


```

SSSSSSSS YY YY SSSSSSSS UU UU P P P P P P P P D D D D D D D D S S S S S S S S F F F F F F F F C C C C C C C C
SSSSSSSS YY YY SSSSSSSS UU UU P P P P P P P P D D D D D D D D S S S S S S S S F F F F F F F F C C C C C C C C
SS SS YY YY SS SSSSSSSS UU UU PP PP PP DD DD DD SS SS F F F F F F F F CC CC
SS SS YY YY SS SSSSSSSS UU UU PP PP PP DD DD DD SS SS F F F F F F F F CC CC
SS SS YY YY SS SSSSSSSS UU UU PP PP PP DD DD DD SS SS F F F F F F F F CC CC
SSSSSSS YY YY SSSSSSSS UU UU P P P P P P P P D D D D D D D D S S S S S S S S F F F F F F F F C C C C C C C C
SSSSSSS YY YY SSSSSSSS UU UU P P P P P P P P D D D D D D D D S S S S S S S S F F F F F F F F C C C C C C C C
SS SS YY YY SS SSSSSSSS UU UU PP PP PP DD DD DD SS SS F F F F F F F F CC CC
SS SS YY YY SS SSSSSSSS UU UU PP PP PP DD DD DD SS SS F F F F F F F F CC CC
SSSSSSSS YY YY SSSSSSSS UUUUUUUUUU PP PP D D D D D D D D S S S S S S S S F F F F F F F F C C C C C C C C
SSSSSSSS YY YY SSSSSSSS UUUUUUUUUU PP PP D D D D D D D D S S S S S S S S F F F F F F F F C C C C C C C C

```

```

LL LL I I I I I I SSSSSSSS
LL LL I I I I I I SSSSSSSS
LL LL I I SS
LL LL I I SS
LL LL I I SS
LL LL I I SSSSSS
LL LL I I SSSSSS
LL LL I I SS
LL LL I I SS
LL LL I I SS
LLLLLLLLLLLL I I I I I I SSSSSSSS
LLLLLLLLLLLL I I I I I I SSSSSSSS

```

(1)	48
(2)	143
(3)	301
(4)	383
(6)	486
(8)	615
(10)	869

DECLARATIONS
UPDSEC - Update Section File
UPDSECPAG - Update Section for First Cluster of Pages
UPDSECAST - Update Section AST
UPDSECQWT - Update Section File for Single Page
WRTPGSBAK - Write Pages Back to Disk
PTEPFNMFY - Get PFN and Modify bit from FTE

```
0000 1 .TITLE SYSUPDSEC - Update Section File System Service
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: UPDATE SECTION SYSTEM SERVICE
0000 29 :
0000 30 : ABSTRACT:
0000 31 :
0000 32 : ENVIRONMENT:
0000 33 :
0000 34 : AUTHOR: PETER H. LIPMAN , CREATION DATE: 21-APR-78
0000 35 :
0000 36 : MODIFIED BY:
0000 37 :
0000 38 : V03-002 WMC001 Wayne Cardoza 02-Mar-1983
0000 39 : MMG$CRECOM2 has gone away, MMG$INADRINI returns status
0000 40 :
0000 41 : V03-001 SOP001 J. R. Sopka 27 August 1982
0000 42 : Add XIP_B MAXACMODE field to IRP extension used by $UPDSEC
0000 43 : and use it for page owner access mode instead of IRP$B_RMOD
0000 44 : which should contain the mode of the requestor.
0000 45 :
0000 46 :--
```

```

0000 48      .SBTTL  DECLARATIONS
0000 49      :
0000 50      : INCLUDE FILES:
0000 51      :
0000 52      $ACBDEF      ;AST control block definitions
0000 53      $CADEF      ;Conditional assembly definitions
0000 54      $DYNDEF     ;Dynamic data structure type codes
0000 55      $GSDDEF     ;Global section descriptor definitions
0000 56      $IRPDEF     ;I/O request packet definitions
0000 57      $IPLDEF     ;Processor priority levels
0000 58      $MMGDEF     ; Offsets from FP into scratch area
0000 59      $PCBDEF     ;Process control block definitions
0000 60      $PFNDEF     ;Page frame number data base definitions
0000 61      $PHDDEF     ;Process header definitions
0000 62      $PRDEF      ;Processor register definitions
0000 63      $PRIDEF     ;Priority increment class definitions
0000 64      $PSLDEF     ;Processor Status Long Word definitions
0000 65      $PTEDEF     ;Page table entry definitions
0000 66      $RSNDEF     ;Resource definitions
0000 67      $SECDEF     ;Section table entry definitions
0000 68      $SHBDEF     ;Shared memory control block definitions
0000 69      $SSDEF      ;System status code definitions
0000 70      $VADEF      ;Virtual address field definitions
0000 71      :
0000 72      : MACROS:
0000 73      :
0000 74      :
0000 75      :
0000 76      : EQUATED SYMBOLS:
0000 77      :
0000 78      : Offset from AP
0000 79      :
00000004 0000 80      INADR      = 4      ;Offset to input range
00000008 0000 81      RETADR     = 8      ;Offset to return range
0000000C 0000 82      ACMODE     = 12     ;Access Mode
00000010 0000 83      FLAGS      = 16     ;Flags parameter
00000014 0000 84      EFN        = 20     ;QI/O Event Flag
00000018 0000 85      IOSB       = 24     ;QI/O I/O Status Block Address
0000001C 0000 86      ASTADR     = 28     ;QI/O AST address
00000020 0000 87      ASTPRM     = 32     ;QI/O AST parameter
0000 88      :
0000 89      : Offsets into I/O packet while being used as scratch storage for clustering
0000 90      :
0000 91      $OFFSET 0, POSITIVE, <-
0000 92      SVAPTE, -      ;Master page table entry address
0000 93      PTEDAT, -     ;Process PIE data
0000 94      < 3>, -
0000 95      < IRP_RMOD, 1>, - ;Request mode
0000 96      MFYCNT, -     ;Cluster count at last modified page
0000 97      IRP_AST, -    ;Ast address
0000 98      IRP_ASTPRM, - ;Ast parameter
0000 99      CLUSTER, -   ;Maximum size of cluster to scan for
0000 100     COUNT, -    ;Number of pages scanned
0000 101     < EXCLWRT, 1>, - ;Exclusive write access flag
0000 102     < 1>, -
0000 103     < IRP_EFN, 1>, - ;Event flag
0000 104     < IRP_PRI, 1>, - ;Priority

```

```

0000 105      IRP_IOSB,-           ;I/O status block address
0000 106      INCT,-             ;+ or - 1 according to direction
0000 107      INC4,-            ;+ or - 4 according to direction
0000 108      BAK,-             ;Backing store address of first PTE
0000 109      <4>,-
0000 110      <IRP_IOST1,8>,-    ;I/O status return area
0000 111      PROCPTE,-         ;Process page table entry address
0000 112      <4>,-
0000 113      IRP_SEGVBN,-      ;Starting virtual address of scan
0000 114      <IRP_LENGTH,0>-   ;Total size of scratch area used
0000 115      >
0000          SVAPTE:
0004          PTEDAT:
000B          IRP_RMOD:
000C          MFYCNT:
0010          IRP_AST:
0014          IRP_ASTPRM:
0018          CLUSTER:
001C          COUNT:
0020          EXCLWRT:
0022          IRP_EFN:
0023          IRP_PRI:
0024          IRP_IOSB:
0028          INCT:
002C          INC4:
0030          BAK:
0038          IRP_IOST1:
0040          PROCPTE:
0048          IRP_SEGVBN:
004C          IRP_LENGTH:
0000 116
0000 117          ASSUME  IRP_LENGTH      LE  IRP$C_LENGTH
0000 118          ASSUME  IRP_RMOD       EQ  IRP$B_RMOD
0000 119          ASSUME  IRP_AST        EQ  IRP$L_AST
0000 120          ASSUME  IRP_ASTPRM     EQ  IRP$L_ASTPRM
0000 121          ASSUME  IRP_EFN       EQ  IRP$B_EFN
0000 122          ASSUME  IRP_PRI       EQ  IRP$B_PRI
0000 123          ASSUME  IRP_IOSB      EQ  IRP$L_IOSB
0000 124          ASSUME  IRP_IOST1     EQ  IRP$L_IOST1
0000 125          ASSUME  IRP_SEGVBN    EQ  IRP$L_SEGVBN
0000 126      :
0000 127      : Offsets off the end of the I/O request packet
0000 128      :
0000 129          $OFFSET  IRP$C_LENGTH,POSITIVE,<-
0000 130          XIP_L_SCANCNT,-      ;Count - 1 of pages remaining to scan
0000 131          XIP_L_DIREC,-        ;+ OR - 200 according to the direction
0000 132          XIP_L_STARTVA,-      ;Starting virtual address to scan
0000 133          <XIP_B_UPDFLG,1>,-    ;Section update flags
0000 134          <XIP_B_MAXACMODE,1>,- ;Maximized access mode for page ownership
0000 135          <2>,-                ;Spare
0000 136          <XIP_C_LENGTH,0> -    ;Length of extended I/O packet
0000 137          >
00C4          XIP_L_SCANCNT:
00C8          XIP_L_DIREC:
00CC          XIP_L_STARTVA:
00D0          XIP_B_UPDFLG:
00D1          XIP_B_MAXACMODE:

```

SYSUPDSEC
V04-000

- Update Section File System Service I 16
DECLARATIONS

16-SEP-1984 02:36:29 VAX/VMS Macro V04-00
5-SEP-1984 03:57:55 [SYS.SRC]SYSUPDSEC.MAR;1

Page 4
(1)

```
00D4      XIP_C_LENGTH:  
0000 138  ::  
0000 139  :: OWN STORAGE:  
0000 140  ::  
0000 141  .LIST  MEB
```

```

0000 173      .SBITL  UPDSEC - Update Section File
0000 144      :++
0000 145      : FUNCTIONAL DESCRIPTION:
0000 146      :
0000 147      : CALLING SEQUENCE:
0000 148      :
0000 149      :     CALLG  ARGLIST,G^SYSS$UPDSEC
0000 150      :
0000 151      :
0000 152      : INPUT PARAMETERS:
0000 153      :
0000 154      :     INADR(AP) = Address of 2 long words the 1st of which specifies
0000 155      :                 the starting virtual address, the 2nd specifies the ending
0000 156      :                 virtual address (inclusive) of the pages to operate on.
0000 157      :     RETADR(AP) = Address of a 2 longword array into which is returned
0000 158      :                 the starting and ending virtual addresses (inclusive)
0000 159      :                 of the pages operated on.
0000 160      :     ACMODE(AP) = The access mode (maximized with calling mode)
0000 161      :                 against which the page ownership is checked.
0000 162      :                 Only the owner of a page may update its section.
0000 163      :     FLAGS(AP)  = Update section control flags
0000 164      :     EFN(AP)    = Event flag number to set on write complete
0000 165      :     IOSB(AP)  = I/O status block address for reporting the
0000 166      :                 write completion and its status
0000 167      :                 First word contains the system status.
0000 168      :                 If error status is returned in the first word,
0000 169      :                 the first bit of the 2nd word (bit 16 of the first
0000 170      :                 long word) will be set if a write error occurred.
0000 171      :                 Other errors (e.g. page owner violation) are possible.
0000 172      :                 The second long word contains the first virtual
0000 173      :                 address not written.
0000 174      :     ASTADR(AP) = AST address for reporting write completion
0000 175      :     ASTPRM(AP) = AST parameter for identifying the AST
0000 176      :
0000 177      : IMPLICIT INPUTS:
0000 178      :
0000 179      :     NONE
0000 180      :
0000 181      : OUTPUT PARAMETERS:
0000 182      :
0000 183      :     R0 = System Status Code
0000 184      :
0000 185      : IMPLICIT OUTPUTS:
0000 186      :
0000 187      :     NONE
0000 188      :
0000 189      : COMPLETION CODE :
0000 190      :
0000 191      :     $$$_NORMAL           ;Successful Completion
0000 192      :     $$$_ACCVIO          ;Access Violation
0000 193      :     $$$_PAGOWNVIO       ;Page Owner Violation
0000 194      :     $$$_EXQUOTA         ;Quota exceeded for pending AST's
0000 195      :     $$$_IVSECFLG       ;Invalid flags set
0000 196      :
0000 197      : SIDE EFFECTS:
0000 198      :
0000 199      :     NONE

```



```
0000 200 :  
0000 201 :--  
0000 202 :  
0000 203 :*****  
0000 204 :  
0000 205 :***** THE FOLLOWING CODE MAY BE PAGED *****  
0000 206 :  
00000000 207 : .PSECT Y$EXEPAGED  
0000 208 :  
0000 209 :*****  
0000 210 :  
0000 211 :  
0000 212 INADRERR:  
04 0000 213 RET  
0001 214  
01FC 0001 215 .ENTRY EXE$UPDSEC, ^M<R2,R3,R4,R5,R6,R7,R8>  
0003 216  
5E 1C C2 0003 217 SUBL S^#-MMG$C_LENGTH,SP ;Reserve area indexed from FP  
58 54 D0 0006 218 MOVL R4,R8 ;Save PCB address  
FFF4' 30 0009 219 BSBW MMG$INADRINI ;Get input address range to R4,R5  
;Init return range to null  
F1 50 E9 000C 221 BLBC R0,INADRERR  
30 BB 000F 222 PUSHR #^M<R4,R5> ;Save input address range  
54 58 D0 0011 223 MOVL R8,R4 ;Restore PCB address  
53 14 AC 9A 0014 224 MOVZBL EFN(AP),R3 ;Get the event flag parameter  
00000000'EF 16 0018 225 JSB SCH$CLREF ;Clear the specified event flag  
56 18 AC D0 001E 226 MOVL IOSB(AP),R6 ;Get I/O status block address  
08 13 0022 227 BEQL 20$ ;Branch if none specified  
0024 228 IFNOWRT #8,(R6),70$ ;Make sure caller could write it  
66 08 00 0D 0024 229 PROBEW #0,#8,(R6)  
7F 13 0028 230 BEQL 70$  
57 10 AC D0 002C 231 CLRQ (R6) ;and initialize it  
01 57 D1 0030 232 MOVL FLAGS(AP),R7 ;Get FLAGS parameter  
00000004'8F D0 0035 233 Cmpl R7,#1 ;Make sure no garbage bits are set  
00000000'EF 16 003C 234 BGTRU 60$ ;Branch if invalid section flags  
51 00000004'8F D0 0035 233 MOVL #XIP_C_LENGTH,R1 ;Size of packet to allocate  
00000000'EF 16 003C 234 JSB EXE$ALLOCBUF ;Allocate, wait if necessary  
67 50 E9 0042 235 BLBC R0,80$ ;Packet type is corrected by WRTPGSBAK  
0045 236 ;Branch if failed to alloc  
0045 237 ;and resource wait disabled  
0045 238 :  
0045 239 : IPL = ASTDEL, I/O request packet allocated  
0045 240 :  
58 52 D0 0045 241 MOVL R2,R8 ;Packet address to stable register  
52 3E A4 9E 0048 242 MOVAB PCB$W_DIOCNT(R4),R2 ;Check for Direct I/O quota  
00000000'EF 16 004C 243 JSB EXE$SNGLEQUOTA ;and wait if none available  
5B 50 E9 0052 244 BLBC R0,120$ ;Branch if exceeded quota  
0055 245 ;and resource wait is disabled  
50 50 02 16 DC 0055 246 MOVPSL R0 ;Get mode of the requestor  
FC AD 90 005C 247 EXTZV #PSL$V PRVMOD,#PSL$S PRVMOD,R0,R0  
00D1 C8 005F 248 MOV B^MMG$C_MAXACMODE(FPT,- ;Get maximized access mode  
10 A8 1C AC 7D 0062 250 XIP B MAXACMODE(R8) ; for page ownership checking  
10 A8 10 A8 D5 0067 251 MOVQ ASTADR(AP),IRPSL_AST(R8) ;Set AST address and parameter  
0C 13 006A 252 TSTL IRPSL_AST(R8) ;AST requested?  
38 A4 B5 006C 253 BEQL 40$ ;Branch if not  
3F 15 006F 254 TSTW P^9$W_ASTCNT(R4) ;Yes, quota exceeded?  
BLEQ 120$ ;Branch if yes, don't wait
```

```

50 38 A4 B7 0071 255      DECW   PCBSW_ASTCNT(R4)      ;Charge for the AST
22 40 3F 88 0074 256      BISB   #ACBSM_QUOTA,R0      ;And note that it is charged
08 A8 50 90 0078 257 40$: MOVVB   R0,IRPSB_RMOD(R8)      ;Set requesting mode and AST flag
24 A8 14 AC 90 007C 258   MOVVB   EFN(AP),IRPSB_EFN(R8) ;Set event flag number
00D0 C8 56 D0 0081 259   MOVL   R6,IRPSL_IOSB(R8)    ;Set I/O status block address
56 E9 AF 9E 0085 260   MOVVB   R7,XIP_B_UPD'R'G(R8) ;Set section update flags
          OC BA 008E 261   MOVAB   B^MMGSOPDSECPAG,R6  ;Address of per page subroutine
          FF6D 30 0090 262   POPR   #^M<R2,R3>          ;Recover saved input address range
          50 DD 0093 263   BSBW   MMGS_CREDEL         ;Common address range loop
          FF68 30 0095 264   PUSHL  R0                  ;Save status
02 50 E9 0098 265   BLBC   R0,45$              ;Use this bad status rather than CREDEL
          50 BA 009B 266   POPR   R0
          58 D5 009D 267   TSTL   R8                  ;I/O packet to be released?
          12 12 009F 268 45$: BNEQ   130$              ;Branch if yes
          04 00A1 269 50$: RET                    ;Write was queued successfully
          00A2 270
50 016C 8F 3C 00A2 271 60$: MOVZWL  #SS$_IVSECFLG,R0      ;Invalid section flags parameter
          03 11 00A7 272   BRB     80$
          50 0C 3C 00A9 273 70$: MOVZWL  #SS$_ACCVIO,R0      ;Access violation
          50 DD 00AC 274 80$: PUSHL  R0                  ;Save the status code
          16 11 00AE 275   BRB     140$
          00B0 276
          00B0 277 ; Release the I/O request packet, it was never used
          00B0 278
          50 1C 3C 00B0 279 120$: MOVZWL  #SS$_EXQUOTA,R0      ;Exceeded quota
03 08 A8 50 DD 00B3 280 130$: PUSHL  R0                  ;Save status
          38 A4 B6 00B5 281   BBCC   #ACBSV_QUOTA,IRPSB_RMOD(R8),135$ ;If charged for AST
          50 58 D0 00BA 282   INCW   PCBSW_ASTCNT(R4)      ;then give back the quota
00000000 EF 16 D0 00BD 283 135$: MOVL   R8,R0              ;Get I/O packet address to release
          00C0 284   JSB     EXES$DEANONPAGED      ;Release the I/O request packet
          00C6 285
          00C6 286 ;
          00C6 287 ; Set the event flag so that the caller may wait for it despite the return
          00C6 288 ; information showing that nothing was queued.
          00C6 289
          53 14 AC 9A 00C6 290 140$: MOVZBL  EFN(AP),R3          ;Get the event flag number
          51 60 A4 D0 00CA 291   MOVL   PCBSL_PID(R4),R1      ;and the process ID
          52 01 9A 00CE 292   MOVZBL #PRIS_IOCOM,R2      ;and the correct priority increment
00000000 EF 16 00D1 293   JSB     SCH$POSTEF         ;Post the event flag, write complete
          01 BA 00D7 294   POPR   #^M<R0>          ;Restore saved status
          51 18 AC D0 00D9 295   MOVL   IOSB(AP),R1        ;I/O status requested?
          09 13 00DD 296   BEQL   150$              ;Branch if not
          00DF 297   IFNOWRT #8,(R1),150$    ;Branch if IOSB not writable
61 08 00 0D 00DF 298   PROBEW #0,#8,(R1)
          03 13 00E3 298   BEQL   150$
          61 50 D0 00E5 298   MOVL   R0,(R1)          ;Return the error status
          04 00E8 299 150$: RET                    ;and return

