```
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP       SSSSSSSSSSS  YYY         YYY
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP       SSSSSSSSSSS  YYY         YYY
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP       SSSSSSSSSSS  YYY         YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS              YYY         YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS              YYY         YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS              YYY         YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS                 YYY   YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS                 YYY   YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS                 YYY   YYY
UUU         UUU  EEEEEEEEEEE            TTT        PPPPPPPPPPPP      SSSSSSSSS             YYY
UUU         UUU  EEEEEEEEEEE            TTT        PPPPPPPPPPPP      SSSSSSSSS             YYY
UUU         UUU  EEEEEEEEEEE            TTT        PPPPPPPPPPPP      SSSSSSSSS             YYY
UUU         UUU  EEE                    TTT        PPP                        SSS           YYY
UUU         UUU  EEE                    TTT        PPP                        SSS           YYY
UUU         UUU  EEE                    TTT        PPP                        SSS           YYY
UUU         UUU  EEE                    TTT        PPP                        SSS           YYY
UUU         UUU  EEE                    TTT        PPP                        SSS           YYY
UUU         UUU  EEE                    TTT        PPP                        SSS           YYY
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE      TTT        PPP                 SSSSSSSSSSS          YYY
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE      TTT        PPP                 SSSSSSSSSSS          YYY
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE      TTT        PPP                 SSSSSSSSSSS          YYY
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE      TTT        PPP                 SSSSSSSSSSS          YYY
```

```
  CCCCCCCC  LL          SSSSSSSS   IIIIII     OOOOOO    DDDDDDDD   BBBBBBBB
  CCCCCCCC  LL          SSSSSSSS   IIIIII     OOOOOO    DDDDDDDD   BBBBBBBB
CC          LL        SS           II      OO      OO  DD      DD  BB      BB
CC          LL        SS           II      OO      OO  DD      DD  BB      BB
CC          LL        SS           II      OO      OO  DD      DD  BB      BB
CC          LL        SS           II      OO      OO  DD      DD  BB      BB
CC          LL          SSSSSS     II      OO      OO  DD      DD  BBBBBBBB
CC          LL          SSSSSS     II      OO      OO  DD      DD  BBBBBBBB
CC          LL                SS   II      OO      OO  DD      DD  BB      BB
CC          LL                SS   II      OO      OO  DD      DD  BB      BB
CC          LL                SS   II      OO      OO  DD      DD  BB      BB
CC          LL                SS   II      OO      OO  DD      DD  BB      BB
  CCCCCCCC  LLLLLLLLLL  SSSSSSSS   IIIIII     OOOOOO    DDDDDDDD   BBBBBBBB
  CCCCCCCC  LLLLLLLLLL  SSSSSSSS   IIIIII     OOOOOO    DDDDDDDD   BBBBBBBB

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

(2) 77 DECLARATIONS
(3) 130 Local Storage for the Routine
(4) 161 CLSIODB - Entry and Initialization
(5) 441 Traversing Subroutines - System Block
(6) 556 Traversing Subroutines - Path Block
(7) 645 Traversing Subroutines - Device Data Blocks
(8) 731 Traversing Subroutines - Unit Control Block
(9) 811 Traversing Subroutines - Shared Memory Blocks
(10) 895 Counting Subroutines
(11) 1034 Utility Subroutines - SANITY_CHECK
(12) 1076 Utility Subroutines - CALL_IT
(13) 1144 Utility Subroutines - EXPREG
(14) 1198 Utility Subroutines - PARAMS_CHECK

```
0000     1          .TITLE  CLSIODB - Build a Local Copy of a Cluster I/O Database
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
0000    29  ;++
0000    30  ;
0000    31  ; FACILITY:  UETP, Regression Tests and Performance Measurement
0000    32  ;
0000    33  ; ABSTRACT:
0000    34  ;       This module steps through the I/O databases maintained by VMS to keep
0000    35  ;       track of peripherals in a cluster.  From that (potentially) changing
0000    36  ;       database in system space accessible only in exec or kernel modes,
0000    37  ;       the module builds a stable representation in P0 space accessible in
0000    38  ;       user mode.  That representation is consistent within itself; by the
0000    39  ;       time the routine which calls UETP$CLSIODB reads the representation,
0000    40  ;       it may no longer reflect the state of the cluster.
0000    41  ;
0000    42  ;
0000    43  ; ENVIRONMENT:
0000    44  ;       Runs in kernel mode.  Runs at elevated IPL and holds a mutex during
0000    45  ;       which time system services and error recovery are prohibited.  Must
0000    46  ;       prevent paging while at elevated IPL.
0000    47  ;
0000    48  ;
0000    49  ;--
0000    50  ;
0000    51  ; AUTHOR:  Richard N. Holstein, CREATION DATE:  06-Jan-1983
0000    52  ;
0000    53  ; MODIFIED BY:
0000    54  ;
0000    55  ;       V03-006 RNH0006         Richard N. Holstein,    02-May-1984
0000    56  ;               Fix bug in V3-5 causing exception above ASTDEL.
0000    57  ;
```

```
0000     58 ;       V03-005 RNH0005         Richard N. Holstein,    30-Mar-1984
0000     59 ;               Ignore UDAs when collecting system blocks.
0000     60 ;
0000     61 ;       V03-004 RNH0004         Richard N. Holstein,    22-Dec-1983
0000     62 ;               Return PB$W_STATE, PB$B_RSTATE, PB$B_CBL_STS, PB$B_P0_STS
0000     63 ;               and PB$B_P1_STS.
0000     64 ;
0000     65 ;       V03-003 RNH0003         Richard N. Holstein,    15-Jul-1983
0000     66 ;               Return UCB$L_DEVCHAR and UCB$L_DEVCHAR2.
0000     67 ;
0000     68 ;       V03-002 RNH0002         Richard N. Holstein,    23-Jun-1983
0000     69 ;               Fix bugs with word length arithmetic on addresses.  Fix bug
0000     70 ;               with allocating sufficient memory.
0000     71 ;
0000     72 ;       V03-001 RNH0001         Richard N. Holstein,    14-Jan-1983
0000     73 ;               Fix access violations.
0000     74 ;
0000     75 ;**
```

```
                        0000     77          .SBTTL  DECLARATIONS
                        0000     78 ;
                        0000     79 ; INCLUDE FILES:
                        0000     80 ;
                        0000     81 ;       SYS$LIBRARY:LIB.MLB             ; For varoius macro definitions
                        0000     82 ;       SYS$LIBRARY:UETP.MLB            ; For UETP-specific definitions
                        0000     83 ;       SYS$SYSTEM:SYS.STB              ; Required during linking
                        0000     84
                        0000     85 ;
                        0000     86 ; MACROS:
                        0000     87 ;
                        0000     88          .ENABLE SUPPRESSION
                        0000     89
                        0000     90          $ADPDEF                         ; Adapter control block
                        0000     91          $DDBDEF                         ; Device data block
                        0000     92          $IPLDEF                         ; Interrupt priority levels
                        0000     93          $MPMDEF                         ; Mulitport memory
                        0000     94          ;$PAPDTDEF                      ; CI-specific extensions to the PDT
                        0000     95          ;$PDTDEF                        ; Port descriptor table
                        0000     96          $PBDEF                          ; Path block offsets
                        0000     97          $PRDEF                          ; Processor register addresses
                        0000     98          $PSLDEF                         ; Processor status longword bits
                        0000     99          $SBDEF                          ; System block offsets
                        0000    100          $SHBDEF                         ; Shared memory control block
                        0000    101          $SHDDEF                         ; Shared memory datapage
                        0000    102          $SHRDEF                         ; Shared messages
                        0000    103          $SSDEF                          ; System Service condition codes
                        0000    104          $STSDEF                         ; Status return values
                        0000    105          $UCBDEF                         ; Unit control block
                        0000    106          $UETIDBDEF                      ; UETP I/O database definitions
                        0000    107          $UETPDEF                        ; UETP
                        0000    108
                        0000    109 ;
                        0000    110 ; EQUATED SYMBOLS:
                        0000    111 ;
00000004                0000    112          CLS_POINTER = 4                 ; Offset on calling argument list of...
                        0000    113                                          ; ...the address to which we return...
                        0000    114                                          ; ...a pointer to our cluster database
00000008                0000    115          LCL_POINTER = 8                 ; Offset on calling argument list of...
                        0000    116                                          ; ...the address to which we return...
                        0000    117                                          ; ...a pointer to our local database
0000000C                0000    118          MPM_POINTER = 12                ; Offset on calling argument list of...
                        0000    119                                          ; ...the address to which we return...
                        0000    120                                          ; ...a pointer to our shared memory...
                        0000    121                                          ; ... database
00000010                0000    122          FLAGS = 16                      ; Offset on calling argument list of...
                        0000    123                                          ; ...flags governing our execution.
                        0000    124                                          ; See $UETIDBDEF for definitions
                        0000    125
                        0000    126          UETP$_NOTCMPLT = -              ; Status return...
                        0000    127            UETP$_FACILITY@STS$V_FAC_NO!- ; ...if we run out of P0 space...
007411C2                0000    128            SHR$_NOTCMPLT!STS$K_ERROR     ; ...before we run out of database
```

```
                     0000   130          .SBTTL  Local Storage for the Routine
                     0000   131 ;
                     0000   132 ; For generality's sake, use the stack for local storage.  We use an absolute
                     0000   133 ; .PSECT to define offsets into the stack.  What appear to be labels below
                     0000   134 ; are really offsets.  Because we are also using the stack for subroutine
                     0000   135 ; calls, we must explicitly pass a reference if we want to access any of these
                     0000   136 ; locations.
                     0000   137 ;
                     0000   138          .PSECT  $ABS$,ABS
00000000             0000   139          . = 0                           ; Reset us to define offsets
                     0000   140
                     0000   141 SAVE_R0:                                 ; Temporary R0 storage
00000004             0000   142          .BLKL   1
                     0004   143
                     0004   144 SETSFM:                                  ; If SETSFM then SS fail mode...
00000005             0004   145          .BLKB   1                       ; ...was set on entry to the routine
00000008             0005   146          .BLKB   3                       ; Keep this longword aligned
                     0008   147
                     0008   148 CURRENT_AREA:                            ; Points to start of current data area.
00000010             0008   149          .BLKQ   1                       ; Quadword so it can accept the...
                     0010   150                                          ; ...result of a memory mgt. service
                     0010   151
                     0010   152 DB_COUNT:                               ; Count of the amount of space we'll...
00000014             0010   153          .BLKL   1                       ; ...need to store a particular P0 db
                     0014   154
                     0014   155 OUR_FLAGS:                              ; Convenient storage of flags...
00000018             0014   156          .BLKL   1                       ; ...governing the info we're to return
                     0018   157
00000018             0018   158 LOCAL_LENGTH = .                        ; Length of stack to reserve...
                     0018   159                                          ; ...for local storage
```

```
                        .SBTTL  CLSIODB - Entry and Initialization

                        .DEFAULT DISPLACEMENT,WORD
00000000                .PSECT  _UETP$CODE,EXE,NOWRT,PIC,SHR

                ;++
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This module steps through the I/O databases maintained by VMS to keep
                ;       track of peripherals in a cluster.  From that (potentially) changing
                ;       database in system space accessible only in exec or kernel modes,
                ;       the module builds a stable representation in P0 space accessible in
                ;       user mode.  That representation is consistent within itself; by the
                ;       time the routine which calls UETP$CLSIODB reads the representation,
                ;       it may no longer reflect the state of the cluster.
                ;
                ;       The representation built consists of seven kinds of records:  a system
                ;       record, a path record, a DDB record, a UCB record, a shared memory
                ;       record, an end record and a null record.  The first five correspond
                ;       to similar items in the VMS I/O database.  The end record gives a
                ;       convenient way to end if one traverses the data structure sequentially
                ;       or if there are no records in a group (groups are defined below).
                ;       The null record is available to allow some housekeeping functions.  It
                ;       typically will not be present.  See the $UETIDBDEF macro for record
                ;       definitions.
                ;
                ;       Those records are organized into three groups, corresponding to the
                ;       way VMS organizes its internal database.  The first group has the
                ;       cluster database:  system blocks, path blocks and devices accessible
                ;       only via SCS protocol.  The second group has the local peripherals
                ;       on a system, devices that can be directly allocated.  The third
                ;       group has shared memory devices.  An end record terminates each group.
                ;
                ;       A formal parameter is allocated to return pointers for each group.
                ;       Another parameter allows the omission of entire groups or of items
                ;       within a group.  The parameter descriptions below give more detail.
                ;
                ;       The database may be read two ways.  One may read through each group
                ;       of records sequentially.  In that case, each group looks like a tree
                ;       stretched out:  a branch followed by all its subbranches, followed
                ;       by the next branch.  If a record points to any other dependent
                ;       records, those records will follow it in sequence.  For example, a
                ;       DDB record will be followed by all of its UCB records.  The address
                ;       of the next record is the sum of the address of the current record
                ;       and the contents of the length field of the current record.  One may
                ;       also selectively go through the database records one wants.  In that
                ;       case, one chains through record types and to their subtypes as one
                ;       would access the VMS databases.
                ;
                ;       This section of the code receives control from the calling program.
                ;       It initializes our data structures and starts the various routines
                ;       which build up the P0 databases.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       NOTE THAT CLSIODB IS TYPICALLY NOT CALLED DIRECTLY!  RATHER IT IS
```

