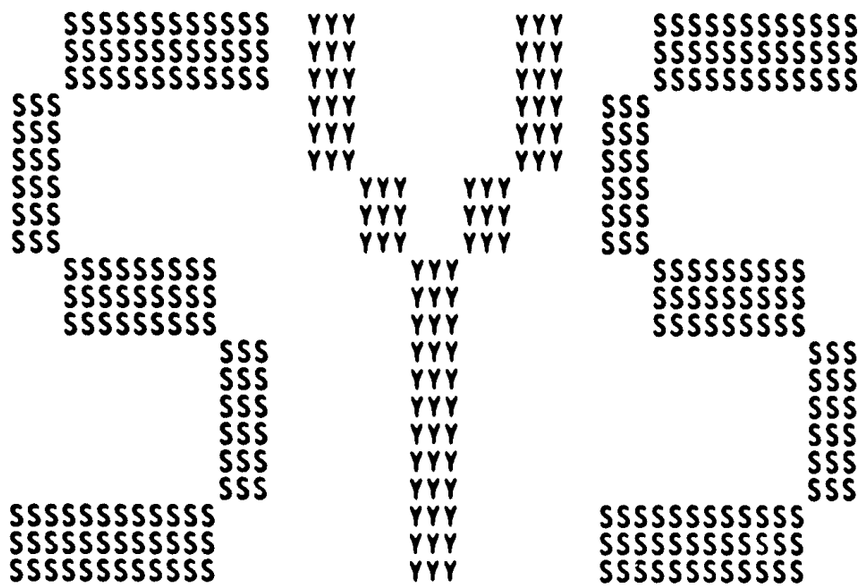

```
IIIIII      000000      SSSSSSSS  UU      UU  BBBBBBBB    PPPPPPPP      AAAAAA      GGGGGGGG  DDDDDDDD
IIIIII      000000      SSSSSSSS  UU      UU  BBBBBBBB    PPPPPPPP      AAAAAA      GGGGGGGG  DDDDDDDD
  II      00      00  SS          UU      UU  BB      BB  PP      PP  AA      AA  GG          DD      DD
  II      00      00  SS          UU      UU  BB      BB  PP      PP  AA      AA  GG          DD      DD
  II      00      00  SS          UU      UU  BB      BB  PP      PP  AA      AA  GG          DD      DD
  II      00      00  SS          UU      UU  BB      BB  PP      PP  AA      AA  GG          DD      DD
  II      00      00    SSSSSS    UU      UU  BBBBBBBB    PPPPPPPP    AA      AA  GG          DD      DD
  II      00      00    SSSSSS    UU      UU  BBBBBBBB    PPPPPPPP    AA      AA  GG          DD      DD
  II      00      00          SS  UU      UU  BB      BB  PP          AAAAAAAAAA  GG  GGGGGG  DD      DD
  II      00      00          SS  UU      UU  BB      BB  PP          AAAAAAAAAA  GG  GGGGGG  DD      DD
  II      00      00          SS  UU      UU  BB      BB  PP          AA      AA  GG      GG  DD      DD
  II      00      00          SS  UU      UU  BB      BB  PP          AA      AA  GG      GG  DD      DD
IIIIII      000000    SSSSSSSS    UUUUUUUUUU  BBBBBBBB    PP          AA      AA    GGGGGG    DDDDDDDD
IIIIII      000000    SSSSSSSS    UUUUUUUUUU  BBBBBBBB    PP          AA      AA    GGGGGG    DDDDDDDD
```

```
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  IOSUBPAGD - PAGED I/O RELATED SUBROUTINES
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  ; D. N. CUTLER 13-JUN-76
0000    29  ;
0000    30  ; PAGED I/O RELATED SUBROUTINES
0000    31  ;
0000    32  ;
0000    33  ; MODIFIED BY:
0000    34  ;
0000    35  ;       V03-023 HH0049          Hai Huang               16-Aug-1984
0000    36  ;               Define IOC$DALLOC_DMT routine for the file systems
0000    37  ;               to deallocate the device on dismount.
0000    38  ;
0000    39  ;       V03-022 RAS0303         Ron Schaefer             1-May-1984
0000    40  ;               Correct RAS0292 to allow 1 or 2 leading "_"s.
0000    41  ;
0000    42  ;       V03-021 ACG0420         Andrew C. Goldstein,    20-Apr-1984  14:03
0000    43  ;               Remove extra kernel mode call in IOC$LOCK_DEV and
0000    44  ;               IOC$UNLOCK_DEV; check status in LKSB in LOCK_DEV.
0000    45  ;               Fix logical name length checks.
0000    46  ;
0000    47  ;       V03-020 RAS0292         Ron Schaefer            12-Apr-1984
0000    48  ;               Correct KPL0110 to allow for leading " " on "NO_TRANS"
0000    49  ;               names.  The NO_TRAN flag merely initializes the translation
0000    50  ;               result block to have the TERMINAL flag set.
0000    51  ;
0000    52  ;       V03-019 KPL0110         Peter Lieberwirth       31-Mar-1984
0000    53  ;               1. Change IOC$SEARCH to allocate a Kernel Request Packet (KRP)
0000    54  ;               to contain $TRNLNM equivalence string because 255 bytes is
0000    55  ;               too much to allocate from the kernel stack.
0000    56  ;
0000    57  ;               2. Change IOC$TRANDEVNAM to honor a new IOC$ bitfield that
```

```
0000     58 ;                  indicates the caller already translated the logical name so
0000     59 ;                  there is no need for TRANDEVNAM to do so.
0000     60 ;
0000     61 ;                  3. Use LNM$C_MAXDEPTH for the maximum logical name recursion
0000     62 ;                  depth.
0000     63 ;
0000     64 ;          V03-018 ACG0039?        Andrew C. Goldstein,    20-Feb-1984  15:45
0000     65 ;                  Rewrite of IOC$SEARCHxxx to break out logical name
0000     66 ;                  translation and device parsing, clean up media type
0000     67 ;                  handling, add handling of device locks, general code
0000     68 ;                  cleanup, move in device lock and unlock routines
0000     69 ;                  from SYSDEVALC. Move low level parse and search code
0000     70 ;                  to IOSUBNPAG so it can be used by IPC.
0000     71 ;
0000     72 ;          V03-017 ROW0288         Ralph O. Weber          24-JAN-1984
0000     73 ;                  Correct stupid bug in ROW0266 which made the cure worse than
0000     74 ;                  the disease.
0000     75 ;
0000     76 ;          V03-016 ROW0266         Ralph O. Weber          28-DEC-1983
0000     77 ;                  Fix error branch in the convert ASCII to integer routine so
0000     78 ;                  that routine return address is popped from the stack.
0000     79 ;
0000     80 ;          V03-015 CDS0001         Christian D. Saether    16-Dec-1983
0000     81 ;                  Add comments reflecting new interpretation of the
0000     82 ;                  CCB$B_AMOD field.  The F11BXQP stores a negative
0000     83 ;                  value in the access mode field of the first channel
0000     84 ;                  available at process creation to reserve it for use
0000     85 ;                  by the F11BXQP exclusively.  It is not actually assigned
0000     86 ;                  to any specific device.
0000     87 ;
0000     88 ;          V03-014 RAS0213         Ron Schaefer            16-Nov-1983
0000     89 ;                  Modify RAS0186 to allow 1 or 2 leading "_"s.   This
0000     90 ;                  is necessary to deal with programs that do a $TRNLOG of
0000     91 ;                  a device name like SYS$INPUT and get an answer
0000     92 ;                  of the form "__TTB3:".
