESS MODE
                           0E    13  07AD  2189          BEQL    2000$                   ;BRANCH IF NOT PRESENT
                                     07AF  2190          IFNORD  #1,(R0),900$            ;CHECK ACCESS
                     55    60    9A  07B5  2191          MOVZBL  (R0),R5                 ;DETERMINE ACCESS MODE OF LOGICAL NAME
                     55    03    D1  07B8  2192          CMPL    #PSL$C_USER,R5          ;INVALID ACCESS MODE?
                           95    1F  07BB  2193          BLSSU   910$                    ;RETURN ERROR IF THIS IS THE CASE
                                     07BD  2194
                                     07BD  2195 ;
                                     07BD  2196 ; RAISE IPL TO AST DELIVERY LEVEL SO THAT THERE ARE NO INTERRUPTIONS WHILE THE
                                     07BD  2197 ; TRANSLATION OF THE LOGICAL NAME IS BEING CARRIED OUT, LOCK THE LOGICAL NAME
                                     07BD  2198 ; MUTEX FOR READING, AND THEN SEARCH FOR THE SPECIFIED LOGICAL NAME.
                                     07BD  2199 ;
                                     07BD  2200
                                     07BD  2201 2000$:  SETIPL  S^#IPL$_ASTDEL          ;RAISE TO AST DELIVERY LEVEL
                        F83D'  30    07C0  2202          BSBW    LNM$LOCKR               ;LOCK TABLES FOR READING
                          54   DD    07C3  2203          PUSHL   R4                      ;SAVE PCB ADDRESS
                                     07C5  2204
                     50    6A    7D  07C5  2205          MOVQ    (R10),R0                ;LOGICAL NAME DESCRIPTOR
                     52    69    7D  07C8  2206          MOVQ    (R9),R2                 ;TABLE NAME DESCRIPTOR
                04 57      19    E1  07CB  2207          BBC     #LNM$V_CASE_BLIND,R7,2010$;CASE INSENSITIVE SEARCH?
                00 55      08    E2  07CF  2208          BBSS    #8,R5,2010$             ;YES - SET CORRESPONDING BIT
                                     07D3  2209
                                     07D3  2210 2010$:
                      00000002       07D3  2211          .IF NE CA$_MEASURE              ;CHECK FOR MEASUREMENT ENABLED
            00000000'EF      D6    07D3  2212          INCL    L^PMS$GL_LOGNAM         ;IF YES COUNT CURRENT TRANSLATION
                                     07D9  2213          .ENDC
                                     07D9  2214
                        F824'  30    07D9  2215          BSBW    LNM$SEARCHLOG           ;SEARCH FOR JUST SUCH A LOGICAL NAME
                          50   DD    07DC  2216          PUSHL   R0                      ;SAVE THE SEARCH STATUS
                      23 50    E9    07DE  2217          BLBC    R0,3020$                ;BRANCH IF NO NAME FOUND
                                     07E1  2218
                                     07E1  2219 ;
                                     07E1  2220 ; CHECK THE CALLER'S ACCESS TO THE LOGICAL TABLE THAT HAS BEEN FOUND TO CONTAIN
                                     07E1  2221 ; THE SPECIFIED LOGICAL NAME. NOTE THAT ACCESS CHECKING IS NOT REQUIRED IF THE
                                     07E1  2222 ; LOGICAL NAME IS A PROCESS-PRIVATE LOGICAL NAME.
                                     07E1  2223 ;
                                     07E1  2224
                     58    51    D0  07E1  2225          MOVL    R1,R8                   ;ADDRESS OF LOGICAL NAME BLOCK
                16 51      1F    E1  07E4  2226          BBC     #31,R1,3010$            ;SKIP ACCESS CHECK IF PROCESS-PRIVATE
                     52    01    D0  07E8  2227          MOVL    #READ_ACCESS,R2         ;CODE FOR ACCESS CHECK
                54     04 AE    D0  07EB  2228          MOVL    4(SP),R4                ;RESTORE PCB ADDR
                51     0C A8    D0  07EF  2229          MOVL    LNMB$L_TABLE(R8),R1     ;ADDRESS OF TABLE HEADER
                        F80A'  30    07F3  2230          BSBW    LNM$CHECK_PROT          ;CHECK THE PROTECTION
                     05 50      E8  07F6  2231          BLBS    R0,3010$                ;CONTINUE IF OKAY
                   6E    50    D0  07F9  2232          MOVL    R0,(SP)                 ;SAVE ERROR STATUS
                           06    11  07FC  2233          BRB     3020$                   ;AND EXIT
                                     07FE  2234
                                     07FE  2235 ;
                                     07FE  2236 ; PROCESS THE ITEM LIST RETURNING WHAT INFORMATION THE CALLER WANTS INTO THE
                                     07FE  2237 ; SPECIFIED ITEM BUFFERS.
                                     07FE  2238 ;
                                     07FE  2239 ; VALID ITEMS ARE:
                                     07FE  2240 ;       ACMODE          - ACCESS MODE OF LOGICAL NAME TABLE ENTRY.
```

