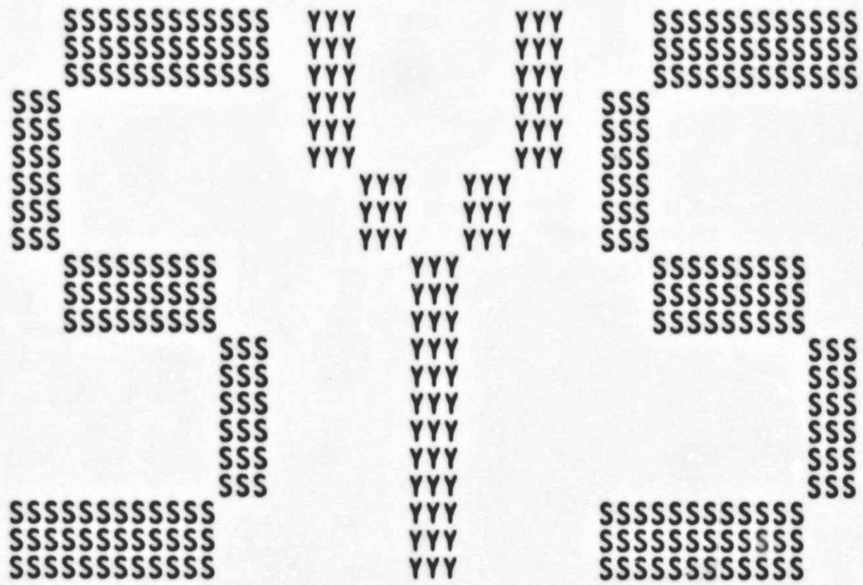
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSSSSSSSSSS YYY SSSSSSSSSSS
SSSSSSSSSSS YYY SSSSSSSSSSS
SSSSSSSSSSS YYY SSSSSSSSSSS

FILEID**SHMGSDRTN

```
  SSSSSSSS  HH      HH  MM      MM   GGGGGGGG   SSSSSSSS  DDDDDDDD   RRRRRRRR   TTTTTTTTTT  NN      NN
  SSSSSSSS  HH      HH  MM      MM   GGGGGGGG   SSSSSSSS  DDDDDDDD   RRRRRRRR   TTTTTTTTTT  NN      NN
SS          HH      HH  MMMM  MMMM  GG         SS        DD      DD RR      RR     TT      NN      NN
SS          HH      HH  MMMM  MMMM  GG         SS        DD      DD RR      RR     TT      NN      NN
SS          HH      HH  MM  MM  MM  GG         SS        DD      DD RR      RR     TT      NNNN    NN
SS          HH      HH  MM  MM  MM  GG         SS        DD      DD RR      RR     TT      NNNN    NN
  SSSSSS    HHHHHHHHHH  MM      MM  GG           SSSSSS  DD      DD RRRRRRRR     TT      NN  NN  NN
  SSSSSS    HHHHHHHHHH  MM      MM  GG           SSSSSS  DD      DD RRRRRRRR     TT      NN  NN  NN
        SS  HH      HH  MM      MM  GG  GGGGG          SS DD      DD RR  RR       TT      NN    NNNN
        SS  HH      HH  MM      MM  GG  GGGGG          SS DD      DD RR  RR       TT      NN    NNNN
        SS  HH      HH  MM      MM  GG    GG           SS DD      DD RR    RR     TT      NN      NN    ....
        SS  HH      HH  MM      MM  GG    GG           SS DD      DD RR    RR     TT      NN      NN    ....
SSSSSSSS    HH      HH  MM      MM    GGGGGG   SSSSSSSS  DDDDDDDD   RR      RR   TT      NN      NN    ....
SSSSSSSS    HH      HH  MM      MM    GGGGGG   SSSSSSSS  DDDDDDDD   RR      RR   TT      NN      NN    ....

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

Table of contents

```
0000     1          .TITLE  SHMGSDRTN - GLOBAL SECTION DESCRIPTOR ROUTINES FOR SHARED MEMORY
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: MEMORY MANAGEMENT
0000    29  ;
0000    30  ; ABSTRACT: ROUTINES TO TRANSLATE LOGICAL NAMES FOR GLOBAL SECTION NAMES,
0000    31  ;           SEARCH ALL GSD LISTS AND TABLES, AND HANDLE SHARED MEMORY
0000    32  ;           GLOBAL SECTION PAGE AND DESCRIPTOR RESOURCES.
0000    33  ;
0000    34  ; ENVIRONMENT: VAX/VMS
0000    35  ;
0000    36  ; AUTHOR: KATHLEEN D. MORSE     , CREATION DATE: 15-JAN-1979
0000    37  ;
0000    38  ; MODIFIED BY:
0000    39  ;
0000    40  ;       V03-009 MSH0042         Michael S. Harvey       4-May-1984
0000    41  ;               Object name buffer also must be zero filled.
0000    42  ;
0000    43  ;               Shared memory name buffer must be zero filled for successful
0000    44  ;               matching of name as stored in shared memory common data page.
0000    45  ;               (This was ident V03-008, 3-May-1984).
0000    46  ;
0000    47  ;       V03-007 TMK0001         Todd M. Katz            23-Apr-1984
0000    48  ;               Completely re-write the routines MMG$GSDTRNLOG, MMG$MBXTRNLOG,
0000    49  ;               and MMG$CEFTRNLOG. The basic changes made include:
0000    50  ;
0000    51  ;               1. Use of the fast internal logical name routine LNM$SEARCH_ONE
0000    52  ;                  to do each iterative translation instead of making iterative
0000    53  ;                  calls to the old $TRNLOG system service.
0000    54  ;
0000    55  ;               2. Extension of the size of logical names from the old 63 byte
0000    56  ;                  value to LNM$C_NAMLENGTH.
0000    57  ;
```

```
0000     58 ;                3. Use of a KRP to provide space for a logical name translation
0000     59 ;                   work area instead of the kernel stack.
0000     60 ;
0000     61 ;                4. Micro-optimiztion and extensive documentation of the
0000     62 ;                   three routines.
0000     63 ;
0000     64 ;        V03-006 MSH0036         Michael S. Harvey       20-Apr-1984
0000     65 ;                Correct upper bounds check on global section names for
0000     66 ;                shared memory global sections.
0000     67 ;
0000     68 ;        V03-005 MSH0004         Michael S. Harvey       26-Jan-1984
0000     69 ;                Add support for lengthened global name field in global
0000     70 ;                section descriptors.
0000     71 ;
0000     72 ;        V03-004 DMW4037         DMWalp                  26-May-1983
0000     73 ;                Intergate new logical name structures.
0000     74 ;
0000     75 ;        V03-003 KDM0028         Kathleen D. Morse       10-Nov-1982
0000     76 ;                Fix demand-zeroing of shared memory global section
0000     77 ;                that is mapped backwards by reversing the INADR range.
0000     78 ;
0000     79 ;--
```

