```
   CCCCCCCCCCC  LLL              IIIIIIIII     UUU         UUU  TTTTTTTTTTTTTTT  LLL
   CCCCCCCCCCC  LLL              IIIIIIIII     UUU         UUU  TTTTTTTTTTTTTTT  LLL
   CCCCCCCCCCC  LLL              IIIIIIIII     UUU         UUU  TTTTTTTTTTTTTTT  LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
CCC             LLL                 III      UUU         UUU        TTT        LLL
   CCCCCCCCCCC  LLLLLLLLLLLLLLL     IIIIIIIII     UUUUUUUUUUUUUUU        TTT        LLLLLLLLLLLLLLL
   CCCCCCCCCCC  LLLLLLLLLLLLLLL     IIIIIIIII     UUUUUUUUUUUUUUU        TTT        LLLLLLLLLLLLLLL
   CCCCCCCCCCC  LLLLLLLLLLLLLLL     IIIIIIIII     UUUUUUUUUUUUUUU        TTT        LLLLLLLLLLLLLLL
```

FILEID**SHOMEMORY

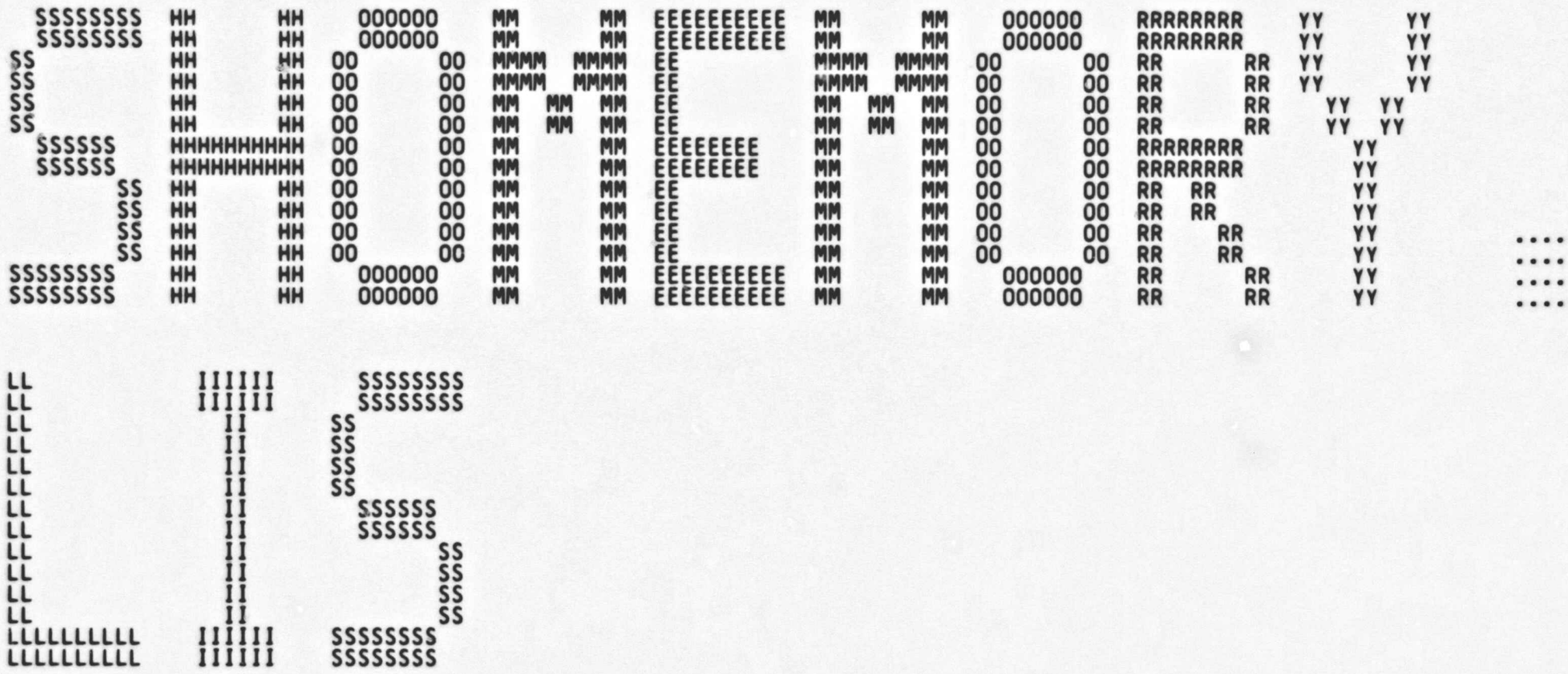

Table of contents

```
0000     1          .TITLE  SHOW$MEMORY - SHOW MEMORY RESOURCES
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;*****************************************************************************
0000     5  ;*                                                                           *
0000     6  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8  ;*  ALL RIGHTS RESERVED.                                                     *
0000     9  ;*                                                                           *
0000    10  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    11  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    12  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    13  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    14  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    15  ;*  TRANSFERRED.                                                             *
0000    16  ;*                                                                           *
0000    17  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    18  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    19  ;*  CORPORATION.                                                             *
0000    20  ;*                                                                           *
0000    21  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    22  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23  ;*                                                                           *
0000    24  ;*                                                                           *
0000    25  ;*****************************************************************************
0000    26
0000    27  ;++
0000    28  ; FACILITY:     SHOW COMMAND
0000    29  ;
0000    30  ; ABSTRACT:
0000    31  ;
0000    32  ;       This image implements the SHOW MEMORY command option.
0000    33  ;
0000    34  ; ENVIRONMENT:
0000    35  ;
0000    36  ;       Runs in User, Exec and Kernel mode. Raises IPL to ASTDEL and MAILBOX.
0000    37  ;       Holds PGDYNMTX Mutex to collect paged pool statistics.
0000    38  ;       Holds I/O Data Base Mutex to determine paging device.
0000    39  ;
0000    40  ; AUTHOR : Thomas S. Clark, Creation Date: 30-Jul-1980
0000    41  ;
0000    42  ; MODIFIED BY:
0000    43  ;
0000    44  ;       V03-010 AEW0002         Anne E. Warner          24-Jul-1984
0000    45  ;               Change 'packet size/upper bound' to be 'LRP+80' instead
0000    46  ;               'LRP+64' for the display of Large Packet (LRP) Lookaside
0000    47  ;               List for the command SHOW MEMORY/POOL/FULL.
0000    48  ;
0000    49  ;       V03-009 AEW0001         Anne E. Warner          24-May-1984
0000    50  ;               Change call to SCAN_BAD_LIST to a $CMEXEC call to
0000    51  ;               stop the program from access violating when bad pages
0000    52  ;               are found.
0000    53  ;
0000    54  ;       V03-008 KPL0001         Peter Lieberwirth       5-Mar-1984
0000    55  ;               Change use of CONFREG to CONFREGL.  Missed this reference in
0000    56  ;               first pass.
0000    57  ;
```

```
0000     58 ;
0000     59 ;       V03-007 SOP0001         J. R. Sopka             14 October 1983
0000     60 ;               Replace hand-crafted paging device name string extracted
0000     61 ;               from UCB & DDB, with string returned by IOC$CVT_DEVNAM.
0000     62 ;               Few other minor cleanup modifications also made.
0000     63 ;
0000     64 ;       V03-006 TCM0002         Trudy C. Matthews       13-Apr-1983
0000     65 ;               Preserve R2 across call to SCAN_BAD_PAGES in bad pages memory
0000     66 ;               display.  Change default displacement length from ^W to ^L.
0000     67 ;
0000     68 ;       V03-005 TCM0001         Trudy C. Matthews       22-Feb-1983
0000     69 ;               Add "number of pages discarded during bootstrap memory test"
0000     70 ;               to bad memory pages display.
0000     71 ;
0000     72 ;       V03-004 GAS0099         Gerry Smith             7-Jan-1983
0000     73 ;               Modify to run with new SHOW.
0000     74 ;
0000     75 ;       V03-003 JWH0117         Jeffrey W. Horn         19-Nov-1982
0000     76 ;               Make SHOW PROCESS/MEMORY reflect that the size of the
0000     77 ;               Process Allocation Region is now controlable via a
0000     78 ;               SYSGEN parameter.
0000     79 ;
0000     80 ;       V03-002 KDM0002         Kathleen D. Morse       28-Jun-1982
0000     81 ;               Added $IPLDEF, $SSDEF, and $PRDEF.
0000     82 ;--
```

```
                        .SBTTL  DECLARATIONS

;
; INCLUDE FILES:
;

                        .nocross
                        $DDBDEF                         ;DDB DEFINITIONS
                        $DVIDEF                         ;$GETDVI REQUEST CODES
                        $FCBDEF                         ;FCB DEFINITIONS
                        $IPLDEF                         ;IPL DEFINITIONS
                        $IRPDEF                         ;IRP DEFINITIONS
                        $JPIDEF                         ;$GETJPI REQUEST CODES
                        $NDTDEF                         ;ADAPTER TYPE CODES
                        $PCBDEF                         ;PROC CTL BLK DEFINITIONS
                        $PFLDEF                         ;PAGING FILE DEFINITIONS
                        $PFNDEF                         ;PFN DATABASE DEFINITIONS
                        $PRDEF                          ;PROCESSOR REGISTER NUMBERS
                        $RPBDEF                         ;RESTART PARAMETER BLOCK DEFS
                        $SSDEF                          ;SYSTEM STATUS CODES
                        $UCBDEF                         ;UCB DEFINITIONS
                        $WCBDEF                         ;WCB DEFINITIONS
                        .cross

;
; MACROS:
;

;
; MACRO TO CALL SHOW$PRINT_MSG TO TYPE A LINE(S)
;
                        .MACRO  TYPEMSG MESSAGEID,ARGLIST
                                .IF     B,ARGLIST
                                PUSHL   #0
                                .IFF
                                PUSHAL  G^ARGLIST
                                .ENDC
                                PUSHAL  MESSAGEID
                                CALLS   #2,G^SHOW$WRITE_LINE
                        .ENDM   TYPEMSG

;
; EQUATED SYMBOLS:
;
;       LENGTHS FOR PAGING AND SWAP FILE NAMES
;

00000028                SHOW$C_MEM_SHORT_NAME == 40     ; 40 characters for single-line display
0000004E                SHOW$C_MEM_LONG_NAME  == 78     ; 78 characters for full display


00000001                EVENT_FLAG = 1                  ; Event flag for $GETJPI use

;
; BIT FIELD DEFINITIONS FOR QUALIFIER PRESENCE LONGWORD
;
```

```
                                                E 3
                     0000   141          _VIELD  MEMORY,0,<-
                     0000   142                  <PHYS,,M>,-             ; /PHYSICAL_MEMORY
                     0000   143                  <SLOT,,M>,-             ; /SLOTS
                     0000   144                  <POOL,,M>,-             ; /POOL
                     0000   145                  <FILE,,M>,-             ; /FILES
                     0000   146                  <FULL,,M>,-             ; /FULL
                     0000   147                  <ALL,,M>,-              ; /ALL
                     0000   148                  >
                     0000   149
                     0000   150 ; Define offset into argument list for kernel mode procedure that
                     0000   151 ; scans fixed-size (lookaside) lists.
                     0000   152
00000004             0000   153          XRPFL = 4
                     0000   154
                     0000   155 ; Define offsets into extended PFL control structure that exists for
                     0000   156 ; each paging or swap file currently installed.
                     0000   157
                     0000   158          $DEFINI PFL
                     0000   159
00000024             0000   160          . = PFL$K_LENGTH
                     0024   161
                     0024   162          $DEF    PFL_W_PFL_INDEX         ; PFL index
00000026             0024   163                  .BLKW   1
                     0026   164
                     0026   165          $DEF    PFL_W_FID               ; File ID
                     0026   166          $DEF    PFL_W_FID_NUM           ; File ID - file number
00000028             0026   167                  .BLKW   1
                     0028   168          $DEF    PFL_W_FID_SEQ           ; File ID - sequence number
0000002A             0028   169                  .BLKW   1
                     002A   170          $DEF    PFL_W_FID_RVN           ; File ID - relative volume number
0000002C             002A   171                  .BLKW   1
                     002C   172
00000018             002C   173          PFL_S_DEVNAM = DDB$S_NAME + 8   ; Allow room for 5-digit unit number
                     002C   174
                     002C   175          $DEF    PFL_T_DEVNAM            ; Space for .ASCIC device name
00000044             002C   176                  .BLKB   PFL_S_DEVNAM
                     0044   177
00000044             0044   178          PFL_K_EXT_LENGTH = .            ; Define length of extended PFL
                     0044   179
                     0044   180          $DEFEND
                     0000   181
                     0000   182 ;
                     0000   183 ; OWN STORAGE:
                     0000   184 ;
                 00000000   185          .PSECT  SHOW$RODATA     LONG,RD,NOWRT,NOEXE
                     0000   186 ;
                     0000   187 ; Define CLI qualifier descriptors
                     0000   188 ;
                     0000   189 MEMORY_D_PHYS:
43 49 53 59 48 50 00000008'010E0000'  0000   190          .ASCID  /PHYSICAL_MEMORY/
      59 52 4F 4D 45 4D 5F 4C 41      000E
                     0017   191 MEMORY_D_SLOTS:
   53 54 4F 4C 53 0000001F'010E0000'  0017   192          .ASCID  /SLOTS/
                     0024   193 MEMORY_D_POOL:
      4C 4F 4F 50 0000002C'010E0000'  0024   194          .ASCID  /POOL/
                     0030   195 MEMORY_D_FILES:
   53 45 4C 49 46 00000038'010E0000'  0030   196          .ASCID  /FILES/
```

```
                         003D   197 MEMORY_D_FULL:
4C 4C 55 46 00000045'010E0000' 003D   198         .ASCID  /FULL/
                         0049   199 MEMORY_D_ALL:
   4C 4C 41 00000051'010E0000' 0049   200         .ASCID  /ALL/
                         0054   201
                     00000000   202         .PSECT  SHOW$RWDATA     LONG,RD,WRT,NOEXE
                         0000   203         .ALIGN  LONG                    ; LOCATION COUNTER BACK TO LONGWORD
                         0000   204
                         0000   205 LOCKED_CODE_RANGE:                     ; Range of code that executes
                  0000059B' 0000   206         .ADDRESS        BEGIN_LOCKED_CODE       ; above ASTDEL
                  000008AD' 0004   207         .ADDRESS        END_LOCKED_CODE - 1
                         0008   208
                         0008   209 MEMORY_L_BITLIS:
                  00000000  0008   210         .LONG   0                       ; QUALIFIER BIT LIST
                         000C   211
                         000C   212 HEADER_LIST:
            00000000 00000000   000C   213         .LONG   0,0                     ; TIME/DATE TO FORCE CURRENT TIME/DATE
                         0014   214
                         0014   215 ;
                         0014   216 ;       MEMORY FAO ARGUMENT LIST
                         0014   217 ;
                         0014   218
                         0014   219 SHOW_MEM_PHY:
                         0014   220 MEM_MB_1:
                  00000018  0014   221         .BLKL   1                       ; SPACE FOR PHYSICAL COUNT IN MB (INTEGER)
                  0000003C' 0018   222         .LONG   MEM_MB_DESC             ; DESCRIPTOR FOR FRACTIONAL MB COUNT
                         001C   223 MEM_PHY_PAGES:
                  00000020  001C   224         .BLKL   1                       ; SPACE FOR COUNT OF PHYSICAL PAGES
                         0020   225 MEM_FREE_PAGES:
                  00000024  0020   226         .BLKL   1                       ; SPACE FOR COUNT OF FREE PAGES
                         0024   227 MEM_USED_PAGES:
                  00000028  0024   228         .BLKL   1                       ; SPACE FOR COUNT OF PAGES IN USE
                         0028   229 MEM_MODF_PAGES:
                  0000002C  0028   230         .BLKL   1                       ; SPACE FOR COUNT OF MODIFIED PAGES
                         002C   231
                         002C   232 MEM_BAD_LIST:
                  00000030  002C   233         .BLKL   1                       ; SPACE FOR SIZE OF BAD PAGE LIST
                         0030   234 MEM_BAD_PAGES:
                  00000034  0030   235         .BLKL   1                       ; SPACE FOR COUNT OF BAD PAGES
                         0034   236 MEM_OTHER_PAGES:
                  00000038  0034   237         .BLKL   1                       ; COUNT OF OTHER PAGES ON BAD PAGE LIST
                         0038   238 MEM_BOOT_PAGES:
                  0000003C  0038   239         .BLKL   1                       ; PAGES DISCARDED DURING BOOTSTRAP
                         003C   240
                         003C   241 MEM_MB_DESC:
                  00000002  003C   242         .LONG   2                       ; DESCRIPTOR FOR FRACTIONAL PART
                  00000044  0040   243         .BLKL   1                       ; OF COUNT IN MB
                         0044   244 MEM_MB_TEXT:
20 20 30 35 20 20 35 32 20 20 30 30  0044   245         .ASCII  /00  25  50  75  /      ; FRACTIONS
                  20 20 35 37  0050
                         0054   246
                         0054   247 LOCAL_MEMORY:
                  00000058  0054   248         .BLKL   1                       ; TOTAL AMOUNT OF LOCAL MEMORY
                         0058   249 SHARED_MEMORY:
                  0000005C  0058   250         .BLKL   1                       ; TOTAL AMOUNT OF MULTIPORT MEMORY
                         005C   251
                         005C   252 ;
```

```
                005C   253 ;       LAST PARAGRAPH FAO ARGUMENT LISTS
                005C   254 ;
                005C   255
                005C   256 PARA_VMS:
00000060        005C   257         .BLKL   1                        ; SPACE FOR SIZE OF VMS
                0060   258
                0060   259 ;
                0060   260 ;       SLOT FAO ARGUMENT LIST
                0060   261 ;
                0060   262
                0060   263 SHOW_SLOTS_LIST:
                0060   264 SLOTS_TOTAL:
00000064        0060   265         .BLKL   1                        ; SPACE FOR TOTAL # OF SLOTS
                0064   266 SLOTS_FREE:
00000068        0064   267         .BLKL   1                        ; SPACE FOR # OF FREE SLOTS
                0068   268 SLOTS_RES:
0000006C        0068   269         .BLKL   1                        ; SPACE FOR # OF RESIDENT SLOTS
                006C   270 SLOTS_NONRES:
00000070        006C   271         .BLKL   1                        ; SPACE FOR # OF "NON-RESIDENT" SLOTS
                0070   272
                0070   273 ; FAO argument list for variable sized pool displays
                0070   274
                0070   275 SHOW_POOL_LIST:
                0070   276 POOL_NAME:
00000074        0070   277         .BLKL   1                        ; ADDRESS OF STRING DESCRIPTOR OF AREA
                0074   278 SHOW_POOL_LIST2:
                0074   279 POOL_SIZE:
00000078        0074   280         .BLKL   1                        ; ADDRESS OF DESCRIPTOR OF SIZE PARAMETER
                0078   281 SHOW_POOL_LIST3:
                0078   282 SHOW_POOL_LIST4:
                0078   283 POOL_TOTAL:
0000007C        0078   284         .BLKL   1                        ; SPACE FOR TOTAL SIZE OF POOL IN BYTES
                007C   285 POOL_TOTAL_PAGES:
00000080        007C   286         .BLKL   1                        ; SPACE FOR TOTAL SIZE OF POOL IN PAGES
                0080   287 SHOW_POOL_LIST5:
                0080   288 POOL_FREE:
00000084        0080   289         .BLKL   1                        ; SPACE FOR FREE BYTES IN POOL
                0084   290 POOL_INUSE:
00000088        0084   291         .BLKL   1                        ; SPACE FOR BYTES IN USE IN POOL
                0088   292 SHOW_POOL_LIST6:
                0088   293 POOL_MAX_BLOCK:
0000008C        0088   294         .BLKL   1                        ; SIZE OF LARGEST BLOCK IN POOL
                008C   295 POOL_MIN_BLOCK:
00000090        008C   296         .BLKL   1                        ; SIZE OF SMALLEST BLOCK IN POOL
                0090   297 SHOW_POOL_LIST7:
                0090   298 POOL_FREE_COUNT:
00000094        0090   299         .BLKL   1                        ; COUNT OF NUMBER OF HOLES IN POOL
                0094   300 POOL_FREE_LEQU_32:
00000098        0094   301         .BLKL   1                        ; COUNT OF HOLES 32 BYTES OR SMALLER
                0098   302
                0098   303 ; FAO parameter list for fixed-size (lookaside) list displays
                0098   304
                0098   305 SHOW_LOOK_LIST:
                0098   306 SHOW_LOOK_LIST3:
                0098   307 SHOW_LOOK_LIST4:
                0098   308 LOOK_LIST_NAME:
0000009C        0098   309         .BLKL   1                        ; Descriptor for name of lookaside list
```

```
                                                     H 3

                     009C    310 SHOW_LOOK_LIST2:
                     009C    311 LOOK_LIST_SIZE:
000000A8             009C    312         .BLKL   3                       ; Size of list in packets, bytes, pages
                     00A8    313 SHOW_LOOK_LIST5:
                     00A8    314 LOOK_FREE_COUNT:
000000AC             00A8    315         .BLKL   1                       ; Number of free packets
                     00AC    316 LOOK_FREE_BYTES:
000000B0             00AC    317         .BLKL   1                       ; Number of free bytes
                     00B0    318 SHOW_LOOK_LIST6:
                     00B0    319 LOOK_INUSE_COUNT:
000000B4             00B0    320         .BLKL   1                       ; Number of packets being used
                     00B4    321 LOOK_INUSE_BYTES:
000000B8             00B4    322         .BLKL   1                       ; Number of bytes in use
                     00B8    323 SHOW_LOOK_LIST7:
                     00B8    324 LOOK_SIZE_DESC:
000000BC             00B8    325         .BLKL   1                       ; Descriptor of parameter for block size
                     00BC    326 LOOK_BLOCK_SIZE:
000000C0             00BC    327         .BLKL   1                       ; Size of blocks in list
                     00C0    328 SHOW_LOOK_LIST8:
                     00C0    329 LOOK_BLOCK_MIN:
000000C4             00C0    330         .BLKL   1                       ; Lower limit on blocks allocated
                     00C4    331                                         ;  from this list
                     00C4    332 LOOK_CMKRNL_ARGLIST:
00000001             00C4    333         .LONG   1                       ; A single parameter that contains
000000CC             00C8    334         .BLKL   1                       ;  the address of the listhead
                     00CC    335
                     00CC    336 ; The next three longwords are used to pass information related to the
                     00CC    337 ; initial and maximum sizes of each lookaside list into the common
                     00CC    338 ; output routine.
