



```

PPPPPPPP      AAAAAA      GGGGGGGG      EEEEEEEEEEE      FFFFFFFFFFF      IIIIIII      LL      EEEEEEEEEEE
PPPPPPPP      AAAAAA      GGGGGGGG      EEEEEEEEEEE      FFFFFFFFFFF      IIIIIII      LL      EEEEEEEEEEE
PP      PP      AA      AA      GG      EEEEEEEEEEE      FF      II      LL      EEEEEEEEEEE
PP      PP      AA      AA      GG      EEEEEEEEEEE      FF      II      LL      EEEEEEEEEEE
PP      PP      AA      AA      GG      EEEEEEEEEEE      FF      II      LL      EEEEEEEEEEE
PPPPPPPP      AA      AA      GG      EEEEEEEEEEE      FFFFFFFFFF      II      LL      EEEEEEEEEEE
PPPPPPPP      AA      AA      GG      EEEEEEEEEEE      FFFFFFFFFF      II      LL      EEEEEEEEEEE
PP      AAAAAAAAAA      GG      GGGGGG      EE      FF      II      LL      EEEEEEEEEEE
PP      AAAAAAAAAA      GG      GGGGGG      EE      FF      II      LL      EEEEEEEEEEE
PP      AA      AA      GG      GG      EE      FF      II      LL      EEEEEEEEEEE
PP      AA      AA      GG      GG      EE      FF      II      LL      EEEEEEEEEEE
PP      AA      AA      GG      GG      EE      FF      II      LL      EEEEEEEEEEE
PP      AA      AA      GGGGGG      EEEEEEEEEEE      FF      IIIIIII      LLLLLLLLLL      EEEEEEEEEEE
PP      AA      AA      GGGGGG      EEEEEEEEEEE      FF      IIIIIII      LLLLLLLLLL      EEEEEEEEEEE

```

```

LL      IIIIIII      SSSSSSSS
LL      IIIIIII      SSSSSSSS
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      IIIIIII      SSSSSSSS
LL      IIIIIII      SSSSSSSS

```

(1)	42
(2)	57
(3)	120
(5)	182
(6)	323
(10)	425
(12)	532

HISTORY ; DETAILED  
DECLARATIONS  
ALLOCSWPAREA - ALLOCATE A SWAP AREA IN A PAGE FILE  
ALLOCPAGFIL - ALLOCATE A PAGING FILE SPACE  
ALLOCPAGFIL - ALLOCATE A PAGING FILE SPACE  
DALCPAGFIL - DEALLOCATE PAGE IN PAGING FILE  
ALC\_PGFLVBN Allocate specific blocks in paging file

```

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 WMC0001 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$GL_NULLPFL::          ; NULL PFL SERVES AS PLACEHOLDER
00000000 0000 86          .LONG 0          ; BITMAP POINTER, 0 IF TABLE NOT IN USE
00000000 0004 87          .LONG 0          ; ADDRESS OF MPW_WRTCLUSTER SIZE AREA
00000000 0008 88          .WORD PFL$C_LENGTH
00000000 000A 89          .BYTE DYN$C_PFL
00000000 000B 90          .BYTE 0          ; PAGE FAULT CLUSTER
00000000 000C 91          .LONG 0          ; WINDOW POINTER, *** FILLED IN BY INIT
00000000 0010 92          .LONG 0          ; VBN, *** FILLED IN BY INIT
00000000 0014 93          .LONG 0          ; BITMAP SIZE
00000000 0018 94          .LONG 0          ; FREE PAGE COUNT IN THIS FILE
003FFFFFF 001C 95          .LONG PTE$M_PGFLVB ; PAGE FILE VBN MASK
00000000 0020 96          .LONG 0          ; ACCOUNT FOR EXTENDED LENGTH
0000 97 :
0000 98 : POINTER TO VECTOR OF PAGE/SWAP FILE CONTROL BLOCKS
0000 99 :
0000 100
00000000 0024 101 MMG$GL_PAGSWPVC::
00000000 0024 102          .LONG 0
0000 103
0000 104 :
0000 105 : MAXIMUM PAGE FILE INDEX CURRENTLY IN USE
0000 106 :
0000 107
00000000 0028 108 MMG$GL_MAXPFIDX::
00000000 0028 109          .LONG 0
0000 110
0000 111 MMG$GW_MINPFIDX::
0000 112 SGN$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 :--
```

PAC  
Syn  
BAC  
BUG  
BUG  
CRI  
DYN  
EXE  
EXE  
EXE  
FRA  
IOC  
MMC  
MMC  
MMC  
MMC  
MMC  
MMC  
MMC  
MP  
OPA  
PFL  
PFL  
PFL  
PFL  
PFL  
PFL  