```
                                                                ;
                                                                ;       ATTRIBUTES      - ATTRIBUTES OF LOGICAL NAME AND CURRENT TRANSLATION.
                                                                ;       INDEX           - INDEX OF TRANSLATION (INPUT ITEM).
                                                                ;       LENGTH          - LENGTH OF CURRENT TRANSLATION STRING.
                                                                ;       MAX_INDEX       - MAXIMUM INDEX IN LOGICAL NAME TABLE ENTRY.
                                                                ;       STRING          - TRANSLATION STRING.
                                                                ;       TABLE           - LOGICAL NAME TABLE NAME STRING.
                                                                ;       PARENT          - PARENT LOGICAL NAME TABLE NAME STRING.
                                                                ;
                                                                ;
                                                                ; REGISTER USAGE IS AS FOLLOWS:
                                                                ;
                                                                ;       R6  = CURRENT TRANSLATION INDEX.
                                                                ;       R7  = ADDRESS OF FIRST LOGICAL NAME TRANSLATION BLOCK
                                                                ;       R8  = ADDRESS OF LOGICAL NAME TABLE ENTRY.
                                                                ;       R9  = ADDRESS OF CURRENT LOGICAL NAME TRANSLATION BLOCK.
                                                                ;       R11 = ADDRESS OF CURRENT ITEM IN ITEM LIST.
                                                                ;
                                                                ;
5B    14 AC    D0  07FE  2259 3010$:  MOVL    TRITMLST(AP),R11         ;ADDRESS OF ITEM LIST
         03    12  0802  2260         BNEQ    3030$                    ;PROCESS ITEM LIST IF THERE IS ONE
       01F1    31  0804  2261 3020$:  BRW     4010$                    ;BRANCH IF NONE
                   0807  2262
50     11 A8    9A  0807  2263 3030$:  MOVZBL  LNMB$T_NAME(R8),R0       ;SIZE OF LOGICAL NAME STRING
59  12 A840    9E  080B  2264         MOVAB   LNMB$T_NAME+1(R8)[R0],R9 ;ADDRESS OF FIRST TRANSLATION BLOCK
   57    59    D0  0810  2265         MOVL    R9,R7                    ;SAVE THIS ADDRESS
         56    D4  0813  2266         CLRL    R6                       ;DEFAULT INDEX IS 0
         02    E0  0815  2267 3035$:  BBS     #LNMX$V_XEND,-           ;LAST TRANSLATION BLOCK?
      10 69        0817  2268                 LNMX$B_FLAGS(R9),3038$   ;YES - THEN POSITION TO GET ITEM
      01 A9    95  0819  2269         TSTB    LNMX$B_INDEX(R9)         ;POSITION TO (OR PAST) 0 LNMX?
         0B    18  081C  2270         BGEQ    3038$                    ;YES - THEN POSITION TO GET ITEM
50     04 A9    9A  081E  2271         MOVZBL  LNMX$T_XLATION(R9),R0    ;SIZE OF TRANSLATION STRING
59  05 A940    9E  0822  2272         MOVAB   LNMX$T_XLATION+1(R9)[R0],R9 ;POSITION TO NEXT TRANSLATION BLOCK
         EC    11  0827  2273         BRB     3035$                    ;CONTINUE LOOKING FOR SPECIFIED LNMX
                   0829  2274 3038$:  IFNORD  #4,(R11),3060$           ;CHECK IF FIRST LONGWORD READABLE
                   082F  2275 3040$:
52     02 AB    3C  082F  2276         MOVZWL  2(R11),R2                ;GET ITEM TYPE
         CF    13  0833  2277         BEQL    3020$                    ;DONE IF ITEM TYPE IS ZERO
                   0835  2278         IFNORD  #12,4(R11),3060$         ;CHECK REST OF THIS DESCRIPTOR
                   083C  2279                                          ;PLUS FIRST LONGWORD OF NEXT ONE
                   083C  2280
                   083C  2281         CASE    R2,<-                    ;HANDLE EACH ITEM TYPE SEPARATELY
                   083C  2282                     3100$, -             ;INDEX ITEM
                   083C  2283                     3200$, -             ;STRING ITEM
                   083C  2284                     3300$, -             ;ATTRIBUTES ITEM
                   083C  2285                     3400$, -             ;TABLE ITEM
                   083C  2286                     3500$, -             ;LENGTH ITEM
                   083C  2287                     3600$, -             ;ACMODE ITEM
                   083C  2288                     3700$, -             ;MAX_INDEX ITEM
                   083C  2289                     3800$, -             ;PARENT ITEM
                   083C  2290                 >,W,#1
FFFF 8F    52    B1  0850  2291         CMPW    R2,#^XLNM$_CHAIN                 ;CHECK FOR CHAINED ITEM LIST
         06    13  0855  2292         BEQL    3070$
       019B    31  0857  2293 3050$:  BRW     3980$                    ;ILLEGAL ITEM
       0193    31  085A  2294 3060$:  BRW     3970$                    ;ACCVIO
                   085D  2295
                   085D  2296 ;
                   085D  2297 ; PROCESS A CHAINED ITEM LIST.
```