                     00CC    339
                     00CC    340 LOOK_SIZE_ARRAY:
                000000D0     00CC    341         .BLKL   1                       ; Descriptor for parameter name
                000000D4     00D0    342         .BLKL   1                       ; Initial size of list
                000000D8     00D4    343         .BLKL   1                       ; Maximum size of list
                             00D8    344
                             00D8    345 ; Text descriptors that describe each portion of dynamic memory
                             00D8    346
                        00000000     347         .PSECT  SHOW$MSG_TEXT   BYTE,RD,NOWRT,NOEXE
                             0000    348
                             0000    349 NPAGEDYN_DESC:
67 61 70 6E 6F 4E 00000008'010E0000' 0000    350         .ASCID  \Nonpaged Dynamic Memory      \
4D 20 63 69 6D 61 6E 79 44 20 64 65  000E
   20 20 20 20 20 20 79 72 6F 6D 65  001A
                             0025    351
                             0025    352 PAGEDYN_DESC:
20 64 65 67 61 50 0000002D'010E0000' 0025    353         .ASCID  \Paged Dynamic Memory         \
6F 6D 65 4D 20 63 69 6D 61 6E 79 44  0033
   20 20 20 20 20 20 20 20 20 79 72  003F
                             004A    354
                             004A    355 PRCALLREG_DESC:
73 65 63 6F 72 50 00000052'010E0000' 004A    356         .ASCID  \Process Dynamic Memory Area  \
65 4D 20 63 69 6D 61 6E 79 44 20 73  0058
   20 20 61 65 72 41 20 79 72 6F 6D  0064
                             006F    357
                             006F    358 BYTES_SIZE_DESC:
   73 65 74 79 62 00000077'010E0000' 006F    359         .ASCID  \bytes\
                             007C    360
```

```
                                                                 I  3
                                        007C   361 PAGEDYN_SIZE_DESC:
59 44 45 47 41 50 00000084'010E0000'    007C   362         .ASCID  \PAGEDYN\
                                     4E 008A
                                        008B   363
                                        008B   364 SRP_NAME_DESC:
         50 52 53 00000093'010E0000'    008B   365         .ASCID  \SRP\
                                        0096   366
                                        0096   367 SRPLIST_DESC:
20 6C 6C 61 6D 53 0000009E'010E0000'    0096   368         .ASCID  \Small Packet (SRP)\
29 50 52 53 28 20 74 65 6B 63 61 50     00A4
                                        00B0   369
                                        00B0   370 SRP_SIZE_DESC:
5A 49 53 50 52 53 000000B8'010E0000'    00B0   371         .ASCID  \SRPSIZE\
                                     45 00BE
                                        00BF   372
                                        00BF   373 IRP_NAME_DESC:
         50 52 49 000000C7'010E0000'    00BF   374         .ASCID  \IRP\
                                        00CA   375
                                        00CA   376 IRPLIST_DESC:
65 52 20 4F 2F 49 000000D2'010E0000'    00CA   377         .ASCID  \I/O Request Packet (IRP)\
74 65 6B 63 61 50 20 74 73 65 75 71     00D8
                  29 50 52 49 28 20     00E4
                                        00EA   378
                                        00EA   379 IRP_SIZE_DESC:
   64 65 78 69 66 000000F2'010E0000'    00EA   380         .ASCID  \fixed\
                                        00F7   381
                                        00F7   382 LRP_NAME_DESC:
         50 52 4C 000000FF'010E0000'    00F7   383         .ASCID  \LRP\
                                        0102   384
                                        0102   385 LRPLIST_DESC:
20 65 67 72 61 4C 0000010A'010E0000'    0102   386         .ASCID  \Large Packet (LRP)\
29 50 52 4C 28 20 74 65 6B 63 61 50     0110
                                        011C   387
                                        011C   388 LRP_SIZE_DESC:
5A 49 53 50 52 4C 00000124'010E0000'    011C   389         .ASCID  \LRPSIZE + 80\
                  30 38 20 2B 20 45     012A
                                        0130   390
                                        0130   391 ;
                                        0130   392 ; Text descriptors for the output of SHOW MEMORY
                                        0130   393 ;
                                        0130   394
                                        0130   395 SHOW$_MEM_HEAD1:
20 20 20 20 20 20 00000138'010E0000'    0130   396         .ASCID  \                System Memory Resources on !%D\
74 73 79 53 20 20 20 20 20 20 20 20     013E
65 52 20 79 72 6F 6D 65 4D 20 6D 65     014A
21 20 6E 6F 20 73 65 63 72 75 6F 73     0156
                              44 25     0162
                                        0164   397 SHOW$_MEM_MEMO1:
73 79 68 50 2F 21 0000016C'010E0000'    0164   398         .ASCID  \!/Physical Memory Usage (pages):     Total        Free       In Use
20 79 72 6F 6D 65 4D 20 6C 61 63 69     0172
73 65 67 61 70 28 20 65 67 61 73 55     017E
6C 61 74 6F 54 20 20 20 20 20 3A 29     018A
65 65 72 46 20 20 20 20 20 20 20 20     0196
65 73 55 20 6E 49 20 20 20 20 20 20     01A2
64 65 69 66 69 64 6F 4D 20 20 20 20     01AE
                                        01BA   399 SHOW$_MEM_MEMO2:
6E 69 61 4D 20 20 000001C2'010E0000'    01BA   400         .ASCID  \  Main Memory !10<(!UL.!ASMb)!>          !7UL      !7UL      !7UL
```

```
                                                                J 3
3C 30 31 21 20 79 72 6F 6D 65 4D 20  01C8
21 29 62 4D 53 41 21 2E 4C 55 21 28  01D4
37 21 20 20 20 20 20 20 20 20 3E 20  01E0
20 4C 55 37 21 20 20 20 20 20 4C 55  01EC
20 20 20 20 4C 55 37 21 20 20 20 20  01F8
                  4C 55 37 21 20 20  0204
                                     0209    401
                                     0209    402 SHOW$_MEM_MEMO3:
61 42 20 20 2F 21 00000211'010E0000' 0209    403         .ASCID  \!/  Bad Pages                          Total     Dynamic  I/O Errors
20 20 20 20 20 73 65 67 61 50 20 64  0217
20 20 20 20 20 20 20 20 20 20 20 20  0223
6C 61 74 6F 54 20 20 20 20 20 20 20  022F
63 69 6D 61 6E 79 44 20 20 20 20 20  023B
73 72 6F 72 72 45 20 4F 2F 49 20 20  0247
63 69 74 61 74 53 20 20 20 20 20 20  0253
                        2F 21        025F
20 20 20 20 20 20 20 20 20 20 20 20  0261    404                     \                                !7UL      !7UL      !7UL      !7UL
20 20 20 20 20 20 20 20 20 20 20 20  026D
55 37 21 20 20 20 20 20 20 20 20 20  0279
20 20 4C 55 37 21 20 20 20 20 20 4C  0285
20 20 20 20 20 4C 55 37 21 20 20 20  0291
                  4C 55 37 21        029D
                                     02A1    405 SHOW$_MEM_PARA1:
74 20 66 4F 2F 21 000002A9'010E0000' 02A1    406         .ASCID  \!/Of the physical pages in use, !UL pages are permanently allocated
20 6C 61 63 69 73 79 68 70 20 65 68  02AF
65 73 75 20 6E 69 20 73 65 67 61 70  02BB
20 73 65 67 61 70 20 4C 55 21 20 2C  02C7
6E 65 6E 61 6D 72 65 70 20 65 72 61  02D3
65 74 61 63 6F 6C 6C 61 20 79 6C 74  02DF
         2E 53 4D 56 20 6F 74 20 64  02EB
                                     02F4    407 SHOW$_MEM_SLOT1:
74 6F 6C 53 2F 21 000002FC'010E0000' 02F4    408         .ASCID  \!/Slot Usage (slots):                  Total          Free     Resident
74 6F 6C 73 28 20 65 67 61 73 55 20  0302
20 20 20 20 20 20 20 20 20 3A 29 73  030E
6C 61 74 6F 54 20 20 20 20 20 20 20  031A
65 65 72 46 20 20 20 20 20 20 20 20  0326
74 6E 65 64 69 73 65 52 20 20 20 20  0332
64 65 70 70 61 77 53 20 20 20 20 20  033E
                                     034A    409 SHOW$_MEM_SLOT2:
63 6F 72 50 20 20 00000352'010E0000' 034A    410         .ASCID  \  Process Entry Slots                  !5UL        !5UL        !5UL
6C 53 20 79 72 74 6E 45 20 73 73 65  0358
20 20 20 20 20 20 20 20 20 73 74 6F  0364
20 20 20 4C 55 35 21 20 20 20 20 20  0370
20 20 20 20 4C 55 35 21 20 20 20 20  037C
20 20 20 20 20 4C 55 35 21 20 20 20  0388
                  4C 55 35 21 20 20  0394
                                     039A    411 SHOW$_MEM_SLOT3:
61 6C 61 42 20 20 000003A2'010E0000' 039A    412         .ASCID  \  Balance Set Slots                  !5UL        !5UL        !5UL
74 6F 6C 53 20 74 65 53 20 65 63 6E  03A8
20 20 20 20 20 20 20 20 20 20 20 73  03B4
20 20 20 4C 55 35 21 20 20 20 20 20  03C0
20 20 20 20 4C 55 35 21 20 20 20 20  03CC
20 20 20 20 20 4C 55 35 21 20 20 20  03D8
                  4C 55 35 21 20 20  03E4
                                     03EA    413 SHOW$_MEM_LOOK1:
65 78 69 46 2F 21 000003F2'010E0000' 03EA    414         .ASCID  \!/Fixed-Size Pool Areas (packets):   Total          Free       In Use
20 6C 6F 6F 50 20 65 7A 69 53 2D 64  03F8
```

```
65 6B 63 61 70 28 20 73 61 65 72 41  0404
6C 61 74 6F 54 20 20 20 3A 29 73 74  0410
65 65 72 46 20 20 20 20 20 20 20 20  041C
65 73 55 20 6E 49 20 20 20 20 20 20  0428
65 7A 69 53 20 20 20 20 20 20 20 20  0434
                                     0440   415 SHOW$_MEM_LOOK2:
3C 39 32 21 20 20 00000448'010E0000' 0440   416         .ASCID  \  !29<!AS List!>!9UL   !+!+!9UL   !+!9UL   !+!+!9UL\
39 21 3E 21 74 73 69 4C 20 53 41 21  044E
55 39 21 2B 21 2B 21 20 20 20 4C 55  045A
20 20 4C 55 39 21 2B 21 20 20 20 4C  0466
         4C 55 39 21 2B 21 2B 21 20  0472
                                     047B   417 SHOW$_MEM_LOOK_FULL1:
3C 35 34 21 2F 21 00000483'010E0000' 047B   418         .ASCID  \!/!45<!AS Lookaside List!>Packets       Bytes       Pages\
64 69 73 61 6B 6F 6F 4C 20 53 41 21  0489
6B 63 61 50 3E 21 74 73 69 4C 20 65  0495
79 42 20 20 20 20 20 20 20 73 74 65  04A1
61 50 20 20 20 20 20 20 20 73 65 74  04AD
                        73 65 67 74  04B9
                                     04BC   419 SHOW$_MEM_LOOK_FULL2:
33 21 20 20 20 20 000004C4'010E0000' 04BC   420         .ASCID  \    !39<Current Total Size!>!9UL   !9UL   !9UL\
6F 54 20 74 6E 65 72 72 75 43 3C 39  04CA
39 21 3E 21 65 7A 69 53 20 6C 61 74  04D6
20 20 20 4C 55 39 21 20 20 20 4C 55  04E2
                     4C 55 39 21 20  04EE
                                     04F2   421 SHOW$_MEM_LOOK_FULL3:
33 21 20 20 20 20 000004FA'010E0000' 04F2   422         .ASCID  \    !39<Initial Size (!ASCOUNT)!>!9UL   !9UL   !9UL\
69 53 20 6C 61 69 74 69 6E 49 3C 39  0500
54 4E 55 4F 43 53 41 21 28 20 65 7A  050C
39 21 20 20 20 4C 55 39 21 3E 21 29  0518
         4C 55 39 21 20 20 20 4C 55  0524
                                     052D   423 SHOW$_MEM_LOOK_FULL4:
33 21 20 20 20 20 00000535'010E0000' 052D   424         .ASCID  \    !39<Maximum Size (!ASCOUNTV)!>!9UL   !9UL   !9UL\
69 53 20 6D 75 6D 69 78 61 4D 3C 39  053B
54 4E 55 4F 43 53 41 21 28 20 65 7A  0547
21 20 20 20 4C 55 39 21 3E 21 29 56  0553
      4C 55 39 21 20 20 20 4C 55 39  055F
                                     0569   425 SHOW$_MEM_LOOK_FULL5:
33 21 20 20 20 20 00000571'010E0000' 0569   426         .ASCID  \    !39<Free Space!>!9UL   !9UL\
65 63 61 70 53 20 65 65 72 46 3C 39  0577
55 39 21 20 20 20 4C 55 39 21 3E 21  0583
                                 4C  058F
                                     0590   427 SHOW$_MEM_LOOK_FULL6:
33 21 20 20 20 20 00000598'010E0000' 0590   428         .ASCID  \    !39<Space in Use!>!9UL   !9UL\
55 20 6E 69 20 65 63 61 70 53 3C 39  059E
21 20 20 20 4C 55 39 21 3E 21 65 73  05AA
                        4C 55 39     05B6
                                     05B9   429 SHOW$_MEM_LOOK_FULL7:
35 21 20 20 20 20 000005C1'010E0000' 05B9   430         .ASCID  \    !51<Packet Size/Upper Bound (!AS)!>!9UL\
7A 69 53 20 74 65 6B 63 61 50 3C 31  05C7
6E 75 6F 42 20 72 65 70 70 55 2F 65  05D3
55 39 21 3E 21 29 53 41 21 28 20 64  05DF
                                 4C  05EB
                                     05EC   431 SHOW$_MEM_LOOK_FULL8:
35 21 20 20 20 20 000005F4'010E0000' 05EC   432         .ASCID  \    !51<Lower Bound on Allocation!>!9UL\
6E 75 6F 42 20 72 65 77 6F 4C 3C 31  05FA
74 61 63 6F 6C 6C 41 20 6E 6F 20 64  0606
         4C 55 39 21 3E 21 6E 6F 69  0612
```

```
                                                                L  3
                                                        061B    433 SHOW$_MEM_POOL1:
61 6E 79 44 2F 21 00000623'010E0000'                    061B    434         .ASCID  \!/Dynamic Memory Usage (bytes):      Total        Free      In Use
55 20 79 72 6F 6D 65 4D 20 63 69 6D                     0629
29 73 65 74 79 62 28 20 65 67 61 73                     0635
6C 61 74 6F 54 20 20 20 20 20 20 3A                     0641
65 65 72 46 20 20 20 20 20 20 20 20                     064D
65 73 55 20 6E 49 20 20 20 20 20 20                     0659
74 73 65 67 72 61 4C 20 20 20 20 20                     0665
                                                        0671    435 SHOW$_MEM_POOL2:
41 39 32 21 20 20 00000679'010E0000'                    0671    436         .ASCID  \  !29AS!+!9UL   !+!9UL   !9UL   !9UL\
2B 21 20 20 20 4C 55 39 21 2B 21 53                     067F
20 4C 55 39 21 20 20 20 4C 55 39 21                     068B
                  4C 55 39 21 20 20                     0697
                                                        069D    437 SHOW$_MEM_POOL_FULL1:
   53 41 21 2F 21 000006A5'010E0000'                    069D    438         .ASCID  \!/!AS\
                                                        06AA    439 SHOW$_MEM_POOL_FULL2:
32 21 20 20 20 20 000006B2'010E0000'                    06AA    440         .ASCID  \    !25<Current Size (!AS)!>!9UL    Current Total Size (pages) !7UL
69 53 20 74 6E 65 72 72 75 43 3C 35                     06B8
39 21 3E 21 29 53 41 21 28 20 65 7A                     06C4
6E 65 72 72 75 43 20 20 20 20 4C 55                     06D0
65 7A 69 53 20 6C 61 74 6F 54 20 74                     06DC
55 37 21 20 29 73 65 67 61 70 28 20                     06E8
                                  4C                    06F4
                                                        06F5    441 SHOW$_MEM_POOL_FULL3:
32 21 20 20 20 20 000006FD'010E0000'                    06F5    442         .ASCID  \    !25<Initial Size (NPAGEDYN)!>!9UL    Initial Size (pages)
69 53 20 6C 61 69 74 69 6E 49 3C 35                     0703
4E 59 44 45 47 41 50 4E 28 20 65 7A                     070F
49 20 20 20 20 4C 55 39 21 3E 21 29                     071B
20 65 7A 69 53 20 6C 61 69 74 69 6E                     0727
20 20 20 20 20 29 73 65 67 61 70 28                     0733
                  4C 55 37 21 20 20                     073F
                                                        0745    443 SHOW$_MEM_POOL_FULL4:
32 21 20 20 20 20 0000074D'010E0000'                    0745    444         .ASCID  \    !25<Maximum Size (NPAGEVIR)!>!9UL    Maximum Size (pages)
69 53 20 6D 75 6D 69 78 61 4D 3C 35                     0753
52 49 56 45 47 41 50 4E 28 20 65 7A                     075F
4D 20 20 20 20 4C 55 39 21 3E 21 29                     076B
20 65 7A 69 53 20 6D 75 6D 69 78 61                     0777
20 20 20 20 20 29 73 65 67 61 70 28                     0783
                  4C 55 37 21 20 20                     078F
                                                        0795    445 SHOW$_MEM_POOL_FULL5:
32 21 20 20 20 20 0000079D'010E0000'                    0795    446         .ASCID  \    !25<Free Space (bytes)!>!9UL    Space in Use (bytes)     !9UL\
65 63 61 70 53 20 65 65 72 46 3C 35                     07A3
39 21 3E 21 29 73 65 74 79 62 28 20                     07AF
20 65 63 61 70 53 20 20 20 20 4C 55                     07BB
65 74 79 62 28 20 65 73 55 20 6E 69                     07C7
   4C 55 39 21 20 20 20 20 20 29 73                     07D3
                                                        07DE    447 SHOW$_MEM_POOL_FULL6:
32 21 20 20 20 20 000007E6'010E0000'                    07DE    448         .ASCID  \    !25<Size of Largest Block!>!9UL    Size of Smallest Block   !9U
61 4C 20 66 6F 20 65 7A 69 53 3C 35                     07EC
21 6B 63 6F 6C 42 20 74 73 65 67 72                     07F8
7A 69 53 20 20 20 20 4C 55 39 21 3E                     0804
73 65 6C 6C 61 6D 53 20 66 6F 20 65                     0810
39 21 20 20 20 6B 63 6F 6C 42 20 74                     081C
                                  4C 55                 0828
                                                        082A    449 SHOW$_MEM_POOL_FULL7:
32 21 20 20 20 20 00000832'010E0000'                    082A    450         .ASCID  \    !25<Number of Free Blocks!>!9UL    Free Blocks LEQU 32 Bytes!9U
20 66 6F 20 72 65 62 6D 75 4E 3C 35                     0838
```

```
21 73 6B 63 6F 6C 42 20 65 65 72 46  0844
65 72 46 20 20 20 20 4C 55 39 21 3E  0850
51 45 4C 20 73 6B 63 6F 6C 42 20 65  085C
39 21 73 65 74 79 42 20 32 33 20 55  0868
                        4C 55        0874
                                     0876    451
                                     0876    452 ; The following two constants are used to pass FAO directive size to
                                     0876    453 ; module SHOMEMORY. If the size of either file name string is changed,
                                     0876    454 ; the value of the constant and the FAO descriptor numeric value must
                                     0876    455 ; both be changed.
                                     0876    456
                                     0876    457         ASSUME SHOW$C_MEM_SHORT_NAME EQ 40
                                     0876    458         ASSUME SHOW$C_MEM_LONG_NAME  EQ 78
                                     0876    459
                                     0876    460 SHOW$_MEM_PAGE1:
69 67 61 50 2F 21 0000087E'010E0000' 0876    461         .ASCID  \!/Paging File Usage (pages):                    Free      In Use
67 61 73 55 20 65 6C 69 46 20 67 6E  0884
20 20 3A 29 73 65 67 61 70 28 20 65  0890
20 20 20 20 20 20 20 20 20 20 20 20  089C
65 65 72 46 20 20 20 20 20 20 20 20  08A8
65 73 55 20 6E 49 20 20 20 20 20 20  08B4
6C 61 74 6F 54 20 20 20 20 20 20 20  08C0
                                     08CC    462 SHOW$_MEM_PAGE2:
41 30 34 21 20 20 000008D4'010E0000' 08CC    463         .ASCID  \  !40AS   !7UL     !7UL     !7UL\
20 20 20 20 4C 55 37 21 20 20 20 53  08DA
37 21 20 20 20 20 20 4C 55 37 21 20  08E6
                        4C 55        08F2
                                     08F4    464 SHOW$_MEM_PAGE3:
41 38 37 21 20 20 000008FC'010E0000' 08F4    465         .ASCID  \  !78AS\
                              53     0902
                                     0903    466 SHOW$_MEM_PAGE4:
20 20 20 20 2B 21 0000090B'010E0000' 0903    467         .ASCID  \!+                                      !7UL     !7UL     !7
20 20 20 20 20 20 20 20 20 20 20 20  0911
20 20 20 20 20 20 20 20 20 20 20 20  091D
20 20 20 20 20 20 20 20 20 20 20 20  0929
20 20 20 4C 55 37 21 20 20 20 20 20  0935
21 20 20 20 20 20 4C 55 37 21 20 20  0941
                        4C 55 37     094D
                                     0950    468 SHOW$_MEM_PAGE_FULL1:
41 38 37 21 2F 21 00000958'010E0000' 0950    469         .ASCID  \!/!78AS\
                              53     095E
                                     095F    470 SHOW$_MEM_PAGE_FULL2:
72 46 20 20 20 20 00000967'010E0000' 095F    471         .ASCID  \    Free Blocks                 !7UL    Blocks in Use              !
20 20 20 73 6B 63 6F 6C 42 20 65 65  096D
20 20 20 20 20 20 20 20 20 20 20 20  0979
6F 6C 42 20 20 20 20 4C 55 37 21 20  0985
20 20 65 73 55 20 6E 69 20 73 6B 63  0991
20 20 20 20 20 20 20 20 20 20 20 20  099D
                        4C 55 37 21  09A9
                                     09AD    472 SHOW$_MEM_PAGE_FULL3:
6F 54 20 20 20 20 000009B5'010E0000' 09AD    473         .ASCID  \    Total Size (blocks)         !7UL    Paging File Number         !
6C 62 28 20 65 7A 69 53 20 6C 61 74  09BB
20 20 20 20 20 20 20 29 73 6B 63 6F  09C7
67 61 50 20 20 20 20 4C 55 37 21 20  09D3
6D 75 4E 20 65 6C 69 46 20 67 6E 69  09DF
20 20 20 20 20 20 20 20 20 72 65 62  09EB
                        4C 55 37 21  09F7
```

```
                                        09FB   474 SHOW$_MEM_PAGE_FULL4:
77 53 20 20 20 20 00000A03'010E0000'    09FB   475         .ASCID  \    Swap Usage (processes)     !7UL    Paging Usage (processes)   !
