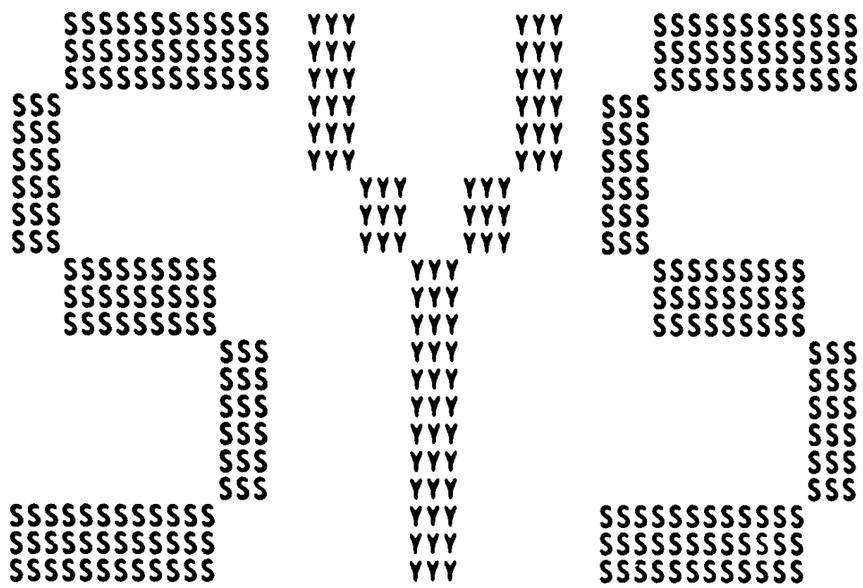

```
PPPPPPPP      AAAAAA      GGGGGGGG  EEEEEEEEEE  FFFFFFFFFF    IIIIII    LL          EEEEEEEEEE
PPPPPPPP      AAAAAA      GGGGGGGG  EEEEEEEEEE  FFFFFFFFFF    IIIIII    LL          EEEEEEEEEE
PP      PP   AA    AA   GG          EE          FF              II      LL          EE
PP      PP   AA    AA   GG          EE          FF              II      LL          EE
PP      PP   AA    AA   GG          EE          FF              II      LL          EE
PP      PP   AA    AA   GG          EE          FF              II      LL          EE
PPPPPPPP     AA    AA   GG          EEEEEEEE    FFFFFFFF        II      LL          EEEEEEEE
PPPPPPPP     AA    AA   GG          EEEEEEEE    FFFFFFFF        II      LL          EEEEEEEE
PP           AAAAAAAAAA GG  GGGGGG  EE          FF              II      LL          EE
PP           AAAAAAAAAA GG  GGGGGG  EE          FF              II      LL          EE
PP           AA    AA   GG      GG  EE          FF              II      LL          EE              ....
PP           AA    AA   GG      GG  EE          FF              II      LL          EE              ....
PP           AA    AA     GGGGGG    EEEEEEEEEE  FF            IIIIII    LLLLLLLLLL  EEEEEEEEEE      ....
PP           AA    AA     GGGGGG    EEEEEEEEEE  FF            IIIIII    LLLLLLLLLL  EEEEEEEEEE      ....

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

Table of contents

00

```
0000     1          .TITLE  PAGEFILE - ALLOCATE / DEALLOCATE PAGING FILE
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:     EXECUTIVE, MEMORY MANAGEMENT SUBROUTINES
0000    29 ;
0000    30 ; ABSTRACT:
0000    31 ;
0000    32 ;       THIS MODULE CONTAINS THE ROUTINES FOR ALLOCATING AND DEALLOCATING
0000    33 ;       PAGES FROM A PAGING FILE.
0000    34 ;
0000    35 ; ENVIRONMENT:
0000    36 ;
0000    37 ;       THESE ROUTINES RUN IN KERNEL MODE AND MUST BE CALLED WITH
0000    38 ;       IPL AT SYNCH OR HIGHER.
0000    39 ;
0000    40 ;--
0000    41 ;
0000    42          .SBTTL  HISTORY                 ; DETAILED
0000    43 ;
0000    44 ; AUTHOR: PETER H. LIPMAN       , CREATION DATE: 29-OCT-76
0000    45 ;
0000    46 ; MODIFIED BY:
0000    47 ;
0000    48 ;       V03-004 WMC00001        Wayne Cardoza           09-Jul-1984
0000    49 ;               Make the pagefile full messages more accurate.
0000    50 ;
0000    51 ;       V03-003 KDM0002         Kathleen D. Morse       28-Jun-1982
0000    52 ;               Add $DYNDEF.
0000    53 ;
0000    54 ;**
0000    55
```