```
                                085D  2298 ;
                                085D  2299
                     FC AD   D6 085D  2300 3070$:  INCL    -4(FP)                  ;UPDATE ITEM LIST CHAIN COUNTER
              0400 8F FC AD   B1 0860  2301        CMPW    -4(FP),#1024
                        EF   14 0866  2302        BGTR    3050$                   ;TOO MANY - ASSUME LOOP
                  5B    04 AB D0 0868  2303        MOVL    4(R11),R11              ;GET POINTER
                        BB   11 086C  2304        BRB     3038$                   ;GO PROCESS IT
                                086E  2305
                                086E  2306 ;
                                086E  2307 ; INDEX ITEM.
                                086E  2308 ;
                                086E  2309 ; VALIDATE THE ITEM. POSITION TO THE TRANSLATION BLOCK WITH THE SAME
                                086E  2310 ; OR A GREATER INDEX THEN THAT SPECIFIED BY THE ITEM.
                                086E  2311 ;
                                086E  2312
                     6B   04    B1 086E  2313 3100$:  CMPW    #4,(R11)                ;IS ITEM BUFFER AT LEAST A LONGWORD?
                          E4    1A 0871  2314        BGTRU   3050$                   ;ERROR IF NOT
                  56    04 AB   D0 0873  2315        MOVL    4(R11),R6               ;PICK UP THE ITEM BUFFER ADDRESS
                                0877  2316        IFNORD  (R11),(R6),3060$        ;PROBE IT FOR READABILITY
                     56   66    F6 087D  2317        CVTLB   (R6),R6                 ;IS THE INDEX BETWEEN -127 AND 127?
                          D5    1D 0880  2318        BVS     3050$                   ;NO  - INVALID INDEX ITEM
                     56   56    9A 0882  2319        MOVZBL  R6,R6                   ;ZERO OUT HIGH ORDER THREE BYTES
                     59   57    D0 0885  2320        MOVL    R7,R9                   ;ADDRESS OF FIRST TRANSLATION BLOCK
                                0888  2321
                          02    E0 0888  2322 3110$:  BBS     #LNMX$V_XEND,-          ;LAST TRANSLATION BLOCK?
                        11 69         088A  2323                LNMX$B_FLAGS(R9),3120$  ;YES - THEN POSITION TO NEXT ITEM
                  01 A9    56   91 088C  2324        CMPB    R6,LNMX$B_INDEX(R9)     ;POSITION TO (OR PAST) REQUESTED LNMX?
                           0B   15 0890  2325        BLEQ    3120$                   ;YES - THEN POSITION TO NEXT ITEM
                  50    04 A9   9A 0892  2326        MOVZBL  LNMX$T_XLATION(R9),R0   ;SIZE OF TRANSLATION STRING
              59  05 A940       9E 0896  2327        MOVAB   LNMX$T_XLATION+1(R9)[R0],R9 ;POSITION TO NEXT TRANSLATION BLOCK
                          EB    11 089B  2328        BRB     3110$                   ;CONTINUE LOOKING FOR SPECIFIED LNMX
                        014A    31 089D  2329 3120$:  BRW     3960$                   ;GO POSITION TO NEXT ITEM
                                08A0  2330
                                08A0  2331 ;
                                08A0  2332 ; STRING ITEM.
                                08A0  2333 ;
                                08A0  2334 ; VALIDATE THE ITEM. IF THE CURRENT TRANSLATION BLOCK IS THE LAST
                                08A0  2335 ; TRANSLATION BLOCK, OR IF THE INDEX OF THE CURRENT TRANSLATION BLOCK
                                08A0  2336 ; DOES NOT MATCH THE INDEX LAST SPECIFIED, THEN NOTHING IS RETURNED
                                08A0  23   ; (A ZERO IS STORED IN THE RETURN ADDRESS LENGTH BUFFER). OTHERWISE,
                                08A0  23     THE EQUIVALENCE STRING OF THE SPECIFIED TRANSLATION IS RETURNED.
                                08A0  233; ;
                                08A0  2340
                     50   7C    08A0  2341 3200$:  CLRQ    R0                      ;ASSUME 0 BYTES WILL BE RETURNED
                          02    E0 08A2  2342        BBS     #LNMX$V_XEND,-          ;POSITION TO LAST LNMX?
                        0D 69         08A4  2343                LNMX$B_FLAGS(R9),3210$  ;YES - 0 BYTES WILL BE RETURNED
                  01 A9    56   91 08A6  2344        CMPB    R6,LNMX$B_INDEX(R9)     ;POSITIONED TO SPECIFIED LNMX?
                           07   12 08AA  2345        BNEQ    3210$                   ;NO  - 0 BYTES WILL BE RETURNED
                                08AC  2346
              51    04 A9   9E 08AC  2347        MOVAB   LNMX$T_XLATION(R9),R1   ;RETRIEVE ADDRESS AND SIZE OF
                 50    81   90 08B0  2348        MOVB    (R1)+,R0                ;TRANSLATION STRING
                        00BF    31 08B3  2349 3210$:  BRW     3900$                   ;GO MOVE STRING INTO CURRENT ITEM
                                08B6  2350
                                08B6  2351 ;
                                08B6  2352 ; ATTRIBUTES ITEM.
                                08B6  2353 ;
                                08B6  2354 ; VALIDATE THE ITEM. THE ATTRIBUTES OF THE LOGICAL NAME BLOCK ARE
```