```
                        .SBTTL  DECLARATIONS

;
; MACROS:
;

        $DYNDEF                         ;DEFINE SYSTEM DATA STRUCTURES
        $GSDDEF                         ;GLOBAL SECTION DESCRIPTOR
        $IPLDEF                         ;INTERRUPT PRIORITY LEVELS
        $IRPDEF                         ;I/O REQUEST PACKET
        $LNMDEF                         ;DEFINE LOGICAL NAME ATTRIBUTES
        $LNMSTRDEF                      ;DEFINE LOGICAL NAME BLOCKS OFFSETS
        $PCBDEF                         ;PROCESS CONTROL BLOCK
        $PHDDEF                         ;PROCESS HEADER
        $PRDEF                          ;PRIVILEGE BITS
        $PRIDEF                         ;PRIORITY LEVELS
        $PSLDEF                         ;PROGRAM STATUS LONGWORD
        $PTEDEF                         ;DEFINE PAGE TABLE ENTRIES
        $SECDEF                         ;SECTION TABLE ENTRY
        $SHBDEF                         ;SHARED MEMORY CONTROL BLOCK
        $SHDDEF                         ;SHARED MEMORY COMMON DATA PAGE
        $SSDEF                          ;SYSTEM STATUS CODES
        $STATEDEF                       ;DEFINE EVENT STATES
        $VADEF                          ;VIRTUAL ADDRESS DEFINITIONS
        $WCBDEF                         ;DEFINE WINDOW CONTROL BLOCK

;
; EQUATED SYMBOLS:
;

;
; OWN STORAGE:
;
```

```
                     0000   116          .SBTTL CLR/SET_BITMAP - CLEAR/SET BITS IN SHARED MEMORY GBL SEC BITMAP
                     0000   117 ;++
                     0000   118 ; FUNCTIONAL DESCRIPTION:
                     0000   119 ;
                     0000   120 ;       THIS ROUTINE CLEARS/SETS THE BITS IN THE GLOBAL SECTION BITMAP
                     0000   121 ;       CORRESPONDING TO SPECIFIC PHYSICAL PAGE FRAME NUMBERS (PFN) ASSOCIATED
                     0000   122 ;       WITH A GLOBAL SECTION SPECIFIED BY A GSD.  THE GSD CONTAINS UP
                     0000   123 ;       TO #GSD$C_PFNBASMAX PIECES, EACH PIECE DESCRIBED BY TWO LONGWORDS:
                     0000   124 ;       THE RELATIVE PFN OF THE FIRST PAGE IN THE PIECE, AND A COUNT OF THE
                     0000   125 ;       NUMBER OF PAGES IN THE PIECE.  USING THIS INFORMATION, THIS ROUTINE
                     0000   126 ;       COMPUTES THE ADDRESS OF THE BITS IN THE BITMAP THAT CORRESPOND TO
                     0000   127 ;       THESE RELATIVE PFN'S.  THESE BITS ARE THEN CLEARED/SET FOR EACH PIECE OF
                     0000   128 ;       THE GLOBAL SECTION.
                     0000   129 ;
                     0000   130 ; CALLING SEQUENCE:
                     0000   131 ;
                     0000   132 ;       BSBW    MMG$SET_BITMAP
                     0000   133 ;       BSBW    MMG$CLR_BITMAP
                     0000   134 ;
                     0000   135 ; INPUT PARAMETERS:
                     0000   136 ;
                     0000   137 ;       R5 - ADDRESS OF THE SHARED MEMORY COMMON DATA PAGE
                     0000   138 ;       R6 - ADDRESS OF THE GLOBAL SECTION DESCRIPTOR
                     0000   139 ;
                     0000   140 ; IMPLICIT INPUTS:
                     0000   141 ;
                     0000   142 ;       THE GLOBAL SECTION DESCRIPTOR HAS BEEN INITIALIZED.
                     0000   143 ;
                     0000   144 ; OUTPUT PARAMETERS:
                     0000   145 ;
                     0000   146 ;       NONE
                     0000   147 ;
                     0000   148 ; IMPLICIT OUTPUTS:
                     0000   149 ;
                     0000   150 ;       THE CORRESPONDING BITS IN THE BITMAP ARE CLEARED/SET.
                     0000   151 ;
                     0000   152 ; COMPLETION CODES:
                     0000   153 ;
                     0000   154 ;       NONE
                     0000   155 ;
                     0000   156 ; SIDE EFFECTS:
                     0000   157 ;
                     0000   158 ;       NONE
                     0000   159 ;
                     0000   160 ;--
                     0000   161 ; ****************************************************************************
                     0000   162 ;
                     0000   163 ; ********************* THE FOLLOWING CODE MAY BE PAGED ********************
                     0000   164 ;
                00000000    165          .PSECT  Y$EXEPAGED
                     0000   166 ;
                     0000   167 ; ****************************************************************************
                     0000   168
                     0000   169          .ENABL  LSB
                     0000   170 MMG$SET_BITMAP::
079F 8F      BB      0000   171          PUSHR   #^M<R0,R1,R2,R3,R4,R7,R8,R9,R10> ;SAVE REGISTERS
54   00      D2      0004   172          MCOML   #0,R4                            ;INDICATE BITS ARE TO BE SET
```