```
0000    218 ;       CALLED BY A CHANGE MODE TO KERNEL SYSTEM SERVICE!
0000    219 ;
0000    220 ;               ret_status.wlc.v = SYS$CMKRNL (UETP$CLSIODB,arglst.rz.ra)
0000    221 ;                       WHERE arglst CONTAINS:
0000    222 ;                       count-of-longwords-to-follow.rl.v
0000    223 ;                       cls-pointer.wa.ra
0000    224 ;                       lcl-pointer.wa.ra
0000    225 ;                       mpm-pointer.wa.ra
0000    226 ;                       flags.rlu.v
0000    227 ;
0000    228 ;       OR IF ALREADY IN KERNEL MODE:
0000    229 ;
0000    230 ;               ret_status.wlc.v = UETP$CLSIODB ([cls-pointer.wa.ra],-
0000    231 ;                       [lcl-pointer.wa.ra],[mpm-pointer.wa.ra][,flags.rlu.v])
0000    232 ;
0000    233 ; INPUT PARAMETERS:
0000    234 ;
0000    235 ;       flags
0000    236 ;               If argument missing or if present and bits are set, return
0000    237 ;               record type assigned to each bit.  If bit is clear, do not
0000    238 ;               return record type corresponding to that bit.  Bits are defined
0000    239 ;               in $UETIDBDEF.  The bits are not all independent; see the
0000    240 ;               PARAMS_CHECK subroutine.  Optional argument.
0000    241 ;
0000    242 ; IMPLICIT INPUTS:
0000    243 ;
0000    244 ;       VMS's internal database of cluster peripherals.
0000    245 ;
0000    246 ; OUTPUT PARAMETERS:
0000    247 ;
0000    248 ;       cls-pointer
0000    249 ;               If reference is present, a 2-longword array to receive
0000    250 ;               starting and ending addresses of the P0 database that we
0000    251 ;               build for cluster items.  Required argument, but may be
0000    252 ;               specified as 0 if information is not required.
0000    253 ;
0000    254 ;       lcl-pointer
0000    255 ;               If reference is present, a 2-longword array to receive
0000    256 ;               starting and ending addresses of the P0 database that we
0000    257 ;               build for local peripherals.  Required argument, but may be
0000    258 ;               specified as 0 if information is not required.
0000    259 ;
0000    260 ;       mpm-pointer
0000    261 ;               If reference is present, a 2-longword array to receive
0000    262 ;               starting and ending addresses of the P0 database that we
0000    263 ;               build for shared memories.  Required argument, but may be
0000    264 ;               specified as 0 if information is not required.
0000    265 ;
0000    266 ; IMPLICIT OUTPUTS:
0000    267 ;
0000    268 ;       The database in P0 space described in the Functional Description,
0000    269 ;       above.
0000    270 ;
0000    271 ; COMPLETION CODES:
0000    272 ;
0000    273 ;       SS$_NORMAL
0000    274 ;               We were able to build a P0 database.
```

```
                                  0000  ;
                                  0000  ;       UETP$_NOTCMPLT
                                  0000  ;               We though enough space was allocated to copy the database, but
                                  0000  ;               we ran out of room anyway.
                                  0000  ;
                                  0000  ;       VMS error code otherwise.
                                  0000  ;
                                  0000  ; SIDE EFFECTS:
                                  0000  ;
                                  0000  ;       P0 space is expanded dynamically.  Note that it is possible for a
                                  0000  ;               process to release this area (or areas) once the process is
                                  0000  ;               finished with it (them).
                                  0000  ;
                                  0000  ;--
                                  0000
                            0FFC  0000          .ENTRY  UETP$CLSIODB,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                  0002
                  5E    18   C2   0002          SUBL2   #LOCAL_LENGTH,SP          ; Allocate space for our local storage
                  59    5E   D0   0005          MOVL    SP,R9                     ; Keep a copy so we can still use stack
 69   18   00   00 8F    00   2C   0008          MOVC5   #0,#0,#0,#LOCAL_LENGTH,(R9) ; Clear out our local storage
                  58   14 A9   DE   000F          MOVAL   OUR_FLAGS(R9),R8          ; Set up defaults and see if...
                       0598   30   0013          BSBW    PARAMS_CHECK              ; ...UETP$CLSIODB was correctly called
                                  0016          $SETSFM_S ENBFLG = #0             ; Clear SS fail mode locally
                  50    09   D1   001F          CMPL    #SS$_WASSET,R0            ; Was it in effect previously?
                        03   12   0022          BNEQ    10$                       ; BR if it was not
                     04 A9   96   0024          INCB    SETSFM(R9)                ; Flag the fact if it was
                                  0027  10$:
                  56   08 A9   DE   0027          MOVAL   CURRENT_AREA(R9),R6       ; R6 retains this address
                  57   10 A9   DE   002B          MOVAL   DB_COUNT(R9),R7           ; R7 retains this address
                     CE AF   DE   002F          MOVAL   UETP$CLSIODB,-            ; We must lock ourself in our WS...
                     08 A9        0032                  CURRENT_AREA(R9)
                  0655'CF    DE   0034          MOVAL   UETP$CLSIODB_END,-        ; ...because...
                     0C A9        0038                  CURRENT_AREA+4(R9)        ; ...we won't be able to page...
                                  003A          $LKWSET_S INADR = (R6)            ; ...once we're at elevated IPL
                                  0047
                     04 AC   D5   0047          TSTL    CLS_POINTER(AP)           ; Need we return shape of the cluster?
                        03   12   004A          BNEQ    20$                       ; BR if not
                      0084   31   004C          BRW     80$
                                  004F  20$:    SETIPL  #IPL$_SCS                 ; Raise IPL to keep a constant database
                  67    07   D0   0052          MOVL    #UIDEND$K_FFREE,(R7)      ; Initialize counter to minimum space
 50   00000000'GF    DE   0055          MOVAL   G^SCS$GQ_CONFIG,R0        ; Get the start of the cluster db...
                  51    60   D0   005C          MOVL    SB$L_FLINK(R0),R1         ; ...and point to the first real SB
                        05   E0   005F          BBS     #UIDFLAG$V_MYSYS,-        ; BR if we want to see the list of...
                03 14 A9        0061                  OUR_FLAGS(R9),30$         ; ...devices on the system calling us
                  51    61   D0   0064          MOVL    SB$L_FLINK(R1),R1         ; Otherwise, check other nodes only
                                  0067  30$:
                        50   DD   0067          PUSHL   R0                        ; Start out counting the cluster db
                     14 A9   DD   0069          PUSHL   OUR_FLAGS(R9)
                        57   DD   006C          PUSHL   R7
                  0487'CF    DF   006E          PUSHAL  SID_COUNT
                        51   DD   0072          PUSHL   R1
            056F'CF    05   FB   0074          CALLS   #5,CALL_IT
                                  0079          SETIPL  #0                        ; Reset IPL
                     03 50   E8   007C          BLBS    R0,40$                    ; BR if everything went OK
                      0153   31   007F          BRW     200$                      ; BR - we failed to count cluster db
                        67   DD   0082  40$:    PUSHL   (R7)                      ; Calculate, allocate and lock in WS...
                     04 AC   DD   0084          PUSHL   CLS_POINTER(AP)           ; ...the space we'll use for...
```

```
      0578'CF    02   FB  0087          CALLS   #2,EXPREG                ; ...the cluster database
            03 50    E8  008C          BLBS    R0,50$                   ; BR if there were no problems
             0143    31  008F          BRW     200$                     ; We had problems - bail out
                         0092   50$:    SETIPL  #IPL$_SCS                ; Raise IPL to keep a constant database
 50   00000000'GF    DE  0095          MOVAL   G^SCS$GQ_CONFIG,R0       ; Get the start of the cluster db...
          51    60   D0  009C          MOVL    SB$L_FLINK(R0),R1        ; ...and point to the first real SB
                05   E0  009F          BBS     #UIDFLAG$V_MYSYS,-       ; BR if we want to see the list of...
          03 14 A9       00A1                  OUR_FLAGS(R9),60$        ; ...devices on the system calling us
          51    61   D0  00A4          MOVL    SB$L_FLINK(R1),R1        ; Otherwise, check other nodes only
                         00A7   60$:
                50   DD  00A7          PUSHL   R0                       ; Copy the cluster database
             14 A9   DD  00A9          PUSHL   OUR_FLAGS(R9)
                00   DD  00AC          PUSHL   #0
          0215'CF    DF  00AE          PUSHAL  SID_TRAVERSE
                51   DD  00B2          PUSHL   R1
      056F'CF    05   FB  00B4          CALLS   #5,CALL_IT
                         00B9          SETIPL  #0                       ; Reset IPL
            03 50    E8  00BC          BLBS    R0,70$                   ; BR if everything went OK
             0113    31  00BF          BRW     200$                     ; BR - we failed to get cluster db
                         00C2   70$:    $ULWSET_S INADR = (R6)           ; Allow that area to page again
                06   D0  00CF          MOVL    #UID$K_END_RTYPE,-       ; Put a cap on what we return
             06 AB       00D1                  UIDGNRC$B_TYPE(R11)
                         00D3
                         00D3   80$:
             08 AC    D5  00D3          TSTL    LCL_POINTER(AP)          ; Need we return local peripherals?
                03   12  00D6          BNEQ    90$                      ; BR if not
             009B    31  00D8          BRW     120$                     ; BR to shared memory
 54   00000000'GF    D0  00DB   90$:    MOVL    G^SCH$GL_CURPCB,R4       ; We want the mutex on the local I/O db
      00000000'GF    16  00E2          JSB     G^SCH$IOLOCKR            ; Grab the mutex, goto IPL$_ASTDEL
          67    07   D0  00E8          MOVL    #UIDEND$K_FFREE,(R7)     ; Initialize counter to minimum space
                00   DD  00EB          PUSHL   #0                       ; Count items in the local I/O database
             14 A9   DD  00ED          PUSHL   OUR_FLAGS(R9)
                57   DD  00F0          PUSHL   R7
          04EB'CF    DF  00F2          PUSHAL  DDB_COUNT
      00000000'GF    DD  00F6          PUSHL   G^IOC$GL_DEVLIST
      056F'CF    05   FB  00FC          CALLS   #5,CALL_IT
                01   BB  0101          PUSHR   #^M<R0>                  ; Save status of counting routines
 54   00000000'GF    D0  0103          MOVL    G^SCH$GL_CURPCB,R4       ; We must release mutex on local I/O db
      00000000'GF    16  010A          JSB     G^SCH$IOUNLOCK           ; Release mutex...
                         0110          SETIPL  #0                       ; ...and reset IPL
                01   BA  0113          POPR    #^M<R0>
            03 50    E8  0115          BLBS    R0,100$                  ; BR if we got local peripheral db
             00BA    31  0118          BRW     200$                     ; We failed to get local db
                67   DD  011B   100$:   PUSHL   (R7)                     ; Calculate, allocate and lock in WS...
             08 AC   DD  011D          PUSHL   LCL_POINTER(AP)          ; ...the space we'll use for...
      0578'CF    02   FB  0120          CALLS   #2,EXPREG                ; ...the local database
            03 50    E8  0125          BLBS    R0,110$                  ; BR if there was no problem
             00AA    31  0128          BRW     200$                     ; There was some problem
 54   00000000'GF    D0  012B   110$:   MOVL    G^SCH$GL_CURPCB,R4       ; We want the mutex on the local I/O db
      00000000'GF    16  0132          JSB     G^SCH$IOLOCKR            ; Grab the mutex, goto IPL$_ASTDEL
                00   DD  0138          PUSHL   #0                       ; Copy the local I/O database
             14 A9   DD  013A          PUSHL   OUR_FLAGS(R9)
                00   DD  013D          PUSHL   #0
          033E'CF    DF  013F          PUSHAL  DDB_TRAVERSE
      00000000'GF    DD  0143          PUSHL   G^IOC$GL_DEVLIST
      056F'CF    05   FB  0149          CALLS   #5,CALL_IT
                01   BB  014E          PUSHR   #^M<R0>                  ; Save status of traversing routines
```