```
                     0000     57          .SBTTL  DECLARATIONS
                     0000     58
                     0000     59 ;
                     0000     60 ; INCLUDE FILES:
                     0000     61 ;
                     0000     62          $DYNDEF                         ;DYNAMIC DATA STRUCTURE TYPE DEFINITIONS
                     0000     63          $PFLDEF                         ;PAGE FILE CONTROL BLOCK DEFINITIONS
                     0000     64          $PTEDEF                         ;PAGE TABLE ENTRY DEFINITIONS
                     0000     65          $RSNDEF                         ;RESOURCE NAME DEFINITIONS
                     0000     66 ;
                     0000     67 ; EXTERNAL SYMBOLS:
                     0000     68 ;
                     0000     69
                     0000     70 ;
                     0000     71 ; MACROS:
                     0000     72 ;
                     0000     73
                     0000     74 ;
                     0000     75 ; EQUATED SYMBOLS:
                     0000     76 ;
                     0000     77
                     0000     78 ;
                     0000     79 ; OWN STORAGE:
                     0000     80 ;
                     0000     81
                 00000000     82          .PSECT  $$$220,LONG             ; SWAPPER/SCHEDULER DATA
                     0000     83          .ALIGN  LONG                    ;
                     0000     84
                     0000     85 MMG$GL_NULLPFL::                         ; NULL PFL SERVES AS PLACEHOLDER
00000000             0000     86          .LONG   0                       ; BITMAP POINTER, 0 IF TABLE NOT IN USE
00000000             0004     87          .LONG   0                       ; ADDRESS OF MPW_WRTCLUSTER SIZE AREA
    0024             0008     88          .WORD   PFL$C_LENGTH
      23             000A     89          .BYTE   DYN$C_PFL
      00             000B     90          .BYTE   0                       ; PAGE FAULT CLUSTER
00000000             000C     91          .LONG   0                       ; WINDOW POINTER, *** FILLED IN BY INIT
00000000             0010     92          .LONG   0                       ; VBN,            *** FILLED IN BY INIT
00000000             0014     93          .LONG   0                       ; BITMAP SIZE
00000000             0018     94          .LONG   0                       ; FREE PAGE COUNT IN THIS FILE
003FFFFF             001C     95          .LONG   PTE$M_PGFLVB            ; PAGE FILE VBN MASK
00000000             0020     96          .LONG   0                       ; ACCOUNT FOR EXTENDED LENGTH
                     0024     97 ;
                     0024     98 ; POINTER TO VECTOR OF PAGE/SWAP FILE CONTROL BLOCKS
                     0024     99 ;
                     0024    100
                     0024    101 MMG$GL_PAGSWPVC::
00000000             0024    102          .LONG   0
                     0028    103
                     0028    104 ;
                     0028    105 ; MAXIMUM PAGE FILE INDEX CURRENTLY IN USE
                     0028    106 ;
                     0028    107
                     0028    108 MMG$GL_MAXPFIDX::
00000000             0028    109          .LONG   0
                     002C    110
                     002C    111 MMG$GW_MINPFIDX::
                     002C    112 SGN$GW_SWPFILCT::                       ; Count of swapfile slots
    0000             002C    113          .WORD   0
```

```
                002E    114
                002E    115 ;
                002E    116 ; Most of the routines in this module are permanently resident
                002E    117 ;
 00000000       118          .PSECT  $MMGCOD
```

```
0000   120          .SBTTL  ALLOCSWPAREA - ALLOCATE A SWAP AREA IN A PAGE FILE
0000   121
0000   122 ;++
0000   123 ; FUNCTIONAL DESCRIPTION:
0000   124 ;
0000   125 ;       THIS ROUTINE ALLOCATES A CLUSTER OF PAGES FROM THE SPECIFIED PAGE FILE.
0000   126 ;
0000   127 ; CALLING SEQUENCE:
0000   128 ;
0000   129 ;       BSBW    MMG$ALLOCSWPAREA
0000   130 ;
0000   131 ; INPUT PARAMETERS:
0000   132 ;
0000   133 ;       r0 = VBN in paging file representing start of current allocation
0000   134 ;       r1 = current allocation size
0000   135 ;       r2 = new request size
0000   136 ;
0000   137 ; IMPLICIT INPUTS:
0000   138 ;
0000   139 ;       none
0000   140 ;
0000   141 ; OUTPUT PARAMETERS:
0000   142 ;
0000   143 ;       r0 = page file vbn (greater than 0) if successful
0000   144 ;       r2 = number of pages allocated
0000   145 ;       r1,r3 destroyed
0000   146 ;
0000   147 ; IMPLICIT OUTPUTS:
0000   148 ;
0000   149 ;       none
0000   150 ;
0000   151 ; COMPLETION CODES:
0000   152 ;
0000   153 ;       positive condition code indicates success
0000   154 ;       negative condition code indicates failure
0000   155 ;       zero condition code indicates failure because request too early in boot
0000   156 ;
0000   157 ; SIDE EFFECTS:
0000   158 ;
0000   159 ;       none
0000   160 ;
0000   161 ;--
```