```
                     06   11  0007   173          BRB     5$                                      ;ENTER COMMON CODE
                                 0009   174
                                 0009   175 MMG$CLR_BITMAP::
                079F 8F   BB  0009   176          PUSHR   #^M<R0,R1,R2,R3,R4,R7,R8,R9,R10> ;SAVE REGISTERS
                     54   D4  000D   177          CLRL    R4                                      ;INDICATE BITS ARE TO BE CLEARED
          57    0C A5   55   C1  000F   178 5$:      ADDL3   R5,SHD$L_GSBITMAP(R5),R7                ;GET ADR OF BITMAP
                50    54 A6   9E  0014   179          MOVAB   GSD$L_BASPFN1(R6),R0                    ;GET ADR OF FIRST BASE PFN
                51    04   9A  0018   180          MOVZBL  #GSD$C_PFNBASMAX,R1                     ;GET # OF BASES ALLOWED IN GSD
                                 001B   181
                                 001B   182 FIND_PIECE:
                52    80   D0  001B   183          MOVL    (R0)+,R2                                ;GET RELATIVE BASE PFN
                53    80   D0  001E   184          MOVL    (R0)+,R3                                ;GET SIZE OF THIS PIECE
                      40   13  0021   185          BEQL    ALL_DONE                                ;BR ON NO MORE PIECES OF SECTION
                                 0023   186 ;
                                 0023   187 ; COMPUTE THE BYTE ADDRESS OF THE FIRST BIT TO CLEAR IN THE BITMAP.
                                 0023   188 ;
          5A    52    FD 8F   78  0023   189          ASHL    #-3,R2,R10                              ;GET # BYTES OFFSET INTO BITMAP
             58    5A   57   C1  0028   190          ADDL3   R7,R10,R8                               ;BYTE ADR OF FIRST BIT TO CLEAR
             59    5A   03   78  002C   191          ASHL    #3,R10,R9                               ;GET # OF BITS SKIPPED
             59    52   59   C3  0030   192          SUBL3   R9,R2,R9                                ;BIT OFFSET FOR FIRST BIT TO CLR
                                 0034   193 ;
                                 0034   194 ; LOOP CLEARING THE REMAINING BITS IN THE FIRST BYTE OF THE BITMAP TO BE
                                 0034   195 ; CHANGED.
                                 0034   196 ;
       68    01    59   54   F0  0034   197 10$:      INSV    R4,R9,#1,(R8)                           ;CLEAR/SET ONE BIT OF BITMAP
                      53   D7  0039   198          DECL    R3                                      ;ONE LESS PAGE TO CLR BIT FOR
                      23   13  003B   199          BEQL    NEXT_PIECE                              ;BR IF NO MORE PAGES IN PIECE
                      59   D6  003D   200          INCL    R9                                      ;POINT TO NEXT BIT OF BYTE
                08    59   91  003F   201          CMPB    R9,#8                                   ;DONE WITH THIS BYTE?
                      F0   19  0042   202          BLSS    10$                                     ;BR ON NO, GO CLEAR ANOTHER BIT
                                 0044   203 ;
                                 0044   204 ; NOW DETERMINE THE NUMBER OF BYTES OF BITMAP THAT ARE TO BE TOTALLY CLEARED.
                                 0044   205 ; CLEAR THESE BYTES WITH CLRB INSTRUCTIONS, THEN LOOP BACK TO CLEAR THE BITS
                                 0044   206 ; AT THE END OF THE PIECE OF BITMAP WHICH DO NOT USE AN ENTIRE BYTE.
                                 0044   207 ;
                      58   D6  0044   208          INCL    R8                                      ;POINT TO NEXT BYTE OF BITMAP
                      59   D4  0046   209          CLRL    R9                                      ;INDICATE FIRST BIT OF BYTE
          5A    53    FD 8F   78  0048   210          ASHL    #-3,R3,R10                              ;COMPUTE # OF BYTES TO CLEAR
             52    5A   03   78  004D   211          ASHL    #3,R10,R2                               ;COMPUTE # OF BITS CLEARED
                      08   13  0051   212          BEQL    25$                                     ;BR IF NO WHOLE BYTES TO CLR
       88    08    59   54   F0  0053   213 20$:      INSV    R4,R9,#8,(R8)+                          ;CLEAR 8 BITS OF BITMAP
                F8    5A   F5  0058   214          SOBGTR  R10,20$                                 ;ONE LESS BYTE TO CLEAR
                53    52   C2  005B   215 25$:      SUBL2   R2,R3                                   ;COMPUTE # OF BITS LEFT TO CLR
                      D4   14  005E   216          BGTR    10$                                     ;BR TO CLEAR REMAINING BITS (<8)
                                 0060   217 ;
                                 0060   218 ; REPEAT CLEARING BITS FOR UP TO #GSD$C_PFNBASMAX PIECES OF SHARED MEMORY.
                                 0060   219 ;
                                 0060   220 NEXT_PIECE:
                   B8 51    F5  0060   221          SOBGTR  R1,FIND_PIECE                           ;BR TO GET NEW BASE PFN AND CNT
                                 0063   222
                                 0063   223 ALL_DONE:
                079F 8F   BA  0063   224          POPR    #^M<R0,R1,R2,R3,R4,R7,R8,R9,R10> ;RESTORE REGISTERS
                      05   0067   225          RSB
                                 0068   226          .DSABL  LSB
```

```
                        .SBTTL  FINDGSDPFN - FIND GSD USING SPECIFIC PFN
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       THIS ROUTINE TAKES A PFN AND SEARCHES THE SHARED MEMORIES TO FIND
                ;       THE GLOBAL SECTION THAT IS MAPPED TO THAT PFN.  IT THEN DECREMENTS THE
                ;       PTE REFERENCE COUNT BY ONE.  THE ROUTINE IS CALLED WHENEVER A PROCESS
                ;       DELETES A VIRTUAL PAGE WHICH IS MAPPED TO A SHARED MEMORY GLOBAL
                ;       SECTION.  NOTE:  ALL PAGES IN SHARED MEMORY ARE ASSUMED TO HAVE PFN'S
                ;       GREATER THAN MMG$GL_MAXPFN (THE MAXIMUM LOCAL MEMORY PFN CONTAINED IN
                ;       THE PFN DATA BASE).
                ;
                ; CALLING SEQUENCE:
                ;
                ;       BSBW    MMG$FINDGSDPFN
                ;
                ; INPUT PARAMETERS:
                ;
                ;       R0 - THE PFN TO BE LOCATED
                ;       R1 - COUNT TO DECREMENT PTE REFERENCE BY
                ;            (0 FROM MMG$PTEPFNMFY)     (1 FROM MMG$DELPAG)
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       THE SHARED MEMORY CONTROL BLOCKS, SHARED MEMORY COMMON DATA PAGES,
                ;       AND THE SHARED MEMORY GLOBAL SECTION DESCRIPTOR TABLES.
                ;
                ; OUTPUT PARAMETERS:
                ;
                ;       R4 - SHARED MEMORY CONTROL BLOCK ADDRESS (SHB)
                ;       R6 - GLOBAL SECTION DESCRIPTOR ADDRESS (GSD)
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       THE PROCESSOR REFERENCE COUNT IN THE GSD THAT IS MAPPED TO THIS PFN IS
                ;       DECREMENTED BY ONE, IF THE GSD IS FOUND.
                ;
                ; COMPLETION CODES:
                ;
                ;       SS$_NOSUCHSEC - NO CORRESPONDING GSD FOUND FOR PFN
                ;       SS$_NORMAL - SUCCESSFUL DECREMENT OF GSD REF COUNT
                ;
                ; SIDE EFFECTS:
                ;
                ;       NONE
                ;
                ;--
                ;
                ; ******************************************************************************
                ;
                ; ************** THE FOLLOWING CODE MUST BE RESIDENT ********************
                ;
00000000        .PSECT  $MMGCOD
                ;
                ; ******************************************************************************
                ;
MMG$FINDGSDPFN::
```