```
                                                    ;  8
                     08B6  2355 ; ALWAYS RETURNED. IF A TRANSLATION WITH THE CURRENT INDEX EXISTS
                     08B6  2356 ; THEN THE TRANSLATION ATTRIBUTES ARE RETURNED AND LNM$V_EXIST IS SET.
                     08B6  2357 ;
                     08B6  2358
    52   10 A8   9A  08B6  2359 3300$:  MOVZBL  LNMB$B_FLAGS(R8),R2       ;LOGICAL NAME BLOCK FLAGS
              02   E0  08BA  2360         BBS     #LNMX$V_XEND,-           ;POSITION TO LAST LNMX?
           17 69       08BC  2361                 LNMX$B_FLAGS(R9),3310$   ;YES - TRANSLATION DOES NOT EXIST
 01 A9      56   91  08BE  2362         CMPB    R6,LNMX$B_INDEX(R9)      ;POSITIONED TO SPECIFIED LNMX?
              11   12  08C2  2363         BNEQ    3310$                    ;NO  - TRANSLATION DOES NOT EXIST
                     08C4  2364
      50      69   9A  08C4  2365         MOVZBL  LNMX$B_FLAGS(R9),R0      ;TRANSLATION BLOCK FLAGS
   50 50      08   9C  08C7  2366         ROTL    #8,R0,R0                 ;ROTATE THEM INTO THEIR PROPER PLACE
      52      50   C8  08CB  2367         BISL2   R0,R2                    ;STORE THEM
52    00000400 8F   C8  08CE  2368         BISL2   #LNM$M_EXISTS,R2         ;SET THE TRANSLATION DOES EXIST BIT
                     08D5  2369
    51   0C A8   D0  08D5  2370 3310$:  MOVL    LNMB$L_TABLE(R8),R1      ;GET CONTAINING TABLE HEADER ADDRESS
              00   E1  08D9  2371         BBC     #LNMTH$V_SHAREABLE,-     ;IS THE CONTAINING TABLE SHAREABLE?
           07 61       08DB  2372                 LNMTH$B_FLAGS(R1),3320$  ;GO MOVE ATTRIBUTES IF NOT
52    00010000 8F   C8  08DD  2373         BISL2   #LNM$M_SHAREABLE,R2      ;SET SHAREABLE BIT
            00E2   31  08E4  2374 3320$:  BRW     3940$                    ;GO MOVE ATTRIBUTES INTO CURRENT ITEM
                     08E7  2375
                     08E7  2376 ;
                     08E7  2377 ; TABLE ITEM.
                     08E7  2378 ;
                     08E7  2379 ; VALIDATE THE ITEM. RETURN THE TABLE NAME STRING OF THE LOGICAL
                     08E7  2380 ; NAME TABLE IN WHICH THIS LOGICAL NAME TABLE ENTRY IS FOUND.
                     08E7  2381 ;
                     08E7  2382
51   0C A8   D0  08E7  2383 3400$:  MOVL    LNMB$L_TABLE(R8),R1      ;ADDRESS OF TABLE HEADER
51   09 A1   D0  08EB  2384         MOVL    LNMTH$L_NAME(R1),R1      ;ADDRESS OF TABLE LNMB
51   11 A1   9E  08EF  2385         MOVAB   LNMB$T_NAME(R1),R1       ;RETRIEVE ADDRESS AND SIZE OF
   50   81   9A  08F3  2386         MOVZBL  (R1)+,R0                 ;TABLE NAME
     007C   31  08F6  2387         BRW     3900$                    ;GO MOVE TABLE INTO CURRENT ITEM
                     08F9  2388
                     08F9  2389 ;
                     08F9  2390 ; LENGTH ITEM.
                     08F9  2391 ;
                     08F9  2392 ; VALIDATE THE ITEM. IF THE CURRENT TRANSLATION BLOCK IS THE LAST
                     08F9  2393 ; TRANSLATION BLOCK, OR IF THE INDEX OF THE CURRENT TRANSLATION
                     08F9  2394 ; BLOCK DOES NOT MATCH THE INDEX LAST SPECIFIED, THEN NOTHING IS
                     08F9  2395 ; RETURNED (A ZERO IS STORED IN THE  RETURN ADDRESS LENGTH BUFFER).
                     08F9  2396 ; OTHERWISE, THE LENGTH OF THE TRANSLATION  STRING OF THE SPECIFIED
                     08F9  2397 ; TRANSLATION IS RETURNED.
                     08F9  2398 ;
                     08F9  2399
      52   D4   08F9  2400 3500$:  CLRL    R2                       ;ASSUME 0 BYTES WILL BE RETURNED
                     08FB  2401
      02   E0  08FB  2402         BBS     #LNMX$V_XEND,-           ;POSITION TO LAST LNMX?
   0A 69       08FD  2403                 LNMX$B_FLAGS(R9),3510$   ;YES - GO RETURN 0 LENGTH
01 A9    56   91  08FF  2404         CMPB    R6,LNMX$B_INDEX(R9)      ;POSITIONED TO SPECIFIED LNMX?
      04   12  0903  2405         BNEQ    3510$                    ;NO  - GO RETURN 0 LENGTH
52 04 A9   9A  0905  2406         MOVZBL  LNMX$T_XLATION(R9),R2    ;LENGTH OF TRANSLATION STRING
     00BD   31  0909  2407 3510$:  BRW     3940$                    ;MOVE LONGOWRD ITEM VALUE
     00E6   31  090C  2408 3520$:  BRW     3980$                    ;BADPARAM
                     090F  2409
                     090F  2410 ;
                     090F  2411 ; ACMODE ITEM.
```