```
                                0000   163 mmg$allocswparea::
                          30    BB   0000   164          pushr   #^m<r4,r5>                      ; save work registers
                          55    D4   0002   165          clrl    r5                              ; indicator for no paging files at all
                    54    01    D0   0004   166          movl    #1,r4                           ; start scan at file index 1
              53    0024'DF44    D0   0007   167 10$:     movl    @w^mmg$gl_pagswpvc[r4],r3 ; get address of next page file block
               10 23 A3   00     E0   000D   168          bbs     #pfl$v_inited,pfl$b_flags(r3),30$ ;branch if this one inited
        ED 54    00000028'GF     F3   0012   169 20$:     aobleq  g^mmg$gl_maxpfidx,r4,10$ ; loop through all page files
                    52    01    CE   001A   170          mnegl   #1,r2                           ; assume unlimited growth size
                    50    55    D0   001D   171          movl    r5,r0                           ; set flag indicating if we are booting
                          13    11   0020   172          brb     40$                             ; use common exit
                          55    D7   0022   173 30$:     decl    r5                              ; indicate valid paging file exists
                    0FF0  8F    BB   0024   174          pushr   #^m<r4,r5,r6,r7,r8,r9,r10,r11> ;save volatile registers
                          0E    10   0028   175          bsbb    mmg$allocpagfil                 ; allocate new area in page file
                    0FF0  8F    BA   002A   176          popr    #^m<r4,r5,r6,r7,r8,r9,r10,r11> ;restore volatile registers
                          E2    13   002E   177          beql    20$                             ; try next page file
           50   08    18    54   F0   0030   178          insv    r4,#24,#8,r0                    ; save swap file index
                          30    BA   0035   179 40$:     popr    #^m<r4,r5>                      ; restore registers
                                05   0037   180          rsb                                     ; return to caller
```

```
0038    182          .SBTTL  ALLOCPAGFIL - ALLOCATE A PAGING FILE SPACE
0038    183
0038    184 ;++
0038    185 ; FUNCTIONAL DESCRIPTION:
0038    186 ;
0038    187 ;       THIS ROUTINE ALLOCATES A CLUSTER OF PAGES FROM THE SPECIFIED PAGE FILE.
0038    188 ;
0038    189 ; CALLING SEQUENCE:
0038    190 ;
0038    191 ;       BSBW    MMG$ALLOCPAGFIL1
0038    192 ;
0038    193 ; INPUT PARAMETERS:
0038    194 ;
0038    195 ;       r0 = VBN in paging file representing start of current allocation
0038    196 ;       r1 = current allocation size
0038    197 ;       r2 = new request size
0038    198 ;       r3 = page file index
0038    199 ;
0038    200 ; IMPLICIT INPUTS:
0038    201 ;
0038    202 ;       NONE
0038    203 ;
0038    204 ; OUTPUT PARAMETERS:
0038    205 ;
0038    206 ;       R0 = PAGE FILE VBN (GREATER THAN 0) IF SUCCESSFUL
0038    207 ;       R2 = NUMBER OF PAGES ALLOCATED
0038    208 ;
0038    209 ; IMPLICIT OUTPUTS:
0038    210 ;
0038    211 ;       NONE
0038    212 ;
0038    213 ; COMPLETION CODES:
0038    214 ;
0038    215 ;       Z-BIT SET IF FAILURE
0038    216 ;       Z-BIT 0 IF SUCCESS
0038    217 ;
0038    218 ; SIDE EFFECTS:
0038    219 ;
0038    220 ;       MMG$ALLOCPAGFIL2 has register content dependencies on this routine!
0038    221 ;
0038    222 ;       This routine depends on allocation sizes to be multiples of 8 for
0038    223 ;       reasonable search times now that this is first fit.  This implies
0038    224 ;       that the modified page writer cluster size should be equal to
0038    225 ;       the swap space allocation increment, to allow the local "memory"
0038    226 ;       to work reasonably.  Also the minimum modified page writer cluster size
0038    227 ;       should be at least 16 blocks for correct resource failure continuation,
003'    228 ;       this allows some emergency 8 byte blocks to be allocated.
0038    229 ;
0038    230 ;--
```