72 70 28 20 65 67 61 73 55 20 70 61     0A09
20 20 20 20 29 73 65 73 73 65 63 6F     0A15
67 61 50 20 20 20 20 4C 55 37 21 20     0A21
70 28 20 65 67 61 73 55 20 67 6E 69     0A2D
20 20 20 29 73 65 73 73 65 63 6F 72     0A39
                        4C 55 37 21     0A45
                                        0A49   476 SHOW$_MEM_PAGE_FULL5:
   53 41 21 20 20 00000A51'010E0000'    0A49   477         .ASCID  \  !AS\
                                        0A56   478
                                        0A56   479
                                000000D8   480         .PSECT  SHOW$RWDATA     LONG,RD,WRT,NOEXE
                                    00D8   481
                                    00D8   482 ;       PAGING FILE FAO ARGUMENT LIST
                                    00D8   483
                                    00D8   484 SHOW_PAGE_LIST:
                            000001FF'   00D8   485         .ADDRESS        FILE_NAME_DESC  ; DESCRIPTOR FOR FILENAME
                                    00DC   486 SHOW_PAGE_LIST2:
                                    00DC   487 PAGE_FREE:
                            000000E0    00DC   488         .BLKL   1                       ; SPACE FOR NUMBER OF FREE PAGES
                                    00E0   489 PAGE_USED:
                            000000E4    00E0   490         .BLKL   1                       ; SPACE FOR NUMBER OF PAGES IN USE
                                    00E4   491 SHOW_PAGE_LIST3:
                                    00E4   492 PAGE_TOTAL:
                            000000E8    00E4   493         .BLKL   1                       ; SPACE FOR SIZE OF PAGING FILE
                                    00E8   494 PAGE_PFL_INDEX:
                            000000EC    00E8   495         .BLKL   1                       ; PAGE/SWAP FILE INDEX
                                    00EC   496 SHOW_PAGE_LIST4:
                                    00EC   497 PAGE_FULL_SWAP_COUNT:
                            000000F0    00EC   498         .BLKL   1                       ; COUNT OF PROCESSES SWAPPING TO FILE
                                    00F0   499 PAGE_FULL_PAGING_COUNT:
                            000000F4    00F0   500         .BLKL   1                       ; COUNT OF PROCESSES PAGING TO FILE
                                    00F4   501 SHOW_PAGE_LIST5:
                                    00F4   502 PAGE_FLAG:
                            00000A56'   00F4   503         .ADDRESS        SWAP_INDIC_DESC ; DESCRIPTOR FOR PAGING INDICATOR
                                    00F8   504
                                    00F8   505 ;
                                    00F8   506 ;       FILENAME SECTION
                                    00F8   507 ;
                                    00F8   508
                                    00F8   509 DEVICE_NAME_DESC:                    ; Descriptor for device name passed
                            00000100    00F8   510         .BLKL           2           ;  to LIB$FID_TO_NAME and $GETDVI
                                    0100   511
                            000000FF    0100   512         FILE_NAME_SIZE = 255
                                    0100   513 FILE_NAME_ADDR:
                            000001FF    0100   514         .BLKB   FILE_NAME_SIZE
                                    01FF   515 FILE_NAME_DESC:                      ; Descriptor for returned filename
                            000000FF    01FF   516         .LONG           FILE_NAME_SIZE  ;  from LIB$FID_TO_NAME routine
                            00000100'   0203   517         .ADDRESS        FILE_NAME_ADDR  ;  and passed to output routines
                                    0207   518
                            00000040    0207   519         DEVICE_NAME_SIZE = 64       ; Alternate output buffer for
                                    0207   520 DEVICE_NAME_ADDR:                    ;  $GETDVI. Contents used if no
                            00000247    0207   521         .BLKB   DEVICE_NAME_SIZE    ;  LOGVOLNAM returned.
                                    0247   522
                                    0247   523
                                    0247   524 SCRATCH_DESC:                       ; Scratch string descriptor
```

```
                                                B 4
                     0000024F  0247   525          .BLKL   2                       ;  used by $FAO and $TRNLOG
                               024F   526
                               024F   527 ; Space for returned length from LIB$FID_TO_NAME routine. Also used by
                               024F   528 ; $FAO, $GETDVI, and $TRNLOG.
                               024F   529
                               024F   530 RETURN_LENGTH:
                     00000253  024F   531          .BLKL   1
                               0253   532
                               0253   533 ;  Static pieces of default file name
                               0253   534
                               0253   535 DEFAULT_DIRECTORY_NAME:                 ; Device name is loaded by $GETDVI
45 5B 45 53 59 53 0000025B'010E0000'   536          .ASCID  /SYSEXE]/              ; ":]" are loaded dynamically
                                  5D   0261
                               0262   537 DEFAULT_FILE_NAME:                     ; First 4 characters may become
53 2E 45 4C 49 46 0000026A'010E0000'   538          .ASCID  /FILE.SYS/             ;  either "PAGE" or "SWAP"
                                 53 59 0270
                               0272   539
                               0272   540          .ALIGN  LONG                   ; Location counter back to longword
                               0274   541
                               0274   542 PFL_TABLE_SIZE:
                     00000278  0274   543          .BLKL   1                      ; Size of scratch area
                               0278   544 PFL_TABLE_ADDR:
                     0000027C  0278   545          .BLKL   1                      ; Address of scratch area for PFLs
                               027C   546 SWAP_FILE_COUNT:
                     00000280  027C   547          .BLKL   1                      ; Maximum number of swap files (SWPFILCNT)
                               0280   548 PAGE_FILE_COUNT:
                     00000284  0280   549          .BLKL   1                      ; Maximum number of paging files (PAGFILCNT)
                               0284   550 SWAP_FILE_TABLE:                       ; Address of swap file usage array
                     00000288  0284   551          .BLKL   1                      ; (PAGFILCNT + SWPFILCNT entries long)
                               0288   552 PAGE_FILE_TABLE:                       ; Address of paging file usage array
                     0000028C  0288   553          .BLKL   1                      ; (PAGFILCNT + SWPFILCNT entries long)
                               028C   554
                               028C   555 ; Text descriptors that distinguish files that are used for paging
                               028C   556 ; and swapping from files used only for swapping.
                               028C   557
                          00000A56   558          .PSECT  SHOW$MSG_TEXT   BYTE,RD,NOWRT,NOEXE
                               0A56   559
                               0A56   560 SWAP_INDIC_DESC:
66 20 73 69 68 54 00000A5E'010E0000'   561          .ASCID  /This file is used exclusively for swapping./
20 64 65 73 75 20 73 69 20 65 6C 69    0A64
20 79 6C 65 76 69 73 75 6C 63 78 65    0A70
67 6E 69 70 70 61 77 73 20 72 6F 66    0A7C
                                  2E   0A88
                               0A89   562 PAGE_INDIC_DESC:
66 20 73 69 68 54 00000A91'010E0000'   563          .ASCID  /This file can be used for either paging or swapping./
75 20 65 62 20 6E 61 63 20 65 6C 69    0A97
68 74 69 65 20 72 6F 66 20 64 65 73    0AA3
7. 6F 20 67 6E 69 67 61 70 20 72 65    0AAF
      2E 67 6E 69 70 70 61 77 73 20    0ABB
                               0AC5   564
                          0000028C   565          .PSECT  SHOW$RWDATA     LONG,RD,WRT,NOEXE
                               028C   566
                               028C   567 ; Data area for call to $GETJPI to retrieve page and swap file data
                               028C   568
                               028C   569 PAGE_FILE_LOC:
                     00000290  028C   570          .BLKL   1                      ; Paging file address
                     0000028F  0290   571 PAGE_FILE_INDEX = PAGE_FILE_LOC + 3
```

```
                     0290   572
                     0290   573 SWAP_FILE_LOC:
00000294             0290   574         .BLKL   1                       ; Swap file location
00000293             0294   575 SWAP_FILE_INDEX = SWAP_FILE_LOC + 3
                     0294   576
                     0294   577 GETJPI_STATUS:
0000029C             0294   578         .BLKQ   1                       ; Status block for asynchronous $GETJPI
                     029C   579
                     029C   580 PID:
FFFFFFFF             029C   581         .LONG   -1                      ; Wild card PID for $GETJPI
                     02A0   582
                     02A0   583 ; Argument list for call to LIB$FID_TO_NAME
                     02A0   584
                     02A0   585 FID_TO_NAME_ARG_LIST:
00000004             02A0   586         .LONG           4               ; Argument count
000000F8'            02A4   587         .ADDRESS        DEVICE_NAME_DESC        ; Descriptor for device name
                     02A8   588 FID_TO_NAME_FID_ADDR:
000002AC             02A8   589         .BLKL           2               ; Space for FID address
000001FF'            02AC   590         .ADDRESS        FILE_NAME_DESC  ; File name descriptor
0000024F'            02B0   591         .ADDRESS        RETURN_LENGTH   ; File name length goes here
                     02B4   592
                     02B4   593 ; This FAO list is required to convert the unit number to an unsigned
                     02B4   594 ; decimal integer. The unit number itself is stored in the $FAO
                     02B4   595 ; argument list at execution time but we must reserve space for it
                     02B4   596 ; at assembly time so that the $FAO argument is the correct length.
                     02B4   597
                     02B4   598 FAO_LIST:
                     02B4   599         $FAO    CTRSTR=FAO_CONTROL_STRING,-
                     02B4   600                 OUTLEN=RETURN_LENGTH,-
                     02B4   601                 OUTBUF=SCRATCH_DESC,-
                     02B4   602                 P1=0
                     02C8   603
                00000054    604         .PSECT  SHOW$RODATA     LONG,RD,NOWRT,NOEXE
                     0054   605
                     0054   606 JPI_ITEM_LIST:
                0004 0054   607         .WORD           4               ; Destination is a longword
                0419 0056   608         .WORD           JPI$_PAGFILLOC  ; Request paging file address
0000028C'            0058   609         .ADDRESS        PAGE_FILE_LOC   ; Store result here
00000000             005C   610         .LONG           0               ; Do not return length
                     0060   611
                0004 0060   612         .WORD           4               ; Destination is a longword
                0321 0062   613         .WORD           JPI$_SWPFILLOC  ; Request swap file location
00000290'            0064   614         .ADDRESS        SWAP_FILE_LOC   ; Store result here
00000000             0068   615         .LONG           0               ; Do not return length
                     006C   616
00000000             006C   617         .LONG           0               ; End of $GETJPI request list
                     0070   618
                     0070   619 GETJPI_LIST:
                     0070   620         $GETJPI         EFN=EVENT_FLAG,-
                     0070   621                         PIDADR=PID,-
                     0070   622                         ITMLST=JPI_ITEM_LIST,-
                     0070   623                         IOSB=GETJPI_STATUS
                     0090   624
                     0090   625 DVI_ITEM_LIST:
                00FF 0090   626         .WORD           FILE_NAME_SIZE
                002C 0092   627         .WORD           DVI$_LOGVOLNAM  ; Request logical volume name
00000100'            0094   628         .ADDRESS        FILE_NAME_ADDR  ; Store string result here
```

```
                                                D 4
                    000001FF' 0098   629          .ADDRESS        FILE_NAME_DESC  ;  and size here
                              009C   630
                        0040  009C   631          .WORD           DEVICE_NAME_SIZE
                        0020  009E   632          .WORD           DVI$_DEVNAM     ; Request logical volume name
                    00000207' 00A0   633          .ADDRESS        DEVICE_NAME_ADDR ; Store string result here
                    0000024F' 00A4   634          .ADDRESS        RETURN_LENGTH   ;  and size here
                              00A8   635
                    00000000  00A8   636          .LONG           0               ; End of $GETDVI request list
                              00AC   637
                              00AC   638 GETDVI_LIST:
                              00AC   639          $GETDVI         EFN=EVENT_FLAG,-
                              00AC   640                          DEVNAM=DEVICE_NAME_DESC,-
                              00AC   641                          ITMLST=DVI_ITEM_LIST
                              00D0   642
                              00D0   643 FAO_CONTROL_STRING:
        57 55 21 000000D8'010E0000'   00D0   644          .ASCID          /!UW/
                              00DB   645
                              00DB   646 TOPSYS_DESC:
4F 54 24 53 59 53 000000E3'010E0000'  00DB   647          .ASCID          /SYS$TOPSYS/
                     53 59 53 50      00E9
                              00ED   648
                              00ED   649 TRNLOG_LIST:
                              00ED   650          $TRNLOG         LOGNAM=TOPSYS_DESC,-
                              00ED   651                          RSLLEN=RETURN_LENGTH,-
                              00ED   652                          RSLBUF=SCRATCH_DESC,-
                              00ED   653                          DSBMSK=<^B110>  ; Only search system name table
                              0109   654
```

```
                                0109   656          .SBTTL  SHOW$MEMORY Show System Memory Resources
                                0109   657 ;++
                                0109   658 ; Functional Description:
                                0109   659 ;
                                0109   660 ;       This routine retrieves information about various system resources,
                                0109   661 ;       formats and prints it on SYS$OUTPUT.
                                0109   662 ;
                                0109   663 ; Calling Sequence:
                                0109   664 ;
                                0109   665 ;       CALLS   #0,SHOW$MEMORY
                                0109   666 ;
                                0109   667 ;               The routine is actually called by the CLI as a result of
                                0109   668 ;               parsing parameter MEMORY on the SHOW command.
                                0109   669 ;
                                0109   670 ; Input Parameters:
                                0109   671 ;
                                0109   672 ;       None
                                0109   673 ;
                                0109   674 ; Implicit Input:
                                0109   675 ;
                                0109   676 ;       Qualifiers specified on the SHOW MEMORY command
                                0109   677 ;
                                0109   678 ; Output Parameters:
                                0109   679 ;
                                0109   680 ;       None
                                0109   681 ;
                                0109   682 ; Implicit Output:
                                0109   683 ;
                                0109   684 ;       Memory resource information is displayed on SYS$OUTPUT.
                                0109   685 ;
                                0109   686 ; Completion Codes:
                                0109   687 ;
                                0109   688 ;       SS$_NORMAL              Normal completion
                                0109   689 ;       SS$_LKWSETFUL           Error in locking data for elevated IPL
                                0109   690 ;--
                                0109   691
                          00000000   692          .PSECT  SHOW$CODE BYTE,RD,NOWRT,EXE
                                0000   693
                          0000  0000   694          .ENTRY  SHOW$MEMORY,0           ; SHOW MEMORY resources routine
                                0002   695
              00000000'EF   DF  0002   696          PUSHAL  MEMORY_D_PHYS           ; /PHYSICAL_MEMORY
          00000000'EF   01    FB  0008   697          CALLS   #1,CLI$PRESENT
00000008'EF   01    00    50    F0  000F   698          INSV    R0,#MEMORY_V_PHYS,#1,MEMORY_L_BITLIS
                                0018   699
              00000017'EF   DF  0018   700          PUSHAL  MEMORY_D_SLOTS          ; /SLOTS
          00000000'EF   01    FB  001E   701          CALLS   #1,CLI$PRESENT
00000008'EF   01    01    50    F0  0025   702          INSV    R0,#MEMORY_V_SLOT,#1,MEMORY_L_BITLIS
                                002E   703
              00000024'EF   DF  002E   704          PUSHAL  MEMORY_D_POOL           ; /POOL
          00000000'EF   01    FB  0034   705          CALLS   #1,CLI$PRESENT
00000008'EF   01    02    50    F0  003B   706          INSV    R0,#MEMORY_V_POOL,#1,MEMORY_L_BITLIS
                                0044   707
              00000030'EF   DF  0044   708          PUSHAL  MEMORY_D_FILES          ; /FILES
          00000000'EF   01    FB  004A   709          CALLS   #1,CLI$PRESENT
00000008'EF   01    03    50    F0  0051   710          INSV    R0,#MEMORY_V_FILE,#1,MEMORY_L_BITLIS
                                005A   711
              0000003D'EF   DF  005A   712          PUSHAL  MEMORY_D_FULL           ; /FULL
```

```
                00000000'EF   01   FB  0060   713          CALLS   #1,CLI$PRESENT
    00000008'EF   01   04   50   F0  0067   714          INSV    R0,#MEMORY_V_FULL,#1,MEMORY_L_BITLIS
                                        0070   715
                                        0070   716
                00000049'EF   DF   0070   717          PUSHAL  MEMORY_D_ALL                ; /ALL
                00000000'EF   01   FB  0076   718          CALLS   #1,CLI$PRESENT
                         07   50   E9  007D   719          BLBC    R0,5$                       ; Branch if /ALL not set
                                        0080   720
                00000008'EF   0F   C8  0080   721          BISL2   #<MEMORY_M_PHYS!-
                                        0087   722                    MEMORY_M_SLOT!-
                                        0087   723                    MEMORY_M_POOL!-
                                        0087   724                    MEMORY_M_FILE-
                                        0087   725                   >,MEMORY_L_BITLIS      ; Set all bits except /FULL
                                        0087   726
    50          00000008'EF   10   CB  0087   727 5$:      BICL3   #MEMORY_M_FULL,MEMORY_L_BITLIS,R0 ; Anything other than /FULL?
                         07   12   8F  008F   728          BNEQ    10$                         ; Branch if any other qualifier present
                00000008'EF   0F   D0  0091   729          MOVL    #<MEMORY_M_PHYS!-           ; Default is these four displays
                                        0098   730                    MEMORY_M_SLOT!-
                                        0098   731                    MEMORY_M_POOL!-
                                        0098   732                    MEMORY_M_FILE-
                                        0098   733                   >,MEMORY_L_BITLIS      ; Set all bits except /FULL
                                        0098   734
                                        0098   735 ;  Lock down code that will be accessed at elevated IPL.
                                        0098   736
                                        0098   737 10$:     $LKWSET_S       LOCKED_CODE_RANGE       ; Lock code in working set
                         74   50   E9  00A9   738          BLBC    R0,90$                      ; Exit if error occurred
                                        00AC   739                                              ; Will be unlocked by image rundown
                                        00AC   740 ;  Print header line for all displays
                                        00AC   741
                                        00AC   742          TYPEMSG SHOW$_MEM_HEAD1,HEADER_LIST
                                        00BF   743
                                        00BF   744 ; Show the information based on the actual or implied setting of each
                                        00BF   745 ; qualifier bit in the control mask.
                                        00BF   746
    07 00000008'EF   00   E1  00BF   747          BBC     #MEMORY_V_PHYS,MEMORY_L_BITLIS,20$      ; /PHYSICAL_MEMORY
       00000121'EF   00   FB  00C7   748          CALLS   #0,MEMORY                       ; Print physical memory usage
    07 00000008'EF   01   E1  00CE   749 20$:     BBC     #MEMORY_V_SLOT,MEMORY_L_BITLIS,30$      ; /SLOTS
       000002FE'EF   00   FB  00D6   750          CALLS   #0,SLOTS                        ; Print slot usage
    0E 00000008'EF   02   E1  00DD   751 30$:     BBC     #MEMORY_V_POOL,MEMORY_L_BITLIS,40$      ; /POOL
       00000428'EF   00   FB  00E5   752          CALLS   #0,LOOKASIDE    ; Print fixed-size pool usage
       000006DB'EF   00   FB  00EC   753          CALLS   #0,POOL         ; Print variable-sized pool usage
    07 00000008'EF   03   E1  00F3   754 40$:     BBC     #MEMORY_V_FILE,MEMORY_L_BITLIS,50$      ; /PAGEFILE
       000009B6'EF   00   FB  00FB   755          CALLS   #0,PAGEFILE                     ; Print paging file usage
    13 00000008'EF   00   E1  0102   756 50$:     BBC     #MEMORY_V_PHYS,MEMORY_L_BITLIS,60$      ; /PHYSICAL_MEMORY
                                        010A   757          TYPEMSG SHOW$_MEM_PARA1,PARA_VMS ; Print bottom paragraph
                    50   01   3C  011D   758 60$:     MOVZWL  #SS$_NORMAL,R0                  ; Store status
                                        0120   759 90$:
                         04   0120   760          RET                                     ;  and exit
                                        0121   761
```

```
                                0121   763          .SBTTL  SHOW MEMORY USAGE
                                0121   764 ;
                                0121   765 ;       SHOW PHYSICAL MEMORY
                                0121   766 ;
                                0121   767 ;       THIS ROUTINE DISPLAYS INFORMATION ABOUT THE SYSTEM MEMORY.
                                0121   768 ;       THE TOTAL NUMBER OF PAGES AVAILABLE TO THE SYSTEM IS DISPLAYED
                                0121   769 ;       BOTH AS A NUMBER OF PAGES AND IN APPROXIMATE MEGABYTES. THE
                                0121   770 ;       NUMBER OF PAGES ON THE MODIFIED AND FREE LIST ARE ALSO SHOWN.
                                0121   771 ;       THE NUMBER OF PAGES IN USE BY BOTH THE SYSTEM AND USERS ARE SHOWN,
                                0121   772 ;       AND THE NUMBER OF PAGES ALWAYS IN USE BY THE SYSTEM IS DISPLAYED
                                0121   773 ;       IN THE CONCLUDING PARAGRAPH. IF THERE SHOULD BE BAD MEMORY,
                                0121   774 ;       AN ADDITIONAL LINE IS PRINTED GIVING THE NUMBER OF BAD PAGES.