```
54   00000000'GF   D0  0150          MOVL    G^SCH$GL_CURPCB,R4        ; We must release mutex on local I/O db
     00000000'GF   16  0157          JSB     G^SCH$IOUNLOCK            ; Release mutex...
                       015D          SETIPL  #0                        ; ...and reset IPL
              01   BA  0160          POPR    #^M<R0>
           70 50   E9  0162          BLBC    R0,200$                   ; BR if we failed to get local db
                       0165          $ULWSET_S INADR = (R6)            ; Allow normal paging in that area
              06   D0  0172          MOVL    #UID$K_END_RTYPE,-        ; Put a cap on what we return
           06 AB       0174                  UIDGNRC$B_TYPE(R11)
                       0176
                       0176  120$:
           0C AC   D5  0176          TSTL    MPM_POINTER(AP)           ; Need we return shared memory info?
              5A   13  0179          BEQL    200$                      ; BR if not
        67    07   D0  017B          MOVL    #UIDEND$K_FFREE,(R7)      ; Initialize counter to minimum space
              00   DD  017E          PUSHL   #0                        ; Now count the shared memory list
           14 A9   DD  0180          PUSHL   OUR_FLAGS(R9)
              57   DD  0183          PUSHL   R7
         0526'CF   DF  0185          PUSHAL  MPM_COUNT
     00000000'GF   DD  0189          PUSHL   G^EXE$GL_SHBLIST
    056F'CF    05   FB  018F          CALLS   #5,CALL_IT
           3E 50   E9  0194          BLBC    R0,200$                   ; BR if we failed to count it
              67   DD  0197          PUSHL   (R7)                      ; Calculate, allocate and lock in WS...
           0C AC   DD  0199          PUSHL   MPM_POINTER(AP)           ; ...the space we'll use for...
    0578'CF    02   FB  019C          CALLS   #2,EXPREG                 ; ...the shared memory database
           31 50   E9  01A1          BLBC    R0,200$                   ; BR if there was any problem
              00   DD  01A4          PUSHL   #0                        ; Copy the shared memory database
           14 A9   DD  01A6          PUSHL   OUR_FLAGS(R9)
              00   DD  01A9          PUSHL   #0
         0417'CF   DF  01AB          PUSHAL  MPM_TRAVERSE
     00000000'GF   DD  01AF          PUSHL   G^EXE$GL_SHBLIST
    056F'CF    05   FB  01B5          CALLS   #5,CALL_IT
           18 50   E9  01BA          BLBC    R0,200$                   ; BR if we failed to copy it
                       01BD          $ULWSET_S INADR = (R6)            ; Allow that area to page again
              06   D0  01CA          MOVL    #UID$K_END_RTYPE,-        ; Put a cap on what we return
           06 AB       01CC                  UIDGNRC$B_TYPE(R11)
                       01CE
50   007480C9 8F   D0  01CE          MOVL    #UETP$_NORMAL,R0          ; Indicate that all went well
                       01D5  200$:
        69    50   D0  01D5          MOVL    R0,SAVE_R0(R9)            ; Save our return status over SS calls
                       01D8          $ULWSET_S INADR = (R6)            ; Allow paging of db in case error path
        FE17 CF    DE  01E5          MOVAL   UETP$CLSIODB,-            ; We can unlock our code, too
           08 A9       01E9                  CURRENT_AREA(R9)
         0655'CF   DE  01EB          MOVAL   UETP$CLSIODB_END,-
           0C A9       01EF                  CURRENT_AREA+4(R9)
                       01F1          $ULWSET_S INADR = (R6)            ; Allow regular paging - we're IPL 0
     58    04 A9   D0  01FE          MOVL    SETSFM(R9),R8
                       0202          $SETSFM_S ENBFLG = R8             ; Reset original SS fail mode
        50    69   D0  020B          MOVL    SAVE_R0(R9),R0            ; Restore our return status
           03 50   E9  020E          BLBC    R0,210$                   ; Exit now if there was an error
        50    01   D0  0211          MOVL    #SS$_NORMAL,R0            ; Otherwise, give status people expect
                       0214  210$:
                   04  0214          RET
```

```
                                                L 13
0215   441          .SBTTL  Traversing Subroutines - System Block
0215   442
0215   443 ;++
0215   444 ;
0215   445 ; FUNCTIONAL DESCRIPTION:
0215   446 ;       Extracts information from a system block in VMS's cluster
0215   447 ;       database and puts it into a SID record in P0 space.
0215   448 ;
0215   449 ; CALLING SEQUENCE:
0215   450 ;       BSBW    SID_TRAVERSE    ; Always called by CALL_IT,
0215   451 ;                               ; ...uses CALL_IT's params
0215   452 ;
0215   453 ; INPUT PARAMETERS:
0215   454 ;       Same as those for CALL_IT.
0215   455 ;       12(AP) is 0 or the address of a longword which will link to the current
0215   456 ;              record.
0215   457 ;
0215   458 ; IMPLICIT INPUTS:
0215   459 ;       R10 and R11 form a descriptor for the amount of free space left in the
0215   460 ;              current area of P1 space we've allocated for our database.
0215   461 ;
0215   462 ; OUTPUT PARAMETERS:
0215   463 ;       12(AP) - Updated for link to subtree in current record.
0215   464 ;       04(AP) - Updated to point to the next SID block if there is one.  If
0215   465 ;              there is none, undefined.
0215   466 ;
0215   467 ; IMPLICIT OUTPUTS:
0215   468 ;       R10 and R11 are updated to point to the next free byte in P0 space.
0215   469 ;
0215   470 ; COMPLETION CODES:
0215   471 ;       UETP$_NORMAL if we were able to extract useful information and there
0215   472 ;              is presumed to be another SID block to go.
0215   473 ;       UETP$_FINISHED if we've reached the last SID or there are none.
0215   474 ;       UETP$_NOTCMPLT if we run out of room to copy the database.
0215   475 ;
0215   476 ; SIDE EFFECTS:
0215   477 ;       NONE
0215   478 ;
0215   479 ;--
0215   480
0215   481 SID_TRAVERSE:
0215   482
58   04 AC    D0  0215   483          MOVL    04(AP),R8               ; Get address of presumed system block
         09    13  0219   484          BEQL    10$                     ; BR if obviously bogus
      68 58    D1  021B   485          CMPL    R8,(R8)                 ; Check for an empty circular list
         04    13  021E   486          BEQL    10$                     ; BR if we have one
58   14 AC    D1  0220   487          CMPL    20(AP),R8               ; Check for pointing to list header
                   0224   488 ;        BEQL    10$                     ; BR if we are
                   0224   489 ;        BBS     #UIDFLAG$V_SID,16(AP),10$ ; We know caller wants SID records
                   0224   490 ;        CLRL    R8                      ; Force bogus address otherwise
                   0224   491 10$:
      033D         30  0224   492          BSBW    SANITY_CHECK            ; See if we have a good address
   01    2F        EB  0227   493          FFC     #<SB$S_SYSTEMID*8>-1,#1,- ; A UDA is indicated by the high...
50   18 A8              022A   494                  SB$B_SYSTEMID(R8),R0    ; ...bit in the system id...
         03    12  022D   495          BNEQ    15$
       008A    31  022F   496          BRW     40$                     ; ...and we want to ignore them
5A 00000045 8F    D1  0232   497 15$:  CMPL    #UIDSID$K_FFREE,R10     ; Have we enough space for this record?
```

```
                08   15  0239          BLEQ    20$                          ; BR if we have
50   007411C2 8F     D0  023B          MOVL    #UETP$_NOTCMPLT,R0           ; Return with an error if not
                     05  0242          RSB
                         0243  20$:
          0045 8F    B0  0243          MOVW    #UIDSID$K_FFREE,-            ; Save length of record
            04 AB        0247                  UIDGNRC$W_SIZE(R11)
                01   90  0249          MOVB    #UID$K_SID_RTYPE,-           ; Indicate that it's a SID record
            06 AB        024B                  UIDGNRC$B_TYPE(R11)
            0C AC    D5  024D          TSTL    12(AP)                       ; Are we chained to a previous record?
                04   13  0250          BEQL    30$                          ; BR if not
      0C BC     5B   D0  0252          MOVL    R11,@12(AP)                  ; Forge a link if we are
                         0256  30$:
                06   28  0256          MOVC3   #SB$S_SYSTEMID,-             ; Transfer...
            18 A8        0258                  SB$B_SYSTEMID(R8),-          ; ...the system id...
            0B AB        025A                  UIDSID$B_SYSTEMID(R11)       ; ...to our SID record in P0 space
            24 A8    D0  025C          MOVL    SB$T_SWTYPE(R8),-            ; Transfer the software type...
            11 AB        025F                  UIDSID$T_SWTYPE(R11)
            28 A8    D0  0261          MOVL    SB$T_SWVERS(R8),-            ; ...the software version...
            15 AB        0264                  UIDSID$T_SWVERS(R11)
            2C A8    7D  0266          MOVQ    SB$Q_SWINCARN(R8),-          ; ...the software incarnation number...
            19 AB        0269                  UIDSID$Q_SWINCARN(R11)
            34 A8    D0  026B          MOVL    SB$T_HWTYPE(R8),-            ; ...the hardware type...
            21 AB        026E                  UIDSID$T_HWTYPE(R11)
                0C   28  0270          MOVC3   #SB$S_HWVERS,-               ; ...the hardware version...
            38 A8        0272                  SB$B_HWVERS(R8),-
            25 AB        0274                  UIDSID$B_HWVERS(R11)
                10   28  0276          MOVC3   #SB$S_NODENAME,-             ; ...and the SCS nodename...
            44 A8        0278                  SB$T_NODENAME(R8),-
            31 AB        027A                  UIDSID$T_NODENAME(R11)       ; ...to our SID record
                         027C
          59     5B   D0  027C          MOVL    R11,R9                       ; Make a temporary copy of our base...
      50     04 A9   3C  027F          MOVZWL  UIDGNRC$W_SIZE(R9),R0        ; ...
          5A     50   C2  0283          SUBL2   R0,R10                       ; ...because we and routines we call...
          5B     50   C0  0286          ADDL2   R0,R11                       ; ...update the free space descriptor
                         0289
                00   DD  0289          PUSHL   #0                           ; DDB list terminates by itself
            10 AC    DD  028B          PUSHL   16(AP)                       ; Form a copy of flags for subtree
            41 A9    DF  028E          PUSHAL  UIDSID$L_DDB(R9)             ; Save DDB link for CALL_IT call
          033E'CF    DF  0291          PUSHAL  DDB_TRAVERSE                 ; Now set up all DDBs and UCBs...
            54 A8    DD  0295          PUSHL   SB$L_DDB(R8)                 ; ...attached to this system block
    056F'CF     05   FB  0298          CALLS   #5,CALL_IT
            27 50    E9  029D          BLBC    R0,50$                       ; Exit immediately on an error
                         02A0
            0C A8    DD  02A0          PUSHL   SB$L_PBFL(R8)                ; Path blocks are in a circular list
            10 AC    DD  02A3          PUSHL   16(AP)                       ; Form a copy of flags for subtree
            07 A9    DF  02A6          PUSHAL  UIDSID$L_PBFL(R9)            ; Save path block link for CALL_IT call
          02C8'CF    DF  02A9          PUSHAL  PATH_TRAVERSE                ; Now set up all the path blocks...
            0C A8    DD  02AD          PUSHL   SB$L_PBFL(R8)                ; ...attached to this system block
    056F'CF     05   FB  02B0          CALLS   #5,CALL_IT
            0F 50    E9  02B5          BLBC    R0,50$                       ; Exit immediately on an error
                         02B8
      0C AC     69   DE  02B8          MOVAL   UIDGNRC$A_FLINK(R9),12(AP) ; Set up link for next SID record
                         02BC  40$:
      04 AC     68   D0  02BC          MOVL    SB$L_FLINK(R8),04(AP)        ; Point to next possible system block
50   007480C9 8F     D0  02C0          MOVL    #UETP$_NORMAL,R0             ; Return fat and happy
                         02C7  50$:
                     05  02C7          RSB
```