```

```
00E9 301 .SBTTL UPDSECPAG - Update Section for First Cluster of Pages
00E9 302 :
00E9 303 : *****
00E9 304 :
00E9 305 : ***** THE FOLLOWING CODE MAY BE PAGED *****
00E9 306 :
0000 00E9 307 .PSECT YSEXEPAGED
00E9 308 :
00E9 309 : *****
00E9 310 :
00E9 311 : ++
00E9 312 : FUNCTIONAL DESCRIPTION:
00E9 313 :
00E9 314 :
00E9 315 : CALLING SEQUENCE:
00E9 316 :
00E9 317 : BSBW MMG$UPDSECPAG
00E9 318 :
00E9 319 :
00E9 320 : INPUT PARAMETERS:
00E9 321 :
00E9 322 : R0 = Access Mode for page ownership check
00E9 323 : R2 = Virtual Address
00E9 324 : R4 = Current PCB address
00E9 325 : R5 = Process Header Address - P1 or System Space
00E9 326 : R6 = Count - 1 of pages to be processed including this one
00E9 327 : R7 = +^X200 if going forward in the address space
00E9 328 : = -^X200 if going backwards in the address space
00E9 329 : R8 = Address of an extended length I/O request packet
00E9 330 : IRP$W_SIZE = size of extended IRP (XIP_C_LENGTH)
00E9 331 : type filled in by WRTPGSB$K
00E9 332 : IRP$L_ASTADR = AST address if desired
00E9 333 : IRP$L_ASTPRM = AST parameter
00E9 334 : IRP$B_RMOD = Requesting mode
00E9 335 : ACBSV_QUOTA set if AST desired
00E9 336 : IRP$B_EFN = Event flag number
00E9 337 : XIP_L_DIREC = + OR - ^X200 according to direction of scan
00E9 338 : XIP_B_UPDFLG = Update section flags
00E9 339 :
00E9 340 : IPL = ASTDEL
00E9 341 :
00E9 342 : IMPLICIT INPUTS:
00E9 343 : NONE
00E9 344 :
00E9 345 : OUTPUT PARAMETERS:
00E9 346 :
00E9 347 : R0 = Status Code
00E9 348 : R2 Preserved
00E9 349 :
00E9 350 : IMPLICIT OUTPUTS:
00E9 351 : NONE
00E9 352 :
00E9 353 : COMPLETION CODES:
00E9 354 :
00E9 355 : SSS_NORMAL ;Successful Completion
00E9 356 : SSS_PAGOWNVIO ;Page Owner Violation
00E9 357 : SSS_LENvio ;Length Violation
```

```

00E9 358 : SSS_ACCVIO ;Access Violation
00E9 359 :
00E9 360 : SIDE EFFECTS:
00E9 361 :
00E9 362 : NONE
00E9 363 :
00E9 364 : --
00E9 365 :
00E9 366 MMG$UPDSECPAG:
00CR C8 57 D0 00E9 367 MOVL R7,XIP_L DIREC(R8) ;Save direction of scan
00000000'EF 16 00EE 368 JSB MMG$UPDSECQWT ;Find and queue the next cluster
51 D5 00F4 369 TSTL R1 ;Anything queued for writing?
OD 12 00F6 370 BNEQ 20$ ;Branch if yes
F4 AD D4 00F8 371 CLRL B^MMG$L_SAVRETADR(FP) ;Return a null range
15 50 E9 00FB 372 BLBC R0,60$ ;Branch if error status
50 0659 8F 3C 00FE 373 MOVZWL #SS$_NOTMODIFIED,R0 ;Otherwise return alternate success code
OE 11 0103 374 BRB 60$
58 D4 0105 375 20$: CLRL R8 ;Note I/O packet in use
EC AD 52 D0 0107 376 MOVL R2,B^MMG$L_SVSTARTVA(FP) ;Return first address queued
51 57 C4 010B 377 DECL R1 ;Page count - 1
52 51 C0 0110 378 MULL R7,R1 ;Byte count
56 D4 0113 379 ADDL R1,R2 ;Address of last page queued
05 0115 380 60$: CLRL R6 ;Force end of range
381 RSB ;and return
  
```

```

0116 383 .SBTTL UPDSECAST - Update Section AST
0116 384 :++
0116 385 : FUNCTIONAL DESCRIPTION:
0116 386 :
0116 387 : This is a special kernel AST routine invoked by IOPOST at the
0116 388 : completion of a PAGIO write request with an extended I/O packet.
0116 389 : It's job is to find the next cluster of modified pages to write
0116 390 : and either queue the request or post the I/O completion.
0116 391 :
0116 392 : CALLING SEQUENCE:
0116 393 :
0116 394 : BSBW MMG$UPDSECAST
0116 395 :
0116 396 :
0116 397 : INPUT PARAMETERS:
0116 398 :
0116 399 : R4 = Current PCB address
0116 400 : R5 = Address of an extended length I/O request packet
0116 401 : IRP$W_SIZE = size of extended IRP (XIP_C_LENGTH)
0116 402 : IRP$B_TYPE = DYN$C IRP
0116 403 : IRP$L_ASTADR = AST address if desired
0116 404 : IRP$L_ASTPRM = AST parameter
0116 405 : IRP$B_RMOD = Requesting mode
0116 406 : ACBSV_QUOTA set if AST desired
0116 407 : IRP$B_EFN = Event flag number
0116 408 : XIP_L_SCANCNT = Count - 1 of pages left to scan
0116 409 : before this transfer completed
0116 410 : XIP_L_DIREC = + OR - ^X200 according to direction of scan
0116 411 : XIP_L_STARTVA = First VA used for this transfer
0116 412 : XIP_B_UPDFLG = Update section flags
0116 413 : XIP_B_MAXACMODE = Maximized access mode for page ownership
0116 414 : IRP$L_IOST1 = Status of previous write (0:15)
0116 415 : = Number of bytes successfully written (16:31)
0116 416 :
0116 417 : IPL = ASTDEL
0116 418 :
0116 419 : IMPLICIT INPUTS:
0116 420 : NONE
0116 421 :
0116 422 : OUTPUT PARAMETERS:
0116 423 :
0116 424 :
0116 425 : IMPLICIT OUTPUTS:
0116 426 : NONE
0116 427 :
0116 428 : COMPLETION CODES:
0116 429 :
0116 430 :
0116 431 : SIDE EFFECTS:
0116 432 :
0116 433 : NONE
0116 434 :
0116 435 :--

```



```

0189 486          .SBTTL  UPDSECQWT - Update Section File for Single Page
0189 487
0189 488      :++
0189 489      : FUNCTIONAL DESCRIPTION:
0189 490      :
0189 491      :
0189 492      : CALLING SEQUENCE:
0189 493      :
0189 494      :         BSBW  MMGSUPDSECQWT
0189 495      :
0189 496      :
0189 497      : INPUT PARAMETERS:
0189 498      :
0189 499      :         R2 = Virtual Address
0189 500      :         R4 = Current PCB address
0189 501      :         R5 = Process Header Address - P1 or System Space
0189 502      :         R6 = Count - 1 of pages to be processed including this one
0189 503      :         R7 =  +^X200 if going forward in the address space
0189 504      :             = -^X200 if going backwards in the address space
0189 505      :         R8 = Address of an extended length I/O request packet
0189 506      :             IRPSW_SIZE      = size of extended IRP (XIP_C_LENGTH)
0189 507      :                 type filled in by WRTPGSBAK
0189 508      :             IRPSL_ASTADR     = AST address if desired
0189 509      :             IRPSL_ASTPRM     = AST parameter
0189 510      :             IRPSB_RMOD       = Requesting mode
0189 511      :             ACBSV_QUOTA      = ACBSV_QUOTA set if AST desired
0189 512      :             IRPSB_EFN        = Event flag number
0189 513      :             XIP_L_DIRECT      = + OR - ^X200 according to direction of scan
0189 514      :             XIP_B_UPDFLG      = Update section flags
0189 515      :             XIP_B_MAXACMODE   = Maximized access mode for page ownership
0189 516      :
0189 517      :         IPL = ASTDEL
0189 518      :
0189 519      : IMPLICIT INPUTS:
0189 520      :         NONE
0189 521      :
0189 522      : OUTPUT PARAMETERS:
0189 523      :
0189 524      :         If write has been queued, then
0189 525      :
0189 526      :         R0 = #SS$_NORMAL
0189 527      :         R1 = number of pages queued for writing
0189 528      :         R2 = virtual address of first page (scan order) queued
0189 529      :         R6 = count - 1 of pages remaining to scan starting with VA in R2;
0189 530      :
0189 531      :         Extended portion of I/O request packet updated if write queued
0189 532      :             XIP_L_STARTVA      = starting virtual address of request just queued
0189 533      :             XIP_L_SCANCNT      = count - 1 of pages remaining to scan
0189 534      :                 starting with the first page just queued
0189 535      :
0189 536      :         If write has not been queued, then
0189 537      :
0189 538      :         R0 = system status code
0189 539      :         R1 = 0
0189 540      :         R2 = last virtual address scanned
0189 541      :             in the case of an error, this is the address that caused it
0189 542      :             if ran off the end of range, this is the last VA in the range
  
```

```

0189 543 : R6 = count - 1 of pages remaining to scan starting with VA in R2
0189 544 : = 0 if at end of range and no more to do
0189 545 :
0189 546 : IMPLICIT OUTPUTS:
0189 547 :
0189 548 : NONE
0189 549 :
0189 550 : COMPLETION CODES:
0189 551 :
0189 552 : SSS_NORMAL ;Successful Completion
0189 553 : SSS_PAGOWNVIO ;Page Owner Violation
0189 554 : SSS_LENvio ;Length Violation
0189 555 : SSS_ACCVIO ;Access Violation
0189 556 :
0189 557 : SIDE EFFECTS:
0189 558 :
0189 559 : NONE
0189 560 :
0189 561 : --
0189 562 :
0189 563 : *****
0189 564 : ***** THE FOLLOWING CODE MUST BE RESIDENT *****
0189 565 :
00000000 566 :
0000 567 : .PSECT $MMGCOD
0000 568 :
0000 569 : *****
0000 570 :