                                0121   775 ;
                                0121   776
                                0121   777 MEMORY:
                         001C   0121   778          .WORD   ^M<R2,R3,R4>             ; Save some registers
                                0123   779          TYPEMSG SHOW$_MEM_MEMO1                         ; PRINT HEADER
                                0132   780          $CMEXEC_S       SIZE_MEMORY     ; Calculate physical memory size
00000020'EF   00000000'GF   D0  0141   781          MOVL    G^SCH$GL_FREECNT,MEM_FREE_PAGES ; GET # OF FREE PAGES
00000028'EF   00000000'GF   D0  014C   782          MOVL    G^SCH$GL_MFYCNT,MEM_MODF_PAGES  ; GET # OF MODIFIED PAGES
00000000'GF   0000001C'EF   D1  0157   783          CMPL    MEM_PHY_PAGES,G^MMG$GL_PHYPGCNT ; MINIMIZE PHYSICAL PAGE
                      0B    1B  0162   784          BLEQU   10$                             ; COUNT WITH SYSGEN SPECIFIE
0000001C'EF   00000000'GF   D0  0164   785          MOVL    G^MMG$GL_PHYPGCNT,MEM_PHY_PAGES ; PAGE COUNT
0000001C'EF   00000020'EF   C3  016F   786 10$:     SUBL3   MEM_FREE_PAGES,MEM_PHY_PAGES,MEM_USED_PAGES
              00000024'EF       017A
00000024'EF   00000028'EF   C2  017F   787          SUBL2   MEM_MODF_PAGES,MEM_USED_PAGES    ; GET # OF PAGES IN USE
0000001C'EF   00000000'GF   C3  018A   788          SUBL3   G^PFN$GL_PHYPGCNT,MEM_PHY_PAGES,PARA_VMS
              0000005C'EF       0195
      0000001C'EF   F5 8F   78  019A   789          ASHL    #-11,MEM_PHY_PAGES,MEM_MB_1             ; CONVERT COUNT OF
              00000014'EF       01A2
   52 0000001C'EF   F7 8F   78  01A7   790          ASHL    #-9,MEM_PHY_PAGES,R2                    ; PHYSICAL PAGES TO
      53 04 00000014'EF   C5  01B0   791          MULL3   MEM_MB_T,#4,R3                          ; MEGABYTES
                 52   53   C2  01B8   792          SUBL2   R3,R2
00000040'EF   00000044'EF42   DE  01BB   793          MOVAL   MEM_MB_TEXT[R2],MEM_MB_DESC+4
                             01C7   794          TYPEMSG SHOW$_MEM_MEMO2,SHOW_MEM_PHY            ; TYPE TEXT
           52   00000000'GF   D0  01DA   795          MOVL    G^EXE$GL_RPB,R2                 ; GET ADDR OF RPB
         52   0104 C2       D0  01E1   796          MOVL    RPB$L_BADPGS(R2),R2             ; GET COUNT OF BAD PAGES AT BOOT
      54   00000008'GF       D0  01E6   797          MOVL    G^SCH$GL_FREECNT+<4*PFN$C_BADPAGLST>,R4 ; BAD PAGES AFTER BOOT
   0000002C'EF   54   52      C1  01ED   798          ADDL3   R2,R4,MEM_BAD_LIST              ; TOTAL BAD PAGES
                      40   13  01F5   799          BEQL    20$                             ; IF NONE SKIP THIS DISPLAY
0000005C'EF   0000002C'EF   C2  01F7   800          SUBL2   MEM_BAD_LIST,PARA_VMS           ; DON'T COUNT BAD PAGES AS
                             0202   801                                                  ; ALLOCATED TO VMS
                             0202   802          $CMEXEC_S ROUTIN = SCAN_BAD_LIST         ; COUNT ''REALLY'' BAD PAGES
00000034'EF   54   00000030'EF   C3  0211   803          SUBL3   MEM_BAD_PAGES,R4,MEM_OTHER_PAGES ; STORE COUNT OF ''OTHER'' PAGES
              00000038'EF   52   D0  021D   804          MOVL    R2,MEM_BOOT_PAGES               ; STORE # BAD PAGES AT BOOT
                             0224   805          TYPEMSG SHOW$_MEM_MEMO3,MEM_BAD_LIST    ; THEN TELL THE USER
                         04  0237   806 20$:     RET
                             0238   807
```

```
                                                H 4
                        809          .SUBTITLE       SIZE_MEMORY     Get Amount of Physical Memory
                        810
                        811 ;+
                        812 ;       SIZE_MEMORY     Get Amount of Physical Memory
                        813 ;
                        814 ;  This routine uses the memory descriptors in the Restart Parameter Block
                        815 ;  to determine the amount of physical memory in use. A check is made to
                        816 ;  see if multiport memory should be counted as local memory.
                        817 ;
                        818 ;  Calling sequence:
                        819 ;
                        820 ;       CALLS   #0,SIZE_MEMORY
                        821 ;
                        822 ;  Input parameters:
                        823 ;
                        824 ;       None
                        825 ;
                        826 ;  Implicit Input:
                        827 ;
                        828 ;       Memory descriptors in RPB
                        829 ;
                        830 ;  Output parameters:
                        831 ;
                        832 ;       LOCAL_MEMORY    Total memory in local memory controllers
                        833 ;
                        834 ;       SHARED_MEMORY   Total memory in multiport emmory controllers
                        835 ;
                        836 ;       MEM_PHY_PAGES   Total amount of physical memory in use by system
                        837 ;                       (This total does not include multiport memory
                        838 ;                        being used as shared memory.)
                        839 ;
                        840 ;-
                        841
                        842 SIZE_MEMORY:
                  001C  843          .WORD   ^M<R2,R3,R4>                    ; Save some registers
 50 00000000'GF   D0    844          MOVL    G^EXE$GL_CONFREGL,R0            ; Get address of TR/adapter type array
 51 00000000'GF   D0    845          MOVL    G^EXE$GL_RPB,R1                                 ; GET ADDR OF RPB
    52 00BC C1    DE    846          MOVAL   RPB$L_MEMDSC(R1),R2                             ; GET ADDR OF MEMORY DESCRIP
    00000054'EF   D4    847          CLRL    LOCAL_MEMORY                            ; INIT PAGE COUNT
    00000058'EF   D4    848          CLRL    SHARED_MEMORY                           ; INIT PAGE COUNT
          62      D5    849 10$:     TSTL    (R2)                                    ; END OF MEMDSC LIST?
          2F      13    850          BEQL    40$                                     ; YES - GO PRINT INFO
 53 62    08      18    EF  851          EXTZV   #RPB$V_TR,#RPB$S_TR,(R2),R3             ; GET TR NUMBER
       53    6043    D0  852          MOVL    (R0)[R3],R3                             ; CONVERT TO ADAPTER TYPE
 54 62    18      00    EF  853          EXTZV   #RPB$V_PAGCNT,#RPB$S_PAGCNT,(R2),R4     ; GET PAGE COUNT
                        026B  854
                        026B  855 ; The following set of assumptions state that all multiport memory adapter
                        026B  856 ; type codes are bounded by NDT$_MPM0 and NDT$_MPM3 and that no adapter
                        026B  857 ; type codes in this range represent anything other than multiport memory.
                        026B  858
                        026B  859          ASSUME  NDT$_MPM0 LT NDT$_MPM1
                        026B  860          ASSUME  NDT$_MPM1 LT NDT$_MPM2
                        026B  861          ASSUME  NDT$_MPM2 LT NDT$_MPM3
                        026B  862
    40 8F    53    91   026B  863          CMPB    R3,#NDT$_MPM0                   ; Is adapter number below MPM range
          0F    1F    026F  864          BLSSU   20$                             ; If so, this is local memory
    43 8F    53    91   0271  865          CMPB    R3,#NDT$_MPM3                   ; Is adapter number above MPM range
```

```
                     09   1A  0275    866          BGTRU   20$                     ; If so, this is also local memory
          00000058'EF    54  C0  0277    867          ADDL2   R4,SHARED_MEMORY      ; Otherwise, this is multiport memory
                     07   11  027E    868          BRB     30$                     ; Go to end of loop
                              0280    869
          00000054'EF    54  C0  0280    870 20$:     ADDL2   R4,LOCAL_MEMORY       ; This is local memory
                    52   08  C0  0287    871 30$:     ADDL2   #RPB$C_MEMDSCSIZ,R2   ; Point to next memory descriptor
                     CD   11  028A    872          BRB     10$                     ;  and go back to top of loop
                              028C    873
                              028C    874 ; There are four cases that can occur here.
                              028C    875 ;
                              028C    876 ; 1.  There are no multiport memory controllers on the system.
                              028C    877 ;
                              028C    878 ; 2.  Multiport memory is being used as global shared memory.
                              028C    879 ;
                              028C    880 ; 3.  Multiport memory is being used as local memory. This case is
                              028C    881 ;     distinguished by RPB$V_USEMPM being set in the RPB copy of R5.
                              028C    882 ;
                              028C    883 ; 4.  Only multiport memory is being used as local memory. Any memory
                              028C    884 ;     in local controllers is ignored. This is the multiprocessor
                              028C    885 ;     configuration. This case is distinguished by RPB$V_USEMPM
                              028C    886 ;     being set in the RPB copy of R5.
                              028C    887
          1D 30 A1     0B   E0  028C    888 40$:     BBS     #RPB$V_MPM,RPB$L_BOOTR5(R1),50$ ; Multiprocessor configuration?
0000001C'EF    00000054'EF    D0  0291    889          MOVL    LOCAL_MEMORY,MEM_PHY_PAGES      ; Local memory is always counted
          18 30 A1     0C   E1  029C    890          BBC     #RPB$V_USEMPM,RPB$L_BOOTR5(R1),60$ ; Also count shared memory?
0000001C'EF    00000058'EF    C0  02A1    891          ADDL2   SHARED_MEMORY,MEM_PHY_PAGES     ; Add it in if using as local memory
                     0B   11  02AC    892          BRB     60$                     ;  and return
                              02AE    893
0000001C'EF    00000058'EF    D0  02AE    894 50$:     MOVL    SHARED_MEMORY,MEM_PHY_PAGES     ; Only count shared memory
                    50   01   3C  02B9    895 60$:     MOVZWL  #SS$_NORMAL,R0          ; Indicate success
                              04  02BC    896          RET                             ;  and return
                              02BD    897
```

```
                                                J 4
                     02BD   899          .SUBTITLE       SCAN_BAD_LIST   Scan Bad Page List
                     02BD   900
                     02BD   901 ;+
                     02BD   902 ;
                     02BD   903 ;
                     02BD   904 ;  This routine looks at all pages on the bad page list to distinguish those
                     02BD   905 ;  pages that exhibit memory errors (are marked as bad) form those pages
                     02BD   906 ;  placed there due to an I/O error.
                     02BD   907 ;
                     02BD   908 ;  Calling sequence:
                     02BD   909 ;
                     02BD   910 ;       BSBW    SCAN_BAD_LIST
                     02BD   911 ;
                     02BD   912 ;  Input parameters:
                     02BD   913 ;
                     02BD   914 ;       None
                     02BD   915 ;
                     02BD   916 ;  Implicit Input:
                     02BD   917 ;
                     02BD   918 ;       PFN data base listheads
                     02BD   919 ;
                     02BD   920 ;  Output parameter:
                     02BD   921 ;
                     02BD   922 ;       MEM_BAD_PAGES   Count of pages marked as bad
                     02BD   923 ;
                     02BD   924 ;-
                     02BD   925
                     02BD   926 SCAN_BAD_LIST:
                000C 02BD   927          .WORD   ^M<R2,R3>                       ; Mask these registers
                  53   D4   02BF   928          CLRL    R3                      ; Initialize bad page counter
50   00000008'GF     D0   02C1   929          MOVL    G^<PFN$AL_HEAD+<4*PFN$C_BADPAGLST>>,R0  ; Get first bad PFN
                  29   13   02C8   930          BEQL    30$                     ; Zero implies none (shouldn't happen)
51   00000000'GF     D0   02CA   931          MOVL    G^PFN$Ax_FLINK,R1       ; Forward link array listhead to R1
52   00000000'GF     D0   02D1   932          MOVL    G^PFN$AB_TYPE,R2        ; PFN STATE array listhead to R2
    02 6240      05   E1   02D8   933 10$:     BBC     #PFN$V_BADPAG,(R2)[R0],20$      ; Is this page really bad?
                  53   D6   02DD   934          INCL    R3                      ; Count another bad page
                            02DF   935 20$:          PFN_REFERENCE   -
                            02DF   936          MOVZWL  <(R1)[R0],R0>,-         ; Follow FLINK to next PFN
                            02DF   937                  LONG_OPCODE=MOVL,-
                            02DF   938                  IMAGE=SHOW_MEMORY
                  E5   12   02F1   939          BNEQ    10$                     ; To top of loop if another PFN
00000030'EF       53   D0   02F3   940 30$:     MOVL    R3,MEM_BAD_PAGES        ; Store the number for output
                  50   01   D0   02FA   941          MOVL    #1,R0                   ; Successful completion of routine
                            04   02FD   942          RET
                            02FE   943
```

```
                     02FE   945          .SBTTL  SHOW SLOT USAGE
                     02FE   946 ;
                     02FE   947 ;+
                     02FE   948 ;
                     02FE   949 ;       THIS ROUTINE DISPLAYS INFORMATION ABOUT THE PROCESS AND BALANCE
                     02FE   950 ;       SLOTS. THE TOTAL NUMBER OF EACH TYPE OF SLOT IS SHOWN. THE NUMBER
                     02FE   951 ;       OF FREE SLOTS IS DISPLAYED. THE SLOTS IN USE ARE BROKEN DOWN INTO
                     02FE   952 ;       TWO CATAGORIES: RESIDENT AND NON-RESIDENT. A NON-RESIDENT BALANCE
                     02FE   953 ;       SLOT IS ONE FOR WHICH NOT ALL OF THE PROCESSES WORKING SET IS
                     02FE   954 ;       RESIDENT (E.G. SWAPPED BUT WAITING FOR I/O)
                     02FE   955 ;-
                     02FE   956
                     02FE   957 SLOTS:
               0000  02FE   958          .WORD   0                               ; Save nothing
                     0300   959          TYPEMSG SHOW$_MEM_SLOT1                 ; Print header line
                     030F   960
                     030F   961 ; Show usage of PCB vector
                     030F   962
                     030F   963          $CMEXEC_S       ROUTIN=SLOTS_PCBVEC     ; Gather the PCB vector data
                     031E   964          TYPEMSG SHOW$_MEM_SLOT2,SHOW_SLOTS_LIST ;  and print it
                     0331   965
                     0331   966 ; Show balance slot usage
                     0331   967
                     0331   968          $CMEXEC_S       ROUTIN=SLOTS_BALANCE    ; Gather the balance slot data
                     0340   969          TYPEMSG SHOW$_MEM_SLOT3,SHOW_SLOTS_LIST ;  and print it
          50   01   3C  0353   970          MOVZWL  #SS$_NORMAL,R0                  ; Load success status
                     04  0356   971          RET                                     ;  and return
                         0357   972
```

```
                                                L 4
                                0357   974          .SUBTITLE       SLOTS_PCBVEC    Compute occupation of PCB vector
                                0357   975
                                0357   976 ;+
                                0357   977 ;       SLOTS_PCBVEC    Compute occupation of PCB vector
                                0357   978 ;
                                0357   979 ;  This routine determines the number of processes that occupy the PCB
                                0357   980 ;  vector and the number of those processes that are currently resident.
                                0357   981 ;
                                0357   982 ;  Calling sequence:
                                0357   983 ;
                                0357   984 ;       CALLS   #0,SLOTS_PCBVEC
                                0357   985 ;
                                0357   986 ;  Input parameter:
                                0357   987 ;
                                0357   988 ;       SCH$GL_PCBVEC   Pointer to PCB vector
                                0357   989 ;
                                0357   990 ;  Output parameters:
                                0357   991 ;
                                0357   992 ;       SLOTS_TOTAL     Number of slots in the vector (MAXPROCESSCNT)
                                0357   993 ;
                                0357   994 ;       SLOTS_FREE      Number of unused slots in the vector
                                0357   995 ;
                                0357   996 ;       SLOTS_RES       Number of slots that are occupied by processes
                                0357   997 ;                       that are resident (PCB$V_RES set in PCB$L_STS)
                                0357   998 ;
                                0357   999 ;       SLOTS_NONRES    Number of slots that are occupied by processes
                                0357  1000 ;                       that are outswapped (PCB$V_RES set in PCB$L_STS)
                                0357  1001 ;
                                0357  1002 ;-
                                0357  1003
                                0357  1004 SLOTS_PCBVEC:
                           003C 0357  1005          .WORD   ^M<R2,R3,R4,R5>          ; Save some registers
     00000060'EF   00000000'GF   3C  0359  1006          MOVZWL  G^SCH$GW_PROCLIM,SLOTS_TOTAL     ; GET TOTAL # OF SLOTS
             52   00000000'GF   3C  0364  1007          MOVZWL  G^SCH$GW_PROCCNT,R2
                    52   02     C0  036B  1008          ADDL2   #2,R2                            ; INCLUDE NULL AND SWAPPER
00000064'EF   00000060'EF   52   C3  036E  1009          SUBL3   R2,SLOTS_TOTAL,SLOTS_FREE        ; GET # OF FREE SLOTS
             52   00000000'GF   D0  037A  1010          MOVL    G^SCH$GL_PCBVEC,R2               ; GET BASE ADDR OF PIX ARRAY
             53   00000000'GF   DE  0381  1011          MOVAL   G^SCH$GL_NULLPCB,R3              ; SAVE NULL PCB
             55   00000000'GF   3C  0388  1012          MOVZWL  G^SCH$GL_SWPPID,R5               ; GET SWAPPER'S PIX
                          55     D6  038F  1013          INCL    R5                               ; START WITH NEXT SLOT
             00000068'EF   55     D0  0391  1014          MOVL    R5,SLOTS_RES                     ; INITIALIZE COUNTS
                   0000006C'EF    D4  0398  1015          CLRL    SLOTS_NONRES
                     54   6245    D0  039E  1016 10$:     MOVL    (R2)[R5],R4                      ; GET PCB ADDRESS
                          53   54  D1  03A2  1017          CMPL    R4,R3                            ; IS THIS THE NULL PCB?
                               16  13  03A5  1018          BEQLU   30$                              ; YES - IGNORE IT
                                   03A7  1019          ASSUME  PCB$V_RES EQ 0
                        08 24 A4   E9  03A7  1020          BLBC    PCB$L_STS(R4),20$                ; CHECK STATUS
                   00000068'EF     D6  03AB  1021          INCL    SLOTS_RES                        ; RESIDENT-BUMP COUNTER
                               0A  11  03B1  1022          BRB     30$
                   0000006C'EF     D6  03B3  1023 20$:     INCL    SLOTS_NONRES                     ; NONRESIDENT-BUMP COUNTER
                     50   23 A4   9A  03B9  1024          MOVZBL  PCB$L_WSSWP+3(R4),R0             ; GET SWAP FILE NUMBER
                D9 55   00000000'GF   F3  03BD  1025 30$:     AOBLEQ  G^SCH$GL_MAXPIX,R5,10$           ; LOOP FOR ALL PIX
                          50   01     3C  03C5  1026          MOVZWL  #SS$_NORMAL,R0
                                      04  03C8  1027          RET
                                          03C9  1028
```

```
                                                M 4

                     03C9  1030          .SUBTITLE       SLOTS_BALANCE   Compute occupation of PCB vector
                     03C9  1031
                     03C9  1032 ;+
                     03C9  1033 ;       SLOTS_BALANCE   Compute occupation of PCB vector
                     03C9  1034 ;
                     03C9  1035 ;  This routine determines the number of processes that occupy the PCB
                     03C9  1036 ;  vector and the number of those processes that are currently resident.
                     03C9  1037 ;
                     03C9  1038 ;  Calling sequence:
                     03C9  1039 ;
                     03C9  1040 ;       CALLS   #0,SLOTS_BALANCE
                     03C9  1041 ;
                     03C9  1042 ;  Input parameters:
                     03C9  1043 ;
                     03C9  1044 ;       SCH$GL_PCBVEC   Pointer to PCB vector
                     03C9  1045 ;       PHV$GL_PIXBAS   Address of process index array associated with
                     03C9  1046 ;                       process header vector
                     03C9  1047 ;
                     03C9  1048 ;  Output parameters:
                     03C9  1049 ;
                     03C9  1050 ;       SLOTS_TOTAL     Number of balance slots (BALSETCNT)
                     03C9  1051 ;
                     03C9  1052 ;       SLOTS_FREE      Number of unused blance slots
                     03C9  1053 ;
                     03C9  1054 ;       SLOTS_RES       Number of balance slots that are occupied by processes
                     03C9  1055 ;                       that are resident (PCB$V_PHDRES set in PCB$L_STS)
                     03C9  1056 ;
                     03C9  1057 ;       SLOTS_NONRES    Number of balance slots that are occupied by processes
                     03C9  1058 ;                       that are outswapped (PCB$V_PHDRES set in PCB$L_STS)
                     03C9  1059 ;                       An outswapped process that still occupies a balance
                     03C9  1060 ;                       slot is a process whose process body is outswapped
                     03C9  1061 ;                       but whose process header is still resident.
                     03C9  1062 ;-
                     03C9  1063
                     03C9  1064 SLOTS_BALANCE:
                 003C  03C9  1065          .WORD   ^M<R2,R3,R4,R5>                         ; Save some registers
00000060'EF  00000000'GF  D0  03CB  1066          MOVL    G^SGN$GL_BALSETCT,SLOTS_TOTAL   ; GET # OF SLOTS
             00000064'EF  D4  03D6  1067          CLRL    SLOTS_FREE                      ; INITIALIZE COUNTERS
             00000068'EF  D4  03DC  1068          CLRL    SLOTS_RES
             0000006C'EF  D4  03E2  1069          CLRL    SLOTS_NONRES
          55 00000000'GF  D0  03E8  1070          MOVL    G^SCH$GL_PCBVEC,R5              ; GET BASE OF PCB ADDRS
          52 00000000'GF  D0  03EF  1071          MOVL    G^PHV$GL_PIXBAS,R2              ; GET BASE OF PIX ARRAY
                    53    D4  03F6  1072          CLRL    R3                              ; START AT SLOT 0
             54    6243    32  03F8  1073 10$:     CVTWL   (R2)[R3],R4                     ; GET PIX POINTER
                     08    14  03FC  1074          BGTR    20$                             ; IS SLOT IN USE?
             00000064'EF  D6  03FE  1075          INCL    SLOTS_FREE                      ; NO - COUNT IT AS FREE
                     16    11  0404  1076          BRB     40$                             ; AND CONTINUE
             54    6544    D0  0406  1077 20$:     MOVL    (R5)[R4],R4                     ; GET PCB ADDRESS
                           040A  1078          ASSUME  PCB$V_RES EQ 0
                08 24 A4   E9  040A  1079          BLBC    PCB$L_STS(R4),30$               ; IS PROCESS RESIDENT?
             00000068'EF  D6  040E  1080          INCL    SLOTS_RES                       ; YES-COUNT IT AS SUCH
                     06    11  0414  1081          BRB     40$
             0000006C'EF  D6  0416  1082 30$:     INCL    SLOTS_NONRES                    ; NO-COUNT AS NON-RES
       D4 53 00000060'EF  F2  041C  1083 40$:     AOBLSS  SLOTS_TOTAL,R3,10$              ; LOOP FOR ALL PIX
                  50    01  3C  0424  1084          MOVZWL  #SS$_NORMAL,R0
                              04  0427  1085          RET
                              0428  1086
```

```
                     0428  1088          .SUBTITLE       LOOKASIDE - Display Routine for Lookaside Lists
                     0428  1089
                     0428  1090 ;-
                     0428  1091 ; Functional Description:
                     0428  1092 ;
                     0428  1093 ;       This routine displays nonpaged pool statistics for fixed-size block
                     0428  1094 ;       lists. These include the small packet (SRP) lookaside list, the I/O
                     0428  1095 ;       request packet (IRP) list, and the large packet (LRP) lookaside list.
                     0428  1096 ;
                     0428  1097 ; Input Parameters:
                     0428  1098 ;
                     0428  1099 ;       None
                     0428  1100 ;
                     0428  1101 ; Implicit Input:
                     0428  1102 ;
                     0428  1103 ;       Listheads for three lookaside lists
                     0428  1104 ;
                     0428  1105 ; Output Parameters:
                     0428  1106 ;
                     0428  1107 ;       None
                     0428  1108 ;
                     0428  1109 ; Implicit Output:
                     0428  1110 ;
                     0428  1111 ;       Three lookaside list displays are written to SYS$OUTPUT
                     0428  1112 ;-
                     0428  1113
                     0428  1114 LOOKASIDE:
                000C 0428  1115          .WORD   ^M<R2,R3>                       ; Save some registers
    OF 00000008'EF   04  E0  042A  1116          BBS     #MEMORY_V_FULL,MEMORY_L_BITLIS,10$ ; Skip header in full display
                     0432  1117          TYPEMSG SHOW$_MEM_LOOK1                 ; Print header line
                     0441  1118
                     0441  1119 ;  Get fixed-length nonpaged pool statistics. Do small packet (SRP)
                     0441  1120 ;  lookaside list first.