```
                                                                L 8
                     090F  2412 ;
                     090F  2413 ; VALIDATE THE ITEM. RETURN THE ACCESS MODE OF THE LOGICAL NAME TABLE
                     090F  2414 ; ENTRY.
                     090F  2415 ;
                     090F  2416
   6B   01   B1      090F  2417 3600$:  CMPW    #1,(R11)                ;IS ITEM BUFFER AT LEAST A BYTE?
        F8   1A      0912  2418         BGTRU   3520$                   ;ERROR IF NOT
                     0914  2419
50   04 AB   D0      0914  2420         MOVL    4(R11),R0               ;RETRIEVE ADDRESS OF ITEM BUFFER
                     0918  2421         IFNOWRT #1,(R0),3620$           ;PROBE ITEM BUFFER
60   08 A8   90      091E  2422         MOVB    LNMB$B_ACMODE(R8),(R0)  ;RETURN ACMODE IN CURRENT ITEM
50   08 AB   D0      0922  2423         MOVL    8(R11),R0               ;WAS A RETURN LENGTH BUFFER SPECIFIED?
        09   13      0926  2424         BEQL    3610$                   ;DON'T RETURN LENGTH IF ONE WASN'T
                     0928  2425         IFNOWRT #2,(R0),3620$           ;PROBE RETURN LENGTH BUFFER
   60   01   B0      092E  2426         MOVW    #1,(R0)                 ;MOVE IN NUMBER OF BYTES RETURNED
      0086   31      0931  2427 3610$:  BRW     3960$                   ;GO POSITION TO NEXT ITEM
      00B9   31      0934  2428 3620$:  BRW     3970$                   ;ACCVIO
                     0937  2429
                     0937  2430 ;
                     0937  2431 ; MAX_INDEX ITEM.
                     0937  2432 ;
                     0937  2433 ; VALIDATE THE ITEM. RETURN THE MAXIMUM POSITIVE INDEX DEFINED FOR
                     0937  2434 ; THE LOGICAL  NAME TABLE ENTRY.
                     0937  2435 ;
                     0937  2436
   52   01   CE      0937  2437 3700$:  MNEGL   #1,R2                   ;INITIALIZE MAX_INDEX TO -1
   51   57   D0      093A  2438         MOVL    R7,R1                   ;ADDRESS OF FIRST TRANSLATION BLOCK
                     093D  2439
        02   E0      093D  2440 3710$:  BBS     #LNMX$V_XEND,-          ;POSITION TO LAST LNMX?
     0F 61           093F  2441                 LNMX$B_FLAGS(R1),3720$  ;YES - HAVE FOUND MAXIMUM INDEX
   52   01 A1   98    0941  2442         CVTBL   LNMX$B_INDEX(R1),R2     ;NO  - SAVE INDEX
   50   04 A1   9A    0945  2443         MOVZBL  LNMX$T_XLATION(R1),R0   ;SIZE OF TRANSLATION STRING
51   05 A140   9E    0949  2444         MOVAB   LNMX$T_XLATION+1(R1)[R0],R1 ;POSITION TO NEXT TRANSLATION BLOCK
        ED   11      094E  2445         BRB     3710$                   ;CONTINUE LOOKING FOR LAST LNMX
                     0950  2446
        52   D5      0950  2447 3720$:  TSTL    R2                      ;IS MAXIMUM INDEX A NEGATIVE NUMBER?
        75   18      0952  2448         BGEQ    3940$                   ;NO -  GO RETURN MAXIMUM INDEX
   52   01   CE      0954  2449         MNEGL   #1,R2                   ;YES - SET MAXIMUM INDEX TO -1
        70   11      0957  2450         BRB     3940$                   ;GO MOVE MAXIMUM INDEX INTO CURRENT ITEM
                     0959  2451
                     0959  2452 ;
                     0959  2453 ; PARENT TABLE ITEM.
                     0959  2454 ;
                     0959  2455 ; VALIDATE THE ITEM. IF THIS LNMB IS FOR A LOGICAL NAME TABLE, THEN RETURN
                     0959  2456 ; THE TABLE NAME STRING OF THIS TABLE'S PARENT LOGICAL NAME TABLE; OTHERWISE,
                     0959  2457 ; RETURN 0.
                     0959  2458 ;
                     0959  2459
        50   7C      0959  2460 3800$:  CLRQ    R0                      ;ASSUME 0 BYTES WILL BE RETURNED
        03   E1      095B  2461         BBC     #LNMB$V_TABLE,-         ;RETURN 0 BYTES IF THIS IS NOT A LNMB
   15 10 A8           095D  2462                 LNMB$B_FLAGS(R8),3900$  ;FOR A LOGICAL NAME TABLE
52   05 A7   9E      0960  2463         MOVAB   LNMX$T_XLATION+1(R7),R2 ;POSITION TO TABLE HEADER
52   0D A2   D0      0964  2464         MOVL    LNMTH$L_PARENT(R2),R2   ;RETRIEVE PARENT TABLE HEADER ADDRESS
        0B   13      0968  2465         BEQL    3900$                   ;RETURN 0 BYTES IF NO PARENT TABLE
51   09 A2   D0      096A  2466         MOVL    LNMTH$L_NAME(R2),R1     ;RETRIEVE PARENT LNMB ADDRESS
51   11 A1   9E      096E  2467         MOVAB   LNMB$T_NAME(R1),R1      ;RETRIEVE ADDRESS AND SIZE OF
   50   81   90      0972  2468 3810$:  MOVB    (R1)+,R0                ;PARENT TABLE NAME STRING
```