```
                                02C8   556          .SBTTL  Traversing Subroutines - Path Block
                                02C8   557
                                02C8   558 ;++
                                02C8   559 ;
                                02C8   560 ; FUNCTIONAL DESCRIPTION:
                                02C8   561 ;       Extracts information from a path block in VMS's cluster database and
                                02C8   562 ;       puts it into a PATH record in P0 space.
                                02C8   563 ;
                                02C8   564 ; CALLING SEQUENCE:
                                02C8   565 ;       BSBW    PATH_TRAVERSE   ; Always called by CALL_IT,
                                02C8   566 ;                               ; ...uses CALL_IT's params
                                02C8   567 ;
                                02C8   568 ; INPUT PARAMETERS:
                                02C8   569 ;       Same as those for CALL_IT.
                                02C8   570 ;       12(AP) is 0 or the address of a longword which will link to the current
                                02C8   571 ;              record.
                                02C8   572 ;
                                02C8   573 ; IMPLICIT INPUTS:
                                02C8   574 ;       R10 and R11 form a descriptor for the amount of free space left in the
                                02C8   575 ;              current area of P0 space we've allocated for our database.
                                02C8   576 ;
                                02C8   577 ; OUTPUT PARAMETERS:
                                02C8   578 ;       12(AP) - Updated for link to subtree in current record.
                                02C8   579 ;       04(AP) - Updated to point to the next path block if there is one.  If
                                02C8   580 ;              there is none, undefined.
                                02C8   581 ;
                                02C8   582 ; IMPLICIT OUTPUTS:
                                02C8   583 ;       R10 and R11 are updated to point to the next free byte in P0 space.
                                02C8   584 ;
                                02C8   585 ; COMPLETION CODES:
                                02C8   586 ;       UETP$_NORMAL if we were able to extract useful information and there
                                02C8   587 ;              is presumed to be another path block to go.
                                02C8   588 ;       UETP$_FINISHED if we've reached the last path block or there are none.
                                02C8   589 ;       UETP$_NOTCMPLT if we run out of room to copy the database.
                                02C8   590 ;
                                02C8   591 ; SIDE EFFECTS:
                                02C8   592 ;       NONE
                                02C8   593 ;
                                02C8   594 ;--
                                02C8   595
                                02C8   596 PATH_TRAVERSE:
                                02C8   597
      58    04 AC    D0         02C8   598          MOVL    04(AP),R8               ; Get address of presumed path block
                  0A 13         02CC   599          BEQL    10$                     ; BR if obviously bogus
                68    58    D1  02CE   600          CMPL    R8,(R8)                 ; Check for an empty circular list
           02 10 AC    01    E0  02D1   601          BBS     #UIDFLAG$V_PATH,16(AP),10$ ; BR if caller wants path records
                      58    D4  02D6   602          CLRL    R8                      ; Force bogus address otherwise
                                02D8   603 10$:
                  0289    30    02D8   604          BSBW    SANITY_CHECK            ; See if we have a good address
                   5A    11    D1  02DB   605          CMPL    #UIDPATH$K_FFREE,R10    ; Have we enough space for this record?
                      08 15     02DE   606          BLEQ    20$                     ; BR if we have
      50    007411C2    8F    D0  02E0   607          MOVL    #UETP$_NOTCMPLT,R0      ; Return with an error if not
                            05  02E7   608          RSB
                                02E8   609 20$:
                      11    B0  02E8   610          MOVW    #UIDPATH$K_FFREE,-      ; Save length of record
                   04 AB          02EA   611                  UIDGNRC$W_SIZE(R11)
                      02    90  02EC   612          MOVB    #UID$K_PATH_RTYPE,-     ; Indicate it's a path block record
```

```
          06 AB        02EE                    UIDGNRC$B_TYPE(R11)
          0C AC   D5   02F0          TSTL     12(AP)                    ; Are we chained to a previous record?
             04   13   02F3          BEQL     30$                       ; BR if not
     0C BC   5B   D0   02F5          MOVL     R11,@12(AP)               ; Forge a link if we are
                      02F9   30$:
          24 A8   D0   02F9          MOVL     PB$T_LPORT_NAME(R8),-     ; Copy over the local port name
          09 AB        02FC                    UIDPATH$T_LPORT_NAME(R11)
          12 A8   B0   02FE          MOVW     PB$W_STATE(R8),-          ; Copy the state of the virtual circuit
          07 AB        0301                    UIDPATH$W_STATE(R11)
          21 A8   90   0303          MOVB     PB$B_RSTATE(R8),-         ; Copy the state of the remote node
          0D AB        0306                    UIDPATH$B_RSTATE(R11)
          28 A8   90   0308          MOVB     PB$B_CBL_STS(R8),-        ; Copy the overall state of the cable
          0E AB        030B                    UIDPATH$B_CBL_STS(R11)
          29 A8   90   030D          MOVB     PB$B_P0_STS(R8),-         ; Copy the state of path A
          0F AB        0310                    UIDPATH$B_P0_STS(R11)
          2A A8   90   0312          MOVB     PB$B_P1_STS(R8),-         ; Copy the state of path B
          10 AB        0315                    UIDPATH$B_P1_STS(R11)
                       0317
        59   5B   D0   0317          MOVL     R11,R9                    ; Make a temporary copy of our base...
     50   04 A9   3C   031A          MOVZWL   UIDGNRC$W_SIZE(R9),R0     ; ...
        5A   50   C2   031E          SUBL2    R0,R10                    ; ...because we...
        5B   50   C0   0321          ADDL2    R0,R11                    ; ...update the free space descriptor
                       0324
     0C AC   69   DE   0324          MOVAL    UIDGNRC$A_FLINK(R9),12(AP) ; Set up link for next path block
     04 AC   68   D0   0328          MOVL     PB$L_FLINK(R8),04(AP)     ; Point to the next possible path block
  14 AC   04 BC   D1   032C          CMPL     @04(AP),20(AP)            ; We are in a circular queue...
             03   12   0331          BNEQ     40$                       ; ...so BR if we've not gone around...
          04 AC   D4   0333          CLRL     04(AP)                    ; ...but force an end if we have
                       0336   40$:
50   007480C9 8F   D0   0336          MOVL     #UETP$_NORMAL,R0          ; Return fat and happy
                   05   033D          RSB
```

```
                     033E   645          .SBTTL  Traversing Subroutines - Device Data Blocks
                     033E   646
                     033E   647 ;++
                     033E   648 ;
                     033E   649 ; FUNCTIONAL DESCRIPTION:
                     033E   650 ;       Extracts information from a DDB in VMS's I/O database and puts it
                     033E   651 ;       into a DDB record in P0 space.
                     033E   652 ;
                     033E   653 ; CALLING SEQUENCE:
                     033E   654 ;       BSBW    DDB_TRAVERSE    ; Always called by CALL_IT,
                     033E   655 ;                               ; ...uses CALL_IT's params
                     033E   656 ;
                     033E   657 ; INPUT PARAMETERS:
                     033E   658 ;       Same as those for CALL_IT.
                     033E   659 ;       12(AP) is 0 or the address of a longword which will link to the current
                     033E   660 ;              record.
                     033E   661 ;
                     033E   662 ; IMPLICIT INPUTS:
                     033E   663 ;       R10 and R11 form a descriptor for the amount of free space left in the
                     033E   664 ;              current area of P0 space we've allocated for our database.
                     033E   665 ;
                     033E   666 ; OUTPUT PARAMETERS:
                     033E   667 ;       12(AP) - Updated for link to subtree in current record.
                     033E   668 ;       04(AP) - Updated to point to the next DDB if there is one.  If
                     033E   669 ;              there is none, undefined.
                     033E   670 ;
                     033E   671 ; IMPLICIT OUTPUTS:
                     033E   672 ;       R10 and R11 are updated to point to the next free byte in P0 space.
                     033E   673 ;
                     033E   674 ; COMPLETION CODES:
                     033E   675 ;       UETP$_NORMAL if we were able to extract useful information and there
                     033E   676 ;                  is presumed to be another DDB to go.
                     033E   677 ;       UETP$_FINISHED if we've reached the last DDB or there are none.
                     033E   678 ;       UETP$_NOTCMPLT if we run out of room to copy the database.
                     033E   679 ;
                     033E   680 ; SIDE EFFECTS:
                     033E   681 ;       NONE
                     033E   682 ;
                     033E   683 ;--
                     033E   684
                     033E   685 DDB_TRAVERSE:
                     033E   686
     58    04 AC     D0   033E   687          MOVL    04(AP),R8                       ; Get address of the DDB
  02 10 AC    02     E0   0342   688          BBS     #UIDFLAG$V_DDB,16(AP),10$ ; BR if caller wants DDB records
                58     D4   0347   689          CLRL    R8                              ; Force bogus address otherwise
                            0349   690 10$:
           0218     30   0349   691          BSBW    SANITY_CHECK                    ; See if we have a good address
             0C     81   034C   692          ADDB3   #UIDDDB$K_FFREE,-               ; Figure how much space we'll need...
     57    14 A8          034E   693                  DDB$T_NAME(R8),R7
        57  57     9A   0351   694          MOVZBL  R7,R7                           ; ...for our P0 DDB record
        5A  57     D1   0354   695          CMPL    R7,R10                          ; Have we enough?
            08     15   0357   696          BLEQ    20$                             ; BR if we have
50   007411C2 8F     D0   0359   697          MOVL    #UETP$_NOTCMPLT,R0              ; Return with an error if not
                05     0360   698          RSB
                            0361   699 20$:
     04 AB  57     B0   0361   700          MOVW    R7,UIDGNRC$W_SIZE(R11)  ; Save length of record
                03     90   0365   701          MOVB    #UID$K_DDB_RTYPE,-      ; Indicate that it's a DDB record
```

```
          06 AB        0367                    UIDGNRC$B_TYPE(R11)
          0C AC   D5   0369          TSTL     12(AP)                    ; Are we chained to a previous record?
             04   13   036C          BEQL     30$                       ; BR if not
     0C BC    5B   D0   036E          MOVL     R11,@12(AP)               ; Forge a link if we are
                       0372   30$:
     57    14 A8   9B   0372          MOVZBW   DDB$T_NAME(R8),R7         ; Transfer...
     15 A8    57   28   0376          MOVC3    R7,DDB$T_NAME+1(R8),-     ; ...the device controller name...
          0C AB        037A                    UIDDDB$K_FFREE(R11)       ; ...to our DDB record in P0 space
     0B AB    57   90   037C          MOVB     R7,UIDDDB$T_NAME(R11)     ; Make the name an ASCIC string
                       0380
        59    5B   D0   0380          MOVL     R11,R9                    ; Make a temporary copy of our base...
     50    04 A9   3C   0383          MOVZWL   UIDGNRC$W_SIZE(R9),R0     ; ...
        5A    50   C2   0387          SUBL2    R0,R10                    ; ...because we and routines we call...
        5B    50   C0   038A          ADDL2    R0,R11                    ; ...update the free space descriptor
                       038D
             00   DD   038D          PUSHL    #0                        ; UCB list terminates by itself
          10 AC   DD   038F          PUSHL    16(AP)                    ; Form a copy of flags for subtree
          07 A9   DF   0392          PUSHAL   UIDDDB$L_UCB(R9)          ; Save UCB link for CALL_IT call
        03B4'CF   DF   0395          PUSHAL   UCB_TRAVERSE              ; Now set up all the UCBs...
          04 A8   DD   0399          PUSHL    DDB$L_UCB(R8)             ; ...attached to this DDB
   056F'CF    05   FB   039C          CALLS    #5,CALL_IT
          0F 50   E9   03A1          BLBC     R0,40$                    ; Exit immediately on an error
                       03A4
     0C AC    69   DE   03A4          MOVAL    UIDGNRC$A_FLINK(R9),12(AP) ; Set up link for next DDB
     04 AC    68   D0   03A8          MOVL     DDB$L_LINK(R8),04(AP)     ; Point to the next possible DDB
50   007480C9 8F   D0   03AC          MOVL     #UETP$_NORMAL,R0          ; Return fat and happy
                       03B3   40$:
                  05   03B3          RSB
```