0000     93 ;
0000     94 ;          V03-013 RAS0186         Ron Schaefer            3-Nov-1983
0000     95 ;                  Convert IOC$SEARCHxxx to use $TRNLNM.  For compatibility
0000     96 ;                  a leading "_" is recognized as "SS$_NOTRAN" and discarded.
0000     97 ;
0000     98 ;          V03-012 ROW0238         Ralph O. Weber          11-OCT-1983
0000     99 ;                  Fix wrong direction branch in ROW0232.
0000    100 ;
0000    101 ;          V03-011 ROW0232         Ralph O. Weber          4-OCT-1983
0000    102 ;                  Modify IOC$SEARCHxxx to return UCB of local path to a device
0000    103 ;                  if both a local path and a served path exist.
0000    104 ;
0000    105 ;          V03-010 ROW0228         Ralph O. Weber          23-SEP-1983
0000    106 ;                  Modify IOC$SEARCHxxx device name parser and I/O database
0000    107 ;                  lookup to support device names containing an allocation class
0000    108 ;                  number in place of a node name.  For example, $1$DUA5:, which
0000    109 ;                  means the device DUA5 in allocation class 1.
0000    110 ;
0000    111 ;          V03-009 ROW0217         Ralph O. Weber          7-SEP-1983
0000    112 ;                  Change SEARCHUNIT in IOC$SEARCHxxx to also look for devices on
0000    113 ;                  the secondary DDB chain.
0000    114 ;
```

```
                                K 8
0000    115 ;       V03-008 RAS0175         Ron Schaefer            28-Jul-1983
0000    116 ;               Prevent IOC$SEARCHxxx from recognizing "_".
0000    117 ;               This is temporary until IOC$SEARCHxxx is re-written to
0000    118 ;               use $TRNLNM rather than $TRNLOG.
0000    119 ;
0000    120 ;       V03-007 DMW4036         DMWalp                  26-May-1983
0000    121 ;               Intergate new logical name structures.
0000    122 ;
0000    123 ;       V03-006 KTA3050         Kerbey T. Altmann       16-May-1983
0000    124 ;               Fix 'off by one' bug in KTA3044.
0000    125 ;
0000    126 ;       V03-005 KTA3044         Kerbey T. Altmann       21-Mar-1983
0000    127 ;               Add support for media type allocation.
0000    128 ;
0000    129 ;       V03-004 KTA3022         Kerbey T. Altmann       29-Dec-1982
0000    130 ;               Enhanced KTA3011.
0000    131 ;
0000    132 ;       V03-003 KTA3011         Kerbey T. Altmann       15-Oct-1982
0000    133 ;               Fixed bug that prevented node names with prefixed "_"
0000    134 ;               from being recognized.  Allow "$" in device names.
0000    135 ;               Support for SCA node names.
0000    136 ;
0000    137 ;       V03-002 ROW0130         Ralph O. Weber           5-OCT-1982
0000    138 ;               Remove IOC$CREATE_UCB whose functionality is replaced by
0000    139 ;               routines in module UCBCREDEL.
0000    140 ;
0000    141 ;       V03-001 PHL0100         Peter H. Lipman         01-Jun-1982
0000    142 ;               The sequence $CREMBX, $ASSIGN with a lower case logical
0000    143 ;               name was broken by upcasing device names in IOC$SEARCHDEV.
0000    144 ;               Fix this by trying the TRNLOG with the original string
0000    145 ;               and if NOTRAN, try again with upcased string.
0000    146 ;
0000    147 ;**
```

```
                                0000   149          .SBTTL  Declarations
                                0000   150 ;
                                0000   151 ; Macro library calls
                                0000   152 ;
                                0000   153
                                0000   154          $ACBDEF                         ; define AST control block
                                0000   155          $CCBDEF                         ; define CCB offsets
                                0000   156          $CRBDEF                         ; define CRB offsets
                                0000   157          $DDBDEF                         ; define DDB offsets
                                0000   158          $DEVDEF                         ; define device characteristics
                                0000   159          $IOCDEF                         ; define flag bits
                                0000   160          $IPLDEF                         ; define IPL levels
                                0000   161          $JIBDEF                         ; define JIB offsets
                                0000   162          $LCKDEF                         ; define lock manager symbols
                                0000   163          $LNMDEF                         ; define LNM offsets
                                0000   164          $LNMSTRDEF                      ; define LNM block offsets
                                0000   165          $MSCPDEF                        ; define MSCP offsets
                                0000   166          $PCBDEF                         ; define PCB offsets
                                0000   167          $PRDEF                          ; define processor registers
                                0000   168          $PRVDEF                         ; define privilege bits
                                0000   169          $PSLDEF                         ; define processor status fields
                                0000   170          $SBDEF                          ; define system block
                                0000   171          $SSDEF                          ; define system status values
                                0000   172          $UCBDEF                         ; define UCB offsets
                                0000   173
                                0000   174          ASSUME  IOC$V_PHY EQ 0          ; optimized to BLBx all over
                                0000   175
                            00000000   176          .PSECT  Y$EXEPAGED
                                0000   177 ;
                                0000   178 ; Local data
                                0000   179 ;
                                0000   180 LNM_TBL:
49 46 24 4D 4E 4C 00000008'010E0000'    0000   181          .ASCID  "LNM$FILE_DEV"          ; logical name table for devices
                  56 45 44 5F 45 4C     000E
                                0014   182
                         0000001B       0014   183 ESCAPE  = 27                       ; escape character
```

```
                                                M 8
                     0014   185          .SBTTL  Find Free I/O Channel
                     0014   186
                     0014   187 ;+
                     0014   188 ; IOC$FFCHAN - Find Free I/O Channel
                     0014   189 ;
                     0014   190 ; This routine is called to search the I/O channel table for a free channel.
                     0014   191 ;
                     0014   192 ; INPUTS:
                     0014   193 ;
                     0014   194 ;       NONE
                     0014   195 ;
                     0014   196 ; OUTPUTS:
                     0014   197 ;
                     0014   198 ;       R0 low bit clear indicates failure to find free I/O channel.
                     0014   199 ;
                     0014   200 ;               R0 = SS$_NOIOCHAN - no I/O channel available.
                     0014   201 ;
                     0014   202 ;       R0 low bit set indicates success with:
                     0014   203 ;
                     0014   204 ;               R1 = available channel number.
                     0014   205 ;               R2 = CCB address for channel in R1
                     0014   206 ;
                     0014   207 ;       R3 is preserved across call.
                     0014   208 ;-
                     0014   209
                     0014   210 IOC$FFCHAN::                                ; find free I/O channel
00000000'9F   C1     0014   211          ADDL3   @#CTL$GL_CCBBASE,-
      50   09        001A   212                  #CCB$B_AMOD,R0             ; base and offset to test assignment
      51   10   CE   001C   213          MNEGL   #CCB$C_LENGTH,R1           ; set starting channel index
52    00000000'9F   3C   001F   214          MOVZWL  @#CTL$GW_NMIOCH,R2         ; get number of I/O channels
            0B   13   0026   215          BEQL    20$                        ; there are none
          6041   95   0028   216 10$:     TSTB    (R0)[R1]                   ; channel assigned?
            0C   13   002B   217          BEQL    30$                        ; if eql no
      51    10   C2   002D   218          SUBL    #CCB$C_LENGTH,R1           ; calculate next channel index
        F5  52   F5   0030   219          SOBGTR  R2,10$                     ; any more CCB's to examine?