```
                                0975  2469
                                0975  2470 ;
                                0975  2471 ; THE CURRENT ITEM IS TO BE FILLED IN WITH A VARIABLE LENGTH CHARACTER STRING
                                0975  2472 ; (TABLE, STRING, OR PARENT ITEM). IF THE ITEM BUFFER PROVIDED IS NOT
                                0975  2473 ; SUFFICIENTLY LARGE TO CONTAIN ALL OF THE STRING, THEN THE RETURN STATUS IS
                                0975  2474 ; CHANGED TO INDICATE THIS, AND AS MUCH OF THE INFORMATION AS CAN BE STORED IS
                                0975  2475 ; RETURNED.
                                0975  2476 ;
                                0975  2477
            7E    50    7D    0975  2478 3900$:  MOVQ    R0,-(SP)                ;SAVE CHARACTER STRING DESCRIPTOR
            7E    6B    7D    0978  2479         MOVQ    (R11),-(SP)             ;SAVE ITEM BUFFER DESCRIPTOR
            51    6E    3C    097B  2480         MOVZWL  (SP),R1                 ;SIZE OF ITEM BUFFER
     50     04    AE    D0    097E  2481         MOVL    4(SP),R0                ;ADDRESS OF ITEM BUFFER
                  53    D4    0982  2482         CLRL    R3                      ;ACCESS MODE
     00000000'GF         16    0984  2483         JSB     G^EXE$PROBEW            ;PROBE ITEM BUFFER
            51    8E    7D    098A  2484         MOVQ    (SP)+,R1                ;RESTORE ITEM BUFFER DESCRIPTOR
        08  50    E8    098D  2485         BLBS    R0,3910$                ;CONTINUE IF WRITEABLE
            5E    08    C0    0990  2486         ADDL2   #8,SP                   ;REMOVE STRING DESCRIPTOR FROM STACK
            6E    50    D0    0993  2487         MOVL    R0,(SP)                 ;SAVE NEW STATUS
                  60    11    0996  2488         BRB     4010$                   ;GO RELEASE MUTEX AND RETURN
                              0998  2489
                  50    6E    D0    0998  2490 3910$:  MOVL    (SP),R0                 ;SIZE OF CHARACTER STRING TO BE RETURNED
                  18    13    099B  2491         BEQL    3930$                   ;NOTHING TO RETURN IF SIZE IS 0
                  51    50    B1    099D  2492         CMPW    R0,R1                   ;ENOUGH ROOM IN BUFFER?
                  09    1B    09A0  2493         BLEQU   3920$                   ;YES - GO MOVE INFORMATION
08 AE     0601    8F    3C    09A2  2494         MOVZWL  #SS$_BUFFEROVF,8(SP)    ;NO  - CHANGE STATUS TO BUFFER OVERFLOW
                  50    51    3C    09A8  2495         MOVZWL  R1,R0                   ;       AND MOVE ONLY PART OF STRING
                              09AB  2496
                  50    DD    09AB  2497 3920$:  PUSHL   R0                      ;SAVE SIZE OF STRING TO BE MOVED
62     08 BE      50    28    09AD  2498         MOVC3   R0,@8(SP),(R2)          ;MOVE THE CHARACTER STRING
                  50  8ED0    09B2  2499         POPL    R0                      ;RESTORE NUMBER OF BYTES MOVED
                              09B5  2500
               5E    08    C0    09B5  2501 3930$:  ADDL2   #8,SP                   ;REMOVE STRING DESCRIPTOR FROM STACK
            52    08    AB    D0    09B8  2502         MOVL    8(R11),R2               ;WAS A RETURN LENGTH BUFFER SPECIFIED?
                  2C    13    09BC  2503         BEQL    3960$                   ;DON'T PROBE IF ONE WASN'T
                              09BE  2504         IFNOWRT #2,(R2),3970$           ;PROBE RETURN LENGTH BUFFER
               62    50    B0    09C4  2505         MOVW    R0,(R2)                 ;MOVE IN NUMBER OF BYTES RETURNED
                  21    11    09C7  2506         BRB     3960$                   ;GO POSITION TO NEXT ITEM
                              09C9  2507
                              09C9  2508 ;
                              09C9  2509 ; THE CURRENT ITEM IS TO BE FILLED IN WITH A LONGWORD OF INFORMATION
                              09C9  2510 ; (LENGTH, MAX_INDEX, OR ATTRIBUTES ITEM).
                              09C9  2511 ;
                              09C9  2512
               6B    04    B1    09C9  2513 3940$:  CMPW    #4,(R11)                ;IS ITEM BUFFER AT LEAST A LONGWORD?
                  27    1A    09CC  2514         BGTRU   3980$                   ;ERROR IF NOT
            50    04    AB    D0    09CE  2515         MOVL    4(R11),R0               ;RETRIEVE ADDRESS OF ITEM BUFFER
                              09D2  2516         IFNOWRT #4,(R0),3970$           ;PROBE ITEM BUFFER
               60    52    D0    09D8  2517         MOVL    R2,(R0)                 ;RETURN VALUE IN CURRENT ITEM
            50    08    AB    D0    09DB  2518         MOVL    8(R11),R0               ;HAS A RETURN LENGTH BUFFER SPECIFIED?
                  09    13    09DF  2519         BEQL    3960$                   ;DON'T RETURN LENGTH IF ONE WASN'T
                              09E1  2520         IFNOWRT #2,(R0),3970$           ;PROBE RETURN LENGTH BUFFER
               60    04    B0    09E7  2521         MOVW    #4,(R0)                 ;MOVE IN NUMBER OF BYTES RETURNED
                              09EA  2522
                              09EA  2523 ;
                              09EA  2524 ; POSITION TO THE NEXT ITEM IN THE ITEM LIST.
                              09EA  2525 ;
```