```
                                0038   232 MMG$ALLOCPAGFIL1::
                          56    63   D0  0038   233          movl    pfl$l_bitmap(r3),r6     ;address of start of map
                    57    14   A3   D0  003B   234          movl    pfl$l_bitmapsiz(r3),r7  ;number of bytes in map
                5A  57          56   C1  003F   235          addl3   r6,r7,r10               ;get end of map address
                    5B          01   CE  0043   236          mnegl   #1,r11                  ;materialize a minus for use
                                0F   BB  0046   237 20$:     pushr   #^m<r0,r1,r2,r3>        ;save the inputs, (r3 is now address)
             58   52    FD      8F   78  0048   238          ashl    #-3,r2,r8               ;make size into byte count
          59   50    08        18   EF  004D   239          extzv   #24,#8,r0,r9            ;get the page file index
             53   0024'DF49          D1  0052   240          cmpl    aw^mmg$gl_pagswpvc[r9],r3 ;is this in the same page file?
                                47   12  0058   241          bneq    60$                     ;branch if no., try for simple allocate
          50   50    18        00   EF  005A   242          extzv   #0,#24,r0,r0            ;get the input VBN
                                40   13  005F   243          beql    60$                     ;branch if not holding current space
                5A    51        50   C1  0061   244          addl3   r0,r1,r9                ;get ending block
             59   59    FD      8F   78  0065   245          ashl    #-3,r9,r9               ;get byte offset of area after this one
                                     006A   246                                          ;r0+r1 always yield (multiple of 8)+1
             54   51    FD      8F   78  006A   247          ashl    #-3,r1,r4               ;current size in groups of 8
                55    58        54   C3  006F   248          subl3   r4,r8,r5                ;number of additional needed blocks
                                09   14  0073   249          bgtr    30$                     ;branch if this is an expansion
                             01CB    30  0075   250          bsbw    mmg$deallocpagfil       ;free current holding if contraction
                                0F   BA  0078   251          popr    #^m<r0,r1,r2,r3>        ;restore regs
                                50   7C  007A   252          clrq    r0                      ;indicate holding freed
                                C8   11  007C   253          brb     20$                     ;now do the allocation
                                     007E   254 ;
                                     007E   255 ; The end of map condition is handled by having a non-allocatable byte at
                                     007E   256 ; the end of the map.  This allows the skpc to failure terminate.
                                     007E   257 ;
                6649    55    5B    3B  007E   258 30$:     skpc    r11,r5,(r6)[r9]         ;find additional contiguous free space
                                1C   12  0083   259          bneq    60$                     ;branch if non-free blocks in area
          6649  55    00    61    00  2C  0085   260          movc5   #0,(r1),#0,r5,(r6)[r9]  ;mark these blocks allocated
                                     008C   261 ;
                                     008C   262 ; It is safe not to update STARTBYTE down this path since this is an allocate.
                                     008C   263 ; This is also probably desirable to lessen start of map searches.
                                     008C   264 ;
                                0F   BA  008C   265          popr    #^m<r0,r1,r2,r3>        ;restore regs
                                     008E   266                                          ;note input VBN is output VBN
                   51   52    C2        008E   267          subl    r2,r1                   ;get additionally allocated blocks
                                     0091   268                                          ;(count is negative)
             18 A3 51   C0            0091   269          addl    r1,pfl$l_frepagcnt(r3)  ;update count of available pages
                                04   B9  0095   270          bicpsw  #4                      ;indicate success
                                05   0097   271          rsb                             ;return VBN in R0, count in R2, z-bit=0
                                     0098   272 ;
                                     0098   273 ; allocation failure return
                                     0098   274 ;
                                0F   BA  0098   275 40$:     popr    #^m<r0,r1,r2,r3>        ;restore regs
             23 A3              04   B8  009A   276          bisb    #pfl$m_swpfilful,pfl$b_flags(r3) ;set flag indicating file full
                                04   B8  009E   277          bispsw  #4                      ;indicate failure, no deallocation!
                                05   00A0   278          rsb                             ;z-bit set
                                     00A1   279 ;
                                     00A1   280 ; new allocation
                                     00A1   281 ;
                   55   5B    3C        00A1   282 60$:     movzwl  r11,r5                  ;set up for 65536 byte locate
             22 A3 52   91            00A4   283          cmpb    r2,pfl$b_allocsiz(r3)   ;is this standard request size?
                                06   19  00A8   284          blss    70$                     ;branch if not, search from start
          51    04 A3              00AA   285          movl    pfl$l_startbyte(r3),r1  ;set up to start from first known free
                                03   12  00AE   286          bneq    80$                     ;branch if we know where
                   51   56    D0        00B0   287 70$:     movl    r6,r1                   ;set up to scan map from start
                                     00B3   288
```

```
                57   5A   51   C3   00B3   289 80$:    subl3   r1,r10,r7               ;calc number of bytes remaining to scan
                          DF   13   00B7   290         beql    40$                     ;branch if at end of map
                     55   57   D1   00B9   291         cmpl    r7,r5                   ;less than 65536 bytes to scan?
                          03   18   00BC   292         bgeq    90$                     ;branch if not
                     55   57   D0   00BE   293         movl    r7,r5                   ;set scan amount to what's left
                61   55   5B   3A   00C1   294 90$:    locc    r11,r5,(r1)             ;find a byte aligned area with 8 blocks
                          EC   13   00C5   295         beql    80$                     ;branch if no free clusters in area
                                   00C7   296 ;
                                   00C7   297 ; The end of map condition is handled by having a non-allocatable byte at
                                   00C7   298 ; the end of the map.  This allows the skpc to failure terminate.
                                   00C7   299 ;
                6.   58   5B   3B   00C7   300         skpc    r11,r8,(r1)             ;is this sequence long enough?
                          E6   12   00CB   301         bneq    80$                     ;branch if not, look for another
                     51   58   C2   00CD   302         subl    r8,r1                   ;get back start address of field
          61   58   00   61   00   2C   00D0   303         movc5   #0,(r1),#0,r8,(r1)      ;allocate area, preserve r1 address
                57   51   56   C3   00D6   304         subl3   r6,r1,r7                ;save start byte to return it
                     59   53   D0   00DA   305         movl    r3,r9                   ;save address of end of this area
                          03   BA   00DD   306         popr    #^m<r0,r1>              ;restore regs for deallocations, if any
          53   50   08   18   EF   00DF   307         extzv   #24,#8,r0,r3            ;get the page file index
          50   50   18   00   EF   00E4   308         extzv   #0,#24,r0,r0            ;get the input VBN
                          09   13   00E9   309         beql    95$                     ;branch if no previous holding
             53   0024'DF43   D0   00EB   310         movl    @w^mmg$gl_pagswpvc[r3],r3 ;get page file control block address
                     014F   30   00F1   311         bsbw    mmg$deallocpagfil       ;free up the space
                       0C   BA   00F4   312 95$:    popr    #^m<r2,r3>              ;restore the request size, PFL addr
                   22 A3   52   91   00F6   313         cmpb    r2,pfl$b_allocsiz(r3)   ;was this for current request size
                          04   12   00FA   314         bneq    100$                    ;branch if not, don't affect memory
                   04 A3   59   D0   00FC   315         movl    r9,pfl$l_startbyte(r3)  ;update memory for future reference
                   18 A3   52   C2   0100   316 100$:   subl    r2,pfl$l_frepagcnt(r3)  ;update count of available pages
                50   57   03   78   0104   317         ashl    #3,r7,r0                ;multiply byte number*8 to get VBN
                          50   D6   0108   318         incl    r0                      ;VBN's need to be based at 1
                               05   010A   319         rsb                             ;return, z-bit=0
                                   010B   320
                                   010B   321 BADALLOC:
                                   010B   322         BUG_CHECK BADPAGFILA,FATAL      ;BAD PAGE FILE ADDRESS ALLOCATED
                                   010F   323         .SBTTL  ALLOCPAGFIL - ALLOCATE A PAGING FILE SPACE
```