PFL  
PFL  
PTE  
PTE  
RSA  
SCH  
SEN  
SGN  
PSE  
---  
SAE  
SSI  
SMI  
YSI

```

      30      BB 0000 163 mmg$allocswparea::
      55      D4 0000 164      pushr  #^m<r4,r5>          ; save work registers
      01      D0 0002 165      clr    r5              ; indicator for no paging files at all
      53      54 0024'DF44 D0 0004 166      movl  #1,r4              ; start scan at file index 1
      10 23 A3 00      D0 0007 167 10$:  movl  @w^mmg$gl_pagswpvc[r4],r3 ; get address of next page file block
ED 54 00000028'GF F3 0012 168      bbs   #pfl$y_initd,pfl$b_flags(r3),30$ ;branch if this one initd
      52 01      CE 001A 169 20$:  aobleq g^mmg$gl_maxpidx,r4,10$ ; loop through all page files
      50 55      D0 001D 170      mnegl #1,r2              ; assume unlimited growth size
      13      11 0020 171      movl  r5,r0              ; set flag indicating if we are booting
      55      D7 0022 172      brb   40$              ; use common exit
      OFF0 8F BB 0024 173 30$:  decl  r5              ; indicate valid paging file exists
      OE 10 0028 174      pushr  #^m<r4,r5,r6,r7,r8,r9,r10,r11> ;save volatile registers
      OFF0 8F BA 002A 175      bsbb  mmg$allocpagfil1 ; allocate new area in page file
      E2 13 002E 176      popr  #^m<r4,r5,r6,r7,r8,r9,r10,r11> ;restore volatile registers
      50 08 18 54      FO 0030 177      beql  20$              ; try next page file
      30      BA 0035 178      insv  r4,#24,#8,r0      ; save swap file index
      05      05 0037 179 40$:  popr  #^m<r4,r5>          ; restore registers
      05      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,
0038 228 : this allows some emergency 8 byte blocks to be allocated.
0038 229 :
0038 230 :--
```

```

232 MMGSALLOCPAGFIL1:
233      movl   pfl$l_bitmap(r3),r6      ;address of start of map
234      movl   pfl$l_bitmapsiz(r3),r7   ;number of bytes in map
235      addl3  r6,r7,r10                ;get end of map address
236      mnegl  #1,r11                    ;materialize a minus for use
20$:   237      pushr  #^m<r0,r1,r2,r3>    ;save the inputs, (r3 is now address)
238      ashl   #-3,r2,r8                ;make size into byte count
239      extzv  #24,#8,r0,r9            ;get the page file index
240      cmpl   @w^mmg$gl_pagswpc[r9],r3 ;is this in the same page file?
241      bneq   60$                       ;branch if not, try for simple allocate
242      extzv  #0,#24,r0,r0            ;get the input VBN
243      beql   60$                       ;branch if not holding current space
244      addl3  r0,r1,r9                ;get ending block
245      ashl   #-3,r9,r9                ;get byte offset of area after this one
246      ;r0+r1 always yield (multiple of 8)+1
247      ashl   #-3,r1,r4                ;current size in groups of 8
248      subl3  r4,r8,r5                ;number of additional needed blocks
249      bgtr   30$                       ;branch if this is an expansion
250      bsbw   mmg$deallocpagfil        ;free current holding if contraction
251      popr   #^m<r0,r1,r2,r3>        ;restore regs
252      clrq   r0                       ;indicate holding freed
253      brb    20$                       ;now do the allocation
254      ;
255      ; The end of map condition is handled by having a non-allocatable byte at
256      ; the end of the map. This allows the skpc to failure terminate.