```
                     731          .SBTTL  Traversing Subroutines - Unit Control Block
                     732
                     733 ;++
                     734 ;
                     735 ; FUNCTIONAL DESCRIPTION:
                     736 ;       Extracts information from a UCB in VMS's I/O database and puts it
                     737 ;       into a UCB record in P0 space.
                     738 ;
                     739 ; CALLING SEQUENCE:
                     740 ;       BSBW    UCB_TRAVERSE    ; Always called by CALL_IT,
                     741 ;                               ; ...uses CALL_IT's params
                     742 ;
                     743 ; INPUT PARAMETERS:
                     744 ;       Same as those for CALL_IT.
                     745 ;       12(AP) is 0 or the address of a longword which will link to the current
                     746 ;               record.
                     747 ;
                     748 ; IMPLICIT INPUTS:
                     749 ;       R10 and R11 form a descriptor for the amount of free space left in the
                     750 ;               current area of P0 space we've allocated for our database.
                     751 ;
                     752 ; OUTPUT PARAMETERS:
                     753 ;       04(AP) - Updated to point to the next UCB if there is one.  If
                     754 ;               there is none, undefined.
                     755 ;
                     756 ; IMPLICIT OUTPUTS:
                     757 ;       R10 and R11 are updated to point to the next free byte in P0 space.
                     758 ;
                     759 ; COMPLETION CODES:
                     760 ;       UETP$_NORMAL if we were able to extract useful information and there
                     761 ;               is presumed to be another UCB to go.
                     762 ;       UETP$_FINISHED if we've reached the last UCB or there are none.
                     763 ;       UETP$_NOTCMPLT if we run out of room to copy the database.
                     764 ;
                     765 ; SIDE EFFECTS:
                     766 ;       NONE
                     767 ;
                     768 ;--
                     769
                     770 UCB_TRAVERSE:
                     771
     58    04 AC    D0  03B4   772          MOVL    04(AP),R8                       ; Get the address of the UCB
  02 10 AC    03    E0  03B8   773          BBS     #UIDFLAG$V_UCB,16(AP),10$ ; BR if caller wants UCB records
                58    D4  03BD   774          CLRL    R8                              ; Force bogus address otherwise
                          03BF   775 10$:
          01A2      30  03BF   776          BSBW    SANITY_CHECK                    ; See if we have a good address
        5A    13    D1  03C2   777          CMPL    #UIDUCB$K_FFREE,R10             ; Have we enough space for this record?
                08    15  03C5   778          BLEQ    20$                             ; BR if we have
50   007411C2 8F    D0  03C7   779          MOVL    #UETP$_NOTCMPLT,R0              ; Return with an error if not
                05    05  03CE   780          RSB
                          03CF   781 20$:
                13    B0  03CF   782          MOVW    #UIDUCB$K_FFREE,-               ; Save length of record
                04 AB       03D1   783                  UIDGNRC$W_SIZE(R11)
                04    90  03D3   784          MOVB    #UID$K_UCB_RTYPE,-              ; Indicate that it's a UCB record
                06 AB       03D5   785                  UIDGNRC$B_TYPE(R11)
                0C AC    D5 03D7   786          TSTL    12(AP)                          ; Are we chained to a previous record?
                04    13  03DA   787          BEQL    30$                             ; BR if not
```

```
                OC BC    5B    D0  03DC   788
                                03E0   789 30$:
                      54 A8    B0  03E0   790          MOVW    UCB$W_UNIT(R8),-          ; Save the unit number...
                      07 AB        03E3   791                  UIDUCB$W_NUMBER(R11)      ; ...in our P0 UCB record
                      40 A8    90  03E5   792          MOVB    UCB$B_DEVCLASS(R8),-      ; Save the device class
                      09 AB        03E8   793                  UIDUCB$B_DEVCLASS(R11)
                      41 A8    90  03EA   794          MOVB    UCB$B_DEVTYPE(R8),-       ; Save the device type
                      0A AB        03ED   795                  UIDUCB$B_DEVTYPE(R11)
                      38 A8    D0  03EF   796          MOVL    UCB$L_DEVCHAR(R8),-       ; Save all the device...
                      0B AB        03F2   797                  UIDUCB$L_DEVCHAR(R11)
                      3C A8    D0  03F4   798          MOVL    UCB$L_DEVCHAR2(R8),-      ; ...independent characteristics
                      0F AB        03F7   799                  UIDUCB$L_DEVCHAR2(R11)
                                   03F9   800
                         59    5B  03F9   801          MOVL    R11,R9                    ; Make a temporary copy of our base...
                   50     04 A9    3C  03FC   802          MOVZWL  UIDGNRC$W_SIZE(R9),R0     ; ...
                      5A    50    C2  0400   803          SUBL2   R0,R10                    ; ...because we...
                      5B    50    C0  0403   804          ADDL2   R0,R11                    ; ...update the free space descriptor
                                   0406   805
                   OC AC    69    DE  0406   806          MOVAL   UIDGNRC$A_FLINK(R9),12(AP) ; Set up link for next UCB
                04 AC    30 A8    D0  040A   807          MOVL    UCB$L_LINK(R8),04(AP)     ; Point to the next possible UCB
             50    007480C9 8F    D0  040F   808          MOVL    #UETP$_NORMAL,R0          ; Return fat and happy
                                   05  0416   809          RSB
```

```
                     0417   811          .SBTTL  Traversing Subroutines - Shared Memory Blocks
                     0417   812
                     0417   813 ;++
                     0417   814 ;
                     0417   815 ; FUNCTIONAL DESCRIPTION:
                     0417   816 ;       Extracts information from a SHB-SHD-ADP set in VMS's I/O database and
                     0417   817 ;       puts it into an MPM record in P0 space.
                     0417   818 ;
                     0417   819 ; CALLING SEQUENCE:
                     0417   820 ;       BSBW    MPM_TRAVERSE    ; Always called by CALL_IT,
                     0417   821 ;                               ; ...uses CALL_IT's params
                     0417   822 ;
                     0417   823 ; INPUT PARAMETERS:
                     0417   824 ;       Same as those for CALL_IT.
                     0417   825 ;       12(AP) is 0 or the address of a longword which will link to the current
                     0417   826 ;              record.
                     0417   827 ;
                     0417   828 ; IMPLICIT INPUTS:
                     0417   829 ;       R10 and R11 form a descriptor for the amount of free space left in the
                     0417   830 ;              current area of P0 space we've allocated for our database.
                     0417   831 ;
                     0417   832 ; OUTPUT PARAMETERS:
                     0417   833 ;       04(AP) - Updated to point to the next SHB if there is one.  If
                     0417   834 ;              there is none, undefined.
                     0417   835 ;
                     0417   836 ; IMPLICIT OUTPUTS:
                     0417   837 ;       R10 and R11 are updated to point to the next free byte in P0 space.
                     0417   838 ;
                     0417   839 ; COMPLETION CODES:
                     0417   840 ;       UETP$_NORMAL if we were able to extract useful information and there
                     0417   841 ;              is presumed to be another SHB to go.
                     0417   842 ;       UETP$_FINISHED if we've reached the last SHB or there are none.
                     0417   843 ;       UETP$_NOTCMPLT if we run out of room to copy the database.
                     0417   844 ;
                     0417   845 ; SIDE EFFECTS:
                     0417   846 ;       NONE
                     0417   847 ;
                     0417   848 ;--
                     0417   849
                     0417   850 MPM_TRAVERSE:
                     0417   851
                     0417   852          ; R8 will point to the SHB, R7 to the SHD, and R6 to the ADP.
          58  04 AC  D0  0417   853          MOVL    04(AP),R8               ; Get address of the SHB
                     041B   854 ;         BBS     #UIDFLAG$V_MPM,16(AP),10$ ; We know caller wants MPM records
                     041B   855 ;         CLRL    R8                      ; Force bogus address otherwise
                     041B   856 10$:
              0146   30  041B   857          BSBW    SANITY_CHECK            ; See if we have a good address
          57  04 A8  D0  041E   858          MOVL    SHB$L_DATAPAGE(R8),R7   ; Point to corresponding SHD
                0A   81  0422   859          ADDB3   #UIDMPM$K_FFREE,-       ; Figure how much space we'll need...
          55  20 A7       0424   860                  SHD$T_NAME(R7),R5
             55   55  9A  0427   861          MOVZBL  R5,R5                   ; ...for our P0 MPM record
             5A   55  D1  042A   862          CMPL    R5,R10                  ; Have we enough?
                  08  15  042D   863          BLEQ    20$                     ; BR if we have
     50 007411C2  8F  D0  042F   864          MOVL    #UETP$_NOTCMPLT,R0      ; Return with an error if not
                      05  0436   865          RSB
                          0437   866 20$:
          04 AB   55  B0  0437   867          MOVW    R5,UIDGNRC$W_SIZE(R11)  ; Save length of record
```

```
                05   90  043B          MOVB    #UID$K_MPM_RTYPE,-        ; Indicate that it's an MPM record
            06 AB       043D                  UIDGNRC$B_TYPE(R11)
            0C AC   D5  043F          TSTL    12(AP)                    ; Are we chained to a previous record?
                04   13  0442          BEQL    30$                       ; BR if not
      0C BC     5B   D0  0444          MOVL    R11,@12(AP)               ; Forge a link if we are
                         0448  30$:
      56    1C A8   D0  0448          MOVL    SHB$L_ADP(R8),R6          ; Point to the ADP for this memory
         55     66   D0  044C          MOVL    ADP$L_CSR(R6),R5          ; Point to the ADP's CSR
      55    1C A5   D0  044F          MOVL    MPM$L_MR(R5),R5           ; (I/O space refernce restriction)
                0E   EF  0453          EXTZV   #MPM$V_MR_UNIT,-          ; Get the unit number of our memory
   55    55     02       0455                  #MPM$S_MR_UNIT,R5,R5
      07 AB     55   B0  0458          MOVW    R5,UIDMPM$W_NUMBER(R11)   ; Save it in our P0 MPM record
      56    20 A7   9B  045C          MOVZBW  SHD$T_NAME(R7),R6         ; Transfer...
      21 A7     56   28  0460          MOVC3   R6,SHD$T_NAME+1(R7),-     ; ...the device controller name...
            0A AB       0464                  UIDMPM$K_FFREE(R11)       ; ...to our MPM record in P0 space
      09 AB     56   90  0466          MOVB    R6,UIDMPM$T_NAME(R11)     ; Make the name an ASCIC string
                         046A
         59     5B   D0  046A          MOVL    R11,R9                    ; Make a temporary copy of our base...
      50    04 A9   3C  046D          MOVZWL  UIDGNRC$W_SIZE(R9),R0     ; ...
         5A     50   C2  0471          SUBL2   R0,R10                    ; ...because we...
         5B     50   C0  0474          ADDL2   R0,R11                    ; ...update the free space descriptor
                         0477
      0C AC     69   DE  0477          MOVAL   UIDGNRC$A_FLINK(R9),12(AP) ; Set up link for next MPM record
      04 AC     68   D0  047B          MOVL    SHB$L_LINK(R8),04(AP)     ; Point to the next possible SHB
50   007480C9   8F   D0  047F          MOVL    #UETP$_NORMAL,R0          ; Return fat and happy
                     05  0486          RSB
```