```

SYS
VAX

Pha

Ini
Com
Pas
Sym
Pas
Sym
Pse
Cro
Ass

The
947
The
105
36

Mac

_S2
-S2
TOT

159

The
MAC


```

51 53 57 6C B443 51 D4 0000 572 MMGSUPDSECQWT:
      FFFB' 30 0000 573          CLRL R1          ;Initialize indicator to no pages queued
      64 50 E9 0005 574          BSBW MMG$PTEINDX ;Get index to page table entry
      7E 12 DB 0008 575          BLBC R0,100$    ;Branch if length violation
      12 08 DA 000B 576          DSBINT #IPL$_SYNCH ;Push current IPL
      000E 577          MTPR S^#PR$_IPL,-(SP) ;and raise to SYNCH
      000E 578          MTPR #IPL$_SYNCH,S^#PR$_IPL ;Form system virtual address of PTE
51 53 57 6C B443 DE 000E 578          MOVAL @PCBSL_PHD(R4)[R3],R3 ;+ OR - 4 for adding to SVAPTE
      F9 8F 78 0013 579          ASHL #7,R7,R1
      0E BB 0018 580 10$: PUSHR #^M<R1,R2,R3>
      52 D4 001A 581          CLRL R2          ;PTEPFNMFY should return section/GPTX
50 00D1 C8 9A 001C 582          MOVZBL XIP_B_MAXACMODE(R8),R0 ;Access mode to check against page owner
51 00D0 C8 9A 0021 583          MOVZBL XIP_B_UPDFLG(R8),R1 ;Exclusive writer indication
      0254 30 0026 584          BSBW MMG$PTEPFNMFY ;Get PFN and modify bit for this PTE
      06 51 E9 0029 585          BLBC R1,20$    ;Branch if page not a candidate for write
      51 95 002C 586          TSTB R1          ;Could be written, is it modified?
      21 19 002E 587          BLSS 70$      ;Branch if yes, go write a cluster
      05 11 0030 588          BRB 30$      ;No, try the next page if any
50 51 D0 0032 589 20$: MOVL R1,R0      ;Error, or just not a candidate?
      14 12 0035 590          BNEQ 60$     ;Branch if error
      0E BA 0037 591 30$: POPR #^M<R1,R2,R3> ;R3=SVAPTE, R2=VA, R1=+ or - 4
      56 D5 0039 592          TSTL R6          ;Check for end of loop
      09 13 003B 593          BEQL 40$     ;Avoid modifying VA and Count
52 57 C0 003D 594          ADDL R7,R2    ;Next virtual address
53 51 C0 0040 595          ADDL R1,R3    ;and next PTE address
      D2 56 F4 0043 596          SOBGEQ R6,10$ ;Try the next page
50 01 3C 0046 597 40$: MOVZWL #$$$_NORMAL,R0 ;End of range, no more to do
      02 11 0049 598          BRB 65$
      0E BA 004B 599 60$: POPR #^M<R1,R2,R3>
      51 D4 004D 600 65$: CLRL R1          ;No pages queued for writing
      18 11 004F 601          BRB 80$
      0051 602          ;
      0051 603          ; Found a page to start the cluster, queue a cluster of pages
      0051 604          ;
      02 BA 0051 605 70$: POPR #^M<R1> ;Clean off + or - 4
50 00CC C8 6E D0 0053 606          MOVL (SP),XIP_L_STARTVA(R8) ;Save starting VA for UPDSECAST
      48 A8 6E D0 0058 607          MOVL (SP),IRP$L_SEGVBN(R8) ;and for WRTPGSBAK
50 00C4 C8 56 D0 005C 608          MOVL R6,XIP_L_SCANCNT(R8) ;and remaining count for this write
      51 58 D0 0061 609          MOVL R8,R1    ;I/O request packet (extended)
      0006 30 0064 610          BSBW MMG$WRTPGSBAK ;Queue a cluster for write back
      0C BA 0067 611          POPR #^M<R2,R3> ;Restore saved VA, clean off SVAPTE
      0069 612 80$: ENBINT ;Back to called IPL
      12 8E DA 0069 613 100$: MTPR (SP)+,S^#PR$_IPL
      05 05 006C          RSB
  
```

```

006D 615      .SBTTL WRTPGSBAK - Write Pages Back to Disk
006D 616      :++
006D 617      : FUNCTIONAL DESCRIPTION:
006D 618      :
006D 619      :
006D 620      : CALLING SEQUENCE:
006D 621      :
006D 622      :     BSBW     MMGS$WRTPGSBAK
006D 623      :
006D 624      :
006D 625      : INPUT PARAMETERS:
006D 626      :
006D 627      :     R0 = Page Frame Number of starting page
006D 628      :     R1 = Address of an I/O request packet
006D 629      :           IRPSW_SIZE      = XIP_C_LENGTH if called by UPDSEC
006D 630      :           = IRPSC_LENGTH if called by DELPAG
006D 631      :           IRPSB_TYPE       = type filled in by WRTPGSBAK
006D 632      :           IRPSL_ASTADR     = AST address if desired
006D 633      :           IRPSL_ASTPRM    = AST parameter
006D 634      :           IRPSB_RMOD      = Requesting mode
006D 635      :           = ACBSV_QUOTA set if AST desired
006D 636      :           IRPSB_EFN       = Event flag number
006D 637      :           IRPSL_SEGVBN    = Starting virtual address of scan
006D 638      :           XIP_B_UPDFLG    = Update section flags (if extended packet)
006D 639      :           XIP_B_MAXACMODE = Maximized access mode for page ownership
006D 640      :     R2 = Section backing store address (PFNSAL_BAK[R0])
006D 641      :           if process section page or shared memory global page
006D 642      :           = Global page table index if global page
006D 643      :     R3 = System virtual address of process page table entry for first page
006D 644      :     R4 = PCB address
006D 645      :     R5 = Process header address - P1 or System Space
006D 646      :     R6 = Count - 1 of pages remaining to be processed including this one
006D 647      :     R7 = +^X200 if going forward in address space
006D 648      :           = -^X200 if going backwards in address space
006D 649      :     IPL = SYNCH
006D 650      :
006D 651      : IMPLICIT INPUTS:
006D 652      :
006D 653      :     NONE
006D 654      :
006D 655      : OUTPUT PARAMETERS:
006D 656      :
006D 657      :     R0 = #SS$_NORMAL
006D 658      :     R1 = Number of pages queued for writing
006D 659      :     R2,R3 Scratched
006D 660      :
006D 661      : IMPLICIT OUTPUTS:
006D 662      :     NONE
006D 663      :
006D 664      : COMPLETION CODES:
006D 665      :
006D 666      :
006D 667      : SIDE EFFECTS:
006D 668      :
006D 669      :
006D 670      :--
  
```

```

006D 672 :
006D 673 : *****
006D 674 :
006D 675 : ***** THE FOLLOWING CODE MUST BE RESIDENT *****
006D 676 :
0000006D 677 : .PSECT $MMGCOD
006D 678 :
006D 679 : *****
006D 680 :
006D 681 MMG$WRTPGSBAK::
20 BB 006D 682 PUSHR #*M<R4,R5> ;Preserve R4 and R5 across call
006F 683 :
006F 684 : Initialize I/O packet for cluster scan
006F 685 :
2C A1 57 F9 8F 78 006F 686 ASHL #-7,R7,INC4(R1) ;+ or - 4 according to direction
28 A1 57 F7 8F 78 0075 687 ASHL #-9,R7,INC1(R1) ;+ or - 1 according to direction
006F 688 MOVL R1,R7 ;Packet address in stable register
00A A7 0A 90 007E 689 MOVNB #DYN$C IRP,IRP$B_TYPE(R7) ;Set packet type, size already set
00000000'EF 50 D1 0082 690 CMLP R0,MMG$GL_MAXPFN ;Is page in shared memory?
30 A7 0000'DF40 32 1A 0089 691 BGTRU 50$ ;Br if page is in shared memory gbl sec.
008B 692 MOVL @W^PFNSAL_BAK[R0],BAK(R7) ;Actual section backing store
67 0000'DF40 0092 693 ;address even if global page
40 A7 53 D0 0092 694 MOVL @W^PFNSAL_PTE[R0],SVAPTE(R7) ;Master PTE address even if global
04 A7 52 D0 0098 695 30$: MOVL R3,PROCPTE(R7) ;Keep process pte address
23 A7 2F A4 90 009C 696 MOVL R2,PTEDAT(R7) ;Save section adr/GPTX
00A0 697 MOVNB PCBSB_PRI8(R4),IRP$B_PRI(R7) ;Set transfer priority
00A5 698 :
00A5 699 : Calculate largest cluster size as the minimum of the default cluster
00A5 700 : size and the number of pages left to operate on.
00A5 701 :
51 0000'CF 3C 00A5 702 MOVZWL W^MPW$GW_MPWPFC,R1 ;Default cluster size
51 51 56 D1 00AA 703 CMLP R6,R1 ;If count-1 is smaller
01 04 18 00AD 704 BGEQ 40$
18 A7 01 A6 DE 00AF 705 MOVAL 1(R6),R1 ;then use count as max cluster size
1C A7 01 D0 00B3 706 40$: MOVL R1,CLUSTER(R7) ;Set maximum cluster size
01 D0 00B7 707 MOVL #1,COUNT(R7) ;Count the first page in the cluster
3C 11 00BB 708 BRB 80$ ;and loop zero or more times
00BD 709 :
00BD 710 : Shared Memory global section pages have no PFN data base.
00BD 711 :
30 A7 52 D0 00BD 712 50$: MOVL R2,BAK(R7) ;Use section table index
67 53 D0 00C1 713 MOVL R3,SVAPTE(R7) ;Process PTE is the Master PTE
02 11 00C4 714 BRB 30$ ;Join common code
00C6 715 :
00C6 716 : The loop that follows gathers pages to cluster write from the same section
00C6 717 : The pages must (of course) be resident, but not all of them must actually
00C6 718 : be modified. For process section pages, cluster from the first page
00C6 719 : (guaranteed modified) through the last modified page up to the cluster size.
00C6 720 : For global pages, cluster write all the pages in the global writable
00C6 721 : section. The state of the modified bit is indeterminate since it is
00C6 722 : maintained in the individual PTE's of the processes which map the section
00C6 723 :
53 2C A7 C0 00C6 724 60$: ADDL INC4(R7),R3 ;Next PTE address
04 52 16 E0 00CA 725 BBS #PTE$V_TYPO,R2,70$ ;If global page (not in sh mem)
52 28 A7 C0 00CE 726 ADDL INC1(R7),R2 ;then next GPTX as well
50 0B A7 02 00 00D2 727 70$: EXTZV #0,#2,IRP$B_RMOD(R7),R0 ;Requesting mode
51 D4 00D8 728 CLRL R1 ;Assume no update section flags