```
                                0000   285          .ENABL  LSB
                    05AE 8F     BB 0000   286          PUSHR   #^M<R1,R2,R3,R5,R7,R8,R10>       ;SAVE REGISTERS
                         01     DD 0004   287          PUSHL   #1                              ;PUSH A POSITIVE VALUE TO
                    5A   5E     D0 0006   288          MOVL    SP,R10                          ;INDICATE TO MMG$VALIDATE AND
                                0009   289                                                  ;MMG$GETNXTGSD NOT TO USE ALL
                                0009   290                                                  ;SHARED MEMORIES IN SEARCH
                                0009   291                                                  ;JUST THE ONE PASSED IN R4,R5
              54  00000000'GF   D0 0009   292          MOVL    G^EXE$GL_SHBLIST,R4             ;GET FIRST SH MEM CONTROL BLOCK
                         5D     13 0010   293          BEQL    NOT_FOUND                       ;BR ON NO SH MEMORIES CONNECTED
                                0012   294 ;
                                0012   295 ; A SHARED MEMORY MAY HAVE A SHARED MEMORY CONTROL BLOCK (SHB) BUT MAY NOT BE
                                0012   296 ; CONNECTED (I.E., AVAILABLE FOR USE).  THEREFORE, ONCE AN SHB IS FOUND, THE
                                0012   297 ; BIT, SHB$V_CONNECT, MUST BE SET FOR IT TO BE USED.
                                0012   298 ;
                 53 0B A4       00 E1 0012   299 10$:    BBC     #SHB$V_CONNECT,SHB$B_FLAGS(R4),GET_NXT_SHM ;BR ON SHM DISCONCT
                      55    04 A4 D0 0017   300          MOVL    SHB$L_DATAPAGE(R4),R5           ;GET ADR OF COMMON DATA PAGE
                52    50    10 A4 C3 001B   301          SUBL3   SHB$L_BASGSPFN(R4),R0,R2        ;GET RELATIVE PFN WITHIN MEM
                            48 19 0020   302          BLSS    GET_NXT_SHM                     ;BR IF PFN NOT IN THIS SH MEM
                      10 A5    52 D1 0022   303          CMPL    R2,SHD$L_GSPAGCNT(R5)           ;IS PFN < MAX REL PFN FOR GS?
                            42 14 0026   304          BGTR    GET_NXT_SHM                     ;BR IF PFN NOT IN THIS SH MEM
                                0028   305 ;
                                0028   306 ; THE SHARED MEMORY CONTAINING THIS PFN HAS BEEN FOUND.  NOW FIND THE GSD
                                0028   307 ; WHICH IS MAPPED TO THIS PFN.
                                0028   308 ;
              56    55    04 A5 C1 0028   309          ADDL3   SHD$L_GSDPTR(R5),R5,R6          ;GET ADR OF FIRST GSD IN SHM TBL
                       0068 30 002D   310          BSBW    MMG$VALIDATEGSD                 ;CHECK THAT GSD IS VALID
                            56 D5 0030   311 30$:    TSTL    R6                              ;WAS A VALID GSD FOUND?
                            3B 13 0032   312          BEQL    NOT_FOUND                       ;BR ON GSD FOR PFN NOT FOUND
                         53 04 9A 0034   313          MOVZBL  #GSD$C_PFNBASMAX,R3             ;GET # OF MAX BASE PFN'S IN GSD
                      57    54 A6 9E 0037   314          MOVAB   GSD$L_BASPFN1(R6),R7            ;GET ADR OF FIRST BASE PFN
                            67 52 D1 003B   315 40$:    CMPL    R2,(R7)                         ;IS PFN > BASE PFN?
                            0C 19 003E   316          BLSS    50$                             ;BR IF PFN IS NOT IN THIS PIECE
              58    04 A7       67 C1 0040   317          ADDL3   (R7),4(R7),R8                   ;GET PFN OF PAGE BEYOND PIECE
                            0B 13 0045   318          BEQL    60$                             ;BR IF NO MORE PIECES USED
                               58 52 D1 0047   319          CMPL    R2,R8                           ;IS PFN < LAST PAGE IN PIECE?
                            0B 19 004A   320          BLSS    FOUND_IT                        ;BR IF PFN IS IN THIS PIECE
                               57 08 C0 004C   321 50$:    ADDL2   #8,R7                           ;POINT TO NEXT BASE PFN
                             E9 53 F5 004F   322          SOBGTR  R3,40$                          ;GO CHECK IF PFN IS IN NXT PIECE
                          0047 30 0052   323 60$:    BSBW    MMG$GETNXTGSD                   ;GET THE NEXT GSD IN SHM TBL
                            D9 11 0055   324          BRB     30$                             ;GO CHECK IF PFN IS IN THIS GSD
                                0057   325 ;
                                0057   326 ; THE GSD MAPPED TO THE SPECIFIC PFN HAS BEEN FOUND.  THE PTE CORRESPONDING
                                0057   327 ; TO THIS PFN IS BEING DELETED.  THEREFORE, THE PROCESSOR REFERENCE COUNT
                                0057   328 ; IN THE GSD MUST BE DECREMENTED BY ONE. (IN OTHER WORDS, THERE WILL BE ONE
                                0057   329 ; LESS PTE MAPPED TO THIS GLOBAL SECTION)
                                0057   330 ;
                                0057   331 FOUND_IT:
              50    04 AE       01 C1 0057   332          ADDL3   #1,4(SP),R0                     ;GET REFCNT + LCK TO DECREMENT
                          0017 30 005C   333          BSBW    MMG$DECSHMREF                   ;ONE LESS REF FOR THIS PTE
                               50 01 9A 005F   334          MOVZBL  #SS$_NORMAL,R0                  ;SET RETURN CODE TO SUCCESS
                                0062   335 ;
                                0062   336 ; RETURN SUCCESSFULLY HERE.
                                0062   337 ;
                                0062   338 RSB_HERE:
                               5E 04 C0 0062   339          ADDL2   #4,SP                           ;RESTORE STACK POINTER
                    05AE 8F     BA 0065   340          POPR    #^M<R1,R2,R3,R5,R7,R8,R10>      ;RETURN TO CALLER
                            05 0069   341          RSB                                     ;RETURN TO CALLER
```