                     0441  1121
000000C8'EF   00000000'GF  DE  0441  1122 10$:     MOVAL   G^IOC$GL_SRPFL,LOOK_CMKRNL_ARGLIST+XRPFL ; Listhead address
                     044C  1123          $CMKRNL_S       =                       ; Scan the list
                     044C  1124                  ROUTIN=LOOK_XRPLIST,-
                     044C  1125                  ARGLST=LOOK_CMKRNL_ARGLIST
00000098'EF   00000096'EF  3E  045F  1126          MOVAW   SRPLIST_DESC,LOOK_LIST_NAME ; Add an identifier
0000009C'EF   00000000'GF  D0  046A  1127          MOVL    G^IOC$GL_SRPCNT,LOOK_LIST_SIZE ; Get current list size
         52   00000000'GF  D0  0475  1128          MOVL    G^IOC$GL_SRPSIZE,R2     ; Pass block size in R2
000000B8'EF   000000B0'EF  3E  047C  1129          MOVAW   SRP_SIZE_DESC,LOOK_SIZE_DESC ; SYSGEN parameter name for size
000000C0'EF   00000000'GF  D0  0487  1130          MOVL    G^IOC$GL_SRPMIN,LOOK_BLOCK_MIN ; Lower limit for allocation
                     0492  1131
         53  000000CC'EF  DE  0492  1132          MOVAL   LOOK_SIZE_ARRAY,R3      ; Address of auxiliary array
         63  0000008B'EF  3E  0499  1133          MOVAW   SRP_NAME_DESC,(R3)      ; Descriptor for list name
      04 A3  00000000'GF  D0  04A0  1134          MOVL    G^SGN$GL_SRPCNT,4(R3)   ; Initial list size
      08 A3  00000000'GF  D0  04A8  1135          MOVL    G^SGN$GL_SRPCNTV,8(R3)  ; Maximum list size
                 011B     30  04B0  1136          BSBW    DISPLAY_LOOK            ; Display SRP statistics
                     04B3  1137
                     04B3  1138 ; Gather statistics for I/O Request Packet (IRP) Lookaside List
                     04B3  1139
000000C8'EF   00000000'GF  DE  04B3  1140          MOVAL   G^IOC$GL_IRPFL,LOOK_CMKRNL_ARGLIST+XRPFL ; Listhead address
                     04BE  1141          $CMKRNL_S       =                       ; Scan the list
                     04BE  1142                  ROUTIN=LOOK_XRPLIST,-
                     04BE  1143                  ARGLST=LOOK_CMKRNL_ARGLIST
00000098'EF   000000CA'EF  3E  04D1  1144          MOVAW   IRPLIST_DESC,LOOK_LIST_NAME ; Add an identifier
```

```
0000009C'EF   00000000'GF   D0   04DC  1145          MOVL    G^IOC$GL_IRPCNT,LOOK_LIST_SIZE  ; Get current list size
        52    000000C4'8F   D0   04E7  1146          MOVL    #<IRP$K_LENGTH+EXE$C_ALCGRNMSK>&^C<EXE$C_ALCGRNMSK>,R2
                            04EE  1147                                                  ; Pass block size in R2
000000B8'EF   000000EA'EF   3E   04EE  1148          MOVAW   IRP_SIZE_DESC,LOOK_SIZE_DESC    ; Descriptor for "fixed"
000000C0'EF   00000000'GF   D0   04F9  1149          MOVL    G^IOC$GL_IRPMIN,LOOK_BLOCK_MIN  ; Lower limit for allocation
                            0504  1150
        53    000000CC'EF   DE   0504  1151          MOVAL   LOOK_SIZE_ARRAY,R3              ; Address of auxiliary array
        63    000000BF'EF   3E   050B  1152          MOVAW   IRP_NAME_DESC,(R3)              ; Descriptor for list name
     04 A3    0C000000'GF   D0   0512  1153          MOVL    G^SGN$GL_IRPCNT,4(R3)           ; Initial list size
     08 A3    00000000'GF   D0   051A  1154          MOVL    G^SGN$GL_IRPCNTV,8(R3)          ; Maximum list size
                     00A9   30   0522  1155          BSBW    DISPLAY_LOOK                    ; Display IRP statistics
                            0525  1156
                            0525  1157 ;  Finally, perform the same steps for the large packet (LRP) lookaside list
                            0525  1158
000000C8'EF   00000000'GF   DE   0525  1159          MOVAL   G^IOC$GL_LRPFL,LOOK_CMKRNL_ARGLIST+XRPFL         ; Listhead address
                            0530  1160          $CMKRNL_S       =                       ; Scan the list
                            0530  1161                  ROUTIN=LOOK_XRPLIST,-
                            0530  1162                  ARGLST=LOOK_CMKRNL_ARGLIST
00000098'EF   00000102'EF   3E   0543  1163          MOVAW   LRPLIST_DESC,LOOK_LIST_NAME     ; Add an identifier
0000009C'EF   00000000'GF   D0   054E  1164          MOVL    G^IOC$GL_LRPCNT,LOOK_LIST_SIZE  ; Get current list size
        52    00000000'GF   D0   0559  1165          MOVL    G^IOC$GL_LRPSIZE,R2             ; Pass block size in R2
000000B8'EF   0000011C'EF   3E   0560  1166          MOVAW   LRP_SIZE_DESC,LOOK_SIZE_DESC    ; Descriptor for "LRPSIZE + 64"
000000C0'EF   00000000'GF   D0   056B  1167          MOVL    G^IOC$GL_LRPMIN,LOOK_BLOCK_MIN  ; Lower limit for allocation
                            0576  1168
        53    000000CC'EF   DE   0576  1169          MOVAL   LOOK_SIZE_ARRAY,R3              ; Address of auxiliary array
        63    000000F7'EF   3E   057D  1170          MOVAW   LRP_NAME_DESC,(R3)              ; Descriptor for list name
     04 A3    00000000'GF   D0   0584  1171          MOVL    G^SGN$GL_LRPCNT,4(R3)           ; Initial list size
     08 A3    00000000'GF   D0   058C  1172          MOVL    G^SGN$GL_LRPCNTV,8(R3)          ; Maximum list size
                     0037   30   0594  1173          BSBW    DISPLAY_LOOK                    ; Display LRP statistics
                            0597  1174
              50    01      3C   0597  1175          MOVZWL  #SS$_NORMAL,R0                  ; Signal success
                            04   059A  1176          RET                                     ;  and return
                            059B  1177
```

```
                                                C 5
                     059B  1179          .SUBTITLE       POOL_XRPLIST    Scan a Lookaside List
                     059B  1180
                     059B  1181 ;+
                     059B  1182 ; Functional Description:
                     059B  1183 ;
                     059B  1184 ;       This routine counts the number of free blocks on the lookaside
                     059B  1185 ;       list pointed to by the input parameter.
                     059B  1186 ;
                     059B  1187 ;  Calling sequence:
                     059B  1188 ;
                     059B  1189 ;       CALLS   #1,POOL_IRPLIST
                     059B  1190 ;
                     059B  1191 ;  Input parameter:
                     059B  1192 ;
                     059B  1193 ;       XRPFL(AP)       Listhead of doubly linked list
                     059B  1194 ;
                     059B  1195 ;  Output parameter:
                     059B  1196 ;
                     059B  1197 ;       LOOK_FREE_COUNT Number of free blocks in this list
                     059B  1198 ;-
                     059B  1199
                     059B  1200 BEGIN_LOCKED_CODE:                      ; The following code executes above IPL 2
                     059B  1201
                     059B  1202 LOOK_XRPLIST:
                000C 059B  1203          .WORD   ^M<R2,R3>                      ; Save some registers
          52  04 AC   D0  059D  1204          MOVL    XRPFL(AP),R2           ; Get address of forward link
                     05A1  1205          DSBINT  G^EXE$GL_NONPAGED      ; Set IPL for pool access
              000E   30  05AB  1206          BSBW    SCAN_DOUBLY_LINKED_LIST ; Count number of blocks in list
                     05AE  1207          ENBINT                         ; Enable interrupts
000000A8'EF   53   D0  05B1  1208          MOVL    R3,LOOK_FREE_COUNT     ; Store number of free blocks
        50   01   3C  05B8  1209          MOVZWL  #SS$_NORMAL,R0
                     05BB  1210          RET
                     05BC  1211
```

```
                        05BC  1213          .SUBTITLE       SCAN_DOUBLY_LINKED_LIST Scan doubly linked list
                        05BC  1214
                        05BC  1215 ;+
                        05BC  1216 ;       SCAN_DOUBLY_LINKED_LIST Scan a of fixed-sized blocks
                        05BC  1217 ;
                        05BC  1218 ;  This routine scans a doubly linked list of fixed-size blocks and
                        05BC  1219 ;  returns the number of blocks in the list.
                        05BC  1220 ;
                        05BC  1221 ;  Calling sequence:
                        05BC  1222 ;
                        05BC  1223 ;       BSBW    SCAN_DOUBLY_LINKED_LIST
                        05BC  1224 ;
                        05BC  1225 ;  Input parameter:
                        05BC  1226 ;
                        05BC  1227 ;       R2      Address of listhead for list
                        05BC  1228 ;
                        05BC  1229 ;  Output parameter:
                        05BC  1230 ;
                        05BC  1231 ;       R3      Number of blocks in list
                        05BC  1232 ;
                        05BC  1233 ;  Side effect:
                        05BC  1234 ;
                        05BC  1235 ;       The contents of R1 are modified
                        05BC  1236 ;
                        05BC  1237 ;  This routine assumes that the caller has taken whatever synchronization
                        05BC  1238 ;  measures are necessary for the pool area being scanned.
                        05BC  1239 ;-
                        05BC  1240
                        05BC  1241 SCAN_DOUBLY_LINKED_LIST:
        53    D4        05BC  1242          CLRL    R3                      ; Set counter to zero
51      52    D0        05BE  1243          MOVL    R2,R1                   ; Make a working copy
51      61    D0        05C1  1244 10$:     MOVL    (R1),R1                 ; Get address of next block
52      51    D1        05C4  1245          CMPL    R1,R2                   ; At end of list yet?
        04    13        05C7  1246          BEQL    20$                     ; Equal implies end of list
        53    D6        05C9  1247          INCL    R3                      ; Indicate another block
        F4    11        05CB  1248          BRB     10$                     ;  and go get the next one
                        05CD  1249
              05        05CD  1250 20$:     RSB                             ; Return to caller
                        05CE  1251
```

```
                                        .SUBTITLE       DISPLAY_LOOK Output Routine for Lookaside List Displays
                                ;+
                                ; Functional Description:
                                ;
                                ;       This routine performs the common output and display functions for
                                ;       the three fixed-sized dynamic memory areas. The routine decides
                                ;       whether a normal or full display is requested.
                                ;
                                ; Calling Sequence:
                                ;
                                ;       BSBW    DISPLAY_LOOK
                                ;
                                ; Input Parameters:
                                ;
                                ;       R2      Size of packets in this list
                                ;
                                ;       R3      Address of three-longword array containing information
                                ;               that describes the initial and maximum sizes of the list
                                ;
                                ; Implicit Input:
                                ;
                                ;       Setting of MEMORY_V_FULL bit in MEMORY_L_BITLIS
                                ;
                                ;       Contents of cells in FAO parameter list for lookaside list displays
                                ;
                                ; Output Parameters:
                                ;
                                ;       Several cells in FAO parameter list for lookaside list displays
                                ;
                                ;               LOOK_LIST_SIZE          Size in packets, bytes, and pages
                                ;               LOOK_FREE_BYTES         /FULL display only
                                ;               LOOK_INUSE_COUNT
                                ;               LOOK_INUSE_BYTES        /FULL display only
                                ;               LOOK_BLOCK_SIZE         Passed into this routine in R2
                                ;
                                ; Implicit Output:
                                ;
                                ;       Displays of usage statistics for specified lookaside list
                                ;       are written to SYS$OUTPUT.
                                ;-
                                
                                DISPLAY_LOOK:
000000BC'EF   52    D0          MOVL    R2,LOOK_BLOCK_SIZE      ; Store block size in parameter list
0000009C'EF 000000A8'EF C3      SUBL3   LOOK_FREE_COUNT,LOOK_LIST_SIZE,LOOK_INUSE_COUNT
            000000B0'EF
      14 00000008'EF   04   E0  BBS     #MEMORY_V_FULL,MEMORY_L_BITLIS,10$      ; Was /FULL specified?
                            05  TYPEMSG SHOW$_MEM_LOOK2,SHOW_LOOK_LIST  ; No. Type normal display line
                                RSB                                     ;  and return to caller
                                
      51   0000009C'EF   DE     MOVAL   LOOK_LIST_SIZE,R1       ; Store address of size array
                  00BE   30     BSBW    CONVERT_PACKET_COUNT    ; Convert packets to bytes and pages
           81   81   52   C5   MULL3   R2,(R1)+,(R1)+           ; Convert free packets to free bytes
           81   81   52   C5   MULL3   R2,(R1)+,(R1)+           ; Convert packets in use to bytes in use
                               TYPEMSG SHOW$_MEM_LOOK_FULL1,SHOW_LOOK_LIST     ; Display name of list
                               TYPEMSG SHOW$_MEM_LOOK_FULL2,SHOW_LOOK_LIST2    ; Display current size
      51   00000098'EF   DE     MOVAL   LOOK_LIST_NAME,R1       ; Use first four parameters again
```

```
                81   83    D0  0640  1309          MOVL    (R3)+,(R1)+             ; Store SYSGEN parameter name
                61   83    D0  0643  1310          MOVL    (R3)+,(R1)              ;  and initial size
                   0080    30  0646  1311          BSBW    CONVERT_PACKET_COUNT    ; Convert packet count to bytes and pages
                           0649  1312          TYPEMSG SHOW$_MEM_LOOK_FULL3,SHOW_LOOK_LIST3    ; Display initial size
           51   0000009C'EF  DE  065C  1313          MOVAL   LOOK_LIST_SIZE,R1       ; Reset size array pointer
                61   83    D0  0663  1314          MOVL    (R3)+,(R1)              ; Store maximum size
                   0060    30  0666  1315          BSBW    CONVERT_PACKET_COUNT    ; Convert packets to bytes and pages
                           0669  1316          TYPEMSG SHOW$_MEM_LOOK_FULL4,SHOW_LOOK_LIST4    ; Display maximum size
                           067C  1317
                           067C  1318          TYPEMSG SHOW$_MEM_LOOK_FULL5,SHOW_LOOK_LIST5    ; Display free space
                           068F  1319          TYPEMSG SHOW$_MEM_LOOK_FULL6,SHOW_LOOK_LIST6    ; Display space in use
                           06A2  1320          TYPEMSG SHOW$_MEM_LOOK_FULL7,SHOW_LOOK_LIST7    ; Display block size
                           06B5  1321          TYPEMSG SHOW$_MEM_LOOK_FULL8,SHOW_LOOK_LIST8    ; Display lower limit
                           06C8  1322
                      05   06C8  1323          RSB
                           06C9  1324
```

```
                     06C9  1326          .SUBTITLE       CONVERT_PACKET_COUNT    Convert Packets to Bytes and Pages
                     06C9  1327
                     06C9  1328 ;+
                     06C9  1329 ; Functional Description:
                     06C9  1330 ;
                     06C9  1331 ;       This routine converts a packet count and a packet size to a byte
                     06C9  1332 ;       count and the minimum number of pages required to hold that
                     06C9  1333 ;       number of bytes.
                     06C9  1334 ;
                     06C9  1335 ; Input Parameters:
                     06C9  1336 ;
                     06C9  1337 ;       R1      Address of 3-longword array of sizes
                     06C9  1338 ;       (R1)    Number of packets
                     06C9  1339 ;       R2      Packet size
                     06C9  1340 ;
                     06C9  1341 ; Output Parameters:
                     06C9  1342 ;
                     06C9  1343 ;       4(R1)   Byte count (packets * packet size)
                     06C9  1344 ;       8(R1)   Page count necessary to contain byte count
                     06C9  1345 ;
                     06C9  1346 ; Implicit Output:
                     06C9  1347 ;
                     06C9  1348 ;       R1 points at the next longword after the page count
                     06C9  1349 ;
                     06C9  1350 ; Side Effects:
                     06C9  1351 ;
                     06C9  1352 ;       R0 is destroyed by this routine
                     06C9  1353 ;-
                     06C9  1354
                     06C9  1355 CONVERT_PACKET_COUNT:
          61    81   52  C5  06C9  1356          MULL3   R2,(R1)+,(R1)           ; Convert packets to bytes
     50   81  000001FF 8F  C1  06CD  1357          ADDL3   #511,(R1)+,R0           ; Round up to next page boundary
          81    50   F7 8F  78  06D5  1358          ASHL    #-9,R0,(R1)+            ; Convert bytes to pages
                           05  06DA  1359          RSB
                     06DB  1360
```

```
                                        06DB  1362          .SBTTL  SHOW POOL USAGE
                                        06DB  1363 ;+
                                        06DB  1364 ;       SHOW PAGED AND NON-PAGED POOL USAGE
                                        06DB  1365 ;
                                        06DB  1366 ;       THIS CODE MUST NOT PAGEFAULT WHILE AT ELEVATED IPL; THEREFORE
                                        06DB  1367 ;       IT (AND THE DATA ITEMS IT REFERENCES) ARE LOCKED IN THE WORKING
                                        06DB  1368 ;       SET PRIOR TO THE ROUTINE BEING CALLED.
                                        06DB  1369 ;
                                        06DB  1370 ;       THIS ROUTINE DISPLAYS THE TOTAL NUMBER OF BYTES IN EACH POOL,
                                        06DB  1371 ;       THE NUMBER OF BYTES IN USE, AND THE NUMBER OF FREE BYTES.
                                        06DB  1372 ;       THE NON-PAGED POOL IS SUBDIVIDED INTO THE FIXED LENGTH LOOKASIDE
                                        06DB  1373 ;       LISTS AND THE VARIABLE-LENGTH SEGMENTS. THE FIXED LENGTH NON-PAGED
                                        06DB  1374 ;       POOL IS SUBDIVIDED INTO IRP PACKETS AND BIG BLOCKS.
                                        06DB  1375 ;-
                                        06DB  1376
                                        06DB  1377 POOL:
                                  0004  06DB  1378          .WORD   ^M<R2>                                  ; Save R2
        0F 00000008'EF    04    E0    06DD  1379          BBS     #MEMORY_V_FULL,MEMORY_L_BITLIS,10$ ; Skip header in full display
                                        06E5  1380          TYPEMSG SHOW$_MEM_POOL1                         ; Print header line
                                        06F4  1381
                                        06F4  1382 ;  Get variable length nonpaged pool statistics
                                        06F4  1383
                                        06F4  1384 10$:     $CMKRNL_S       ROUTIN=POOL_NPAGEDYN        ; Scan the list ...
    00000070'EF    00000000'EF    3E    0703  1385          MOVAW   L^NPAGEDYN_DESC,L^POOL_NAME          ;  add a name identifier,
    00000074'EF    0000006F'EF    3E    070E  1386          MOVAW   L^BYTES_SIZE_DESC,L^POOL_SIZE       ;  and a size identifier.
7E  00000000'GF    000001FF  8F    CB    0719  1387          BICL3   #^X1FF,G^MMG$GL_NPAGNEXT,-(SP)    ; Get current end of pool
        50    8E   00000000'GF    C3    0725  1388          SUBL3   G^MMG$GL_NPAGEDYN,(SP)+,R0           ; Compute size of nonpaged pool
                          52    01    9A    072D  1389          MOVZBL  #1,R2                               ; Indicate nonpaged pool
                        017B    30    0730  1390          BSBW    DISPLAY_POOL                        ;  and print this information
                                        0733  1391
                                        0733  1392 ;  Get paged pool statistics
                                        0733  1393
                                        0733  1394          $CMKRNL_S       ROUTIN=POOL_PAGEDYN         ; Scan the list ...
00000070'EF    00000025'EF    3E    0742  1395          MOVAW   L^PAGEDYN_DESC,L^POOL_NAME          ;  add a name identifier,
00000074'EF    0000007C'EF    3E    074D  1396          MOVAW   L^PAGEDYN_SIZE_DESC,L^POOL_SIZE     ;  and a size identifier.
        50    00000000'GF    D0    0758  1397          MOVL    G^SGN$GL_PAGEDYN,R0                 ; Get total pool size
                          52    D4    075F  1398          CLRL    R2                                  ; Indicate not nonpaged pool
                        014A    30    0761  1399          BSBW    DISPLAY_POOL                        ;  and print the information
                                04    0764  1400          RET                                         ; That's all for SHOW MEMORY
                                        0765  1401
                                        0765  1402 ; Get statistics for process allocation region if /MEMORY qualifier
                                        0765  1403 ; was specified to the SHOW PROCESS command
                                        0765  1404
                                        0765  1405 SHOW$PRCALLREG::
                                  0004  0765  1406          .WORD   ^M<R2>                              ; Save volatile register
     00 00000008'EF    04    E2    0767  1407          BBSS    #MEMORY_V_FULL,MEMORY_L_BITLIS,20$      ; Always a full display
                                        076F  1408 20$:     $CMKRNL_S       ROUTIN=POOL_PRCALLREG       ; Scan the list ...
00000070'EF    0000004A'EF    3E    077E  1409          MOVAW   L^PRCALLREG_DESC,L^POOL_NAME        ;  add a name identifier,
00000074'EF    0000006F'EF    3E    0789  1410          MOVAW   L^BYTES_SIZE_DESC,L^POOL_SIZE       ;  and a size identifier.
        50    00000000'GF    3C    0794  1411          MOVZWL  G^SGN$GQ_CTLPAGES,R0                ; Calculate total size
                    50    50    09    78    079B  1412          ASHL    #9,R0,R0                            ; Convert to bytes
                          52    D4    079F  1413          CLRL    R2                                  ; Indicate not nonpaged pool
                        010A    30    07A1  1414          BSBW    DISPLAY_POOL                        ;  and print the information
                      50    01    3C    07A4  1415          MOVZWL  #SS$_NORMAL,R0                      ; Signal success
                                04    07A7  1416          RET                                         ; Return to caller
                                        07A8  1417
```

```
                                                .SUBTITLE       POOL_NPAGEDYN   Scan Nonpaged Dynamic Memory

                                                ;+
                                                ;       POOL_NPAGEDYN   Scan Nonpaged Dynamic Memory
                                                ;
                                                ;  This routine scans nonpaged pool and returns current usage information.