50    01B4  8F   3C   0033   220 20$:     MOVZWL  #SS$_NOIOCHAN,R0           ; indicate failure
                   05   0038   221          RSB                            ;
                        0039   222
      52    51   CE   0039   223 30$:     MNEGL   R1,R2                      ; convert to positive value
00000000'9F   52   B1   003C   224          CMPW    R2,@#CTL$GW_CHINDX         ; check against current hi-water mark
            07   1F   0043   225          BLSSU   40$                        ; no, just leave
00000000'9F   52   B0   0045   226          MOVW    R2,@#CTL$GW_CHINDX         ; yes, set new mark
52    F7 A041   9E   004C   227 40$:     MOVAB   -CCB$B_AMOD(R0)[R1],R2     ; load R2 with CCB address
      51    51   CE   0051   228          MNEGL   R1,R1                      ; make positive
      50    01   3C   0054   229          MOVZWL  #SS$_NORMAL,R0             ; indicate success
                   05   0057   230          RSB                            ;
```

```
                                        .SBTTL  General I/O Database Search

                                ;+
                                ;
                                ; IOC$SEARCH - general I/O database search
                                ; IOC$SEARCHDEV - search for specific physical device
                                ; IOC$SEARCHALL - generic search for any device
                                ;
                                ; This routine searches the I/O database for the specified device, using
                                ; the specified search rules. Depending on the search, a lock may or may
                                ; not be taken out on the device when it is found.
                                ;
                                ; INPUTS:
                                ;
                                ;       R1 = address of descriptor of device / logical name string
                                ;       R2 = flags
                                ;       R3 = address to store lock value block
                                ;       I/O database mutex held, IPL 2
                                ;
                                ; OUTPUTS:
                                ;
                                ;       R0 = SS$_NORMAL - device found
                                ;          = SS$_ACCVIO - name string is not readable
                                ;          = SS$_NONLOCAL - nonlocal device
                                ;          = SS$_IVLOGNAM - invalid logical name (e.g., too long)
                                ;          = SS$_TOOMANYLNAM - max. logical name recursion exceeded
                                ;          = SS$_IVDEVNAM - invalid device name string
                                ;          = SS$_NOSUCHDEV - device not found
                                ;          = SS$_NODEVAVL - device exists but not available according to rules
                                ;          = SS$_DEVALLOC - device allocated to other user
                                ;          = SS$_NOPRIV - failed device protection
                                ;          = SS$_TEMPLATEDEV - can't allocate template device
                                ;          = SS$_DEVMOUNT - device already mounted
                                ;          = SS$_DEVOFFLINE - device marked offline
                                ;       R1 = UCB
                                ;       R2 = DDB
                                ;       R3 = system block
                                ;       R4 - R11 preserved
                                ;
                                ;       Note: If failure, R1 - R3 point to the last structures looked at.
                                ;
                                ;       R2 and R3 are input only to IOC$SEARCH.
                                ;
                                ;       IOC$SEARCHDEV:  R2 = IOC$M_PHY ! IOC$M_ANY
                                ;                       R3 = 0
                                ;       IOC$SEARCHALL:  R2 = IOC$M_ANY ! IOC$M_LOCAL
                                ;                       R3 = 0
                                ;
                                ;-
                                
                                        .ENABLE LSB
                                
                                IOC$SEARCHDEV::                         ; search for specific device
52   41 8F    9A 0058                   MOVZBL  #IOC$M_PHY!IOC$M_ANY,R2 ; physical device name, no checks
     04 11     005C                      BRB     10$
               005E
               005E                     IOC$SEARCHALL::                  ; generic search for any device
```

```
                                                B 9
                52   48 8F    9A  005E   289          MOVZBL  #IOC$M_ANY!IOC$M_LOCAL,R2 ; no activity checks, local only
                     53    D4  0062   290 10$:      CLRL    R3                      ; no value block
                                0064   291
                                0064   292 IOC$SEARCH::                            ; general purpose I/O search
                   0FE0 8F     BB  0064   293          PUSHR   #^M<R5,R6,R7,R8,R9,R10,R11>
           5A   00000000'GF     9E  0068   294          MOVAB   G^CTL$GL_KRPFL,R10      ; get pointer to KRP lookaside list
                59    04 BA     0F  006F   295          REMQUE  @4(R10),R9              ; allocate a KRP
                        2D     1D  0073   296          BVS     30$                     ; bugcheck if no KRP to use
                     7E   59    D0  0075   297          MOVL    R9,-(SP)                ; save KRP pointer
                     5A   52    7D  0078   298          MOVQ    R2,R10                  ; move flags and val block
                          29    10  007B   299          BSBB    IOC$TRANDEVNAM          ; translate device / logical name
                        09 50    E9  007D   300          BLBC    R0,20$                  ; exit if error
                         FF7D'   30  0080   301          BSBW    IOC$PARSDEVNAM          ; parse device name
                        03 50    E9  0083   302          BLBC    R0,20$                  ; exit if error
                         FF77'   30  0086   303          BSBW    IOC$SEARCHINT           ; and do the search
                     51   55    7D  0089   304 20$:      MOVQ    R5,R1                   ; move UCB and DDB address
                     53   57    D0  008C   305          MOVL    R7,R3                   ; and system block
                     59   8E    D0  008F   306          MOVL    (SP)+,R9                ; restore KRP address
           5A   00000000'GF     9E  0092   307          MOVAB   G^CTL$GL_KRPFL,R10      ; get address of KRP lookaside list
                04 BA    69     0E  0099   308          INSQUE  (R9),@4(R10)            ; deallocate the KRP to its list
                   0FE0 8F     BA  009D   309          POPR    #^M<R5,R6,R7,R8,R9,R10,R11>
                               05  00A1   310          RSB
                                   00A2   311
                                   00A2   312 30$:      BUG_CHECK       KRPEMPTY,FATAL
                                   00A6   313          .DISABLE LSB
```

```
                     00A6   315          .SBTTL  Translate Logical Device Name
                     00A6   316
                     00A6   317 ;+
                     00A6   318 ;
                     00A6   319 ; IOC$TRANDEVNAM - translate logical device name
                     00A6   320 ;
                     00A6   321 ; This routine applies iterative logical name translation to the specified
                     00A6   322 ; device name. In addition, the string is upcased, if translated.
                     00A6   323 ;
                     00A6   324 ; Input buffer should be large enough to contain a logical name equivalence
                     00A6   325 ; string and 5 bytes of logical name block overhead.  The overhead is
                     00A6   326 ; required because this routine calls an internal logical name routine to
                     00A6   327 ; do the translation instead of the slower $TRNLNM.  The additional 5 bytes
                     00A6   328 ; are lnm processing overhead, specifically a LNMX.
                     00A6   329 ;
                     00A6   330 ; INPUTS:
                     00A6   331 ;
                     00A6   332 ;       R1 = address of logical name string descriptor.
                     00A6   333 ;            ***** this string has not yet been probed,
                     00A6   334 ;            ***** but the descriptor has been.