```
                        ;++
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       THIS ROUTINE ALLOCATES THE FIRST CONTIGOUS SET OF BLOCKS FROM
                        ;       THE SPECIFIED PAGE FILE.
                        ;
                        ; CALLING SEQUENCE:
                        ;
                        ;       BSBW    MMG$ALLOCPAGFIL2        ; must occur just after a call
                        ;                                       ; to MMG$ALLOCPAGFIL
                        ;
                        ; INPUT PARAMETERS:
                        ;
                        ;       r3 = page file control block address
                        ;       r6 = address of start of bitmap
                        ;       r10= end address of bitmap
                        ;       r11= 65536 (maximum size for a string instruction length)
                        ;
                        ; IMPLICIT INPUTS:
                        ;
                        ;       NONE
                        ;
                        ; OUTPUT PARAMETERS:
                        ;
                        ;       R0 = PAGE FILE VBN (GREATER THAN 0) IF SUCCESSFUL
                        ;       R2 = NUMBER OF PAGES ALLOCATED
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;
                        ;       NONE
                        ;
                        ; COMPLETION CODES:
                        ;
                        ;       Z-BIT SET IF FAILURE
                        ;       Z-BIT 0 IF SUCCESS
                        ;
                        ; SIDE EFFECTS:
                        ;
                        ;       none
                        ;--
```

```
                                                010F    366 fragmsg:
                                    00000045'   010F    367             .long   20$-10$
                                    00000117'   0113    368             .long   10$
                                                0117    369 10$:
2D 57 2D 4D 45 54 53 59 53 25 0A 0D             0117    370             .ascii  <13><10>-
61 50 20 2C 47 41 52 46 45 47 41 50             0123
6C 64 61 62 20 65 6C 69 66 20 65 67             012F
64 65 74 6E 65 6D 67 61 72 66 20 79             013B
6E 6F 63 20 6D 65 74 73 79 73 20 2C             0147
                  67 6E 69 75 6E 69 74          0153
                                                015A    371             \%SYSTEM-W-PAGEFRAG, Page file badly fragmented, system continuing\-
                                        0A 0D   015A    372                     <13><10>
                                                015C    373 20$:
                                                015C    374 critmsg:
                                    0000004B'   015C    375             .long   40$-30$
                                    00000164'   0160    376             .long   30$
                                          0D    0164    377 30$:        .ascii  <13><10>-
50 2D 57 2D 4D 45 54 53 59 53 25 0A             0165    378             \%SYSTEM-W-PAGECRIT, Page file space critical, system trying to continue\-
67 61 50 20 2C 54 49 52 43 45 47 41             0171
65 63 61 70 73 20 65 6C 69 66 20 65             017D
73 20 2C 6C 61 63 69 74 69 72 63 20             0189
67 6E 69 79 72 74 20 6D 65 74 73 79             0195
65 75 6E 69 74 6E 6F 63 20 6F 74 20             01A1
                                        0A 0D   01AD    379                     <13><10>
                                                01AF    380 40$:
```