```
                     0487   895          .SBTTL  Counting Subroutines
                     0487   896
                     0487   897 ;++
                     0487   898 ;
                     0487   899 ; FUNCTIONAL DESCRIPTION:
                     0487   900 ;       Counts up the space we'll need to allocate in order to copy VMS's
                     0487   901 ;       database into P0 space.
                     0487   902 ;
                     0487   903 ; CALLING SEQUENCE:
                     0487   904 ;       BSBW    xxxx_COUNT                      ; Always called by CALL_IT...
                     0487   905 ;                                               ; ...and uses CALL_IT's parameters
                     0487   906 ;
                     0487   907 ; INPUT PARAMETERS:
                     0487   908 ;       Same as those for CALL_IT.
                     0487   909 ;       12(AP) is the address of a longword in which we accumulate a count.
                     0487   910 ;
                     0487   911 ; IMPLICIT INPUTS
                     0487   912 ;       NONE
                     0487   913 ;
                     0487   914 ; OUTPUT PARAMETERS:
                     0487   915 ;       12(AP) is incremented by the length of a record.
                     0487   916 ;
                     0487   917 ; IMPLICIT OUTPUTS:
                     0487   918 ;       NONE
                     0487   919 ;
                     0487   920 ; COMPLETION CODES:
                     0487   921 ;       UETP$_NORMAL if we counted a record.
                     0487   922 ;       UETP$_FINISHED if the address of the VMS data structure is bogus.
                     0487   923 ;
                     0487   924 ; SIDE EFFECTS:
                     0487   925 ;       Will not return to calling routine if the address is bogus.  Since
                     0487   926 ;               this routine and address checking routine are called by BSBx,
                     0487   927 ;               address checking routine can RET to return to caller's caller.
                     0487   928 ;
                     0487   929 ;--
                     0487   930
                     0487   931 SID_COUNT:
54   04 AC    D0     0487   932          MOVL    04(AP),R4                       ; Get address of presumed system block
      09     13     048B   933          BEQL    10$                             ; BR if obviously bogus
   64   54    D1     048D   934          CMPL    R4,(R4)                         ; Check for an empty circular list
      04     13     0490   935          BEQL    10$                             ; BR if we have one
54   14 AC    D1     0492   936          CMPL    20(AP),R4                       ; Check for pointing to list header
                     0496   937 ;        BEQL    10$                             ; BR if we are
                     0496   938 ;        BBS     #UIDFLAG$V_SID,16(AP),10$       ; We know caller wants SID records
                     0496   939 ;        CLRL    R4                              ; Force bogus address otherwise
                     0496   940 10$:
     00CB    30     0496   941          BSBW    SANITY_CHECK                    ; See if we have a good address
       50    7C     0499   942          CLRQ    R0                              ; Assume for now that we see a UDA
01     2F    EB     049B   943          FFC     #<SB$S_SYSTEMID*8>-1,#1,-       ; A UDA is indicated by the high...
52   18 A4          049E   944                  SB$B_SYSTEMID(R4),R2            ; ...bit in the system id...
      1B     13     04A1   945          BEQL    20$                             ; ...and we want to ignore them
50   45 8F    9A     04A3   946          MOVZBL  #UIDSID$K_FFREE,R0              ; SID block records are fixed length
51 04EB'CF    DE     04A7   947          MOVAL   DDB_COUNT,R1                    ; We count DDB space...
52   54 A4    D0     04AC   948          MOVL    SB$L_DDB(R4),R2
                     04B0   949 ;        CLRL    R5
     008C    30     04B0   950          BSBW    COMMON_COUNT
       50    D4     04B3   951          CLRL    R0                              ; ...this SID block space, once...
```

```
51   04C6'CF   DE  04B5          MOVAL   PATH_COUNT,R1              ; ...and path block space, too
  52   0C A4   D0  04BA          MOVL    SB$L_PBFL(R4),R2
                   04BE   20$:
     53   64   D0  04BE          MOVL    SB$L_FLINK(R4),R3          ; Point to the next possible sys block
     55   52   D0  04C1          MOVL    R2,R5                      ; Set up terminator for path block list
          79   11  04C4          BRB     COMMON_COUNT
                   04C6
                   04C6   PATH_COUNT:
  54   04 AC   D0  04C6          MOVL    04(AP),R4                  ; Get address of the presumed path block
          0A   13  04CA          BEQL    10$                        ; BR if obviously bogus
     64   54   D1  04CC          CMPL    R4,(R4)                    ; Check for an empty circular list
02 10 AC   01   E0  04CF          BBS     #UIDFLAG$V_PATH,16(AP),10$ ; BR if caller wants path records
          54   D4  04D4          CLRL    R4                         ; Force bogus address otherwise
                   04D6   10$:
        008B   30  04D6          BSBW    SANITY_CHECK               ; See if we have a good address
     50   11   D0  04D9          MOVL    #UIDPATH$K_FFREE,R0        ; Path block records are fixed length
          51   D4  04DC          CLRL    R1                         ; We have no subtrees to count
     53   64   D0  04DE          MOVL    PB$L_FLINK(R4),R3          ; Point to the next possible path block
  14 AC   63   D1  04E1          CMPL    (R3),20(AP)                ; We are in a circular queue...
          02   12  04E5          BNEQ    20$                        ; ...so BR if we've not gone around...
          53   D4  04E7          CLRL    R3                         ; ...but force an end if we have
                   04E9   20$:
                   04E9   ;       CLRL    R5
          54   11  04E9          BRB     COMMON_COUNT
                   04EB
                   04EB   DDB_COUNT:
  54   04 AC   D0  04EB          MOVL    04(AP),R4                  ; Get address of the VMS data structure
02 10 AC   02   E0  04EF          BBS     #UIDFLAG$V_DDB,16(AP),10$ ; BR if caller wants DDB records
          54   D4  04F4          CLRL    R4                         ; Force bogus address otherwise
                   04F6   10$:
          6C   10  04F6          BSBB    SANITY_CHECK               ; See if we have a good address
          0C   81  04F8          ADDB3   #UIDDDB$K_FFREE,-          ; DDB records have variable length...
  50   14 A4       04FA                  DDB$T_NAME(R4),R0          ; ...device names
     50   50   9A  04FD          MOVZBL  R0,R0
51   050E'CF   DE  0500          MOVAL   UCB_COUNT,R1               ; We'll have to count UCB space...
  52   04 A4   D0  0505          MOVL    DDB$L_UCB(R4),R2
     53   64   D0  0509          MOVL    DDB$L_LINK(R4),R3          ; Point to next DDB
                   050C   ;       CLRL    R5
          31   11  050C          BRB     COMMON_COUNT
                   050E
                   050E   UCB_COUNT:
  54   04 AC   D0  050E          MOVL    04(AP),R4                  ; Get address of the VMS data structure
02 10 AC   03   E0  0512          BBS     #UIDFLAG$V_UCB,16(AP),10$ ; BR if caller wants UCB records
          54   D4  0517          CLRL    R4                         ; Force bogus address otherwise
                   0519   10$:
          49   10  0519          BSBB    SANITY_CHECK               ; See if we have a good address
     50   13   D0  051B          MOVL    #UIDUCB$K_FFREE,R0         ; UCB records are fixed length
          51   D4  051E          CLRL    R1                         ; We have no subtrees to count
  53   30 A4   D0  0520          MOVL    UCB$L_LINK(R4),R3          ; Point to next UCB
                   0524   ;       CLRL    R5
          19   11  0524          BRB     COMMON_COUNT
                   0526
                   0526   MPM_COUNT:
  54   04 AC   D0  0526          MOVL    04(AP),R4                  ; Get address of the VMS data structure
                   052A   ;       BBS     #UIDFLAG$V_MPM,16(AP),10$ ; We know caller wants MPM records
                   052A   ;       CLRL    R4                         ; Force bogus address otherwise
                   052A   10$:
```

```
                38   10  052A  1009                BSBB    SANITY_CHECK                    ; See if we have a good address
          51    04 A4   D0  052C  1010                MOVL    SHB$L_DATAPAGE(R4),R1           ; Point to corresponding SHD
                0A   81   0530  1011                ADDB3   #UIDMPM$K_FFREE,-               ; MPM records have variable length...
          50    20 A1         0532  1012                SHD$T_NAME(R1),R0               ; ...memory names
             50    50   9A   0535  1013                MOVZBL  R0,R0
                51    D4   0538  1014                CLRL    R1                              ; We have no subtrees to count
             53    64   D0   053A  1015                MOVL    SHB$L_LINK(R4),R3               ; Point to next SHB
                         053D  1016 ;              CLRL    R5
                00   11   053D  1017                BRB     COMMON_COUNT
                         053F  1018
                         053F  1019 COMMON_COUNT:
          0C BC    50   C0   053F  1020                ADDL2   R0,@12(AP)                      ; It's good, add in space we'll need
                51    D5   0543  1021                TSTL    R1                              ; Have we a subtree to count?
                11   13   0545  1022                BEQL    10$                             ; BR if not
                55    DD   0547  1023                PUSHL   R5                              ; Count subtree space - terminator...
             10 AC    DD   0549  1024                PUSHL   16(AP)                          ; ...flags...
             0C AC    DD   054C  1025                PUSHL   12(AP)                          ; ...current count...
                51    DD   054F  1026                PUSHL   R1                              ; ...counting routine address...
                52    DD   0551  1027                PUSHL   R2                              ; ...and first item in subtree
       056F'CF    05   FB   0553  1028                CALLS   #5,CALL_IT
                         0558  1029 10$:
          04 AC    53   D0   0558  1030                MOVL    R3,04(AP)                       ; Point to current level's next str
    50  007480C9  8F   D0   055C  1031                MOVL    #UETP$_NORMAL,R0                ; Indicate that we finished normally
                05   0563  1032                RSB
```

```
                                                L 14
0564  1034          .SBTTL  Utility Subroutines - SANITY_CHECK
0564  1035
0564  1036 ;++
0564  1037 ;
0564  1038 ; FUNCTIONAL DESCRIPTION:
0564  1039 ;       Checks that we have a reasonable address in system space.
0564  1040 ;
0564  1041 ; CALLING SEQUENCE:
0564  1042 ;       BSBW    SANITY_CHECK
0564  1043 ;
0564  1044 ; INPUT PARAMETERS:
0564  1045 ;       NONE
0564  1046 ;
0564  1047 ; IMPLICIT INPUTS
0564  1048 ;       PSL condition codes have been set by accessing the address of a
0564  1049 ;             supposed VMS data structure.
0564  1050 ;
0564  1051 ; OUTPUT PARAMETERS:
0564  1052 ;       NONE
0564  1053 ;
0564  1054 ; IMPLICIT OUTPUTS:
0564  1055 ;       NONE
0564  1056 ;
0564  1057 ; COMPLETION CODES:
0564  1058 ;       UETP$_FINISHED if the address is bogus.
0564  1059 ;       None, otherwise.
0564  1060 ;
0564  1061 ; SIDE EFFECTS:
0564  1062 ;       Will not return to calling routine if the address is bogus.  Since
0564  1063 ;             this routine is called by BSBx, does a RET to return to
0564  1064 ;             caller's caller.
0564  1065 ;
0564  1066 ;--
0564  1067
0564  1068 SANITY_CHECK:
0564  1069
                01    13  0564  1070          BEQL    10$                     ; Assume condition codes are set
                      05  0566  1071          RSB                             ; We have a reasonable address
                          0567  1072 10$:
50  007480D1    8F        D0  0567  1073          MOVL    #UETP$_FINISHED,R0      ; A bogus address - we're through
                          04  056E  1074          RET                             ; Force our caller to exit
```