```

```

00D4 8F 08 A7 B1 00DA 729 CMPW IRPSW_SIZE(R7),#XIP_C_LENGTH ;If extended I/O packet
      19 00E0 730 BLSS 75$ ;Then
51 00D0 C7 90 00E2 731 MOVB XIP_B_UPDFLG(R7),R1 ; Use the save update section flags
50 00D1 C7 9A 00E7 732 MOVZBL XIP_B_MAXACMODE(R7),R0 ; Use maximized mode not requesting mode
      018E 30 00EC 733 75$: BSBW MMG$PTEPFNMFY ;Get PFN and modify bit if resident
      10 51 E9 00EF 734 BLBC R1,120$ ;Branch if not resident
      1C A7 D6 00F2 735 INCL COUNT(R7) ;Found another resident page
      51 95 00F5 736 TSTB R1 ;See if it was modified
      05 18 00F7 737 BGEQ 100$ ;Branch if it was not
OC A7 1C A7 D0 00F9 738 80$: MOVL COUNT(R7),MFYCNT(R7) ;then update last modified page seen
      C4 18 A7 F5 00FE 739 100$: SOBGTR CLUSTER(R7),60$ ;Try the next page too
      0102 740 ;
      0102 741 ; Now lock all the pages in the cluster just found
      0102 742 ;
51 OC 53 67 D0 0102 743 120$: MOVL SVAPTE(R7),R3 ;Get starting Master PTE
      51 A7 01 C3 0105 744 SUBL3 #1,MFYCNT(R7),R1 ;Count - 1 of pages in cluster
      51 2C A7 C4 010A 745 MULL INC4(R7),R1 ;* -4 if going backwards in address space
      12 18 010E 746 BGEQ 130$ ;Branch if only 1 page or going forwards
      0110 747 ;
      0110 748 ; Going backwards in the address space, form the correct starting
      0110 749 ; PTE addresses and virtual address.
      0110 750 ;
      53 51 C0 0110 751 ADDL R1,R3 ;Form starting master PTE address
      67 53 D0 0113 752 MOVL R3,SVAPTE(R7) ;and save it
      40 A7 51 C0 0116 753 ADDL R1,PROCPTE(R7) ;Form starting process PTE address
51 51 07 78 011A 754 ASHL #7,R1,R1 ;(count - 1) * -512
      48 A7 51 C0 011E 755 ADDL R1,IRPSL_SEGVBN(R7) ;Form starting virtual address
18 A7 OC A7 D0 0122 756 130$: MOVL MFYCNT(R7),CLUSTER(R7) ;Loop count is to last modified page
      0127 757 ;
      0127 758 ; Given the Master PTE address get each page ready for the write request
      0127 759 ;
50 83 7B800000 8F CB 0127 760 150$: BICL3 #^C<PTESM_VALID !- ;Get relevant bits from PTE
      012F 761 PTESM_TYPT ! PTESM_TYPO !-
      012F 762 PTESM_PGFLVB>,(R3)†,R0
      35 19 012F 763 BLSS 260$ ;Branch if page is valid
      51 50 EA 8F 78 0131 764 BEQL 200$ ;Demand zero is inconsistent
      17 12 0138 765 ASHL #-PTESV_TYPO,R0,R1 ;as would be anything other
      03 00 EE 013A 766 BNEQ 200$ ;than transition
52 0000'DF40 013D 767 EXTV #PFNSV_LOC,#PFNS$LOC,- ;Get the page location (-4 to 3)
      0142 768 @W^PFNS$AB_STATE[R0],R2
      0142 769 CASE R2,<-
      0142 770 270$,- ;-1 = active
      0142 771 220$,- ;0 = on free page list
      0142 772 220$,- ;1 = on modified page list
      0142 773 220$,- ;2 = on bad page list
      0142 774 240$,- ;3 = release pending
      0142 775 >,TYPE=B,LIMIT=#-1
04' FF 8F 52 8F 0142 CASEB R2,#-1,S^#<<30001$-30000$>/2>-1
      0147 30000$: .SIGNED_WORD 270$-30000$
      0044' 0147 .SIGNED_WORD 220$-30000$
      000E' 0149 .SIGNED_WORD 220$-30000$
      000E' 014B .SIGNED_WORD 220$-30000$
      000E' 014D .SIGNED_WORD 220$-30000$
      0015' 014F .SIGNED_WORD 240$-30000$
      0151 30001$:
      0151 776 200$: BUG_CHECK WRTPGSBAK,FATAL ;Write pages tack - inconsistent data base
      FEFF 0151 .WORD ^XFÉFF

```

```

0004' 0153      .IIF IDN <FATAL>,<FATAL> , .WORD      BUG$_WRTPGSBAK!4
      0155      777 ;
      0155      778 ; Page is on the free, modified, or bad page list, must remove it
      0155      779 ;
      53 DD 0155 780 220$: PUSHL R3 ; Save next PTE address
FEA6' 30 0157 781 ; BSBW MMG$REMPFN ; Remove page from free or modified page list
      08 BA 015A 782 ; POPR #^M<R3> ; Restore next PTE address
      05 FO 015C 783 240$: INSV #PFNSC_WRTINPROG,#PFNSV LOC,- ; Set state to
0000'DF40 03 015F 784 ; #PFNS$_LOC,@W^PFNSAB_STATE[R0] ; Write in progress
      25 11 0164 785 ; BRB 270$
      0166 786 ;
      0166 787 ;
      0166 788 ; Master page table entry is valid, shut off PTE copy of Modify bit, and get PFN
      0166 789 ;
      51 40 A7 DO 0166 790 260$: MOVL PROCPTC(R7),R1 ; Process page table entry address
      61 D5 016A 791 ; TSTL (R1) ; See if it contains a valid PTE
      08 18 016C 792 ; BGEQ 265$ ; Branch if it does not
      07 61 1A E5 016E 793 ; BBCC #PTESV_MODIFY,(R1),265$ ; Shut off process PTE modify bit
      0172 794 ; ; Branch if it was already off
      51 48 A7 DO 0172 795 ; INVALID IRP$$_SEGVBN(R7),R1 ; Invalidate translation buffer for
      3A 51 DA 0176 ; MOVL IRP$$_SEGVBN(R7),R1
      0179 796 ; MTPR R1,S^#PR$_TBIS ; process virtual address
      0179 797 ;
      0179 798 ;
      FF A3 04 8A 0179 799 265$: ASSUME PTESV_MODIFY GE 24 ; PTE modify bit is in high byte
      50 50 15 00 EF 017D 800 ; BICB #PTES$_MODIFY@-24,-1(R3) ; Shut off modify in master PTE
      00000000'EF 50 D1 0182 801 ; EXTZV #PTESV_PFN,#PTES$_PFN,R0,R0 ; Isolate PFN
      0C 1A 0189 802 ; CMPL R0,MMG$GL_MAXPFN ; Is there PFN data base? (SH MEM page)
      0000'DF40 80 8F 8A 018B 803 270$: BICB #PFNSM_MODIFY,@W^PFNSAB_STATE[R0] ; Br if there is none, page is in SH MEM
      0000'DF40 B6 0192 804 ; INCW @W^PFNSAW_REF[NT[R0] ; Page not modified
      40 A7 04 C0 0197 805 280$: ADDL #4,PROCPTC(R7) ; Count an I/O reference
      48 A7 00000200 8F C0 019B 806 ; ADDL #512,IRP$$_SEGVBN(R7) ; Next process PTE address
      80 18 A7 F5 01A3 807 ; SOBGTR CLUSTER(R7),150$ ; Next process virtual address
      01A7 808 ; ; Loop through each page in the cluster
      01A7 809 ;
      01A7 810 ; Now set up to queue the packet for writing
      52 30 A7 DO 01A7 811 ; MOVL BAK(R7),R2 ; Get original backing store address
      01AB 812 ; ; section address is same for all pages
      53 67 DO 01AB 813 ; MOVL SVAPTE(R7),R3 ; Starting master PTE address
      50 63 15 00 EF 01AE 814 ; EXTZV #PTESV_PFN,#PTES$_PFN,(R3),R0 ; Get PFN for first page to write
      00000000'EF 50 D1 01B3 815 ; CMPL R0,MMG$GL_MAXPFN ; Is this a shared memory gbl sec page?
      31 1A 01BA 816 ; BGTRU 320$ ; Br if page is in shared memory gbl sec
      05 04 A7 16 E0 01BC 817 ; BBS #PTESV_TYPO,PTEDAT(R7),300$ ; Branch if process section page
      55 0000'CF DO 01C1 818 ; MOVL W^MMG$GL_SYSPHD,R5 ; System header for global page
      FE37' 30 01C6 819 300$: BSBW MMG$INIB[DPKT] ; Convert to file vbn and window
      51 0C A7 DO 01C9 820 310$: MOVL MFYCNT(R7),R1 ; Count of pages to queue
      01CD 821 ;
      00000002 01CD 822 ; .IF GT,CAS MEASURE
      0000'CF D6 01CD 823 ; INCL W^PMS$GL_PWRITIO ; Count number of write I/O requests
      0000'CF 51 C0 01D1 824 ; ADDL R1,W^PMS$GL_PWRITES ; Count number of pages written
      01D6 825 ; .ENDC
      01D6 826 ;
      55 57 DO 01D6 827 ; MOVL R7,R5 ; I/O packet address
      57 28 A5 09 78 01D9 828 ; ASHL #9,INC1(R5),R7 ; Restore R7
      51 DD 01DE 829 ; PUSHL R1 ; Save page count to return to caller
      51 51 09 9C 01E0 830 ; ROTL #9,R1,R1 ; Form byte count to queue