                     00A6   335 ;
                     00A6   336 ;       R2 = IOC$ flags, specifically:
                     00A6   337 ;               IOC$V_NO_TRANS - if set, caller already translated logical name
                     00A6   338 ;
                     00A6   339 ;       R9 = buffer in which to store translated device name
                     00A6   340 ;            (length is assumed to be <LNM$C_NAMLENGTH + LNMX$T_XLATION+1>)
                     00A6   341 ;
                     00A6   342 ; OUTPUTS:
                     00A6   343 ;
                     00A6   344 ;       R0 = SS$_NORMAL - successful translation
                     00A6   345 ;          = SS$_ACCVIO - name string is not readable
                     00A6   346 ;          = SS$_NONLOCAL - nonlocal device
                     00A6   347 ;          = SS$_IVLOGNAM - invalid logical name (e.g., too long)
                     00A6   348 ;          = SS$_TOOMANYLNAM - logical name recursion depth exceeded
                     00A6   349 ;       R8 = length of translated string
                     00A6   350 ;       R9 = address of translated string
                     00A6   351 ;          Note: translated string may not begin at the beginning of the
                     00A6   352 ;          output buffer, ie R9 may point into the input buffer, ie
                     00A6   353 ;          R9 not preserved
                     00A6   354 ;
                     00A6   355 ;-
                     00A6   356
                     00A6   357 ;
                     00A6   358 ; case_blind flag (r5 input) for lnm$search_one, concatenate user mode for
                     00A6   359 ; now
                     00A6   360 ;
          00000103   00A6   361 M_CASE_BLIND    = ^X0103
                     00A6   362
                     00A6   363          .ENABLE LSB
                     00A6   364
                     00A6   365 IOC$TRANDEVNAM::
     00FC 8F    BB   00A6   366          PUSHR   #^M<R2,R3,R4,R5,R6,R7>  ; save working registers
     58   61    3C   00AA   367          MOVZWL  (R1),R8                 ; get length of device/logical name
          38    13   00AD   368          BEQL    20$                     ; if eql invalid name
00FF 8F   58    B1   00AF   369          CMPW    R8,#LNM$C_NAMLENGTH     ; name too long?
          31    1A   00B4   370          BGTRU   20$                     ; if gtru yes
  50   04 A1    D0   00B6   371          MOVL    4(R1),R0                ; get address of device/logical name
```

```
                                00BA   372          ASSUME  LNM$C_NAMLENGTH LE 512  ; ok to use single probe
                                00BA   373          IFNORD  R8,(R0),10$             ; probe logical name buffer
                           69   7C     00C0   374          CLRQ    (R9)            ; and init output buffer lnmx
                     04    52   09     E1     00C2   375          BBC     #IOC$V_NO_TRANS,R2,1$   ; branch if RMS did not do the $TRNLNM
                                01     E2     00C6   376          BBSS    #LNMX$V_TERMINAL,-      ; set terminal flag so no translations
                              00    69          00C8   377                  LNMX$B_FLAGS(R9),1$     ;  are actually done
                           60    58   28     00CA   378 1$:       MOVC3   R8,(R0),-               ; copy logical name into buffer
                             05    A9          00CD   379                  <LNMX$T_XLATION+1>(R9)  ;
                54   00000000'    9F   D0     00CF   380          MOVL    @#CTL$GL_PCB,R4         ; set up PCB address for search_one
                           57    0A   D0     00D6   381          MOVL    #LNM$C_MAXDEPTH,R7      ; maximum number of translations
                           56    59   D0     00D9   382          MOVL    R9,R6                   ;  r6 is output lnmx from search_one
                             05    A9   9E     00DC   383          MOVAB   <LNMX$T_XLATION+1>(R9),-;  r9 is input buffer for search_one
                                59          00DF   384                  R9
                                3A   11     00E0   385          BRB     50$                     ; previous (non-existent) translation
                                            00E2   386
                           50    0C   3C     00E2   387 10$:      MOVZWL  #SS$_ACCVIO,R0          ; name buffer not readable
                                05   11     00E5   388          BRB     25$
                     50    0154 8F   3C     00E7   389 20$:      MOVZWL  #SS$_IVLOGNAM,R0        ; invalid logical name
                                7A   11     00EC   390 25$:      BRB     120$
                                            00EE   391 ;
                                            00EE   392 ; Try to translate a logical name, using a fast, internal interface.
                                            00EE   393 ; Note that LNM$SEARCH_ONE only returns translations for equivalence
                                            00EE   394 ; names for index 0, IE, no search_lists.
                                            00EE   395 ;
                                            00EE   396 ; R8 = size of name string to translate
                                            00EE   397 ; R9 = address of name string to translate
                                            00EE   398
                           50    58   7D     00EE   399 30$:      MOVQ    R8,R0                   ; descriptor of logical name
                     52    FF0B CF   3C     00F1   400          MOVZWL  LNM_TBL,R2              ; get table name length
                     53    FF0A CF   D0     00F6   401          MOVL    LNM_TBL+4,R3            ; table name address
                     55    0103 8F   3C     00FB   402          MOVZWL  #M_CASE_BLIND,R5        ; indicate case_blind, user mode
                      00000000'EF    16     0100   403          JSB     LNM$SEARCH_ONE          ; translate the logical name
                             0B    50   E8     0106   404          BLBS    R0,35$                  ; successful translation
                     50    01BC 8F   B1     0109   405          CMPW    #SS$_NOLOGNAM,R0        ; if failed to translate logical name
                                58   12     010E   406          BNEQ    120$                    ; quit if abnormal
                                01   E3     0110   407          BBCS    #LNMX$V_TERMINAL,-      ; no more translations
                              04    66          0112   408                  LNMX$B_FLAGS(R6),40$    ;
                        58    04   A6   9A     0114   409 35$:      MOVZBL  LNMX$T_XLATION(R6),R8   ; size of translated string
                        59    05   A6   9E     0118   410 40$:      MOVAB   <LNMX$T_XLATION+1>(R6),R9 ; and address of equivalence string
                                            011C   411
                                            011C   412 ;
                                            011C   413 ; R8 = size of (logical) device name string
                                            011C   414 ; R9 = address of (logical) device name string
                                            011C   415 ;
                           69    1B   91     011C   416 50$:      CMPB    #ESCAPE,(R9)            ; RMS IFI on the front?
                                08   12     011F   417          BNEQ    70$                     ; branch if not
                           59    04   C0     0121   418          ADDL    #4,R9                   ; skip around the PPF data
                           58    04   C2     0124   419          SUBL    #4,R8                   ; and adjust size of device string
                                BE   15     0127   420 60$:      BLEQ    20$                     ; branch if bad device name
                                            0129   421
                                            0129   422 ;
                                            0129   423 ; Take an underscore or two off the front.  If any are removed, then
                                            0129   424 ; the device that follows must not be translated any further.  Note
                                            0129   425 ; that the device after the RMS process permanent file data may be
                                            0129   426 ; a logical device name.
                                            0129   427 ;
                                            0129   428
```

```
                     69    5F 8F    91    0129   429 70$:    CMPB    #^A'_',(R9)             ; leading underscore?
                                16    12    012D   430         BNEQ    80$                     ; branch if not
                                59    D6    012F   431         INCL    R9                      ; strip it off
                                58    D7    0131   432         DECL    R8                      ; and adjust the count
                                F2    15    0133   433         BLEQ    60$                     ; branch if bad device name
                                01    E2    0135   434         BBSS    #LNMX$V_TERMINAL,-      ; flag no more translations
                             00 66          0137   435                 LNMX$B_FLAGS(R6),75$
                     69    5F 8F    91    0139   436 75$:    CMPB    #^A'_',(R9)             ; try for a second "_"
                                06    12    013D   437         BNEQ    80$                     ; branch if not
                                59    D6    013F   438         INCL    R9                      ; strip it off
                                58    D7    0141   439         DECL    R8                      ; and adjust the count
                                E2    15    0143   440         BLEQ    60$                     ; branch if bad device name
                                      0145   441
                                      0145   442 ;
                                      0145   443 ; At this point R8,R9 describe a string which is either the initial
                                      0145   444 ; string passed to this routine or a translation of it.  A check will
                                      0145   445 ; now be made to see if this string contains a "::" and is thus a
                                      0145   446 ; nodename.  If not and there were leading "_", then it is a physical
                                      0145   447 ; device name and the translations will be skipped. If no leading "_"
                                      0145   448 ; then the string up to but not including the ":" (if present) will
                                      0145   449 ; be a candidate for translation.  This translation will be attempted
                                      0145   450 ; if the result of a previous translation was not SS$_NOTRAN and if
                                      0145   451 ; the iteration counter has not expired.