```
                     006A   342
                     006A   343 ;
                     006A   344 ; THE PFN WAS NOT WITHIN THE LAST SHARED MEMORY.  CHECK IF THERE IS ANOTHER
                     006A   345 ; SHARED MEMORY TO SEARCH.
                     006A   346 ;
                     006A   347 GET_NXT_SHM:
          54   64    D0  006A   348          MOVL    SHB$L_LINK(R4),R4               ;GET NEXT SH MEM CONTROL BLK
               A3    12  006D   349          BNEQ    10$                             ;BR IF ANOTHER MEM TO SEARCH
                     006F   350
                     006F   351 ;
                     006F   352 ; THE PFN WAS NOT FOUND IN ANY OF THE SHARED MEMORIES.  REPORT FAILURE.
                     006F   353 ;
                     006F   354 NOT_FOUND:
                     006F   355          ASSUME  SS$_NOSUCHSEC LT <^X10000>
          50   0978 8F    3C  006F   356          MOVZWL  #SS$_NOSUCHSEC,R0               ;REPORT FAILURE TO FIND GSD
               EC    11  0074   357          BRB     RSB_HERE                        ;GO RETURN TO CALLER
                     0076   358
                     0076   359          .DSABL  LSB
```

```
                     0076   361          .SBTTL  DECSHMREF/INCSHMREF - MODIFY SHARED MEMORY GSD PTE REF COUNT
                     0076   362 ;++
                     0076   363 ; FUNCTIONAL DESCRIPTION:
                     0076   364 ;
                     0076   365 ;       THIS ROUTINE MODIFIES THE PTE REFERENCE COUNTS IN A SHARED MEMORY
                     0076   366 ;       GLOBAL SECTION DESCRIPTOR.  THERE IS ONE REFERENCE COUNT FOR EACH
                     0076   367 ;       PROCESSOR ON THE SHARED MEMORY.  THE PORT NUMBER OF THE PROCESSOR
                     0076   368 ;       IS THE INDEX TO THE CORRESPONDING REFERENCE COUNT.  THE FIRST
                     0076   369 ;       ENTRY POINT, MMG$DECSHMREF, CAUSES THE COUNT TO BE DECREMENTED
                     0076   370 ;       WHILE THE ENTRY POINT, MMG$INCSHMREF, INCREMENTS THE COUNT.
                     0076   371 ;
                     0076   372 ; CALLING SEQUENCE:
                     0076   373 ;
                     0076   374 ;       BSBW    MMG$DECSHMREF
                     0076   375 ;       BSBW    MMG$INCSHMREF
                     0076   376 ;
                     0076   377 ; INPUT PARAMETERS:
                     0076   378 ;
                     0076   379 ;       R0 - THE NUMBER OF REFERENCES TO BE ADDED OR SUBTRACTED
                     0076   380 ;       R4 - ADDRESS OF THE SHARED MEMORY CONTROL BLOCK
                     0076   381 ;       R6 - ADDRESS OF THE GLOBAL SECTION DESCRIPTOR
                     0076   382 ;
                     0076   383 ; IMPLICIT INPUTS:
                     0076   384 ;
                     0076   385 ;       THE GLOBAL SECTION DESCRIPTOR HAS BEEN INITIALIZED.
                     0076   386 ;
                     0076   387 ; OUTPUT PARAMETERS:
                     0076   388 ;
                     0076   389 ;       NONE
                     0076   390 ;
                     0076   391 ; IMPLICIT OUTPUTS:
                     0076   392 ;
                     0076   393 ;       THE REFERENCE COUNT CORRESPONDING TO THIS PROCESSOR IS UPDATED
                     0076   394 ;       IN THE GSD.
                     0076   395 ;
                     0076   396 ; COMPLETION CODES:
                     0076   397 ;
                     0076   398 ;       NONE
                     0076   399 ;
                     0076   400 ; SIDE EFFECTS:
                     0076   401 ;
                     0076   402 ;       NONE
                     0076   403 ;
                     0076   404 ;--
                     0076   405
                     0076   406 ; ****************************************************************************
                     0076   407 ;
                     0076   408 ; ************** THE FOLLOWING CODE MUST BE RESIDENT ********************
                     0076   409 ;
                00000076    410          .PSECT  $MMGCOD
                     0076   411 ;
                     0076   412 ; ****************************************************************************
                     0076   413
                     0076   414 MMG$DECSHMREF::
 50    50    CE    0076   415          MNEGL   R0,R0                           ;NEGATE REFERENCE COUNT
                     0079   416
                     0079   417 MMG$INCSHMREF::
```

```
                51   DD  0079   418          PUSHL   R1                              ;SAVE REGISTER
          51    15 A4  9A  007B   419          MOVZBL  SHB$B_PORT(R4),R1               ;GET PROCESSOR PORT NUMBER
      74 A641      50  C0  007F   420          ADDL2   R0,GSD$L_PTECNT1(R6)[R1]        ;INCR REF CNT FOR CORRES PROCESSOR
                   0A  19  0084   421          BLSS    10$                             ;BUGCHK IF NEGATIVE REF COUNT
        0C A4      50  C0  0086   422          ADDL2   R0,SHB$L_REFCNT(R4)             ;INCR PORT'S REF COUNT
                       0089   423 ;            BLSS    20$                             ;BUGCHK IF NEGATIVE REF COUNT
                   01  008A   424          NOP                                     ;  (These should be removed
                   01  008B   425          NOP                                     ;  and the BLSS restored after
                       008C   426                                                  ;  the refcnt bug is found.)
            51     8E  D0  008C   427          MOVL    (SP)+,R1                        ;RESTORE REGISTER
                   05  008F   428          RSB                                     ;RETURN TO CALLER
                       0090   429
                       0090   430 10$:      BUG_CHECK       REFCNTNEG,FATAL         ;FATAL ERROR
                       0094   431 20$:      BUG_CHECK       NEGSHBREF,FATAL         ;FATAL ERROR
```