```

```

FE19' 30 01E4 831      BSBW  EXE$BUILDPKTW      ;Build and queue the packet for writing
50  01 3C 01E7 832      MOVZWL #SS$ NORMAL,R0      ;Indicate packet successfully queued
    32 BA 01EA 833      POPR   #^M<R1,R4,R5>      ;Return byte count in R1, restore R4,R5
    05 01EC 834      RSB      ;and return
    01ED 835
    01ED 836
    01ED 837      ; COMPUTE THE VBN FOR THE FIRST PAGE IN THE CLUSTER, THE SECTION TABLE ADDRESS,
    01ED 838      ; AND THE WINDOW ADDRESS.
    01ED 839
55  0000'CF D0 01ED 840 320$:  MOVL   W^MMG$GL_SYSPHD,R5      ;System process header (for gbl pages)
    52  52 32 01F2 841      CVTWL  R2,R2              ;Section table index
51  55  20 A5 C1 01F5 842      ADDL3  PHD$L PSTBASOFF(R5),R5,R1 ;Base of section table
    51  6142 DE 01FA 843      MOVAL  (R1)[R2],R1      ;Section table entry address
    0050 8F DB 01FE 844      PUSHR  #^M<R4,R6>      ;Save registers
    56  61 D0 0202 845      MOVL   SEC$L_GSD(R1),R6 ;Address of Global Section Descriptor
    0205 846
    0205 847      ; Find the relative position of this page within the section.
    0205 848
    50  FD18' 30 0205 849      BSBW  MMG$FINDSHD      ;Get sh mem ctl blk & common data page
    50  10 A4 C2 0208 850      SUBL2  SHB$L_BASGSPFN(R4),R0 ;Get relative PFN within the sh mem
    56  54 A6 9E 020C 851      MOVAB  GSD$L_BASPFN1(R6),R6 ;Get adr of first PFN base in GSD
    52  64 9A 0210 852      MOVZBL #GSD$L_PFNBSMAX,R2 ;Get number of PFN bases allowed
    66  50 D4 0213 853      CLRL  R5              ;Zero relative page offset within sec
    66  09 19 0215 854 330$:  CMPL  R0,(R6)          ;Is PFN less than this base?
    54  66 86 C1 0218 855      BLSS  340$            ;Br if less than, not within this piece
    54  54 50 D1 021A 856      ADDL3  (R6)+,(R6),R4 ;Get PFN past end of this piece
    54  0A 19 021E 857      CMPL  R0,R4          ;Is PFN less than end of piece?
    55  86 C0 0221 858      BLSS  350$            ;Br if less than, is within this piece
    EC 52 F5 0223 859 340$:  ADDL2  (R6)+,R5 ;Add pagcnt to relative page offset
    0226 860      SOBGTR R2,330$ ;Go check if PFN is in next piece
    0229 861      BUG_CHECK SCANDEADPT ;Error, PFN must be within this GSD
    FEFF 0229
    0000' 022B
    50  76 C2 022D 862 350$:  SUBL2  -(R6),R0 ;Get relative page within this piece
    50  55 C0 0230 863      ADDL2  R5,R0 ;Add page counts of other pieces to off
    50  10 A1 C0 0233 864      ADDL2  SEC$L_VBN(R1),R0 ;Add in base VBN
    0050 8F BA 0237 865      POPR   #^M<R4,R6> ;Restore registers
    52  0C A1 D0 023B 866      MOVL  SEC$L_WINDOW(R1),R2 ;Get window address
    88  11 023F 867      BRB   310$ ;Join common code

```

```
0241 869 .SBTTL PTEPFNMFY - Get PFN and Modify bit from PTE
0241 870
0241 871 :+
0241 872 :
0241 873 : FUNCTIONAL DESCRIPTION:
0241 874 :
0241 875 : Return PFN and modify bit if page is a candidate for write
0241 876 : back clustering.
0241 877 :
0241 878 : CALLING SEQUENCE:
0241 879 :
0241 880 : BSBW MMG$PTEPFNMFY
0241 881 :
0241 882 : INPUTS:
0241 883 :
0241 884 : R0 = Access mode to check against page owner
0241 885 : R1 = Exclusive writer indicator
0241 886 : R2 = Process section backing store address or GPTX
0241 887 : = 0 if supposed to return the above or shared memory global page
0241 888 : R3 = System Virtual Address of Page Table Entry
0241 889 : IPL = SYNCH
0241 890 :
0241 891 : OUTPUTS:
0241 892 :
0241 893 : R0 = Page Frame Number if successful
0241 894 : R1 = low bit clear if page is not a candidate for write back clustering
0241 895 : non-zero if actual error, 0 if just not a candidate
0241 896 : = low bit set if page could be cluster written
0241 897 : bit 7 set if modified page
0241 898 : R2 = Process section address if process page
0241 899 : = GPTX if global page
0241 900 : R3 preserved
0241 901 :
0241 902 :-
0241 903 :
0241 904 : *****
0241 905 :
0241 906 : ***** THE FOLLOWING CODE MUST BE RESIDENT *****
0241 907 :
00000241 908 : .PSECT $MMGCODE
0241 909 :
0241 910 : *****
0241 911 :
```

```

0241 913 .ENABL LSB
0241 914 :
0241 915 : Pages with PFN's greater than MAXPFN must be in shared memory (or PFN-mapped,
0241 916 : PTE$V_WINDOW set). Shared memory pages are always mapped via global sections.
0241 917 : There is no PFN data base for shared memory global section pages.
0241 918 :
0241 919 SHM_PAGE:
0053 8F BB 0241 920 PUSHR #*M<R0,R1,R4,R6> ;Save registers
51 D4 0245 921 CLRL R1 ;Indicate no decrement to PTE ref count
FDB6 30 0247 922 BSBW MMG$FINDGSDPFN ;Find SHMGSD for this PFN
25 50 E9 024A 923 BLBC R0,30$ ;Branch if none found (ERROR CONDITION)
52 A6 15 A4 91 024D 924 CMPB SHB$B_PORT(R4),GSD$B_CREATPORT(R6) ;Is process on creator port?
19 12 0252 925 BNEQ 20$ ;Br if different port, cannot do update
52 16 A6 3C 0254 926 MOVZWL GSD$W_GSTX(R6),R2 ;Get global section table index
50 D4 025A 927 CLRL R0 ;Assume page not a wrt candidate
13 20 A6 03 E1 025A 928 BBC #SECSV_WRT,GSD$W_FLAGS(R6),30$ ;Br if section not writeable
0053 8F BA 025F 929 POPR #*M<R0,R1,R4,R6> ;Restore registers
5E 04 C0 0263 930 ADDL2 #4,SP ;Clean off saved input backing store adr
00 52 16 E3 0266 931 BBCS #PTE$V_TYPO,R2,10$ ;Treat section as a process section
008E 31 026A 932 10$: BRW 100$ ;in WRTPGSBAK routine
50 0384 8F 3C 026D 933 20$: MOVZWL #SS$ NOTCREATOR,R0 ;Return error code
04 AE 50 D0 0272 934 30$: MOVL R0,47(SP) ;Insure that error code gets to R1
0053 8F BA 0276 935 POPR #*M<R0,R1,R4,R6> ;Restore registers
00D8 31 027A 936 BRW 180$ ;Page not candidate for update
027D 937
027D 938 MMG$PTEPFNMfY:
51 DD 027D 939 PUSHL R1 ;Save exclusive writer bit
52 DD 027F 940 PUSHL R2 ;and the input backing store address
51 53 15 09 EF 0281 941 EXTZV #VASV_VPN,#VASS_VPN,R3,R1 ;Check for presence of page table
0000'DF41 D5 0286 942 TSTL @W^MMG$GL_SPTBASE[R1] ;If SPT entry is not valid then
51 18 028B 943 BGEQ 70$ ;this page table is not resident
50 63 02 17 ED 028D 944 CMPZV #PTE$V_OWN,#PTE$S_OWN,(R3),R0 ;Check for page owner violation
75 19 0292 945 BLSS 130$ ;Branch if it is
50 63 7B800000 8F CB 0294 946 BICL3 #*C<PTE$M_VALID !- ;Get valid bit
029C 947 PTE$M_TYPT ! PTE$M_TYPO !- ;type bits
029C 948 PTE$M_PGFLVB>,(R3),R0 ;and PFN/GPTX from the PTE
72 18 029C 949 BGEQ 140$ ;Branch if not valid
3C 50 15 E0 029E 950 BBS #PTE$V_WINDOW,R0,70$ ;Branch if PFN-mapped
51 50 0D 9C 02A2 951 40$: ROTL #<32-<PTE$V_MODIFY-PFN$V_MODIFY>>,R0,R1 ;R1<7> = Modify bit
50 50 15 00 EF 02A6 952 EXTZV #PTE$V_PFN,#PTE$S_PFN,R0,R0 ;Isolate PFN
00000000'EF 50 D1 02AB 953 CMPL R0,MMG$GL_MAXPFN ;Is this a SH MEM page?
8D 1A 02B2 954 BGTRU SHM_PAGE ;Br if it is a SH MEM page
51 0000'DF40 88 02B4 955 50$: BISB @W^PFNSAB_STATE[R0],R1 ;Or in PFN copy of Modify bit
52 0000'DF40 D0 02BA 956 MOVL @W^PFNSAL_BAK[R0],R2 ;Backing store address to check
02C0 957 ;if page is not global
0000'DF40 53 D1 02C0 958 CMPL R3,@W^PFNSAL_PTE[R0] ;If process PTE address is different
52 0000'DF40 0000'CF 13 02C6 959 BEQL 60$ ;Branch if process page
52 52 1E 9C 02C8 960 SUBL3 W^MMG$GL_GPTBASE,@W^PFNSAL_PTE[R0],R2 ;Offset from GPT base
6E D5 02D5 962 60$: ROTL #<32-2>,R2,R2 ;Form Global Page Table Index
07 13 02D7 963 TSTL (SP) ;Specified section or GPTX?
6E 52 D1 02D9 964 BEQL 80$ ;Branch if not, return section or GPTX
05 13 02DC 965 CMPL R2,(SP) ;Yes, check that this one matches
73 11 02DE 966 BEQL 90$ ;Branch if it is
6E 52 D0 02E0 967 70$: BRB 170$ ;Not the same, end of cluster
52 0000'DF40 D0 02E3 968 80$: MOVL R2,(SP) ;Return the section or GPTX
66 52 16 E1 02E9 969 90$: MOVL @W^PFNSAL_BAK[R0],R2 ;Check that page is really writable
BBC #PTE$V_TYPO,R2,170$ ;making sure it is a section,