                                                ;
                                                ;  Calling sequence:
                                                ;
                                                ;       CALLS   #0,POOL_NPAGEDYN
                                                ;
                                                ;  Input parameters:
                                                ;
                                                ;       EXE$GL_NONPAGED Listhead of paged pool
                                                ;
                                                ;  Output parameters:
                                                ;
                                                ;       POOL_TOTAL      Total amount of space set aside for this area
                                                ;       POOL_FREE       Total amount of unallocated (free) space
                                                ;       POOL_INUSE      Amount of space currently in use (TOTAL - FREE)
                                                ;       POOL_FREE_COUNT Number of discontiguous free blocks
                                                ;       POOL_MAX_BLOCK  Size of largest contiguous area
                                                ;       POOL_MIN_BLOCK  Size of smallest unallocated block
                                                ;-

                                                POOL_NPAGEDYN:
                                        00FC    .WORD   ^M<R2,R3,R4,R5,R6,R7>   ; Save some registers
52  00000000'GF                          DE      MOVAL   G^EXE$GL_NONPAGED,R2    ; Get nonpaged pool listhead
                                                DSBINT  (R2)+                   ; Set IPL for pool access
                                        00BD    BSBW    SCAN_SINGLY_LINKED_LIST ; Get free space, minimum, and maximum
                                                ENBINT                          ; Allow interrupts
00000090'EF                             53      MOVL    R3,POOL_FREE_COUNT      ; Save total number of free blocks,
00000094'EF                             54      MOVL    R4,POOL_FREE_LEQU_32    ;  count of blocks 32 bytes or smaller,
00000080'EF                             55      MOVL    R5,POOL_FREE            ;  total number of free bytes,
00000088'EF                             56      MOVL    R6,POOL_MAX_BLOCK       ;  size of maximum block,
0000008C'EF                             57      MOVL    R7,POOL_MIN_BLOCK       ;  and size of minimum block
                                        50  01  3C      MOVZWL  #SS$_NORMAL,R0
                                                04      RET
                                                
```

```
                                                J 5

                        07E4  1464          .SUBTITLE       POOL_PAGEDYN    Scan Paged Dynamic Memory
                        07E4  1465
                        07E4  1466 ;+
                        07E4  1467 ;       POOL_PAGEDYN    Scan Paged Dynamic Memory
                        07E4  1468 ;
                        07E4  1469 ;  This routine scans paged pool and returns current usage information.
                        07E4  1470 ;
                        07E4  1471 ;  Calling sequence:
                        07E4  1472 ;
                        07E4  1473 ;       CALLS   #0,POOL_PAGEDYN
                        07E4  1474 ;
                        07E4  1475 ;  Input parameters:
                        07E4  1476 ;
                        07E4  1477 ;       EXE$GL_PAGED    Listhead of paged pool
                        07E4  1478 ;
                        07E4  1479 ;  Output parameters:
                        07E4  1480 ;
                        07E4  1481 ;       POOL_TOTAL      Total amount of space set aside for this area
                        07E4  1482 ;
                        07E4  1483 ;       POOL_FREE       Total amount of unallocated (free) space
                        07E4  1484 ;
                        07E4  1485 ;       POOL_INUSE      Amount of space currently in use (TOTAL - FREE)
                        07E4  1486 ;
                        07E4  1487 ;       POOL_FREE_COUNT Number of discontiguous free blocks
                        07E4  1488 ;
                        07E4  1489 ;       POOL_MAX_BLOCK  Size of largest contiguous area
                        07E4  1490 ;
                        07E4  1491 ;       POOL_MIN_BLOCK  Size of smallest unallocated block
                        07E4  1492 ;-
                        07E4  1493
                        07E4  1494 POOL_PAGEDYN:
                   00FC 07E4  1495          .WORD   ^M<R2,R3,R4,R5,R6,R7>   ; Save some registers
                        07E6  1496          SAVIPL                          ; Save current IPL
50  00000000'GF   9E    07E9  1497          MOVAB   G^EXE$GL_PGDYNMTX,R0    ; Get address of paged memory mutex
54  00000000'GF   D0    07F0  1498          MOVL    G^SCH$GL_CURPCB,R4      ; Get current process PCB address
          11      BB    07F7  1499          PUSHR   #^M<R0,R4>              ; Save these for UNLOCK call
    00000000'GF   16    07F9  1500          JSB     G^SCH$LOCKR             ; Lock paged pool data base
                        07FF  1501                                          ;;;  returns at ASTDEL
52  00000000'GF   DE    07FF  1502          MOVAL   G^EXE$GL_PAGED,R2       ;;; Get header link for free list
              006E  30    0806  1503          BSBW    SCAN_SINGLY_LINKED_LIST ;;; Get free space, minimum, and maximum
00000090'EF   53    D0    0809  1504          MOVL    R3,POOL_FREE_COUNT      ;;; Save total number of free blocks,
00000094'EF   54    D0    0810  1505          MOVL    R4,POOL_FREE_LEQU_32    ;;;  count of blocks 32 bytes or smaller,
00000080'EF   55    D0    0817  1506          MOVL    R5,POOL_FREE            ;;;  total number of free bytes,
00000088'EF   56    D0    081E  1507          MOVL    R6,POOL_MAX_BLOCK       ;;;  size of maximum block,
0000008C'EF   57    D0    0825  1508          MOVL    R7,POOL_MIN_BLOCK       ;;;  and size of minimum block
          11      BA    082C  1509          POPR    #^M<R0,R4>              ;;; Restore mutex address and PCB address
    00000000'GF   16    082E  1510          JSB     G^SCH$UNLOCK            ;;; Unlock the data base
                        0834  1511          ENBINT                          ;;; Return to original IPL
          50    01    3C    0837  1512          MOVZWL  #SS$_NORMAL,R0          ; Return SUCCESS status to caller
                      04    083A  1513          RET                             ;
                        083B  1514
```

```
                                                K 5

                     083B  1516          .SUBTITLE       POOL_PRCALLREG  Scan Process Allocation Region
                     083B  1517
                     083B  1518 ;+
                     083B  1519 ;       POOL_PRCALLREG  Scan Process Allocation Region
                     083B  1520 ;
                     083B  1521 ;  This routine scans the process allocation region, a process-private
                     083B  1522 ;  pool area in P1 space, and returns current usage information.
                     083B  1523 ;
                     083B  1524 ;  Calling sequence:
                     083B  1525 ;
                     083B  1526 ;       CALLS   #0,POOL_PRCALLREG
                     083B  1527 ;
                     083B  1528 ;  Input parameters:
                     083B  1529 ;
                     083B  1530 ;       CTL$GQ_ALLOCREG Listhead of process allocation region
                     083B  1531 ;
                     083B  1532 ;  Output parameters:
                     083B  1533 ;
                     083B  1534 ;       POOL_TOTAL      Total amount of space set aside for this area
                     083B  1535 ;
                     083B  1536 ;       POOL_FREE       Total amount of unallocated (free) space
                     083B  1537 ;
                     083B  1538 ;       POOL_INUSE      Amount of space currently in use (TOTAL - FREE)
                     083B  1539 ;
                     083B  1540 ;       POOL_FREE_COUNT Number of discontiguous free blocks
                     083B  1541 ;
                     083B  1542 ;       POOL_MAX_BLOCK  Size of largest contiguous area
                     083B  1543 ;
                     083B  1544 ;       POOL_MIN_BLOCK  Size of smallest unallocated block
                     083B  1545 ;
                     083B  1546 ;-
                     083B  1547
                     083B  1548 POOL_PRCALLREG:
                 00FC 083B  1549          .WORD   ^M<R2,R3,R4,R5,R6,R7>   ; Save some registers
52  00000000'9F      DE 083D  1550          MOVAL   @#CTL$GQ_ALLOCREG,R2    ; Get listhead for this pool area
                     0844  1551          DSBINT  #IPL$_ASTDEL            ; Prevent ASTs while scanning this list
                  002A   30 084A  1552          BSBW    SCAN_SINGLY_LINKED_LIST ; Get free space, minimum, and maximum
                     084D  1553          ENBINT                          ; ASTs are OK now
00000090'EF       53   D0 0850  1554          MOVL    R3,POOL_FREE_COUNT      ; Save total number of free blocks,
00000094'EF       54   D0 0857  1555          MOVL    R4,POOL_FREE_LEQU_32    ;  count of blocks 32 bytes or smaller,
00000080'EF       55   D0 085E  1556          MOVL    R5,POOL_FREE            ;  total number of free bytes,
00000088'EF       56   D0 0865  1557          MOVL    R6,POOL_MAX_BLOCK       ;  size of maximum block,
0000008C'EF       57   D0 086C  1558          MOVL    R7,POOL_MIN_BLOCK       ;  and size of minimum block
              50  01   3C 0873  1559          MOVZWL  #SS$_NORMAL,R0
                   04   0876  1560          RET
                        0877  1561
```

```
                     0877  1563          .SUBTITLE       SCAN_SINGLY_LINKED_LIST Scan memory-ordered list
                     0877  1564
                     0877  1565 ;+
                     0877  1566 ;  Functional Description:
                     0877  1567 ;
                     0877  1568 ;       This routine scans a memory-ordered singly linked list of blocks and
                     0877  1569 ;       returns the total amount of free space , the number of free blocks,
                     0877  1570 ;       the number of free blocks 32 bytes or smaller, and the sizes of the
                     0877  1571 ;       largest and smallest blocks.
                     0877  1572 ;
                     0877  1573 ;  Calling sequence:
                     0877  1574 ;
                     0877  1575 ;       BSBW    SCAN_SINGLY_LINKED_LIST
                     0877  1576 ;
                     0877  1577 ;  Input parameter:
                     0877  1578 ;
                     0877  1579 ;       R2      Address of listhead for pool area.
                     0877  1580 ;
                     0877  1581 ;  Output parameters:
                     0877  1582 ;
                     0877  1583 ;       R3      Number of distinct free blocks
                     0877  1584 ;       R4      Number of free blocks 32 bytes or smaller
                     0877  1585 ;       R5      Total amount of free space
                     0877  1586 ;       R6      Size of largest block
                     0877  1587 ;       R7      Size of smallest block
                     0877  1588 ;
                     0877  1589 ;  This routine assumes that the caller has taken whatever synchronization
                     0877  1590 ;  measures are necessary for the pool area being scanned.
                     0877  1591 ;-
                     0877  1592
                     0877  1593 SCAN_SINGLY_LINKED_LIST:
          53    7C   0877  1594          CLRQ    R3                      ; Clear two free block counters
          55    7C   0879  1595          CLRQ    R5                      ; Set sum and maximum to zero
       57    00    D2   087B  1596          MCOML   #0,R7                   ; Set minimum to "infinite"
       52    62    D0   087E  1597          MOVL    (R2),R2                 ; Get contents of first block
          28    13   0881  1598          BEQL    40$                     ; If zero, then pool is empty
          53    D6   0883  1599 10$:     INCL    R3                      ; Count another free block
    55    04    A2   C0   0885  1600          ADDL2   4(R2),R5                ; Count this block in sum
 04 A2          20   D1   0889  1601          CMPL    #32,4(R2)               ; Is block 32 bytes or smaller?
          02    1F   088D  1602          BLSSU   15$                     ; Branch if larger than 32 bytes
          54    D6   088F  1603          INCL    R4                      ; Otherwise, count another "small" block
    56    04    A2   D1   0891  1604 15$:     CMPL    4(R2),R6                ; Is this block bigger than maximum?
          04    1B   0895  1605          BLEQU   20$                     ; Branch if not bigger
    56    04    A2   D0   0897  1606          MOVL    4(R2),R6                ; Otherwise, record new maximum
    57    04    A2   D1   089B  1607 20$:     CMPL    4(R2),R7                ; Is this block smaller than minimum?
          04    1E   089F  1608          BGEQU   30$                     ; Branch if not smaller
    57    04    A2   D0   08A1  1609          MOVL    4(R2),R7                ; Otherwise, record new minimum
       52    62    D0   08A5  1610 30$:     MOVL    (R2),R2                 ; Get next block
          D9    12   08A8  1611          BNEQ    10$                     ; Go back to top of loop if more
                05   08AA  1612          RSB                             ; Return to caller
                     08AB  1613
                     08AB  1614 ; This pool area is empty. Set minimum size to zero.
                     08AB  1615
          57    D4   08AB  1616 40$:     CLRL    R7                      ; Set minimum to zero
                05   08AD  1617          RSB                             ; Return to caller
                     08AE  1618
                     08AE  1619 END_LOCKED_CODE:                         ; End of code that executes above IPL 2
```

08AE 1620

```
                        08AE  1622          .SUBTITLE       DISPLAY_POOL Output Routine for Dynamic Memory Displays
                        08AE  1623
                        08AE  1624 ;+
                        08AE  1625 ; Functional Description:
                        08AE  1626 ;
                        08AE  1627 ;       This routine performs the common output and display functions for
                        08AE  1628 ;       the three variable sized dynamic memory areas. The routine decides
                        08AE  1629 ;       whether a normal or full display is requested. If a full display
                        08AE  1630 ;       is being produced, and thnonpaged dynaimc memory is the area being
                        08AE  1631 ;       displayed, two additional lines of output are produced.
                        08AE  1632 ;
                        08AE  1633 ; Calling Sequence:
                        08AE  1634 ;
                        08AE  1635 ;       BSBW    DISPLAY_POOL
                        08AE  1636 ;
                        08AE  1637 ; Input Parameters:
                        08AE  1638 ;
                        08AE  1639 ;       R0      Size in bytes of area being displayed
                        08AE  1640 ;
                        08AE  1641 ;       R2      Nonpaged pool indicator
                        08AE  1642 ;               R2<0> = 1 => nonpaged dynamic memory
                        08AE  1643 ;               R2<0> = 0 => Some other area than nonpaged pool
                        08AE  1644 ;
                        08AE  1645 ; Implicit Input:
                        08AE  1646 ;
                        08AE  1647 ;       Setting of MEMORY_V_FULL bit in MEMORY_L_BITLIS
                        08AE  1648 ;
                        08AE  1649 ;       Contents of cells in FAO parameter list for pool displays
                        08AE  1650 ;
                        08AE  1651 ; Output Parameters:
                        08AE  1652 ;
                        08AE  1653 ;       Several cells in FAO parameter list for pool displays
                        08AE  1654 ;
                        08AE  1655 ;               POOL_TOTAL
                        08AE  1656 ;               POOL_INUSE
                        08AE  1657 ;               POOL_TOTAL_PAGE (full display only)
                        08AE  1658 ;
                        08AE  1659 ; Implicit Output:
                        08AE  1660 ;
                        08AE  1661 ;       Displays of pool statistics for specified pool area are written
                        08AE  1662 ;       to SYS$OUTPUT.
                        08AE  1663 ;-
                        08AE  1664
                        08AE  1665 DISPLAY_POOL:
          00000078'EF   50  D0  08AE  1666          MOVL    R0,POOL_TOTAL                   ; Store pool size in FAO parameter list
00000084'EF 50  00000080'EF C3  08B5  1667          SUBL3   POOL_FREE,R0,POOL_INUSE ; INUSE = TOTAL - FREE
          50  000001FF 8F  C0  08C1  1668          ADDL2   #511,R0                         ; Round size to next page boundary
     0000007C'EF  50   F7 8F  78  08C8  1669          ASHL    #-9,R0,POOL_TOTAL_PAGES ; Convert to page count
        14 00000008'EF  04  E0  08D1  1670          BBS     #MEMORY_V_FULL,MEMORY_L_BITLIS,10$      ; Was /FULL specified?
                            08D9  1671          TYPEMSG SHOW$_MEM_POOL2,SHOW_POOL_LIST  ; No. Print normal display
                        05  08EC  1672          RSB                                     ;  and return to caller
                            08ED  1673
                            08ED  1674 ; A full display was requested in the SHOW MEMORY command
                            08ED  1675
                            08ED  1676 10$:     TYPEMSG SHOW$_MEM_POOL_FULL1,SHOW_POOL_LIST
                            0900  1677          TYPEMSG SHOW$_MEM_POOL_FULL2,SHOW_POOL_LIST2
                            0913  1678
```

```
                                0913  1679 ; Skip next two displays unless nonpaged pool
                                0913  1680
                          66 52 E9  0913  1681          BLBC    R2,20$
    00000078'EF   00000000'GF   D0  0916  1682          MOVL    G^SGN$GL_NPAGEDYN,POOL_TOTAL    ; Get initial pool size
50  00000078'EF   000001FF  8F   C1  0921  1683          ADDL3   #511,POOL_TOTAL,R0      ; Round up to next page boundary
     0000007C'EF    50    F7  8F   78  092D  1684          ASHL    #-9,R0,POOL_TOTAL_PAGES ; Convert to pages
                                0936  1685          TYPEMSG SHOW$_MEM_POOL_FULL3,SHOW_POOL_LIST3
    00000078'EF   00000000'GF   D0  0949  1686          MOVL    G^SGN$GL_NPAGEVIR,POOL_TOTAL    ; Get maximum pool size
50  00000078'EF   000001FF  8F   C1  0954  1687          ADDL3   #511,POOL_TOTAL,R0      ; Round up to next page boundary
     0000007C'EF    50    F7  8F   78  0960  1688          ASHL    #-9,R0,POOL_TOTAL_PAGES ; Convert to pages
                                0969  1689          TYPEMSG SHOW$_MEM_POOL_FULL4,SHOW_POOL_LIST4
                                097C  1690
                                097C  1691 20$:       TYPEMSG SHOW$_MEM_POOL_FULL5,SHOW_POOL_LIST5    ; Display usage data
                                098F  1692          TYPEMSG SHOW$_MEM_POOL_FULL6,SHOW_POOL_LIST6    ; Display upper bound
                                09A2  1693          TYPEMSG SHOW$_MEM_POOL_FULL7,SHOW_POOL_LIST7    ; Display lower bound
                             05 09B5  1694          RSB                             ; Return to caller
                                09B6  1695
```

```
                                09B6  1697          .SUBTITLE       PAGEFILE - Display Paging File Statistics
                                09B6  1698 ;+
                                09B6  1699 ; Functional Description:
                                09B6  1700 ;
                                09B6  1701 ;       This routine gathers information about each paging and swap file.
                                09B6  1702 ;       In particular, the size of each file and the amount of free space
                                09B6  1703 ;       is displayed. In the display selected when the /FULL qualifier is
                                09B6  1704 ;       specified, the number of processes paging and swapping to each
                                09B6  1705 ;       file is added to the list of information.
                                09B6  1706 ;
                                09B6  1707 ; Input Parameters:
                                09B6  1708 ;
                                09B6  1709 ;       None
                                09B6  1710 ;
                                09B6  1711 ; Implicit Input:
                                09B6  1712 ;
                                09B6  1713 ;       SGN$GW_SWPFILCT Maximum number of swap files that can be installed
                                09B6  1714 ;
                                09B6  1715 ;       SGN$GW_PAGFILCT Maximum number of paging files that can be installed
                                09B6  1716 ;
                                09B6  1717 ;       Setting of MEMORY_V_FULL bit in MEMORY_L_BITLIS controls the
                                09B6  1718 ;       amount of information displayed for each file.
                                09B6  1719 ;
                                09B6  1720 ; Output Parameters:
                                09B6  1721 ;
                                09B6  1722 ;       None
                                09B6  1723 ;
                                09B6  1724 ; Implicit Output:
                                09B6  1725 ;
                                09B6  1726 ;       Paging file usage information is displayed on SYS$OUTPUT
                                09B6  1727 ;-
                                09B6  1728
                                09B6  1729 PAGEFILE:
                         00FC   09B6  1730          .WORD   ^M<R2,R3,R4,R5,R6,R7>   ; Save some registers
    0000027C'EF  00000000'GF     3C   09B8  1731          MOVZWL  G^SGN$GW_SWPFILCT,SWAP_FILE_COUNT
    00000280'EF  00000000'GF     3C   09C3  1732          MOVZWL  G^SGN$GW_PAGFILCT,PAGE_FILE_COUNT
52  0000027C'EF  00000280'EF     C1   09CE  1733          ADDL3   PAGE_FILE_COUNT,SWAP_FILE_COUNT,R2
      04  00000044 8F    52     7A   09DA  1734          EMUL    R2,#PFL_K_EXT_LENGTH,#4,PFL_TABLE_SIZE
          00000274'EF           09E2
          00000278'EF     DF   09E7  1735          PUSHAL  PFL_TABLE_ADDR          ; Set up argument list for LIB$GET_VM
          00000274'EF     DF   09ED  1736          PUSHAL  PFL_TABLE_SIZE          ; Point to requested block size
       00000000'GF    02     FB   09F3  1737          CALLS   #2,G^LIB$GET_VM         ; Allocate a scratch area
                 01 50     E8   09FA  1738          BLBS    R0,5$                   ; Abandon display if no space available
                           04   09FD  1739 2$:       RET
                                09FE  1740
                                09FE  1741 5$:       $CMKRNL_S       GET_PFL_DATA    ; Gather data from nonpaged pool
                    ED 50      E9   0A0D  1742          BLBC    R0,2$                   ; Skip rest if error occurred
           12 00000008'EF    04   E0   0A10  1743          BBS     #MEMORY_V_FULL,MEMORY_L_BITLIS,10$      ; Was /FULL specified?
                                0A18  1744          TYPEMSG SHOW$_MEM_PAGE1         ; Print header line for normal display
                         005F   31   0A27  1745          BRW     40$                     ; Go to page file loop
                                0A2A  1746
                                0A2A  1747 ; Allocate two arrays of words for each paging and swap file, so that
                                0A2A  1748 ; we can keep a count of how many processes are paging and swapping
                                0A2A  1749 ; into each file. Word arrays can be used because of the VMS architectural
                                0A2A  1750 ; limit of 32767 processes.
                                0A2A  1751
                                0A2A  1752                                          ; R2 = PAGFILCNT + SWPFILCNT
```

```
                                                D 6
                51    52    02    78   0A2A  1753 10$:     ASHL    #2,R2,R1                ; R1 = size of table in bytes
                      5E    51    C2   0A2E  1754           SUBL    R1,SP                   ; Allocate the array on the stack
          00000284'EF     5E    D0   0A31  1755           MOVL    SP,SWAP_FILE_TABLE      ; Store address of swap file table
       00000288'EF   6E42    3E   0A38  1756           MOVAW   (SP)[R2],PAGE_FILE_TABLE ; Store address of paging file table
    6E    51    00    6E    00    2C   0A40  1757           MOVC5   #0,(SP),#0,R1,(SP)      ; Zero the tables
                                    0A46  1758
                                    0A46  1759 ; Now use the wild card mode of $GETJPI to count the number of processes
                                    0A46  1760 ; paging and swapping into each paging and swap file.
                                    0A46  1761
                                    0A46  1762 20$:     $GETJPI_G       GETJPI_LIST     ; Call $GETJPI
                      2E    50    E9   0A51  1763           BLBC    R0,30$                  ; Skip next if error occurred
                                    0A54  1764           $WAITFR_S       EFN=#EVENT_FLAG ; Wait for $GETJPI to complete
          1E 00000294'EF    E9   0A5D  1765           BLBC    GETJPI_STATUS,30$       ; Skip next if error occurred
       50    0000028F'EF    9A   0A64  1766           MOVZBL  PAGE_FILE_INDEX,R0      ; Get page file index for process
          00000288'FF40    B6   0A6B  1767           INCW    @PAGE_FILE_TABLE[R0]    ; Bump appropriate counter
       50    00000293'EF    9A   0A72  1768           MOVZBL  SWAP_FILE_INDEX,R0      ; Get swap file index for process
          00000284'FF40    B6   0A79  1769           INCW    @SWAP_FILE_TABLE[R0]    ; Bump appropriate counter
                      C4    11   0A80  1770           BRB     20$                     ; Back to top of loop
                                    0A82  1771
                09A8 8F    50    B1   0A82  1772 30$:     CMPW    R0,#SS$_NOMOREPROC      ; This error code is loop breaker
                      BD    12   0A87  1773           BNEQ    20$                     ; Go back for more if different error
                                    0A89  1774
                                    0A89  1775 ; Now scan page and swap file array and display information about each file
                                    0A89  1776
          57    00000278'EF    D0   0A89  1777 40$:     MOVL    PFL_TABLE_ADDR,R7               ; R7 will step through scratch area
             50    01    67    C1   0A90  1778 50$:     ADDL3   (R7),#1,R0              ; Is first longword -1?