257      ;
258      30$:   skpc   r11,r5,(r6)[r9]    ;find additional contiguous free space
259      bneq   60$                       ;branch if non-free blocks in area
260      movc5  #0,(r1),#0,r5,(r6)[r9]  ;mark these blocks allocated
261      ;
262      ; It is safe not to update STARTBYTE down this path since this is an allocate.
263      ; This is also probably desirable to lessen start of map searches.
264      ;
265      popr   #^m<r0,r1,r2,r3>        ;restore regs
266      ;note input VBN is output VBN
267      subl   r2,r1                    ;get additionally allocated blocks
268      ;(count is negative)
269      addl   r1,pfl$l_frepagecnt(r3)  ;update count of available pages
270      bicpsw #4                       ;indicate success
271      rsb    ;return VBN in R0, count in R2, z-bit=0
272      ;
273      ; allocation failure return
274      ;
275      40$:   popr   #^m<r0,r1,r2,r3>    ;restore regs
276      bisb   #pfl$m_swpcfilful,pfl$b_flags(r3) ;set flag indicating file full
277      bispsw #4                       ;indicate failure, no deallocation!
278      rsb    ;z-bit set
279      ;
280      ; new allocation
281      ;
282      60$:   movzwl r11,r5              ;set up for 65536 byte locate
283      cmpb   r2,pfl$b_allocsiz(r3)    ;is this standard request size?
284      blss   70$                       ;branch if not, search from start
285      movl   pfl$l_startbyte(r3),r1   ;set up to start from first known free
286      bneq   80$                       ;branch if we know where
287      70$:   movl   r6,r1              ;set up to scan map from start
288
232 0038 232 D0 0038
233 0038 233 D0 0038
234 003F 234 C1 003F
235 0043 235 CE 0043
236 0046 236 BB 0046
237 0048 237 78 0048
238 004D 238 EF 004D
239 0052 239 D1 0052
240 0058 240 12 0058
241 005A 241 EF 005A
242 005F 242 13 005F
243 0061 243 C1 0061
244 0065 244 78 0065
245 006A 245 006A
246 006A 246 78 006A
247 006F 247 C3 006F
248 0073 248 14 0073
249 0075 249 30 0075
250 0078 250 BA 0078
251 007A 251 7C 007A
252 007C 252 11 007C
253 007E 253 007E
254 007E 254 007E
255 007E 255 007E
256 007E 256 007E
257 007E 257 007E
258 007E 258 3B 007E
259 0083 259 12 0083
260 0085 260 2C 0085
261 008C 261 008C
262 008C 262 008C
263 008C 263 008C
264 008C 264 008C
265 008C 265 0F BA 008C
266 008E 266 008E
267 008E 267 51 52 C2 008E
268 0091 268 0091
269 0091 269 18 A3 51 C0 0091
270 0095 270 04 B9 0095
271 0097 271 05 0097
272 0098 272 0098
273 0098 273 0098
274 0098 274 0098
275 0098 275 23 A3 0F BA 0098
276 009A 276 04 88 009A
277 009E 277 04 B8 009E
278 00A0 278 05 00A0
279 00A1 279 00A1
280 00A1 280 00A1
281 00A1 281 00A1
282 00A1 282 22 A3 55 5B 3C 00A1
283 00A4 283 06 91 00A4
284 00A8 284 19 00A8
285 00AA 285 51 04 A3 D0 00AA
286 00AE 286 03 12 00AE
287 00B0 287 51 56 D0 00B0
288 00B3 288 00B3

```

```

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 cml 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
61 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$gl_pagswpc[r3],r3 ;get page file control block address
    014F 30 00F1 311 bsbw mmg$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 A3 04 12 00FA 314 bneq 100$ ;branch if not, don't affect memory
    18 A3 59 D0 00FC 315 movl r9,pfl$l_startbyte(r3) ;update memory for future reference
50 57 03 78 0104 316 100$: subl r2,pfl$l_frepagcnt(r3) ;update count of available pages
    50 D6 0108 317 ashl #3,r7,r0 ;multiply byte number*8 to get VBN
    05 010A 318 incl r0 ;VBN's need to be based at 1
    010B 319 rsb ;return, z-bit=0
    010B 320
    010B 321 BADALLOC:
    010B 322 BUG CHECK BADPAGFILE,FATAL ;BAD PAGE FILE ADDRESS ALLOCATED
    010F 323 .SBTTL ALLOCPAGFIL - ALLOCATE A PAGING FILE SPACE

```

```
010F 325 :++  
010F 326 : FUNCTIONAL DESCRIPTION:  
010F 327 :  
010F 328 :         THIS ROUTINE ALLOCATES THE FIRST CONTIGOUS SET OF BLOCKS FROM  
010F 329 :         THE SPECIFIED PAGE FILE.  