```



```

62 52 12 E1 02ED 970 BBC #PTESV_WRT,R2,170$ ;that it is writable
5E 52 10 E0 02F1 971 BBS #PTESV_CRF,R2,170$ ;and that it is not copy on reference
BA 02F5 972 POPR #*M<R2$ ;Fetch return section/GPTX
07 52 16 E0 02F7 973 BBS #PTESV_TYPO,P2,110$ ;Branch if not a global page
02FB 974 ;
02FB 975 ; For the case of Global pages, the 'complete' test for modified is not
02FB 976 ; possible since all process' which have valid PTE's for the global page
02FB 977 ; have their own copy of the modify bit. This is only folded back into
02FB 978 ; the PFN data base when the page is removed from the process' working
02FB 979 ; set. If the 'exclusive write' flag is set, a Global page is only
02FB 980 ; considered modified if the process PTE or the PFN data base says that
02FB 981 ; the page is modified. Otherwise, all Global Writable pages are considered
02FB 982 ; modified for the purposes of this write back logic.
02FB 983 ;
51 04 6E E8 02FB 984 100$: BLBS (SP),110$ ;Branch if exclusive writer
80 8F 88 02FE 985 BISB #PFNSM_MODIFY,R1 ;Force modify for global writable page
51 01 C8 0302 986 110$: BISL #1,R1 ;Indicate successful return
5E 04 C0 0305 987 120$: ADDL #4,SP ;Clean off save exclusive writer bit
05 0308 988 RSB
0309 989 ;
0309 990 ; Page owner violation
0309 991 ;
51 01EC 8F 3C 0309 992 130$: MOVZWL #SS$_PAGOWNVIO,R1 ;Return error status
45 11 030E 993 BRB 180$
0310 994 ;
0310 995 ; Page table entry was not valid, see if it is transition or global
0310 996 ;
51 50 EA 41 13 0310 997 140$: BEQL 170$ ;Branch if demand zero, end of cluster
8F 78 0312 998 ASHL #-PTESV_TYPO,R0,R1 ;Transition page?
23 13 0317 999 BEQL 160$ ;Branch if yes
0319 1000 ;
0319 1001 ; Process page table entry is not valid and not transition.
0319 1002 ; See if it is global.
0319 1003 ;
51 01 91 0319 1004 CMPB #1,R1 ;TYP1 = 0, TYPO = 1 ?
35 12 031C 1005 BNEQ 170$ ;Branch if not global
50 50 16 00 EF 031E 1006 EXTZV #PTESV_GPTX,#PTES$GPTX,R0,R0 ;Isolate GPTX
CB 0323 1007 BICL3 #*C<PTESM_VALID !- ;Get valid bit
0324 1008 PTESM_TYPT ! PTESM_TYPO !- ;type bits
0324 1009 PTESM_PGFLVB>,- ;and PFN/GPTX
50 0000'DF40 7B800000 8F 0324 1010 @W*MMG$GL_GPTBASE[R0],R0 ;from the global PTE
05 14 032E 1011 BGTR 150$ ;Branch if not valid and not DZRO
21 13 0330 1012 BEQL 170$ ;Branch if demand zero to end cluster
FF6D 31 0332 1013 BRW 40$ ;Process valid master PTE
51 50 EA 8F 78 0335 1014 150$: ASHL #-PTESV_TYPO,R0,R1 ;Check for transition state
17 12 033A 1015 BNEQ 170$ ;End of cluster if not
033C 1016 ;
033C 1017 ; This is a transition page. If it is on the free or modified page list
033C 1018 ; or in the RELPEND or ACTIVE state, then it is still a candidate.
033C 1019 ;
51 03 00 EE 033C 1020 160$: EXTV #PFNSV_LOC,#PFNS$LOC,- ;Get page location (-4 to 3)
0000'DF40 033F 1021 @W*PFNSAB_STATE[R0],R1
0344 1022 ;
0344 1023 ASSUME PFNSC_RDERR EQ 4 ;Page read error -4
0344 1024 ASSUME PFNSC_WRTINPROG EQ 5 ;Write in progress -3
0344 1025 ASSUME PFNSC_RDINPROG EQ 6 ;Read in progress -2
0344 1026 ASSUME PFNSC_ACTIVE EQ 7 ;Active -1

```

```

0344 1027 ASSUME PFNSC_FREPAGLST EQ 0 ;On free page list
0344 1028 ASSUME PFNSC_MFY PAGLST EQ 1 ;On modified page list
0344 1029 ASSUME PFNSC_BADPAGLST EQ 2 ;On bad page list
0344 1030 ASSUME PFNSC_RELPEND EQ 3 ;Release pending
0344 1031
0344 1032 CASE R1, <-
0344 1033 200$, - ; -1 = active
0344 1034 200$, - ; 0 = free page list
0344 1035 200$, - ; 1 = modified page list
0344 1036 190$, - ; 2 = bad page list
0344 1037 200$, - ; 3 = release pending
0344 1038 > TYPE=B, LIMIT=#-1
04' FF 8F 51 8F 0344 CASEB R1, #-1, S^# << 30003$-30002$ > / 2 > -1
0349 30002$:
0017' 0349 .SIGNED_WORD 200$-30002$
0017' 034B .SIGNED_WORD 200$-30002$
0017' 034D .SIGNED_WORD 200$-30002$
0010' 034F .SIGNED_WORD 190$-30002$
0017' 0351 .SIGNED_WORD 200$-30002$
0353 30003$:
0353 1039 : This page is not part of the current cluster
0353 1040 :
0353 1041 :
51 D4 0353 1042 170$: CLRL R1 ;Return error status
04 BA 0355 1043 180$: POPR #^M<R2> ;Clean off saved input backing store adr
AC 11 0357 1044 BRB 120$
0359 1045 :
0359 1046 : This page is on the bad page list, if it does not have the 'bad' bit
0359 1047 : set, then the page was placed there by the modified page writer due to
0359 1048 : a write error. In this case the page should be a candidate for write back.
0359 1049 :
F3 0000'DF40 05 E0 0359 1050 190$: BBS #PFNSV_BADPAG,@W^PFNSAB_TYPE[R0],170$ ;End cluster if bad bit set
0360 1051 :
0360 1052 : This page is resident and has no I/O pending. It may be clustered
0360 1053 :
51 D4 0360 1054 200$: CLRL R1 ;No modify bit from PTE
FF4F 31 0362 1055 BRW 50$
0365 1056 .DSABL LSB
0365 1057
0365 1058
0365 1059 .END

```

SYSUPDSEC
Symbol table

- Update Sect: File System Service

D 2

16-SEP-1984 02:36:29 VAX/VMS Macro V04-00
5-SEP-1984 03:57:55 [SYS.SRC]SYSUPDSEC.MAR;1

Page 24
(11)

SYS
V04

```

ACBSM_QUOTA = 0000004C
ACBSV_QUOTA = 00000006
ACMODE = 0000000C
ASTADR = 0000001C
ASTPRM = 00000020
BAK = 00000030
BUGS_SCANDEADPT ***** X 03
BUGS_WRTPGSBK ***** X 03
CAS_MEASURE = 00000002
CLUSTER = 00000018
COUNT = 0000001C
CTLSGL_PHD ***** X 02
DIR... = 00000001
DYN$C_IRP = 0000000A
EFN = 00000014
EXCLWRT = 00000020
EX$ALLOCBUF ***** X 02
EX$BUILDPKT ***** X 03
EX$DEANONPAGED ***** X 02
EX$SNGLEQUOTA ***** X 02
EX$UPDSEC = 00000001 RG 02
FLAGS = 00000010
GSD$B_CREATPORT = 00000052
GSD$C_PFN$B$MAX = 00000004
GSD$S_BASPFN1 = 00000054
GSD$W_FLAGS = 00000020
GSD$W_GSTX = 00000016
INADR = 00000004
INADRERR = 00000000 R 02
INC1 = 00000028
INC4 = 0000002C
IOC$DIRPCST1 ***** X 02
IOSB = 00000018
IPL$ SYNCH = 00000008
IRP$B_EFN = 00000022
IRP$B_PRI = 00000023
IRP$B_RMOD = 0000000B
IRP$B_TYPE = 0000000A
IRP$C_LENGTH = 000000C4
IRP$S_AST = 00000010
IRP$S_ASTPRM = 00000014
IRP$S_IOSB = 00000024
IRP$S_IOST1 = 00000038
IRP$S_IOST2 = 0000003C
IRP$S_PID = 0000000C
IRP$S_SEGVBN = 00000048
IRP$W_SIZE = 00000008
IRP_AST = 00000010
IRP_ASTPRM = 00000014
IRP_EFN = 00000022
IRP_IOSB = 00000024
IRP_IOST1 = 00000038
IRP_LENGTH = 0000004C
IRP_PRI = 00000023
IRP_RMOD = 0000000B
IRP_SEGVBN = 00000048
MFYCNT = 0000000C