```
056F  1076          .SBTTL  Utility Subroutines - CALL_IT
056F  1077
056F  1078 ;++
056F  1079 ;
056F  1080 ; FUNCTIONAL DESCRIPTION:
056F  1081 ;       Repeatedly calls one of the VMS I/O database traversal routines or
056F  1082 ;       space counting routines until that routine either finishes
056F  1083 ;       (UETP$_FINISHED) or has an error (low bit clear in R0).
056F  1084 ;
056F  1085 ;       This routine must be reentrent because it is called by various levels
056F  1086 ;       of data structure traversing routines to call the next lower level.
056F  1087 ;
056F  1088 ;       Note that CALL_IT itself only RETurns if it sees an error.  The
056F  1089 ;       routines to which we JSB will RETurn to CALL_IT's caller when they
056F  1090 ;       are finished.
056F  1091 ;
056F  1092 ; CALLING SEQUENCE:
056F  1093 ;       CALLS   #5,CALL_IT
056F  1094 ;
056F  1095 ; INPUT PARAMETERS:
056F  1096 ;       20(AP) - For circular lists, the address of the first item on the list.
056F  1097 ;                Trash, otherwise.  (A routine that counts or traverses a list
056F  1098 ;                must understand the data structure of any sublists.)
056F  1099 ;       16(AP) - Copy of the flag longword telling which record types are
056F  1100 ;                to be counted or written.
056F  1101 ;       12(AP) - P0 address of a longword which will point to the current
056F  1102 ;                record, address of a longword in which we accumulate a count
056F  1103 ;                of the space we need for records, or 0.
056F  1104 ;       08(AP) - Address of the routine we'll call to do the traversal or
056F  1105 ;                counting.
056F  1106 ;       04(AP) - Pointer to first data structure on a list in system space.
056F  1107 ;
056F  1108 ;       NOTE THAT THESE INPUTS ARE USED BY THE TRAVERSING AND COUNTING
056F  1109 ;       SUBROUTINES AS WELL, BECAUSE THOSE ROUTINES ARE CALLED VIA JSB!
056F  1110 ;
056F  1111 ; IMPLICIT INPUTS
056F  1112 ;       NONE
056F  1113 ;
056F  1114 ; OUTPUT PARAMETERS:
056F  1115 ;       04(AP) - Updated by repeated calls to the traversing routine.  Should
056F  1116 ;                be regarded as trash upon exiting CALL_IT.
056F  1117 ;       12(AP) - Updated so that chains of records are created or a count is
056F  1118 ;                kept of the space we'll need for records.
056F  1119 ;
056F  1120 ; IMPLICIT OUTPUTS:
056F  1121 ;       P0 database is built by the routines called by this routine.
056F  1122 ;
056F  1123 ; COMPLETION CODES:
056F  1124 ;       UETP$_FINISHED if we're able to build a part of the P0 database.
056F  1125 ;       Error code from a called routine otherwise.
056F  1126 ;
056F  1127 ; SIDE EFFECTS:
056F  1128 ;       Note that R10 and R11 are in use by routines which call CALL_IT and
056F  1129 ;                by routines which CALL_IT calls.  The registers cannot be
056F  1130 ;                altered except for their assigned purpose, and they cannot
056F  1131 ;                be saved in the entry mask either, because they are updated
056F  1132 ;                by the routines which CALL_IT calls.
```

```
                     056F  1133 ;
                     056F  1134 ;--
                     056F  1135
                     056F  1136 CALL_IT:
           03FC      056F  1137          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9>
                     0571  1138
                     0571  1139 10$:
08 BC        16      0571  1140          JSB     @08(AP)                 ; Go explore a system data structure
FA 50        E8      0574  1141          BLBS    R0,10$                  ; Explore next str at this level if OK
              04     0577  1142          RET                             ; Return with status
```

```
                     0578  1144          .SBTTL  Utility Subroutines - EXPREG
                     0578  1145
                     0578  1146 ;++
                     0578  1147 ;
                     0578  1148 ; FUNCTIONAL DESCRIPTION:
                     0578  1149 ;       Sets up a new data area at the end of P0 space and a descriptor to the
                     0578  1150 ;       area.  Request twice the calculated space in case there has been a
                     0578  1151 ;       configuration change since we calculated it.  No explicit check is
                     0578  1152 ;       made for system service success or for the amount of memory we actually
                     0578  1153 ;       receive.  We assume that System Service failure mode is in effect and
                     0578  1154 ;       thus we are assured of getting what we need or a reasonable error.
                     0578  1155 ;       Note that this routine may only run at IPL 0.
                     0578  1156 ;
                     0578  1157 ; CALLING SEQUENCE:
                     0578  1158 ;       CALLS   #2,EXPREG
                     0578  1159 ;
                     0578  1160 ; INPUT PARAMETERS:
                     0578  1161 ;       04(AP) - Pointer to a quadword to get the result of $EXPREG.
                     0578  1162 ;       08(AP) - The amount of space we're told we'll need.
                     0578  1163 ;
                     0578  1164 ; IMPLICIT INPUTS
                     0578  1165 ;       NONE
                     0578  1166 ;
                     0578  1167 ; OUTPUT PARAMETERS:
                     0578  1168 ;       04(AP) - Quadword is updated with the result of $EXPREG.
                     0578  1169 ;
                     0578  1170 ; IMPLICIT OUTPUTS:
                     0578  1171 ;       R10 and R11 become a descriptor for the free space in the new area.
                     0578  1172 ;
                     0578  1173 ; COMPLETION CODES:
                     0578  1174 ;       Result of $EXPREG.
                     0578  1175 ;
                     0578  1176 ; SIDE EFFECTS:
                     0578  1177 ;       P0 space is expanded.
                     0578  1178 ;       System service error if we can't allocate space.
                     0578  1179 ;
                     0578  1180 ;--
                     0578  1181
                     0578  1182 EXPREG:
                03FC 0578  1183          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9>
                     057A  1184
5A   08 AC 00000100 8F  C1  057A  1185          ADDL3   #^X100,08(AP),R10       ; Round up estimate by half a page...
           5A      02  C4  0583  1186          MULL2   #2,R10                  ; ...so we can grab twice required space
      56   5A   F7 8F  78  0586  1187          ASHL    #-9,R10,R6              ; Convert words to pages
                     058B  1188                                               ; (Apologies to numerical analysts.)
                     058B  1189          $EXPREG_S PAGCNT = R6,-              ; Allocate some new address space
                     058B  1190                    RETADR = @04(AP),-
                     058B  1191                    ACMODE = #PSL$C_USER
           5B   04 BC  D0  059B  1192          MOVL    @04(AP),R11             ; Get the address of the new area
                     059F  1193                                               ; R10 and R11 now form a descriptor
                     059F  1194          $LKWSET_S INADR = @04(AP)            ; We can't page the area...
                     05AD  1195                                               ; ...once we elevate IPL
                04   05AD  1196          RET
```

```
                                05AE  1198          .SBTTL  Utility Subroutines - PARAMS_CHECK
                                05AE  1199
                                05AE  1200 ;++
                                05AE  1201 ;
                                05AE  1202 ; FUNCTIONAL DESCRIPTION:
                                05AE  1203 ;
                                05AE  1204 ;       See if UETP$CLSIODB was called with a legal set of parameters.  This
                                05AE  1205 ;       means that the count of parameters must be correct and that
                                05AE  1206 ;       combinations of parameters must make sense.  Fill in defaults for
                                05AE  1207 ;       those parameters which need them.
                                05AE  1208 ;
                                05AE  1209 ; CALLING SEQUENCE:
                                05AE  1210 ;
                                05AE  1211 ;       BSBW    PARAMS_CHECK
                                05AE  1212 ;
                                05AE  1213 ; INPUT PARAMETERS:
                                05AE  1214 ;
                                05AE  1215 ;       NONE
                                05AE  1216 ;
                                05AE  1217 ; IMPLICIT INPUTS:
                                05AE  1218 ;
                                05AE  1219 ;       R8 has the address of our local copy of the FLAGS parameter.
                                05AE  1220 ;
                                05AE  1221 ; OUTPUT PARAMETERS:
                                05AE  1222 ;
                                05AE  1223 ;       NONE
                                05AE  1224 ;
                                05AE  1225 ; IMPLICIT OUTPUTS:
                                05AE  1226 ;
                                05AE  1227 ;       Local copy of the FLAGS parameter is set up with user's value or a
                                05AE  1228 ;           default.
                                05AE  1229 ;
                                05AE  1230 ; COMPLETION CODES:
                                05AE  1231 ;
                                05AE  1232 ;       SS$_ACCVIO if one of the pointers cannot be written by the caller.
                                05AE  1233 ;       SS$_BADPARAM if flags or CLS_POINTER, LCL_POINTER or MPM_POINTER are
                                05AE  1234 ;           inconsistent.
                                05AE  1235 ;       SS$_INSFARG if not all pointers are either present or defaulted.
                                05AE  1236 ;       NONE if we return to UETP$CLSIODB.
                                05AE  1237 ;
                                05AE  1238 ; SIDE EFFECTS:
                                05AE  1239 ;
                                05AE  1240 ;       Returns to UETP$CLSIODB's caller if an illegal call is detected.
                                05AE  1241 ;
                                05AE  1242 ;--
                                05AE  1243
                                05AE  1244 PARAMS_CHECK:
50   00000114 8F    D0          05AE  1245          MOVL    #SS$_INSFARG,R0         ; Set up default for too few args
        6C    03    D1          05B5  1246          CMPL    #3,00(AP)               ; How many args were supplied to us?
              64    14          05B8  1247          BGTR    110$                    ; BR if too few args supplied
              63    13          05BA  1248          BEQL    200$                    ; BR if default flags - check ptrs only
     68    10 AC    D0          05BC  1249          MOVL    FLAGS(AP),(R8)          ; Check caller's flags, too, otherwise
        50    14    D0          05C0  1250          MOVL    #SS$_BADPARAM,R0        ; Set up default for a bum parameter
           04 AC    D5          05C3  1251          TSTL    CLS_POINTER(AP)         ; Check that some info will be...
              13    12          05C6  1252          BNEQ    10$                     ;  ...returned - cluster...
     52 68    01    E0          05C8  1253          BBS     #UIDFLAG$V_PATH,(R8),110$ ; (BR if path wanted but not cluster)
     4E 68    05    E0          05CC  1254          BBS     #UIDFLAG$V_MYSYS,(R8),110$ ; (BR if my own wanted but not cluster)
```