010F 330 :  
010F 331 : CALLING SEQUENCE:  
010F 332 :  
010F 333 :         BSBW    MMG$ALLOCPAGFIL2         ; must occur just after a call  
010F 334 :                                         ; to MMG$ALLOCPAGFIL  
010F 335 :  
010F 336 : INPUT PARAMETERS:  
010F 337 :  
010F 338 :         r3 = page file control block address  
010F 339 :         r6 = address of start of bitmap  
010F 340 :         r10= end address of bitmap  
010F 341 :         r11= 65536 (maximum size for a string instruction length)  
010F 342 :  
010F 343 : IMPLICIT INPUTS:  
010F 344 :  
010F 345 :         NONE  
010F 346 :  
010F 347 : OUTPUT PARAMETERS:  
010F 348 :  
010F 349 :         R0 = PAGE FILE VBN (GREATER THAN 0) IF SUCCESSFUL  
010F 350 :         R2 = NUMBER OF PAGES ALLOCATED  
010F 351 :  
010F 352 : IMPLICIT OUTPUTS:  
010F 353 :  
010F 354 :         NONE  
010F 355 :  
010F 356 : COMPLETION CODES:  
010F 357 :  
010F 358 :         Z-BIT SET IF FAILURE  
010F 359 :         Z-BIT 0 IF SUCCESS  
010F 360 :  
010F 361 : SIDE EFFECTS:  
010F 362 :  
010F 363 :         none  
010F 364 :--
```

2D 57 2D 4D 45 54 53 59 53 25 0A 0D	010F	366 fragmsg:	
61 50 20 2C 47 41 52 46 45 47 41 50	010F	367	.long 20\$-10\$
6C 64 61 62 20 65 6C 69 66 20 65 67	0113	368	.long 10\$
64 65 74 6E 65 6D 67 61 72 66 20 79	0117	369 10\$:	
6E 6F 63 20 6D 65 74 73 79 73 20 2C	0117	370	.ascii <13><10>-
	0123		
	012F		
	013B		
	0147		
	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		
	01AD	379	<13><10>
	01AF	380 40\$:	

```

MMG$ALLOCPAGFIL2::
55 5B 3C 01AF 382
51 56 D0 01AF 383      movzwl  r11,r5      ;set up for 65536 byte locate
                                01B2 384      movl    r6,r1      ;set up to scan map from start
57 5A 51 C3 01B5 385
66 13 01B9 386 10$:  subl3  r1,r10,r7      ;calc number of bytes remaining to scan
55 57 D1 01BB 387      beql    50$        ;branch if at end of map
                                18 01BE 388      cmpl   r7,r5      ;less than 65536 bytes to scan?