                         03    12   0A94  1779           BNEQ    55$                     ; Continue if not -1
                      011F    31   0A96  1780           BRW     90$                     ; Equal to -1 implies end of loop
                                    0A99  1781
                      01D9    30   0A99  1782 55$:     BSBW    GET_FILE_NAME           ; Translate FID to file name
                000001FF'EF    B5   0A9C  1783           TSTW    FILE_NAME_DESC          ;
                         1C    13   0AA2  1784           BEQL    56$                     ; Error returns null string
             52    00000203'EF    D0   0AA4  1785           MOVL    FILE_NAME_DESC+4,R2     ;
                   62    5F 8F    91   0AAB  1786           CMPB    #^A/_/,(R2)             ; If    name returned contains
                         0F    12   0AAF  1787           BNEQ    56$                     ;         a leading underscore
                000001FF'EF    B7   0AB1  1788           DECW    FILE_NAME_DESC          ; Then  strip it out
 62    01 A2    000001FF'EF    28   0AB7  1789           MOVC3   FILE_NAME_DESC,1(R2),(R2)
                                    0AC0  1790 56$:                                     ;
 000000E4'EF     08    14 A7    C5   0AC0  1791           MULL3   PFL$L_BITMAPSIZ(R7),#8,PAGE_TOTAL
                                    0AC9  1792                                           ; Get total number of pages
       000000DC'EF    18 A7    D0   0AC9  1793           MOVL    PFL$L_FREPAGCNT(R7),PAGE_FREE
                                    0AD1  1794                                           ; Get number of free pages
 000000E4'EF    000000DC'EF    C3   0AD1  1795           SUBL3   PAGE_FREE,PAGE_TOTAL,PAGE_USED
                000000E0'EF         0ADC
                                    0AE1  1796                                           ; Get number of pages in use
       47 00000008'EF    04    E0   0AE1  1797           BBS     #MEMORY_V_FULL,MEMORY_L_BITLIS,70$      ; Was /FULL specified?
                                    0AE9  1798
                                    0AE9  1799 ; Either of these next two TYPEMSG calls is used for a normal display
                                    0AE9  1800 ; of a paging or swap file. If the file name and the usage data can fit
                                    0AE9  1801 ; on a single line, a one-line display is used. Otherwise, the file name
                                    0AE9  1802 ; is displayed on one line and the usage data is displayed on the next.
                                    0AE9  1803
          28    000001FF'EF    B1   0AE9  1804           CMPW    FILE_NAME_DESC,#SHOW$C_MEM_SHORT_NAME
                                    0AF0  1805                                           ; Will file name fit on one line?
                         16    1A   0AF0  1806           BGTRU   60$                     ; Branch if name does not fit
                                    0AF2  1807           TYPEMSG SHOW$_MEM_PAGE2,SHOW_PAGE_LIST  ; Print single line display
                      00A6    31   0B05  1808           BRW     80$                     ; Go to common end of loop
```

```
                                E 6
                                0B08  1809
                                0B08  1810 60$:     TYPEMSG SHOW$_MEM_PAGE3,SHOW_PAGE_LIST  ; Print first of two lines
                                0B1B  1811          TYPEMSG SHOW$_MEM_PAGE4,SHOW_PAGE_LIST  ; Print second of two lines
                          7E    11  0B2E  1812          BRB     80$                             ; Go to common end of loop
                                0B30  1813
                                0B30  1814 ; The next several TYPEMSG calls are all used for a full display of
                                0B30  1815 ; each paging and swap file.
                                0B30  1816
                     56   000000E8'EF    D0  0B30  1817 70$:     MOVL    PAGE_PFL_INDEX,R6       ; Retrieve PFL index
000000F0'EF       00000288'FF46    3C  0B37  1818          MOVZWL  @PAGE_FILE_TABLE[R6],PAGE_FULL_PAGING_COUNT
000000EC'EF       00000284'FF46    3C  0B43  1819          MOVZWL  @SWAP_FILE_TABLE[R6],PAGE_FULL_SWAP_COUNT
                                0B4F  1820          TYPEMSG SHOW$_MEM_PAGE_FULL1,SHOW_PAGE_LIST     ; Print file name
                                0B62  1821          TYPEMSG SHOW$_MEM_PAGE_FULL2,SHOW_PAGE_LIST2    ; Print file size
                                0B75  1822          TYPEMSG SHOW$_MEM_PAGE_FULL3,SHOW_PAGE_LIST3    ; Print free space
                                0B88  1823          TYPEMSG SHOW$_MEM_PAGE_FULL4,SHOW_PAGE_LIST4    ; Print file usage
                                0B9B  1824          TYPEMSG SHOW$_MEM_PAGE_FULL5,SHOW_PAGE_LIST5    ; Display type of file
                                0BAE  1825
                57   00000044 8F    C0  0BAE  1826 80$:     ADDL2   #PFL_K_EXT_LENGTH,R7    ; Advance R7 to next slot in scratch area
                           FED8    31  0BB5  1827          BRW     50$                             ;  and go back to top of loop
                                0BB8  1828
                           50   01  3C  0BB8  1829 90$:     MOVZWL  #SS$_NORMAL,R0          ; Signal success
                                04  0BBB  1830          RET                             ;  and return
                                0BBC  1831
```

```
0BBC  1833          .SUBTITLE       GET_PFL_DATA    Gather page file control block data
0BBC  1834 ;+
0BBC  1835 ; Functional Description:
0BBC  1836 ;
0BBC  1837 ;       This routine executes in kernel mode and copies all active PFL control
0BBC  1838 ;       blocks and their associated file name information to a scratch buffer
0BBC  1839 ;       in P1 space.
0BBC  1840 ;
0BBC  1841 ;  Calling sequence:                        >>>> KERNEL MODE REQUIRED <<<<
0BBC  1842 ;
0BBC  1843 ;       CALLS   #0,GET_PFL_DATA
0BBC  1844 ;
0BBC  1845 ;  Input parameters:
0BBC  1846 ;
0BBC  1847 ;       MMG$GL_PAGSWPVC Pointer to array of PFL pointers
0BBC  1848 ;
0BBC  1849 ;       PFL_TABLE_ADDR  Address of scratch area into which all PFLs
0BBC  1850 ;                       currently in use will be copied.
0BBC  1851 ;
0BBC  1852 ; Implicit input:
0BBC  1853 ;
0BBC  1854 ;       Data bases for I/O system and file system
0BBC  1855 ;
0BBC  1856 ; Output parameters:
0BBC  1857 ;
0BBC  1858 ;       None
0BBC  1859 ;
0BBC  1860 ; Implicit Output:
0BBC  1861 ;
0BBC  1862 ;       The contents of each PFL are copied from nonpaged pool to a scratch
0BBC  1863 ;       area. In addition, for each file the file ID is copied and the
0BBC  1864 ;       device name string is produced.
0BBC  1865 ;
0BBC  1866 ;       The default paging and swap files do not have FCBs or FIDs
0BBC  1867 ;       associated with their WCBs. This information is communicated to
0BBC  1868 ;       user mode by storing a -1 in the PFL index field and placing the
0BBC  1869 ;       actual PFL index in PFL_W_FID_NUM.
0BBC  1870 ;
0BBC  1871 ;       The two cases that can occur are as follows.
0BBC  1872 ;
0BBC  1873 ;       1.      PFL index is not negative
0BBC  1874 ;
0BBC  1875 ;               This is the case for all paging and swap files except those
0BBC  1876 ;               installed by SYSINIT at boot time.
0BBC  1877 ;
0BBC  1878 ;       2.      PFL index is negative but FID_NUM is positive
0BBC  1879 ;
0BBC  1880 ;               This is a primary paging or swap file installed by SYSINIT
0BBC  1881 ;               before the file system was operating. The WCB does not point
0BBC  1882 ;               to a FCB and so the FID is not available. The contents of
0BBC  1883 ;               FID_NUM are the PFL index.
0BBC  1884 ;
0BBC  1885 ;       The end of list is indicated by placing a -1 in the first longword
0BBC  1886 ;       after the last entry. This field contains the BITMAP address in a
0BBC  1887 ;       valid PFL so there is no ambiguity.
0BBC  1888 ;
0BBC  1889 ; While the loop executes, the following register conventions are observed.
```

```
                                0BBC  1890 ;
                                0BBC  1891 ;       R6      Index into PFL vector
                                0BBC  1892 ;       R7      Pointer to "real" PFL in nonpaged pool
                                0BBC  1893 ;       R8      Pointer to WCB for this page or swap file
                                0BBC  1894 ;       R10     Pointer to extended PFL in scratch area
                                0BBC  1895 ;       R11     Pointer to PFL vector (of PFL pointers) in nonpaged pool
                                0BBC  1896 ;-
                                0BBC  1897
                                0BBC  1898 GET_PFL_DATA:
                        0FFC    0BBC  1899         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
5B   00000000'GF      DO        0BBE  1900         MOVL    G^MMG$GL_PAGSWPVC,R11   ; R11 points to top of PFL array
5A   00000278'EF      DO        0BC5  1901         MOVL    PFL_TABLE_ADDR,R10      ; R10 points to start of scratch area
            56        D4        0BCC  1902         CLRL    R6                      ; R6 is the PFL index
        57   6B46      DO        0BCE  1903 10$:    MOVL    (R11)[R6],R7            ;  and R7 points to the "real" PFL
    46 23 A7   00      E1        0BD2  1904         BBC     #PFL$V_INITED,PFL$B_FLAGS(R7),40$ ; Skip entire loop if not installe
     6A   67    24      28        0BD7  1905         MOVC3   #PFL$K_LENGTH,(R7),(R10)        ; Copy PFL to scratch area
       58   0C A7      DO        0BDB  1906         MOVL    PFL$L_WINDOW(R7),R8     ; WCB address to R8
            10 A8      DD        0BDF  1907         PUSHL   WCB$L_ORGUCB(R8)        ; Address of UCB for paging device
                               0BE2  1908         ASSUME  PFL_S_DEVNAM LE 256     ; ASCIC size must fit in a byte
          7E    18     9A        0BE2  1909         MOVZBL  #PFL_S_DEVNAM,-(SP)     ; Size of device name string buffer
          2C AA      9F        0BE5  1910         PUSHAB  PFL_T_DEVNAM(R10)       ; Address of device name string buffer
00000C2C'EF     03      FB        0BE8  1911         CALLS   #3,GET_DEV_NAME         ;
            01 50      E8        0BEF  1912         BLBS    R0,15$                  ; If    ERROR on getting device name
                       04        0BF2  1913         RET                             ; Then  Return error status to caller
                                0BF3  1914 15$:                                    ; Else  Continue
       55   18 A8      DO        0BF3  1915         MOVL    WCB$L_FCB(R8),R5        ; Now get FCB address
                15      13        0BF7  1916         BEQL    20$                     ; No FCB for default page or swap file
                                0BF9  1917
                                0BF9  1918 ; Copy three words of File ID from FCB to scratch area for this PFL
                                0BF9  1919
    26 AA    24 A5    B0        0BF9  1920         MOVW    FCB$W_FID_NUM(R5),PFL_W_FID_NUM(R10)
    28 AA    26 A5    B0        0BFE  1921         MOVW    FCB$W_FID_SEQ(R5),PFL_W_FID_SEQ(R10)
    2A AA    28 A5    B0        0C03  1922         MOVW    FCB$W_FID_RVN(R5),PFL_W_FID_RVN(R10)
       24 AA    56    B0        0C08  1923         MOVW    R6,PFL_W_PFL_INDEX(R10) ; Store PFL index
                08    11        0C0C  1924         BRB     30$                     ; Transfer to common end of loop
                                0C0E  1925
                                0C0E  1926 ; The default paging or swap file has a -1 placed in the PFL index field
                                0C0E  1927 ; and the PFL index is stored in the first word of the file ID.
                                0C0E  1928
       24 AA    00    B2        0C0E  1929 20$:    MCOMW   #0,PFL_W_PFL_INDEX(R10) ; Signal default paging or swap file
       26 AA    56    B0        0C12  1930         MOVW    R6,PFL_W_FID_NUM(R10)   ;  but make PFL index available
                                0C16  1931
 5A    00000044 8F    C0        0C16  1932 30$:    ADDL2   #PFL_K_EXT_LENGTH,R10   ; Advance to scratch area for next PFL
A9 56  00000000'GF    F3        0C1D  1933 40$:    AOBLEQ  G^MMG$GL_MAXPFIDX,R6,10$; Bump PFL index & check limit
                                0C25  1934                                        ; Quit when all PFL entries processed
          6A   00     D2        0C25  1935         MCOML   #0,(R10)                ; Indicate end of active PFLs
          50   01     3C        0C28  1936         MOVZWL  #SS$_NORMAL,R0          ; Signal success
                      04        0C2B  1937         RET                             ;  and return
                                0C2C  1938
```

```
                        0C2C  1940          .SUBTITLE       GET_DEV_NAME - Extract device name from UCB
                        0C2C  1941 ;+
                        0C2C  1942 ;
                        0C2C  1943 ; Functional description:
                        0C2C  1944 ;
                        0C2C  1945 ;       This routine invokes IOC$CVT_DEVNAM and returns a counted ASCII
                        0C2C  1946 ;       string for the device name string derived from a given a UCB.
                        0C2C  1947 ;       It handles the protocol for obtaining the I/O Database resource
                        0C2C  1948 ;       lock needed to do this and releases it before returning.
                        0C2C  1949 ;
                        0C2C  1950 ; Calling sequesnce:             >>>> KERNEL MODE REQUIRED <<<<
                        0C2C  1951 ;
                        0C2C  1952 ;       CALL GET_DEV_NAME ( UCB, BUFSIZ, BUFFER )
                        0C2C  1953 ;
                        0C2C  1954 ; Input Parameters:
                        0C2C  1955 ;
                        0C2C  1956 ;       UCB     REFERENCE address of device unit control block (UCB)
                        0C2C  1957 ;       BUFSIZ  VALUE for size of device name buffer
                        0C2C  1958 ;
                        0C2C  1959 ; Output Parameters:
                        0C2C  1960 ;
                        0C2C  1961 ;       BUFFER  REFERENCE address of buffer for the ASCIC device name string
                        0C2C  1962 ;-
                        0C2C  1963 ;
                        0C2C  1964 ; Define offsets from routine's argument pointer:
00000004                0C2C  1965          BUFFER  =  4
00000008                0C2C  1966          BUFSIZ  =  8
0000000C                0C2C  1967          UCB     = 12
                        0C2C  1968
                        0C2C  1969 GET_DEV_NAME:
                   003C 0C2C  1970          .WORD   ^M<R2,R3,R4,R5>
                        0C2E  1971          SAVIPL                          ; Save current IPL for later restore
54 00000000'GF     D0   0C31  1972          MOVL    G^SCH$GL_CURPCB,R4      ; Get address of current process's PCB
             54    DD   0C38  1973          PUSHL   R4                      ; Save argument for UNLOCK later
00000000'GF        16   0C3A  1974          JSB     G^SCH$IOLOCKR           ; Lock the I/O Data Base
                        0C40  1975                                          ;;; Returns at ASTDEL
   50   08 AC     9A   0C40  1976          MOVZBL  BUFSIZ(AP),R0           ;;; Size of device name string buffer
             50    D7   0C44  1977          DECL    R0                      ;;; Less one byte for count field
   51   04 AC     D0   0C46  1978          MOVL    BUFFER(AP),R1           ;;; Address of device name string buffer
             51    D6   0C4A  1979          INCL    R1                      ;;; Leave byte for count field
      54    01     CE   0C4C  1980          MNEGL   #1,R4                   ;;; Include node name only if in cluster
   55   0C AC     D0   0C4F  1981          MOVL    UCB(AP),R5              ;;; Address of UCB for paging device
00000000'GF        16   0C53  1982          JSB     G^IOC$CVT_DEVNAM        ;;; Produce device name string from UCB
             54 8ED0   0C59  1983          POPL    R4                      ;;; Recover current process PCB
      7E     50    7D   0C5C  1984          MOVQ    R0,-(SP)                ;;; Save status & length of dev name str
00000000'GF        16   0C5F  1985          JSB     G^SCH$IOUNLOCK          ;;; Unlock I/O Data Base
      50     8E    7D   0C65  1986          MOVQ    (SP)+,R0                ;;; Restore status & length of dev name str
                        0C68  1987          ENBINT                          ;;; Restore previous IPL
      02 50     E8   0C6B  1988          BLBS    R0,15$                  ; If    ERROR on getting device name
             51    D4   0C6E  1989          CLRL    R1                      ; Then  Return zero length to caller
   04 BC     51    90   0C70  1990 15$:     MOVB    R1,@BUFFER(AP)          ; Store length for ASCIC dev name str
                   04   0C74  1991          RET                             ;
                        0C75  1992
```

```
                                0C75  1994          .SUBTITLE       GET_FILE_NAME - Translate File ID to File Name
                                0C75  1995 ;+
                                0C75  1996 ;
                                0C75  1997 ; This routine translates a device string, a unit number, and a file ID
                                0C75  1998 ; of a paging or swap file into a name for that file. If the file in
                                0C75  1999 ; question is the primary paging or swap file (file ID is not available)
                                0C75  2000 ; then a default file name is constructed.
                                0C75  2001 ;
                                0C75  2002 ; Input Parameters:
                                0C75  2003 ;
                                0C75  2004 ;       R7      Address of extended PFL in scratch area
                                0C75  2005 ;
                                0C75  2006 ; Output Parameters:
                                0C75  2007 ;
                                0C75  2008 ;       FILE_NAME_DESC  contains a string descriptor for the file name
                                0C75  2009 ;-
                                0C75  2010
                                0C75  2011 GET_FILE_NAME:
            52    2C A7   9A    0C75  2012          MOVZBL  PFL_T_DEVNAM(R7),R2             ; Character count to R2
 000000F8'EF      52   D0    0C79  2013          MOVL    R2,DEVICE_NAME_DESC             ; Store in descriptor
 000000FC'EF    2D A7   9E    0C80  2014          MOVAB   PFL_T_DEVNAM+1(R7),DEVICE_NAME_DESC+4   ; Store string address
                                0C88  2015
                                0C88  2016 ; Set file name size in descriptor that points to file name buffer
                                0C88  2017
                                0C88  2018          ASSUME  FILE_NAME_SIZE LT 256           ;
 000001FF'EF    FF  8F   9A    0C88  2019          MOVZBL  #FILE_NAME_SIZE,FILE_NAME_DESC  ; Store buffer size
 000000E8'EF    24 A7    32    0C90  2020          CVTWL   PFL_W_PFL_INDEX(R7),PAGE_PFL_INDEX       ; PFL index to FAO list
                   28    19    0C98  2021          BLSS    10$                             ; Negative index implies default file
 000002A8'EF    26 A7    3E    0C9A  2022          MOVAW   PFL_W_FID(R7),FID_TO_NAME_FID_ADDR      ; Store address of FID
00000000'GF  000002A0'EF   FA    0CA2  2023          CALLG   FID_TO_NAME_ARG_LIST,G^LIB$FID_TO_NAME  ; Convert FID to file name
                06  50   E8    0CAD  2024          BLBS    R0,5$                           ; Check for error
             0000024F'EF   D4    0CB0  2025          CLRL    RETURN_LENGTH                   ; Display nothing if error
000001FF'EF  0000024F'EF   B0    0CB6  2026 5$:      MOVW    RETURN_LENGTH,FILE_NAME_DESC    ; Store actual name length
                   05    05    0CC1  2027          RSB                                     ;  and return to caller
                                0CC2  2028
                                0CC2  2029 ; The file names for the paging and swap files installed by SYSINIT are
                                0CC2  2030 ; fabricated dynamically from the device name and unit number.
                                0CC2  2031 ;
                                0CC2  2032 ; 1.    $GETDVI translates the device name to its logical equivalent.
                                0CC2  2033 ;       If this logical name has been deleted, the device name returned
                                0CC2  2034 ;       by $GETDVI is used in its place.
                                0CC2  2035 ;
                                0CC2  2036 ; 2.    Logical name SYS$TOPSYS is translated to form the first part of
                                0CC2  2037 ;       the directory string.
                                0CC2  2038 ;
                                0CC2  2039 ; 3.    The string "SYSEXE]" is added by hand.
                                0CC2  2040 ;
                                0CC2  2041 ; 4.    The string "PAGE" or "SWAP" is added, depending on whether this
                                0CC2  2042 ;       is the primary paging or swap file.
                                0CC2  2043 ;
                                0CC2  2044 ; 5.    The string "FILE.SYS" is placed at the end.
                                0CC2  2045
                                0CC2  2046 10$:     $GETDVI_G       GETDVI_LIST     ; Get proper device name
            56  50   E9        0CCD  2047          BLBC    R0,17$                  ; Quit if error occurred
 000001FF'EF     B5             0CD0  2048          TSTW    FILE_NAME_DESC          ; Did we get a LOGVOLNAM?
                   1B   12      0CD6  2049          BNEQ    15$                     ; Nonzero implies that we did. Use it.
00000207'EF  0000024F'EF   28    0CD8  2050          MOVC3   RETURN_LENGTH,DEVICE_NAME_ADDR,@FILE_NAME_DESC+4
```

```
                  00000203'FF          0CE3
     000001FF'EF  0000024F'EF    D0    0CE8  2051              MOVL    RETURN_LENGTH,FILE_NAME_DESC    ; Otherwise, use the DEVNAM
53   000001FF'EF  00000203'EF    C1    0CF3  2052  15$:         ADDL3   FILE_NAME_DESC+4,FILE_NAME_DESC,R3 ; R3 will step through string
                83   5B3A 8F    B0    0CFF  2053              MOVW    #^A\:[\,(R3)+
                                      0D04  2054
                                      0D04  2055  ; Use the scratch descriptor as the output descriptor to $TRNLOG. The size of
                                      0D04  2056  ; the area is the device name size (RETURN_LENGTH) plus two (for the ":[").