```

```

MMG$CREDEL ***** X 02
MMG$C_LENGTH = FFFFFFFE4
MMG$FINDGSDPFN ***** X 03
MMG$FINDSHD ***** X 03
MMG$GL_GPTBASE ***** X 03
MMG$GL_MAXPFN ***** X 03
MMG$GL_SPTBASE ***** X 03
MMG$GL_SYSPHD ***** X 03
MMG$INADRINI ***** X 02
MMG$INIBLDPKT ***** X 03
MMG$S_MAXACMODE = FFFFFFFFC
MMG$S_SAVRETADR = FFFFFFFF4
MMG$S_SVSTARTVA = FFFFFFFEC
MMG$PTEINDX ***** X 03
MMG$PTEPFNMFY = 0000027D R 03
MMG$REMPFN ***** X 03
MMG$RETRANGE ***** X 02
MMG$UPDSECAST = 00000116 RG 02
MMG$UPDSECPAG = 000000E9 R 02
MMG$UPDSECQWT = 00000000 R 03
MMG$WRTPGSBK = 0000006D RG 03
MPW$GW_MPW$PFC ***** X 03
PCB$B_PRI = 0000002F
PCB$S_PHD = 0000006C
PCB$S_PID = 00000060
PCB$W_ASTCNT = 00000038
PCB$W_DIOCNT = 0000003E
PFN$B_STATE ***** X 03
PFN$B_TYPE ***** X 03
PFN$AL_BAK ***** X 03
PFN$AL_PTE ***** X 03
PFN$AW_REFCNT ***** X 03
PFN$C_ACTIVE = 00000007
PFN$C_BADPAGLST = 00000002
PFN$C_FREPAGLST = 00000000
PFN$C_MFY$PAGLST = 00000001
PFN$C_RDERR = 00000004
PFN$C_RDINPROG = 00000006
PFN$C_RELPEND = 00000003
PFN$C_WRTINPROG = 00000005
PFN$M_MODIFY = 00000080
PFN$S_LOC = 00000003
PFN$V_BADPAG = 00000005
PFN$V_LOC = 00000000
PFN$V_MODIFY = 00000007
PHD$S_PSTBASOFF = 00000020
PMS$GL_PWRITES ***** X 03
PMS$GL_PWRITIO ***** X 03
PR$IPC = 00000012
PR$TBIS = 0000003A
PRIS_IOCOM = 00000001
PROCPTE = 00000040
PSL$S_PRIVMOD = 00000002
PSL$V_PRIVMOD = 00000016
PTE$M_MODIFY = 04000000
PTE$M_PGFLVB = 003FFFFFFF
PTE$M_TYPO = 00400000

```

SYSUPDSEC
Symbol table

- Update Section File System Service^{E 2}

16-SEP-1984 02:36:29 VAX/VMS Macro V04-00
5-SEP-1984 03:57:55 [SYS.SRC]SYSUPDSEC.MAR;1

Page 25
(11)

SYS
V04

```

PTESH_TYP1      = 04000000
PTESH_VALID     = 80000000
PTESH_GPTX      = 00000016
PTESH_OWN       = 00000002
PTESH_PFN       = 00000015
PTESH_CRF       = 00000010
PTESH_GPTX      = 00000000
PTESH_MODIFY    = 0000001A
PTESH_OWN       = 00000017
PTESH_PFN       = 00000000
PTESH_TYPO      = 00000016
PTESH_WINDOW    = 00000015
PTESH_WRT       = 00000012
PTEDAT          = 00000004
RETADR          = 00000008
SAVABS...       = 000000D4
SCH$CLREF       ***** X 02
SCH$POSTEF      ***** X 02
SECSL_GSD       = 00000000
SECSL_VBN       = 00000010
SECSL_WINDOW    = 0000000C
SECSV_WRT       = 00000003
SHB$B_PORT      = 00000015
SHB$B_BASGSPFN = 00000010
SHM_PAGE        00000241 R 03
SS$ACCVIO       = 0000000C
SS$EXQUOTA      = 0000001C
SS$IVSECFLG     = 0000016C
SS$NORMAL       = 00000001
SS$NOTCREATOR   = 00000384
SS$NOTMODIFIED  = 00000659
SS$PAGOWNVIO    = 000001EC
SVAPTE          = 00000000
VASS_VPN        = 00000015
VASV_VPN        = 00000009
XIP_B_MAXACMODE = 000000D1
XIP_B_UPDFLG    = 000000D0
XIP_C_LENGTH    = 000000D4
XIP_L_DIREC     = 000000C8
XIP_L_SCANCNT   = 000000C4
XIP_L_STARTVA   = 000000CC
  
```

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	000000D4 (212.)	01 (1.)	NCPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
Y\$EXEPAGED	00000189 (393.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$MMGCOD	00000365 (869.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.07	00:00:00.26
Command processing	107	00:00:00.56	00:00:01.06
Pass 1	430	00:00:15.58	00:00:18.15
Symbol table sort	0	00:00:02.32	00:00:02.41
Pass 2	207	00:00:03.69	00:00:04.11
Symbol table output	19	00:00:00.15	00:00:00.15
Psect synopsis output	1	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	797	00:00:22.39	00:00:26.16

The working set limit was 1650 pages.
94749 bytes (186 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1436 non-local and 73 local symbols.
1059 source lines were read in Pass 1, producing 23 object records in Pass 2.
36 pages of virtual memory were used to define 34 macros.

! Macro library statistics !

Macro library name	Macros defined
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	21
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	10
TOTALS (all libraries)	31

1596 GETS were required to define 31 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:SYSUPDSEC/OBJ=OBJ\$:SYSUPDSEC MSRC\$:SYSUPDSEC/UPDATE=(ENH\$:SYSUPDSEC)+EXECMLS/LIB

0388 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SYSSCHVUT LIS	SYSSNDJBC LIS	SYSSNDMSG LIS	SYSSUNWIND LIS	SYSETEXU LIS	SYSETPRU LIS	SYSETPRV LIS	SYSETSTK LIS	SYSETMOD LIS	SYSETPRI LIS	SYSETPRM LIS	SYSETIME LIS	SYSETPRA LIS	SYSETSSF LIS
SYSSCHVUT LIS	SYSSNDJBC LIS	SYSSNDMSG LIS	SYSSUNWIND LIS	SYSETEXU LIS	SYSETPRU LIS	SYSETPRV LIS	SYSETSTK LIS	SYSETMOD LIS	SYSETPRI LIS	SYSETPRM LIS	SYSETIME LIS	SYSETPRA LIS	SYSETSSF LIS
SYSSCHVUT LIS	SYSSNDJBC LIS	SYSSNDMSG LIS	SYSSUNWIND LIS	SYSETEXU LIS	SYSETPRU LIS	SYSETPRV LIS	SYSETSTK LIS	SYSETMOD LIS	SYSETPRI LIS	SYSETPRM LIS	SYSETIME LIS	SYSETPRA LIS	SYSETSSF LIS
SYSSCHVUT LIS	SYSSNDJBC LIS	SYSSNDMSG LIS	SYSSUNWIND LIS	SYSETEXU LIS	SYSETPRU LIS	SYSETPRV LIS	SYSETSTK LIS	SYSETMOD LIS	SYSETPRI LIS	SYSETPRM LIS	SYSETIME LIS	SYSETPRA LIS	SYSETSSF LIS
SYSSCHVUT LIS	SYSSNDJBC LIS	SYSSNDMSG LIS	SYSSUNWIND LIS	SYSETEXU LIS	SYSETPRU LIS	SYSETPRV LIS	SYSETSTK LIS	SYSETMOD LIS	SYSETPRI LIS	SYSETPRM LIS	SYSETIME LIS	SYSETPRA LIS	SYSETSSF LIS
SYSSCHVUT LIS	SYSSNDJBC LIS	SYSSNDMSG LIS	SYSSUNWIND LIS	SYSETEXU LIS	SYSETPRU LIS	SYSETPRV LIS	SYSETSTK LIS	SYSETMOD LIS	SYSETPRI LIS	SYSETPRM LIS	SYSETIME LIS	SYSETPRA LIS	SYSETSSF LIS
SYSSCHVUT LIS	SYSSNDJBC LIS	SYSSNDMSG LIS	SYSSUNWIND LIS	SYSETEXU LIS	SYSETPRU LIS	SYSETPRV LIS	SYSETSTK LIS	SYSETMOD LIS	SYSETPRI LIS	SYSETPRM LIS	SYSETIME LIS	SYSETPRA LIS	SYSETSSF LIS
SYSSCHVUT LIS	SYSSNDJBC LIS	SYSSNDMSG LIS	SYSSUNWIND LIS	SYSETEXU LIS	SYSETPRU LIS	SYSETPRV LIS	SYSETSTK LIS	SYSETMOD LIS	SYSETPRI LIS	SYSETPRM LIS	SYSETIME LIS	SYSETPRA LIS	SYSETSSF LIS
SYSSCHVUT LIS	SYSSNDJBC LIS	SYSSNDMSG LIS	SYSSUNWIND LIS	SYSETEXU LIS	SYSETPRU LIS	SYSETPRV LIS	SYSETSTK LIS	SYSETMOD LIS	SYSETPRI LIS	SYSETPRM LIS	SYSETIME LIS	SYSETPRA LIS	SYSETSSF LIS
SYSSCHVUT LIS	SYSSNDJBC LIS	SYSSNDMSG LIS	SYSSUNWIND LIS	SYSETEXU LIS	SYSETPRU LIS	SYSETPRV LIS	SYSETSTK LIS	SYSETMOD LIS	SYSETPRI LIS	SYSETPRM LIS	SYSETIME LIS	SYSETPRA LIS	SYSETSSF LIS