                                      0145   452 ;
                                      0145   453
                 69    58    3A    3A    0145   454 80$:    LOCC    #^A':',R8,(R9)          ; search string for a colon
                                0A    13    0149   455         BEQL    90$                     ; if eql colon not found
                                50    D7    014B   456         DECL    R0                      ; possibly a node name?
                                06    13    014D   457         BEQL    90$                     ; if eql no
                             01 A1    3A    91    014F   458         CMPB    #^A':',1(R1)            ; next character a colon?
                                1F    13    0153   459         BEQL    130$                    ; if eql yes
                                      0155   460
                 58    51    59    C3    0155   461 90$:    SUBL3   R9,R1,R8                ; size of string up to colon
                                01    E0    0159   462         BBS     #LNMX$V_TERMINAL,-      ; last translation?
                             08 66          015B   463                 LNMX$B_FLAGS(R6),110$   ; branch if yes, don't do another
                             02 57    F4    015D   464         SOBGEQ  R7,100$                 ; loop if iteration count not exhausted
                                      0160   465                                             ; n+1 iterations for n translations
                                0B    11    0160   466         BRB     125$                    ; skip over loop
                              FF89    31    0162   467 100$:   BRW     30$                     ; branch to top of loop
                                      0165   468
                          50    01    D0    0165   469 110$:   MOVL    #SS$_NORMAL,R0          ; indicate success
                          00FC 8F     BA    0168   470 120$:   POPR    #^M<R2,R3,R4,R5,R6,R7>  ; restore registers
                                05    016C   471         RSB
                                      016D   472
                 50    0374 8F     3C    016D   473 125$:   MOVZWL  #SS$_TOOMANYLNAM,R0     ; too many equivalence strings defined
                                F4    11    0172   474         BRB     120$
                                      0174   475 ;
                                      0174   476 ; Nonlocal device
                                      0174   477 ;
                 50    08F0 8F     3C    0174   478 130$:   MOVZWL  #SS$_NONLOCAL,R0        ; set nonlocal device
                                ED    11    0179   479         BRB     120$
                                      017B   480
                                      017B   481         .DISABLE LSB
```

```
                     017B   483          .SBTTL  Take Out Cluster-wide Device Lock
                     017B   484
                     017B   485 ;+
                     017B   486 ;
                     017B   487 ; IOC$LOCK_DEV
                     017B   488 ;
                     017B   489 ; FUNCTIONAL DESCRIPTION
                     017B   490 ;       Determine the device's allocation name and take out a cluster-wide
                     017B   491 ;       lock on that name.
                     017B   492 ;
                     017B   493 ; INPUTS:
                     017B   494 ;       R0      - lock mode for cluster-wide lock (e.g. LCK$K_EXMODE)
                     017B   495 ;       R1      - address of a 16-byte buffer to be used as lock value block,
                     017B   496 ;                 if the contents of the value block are to be returned.
                     017B   497 ;                 If R1 = zero no value block is used.
                     017B   498 ;       R4      - PCB address
                     017B   499 ;       R5      - UCB address
                     017B   500 ;
                     017B   501 ; IMPLICIT INPUTS:
                     017B   502 ;       IPL = IPL$_ASTDEL
                     017B   503 ;       Process is holding I/O data base mutex
                     017B   504 ;
                     017B   505 ; OUTPUTS:
                     017B   506 ;       R0      - LBS means successful lock.
                     017B   507 ;       R1      - if R0 signals success, R1 will contain the lock id.
                     017B   508 ;
                     017B   509 ; IMPLICIT OUTPUTS:
                     017B   510 ;       The lock id is stored in UCB$L_LOCKID.
                     017B   511 ;       If R0 signals success and the lock value block data was requested,
                     017B   512 ;       it is returned in the user's buffer.
                     017B   513 ;
                     017B   514 ;-
                     017B   515
                     017B   516 IOC$LOCK_DEV::
     01CC 8F    BB   017B   517          PUSHR   #^M<R2,R3,R6,R7,R8>     ; Save some registers.
     57   50    7D   017F   518          MOVQ    R0,R7                   ; Save lock mode and val block addr
                     0182   519 ;
                     0182   520 ; We must construct a resource name to use when locking the device.  Allocate
                     0182   521 ; a buffer to hold the name on the stack, then use IOC$CVT_DEVNAM to
                     0182   522 ; construct the resource name.
                     0182   523 ;
      5E   F0 AE    DE   0182   524          MOVAL   -16(SP),SP              ; Reserve space for device name.
           51 5E    D0   0186   525          MOVL    SP,R1                   ; R1 = buffer address for device name.
24535953 8F         DD   0189   526          PUSHL   #^A'SYS$'               ; Prefix system code to resource name.
           52 5E    D0   018F   527          MOVL    SP,R2                   ; Save address of buffer.
           50 10    D0   0192   528          MOVL    #16,R0                  ; R0 = buffer length for device  name.
           53 54    D0   0195   529          MOVL    R4,R3                   ; Save PCB address.
           54 01    D0   0198   530          MOVL    #1,R4                   ; Signal we want alloc_class+device name.
00000000'EF 16      019B   531          JSB     IOC$CVT_DEVNAM          ; Get back device name.
           54 53    D0   01A1   532          MOVL    R3,R4                   ; Restore PCB address.
             7C 50  E9   01A4   533          BLBC    R0,60$                  ; exit on error
           51 04    C0   01A7   534          ADDL    #4,R1                   ; Add space for SYS$ code name to
                     01AA   535                                           ; returned length of device name string.
                     01AA   536 ;
                     01AA   537 ; Now attempt to take out a lock on the device's resource name.
                     01AA   538 ; At this point, the registers contain:
                     01AA   539 ;       R1      - length of resource name
```

```
                                01AA   540 ;       R2      - address of buffer containing resource name
                                01AA   541 ;
     50    0000005D 8F    D0  01AA   542         MOVL    #<LCK$M_CVTSYS-          ; indicate system-owned lock,
                                01B1   543                   !LCK$M_NOQUEUE-     ; return success/failure immediately,
                                01B1   544                   !LCK$M_SYNCSTS-     ; return success synchronously
                                01B1   545                   !LCK$M_SYSTEM-      ; system lock space
                                01B1   546                   !LCK$M_VALBLK>,R0   ; indicate value block present,
                          7E    7C  01B1   547         CLRQ    -(SP)               ; initialize lock value block
                          7E    7C  01B3   548         CLRQ    -(SP)
                          58    D5  01B5   549         TSTL    R8                  ; see if value block supplied
                          08    13  01B7   550         BEQL    10$                 ; branch if none
                    6E    68    7D  01B9   551         MOVQ    (R8),(SP)           ; set up correct value block
               08 AE      08    A8  01BC   552         MOVQ    8(R8),8(SP)         ; just in case we're converting down
                    20    A5    DD  01C1   553 10$:    PUSHL   UCB$L_LOCKID(R5)    ; Get current lock, if any
                          03    13  01C4   554         BEQL    20$                 ; branch if none
                    50    02    C8  01C6   555         BISL    #LCK$M_CONVERT,R0   ; else make this a conversion
                          7E    D4  01C9   556 20$:    CLRL    -(SP)               ; rest of LKSB
                    53    5E    D0  01CB   557         MOVL    SP,R3               ; Save address of lock status block.
                    7E    51    7D  01CE   558         MOVQ    R1,-(SP)            ; Device name string descriptor
                                01D1   559 ;
                                01D1   560 ; We build the arg list and call the system lock manager subroutines
                                01D1   561 ; directly to avoid the system service dispatcher. This permits us to
                                01D1   562 ; retain the I/O database mutex during the lock manager call.