```
0098    433          .SBTTL  ALOSHMPAG - ALLOCATE PAGES GLOBAL SECTION PAGES FROM SHARED MEMORY
0098    434 ;++
0098    435 ; FUNCTIONAL DESCRIPTION:
0098    436 ;
0098    437 ;       THIS ROUTINE ACCEPTS AS INPUT THE SIZE OF THE GLOBAL SECTION TO BE
0098    438 ;       CREATED AND THE ADDRESS OF THE GSD WHICH DESCRIBES THE NUMBER OF
0098    439 ;       NON-CONTIGUOUS PIECES THAT MAY BE ALLOCATED FOR THE SECTION.  IT
0098    440 ;       THEN SEARCHES THE BITMAP IN THE SHARED MEMORY COMMON DATA PAGE
0098    441 ;       FOR THE NUMBER OF PAGES NEEDED AND STORES THE PAGES ALLOCATED IN
0098    442 ;       THE GSD, ALSO CLEARING THE CORRESPONDING BIT IN THE BITMAP.
0098    443 ;
0098    444 ;       THE BITMAP IS LOCKED AGAINST ACCESS BY ANY OTHER PROCESSOR DURING
0098    445 ;       THE ALLOCATION.
0098    446 ;
0098    447 ; CALLING SEQUENCE:
0098    448 ;
0098    449 ;       BSBW    MMG$ALOSHMPAG
0098    450 ;
0098    451 ; INPUT PARAMETERS:
0098    452 ;
0098    453 ;       R4 - ADDRESS OF THE SHARED MEMORY CONTROL BLOCK
0098    454 ;       R6 - ADDRESS OF THE GLOBAL SECTION DESCRIPTOR
0098    455 ;       R7 - COUNT OF PAGES TO BE ALLOCATED
0098    456 ;
0098    457 ; IMPLICIT INPUTS:
0098    458 ;
0098    459 ;       THE SHARED MEMORY BITMAP HAS BEEN INITIALIZED.  IT CONTAINS A BIT
0098    460 ;       FOR EACH PAGE TO BE USED FOR GLOBAL SECTIONS.  IF THE BIT IS SET,
0098    461 ;       THEN THE PAGE IS AVAILABLE FOR ALLOCATION.  IF THE BIT IS CLEAR, THE
0098    462 ;       PAGE IS EITHER (1) IN USE BY ANOTHER GLOBAL SECTION OR (2) IS A BAD
0098    463 ;       PAGE.  THE GLOBAL SECTION DESCRIPTOR CONTAINS THE NUMBER OF
0098    464 ;       PIECES THAT THE SECTION MAY BE BROKEN INTO.
0098    465 ;
0098    466 ; OUTPUT PARAMETERS:
0098    467 ;
0098    468 ;       NONE
0098    469 ;
0098    470 ; IMPLICIT OUTPUTS:
0098    471 ;
0098    472 ;       THE GSD DESCRIBES THE PAGES ALLOCATED FOR THE SECTION AND THE
0098    473 ;       CORRESPONDING BITS ARE CLEARED IN THE BITMAP.
0098    474 ;
0098    475 ; COMPLETION CODES:
0098    476 ;
0098    477 ;       SS$_NORMAL - ALL PAGES FOR SECTION SUCCESSFULLY ALLOCATED
0098    478 ;       SS$_INSFMEM - NOT ENOUGH FREE SHARED MEMORY
0098    479 ;       SS$_INTERLOCK - UNABLE TO ACQUIRE BITMAP LOCK
0098    480 ;
0098    481 ; SIDE EFFECTS:
0098    482 ;
0098    483 ;       IF SUFFICIENT PAGES CANNOT BE FOUND, THE ROUTINE TO SCAN AND
0098    484 ;       FREE GSD'S AND DATA PAGES IS CALLED.
0098    485 ;
0098    486 ;--
0098    487
0098    488 ; ****************************************************************************
0098    489 ;
```