```
                                01AF    382 MMG$ALLOCPAGFIL2::
                        55  5B  3C  01AF  383          movzwl  r11,r5                  ;set up for 65536 byte locate
                        51  56  D0  01B2  384          movl    r6,r1                   ;set up to scan map from start
                                    01B5  385
                    57  5A  51  C3  01B5  386 10$:     subl3   r1,r10,r7               ;calc number of bytes remaining to scan
                            66  13  01B9  387          beql    50$                     ;branch if at end of map
                        55  57  D1  01BB  388          cmpl    r7,r5                   ;less than 65536 bytes to scan?
                            03  18  01BE  389          bgeq    20$                     ;branch if not
                        55  57  D0  01C0  390          movl    r7,r5                   ;set scan amount to what's left
                    61  55  00  3B  01C3  391 20$:     skpc    #0,r5,(r1)              ;find any free blocks
                            EC  13  01C7  392          beql    10$                     ;branch if no free clusters in area
                50  61  08  00  EA  01C9  393          ffs     #0,#8,(r1),r0           ;find the free block
                    52  50  01  C3  01CE  394          subl3   #1,r0,r2                ;save start offset
                            52  D6  01D2  395 30$:     incl    r2                      ;account for block
                      FA 61  52  E4  01D4  396          bbsc    r2,(r1),30$             ;loop through contiguous portion of map
                        52  50  C2  01D8  397          subl    r0,r2                   ;set r2 number of blocks allocated
                      18 A3  52  C2  01DB  398          subl    r2,pfl$l_frepagcnt(r3)  ;update count of available pages
                        51  56  C2  01DF  399          subl    r6,r1                   ;get byte number of free blocks
                    50  01 A041  7E  01E2  400          movaq   1(r0)[r1],r0            ;form 8*byte number + bit number + 1
                51  14 A3  01  78  01E7  401          ashl    #1,pfl$l_bitmapsiz(r3),r1 ;find 1/4 point of VBN's in bitmap
                        51  50  D1  01EC  402          cmpl    r0,r1                   ;is this allocation past 1/4 point?
                            30  1F  01EF  403          blssu   50$                     ;branch if not, no message needed yet
            0B 0000'CF      00' E2  01F1  404          bbss    s^#exe$v_pgflfrag,w^exe$gl_flags,40$ ;branch if reported
                            07  BB  01F7  405          pushr   #^m<r0,r1,r2>           ;save registers
                51  FF12   CF  7D  01F9  406          movq    fragmsg,r1              ;set up message to output
                            22  10  01FE  407          bsbb    sendmsg                 ;output the message
                            07  BA  0200  408          popr    #^m<r0,r1,r2>           ;restore registers
                54  51  51  C1  0202  409 40$:     addl3   r1,r1,r4                ;find 3/4 mark in file
                        54  51  C0  0206  410          addl    r1,r4                   ;now have 3/4 VBN
                        54  50  D1  0209  411          cmpl    r0,r4                   ;is this allocation past 3/4 point
                            13  1F  020C  412          blssu   50$                     ;branch if not
            0D 0000'CF      00' E2  020E  413          bbss    s^#exe$v_pgflcrit,w^exe$gl_flags,50$ ;branch if reported
                            07  BB  0214  414          pushr   #^m<r0,r1,r2>           ;save registers
                51  FF42   CF  7D  0216  415          movq    critmsg,r1              ;set up message to output
                            05  10  021B  416          bsbb    sendmsg                 ;output the message
                            07  BA  021D  417          popr    #^m<r0,r1,r2>           ;restore registers
                            04  B9  021F  418          bicpsw  #4                      ;indicate success
                                05  0221  419 50$:     rsb                             ;return, z-bit=0 success, else failure
                                    0222  420
                                    0222  421 sendmsg:
                55  0000'CF        9E  0222  422          movab   w^opa$ucb0,r5           ;set console terminal for broadcast
                        FDD6'       31  0227  423          brw     ioc$broadcast           ;assume message will get to console
```

```
                        .SBTTL  DALCPAGFIL - DEALLOCATE PAGE IN PAGING FILE

                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       THIS ROUTINE DEALLOCATES A SPECIFIED PAGE IN THE SPECIFIED
                ;       PAGING FILE.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       BSBW    MMG$DALCPAGFIL
                ;
                ; INPUT PARAMETERS:
                ;
                ;       R0 = PAGE FILE VBN TO DEALLOCATE
                ;       R3 = PAGE FILE INDEX
                ;
                ; IMPLICIT INPUTS:
                ;       NONE
                ;
                ; OUTPUT PARAMETERS:
                ;       R0,R1,R2 DESTROYED
                ;       R3 = ADDRESS OF PAGE FILE CONTROL BLOCK
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       IF THE SPECIFIED PAGING FILE BECOMES NON-EMPTY, THE RESOURCE
                ;       AVAILABLE SIGNAL IS ISSUED FOR THE RSN$_PGFILE RESOURCE
                ;
                ; COMPLETION CODES:
                ;       NONE
                ;
                ; SIDE EFFECTS:
                ;       NONE
                ;
                ;--
```