```
                08 AC    D5   05D0  1255          TSTL    LCL_POINTER(AP)          ; ...or local peripherals...
                   1E    12   05D3  1256          BNEQ    30$
                0C AC    D5   05D5  1257          TSTL    MPM_POINTER(AP)          ; ...or shared memory
                   2B    12   05D8  1258          BNEQ    50$
                         04   05DA  1259          RET                              ; Error if no info to be returned
                              05DB  1260 ; A pointer for cluster info was passed.
                              05DB  1261 10$:
                              05DB  1262          IFNOWRT #2,@CLS_POINTER(AP),-    ; BR if we won't be able to store ptrs
                              05DB  1263                  100$,#PSL$C_USER
          38 68    00   E1   05E2  1264          BBC     #UIDFLAG$V_SID,(R8),110$ ; BR if cluster but not system blocks
          04 68    03   E1   05E6  1265          BBC     #UIDFLAG$V_UCB,(R8),20$  ; If no UCBs, don't care about DDBs
          30 68    02   E1   05EA  1266          BBC     #UIDFLAG$V_DDB,(R8),110$ ; If UCBs, must have DDBs, too
                              05EE  1267 ; A pointer for cluster info was passed, we don't know for what detail.
                              05EE  1268 20$:
                08 AC    D5   05EE  1269          TSTL    LCL_POINTER(AP)          ; Must check local stuff, too?
                   0B    13   05F1  1270          BEQL    40$                      ; BR if not - check for shared memory
                              05F3  1271 ; Pointers for local and maybe cluster peripheral info were passed.
                              05F3  1272 30$:
                              05F3  1273          IFNOWRT #2,@LCL_POINTER(AP),-    ; BR if we won't be able to store ptrs
                              05F3  1274                  100$,#PSL$C_USER
          20 68    02   E1   05FA  1275          BBC     #UIDFLAG$V_DDB,(R8),110$ ; DDBs must be set for local info
                              05FE  1276 40$:
                0C AC    D5   05FE  1277          TSTL    MPM_POINTER(AP)          ; Shared memory info as well?
                   07    12   0601  1278          BNEQ    60$                      ; BR if so
                   11    11   0603  1279          BRB     70$                      ; Check that really not wanted
                              0605  1280 ; Only a pointer for shared memory was passed.
                              0605  1281 50$:
                68    2F    D3   0605  1282          BITL    #UIDFLAG$M_SID!-         ; Only shared memory stuff wanted...
                              0608  1283                  UIDFLAG$M_MYSYS!-        ; ...
                              0608  1284                  UIDFLAG$M_PATH!-         ; ...so none of the flags...
                              0608  1285                  UIDFLAG$M_DDB!-          ; ...for other info...
                              0608  1286                  UIDFLAG$M_UCB,(R8)       ; ...may be set
                   14    12   0608  1287          BNEQ    110$                     ; BR if one or more were
                              060A  1288 ; A pointer for shared memory was passed, and maybe others as well.
                              060A  1289 60$:
                              060A  1290          IFNOWRT #2,@MPM_POINTER(AP),-    ; BR if we won't be able to store ptrs
                              060A  1291                  100$,#PSL$C_USER
          09 68    04   E1   0611  1292          BBC     #UIDFLAG$V_MPM,(R8),110$ ; The shared mem stuff must be wanted
                         05   0615  1293          RSB                              ; Passed all consistency checks
                              0616  1294 ; Either cluster or local peripheral pointers were passed, but not sh. memory.
                              0616  1295 70$:
          04 68    04   E0   0616  1296          BBS     #UIDFLAG$V_MPM,(R8),110$ ; Can't be set if no sh. mem stuff
                         05   061A  1297          RSB                              ; Passed all consistency checks
                              061B  1298 100$:
                50    0C    D0   061B  1299          MOVL    #SS$_ACCVIO,R0           ; Argument can't be written by caller
                              061E  1300 110$:
                         04   061E  1301          RET                              ; Miscellaneous consistency problems
                              061F  1302
                              061F  1303 ; Flags argument was defaulted - we must check only pointers.
                              061F  1304 200$:
                68    00    D2   061F  1305          MCOML   #0,(R8)                  ; Default flags - all are set
                50    14    D0   0622  1306          MOVL    #SS$_BADPARAM,R0         ; Set up default for a bum parameter
                   04 AC    D5   0625  1307          TSTL    CLS_POINTER(AP)          ; Check that some info will be...
                   0B    12   0628  1308          BNEQ    210$                     ; ...returned - cluster...
                   08 AC    D5   062A  1309          TSTL    LCL_POINTER(AP)          ; ...or local peripherals...
                   12    12   062D  1310          BNEQ    220$
                   0C AC    D5   062F  1311          TSTL    MPM_POINTER(AP)          ; ...or shared memory
```

```
                     19   12  0632  1312          BNEQ    230$
                          04  0634  1313          RET                                     ; Error if no info to be returned
                              0635  1314 210$:
                              0635  1315          IFNOWRT #2,@CLS_POINTER(AP),-           ; BR if we won't be able to store ptrs
                              0635  1316                  100$,#PSL$C_USER
                  08 AC   D5  063C  1317          TSTL    LCL_POINTER(AP)                 ; Do we want local peripherals, too?
                     00   13  063F  1318          BEQL    220$                            ; BR if not - check shared memory
                              0641  1319 220$:
                              0641  1320          IFNOWRT #2,@LCL_POINTER(AP),-           ; BR if we won't be able to store ptrs
                              0641  1321                  100$,#PSL$C_USER
                  0C AC   D5  0648  1322          TSTL    MPM_POINTER(AP)                 ; Do we want shared memory as well?
                     07   13  064B  1323          BEQL    240$                            ; BR if not
                              064D  1324 230$:
                              064D  1325          IFNOWRT #2,@MPM_POINTER(AP),-           ; BR if we won't be able to store ptrs
                              064D  1326                  100$,#PSL$C_USER
                              0654  1327 240$:
                          05  0654  1328          RSB                                     ; Passed all consistency checks
                              0655  1329
                              0655  1330 UETP$CLSIODB_END:                               ; End of routine for $LKWSET
                              0655  1331          .END
```

```
$$T1                    = 00000000
ADP$L_CSR               = 00000000
CALL_IT                   0000056F R      02
CLS_POINTER             = 00000004
COMMON_COUNT              0000053F R      02
CURRENT_AREA              00000008
DB_COUNT                  00000010
DDB$L_LINK              = 00000000
DDB$L_UCB               = 00000004
DDB$T_NAME              = 00000014
DDB_COUNT                 000004EB R      02
DDB_TRAVERSE              0000033E R      02
EXE$GL_SHBLIST            ******** X      02
EXPREG                    00000578 R      02
FLAGS                   = 00000010
IOC$GL_DEVLIST            ******** X      02
IPL$_SCS                = 00000008
LCL_POINTER             = 00000008
LOCAL_LENGTH            = 00000018
MPM$L_MR                = 0000001C
MPM$S_MR_UNIT           = 00000002
MPM$V_MR_UNIT           = 0000000E
MPM_COUNT                 00000526 R      02
MPM_POINTER             = 0000000C
MPM_TRAVERSE              00000417 R      02
OUR_FLAGS                 00000014
PARAMS_CHECK              000005AE R      02
PATH_COUNT                000004C6 R      02
PATH_TRAVERSE             000002C8 R      02
PB$B_CBL_STS            = 00000028
PB$B_PO_STS             = 00000029
PB$B_P1_STS             = 0000002A
PB$B_RSTATE             = 00000021
PB$L_FLINK              = 00000000
PB$T_LPORT_NAME         = 00000024
PB$W_STATE              = 00000012
PR$_IPL                 = 00000012
PSL$C_USER              = 00000003
SANITY_CHECK              00000564 R      02
SAVE_R0                   00000000
SB$B_HWVERS             = 00000038
SB$B_SYSTEMID           = 00000018
SB$L_DDB                = 00000054
SB$L_FLINK              = 00000000
SB$L_PBFL               = 0000000C
SB$Q_SWINCARN           = 0000002C
SB$S_HWVERS             = 0000000C
SB$S_NODENAME           = 00000010
SB$S_SYSTEMID           = 00000006
SB$T_HWTYPE             = 00000034
SB$T_NODENAME           = 00000044
SB$T_SWTYPE             = 00000024
SB$T_SWVERS             = 00000028
SCH$GL_CURPCB             ******** X      02
SCH$IOLOCKR               ******** X      02
SCH$IOUNLOCK              ******** X      02
SCS$GQ_CONFIG             ******** X      02
SETSFM                    00000004
SHB$L_ADP               = 0000001C
SHB$L_DATAPAGE          = 00000004
SHB$L_LINK              = 00000000
SHD$T_NAME              = 00000020
SHR$_NOTCMPLT           = 000011C0
SID_COUNT                 00000487 R      02
SID_TRAVERSE              00000215 R      02
SS$_ACCVIO              = 0000000C
SS$_BADPARAM            = 00000014
SS$_INSFARG             = 00000114
SS$_NORMAL              = 00000001
SS$_WASSET              = 00000009
STS$K_ERROR             = 00000002
STS$V_FAC_NO            = 00000010
SYS$EXPREG                ******** GX     02
SYS$LKWSET                ******** GX     02
SYS$SETSFM                ******** GX     02
SYS$ULWSET                ******** GX     02
UCB$B_DEVCLASS          = 00000040
UCB$B_DEVTYPE           = 00000041
UCB$L_DEVCHAR           = 00000038
UCB$L_DEVCHAR2          = 0000003C
UCB$L_LINK              = 00000030
UCB$W_UNIT              = 00000054
UCB_COUNT                 0000050E R      02
UCB_TRAVERSE              000003B4 R      02
UETP$CLSIODB              00000000 RG     02
UETP$CLSIODB_END          00000655 R      02
UETP$_FACILITY          = 00000074
UETP$_FINISHED          = 007480D1
UETP$_NORMAL            = 007480C9
UETP$_NOTCMPLT          = 007411C2
UID$K_DDB_RTYPE         = 00000003
UID$K_END_RTYPE         = 00000006
UID$K_MPM_RTYPE         = 00000005
UID$K_PATH_RTYPE        = 00000002
UID$K_SID_RTYPE         = 00000001
UID$K_UCB_RTYPE         = 00000004
UIDDDB$K_FFREE          = 0000000C
UIDDDB$L_UCB            = 00000007
UIDDDB$T_NAME           = 0000000B
UIDEND$K_FFREE          = 00000007
UIDFLAG$M_DDB           = 00000004
UIDFLAG$M_MYSYS         = 00000020
UIDFLAG$M_PATH          = 00000002
UIDFLAG$M_SID           = 00000001
UIDFLAG$M_UCB           = 00000008
UIDFLAG$V_DDB           = 00000002
UIDFLAG$V_MPM           = 00000004
UIDFLAG$V_MYSYS         = 00000005
UIDFLAG$V_PATH          = 00000001
UIDFLAG$V_SID           = 00000000
UIDFLAG$V_UCB           = 00000003
UIDGNRC$A_FLINK         = 00000000
UIDGNRC$B_TYPE          = 00000006
UIDGNRC$W_SIZE          = 00000004
```

```
UIDMPM$K_FFREE                  = 0000000A
UIDMPM$T_NAME                   = 00000009
UIDMPM$W_NUMBER                 = 00000007
UIDPATH$B_CBL_STS               = 0000000E
UIDPATH$B_PO_STS                = 0000000F
UIDPATH$B_P1_STS                = 00000010
UIDPATH$B_RSTATE                = 0000000D
UIDPATH$K_FFREE                 = 00000011
UIDPATH$T_LPORT_NAME            = 00000009
UIDPATH$W_STATE                 = 00000007
UIDSID$B_AWVERS                 = 00000025
UIDSID$B_SYSTEMID               = 0000000B
UIDSID$K_FFREE                  = 00000045
UIDSID$L_DDB                    = 00000041
UIDSID$L_PBFL                   = 00000007
UIDSID$Q_SWINCARN               = 00000019
UIDSID$T_HWTYPE                 = 00000021
UIDSID$T_NODENAME               = 00000031
UIDSID$T_SWTYPE                 = 00000011
UIDSID$T_SWVERS                 = 00000015
UIDUCB$B_DEVCLASS               = 00000009
UIDUCB$B_DEVTYPE                = 0000000A
UIDUCB$K_FFREE                  = 00000013
UIDUCB$L_DEVCHAR                = 0000000B
UIDUCB$L_DEVCHAR2               = 0000000F
UIDUCB$W_NUMBER                 = 00000007
```

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	00000018 (24.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_UETP$CODE	00000655 (1621.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.09	00:00:00.42
Command processing	132	00:00:00.72	00:00:02.87
Pass 1	436	00:00:16.23	00:00:34.72
Symbol table sort	0	00:00:02.31	00:00:04.38
Pass 2	248	00:00:04.10	00:00:08.73
Symbol table output	18	00:00:00.15	00:00:00.29
Psect synopsis output	1	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	873	00:00:23.62	00:00:51.43

```
The working set limit was 1950 pages.
94086 bytes (184 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1505 non-local and 58 local symbols.
1331 source lines were read in Pass 1, producing 20 object records in Pass 2.
```

31 pages of virtual memory were used to define 30 macros.

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

Macro library name	Macros defined
_255DUA28:[SHRLIB]UETP.MLB;1	2
_255DUA28:[SYS.OBJ]LIB.MLB;1	11
_255DUA28:[SYSLIB]STARLET.MLB;2	14
TOTALS (all libraries)	27

1647 GETS were required to define 27 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:CLSIODB/OBJ=OBJ$:CLSIODB MSRC$:CLSIODB/UPDATE=(ENH$:CLSIODB)+EXECML$/LIB+SHRLIB$:UETP/LIB

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SATSUT10
MAP

SATSUT12
MAP

UETINIT00
MAP

SATSUT14
MAP

UETMA7800
MAP

SATSUT01
MAP

SATSUT05
MAP

SATSUT07
MAP

UETDR7800
MAP

UETSSMM00
MAP

SATSUT09
MAP

UETLOCK00
MAP

SATSUT11
MAP

SATSSF01
LIS

SATSUT13
MAP

UETCLIG00
MAP

SATSUT04
MAP

UETINIT01
MAP

SATSUT06
MAP

UETMEMY01
MAP

SATSUT08
MAP

CLSIODB
LIS