55 57 D0 01C0 389      bgeq   20$        ;branch if not
61 55 00 3B 01C3 390      movl   r7,r5      ;set scan amount to what's left
50 61 08 00 EA 01C9 391 20$:  skpc   #0,r5,(r1)  ;find any free blocks
52 50 01 C3 01CE 392      beql   10$        ;branch if no free clusters in area
FA 61 52 E4 01D4 393      ffs    #0,#8,(r1),r0 ;find the free block
18 A3 52 C2 01D8 394      subl3  #1,r0,r2    ;save start offset
51 50 01 A0 01DF 395 30$:  incl   r2         ;account for block
51 14 A3 51 56 C2 01DB 396      bbsc   r2,(r1),30$ ;loop through contiguous portion of map
51 14 A3 51 56 C2 01DB 397      subl   r0,r2      ;set r2 number of blocks allocated
51 14 A3 51 56 C2 01DB 398      subl   r2,pfl$l_frepagcnt(r3) ;update count of available pages
51 14 A3 51 56 C2 01DF 399      subl   r6,r1      ;get byte number of free blocks
51 14 A3 51 56 7E 01E2 400     movaq  1(r0)[r1],r0 ;form 8*byte number + bit number + 1
51 14 A3 51 56 D1 01E7 401     ashl   #1,pfl$l_bitmapsiz(r3),r1 ;find 1/4 point of VBN's in bitmap
51 14 A3 51 56 D1 01EC 402     cmpl   r0,r1      ;is this allocation past 1/4 point?
OB 0000'CF 00' 1F 01EF 403     blssu  50$        ;branch if not, no message needed yet
51 FF12 CF 07 BB 01F7 404     pushr  #^m<r0,r1,r2> ;save registers
51 FF12 CF 22 7D 01F9 405     movq   fragmsg,r1 ;set up message to output
51 FF12 CF 07 BA 0200 406     bsbb   sendmsg    ;output the message
54 51 51 C1 0202 407     popr   #^m<r0,r1,r2> ;restore registers
54 54 51 C0 0206 408     addl3  r1,r1,r4    ;find 3/4 mark in file
54 54 50 D1 0209 409 40$:  addl   r1,r4      ;now have 3/4 VBN
54 54 50 D1 0209 410     cmpl   r0,r4      ;is this allocation past 3/4 point
OD 0000'CF 00' 1F 020C 411     blssu  50$        ;branch if not
51 FF42 CF 07 BB 0214 412     bbsc   s^#exe$v_pgflcrit,w^exe$gl_flags,50$ ;branch if reported
51 FF42 CF 07 BB 0214 413     pushr  #^m<r0,r1,r2> ;save registers
51 FF42 CF 05 10 0216 414     movq   critmsg,r1 ;set up message to output
51 FF42 CF 07 BA 021B 415     bsbb   sendmsg    ;output the message
51 FF42 CF 07 BA 021D 416     popr   #^m<r0,r1,r2> ;restore registers
51 FF42 CF 04 B9 021F 417     bicpsw #4         ;indicate success
51 FF42 CF 05 05 0221 418     rsb    #4         ;return, z-bit=0 success, else failure
51 FF42 CF 05 05 0222 419     rsb    #4
51 FF42 CF 05 05 0222 420     rsb    #4
51 FF42 CF 05 05 0222 421     rsb    #4
51 FF42 CF 05 05 0222 422     rsb    #4
51 FF42 CF 05 05 0222 423     rsb    #4
55 0000'CF 9E 0222 421     sendmsg:
FDD6' 31 0227 422     movab  w^opa$ucb0,r5 ;set console terminal for broadcast
                                31 0227 423     brw    ioc$broadcast ;assume message will get to console

```

```
022A 425 .SBTTL DALCPAGFIL - DEALLOCATE PAGE IN PAGING FILE
022A 426
022A 427 :++
022A 428 : FUNCTIONAL DESCRIPTION:
022A 429 :
022A 430 : THIS ROUTINE DEALLOCATES A SPECIFIED PAGE IN THE SPECIFIED
022A 431 : PAGING FILE.