                                01D1   563 ;
                          7E    7C  01D1   564         CLRQ    -(SP)               ; zero reserved arg & acmode
                          7E    7C  01D3   565         CLRQ    -(SP)               ; zero blkast & astprm
                          7E    7C  01D5   566         CLRQ    -(SP)               ; zero astadr & parid
                       18 AE    9F  01D7   567         PUSHAB  24(SP)              ; resnam
                          50    DD  01DA   568         PUSHL   R0                  ; flags
                          53    DD  01DC   569         PUSHL   R3                  ; lksb
                          57    DD  01DE   570         PUSHL   R7                  ; lkmode
                          7E    D4  01E0   571         CLRL    -(SP)               ; efn
     00000000'EF          0B    FB  01E2   572         CALLS   #11,SYS$ENQ
                             0B 50  E9  01E9   573         BLBC    R0,30$              ; Branch if lock failed.
                       50    63    3C  01EC   574         MOVZWL  (R3),R0             ; get status from lksb
                          05 50    E9  01EF   575         BLBC    R0,30$              ; Branch if lock failed.
                       50    01    3C  01F2   576 35$:    MOVZWL  #SS$_NORMAL,R0      ; Change possible SS$_SYNCH to SS$_NORMAL.
                          13    11  01F5   577         BRB     40$
                09F0 8F    50    B1  01F7   578 30$:    CMPW    R0,#SS$_VALNOTVALID ; check for value block not valid
                          F4    13  01FC   579         BEQL    35$                 ; ignore this error
                09B8 8F    50    B1  01FE   580         CMPW    R0,#SS$_NOTQUEUED   ; see if lock held elsewhere
                          05    12  0203   581         BNEQ    40$                 ; some other error
                50    0840 8F    3C  0205   582         MOVZWL  #SS$_DEVALLOC,R0    ; convert to "device allocated"
                                020A   583 ;
                                020A   584 ; Store user outputs.
                                020A   585 ;
                          58    D5  020A   586 40$:    TSTL    R8                  ; Did user request value block?
                          09    13  020C   587         BEQL    50$                 ; No: skip store of value block.
                    68    08    A3  020E   588         MOVQ    8(R3),(R8)          ; First quadword into user's buffer.
               08 A8      10    A3  0212   589         MOVQ    16(R3),8(R8)        ; Second quadword into user's buffer.
                    51    04    A3  0217   590 50$:    MOVL    4(R3),R1            ; Return lock id in R1.
                    20 A5       51  D0  021B   591         MOVL    R1,UCB$L_LOCKID(R5) ; Also save it in the UCB.
                                021F   592 ;
                                021F   593 ; Clean off stack.
                                021F   594 ;
                    5E    20 AE    DE  021F   595         MOVAL   32(SP),SP           ; Pop lock status and value block.
                    5E    14 AE    DE  0223   596 60$:    MOVAL   20(SP),SP           ; Pop device name buffer off stack.
```

```
01CC 8F   BA  0227    597 70$:    POPR    #^M<R2,R3,R6,R7,R8>     ; Restore the registers.
          05  022B    598         RSB                             ; restore previous PSL
```

```
                     022C   600          .SBTTL  Deallocate Device Cluster-wide
                     022C   601
                     022C   602 ;+
                     022C   603 ;
                     022C   604 ; IOC$DALLOC_DEV
                     022C   605 ;
                     022C   606 ; FUNCTIONAL DESCRIPTION:
                     022C   607 ;
                     022C   608 ;       Deallocate a device.  If the device is available cluster-wide, also
                     022C   609 ;       dequeue the lock on that device.
                     022C   610 ;
                     022C   611 ; INPUTS:
                     022C   612 ;       R4      Address of PCB
                     022C   613 ;       R5      Address of UCB
                     022C   614 ;
                     022C   615 ; IMPLICIT INPUTS:
                     022C   616 ;       IPL = IPL$_ASTDEL
                     022C   617 ;       Process holds I/O data base mutex
                     022C   618 ;
                     022C   619 ; OUTPUTS:
                     022C   620 ;       R0      SS$_NORMAL      - Device deallocated.
                     022C   621 ;               SS$_DEVNOTALLOC - Device wasn't allocated.
                     022C   622 ;
                     022C   623 ;-
                     022C   624
                     022C   625 IOC$DALLOC_DEV::
     50    0858 8F    3C   022C   626          MOVZWL  #SS$_DEVNOTALLOC,R0     ; Assume device not allocated.
   1D 38 A5    17    E5   0231   627          BBCC    #DEV$V_ALL,UCB$L_DEVCHAR(R5),40$
                     0236   628 ;
                     0236   629 ; Clear allocation fields from local UCB. The owner PID is cleared
                     0236   630 ; if the device is shareable or if this is the last reference.
                     0236   631 ;
   03 38 A5    10    E1   0236   632          BBC     #DEV$V_SHR,UCB$L_DEVCHAR(R5),10$
         2C A5    D4   023B   633          CLRL    UCB$L_PID(R5)           ; Clear out owner field.
         5C A5    B7   023E   634 10$:     DECW    UCB$W_REFC(R5)          ; Decrement refcount.
            08    12   0241   635          BNEQ    20$                     ; branch if channels still assigned
         2C A5    D4   0243   636          CLRL    UCB$L_PID(R5)           ; Clear out owner field.
           FDB7'   30   0246   637          BSBW    IOC$LAST_CHAN           ; do final device cleanup
   02 3C A5    00    E1   0249   638          BBC     #DEV$V_CLU,UCB$L_DEVCHAR2(R5),30$
                     024E   639                                          ; Branch if strictly a local device.
            04    10   024E   640 20$:     BSBB    IOC$UNLOCK_DEV          ; Dequeue the cluster-wide lock
         50 01    3C   0250   641 30$:     MOVZWL  #SS$_NORMAL,R0          ; Signal normal successful completion.
                  05   0253   642 40$:     RSB
```

```
                                0254   644          .SBTTL  Release Cluster-wide Device Lock
                                0254   645
                                0254   646 ;+
                                0254   647 ;
                                0254   648 ; IOC$UNLOCK_DEV
                                0254   649 ;
                                0254   650 ; FUNCTIONAL DESCRIPTION:
                                0254   651 ;       Dequeue the cluster-wide lock as called for by the UCB's state.
                                0254   652 ;       If it's still allocated we do nothing. If there are still
                                0254   653 ;       channels assigned, we just demote the lock to CR.
                                0254   654 ;
                                0254   655 ; INPUTS:
                                0254   656 ;       R5 - address of UCB
                                0254   657 ;
                                0254   658 ; IMPLICIT INPUTS:
                                0254   659 ;       UCB$L_LOCKID(R5) contains the ID of the lock to dequeue.
                                0254   660 ;       Caller is at IPL$_ASTDEL, and holds the I/O database mutex.
                                0254   661 ;
                                0254   662 ; OUTPUTS:
                                0254   663 ;       R0 - status of call to $DEQ.
                                0254   664 ;
                                0254   665 ;-
                                0254   666
                                0254   667 IOC$UNLOCK_DEV::
     50    01    D0             0254   668          MOVL    #SS$_NORMAL,R0          ; Assume success.
       2C  A5    D5             0257   669          TSTL    UCB$L_PID(R5)           ; see if it's still allocated
           21    12             025A   670          BNEQ    20$                     ; branch if yes
   51  20  A5    D0             025C   671          MOVL    UCB$L_LOCKID(R5),R1     ; Lock present for this device?
           1B    13             0260   672          BEQL    20$                     ; Branch if no lock for this device.