```
                                                N 8
                     09EA  2526
 5B   0C   C0       09EA  2527 3960$:  ADDL2   #12,R11                 ;POSITION TO NEXT ITEM
    FE3F   31       09ED  2528         BRW     3040$                   ;AND CONTINUE
                     09F0  2529
 6E   0C   3C       09F0  2530 3970$:  MOVZWL  #SS$_ACCVIO,(SP)        ;ACCESS VIOLATION
      03   11       09F3  2531         BRB     4010$                   ;DEALLOCATE NEW LNMB AND RETURN ERROR
                     09F5  2532
 6E   14   3C       09F5  2533 3980$:  MOVZWL  #SS$_BADPARAM,(SP)      ;BAD PARAMETER SEEN
                     09F8  2534
                     09F8  2535
                     09F8  2536 ;
                     09F8  2537 ; RELEASE THE READ LOCK ON THE LOGICAL NAME MUTEX AND RETURN.
                     09F8  2538 ;
                     09F8  2539
54   04 AE   D0       09F8  2540 4010$:  MOVL    4(SP),R4                ;ADDRESS OF PCB
   F601'  30         09FC  2541         BSBW    LNM$UNLOCK              ;RELEASE LOCK ON LOGICAL NAME MUTEX
     50 8ED0         09FF  2542         POPL    R0                      ;RESTORE RETURN STATUS
           04        0A02  2543         RET                             ;RETURN
                     0A03  2544         .DISABLE        LSB
                     0A03  2545
                     0A03  2546 ;       .PAGE
                     0A03  2547
                     0A03  2548         .END
```