```
                                0098    490 ; ****************** THE FOLLOWING CODE MAY BE PAGED ******************
                                0098    491 ;
                         00000068       492         .PSECT  Y$EXEPAGED
                                0068    493 ;
                                0068    494 ; ****************************************************************************
                                0068    495
                                0068    496 MMG$ALOSHMPAG::
                                0068    497         .ENABLE LSB
                     0F3E 8F    BB   0068    498         PUSHR   #^M<R1,R2,R3,R4,R5,R8,R9,R10,R11> ;SAVE REGISTERS
                     50   01    9A   006C    499 3$:     MOVZBL  #SHD$V_BITMAPLCK,R0             ;BIT NUMBER OF LOCK REQUESTED
                        06A2    30   006F    500         BSBW    MMG$SHMTXLK                     ;GET SHM MUTEX AND BIT LOCK
                                0072    501                                                     ;R5=SHD ADR
                       24 50    E9   0072    502         BLBC    R0,LOCK_ERR                     ;BR IF UNABLE TO GET BITMAP LOCK
                009E C5 15 A4   90   0075    503         MOVB    SHB$B_PORT(R4),SHD$B_BITMAPLCK(R5) ;SET BITMAP LOCK OWNER
                     58   57    D0   007B    504         MOVL    R7,R8                           ;COUNT OF PAGES REQUESTED
                     54   04    9A   007E    505         MOVZBL  #GSD$C_PFNBASMAX,R4             ;COUNT OF PFN BASES ALLOWED
                  5B   54 A6    9E   0081    506         MOVAB   GSD$L_BASPFN1(R6),R11           ;GET ADR OF FIRST BASE IN GSD
                51   0C A5 55    C1   0085    507         ADDL3   R5,SHD$L_GSBITMAP(R5),R1        ;VA OF GS BITMAP
                50   10 A5 07    C1   008A    508         ADDL3   #7,SHD$L_GSPAGCNT(R5),R0        ;COMPUTE # BITMAP BYTES, INCL
                50   50   FD 8F   78   008F    509         ASHL    #-3,R0,R0                       ; THE LAST PARTIALLY USED BYTE
                           06    12   0094    510         BNEQ    NXT_PIECE                       ;BR TO ALLOCATE PAGES
                        00BE    31   0096    511         BRW     INSF_MEM                        ;BR IF NO GS PAGES AVAILABLE
                                0099    512
                                0099    513 LOCK_ERR:
                        00B6    31   0099    514         BRW     100$                            ;RETURN TO CALLER
                                009C    515 ;
                                009C    516 ; R0 = LENGTH IN BYTES OF BITMAP LEFT TO SEARCH
                                009C    517 ; R1 = BYTE ADDRESS IN BITMAP TO START SEARCHING
                                009C    518 ; R2 = BYTE ADDRESS IN BITMAP OF FIRST SET BIT
                                009C    519 ; R3 = BIT NUMBER OF FIRST SET BIT
                                009C    520 ; R4 = COUNT OF PFN BASES LEFT TO USE IN GSD
                                009C    521 ; R5 = SHARED MEMORY DATA PAGE ADDRESS
                                009C    522 ; R6 = GLOBAL SECTION DESCRIPTOR ADDRESS
                                009C    523 ; R7 = NUMBER OF PAGES REQUESTED
                                009C    524 ; R8 = NUMBER OF PAGES MORE NEEDED
                                009C    525 ; R9 = BYTE ADDRESS IN BITMAP OF FIRST CLEAR BIT
                                009C    526 ; R10 = BIT NUMBER OF FIRST CLEAR BIT
                                009C    527 ; R11 = BYTE ADDRESS IN GLOBAL SECTION DESCRIPTOR FOR NEXT PFN BASE
                                009C    528 ;
                                009C    529
                                009C    530 ;
                                009C    531 ; THE BITMAP CONTAINS ONE BIT FOR EACH PAGE OF SHARED MEMORY ALLOCATED FOR
                                009C    532 ; GLOBAL SECTION PAGE USAGE.  A SET BIT INDICATES THAT THE PAGE MAY BE
                                009C    533 ; ALLOCATED FOR USE.  A CLEAR BIT INDICATES THAT THE PAGE IS ALREADY BEING
                                009C    534 ; USED OR IS A BAD PAGE.
                                009C    535 ;
                                009C    536 ; THE BITMAP IS SEARCHED FOR SEGMENTS OF CONTIGUOUS BITS THAT ARE SET.
                                009C    537 ; EACH PIECE OF BITMAP THAT CONTAINS CONTIGUOUS SET BITS IS DESCRIBED VIA
                                009C    538 ; FOUR REGISTERS:
                                009C    539 ;       R2 = ADDRESS OF BITMAP BYTE CONTAINING FIRST SET BIT
                                009C    540 ;       R3 = BIT NUMBER OF FIRST SET BIT WITHIN THE BYTE
                                009C    541 ;       R9 = ADDRESS OF BITMAP BYTE CONTAINING FIRST CLEAR BIT
                                009C    542 ;       R10= BIT NUMBER OF FIRST CLEAR BIT WITHIN THE BYTE
                                009C    543 ;
                                009C    544 ; THE SEARCH OF THE BITMAP FOR THESE PIECES WORKS AS FOLLOWS:
                                009C    545 ;       1. FIND THE FIRST BYTE WITH AT LEAST ONE BIT SET   (SKPC #0)
                                009C    546 ;       2. FIND THE BIT NUMBER OF THE FIRST SET BIT        (FFS)
```

```
                                009C    547 ;       3. FIND THE FIRST CLEAR BIT FOLLOWING THE SET BIT
                                009C    548 ;               A. FIRST CHECK IF THE CLEAR BIT IS IN THE
                                009C    549 ;                  SAME BYTE AS THE SET BIT; IF SO THEN    (FFC)
                                009C    550 ;                  THE PIECE IS FOUND                      (BRB GOT_PIECE)
                                009C    551 ;               B. SKIP THE BYTE CONTAINING THE FIRST SET
                                009C    552 ;                  BIT (BY RESETTING R0 AND R1)
                                009C    553 ;               C. FIND THE FIRST BYTE THAT HAS AT LEAST
                                009C    554 ;                  ONE BIT CLEAR                           (SKPC #-1)
                                009C    555 ;               D. FIND THE BIT NUMBER OF THE FIRST CLEAR
                                009C    556 ;                  BIT; THE PIECE IS FOUND                 (FFC)
                                009C    557 ;
                                009C    558 NXT_PIECE:
      61    50    00    3B    009C    559         SKPC    #0,R0,(R1)                      ;FIND NEXT BYTE WITH A BIT SET
                  68    13    00A0    560         BEQL    45$                             ;BR ON NO MORE BITS SET
53    61    08    00    EA    00A2    561         FFS     #0,#8,(R1),R3                   ;FIND BIT # OF FIRST BIT SET
            52    51    D0    00A7    562         MOVL    R1,R2                           ;SAVE ADR OF BYTE WITH BIT SET
                                00AA    563 ;
                                00AA    564 ; NOW FIND THE FIRST CLEAR BIT WHICH INDICATES THE END OF THIS PIECE.
                                00AA    565 ;
                                00AA    566 FIND_PIECE_END:
      5A    08    53    C3    00AA    567         SUBL3   R3,#8,R10                       ;GET # BITS LEFT IN BYTE
5A    61    5A    53    EB    00AE    568         FFC     R3,R10,(R1),R10                 ;IS THERE A BIT CLEAR IN BYTE?
                  05    13    00B3    569         BEQL    15$                             ;BR ON REST OF BITS SET IN BYTE
            59    52    D0    00B5    570         MOVL    R2,R9                           ;SET ADR OF BYTE W/ NXT CLR BIT
                  17    11    00B8    571         BRB     GOT_PIECE                       ;GO SEE IF CAN USE THIS PIECE
                  50    D7    00BA    572 15$:    DECL    R0                              ;SKIP PAST THE BYTE WHICH
                  51    D6    00BC    573         INCL    R1                              ;CONTAINS THE FIRST SET BIT
  61    50    FF    8F    3B    00BE    574         SKPC    #-1,R0,(R1)                     ;LOOK FOR NEXT CLR BIT IN BITMAP
                  07    13    00C3    575         BEQL    ALL_REST_SET                    ;BR IF ALL OF BITMAP SET
5A    61    08    00    EB    00C5    576         FFC     #0,#8,(R1),R10                  ;FIND BIT # OF FIRST CLR BIT
                  02    11    00CA    577         BRB     20$                             ;GO SET BYTE ADR
                                00CC    578 ;
                                00CC    579 ; THIS CODE CAN BE ENHANCED HERE.  IT DOES NOT TAKE INTO ACCOUNT THE LAST
                                00CC    580 ; BYTE OF BITMAP IF THE ENTIRE BYTE IS NOT USED.  A PIECE THAT EXTENDS TO
                                00CC    581 ; THE END OF THE BITMAP WILL HAVE POINTERS THAT POINT TO THE NEXT BIT
                                00CC    582 ; PAST THE END OF THE BITMAP.
                                00CC    583 ;
                                00CC    584 ALL_REST_SET:
            5A    D4    00CC    585         CLRL    R10                             ;SAVE BIT # OF FIRST CLR BIT
      59    51    D0    00CE    586 20$:    MOVL    R1,R9                           ;SAVE ADR OF BYTE WITH BIT CLR
                                00D1    587 ;
                                00D1    588 ; ONCE A CONTIGUOUS PIECE OF BITMAP CONTAINING SET BITS IS FOUND, THE
                                00D1    589 ; FOLLOWING INFORMATION IS IN THE REGISTERS:
                                00D1    590 ;       R2 = ADDRESS OF THE BYTE IN THE BITMAP CONTAINING THE FIRST SET BIT
                                00D1    591 ;       R3 = BIT NUMBER OF THE FIRST SET BIT
                                00D1    592 ;       R9 = ADDRESS OF THE BYTE IN THE BITMAP CONTAINING THE FIRST CLEAR BIT
                                00D1    593 ;       R10= BIT NUMBER OF THE FIRST CLEAR BIT
                                00D1    594 ; THE NEXT STEP IS TO COMPUTE THE NUMBER OF PAGES CONTAINED IN THIS PIECE
                                00D1    595 ; AND THE RELATIVE PFN OF THE FIRST PAGE.
                                00D1    596 ;
                                00D1    597 GOT_PIECE:
59    59    52    C3    00D1    598         SUBL3   R2,R9,R9                        ;GET # BYTES WITH ALL BITS SET
                  59    D7    00D5    599         DECL    R9                              ;CORRECT SUBTRACTION CNT
            59    08    C4    00D7    600         MULL2   #8,R9                           ;GET # PAGES AVAILABLE
            59    5A    C0    00DA    601         ADDL2   R10,R9                          ;ADD # PAGES BEFORE CLR BIT
7E    08    53    C3    00DD    602         SUBL3   R3,#8,-(SP)                     ;GET # PAGES AFTER FIRST SET BIT
            59    8E    C0    00E1    603         ADDL2   (SP)+,R9                        ;GET TOTAL # PAGES IN PIECE
```