                                      0D04  2057
     000000FD 8F  0000024F'EF    C3    0D04  2058              SUBL3   RETURN_LENGTH,#<FILE_NAME_SIZE-2>,SCRATCH_DESC
                  00000247'EF          0D0F
            0000024B'EF   53    D0    0D14  2059              MOVL    R3,SCRATCH_DESC+4       ; Store address
                                      0D1B  2060              $TRNLOG_G       TRNLOG_LIST     ; Translate SYS$TOPSYS
                    65 50     E9    0D26  2061  17$:         BLBC    R0,50$                  ; Quit in case an error occurred
                0629 8F   50    B1    0D29  2062              CMPW    R0,#SS$_NOTRAN          ; Do not update R3 if no translation
                      0A      13    0D2E  2063              BEQL    20$                     ; Go get rest of directory string
            53   0000024F'EF  C0    0D30  2064              ADDL2   RETURN_LENGTH,R3        ; Place R3 beyond translated string
                      83    2E    90    0D37  2065              MOVB    #^A\.\,(R3)+            ; Add "." separator
63   00000257'FF  00000253'EF    28    0D3A  2066  20$:         MOVC3   DEFAULT_DIRECTORY_NAME,@DEFAULT_DIRECTORY_NAME+4,(R3)
        000000E8'EF    26 A7    3C    0D46  2067              MOVZWL  PFL_W_FID_NUM(R7),PAGE_PFL_INDEX        ; Store PFL index
     00000000'GF  000000E8'EF    B1    0D4E  2068              CMPW    PAGE_PFL_INDEX,G^MMG$GW_MINPFIDX        ; Is this the primary
                           09     13    0D59  2069              BEQL    30$                     ;  paging file? Branch if it is.
            83   50415753 8F    D0    0D5B  2070              MOVL    #^A\SWAP\,(R3)+         ; Otherwise, call it SWAPFILE.SYS
                           12     11    0D62  2071              BRB     40$                     ;  and join the common exit code
            83   45474150 8F    D0    0D64  2072  30$:         MOVL    #^A\PAGE\,(R3)+         ; Make the name PAGEFILE.SYS
     000000F4'EF  00000A89'EF    3E    0D6B  2073              MOVAW   PAGE_INDIC_DESC,PAGE_FLAG       ; Indicate that paging is allowed
63   00000266'FF  00000262'EF    28    0D76  2074  40$:         MOVC3   DEFAULT_FILE_NAME,@DEFAULT_FILE_NAME+4,(R3)     ; Fill in rest of na
000001FF'EF   53  00000203'EF    C3    0D82  2075              SUBL3   FILE_NAME_DESC+4,R3,FILE_NAME_DESC      ; Store actual file name
                                      05    0D8E  2076  50$:         RSB                             ;  and return
                                      0D8F  2077
                                      0D8F  2078              .END
```

```
$$ARGS                          =  00000006
$$T1                            =  0000001C
BEGIN_LOCKED_CODE                  0000059B  R      05
BIT...                          =  00000006
BUFFER                          =  00000004
BUFSIZ                          =  00000008
BYTES_SIZE_DESC                    0000006F  R      04
CLI$PRESENT                        ********    X   05
CONVERT_PACKET_COUNT               000006C9  R      05
CTL$GQ_ALLOCREG                    ********    X   05
DDB$S_NAME                      =  00000010
DEFAULT_DIRECTORY_NAME             00000253  R      03
DEFAULT_FILE_NAME                  00000262  R      03
DEVICE_NAME_ADDR                   00000207  R      03
DEVICE_NAME_DESC                   000000F8  R      03
DEVICE_NAME_SIZE                =  00000040
DISPLAY_LOOK                       000005CE  R      05
DISPLAY_POOL                       000008AE  R      05
DVI$_DEVNAM                     =  00000020
DVI$_LOGVOLNAM                  =  0000002C
DVI_ITEM_LIST                      00000090  R      02
END_LOCKED_CODE                    000008AE  R      05
EVENT_FLAG                      =  00000001
EXE$C_ALCGRNMSK                    ********    X   05
EXE$GL_CONFREGL                    ********    X   05
EXE$GL_NONPAGED                    ********    X   05
EXE$GL_PAGED                       ********    X   05
EXE$GL_PGDYNMTX                    ********    X   05
EXE$GL_RPB                         ********    X   05
FAO$_CTRSTR                     =  00000004
FAO$_NARGS                      =  00000014
FAO$_OUTBUF                     =  0000000C
FAO$_OUTLEN                     =  00000008
FAO$_P1                         =  00000010
FAO$_P10                        =  00000034
FAO$_P11                        =  00000038
FAO$_P12                        =  0000003C
FAO$_P13                        =  00000040
FAO$_P14                        =  00000044
FAO$_P15                        =  00000048
FAO$_P16                        =  0000004C
FAO$_P17                        =  00000050
FAO$_P2                         =  00000014
FAO$_P3                         =  00000018
FAO$_P4                         =  0000001C
FAO$_P5                         =  00000020
FAO$_P6                         =  00000024
FAO$_P7                         =  00000028
FAO$_P8                         =  0000002C
FAO$_P9                         =  00000030
FAO_CONTROL_STRING                 000000D0  R      02
FAO_LIST                           000002B4  R      03
FCB$W_FID_NUM                   =  00000024
FCB$W_FID_RVN                   =  00000028
FCB$W_FID_SEQ                   =  00000026
FID_TO_NAME_ARG_LIST               000002A0  R      03
FID_TO_NAME_FID_ADDR               000002A8  R      03
FILE_NAME_ADDR                     00000100  R      03
FILE_NAME_DESC                     000001FF  R      03
FILE_NAME_SIZE                  =  000000FF
GETDVI$_ASTADR                  =  00000018
GETDVI$_ASTPRM                  =  0000001C
GETDVI$_CHAN                    =  00000008
GETDVI$_DEVNAM                  =  0000000C
GETDVI$_EFN                     =  00000004
GETDVI$_IOSB                    =  00000014
GETDVI$_ITMLST                  =  00000010
GETDVI$_NARGS                   =  00000008
GETDVI$_NULLARG                 =  00000020
GETDVI_LIST                        000000AC  R      02
GETJPI$_ASTADR                  =  00000018
GETJPI$_ASTPRM                  =  0000001C
GETJPI$_EFN                     =  00000004
GETJPI$_IOSB                    =  00000014
GETJPI$_ITMLST                  =  00000010
GETJPI$_NARGS                   =  00000007
GETJPI$_PIDADR                  =  00000008
GETJPI$_PRCNAM                  =  0000000C
GETJPI_LIST                        00000070  R      02
GETJPI_STATUS                      00000294  R      03
GET_DEV_NAME                       00000C2C  R      05
GET_FILE_NAME                      00000C75  R      05
GET_PFL_DATA                       00000BBC  R      05
HEADER_LIST                        0000000C  R      03
IOC$CVT_DEVNAM                     ********    X   05
IOC$GL_IRPCNT                      ********    X   05
IOC$GL_IRPFL                       ********    X   05
IOC$GL_IRPMIN                      ********    X   05
IOC$GL_LRPCNT                      ********    X   05
IOC$GL_LRPFL                       ********    X   05
IOC$GL_LRPMIN                      ********    X   05
IOC$GL_LRPSIZE                     ********    X   05
IOC$GL_SRPCNT                      ********    X   05
IOC$GL_SRPFL                       ********    X   05
IOC$GL_SRPMIN                      ********    X   05
IOC$GL_SRPSIZE                     ********    X   05
IPL$_ASTDEL                     =  00000002
IRP$R_LENGTH                    =  000000C4
IRPLIST_DESC                       000000CA  R      04
IRP_NAME_DESC                      000000BF  R      04
IRP_SIZE_DESC                      000000EA  R      04
JPI$_PAGFILLOC                  =  00000419
JPI$_SWPFILLOC                  =  00000321
JPI_ITEM_LIST                      00000054  R      02
LIB$FID_TO_NAME                    ********    X   05
LIB$GET_VM                         ********    X   05
LOCAL_MEMORY                       00000054  R      03
LOCKED_CODE_RANGE                  00000000  R      03
LOOKASIDE                          00000428  R      05
LOOK_BLOCK_MIN                     000000C0  R      03
LOOK_BLOCK_SIZE                    000000BC  R      03
LOOK_CMKRNL_ARGLIST                000000C4  R      03
LOOK_FREE_BYTES                    000000AC  R      03
LOOK_FREE_COUNT                    000000A8  R      03
```

```
LOOK_INUSE_BYTES                000000B4 R      03
LOOK_INUSE_COUNT                000000B0 R      03
LOOK_LIST_NAME                  00000098 R      03
LOOK_LIST_SIZE                  0000009C R      03
LOOK_SIZE_ARRAY                 000000CC R      03
LOOK_SIZE_DESC                  000000B8 R      03
LOOK_XRPLIST                    0000059B R      05
LRPLIST_DESC                    00000102 R      04
LRP_NAME_DESC                   000000F7 R      04
LRP_SIZE_DESC                   0000011C R      04
MEMORY                          00000121 R      05
MEMORY_D_ALL                    00000049 R      02
MEMORY_D_FILES                  00000030 R      02
MEMORY_D_FULL                   0000003D R      02
MEMORY_D_PHYS                   00000000 R      02
MEMORY_D_POOL                   00000024 R      02
MEMORY_D_SLOTS                  00000017 R      02
MEMORY_L_BITLIS                 00000008 R      03
MEMORY_M_ALL                =   00000020
MEMORY_M_FILE               =   00000008
MEMORY_M_FULL               =   00000010
MEMORY_M_PHYS               =   00000001
MEMORY_M_POOL               =   00000004
MEMORY_M_SLOT               =   00000002
MEMORY_V_ALL                =   00000005
MEMORY_V_FILE               =   00000003
MEMORY_V_FULL               =   00000004
MEMORY_V_PHYS               =   00000000
MEMORY_V_POOL               =   00000002
MEMORY_V_SLOT               =   00000001
MEM_BAD_LIST                    0000002C R      03
MEM_BAD_PAGES                   00000030 R      03
MEM_BOOT_PAGES                  00000038 R      03
MEM_FREE_PAGES                  00000020 R      03
MEM_MB_1                        00000014 R      03
MEM_MB_DESC                     0000003C R      03
MEM_MB_TEXT                     00000044 R      03
MEM_MODF_PAGES                  00000028 R      03
MEM_OTHER_PAGES                 00000034 R      03
MEM_PHY_PAGES                   0000001C R      03
MEM_USED_PAGES                  00000024 R      03
MMG$GL_MAXPFIDX                 ********    X   05
MMG$GL_NPAGEDYN                 ********    X   05
MMG$GL_NPAGNEXT                 ********    X   05
MMG$GL_PAGSWPVC                 ********    X   05
MMG$GL_PHYPGCNT                 ********    X   05
MMG$GW_BIGPFN                   ********    X   05
MMG$GW_MINPFIDX                 ********    X   05
NDT$_MPM0                   =   00000040
NDT$_MPM1                   =   00000041
NDT$_MPM2                   =   00000042
NDT$_MPM3                   =   00000043
NPAGEDYN_DESC                   00000000 R      04
PAGEDYN_DESC                    00000025 R      04
PAGEDYN_SIZE_DESC               0000007C R      04
PAGEFILE                        000009B6 R      05
PAGE_FILE_COUNT                 00000280 R      03
PAGE_FILE_INDEX             =   0000028F R      03
PAGE_FILE_LOC                   0000028C R      03
PAGE_FILE_TABLE                 00000288 R      03
PAGE_FLAG                       000000F4 R      03
PAGE_FREE                       000000DC R      03
PAGE_FULL_PAGING_COUNT          000000F0 R      03
PAGE_FULL_SWAP_COUNT            000000EC R      03
PAGE_INDIC_DESC                 00000A89 R      04
PAGE_PFL_INDEX                  000000E8 R      03
PAGE_TOTAL                      000000E4 R      03
PAGE_USED                       000000E0 R      03
PARA_VMS                        0000005C R      03
PCB$C_STS                   =   00000024
PCB$L_WSSWP                 =   00000020
PCB$V_RES                   =   00000000
PFL$B_FLAGS                 =   00000023
PFL$K_LENGTH                =   00000024
PFL$L_BITMAPSIZ             =   00000014
PFL$L_FREPAGCNT             =   00000018
PFL$L_WINDOW                =   0000000C
PFL$V_INITED                =   00000000
PFL_K_EXT_LENGTH            =   00000044
PFL_S_DEVNAM                =   00000018
PFL_TABLE_ADDR                  00000278 R      03
PFL_TABLE_SIZE                  00000274 R      03
PFL_T_DEVNAM                    0000002C
PFL_W_FID                       00000026
PFL_W_FID_NUM                   00000026
PFL_W_FID_RVN                   0000002A
PFL_W_FID_SEQ                   00000028
PFL_W_PFL_INDEX                 00000024
PFN$AB_TYPE                     ********    X   05
PFN$AL_HEAD                     ********    X   05
PFN$AX_FLINK                    ********    X   05
PFN$C_BADPAGLST             =   00000002
PFN$GL_PHYPGCNT                 ********    X   05
PFN$V_BADPAG                =   00000005
PHV$GL_PIXBAS                   ********    X   05
PID                             0000029C R      03
POOL                            000006DB R      05
POOL_FREE                       00000080 R      03
POOL_FREE_COUNT                 00000090 R      03
POOL_FREE_LEQU_32               00000094 R      03
POOL_INUSE                      00000084 R      03
POOL_MAX_BLOCK                  00000088 R      03
POOL_MIN_BLOCK                  0000008C R      03
POOL_NAME                       00000070 R      03
POOL_NPAGEDYN                   000007A8 R      05
POOL_PAGEDYN                    000007E4 R      05
POOL_PRCALLREG                  0000083B R      05
POOL_SIZE                       00000074 R      03
POOL_TOTAL                      00000078 R      03
POOL_TOTAL_PAGES                0000007C R      03
PR$_IPL                     =   00000012
PRCALLREG_DESC                  0000004A R      04
RETURN_LENGTH                   0000024F R      03
RPB$C_MEMDSCSIZ             =   00000008
```

```
RPB$L_BADPGS                    =  00000104
RPB$L_BOOTRS                    =  00000030
RPB$L_MEMDSC                    =  000000BC
RPB$S_PAGCNT                    =  00000018
RPB$S_TR                        =  00000008
RPB$V_MPM                       =  0000000B
RPB$V_PAGCNT                    =  00000000
RPB$V_TR                        =  00000018
RPB$V_USEMPM                    =  0000000C
SCAN_BAD_LIST                      000002BD R      05
SCAN_DOUBLY_LINKED_LIST            000005BC R      05
SCAN_SINGLY_LINKED_LIST            00000877 R      05
SCH$GL_CURPCB                      ********   X    05
SCH$GL_FREECNT                     ********   X    05
SCH$GL_MAXPIX                      ********   X    05
SCH$GL_MFYCNT                      ********   X    05
SCH$GL_NULLPCB                     ********   X    05
SCH$GL_PCBVEC                      ********   X    05
SCH$GL_SWPPID                      ********   X    05
SCH$GW_PROCCNT                     ********   X    05
SCH$GW_PROCLIM                     ********   X    05
SCH$IOLOCKR                        ********   X    05
SCH$IOUNLOCK                       ********   X    05
SCH$LOCKR                          ********   X    05
SCH$UNLOCK                         ********   X    05
SCRATCH_DESC                       00000247 R      03
SGN$GL_BALSETCT                    ********   X    05
SGN$GL_IRPCNT                      ********   X    05
SGN$GL_IRPCNTV                     ********   X    05
SGN$GL_LRPCNT                      ********   X    05
SGN$GL_LRPCNTV                     ********   X    05
SGN$GL_NPAGEDYN                    ********   X    05
SGN$GL_NPAGEVIR                    ********   X    05
SGN$GL_PAGEDYN                     ********   X    05
SGN$GL_SRPCNT                      ********   X    05
SGN$GL_SRPCNTV                     ********   X    05
SGN$GW_CTLPAGES                    ********   X    05
SGN$GW_PAGFILCT                    ********   X    05
SGN$GW_SWPFILCT                    ********   X    05
SHARED_MEMORY                      00000058 R      03
SHOW$C_MEM_LONG_NAME            =  0000004E   G
SHOW$C_MEM_SHORT_NAME           =  00000028   G
SHOW$MEMORY                        00000000 RG     05
SHOW$PRCALLREG                     00000765 RG     05
SHOW$WRITE_LINE                    ********    X   05
SHOW$_MEM_READ1                    00000130 R      04
SHOW$_MEM_LOOK1                    000003EA R      04
SHOW$_MEM_LOOK2                    00000440 R      04
SHOW$_MEM_LOOK_FULL1               0000047B R      04
SHOW$_MEM_LOOK_FULL2               000004BC R      04
SHOW$_MEM_LOOK_FULL3               000004F2 R      04
SHOW$_MEM_LOOK_FULL4               0000052D R      04
SHOW$_MEM_LOOK_FULL5               00000569 R      04
SHOW$_MEM_LOOK_FULL6               00000590 R      04
SHOW$_MEM_LOOK_FULL7               000005B9 R      04
SHOW$_MEM_LOOK_FULL8               000005EC R      04
SHOW$_MEM_MEMOT                    00000164 R      04
SHOW$_MEM_MEMO2                    000001BA R      04
SHOW$_MEM_MEMO3                    00000209 R      04
SHOW$_MEM_PAGE1                    00000876 R      04
SHOW$_MEM_PAGE2                    000008CC R      04
SHOW$_MEM_PAGE3                    000008F4 R      04
SHOW$_MEM_PAGE4                    00000903 R      04
SHOW$_MEM_PAGE_FULL1               00000950 R      04
SHOW$_MEM_PAGE_FULL2               0000095F R      04
SHOW$_MEM_PAGE_FULL3               000009AD R      04
SHOW$_MEM_PAGE_FULL4               000009FB R      04
SHOW$_MEM_PAGE_FULL5               00000A49 R      04
SHOW$_MEM_PARAT                    000002A1 R      04
SHOW$_MEM_POOL1                    0000061B R      04
SHOW$_MEM_POOL2                    00000671 R      04
SHOW$_MEM_POOL_FULL1               0000069D R      04
SHOW$_MEM_POOL_FULL2               000006AA R      04
SHOW$_MEM_POOL_FULL3               000006F5 R      04
SHOW$_MEM_POOL_FULL4               00000745 R      04
SHOW$_MEM_POOL_FULL5               00000795 R      04
SHOW$_MEM_POOL_FULL6               000007DE R      04
SHOW$_MEM_POOL_FULL7               0000082A R      04
SHOW$_MEM_SLOTT                    000002F4 R      04
SHOW$_MEM_SLOT2                    0000034A R      04
SHOW$_MEM_SLOT3                    0000039A R      04
SHOW_LOOK_LIST                     00000098 R      03
SHOW_LOOK_LIST2                    0000009C R      03
SHOW_LOOK_LIST3                    00000098 R      03
SHOW_LOOK_LIST4                    00000098 R      03
SHOW_LOOK_LIST5                    000000A8 R      03
SHOW_LOOK_LIST6                    000000B0 R      03
SHOW_LOOK_LIST7                    000000B8 R      03
SHOW_LOOK_LIST8                    000000C0 R      03
SHOW_MEM_PHY                       00000014 R      03
SHOW_PAGE_LIST                     000000D8 R      03
SHOW_PAGE_LIST2                    000000DC R      03
SHOW_PAGE_LIST3                    000000E4 R      03
SHOW_PAGE_LIST4                    000000EC R      03
SHOW_PAGE_LIST5                    000000F4 R      03
SHOW_POOL_LIST                     00000070 R      03
SHOW_POOL_LIST2                    00000074 R      03
SHOW_POOL_LIST3                    00000078 R      03
SHOW_POOL_LIST4                    00000078 R      03
SHOW_POOL_LIST5                    00000080 R      03
SHOW_POOL_LIST6                    00000088 R      03
SHOW_POOL_LIST7                    00000090 R      03
SHOW_SLOTS_LIST                    00000060 R      03
SIZ...                          =  00000001
SIZE_MEMORY                        00000238 R      05
SLOTS                              000002FE R      05
SLOTS_BALANCE                      000003C9 R      05
SLOTS_FREE                         00000064 R      03
SLOTS_NONRES                       0000006C R      03
SLOTS_PCBVEC                       00000357 R      05
SLOTS_RES                          00000068 R      03
SLOTS_TOTAL                        00000060 R      03
SRPLIST_DESC                       00000096 R      04
SRP_NAME_DESC                      0000008B R      04
```

```
SRP_SIZE_DESC                     000000B0 R      04
SS$_NOMOREPROC                  = 000009A8
SS$_NORMAL                      = 00000001
SS$_NOTRAN                      = 00000629
SWAP_FILE_COUNT                   0000027C R      03
SWAP_FILE_INDEX                 = 00000293 R      03
SWAP_FILE_LOC                     00000290 R      03
SWAP_FILE_TABLE                   00000284 R      03
SWAP_INDIC_DESC                   00000A56 R      04
SYS$CMEXEC                        ******** GX     05
SYS$CMKRNL                        ******** GX     05
SYS$GETDVI                        ******** GX     05
SYS$GETJPI                        ******** GX     05
SYS$LKWSET                        ******** GX     05
SYS$TRNLOG                        ******** GX     05
SYS$WAITFR                        ******** GX     05
TOPSYS_DESC                       000000DB R      02
TRNLOG$_ACMODE                  = 00000014
TRNLOG$_DSBMSK                  = 00000018
TRNLOG$_LOGNAM                  = 00000004
TRNLOG$_NARGS                   = 00000006
TRNLOG$_RSLBUF                  = 0000000C
TRNLOG$_RSLLEN                  = 00000008
TRNLOG$_TABLE                   = 00000010
TRNLOG_LIST                       000000ED R      02
UCB                             = 0000000C
WCB$L_FCB                       = 00000018
WCB$L_ORGUCB                    = 00000010
XRPFL                           = 00000004
```

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	00000044 (68.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SHOW$RODATA	00000109 (265.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
SHOW$RWDATA	000002C8 (712.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
SHOW$MSG_TEXT	00000AC5 (2757.)	04 (4.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC BYTE
SHOW$CODE	00000D8F (3471.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	12	00:00:00.10	00:00:01.05
Command processing	75	00:00:00.86	00:00:05.85
Pass 1	556	00:00:22.54	00:01:11.34
Symbol table sort	0	00:00:03.30	00:00:10.76
Pass 2	397	00:00:06.56	00:00:24.60
Symbol table output	27	00:00:00.31	00:00:01.02
Psect synopsis output	0	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00

```
Assembler run totals          1069     00:00:33.70    00:01:54.65

The working set limit was 2250 pages.
140545 bytes (275 pages) of virtual memory were used to buffer the intermediate code.
There were 120 pages of symbol table space allocated to hold 2086 non-local and 68 local symbols.
2078 source lines were read in Pass 1, producing 45 object records in Pass 2.
49 pages of virtual memory were used to define 45 macros.
```

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

Macro library name	Macros defined
_255DUA28:[CLIUTL.OBJ]CLIUTL.MLB;1	0
_255DUA28:[SYS.OBJ]LIB.MLB;1	15
_255DUA28:[SYSLIB]STARLET.MLB;2	26
TOTALS (all libraries)	41

```
2114 GETS were required to define 41 macros.

There were no errors, warnings or information messages.

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

SHOMSGUTL LIS

SHONET LIS

SHOWAUDIT LIS

SHOWID LIS

SHOWLOG LIS

SHOWERROR LIS

SHOWFILES LIS

SHOMEMORY LIS