```
ARM$M_DELETE                =  00000008
ARM$M_EXECUTE               =  00000004
ARM$M_READ                  =  00000001
ARM$M_WRITE                 =  00000002
CA$_MEASURE                 =  00000002
CNACMODE                    =  00000010
CNATTR                      =  00000004
CNITMLST                    =  00000014
CNLOGNAM                    =  0000000C
CNTABNAM                    =  00000008
CTACMODE                    =  00000020
CTATTR                      =  00000004
CTPARTAB                    =  0000001C
CTPROT                      =  00000014
CTQUOTA                     =  00000010
CTRESLEN                    =  0000000C
CTRESNAM                    =  00000008
CTTABNAM                    =  00000018
DELETE_ACCESS               =  00000008
DNACMODE                    =  0000000C
DNLOGNAM                    =  00000008
DNTABNAM                    =  00000004
DYN$C_LNM                   =  00000040
EXE$ALOP1PROC                  ********    X   02
EXE$ALOPAGED                   ********    X   02
EXE$CRELNM                     00000005 RG     03
EXE$CRELNT                     00000000 RG     03
EXE$DELLNM                     0000000A RG     03
EXE$LNM_SYNTAX_DAT             ********    X   02
EXE$MAXACMODE                  ********    X   02
EXE$PROBEW                     ********    X   02
EXE$PROBEW_DSC                 ********    X   02
EXE$TRNLNM                     0000000F RG     03
EXECRELNM                      00000329 R      02
EXECRELNT                      00000024 R      02
EXEDELLNM                      0000066F R      02
EXETRNLNM                      00000756 R      02
HEXDIGITS                      00000000 R      02
IPL$_ASTDEL                 =  00000002
LNM$CHECK_PROT                 ********    X   02
LNM$C_TABNAMLEN             =  0000001F
LNM$DELBLK                     ********    X   02
LNM$DELETE_LNMB                ********    X   02
LNM$DELETE_TAB                 ********    X   02
LNM$FIRSTTAB                   ********    X   02
LNM$INIT_PROT                  ********    X   02
LNM$INSLOGTAB                  ********    X   02
LNM$LOCKR                      ********    X   02
LNM$LOCKW                      ********    X   02
LNM$M_CASE_BLIND            =  02000000
LNM$M_CONCEALED             =  00000100
LNM$M_CONFINE               =  00000002
LNM$M_CREATE_IF             =  01000000
LNM$M_CRELOG                =  00000004
LNM$M_EXISTS                =  00000400
LNM$M_NO_ALIAS              =  00000001
LNM$M_SHAREABLE             =  00010000
LNM$M_TERMINAL              =  00000200
LNM$PROBER                     ********    X   02
LNM$SEARCHLOG                  ********    X   02
LNM$UNLOCK                     ********    X   02
LNM$V_CASE_BLIND            =  00000019
LNM$V_CONFINE               =  00000001
LNM$V_SHAREABLE             =  00000010
LNM$_CHAIN                  =  FFFFFFFF
LNM$_INDEX                  =  00000001
LNM$_LNMB_ADDR              =  00000009
LNMB$B_ACMODE               =  00000008
LNMB$B_FLAGS                =  00000010
LNMB$B_TYPE                 =  0000000A
LNMB$L_TABLE                =  0000000C
LNMB$M_NODELETE             =  00000010
LNMB$M_TABLE                =  00000008
LNMB$T_NAME                 =  00000011
LNMB$V_CONFINE              =  00000001
LNMB$V_TABLE                =  00000003
LNMB$W_SIZE                 =  00000008
LNMTH$B_FLAGS               =  00000000
LNMTH$K_LENGTH              =  00000025
LNMTH$L_BYTES               =  00000021
LNMTH$L_BYTESLM             =  0000001D
LNMTH$L_CHILD               =  00000011
LNMTH$L_HASH                =  00000001
LNMTH$L_NAME                =  00000009
LNMTH$L_ORB                 =  00000005
LNMTH$L_PARENT              =  0000000D
LNMTH$L_QTABLE              =  00000019
LNMTH$L_SIBLING             =  00000015
LNMTH$V_DIRECTORY           =  00000001
LNMTH$V_SHAREABLE           =  00000000
LNMX$B_FLAGS                =  00000000
LNMX$B_INDEX                =  00000001
LNMX$C_BACKPTR              =  FFFFFF81
LNMX$C_TABLE                =  FFFFFF82
LNMX$M_TERMINAL             =  00000002
LNMX$M_XEND                 =  00000004
LNMX$T_XLATION              =  00000004
LNMX$V_XEND                 =  00000002
LNMX$W_HASH                 =  00000002
LNM_HEX_TAB_LEN             =  00000014
ORB$C_LENGTH                =  00000058
ORB$K_LENGTH                =  00000058
PCB$L_EPID                  =  00000064
PCB$Q_PRIV                  =  00000084
PMS$GL_LOGNAM                  ********    X   02
PR$_IPL                     =  00000012
PRV$V_SYSNAM                =  00000002
PSL$C_USER                  =  00000003
PSL$M_PRVMOD                =  00C00000
QUOTA_ACCESS                =  00000004
READ_ACCESS                 =  00000001
SS$_ACCVIO                  =  0000000C
SS$_BADPARAM                =  00000014
SS$_BUFFEROVF               =  00000601
```

```
SS$_EXLNMQUOTA                  = 0000224C
SS$_INSFMEM                     = 00000124
SS$_IVLOGNAM                    = 00000154
SS$_IVLOGTAB                    = 0000015C
SS$_NOLOGNAM                    = 000001BC
SS$_NORMAL                      = 00000001
SS$_RESULTOVF                   = 00000214
SS$_SUPERSEDE                   = 00000631
TRACMODE                        = 00000010
TRATTR                          = 00000004
TRITMLST                        = 00000014
TRLOGNAM                        = 0000000C
TRTABNAM                        = 00000008
WRITE_ACCESS                    = 00000002
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
YF$$LNM	00000A03 (2563.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
Y$EXEPAGED	00000014 (20.)	03 (3.)	NOPIC USR C N REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.07	00:00:00.34
Command processing	106	00:00:00.55	00:00:01.48
Pass 1	403	00:00:15.02	00:00:31.49
Symbol table sort	0	00:00:01.91	00:00:03.39
Pass 2	417	00:00:05.92	00:00:13.20
Symbol table output	1	00:00:00.10	00:00:00.22
Psect synopsis output	0	00:00:00.03	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	959	00:00:23.60	00:00:50.15

```
The working set limit was 2100 pages.
94360 bytes (185 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1116 non-local and 159 local symbols.
2548 source lines were read in Pass 1, producing 32 object records in Pass 2.
25 pages of virtual memory were used to define 24 macros.
```

D 9

```
                                    +---------------------------+
                                    ! Macro library statistics !
                                    +---------------------------+

Macro library name                   Macros defined
------------------                   --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                11
_$255$DUA28:[SYSLIB]STARLET.MLB;2             10
TOTALS (all libraries)                        21

1208 GETS were required to define 21 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:SYSLNM/OBJ=OBJ$:SYSLNM MSRC$:SYSLNM/UPDATE=(ENH$:SYSLNM)+EXECML$/LIB
```

0386 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SYSPARAM
LIS

SYSLOGNAM
LIS

SYSMTACC
LIS

SYSIMGSTA
LIS

SYSLNM
LIS

SYSLOAVEC
LIS

SYSLKWSET
LIS

SYSMAILBX
LIS