```
                                022A   462          .ENABLE lsb
                                022A   463
                                022A   464 5$:      ;check for checkpoint bit
                          08 50  15  E1 022A   465          bbc     #pte$v_chkpnt,r0,10$    ;checkpoint bit set?
                       03 1C A3  15  E0 022E   466          bbs     #pte$v_chkpnt,pfl$l_maxvbn(r3),10$  ;branch if not a small file
                                 03  BA 0233   467          popr    #^m<r0,r1>              ;clean up
                                 05  05 0235   468          rsb                             ;ignore the deallocation request
                                     0236   469
                                     0236   470 10$:     BUG_CHECK BADPAGFILD,FATAL      ;BAD PAGE FILE ADDRESS DEALLOCATED
                                     023A   471
                                     023A   472 ; r0 = VBN of block to return
                                     023A   473 ; r3 = page file index
                                     023A   474
                                     023A   475 MMG$DALCPAGFIL::
                          53  0024'DF43  D0 023A   476          movl    @w^mmg$gl_pagswpvc[r3],r3 ;get page file control block address
                                51   01  D0 0240   477          movl    #1,r1                   ;set count to 1
                                     0243   478                                          ;fall through
                                     0243   479 ; r0 = VBN of start block to return
                                     0243   480 ; r1 = count
                                     0243   481 ; r3 = address of page file control block
                                     0243   482
                                     0243   483 MMG$DEALLOCPAGFIL::
                                 50  D7 0243   484          decl    r0                      ;get VBN to base 0
                                 EF  19 0245   485          blss    10$                     ;branch if VBN passed was 0
                                 03  BB 0247   486          pushr   #^m<r0,r1>              ;save for later
                             52  51  50  C1 0249   487          addl3   r0,r1,r2                ;high mark for deallocation
                                 52  D7 024D   488          decl    r2                      ;account for count in 0 origin
                          52  52   FD  8F 78 024F   489          ashl    #-3,r2,r2               ;byte # in map
                                14 A3  52  D1 0254   490          cmpl    r2,pfl$l_bitmapsiz(r3)  ;legal page file VBN?
                                 D0  1E 0258   491          bgequ   5$                      ;branch if illegal
                                 52  20 D0 025A   492          movl    #32,r2                  ;max number single insv can set
                                 51  52 D1 025D   493 30$:     cmpl    r2,r1                   ;free more than 32?
                                 03  15 0260   494          bleq    40$                     ;branch if yes
                                 52  51 D0 0262   495          movl    r1,r2                   ;set max number to free
                          00   00 B3  52  50 EC 0265   496 40$:     cmpv    r0,r2,@pfl$l_bitmap(r3),#0 ;temp check for safety
                                     C9  12 026B   497          bneq    10$                     ;bugcheck if any of these bit set
                00 B3  52  50  FFFFFFFF  8F F0 026D   498          insv    #-1,r0,r2,@pfl$l_bitmap(r3) ;set the bits
                                 50  52 C0 0277   499          addl    r2,r0                   ;update to next VBN sequence
                                18 A3  52 C0 027A   500          addl    r2,pfl$l_frepagcnt(r3)  ;count free pages
                                 51  52 C2 027E   501          subl    r2,r1                   ;number of blocks to still free
                                     DA  12 0281   502          bneq    30$                     ;loop through entire set
                                     03  BA 0283   503          popr    #^m<r0,r1>              ;get back VBN and free count
                             50  50   FD  8F 78 0285   504          ashl    #-3,r0,r0               ;set up to check for 8 block unit freed
                                 51  0E C0 028A   505          addl    #14,r1                  ;round count for worst case crossing
                             51  51   FD  8F 78 028D   506          ashl    #-3,r1,r1               ;number of bytes to check
                          00 B340  51   FF  8F 3A 0292   507          locc    #-1,r1,@pfl$l_bitmap(r3)[r0] ;any whole cluster become free?
                                     3E  13 0299   508          beql    60$                     ;branch if not
                                04 A3  51  D1 029B   509          cmpl    r1,pfl$l_startbyte(r3)  ;is freed cluster earlier in map?
                                     38  1A 029F   510          bgtru   60$                     ;branch if not, note bgtru not bgequ
                                 50  71 92 02A1   511 50$:     mcomb   -(r1),r0                ;find start byte of free area
                                     FB  13 02A4   512          beql    50$                     ;loop
                            52   51  01  C1 02A6   513          addl3   #1,r1,r2                ;set start of area
                              51   22 A3  9A 02AA   514          movzbl  pfl$b_allocsiz(r3),r1   ;get current cluster size for this file
                          51  51   FD  8F 78 02AE   515          ashl    #-3,r1,r1               ;get it in bytes rather than blocks
                          62  51   FF  8F 3B 02B3   516          skpc    #-1,r1,(r2)             ;does the area qualify?
                                     1F  12 02B8   517          bneq    60$                     ;branch if not
                                04 A3  52  D0 02BA   518          movl    r2,pfl$l_startbyte(r3)  ;save new starting pointer
```

```
0000'CF    22 A3    91   02BE    519          cmpb    pfl$b_allocsiz(r3),w^mpw$gw_mowpfc ;are we at maximum size
                         02C4    520                                                     ;we should ever try allocations for?
                 04      13   02C4    521          beql    55$                        ;branch if at maximum
              22 A3      08   80   02C6    522          addb    #8,pfl$b_allocsiz(r3)  ;try next higher size next time
           0A 23 A3      02   E5   02CA    523 55$:     bbcc    #pfl$v_swpfilful,pfl$b_flags(r3),60$ ;branch if not transition
                 08      BB   02CF    524          pushr   #^m<r3>                    ;save pfl address
                 50      0A   D0   02D1    525          movl    #rsn$_swpfile,r0       ;set up to return swap file available
                    FD29'    30   02D4    526          bsbw    sch$ravail             ;signal resource available
                 08      BA   02D7    527          popr    #^m<r3>                    ;restore pfl address
                         05   02D9    528 60$:     rsb                                ;return
                              02DA    529
                              02DA    530          .DISABLE lsb
```

```
02DA    532          .SBTTL  ALC_PGFLVBN     Allocate specific blocks in paging file
02DA    533
02DA    534 ;++
02DA    535 ; FUNCTIONAL DESCRIPTION:
02DA    536 ;
02DA    537 ;       This routine allocates a specific set of blocks in a paging file
02DA    538 ;
02DA    539 ; CALLING SEQUENCE:
02DA    540 ;
02DA    541 ;       BSBW    MMG$ALC_PGFLVBN
02DA    542 ;
02DA    543 ; INPUT PARAMETERS:
02DA    544 ;
02DA    545 ;       R0 = VBN of first block to be allocated
02DA    546 ;       R1 = Page file index
02DA    547 ;       R2 = Number of consecutive blocks to be allocated
02DA    548 ;
02DA    549 ; IMPLICIT INPUTS:
02DA    550 ;       none
02DA    551 ;
02DA    552 ; OUTPUT PARAMETERS:
02DA    553 ;       none
02DA    554 ;
02DA    555 ; IMPLICIT OUTPUTS:
02DA    556 ;       none
02DA    557 ;
02DA    558 ; COMPLETION CODES:
02DA    559 ;       NONE
02DA    560 ;
02DA    561 ; SIDE EFFECTS:
02DA    562 ;       NONE
02DA    563 ;
02DA    564 ;--
```