     7E    50    7D             0262   673          MOVQ    R0,-(SP)                ; build lksb on stack
           7E    7C             0265   674          CLRQ    -(SP)                   ; zero flags & acmode
           7E    D4             0267   675          CLRL    -(SP)                   ; zero value block
       5C  A5    B5             0269   676          TSTW    UCB$W_REFC(R5)          ; check reference count
           10    12             026C   677          BNEQ    30$                     ; if non-zero, must convert to CR
           51    DD             026E   678          PUSHL   R1                      ; lkid
00000000'EF    04    FB         0270   679          CALLS   #4,SYS$DEQ
         20    A5    D4         0277   680          CLRL    UCB$L_LOCKID(R5)        ; Clear the lock id field.
       5E    08    C0           027A   681 10$:     ADDL    #8,SP                   ; clean the stack
             05    027D         682 20$:     RSB
                   027E         683 ;
                   027E         684 ; To here if the UCB still has channels assigned. We convert the lock
                   027E         685 ; down to CR. Note that 3 null arguments are already on the stack.
                   027E         686 ;
           7E    7C             027E   687 30$:     CLRQ    -(SP)                   ; zero astprm & astadr
           7E    7C             0280   688          CLRQ    -(SP)                   ; zero parid & resnam
0000004E   8F    DD             0282   689          PUSHL   #<LCK$M_CVTSYS-         ; indicate system-owned lock,
                                0288   690                   !LCK$M_NOQUEUE-       ; return success/failure immediately,
                                0288   691                   !LCK$M_SYNCSTS-       ; return success synchronously
                                0288   692                   !LCK$M_CONVERT>       ; conversion
         20    AE    9F         0288   693          PUSHAB  32(SP)                  ; lksb
           01    DD             028B   694          PUSHL   #LCK$K_CRMODE           ; lkmode
           7E    D4             028D   695          CLRL    -(SP)                   ; efn
00000000'EF    0B    FB         028F   696          CALLS   #11,SYS$ENQ
           E2    11             0296   697          BRB     10$
```

```
                          0298   699          .SBTTL  Unlock I/O Database and Return Status
                          0298   700
                          0298   701 ;+
                          0298   702 ; IOC$UNLOCK - unlock I/O data base and return status
                          0298   703 ;
                          0298   704 ; This routine is jumped to at the end of an I/O related system service to
                          0298   705 ; unlock the I/O data base, set the current processor priority to zero,
                          0298   706 ; and to return status to the change mode dispatcher.
                          0298   707 ;
                          0298   708 ; INPUTS:
                          0298   709 ;
                          0298   710 ;       R0 = final system service status value.
                          0298   711 ;
                          0298   712 ; OUTPUTS:
                          0298   713 ;
                          0298   714 ;       The I/O data base is unlocked, the current processor priority is set
                          0298   715 ;       to zero, and a return to the change mode dispatcher is executed.
                          0298   716 ;-
                          0298   717
                          0298   718 IOC$UNLOCK::                                ; unlock I/O data base and return status
            50    DD      0298   719          PUSHL   R0                        ; save final system service status value
54  00000000'EF    D0      029A   720          MOVL    CTL$GL_PCB,R4            ; get PCB address
        FD5C'      30      02A1   721          BSBW    SCH$IOUNLOCK             ; unlock I/O data base
                           02A4   722          SETIPL  #0                       ; allow all interrupts
          50   8E    D0    02A7   723          MOVL    (SP)+,R0                 ; retrieve final service status value
                     04    02AA   724          RET                              ;
```

```
                     02AB   726
                     02AB   727          .SBTTL  Verify I/O Channel Number
                     02AB   728
                     02AB   729 ;+
                     02AB   730 ; IOC$VERIFYCHAN - verify I/O channel number
                     02AB   731 ;
                     02AB   732 ; This routine is called to verify and translate an I/O channel number to
                     02AB   733 ; a CCB address. The channel is checked for accessibility by the previous
                     02AB   734 ; access mode.
                     02AB   735 ;
                     02AB   736 ; INPUTS:
                     02AB   737 ;
                     02AB   738 ;       R0 = I/O channel number in low order word
                     02AB   739 ;
                     02AB   740 ; OUTPUTS:
                     02AB   741 ;
                     02AB   742 ;       R0 low bit clear indicates failure to verify.
                     02AB   743 ;
                     02AB   744 ;               R0 = SS$_IVCHAN - invalid channel number.
                     02AB   745 ;               R0 = SS$_NOPRIV - no privilege to access channel.
                     02AB   746 ;               R1 = address of CCB if R0 = SS$_NOPRIV
                     02AB   747 ;
                     02AB   748 ;       R0 low bit set indicates verify success with:
                     02AB   749 ;
                     02AB   750 ;               R1 = address of CCB.
                     02AB   751 ;               R2 = channel index.
                     02AB   752 ;-
                     02AB   753
                     02AB   754 IOC$VERIFYCHAN::                        ; verify I/O channel number
  50   FFFF000F 8F    CA   02AB   755          BICL    #<^XFFFF0000!<CCB$C_LENGTH-1>>,R0 ; clear extraneous bits
                 28   13   02B2   756          BEQL    10$             ; if eql invalid channel
  00000000'9F    50   B1   02B4   757          CMPW    R0,@#CTL$GW_CHINDX      ; legal channel number?
                 1F   1A   02BB   758          BGTRU   10$             ; if gtru no
            52   50   CE   02BD   759          MNEGL   R0,R2           ; convert to channel index
51   00000000'FF42   9E   02C0   760          MOVAB   @CTL$GL_CCBBASE[R2],R1  ; get address of corresponding CCB
                 53   DC   02C8   761          MOVPSL  R3              ; read current PSL
 53   53    02    16   EF   02CA   762          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R3,R3 ; extract previous mode field
            50   24   3C   02CF   763          MOVZWL  #SS$_NOPRIV,R0          ; assume caller does not have privilege
         09 A1   53   91   02D2   764          CMPB    R3,CCB$B_AMOD(R1)       ; caller have privilege to access channel?
                 09   18   02D6   765          BGEQ    20$             ; if geq no - this must be a signed test
                      02D8   766 ;
                      02D8   767 ; Note that the privilege test comparing caller's mode to the access mode
                      02D8   768 ; field of the channel must be a signed comparison.  The F11BXQP reserves
                      02D8   769 ; a channel for use by itself by manually locating a free channel (using
                      02D8   770 ; IOC$FFCHAN) and then storing -1 in the access mode field, when the channel
                      02D8   771 ; is not being actively used by the XQP for logical I/O.  This effectively
                      02D8   772 ; blocks anything, including kernel mode rundown, or any other kernel mode
                      02D8   773 ; code, from messing with the channel.  Of course, when the XQP wants to
                      02D8   774 ; use the channel itself, it modifies the CCB$B_AMOD and CCB$L_UCB fields
                      02D8   775 ; to look like a normal kernel mode channel to the device of its choice.
                      02D8   776 ;
                      02D8   777
      05 50    00    E3   02D8   778          BBCS    #0,R0,20$               ; indicate success
   50    013C 8F    3C   02DC   779 10$:     MOVZWL  #SS$_IVCHAN,R0          ; set invalid channel
                      05   02E1   780 20$:     RSB                     ;
                      02E2   781
```

```
                                02E2    783          .SBTTL  Deallocate device on dismount
                                02E2    784 ;++
                                02E2    785 ; IOC$DALLOC_DMT
                                02E2    786 ;
                                02E2    787 ; FUNCTIONAL DESCRIPTION:
                                02E2    788 ;
                                02E2    789 ;       This routine deallocates the device if the device is marked
                                02E2    790 ;       "deallocate on dismount", or if the device owner has gone away.