022A 432 :
022A 433 : CALLING SEQUENCE:
022A 434 :
022A 435 : BSBW MMG$DALCPAGFIL
022A 436 :
022A 437 : INPUT PARAMETERS:
022A 438 :
022A 439 : R0 = PAGE FILE VBN TO DEALLOCATE
022A 440 : R3 = PAGE FILE INDEX
022A 441 :
022A 442 : IMPLICIT INPUTS:
022A 443 : NONE
022A 444 :
022A 445 : OUTPUT PARAMETERS:
022A 446 : R0,R1,R2 DESTROYED
022A 447 : R3 = ADDRESS OF PAGE FILE CONTROL BLOCK
022A 448 :
022A 449 : IMPLICIT OUTPUTS:
022A 450 :
022A 451 : IF THE SPECIFIED PAGING FILE BECOMES NON-EMPTY, THE RESOURCE
022A 452 : AVAILABLE SIGNAL IS ISSUED FOR THE RSN$_PGFILE RESOURCE
022A 453 :
022A 454 : COMPLETION CODES:
022A 455 : NONE
022A 456 :
022A 457 : SIDE EFFECTS:
022A 458 : NONE
022A 459 :
022A 460 :--
```

```

022A 462 .ENABLE lsb
022A 463
022A 464 5$: ;check for checkpoint bit
03 08 50 15 E1 022A 465 bbc #pte$y_chkpnt,r0,10$ ;checkpoint bit set?
03 1C A3 15 E0 022E 466 bbs #pte$y_chkpnt,pfl$l_maxvbn(r3),10$ ;branch if not a small file
03 03 BA 0233 467 popr #^m<r0,r1> ;clean up
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$gl_pagswpc[r3],r3 ;get page file control block address
51 01 D0 0240 477 movl #1,r1 ;set count to 1
0243 478 ;fall through
0243 479 ; r0 = VBN of start block to return
0243 480 ; r1 = count
0243 481 ; r3 = address of page file control block
0243 482
0243 483 MMG$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 52 FD 8F 78 024D 488 decl r2 ;account for count in 0 origin
14 A3 52 C1 0254 489 ashl #-3,r2,r2 ;byte # in map
52 20 D0 0258 490 cmpl r2,pfl$l_bitmapsiz(r3) ;legal page file VBN?
51 52 D1 025A 491 bgequ 5$ ;branch if illegal
03 15 0260 492 movl #32,r2 ;max number single insv can set
52 51 D0 025D 493 30$: cmpl r2,r1 ;free more than 32?
00 00 B3 52 50 D0 0262 494 bleq 40$ ;branch if yes
00 B3 52 50 EC 0265 495 movl r1,r2 ;set max number to free
52 50 C9 12 026B 496 40$: cmpv r0,r2,@pfl$l_bitmap(r3),#0 ;temp check for safety
52 50 FF 8F F0 026D 497 bneq 10$ ;bugcheck if any of these bit set
18 A3 52 C0 0277 498 insv #-1,r0,r2,@pfl$l_bitmap(r3) ;set the bits
51 52 C0 027A 499 addl r2,r0 ;update to next VBN sequence
51 52 C2 027E 500 addl r2,pfl$l_frepagcnt(r3) ;count free pages
51 52 DA 12 0281 501 sub r2,r1 ;number of blocks to still free
50 50 FD 8F 78 0283 502 bneq 30$ ;loop through entire set
51 51 FD 8F 78 0285 503 popr #^m<r0,r1> ;get back VBN and free count
00 B340 51 51 FD 8F 78 028A 504 ashl #-3,r0,r0 ;set up to check for 8 block unit freed
51 51 FF 8F 3A 028D 505 addl #14,r1 ;round count for worst case crossing
04 A3 51 D1 0292 506 ashl #-3,r1,r1 ;number of bytes to check
3E 13 0299 507 locc #-1,r1,@pfl$l_bitmap(r3) ;any whole cluster become free?
04 A3 51 D1 029B 508 beql 60$ ;branch if not
38 1A 029F 509 cmpl r1,pfl$l_startbyte(r3) ;is freed cluster earlier in map?