```
                        00000000   566          .PSECT  Y$LOWUSE                ;This code can page
                              0000 567
                              0000 568 MMG$ALC_PGFLVBN::
51   00000024'FF41    D0   0000 569          MOVL    @L^MMG$GL_PAGSWPVC[R1],R1 ;Get base address from index
                 53    DD   0008 570          PUSHL   R3                      ;Get a scratch regtster
                 50    D7   000A 571          DECL    R0                      ;Bit # is base 0
   53    50    FD 8F    78   000C 572 10$:     ASHL    #-3,R0,R3               ;Byte # in bit map
         14 A1    53    D1   0011 573          CMPL    R3,PFL$L_BITMAPSIZ(R1)  ;Legal page file vbn?
                  05    1E   0015 574          BGEQU   20$                     ;Branch if illegal
      04 00 B1    50    E4   0017 575          BBSC    R0,@PFL$L_BITMAP(R1),30$ ;Free the page and branch
                              001C 576 20$:
                              001C 577          BUG_CHECK BADPAGFILD,FATAL      ;Bad page file address specified
                              0020 578 30$:
               18 A1    D7   0020 579          DECL    PFL$L_FREPAGCNT(R1)     ;Count another free page
                  50    D6   0023 580          INCL    R0                      ;Point to next VBN in file
               E4 52    F5   0025 581          SOBGTR  R2,10$                  ;Go back if not done yet
                  52    D7   0028 582          DECL    R2                      ;Form a minus 1
         52    18 A1    D1   002A 583          CMPL    PFL$L_FREPAGCNT(R1),R2  ;Insure that counts still consistent
                  EC    19   002E 584          BLSS    20$                     ;Bugcheck if not
                  53 8ED0   0030 585          POPL    R3                      ;Restore scratch
                        05   0033 586          RSB                             ; and return
                              0034 587
                              0034 588          .END
```

```
BADALLOC                        0000010B R      03
BUG$_BADPAGFILA                 ********   X    03
BUG$_BADPAGFILD                 ********   X    03
CRITMSG                         0000015C R      03
DYN$C_PFL                     = 00000023
EXE$GL_FLAGS                    ********   X    03
EXE$V_PGFLCRIT                  ********   X    03
EXE$V_PGFLFRAG                  ********   X    03
FRAGMSG                         0000010F R      03
IOC$BROADCAST                   ********   X    03
MMG$ALC_PGFLVBN                 00000000 RG     04
MMG$ALLOCPAGFIL1                00000038 RG     03
MMG$ALLOCPAGFIL2                000001AF RG     03
MMG$ALLOCSWPAREA                00000000 RG     03
MMG$DALCPAGFIL                  0000023A RG     03
MMG$DFALLOCPAGFIL               00000243 RG     03
MMG$GL_MAXPFIDX                 00000028 RG     02
MMG$GL_NULLPFL                  00000000 RG     02
MMG$GL_PAGSWPVC                 00000024 RG     02
MMG$GW_MINPFIDX                 0000002C RG     02
MPW$GW_MPWPFC                   ********   X    03
OPA$UCBO                        ********   X    03
PFL$B_ALLOCSIZ                = 00000022
PFL$B_FLAGS                   = 00000023
PFL$C_LENGTH                  = 00000024
PFL$L_BITMAP                  = 00000000
PFL$L_BITMAPSIZ               = 00000014
PFL$L_FREPAGCNT               = 00000018
PFL$L_MAXVBN                  = 0000001C
PFL$L_STARTBYTE               = 00000004
PFL$M_SWPFILFUL               = 00000004
PFL$V_INITED                  = 00000000
PFL$V_SWPFILFUL               = 00000002
PTE$M_PGFLVB                  = 003FFFFF
PTE$V_CHKPNT                  = 00000015
RSN$_SWPFILE                  = 0000000A
SCH$RAVAIL                      ********   X    03
SENDMSG                         00000222 R      03
SGN$GW_SWPFILCT                 0000002C RG     02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$220	0000002E (46.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$MMGCOD	000002DA (730.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
Y$LOWUSE	00000034 (52.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.05	00:00:02.44
Command processing	117	00:00:00.55	00:00:03.46
Pass 1	187	00:00:03.93	00:00:11.36
Symbol table sort	0	00:00:00.35	00:00:01.55
Pass 2	114	00:00:01.42	00:00:05.32
Symbol table output	5	00:00:00.05	00:00:00.04
Psect synopsis output	2	00:00:00.04	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	462	00:00:06.39	00:00:24.32

The working set limit was 1350 pages.
22714 bytes (45 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 267 non-local and 32 local symbols.
588 source lines were read in Pass 1, producing 19 object records in Pass 2.
12 pages of virtual memory were used to define 11 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	5
_255DUA28:[SYSLIB]STARLET.MLB;2	3
TOTALS (all libraries)	8

319 GETS were required to define 8 macros.

There were no errors, warnings or information messages.

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

0378 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PAGEFAULT
LIS

OSWPSCHED
LIS

PARAMETER
LIS

P1SYSVECT
LIS

PAGEFILE
LIS