                                02E2    791 ;       This routine is called by the file systems' CHECK_DISMOUNT
                                02E2    792 ;       routines, and by IOC$DISMOUNT when dismounting a foreign volume.
                                02E2    793 ;
                                02E2    794 ; CALLING SEQUENCE:
                                02E2    795 ;       JSB IOC$DALLOC_DMT
                                02E2    796 ;
                                02E2    797 ; INPUT:
                                02E2    798 ;       R4 = address of the process PCB
                                02E2    799 ;       R5 = device UCB address
                                02E2    800 ;
                                02E2    801 ; OUTPUT:
                                02E2    802 ;       NONE.
                                02E2    803 ;
                                02E2    804 ; IMPLICIT IMPUT:
                                02E2    805 ;       IPL = IPL$_ASTDEL
                                02E2    806 ;       Process holds I/O database mutex
                                02E2    807 ;
                                02E2    808 ; ROUTINE VALUE:
                                02E2    809 ;       R0 = SS$_NORMAL         - normal successful completion,
                                02E2    810 ;                                 device deallocated when appropriate
                                02E2    811 ;            SS$_DEVNOTALLOC    - device wasn't allocated
                                02E2    812 ;
                                02E2    813 ; SIDE EFFECTS:
                                02E2    814 ;       R1, R3 destroyed.
                                02E2    815 ;
                                02E2    816 ;--
                                02E2    817 IOC$DALLOC_DMT::
     50    0858 8F    3C      02E2    818          MOVZWL  #SS$_DEVNOTALLOC,R0     ; Assume device not allocated.
   1E 38 A5    17    E1      02E7    819          BBC     #DEV$V_ALL, -           ; If device not allocated,
                               02EC    820                  UCB$L_DEVCHAR(R5), 20$  ; return to caller.
                               02EC    821
     50    01    3C      02EC    822          MOVZWL  #SS$_NORMAL,R0          ; Assume success.
   13 64 A5    0A    E4      02EF    823          BBSC    #UCB$V_DEADMO, -        ; Check for deallocate on dismount
                               02F4    824                  UCB$W_STS(R5), 10$      ; branch if yes.
                               02F4    825
     51    2C A5    3C      02F4    826          MOVZWL  UCB$L_PID(R5),R1        ; Pick up device owner's PID.
51   00000000'FF41    D0      02F8    827          MOVL    @L^SCH$GL_PCBVEC[R1],R1 ; Get device owner's PCB address.
   60 A1    2C A5    D1      0300    828          CMPL    UCB$L_PID(R5), -        ; Has the device owner gone away ?
                               0305    829                  PCB$L_PID(R1)           ;
            03    13      0305    830          BEQL    20$                     ; If eql no, return to caller.
                               0307    831
          FF22    30      0307    832 10$:     BSBW    IOC$DALLOC_DEV          ; else complete the deallocation now.
                  05      030A    833 20$:     RSB
                               030B    834
                               030B    835          .END
```

```
BUG$_KRPEMPTY           ********    X   02
CCB$B_AMOD            = 00000009
CCB$C_LENGTH          = 00000010
CTL$GL_CCBBASE          ********    X   02
CTL$GL_KRPFL            ********    X   02
CTL$GL_PCB              ********    X   02
CTL$GW_CHINDX           ********    X   02
CTL$GW_NMIOCH           ********    X   02
DEV$V_ALL             = 00000017
DEV$V_CLU             = 00000000
DEV$V_SHR             = 00000010
ESCAPE                = 0000001B
IOC$CVT_DEVNAM          ********    X   02
IOC$DALLOC_DEV          0000022C RG     02
IOC$DALLOC_DMT          000002E2 RG     02
IOC$FFCHAN              00000014 RG     02
IOC$LAST_CHAN           ********    X   02
IOC$LOCK_DEV            0000017B RG     02
IOC$M_ANY             = 00000040
IOC$M_LOCAL           = 00000008
IOC$M_PHY             = 00000001
IOC$PARSDEVNAM          ********    X   02
IOC$SEARCH              00000064 RG     02
IOC$SEARCHALL           0000005E RG     02
IOC$SEARCHDEV           00000058 RG     02
IOC$SEARCHINT           ********    X   02
IOC$TRANDEVNAM          000000A6 RG     02
IOC$UNLOCK              00000298 RG     02
IOC$UNLOCK_DEV          00000254 RG     02
IOC$VERIFYCHAN          000002AB RG     02
IOC$V_NO_TRANS        = 00000009
IOC$V_PHY             = 00000000
LCK$K_CRMODE          = 00000001
LCK$M_CONVERT         = 00000002
LCK$M_CVTSYS          = 00000040
LCK$M_NOQUEUE         = 00000004
LCK$M_SYNCSTS         = 00000008
LCK$M_SYSTEM          = 00000010
LCK$M_VALBLK          = 00000001
LNM$C_MAXDEPTH        = 0000000A
LNM$C_NAMLENGTH       = 000000FF
LNM$SEARCH_ONE          ********    X   02
LNMX$B_FLAGS          = 00000000
LNMX$T_XLATION        = 00000004
LNMX$V_TERMINAL       = 00000001
LNM_TBL                 00000000 R      02
M_CASE_BLIND          = 00000103
PCB$L_PID             = 00000060
PR$_IPL               = 00000012
PSL$S_PRVMOD          = 00000002
PSL$V_PRVMOD          = 00000016
SCH$GL_PCBVEC           ********    X   02
SCH$IOUNLOCK            ********    X   02
SS$_ACCVIO            = 0000000C
SS$_DEVALLOC          = 00000840
SS$_DEVNOTALLOC       = 00000858
SS$_IVCHAN            = 0000013C
SS$_IVLOGNAM          = 00000154
SS$_NOIOCHAN          = 000001B4
SS$_NOLOGNAM          = 000001BC
SS$_NONLOCAL          = 000008F0
SS$_NOPRIV            = 00000024
SS$_NORMAL            = 00000001
SS$_NOTQUEUED         = 000009B8
SS$_TOOMANYLNAM       = 00000374
SS$_VALNOTVALID       = 000009F0
SYS$DEQ                 ********    X   02
SYS$ENQ                 ********    X   02
UCB$L_DEVCHAR         = 00000038
UCB$L_DEVCHAR2        = 0000003C
UCB$L_LOCKID          = 00000020
UCB$L_PID             = 0000002C
UCB$V_DEADMO          = 0000000A
UCB$W_REFC            = 0000005C
UCB$W_STS             = 00000064
```

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
Y$EXEPAGED	00000308 (779.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.06	00:00:00.58
Command processing	115	00:00:00.56	00:00:03.02
Pass 1	474	00:00:17.80	00:00:52.72
Symbol table sort	0	00:00:02.99	00:00:10.86
Pass 2	159	00:00:03.73	00:00:19.54
Symbol table output	10	00:00:00.10	00:00:00.53
Psect synopsis output	2	00:00:00.03	00:00:00.57
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	797	00:00:25.28	00:01:27.83

The working set limit was 1800 pages.
104438 bytes (204 pages) of virtual memory were used to buffer the intermediate code.
There were 110 pages of symbol table space allocated to hold 1951 non-local and 44 local symbols.
835 source lines were read in Pass 1, producing 16 object records in Pass 2.
30 pages of virtual memory were used to define 29 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	15
_255DUA28:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	26

2078 GETS were required to define 26 macros.

There were no errors, warnings or information messages.

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

IOSUBRAMS
LIS

IPCONTROL
LIS

IOSUBPAGD
LIS

LNMSUB
LIS

IOPERFORM
LIS

IOSUBNPAG
LIS

LINKVEC
LIS