50 71 92 02A1 510 bgtru 60$ ;branch if not, note bgtru not bgequ
52 51 FB 13 02A4 511 50$: mcomb -(r1),r0 ;find start byte of free area
51 51 22 A3 9A 02A6 512 beql 50$ ;loop
62 51 FF 8F 3B 02AA 513 addl3 #1,r1,r2 ;set start of area
04 A3 52 D0 02AE 514 movzbl pfl$b_alloccsiz(r3),r1 ;get current cluster size for this file
04 A3 52 D0 02B3 515 ashl #-3,r1,r1 ;get it in bytes rather than blocks
04 A3 52 D0 02B8 516 skpc #-1,r1,(r2) ;does this area qualify?
04 A3 52 D0 02BA 517 bneq 60$ ;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_mpwffc ;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$b_allocsiz(r3) ;try next higher size next time
OA 23 A3 02 E5 02CA 523 55$:    bbcc    #pfl$y_swpsful,pfl$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$_swpfile,r0 ;set up to return swap file available
                                FD29' 30 02D4 526      bsbw    sch$_avail ;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$LOWUSE ;This code can page
0000 567
51 00000024'FF41 D0 0000 568 MMG$ALC_PGFLVBN::
53 53 DD 0008 569 MOVL @L^MMG$GL_PAGSWPVC[R1],R1 ;Get base address from index
50 50 D7 000A 570 PUSHL R3 ;Get a scratch regtster
53 50 FD 8F 78 000C 571 DECL R0 ;Bit # is base 0
14 A1 53 D1 0011 572 10$: ASHL #-3,R0,R3 ;Byte # in bit map
05 1E 0015 573 CMPL R3,PFL$$_BITMAPSIZ(R1) ;Legal page file vbn?
04 00 B1 50 E4 0017 574 BGEQU 20$ ;Branch if illegal
001C 575 BBSC R0,@PFL$$_BITMAP(R1),30$ ;Free the page and branch
001C 576 20$:
0020 577 BUG_CHECK BADPAGFILD,FATAL ;Bad page file address specified
18 A1 D7 0020 578 30$:
50 50 D6 0023 579 DECL PFL$$_FREPAGECNT(R1) ;Count another free page
E4 52 F5 0025 580 INCL R0 ;Point to next VBN in file
52 52 D7 0028 581 SOBGTR R2,10$ ;Go back if not done yet
52 18 A1 D1 002A 582 DECL R2 ;Form a minus 1
EC 19 002E 583 CMPL PFL$$_FREPAGECNT(R1),R2 ;Insure that counts still consistent
53 8ED0 0030 584 BLSS 20$ ;Bugcheck if not
05 0033 585 POPL R3 ;Restore scratch
0034 586 RSB ; and return
0034 587
0034 588 .END

```

```

BADALLOC          0000010B R    03
BUGS_BADPAGFILA  ***** X    03
BUGS_BADPAGFILD  ***** X    03
CRITMSG          0000015C R    03
DYN$C_PFL        = 00000023
EXESGC_FLAGS     ***** X    03
EXESV_PGFLCRIT  ***** X    03
EXESV_PGFLFRAG  ***** X    03
FRAGMSG          0000010F R    03
IOCSBROADCAST    ***** X    03
MMGSALC_PGFLVBN  00000000 RG   04
MMGSALLOCPAGFIL1 00000038 RG   03
MMGSALLOCPAGFIL2 000001AF RG   03
MMGSALLOCSWPAREA 00000000 RG   03
MMGSDALCPAGFIL   0000023A RG   03
MMG$DEALLOCPAGFIL 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$GW_MPWPFC    ***** X    03
OPASUCBO         ***** X    03
PFL$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_CHKPNL    = 00000015
RSNS_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
\$AB\$\$	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
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	5
-\$255\$DUA28:[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)+EXECMLS/LIB

This table contains 240 individual technical diagrams and listings, arranged in a 12x20 grid. Each cell in the grid contains a small-scale version of a technical drawing or a list of parameters. The diagrams vary in complexity, showing different views of components or data structures. Some cells contain text labels for specific listings:

- Row 2, Column 5: PAGEFAULT LIS
- Row 4, Column 1: OSMPSCHEG LIS
- Row 6, Column 15: PARAMETER LIS
- Row 8, Column 2: PLSYSVECT LIS
- Row 12, Column 15: PAGEFILE LIS

The diagrams are rendered in a high-contrast, black-and-white style, typical of technical documentation from the late 1970s or early 1980s.