```
                52   0C A5   C2   00E4    604          SUBL2   SHD$L_GSBITMAP(R5),R2           ;GET BYTE OFFSET TO 1ST SET BIT
                  52   55   C2   00E8    605          SUBL2   R5,R2                           ;MINUS GS PTR AND SHD ADR
                  52   08   C4   00EB    606          MULL2   #8,R2                           ;COMPUTE RELATIVE BIT # OF 1ST
                  52   53   C0   00EE    607          ADDL2   R3,R2                           ;SET BIT FROM START OF BITMAP
                               00F1    608 ;
                               00F1    609 ; NOW THE REGISTERS CONTAIN:
                               00F1    610 ;       R2 = RELATIVE PFN OF THE FIRST PAGE IN THIS PIECE (FROM START OF BITMAP)
                               00F1    611 ;       R7 = TOTAL NUMBER OF PAGES REQUESTED BY CALLER
                               00F1    612 ;       R8 = NUMBER OF PAGES STILL NEEDED TO FULFILL REQUEST OF CALLER
                               00F1    613 ;            (THE PIECES ALREADY FOUND HAVE DECREMENTED THIS VALUE FROM THE
                               00F1    614 ;             VALUE CONTAINED IN R7.  REMEMBER A GLOBAL SECTION MAY BE
                               00F1    615 ;             ALLOCATED IN UP TO #GSD$C_PFNBASMAX PIECES.)
                               00F1    616 ;       R9 = NUMBER OF CONTIGUOUS PAGES IN THIS PIECE
                               00F1    617 ;       R1 = ADDRESS OF BYTE CONTAINING FIRST CLEAR BIT
                               00F1    618 ;       R10= BIT NUMBER OF FIRST CLEAR BIT
                               00F1    619 ;
                59   57   D1   00F1    620          CMPL    R7,R9                           ;DOES ALL GS FIT IN THIS PIECE?
                     39   15   00F4    621          BLEQ    FOUND_1_PIECE                   ;YES, GO USE IT
                     58   D5   00F6    622          TSTL    R8                              ;MORE DISCONTIG PAGES NEEDED?
                     0E   15   00F8    623          BLEQ    40$                             ;BR ON NO
                  02 54   F5   00FA    624          SOBGTR  R4,30$                          ;USE ONLY # OF BASES ALLOWED
                     09   11   00FD    625          BRB     40$                             ;BR IF > MAX BASES ALLOWED
                58   59   C2   00FF    626 30$:     SUBL2   R9,R8                           ;USE ALL THIS PIECE
                8B   52   D0   0102    627          MOVL    R2,(R11)+                       ;SET THIS PFN BASE IN GSD
                8B   59   D0   0105    628          MOVL    R9,(R11)+                       ;SET SIZE OF PIECE IN GSD
                               0108    629 ;
                               0108    630 ; NOW FIND THE NEXT PIECE OF THE BITMAP WITH PAGES AVAILABLE FOR ALLOCATION.
                               0108    631 ; THIS IS DONE BY REPEATING THE SEARCH PROCESS ABOVE.  HOWEVER, THE "FIRST"
                               0108    632 ; SET BIT MAY BE WITHIN THE BYTE CONTAINING THE "LAST" CLEAR BIT.  CHECK FOR
                               0108    633 ; THIS FIRST.  IF THE NEW "FIRST" SET BIT IS WITHIN THIS BYTE, THEN CONTINUE
                               0108    634 ; ISOLATING THE PIECE BY FINDING THE NEXT CLEAR BIT (BRB FIND_PIECE_END).
                               0108    635 ; IF REST OF BITS IN BYTE ARE ALSO CLEAR, THEN UPDATE THE BITMAP SEARCH
                               0108    636 ; POINTER AND LENGTH (R1,R0) TO START THE SEARCH PAST THIS BYTE (BRB NO_BIT_SET)
                               0108    637 ; AND CONTINUE IN THE SEARCH LOOP (BRB NXT_PIECE).
                               0108    638 ;
                     50   D5   0108    639 40$:     TSTL    R0                              ;ANY MORE BITMAP TO SEARCH?
                     19   13   010A    640 45$:     BEQL    END_OF_BITMAP                   ;BR IF NO MORE TO SEARCH
                53   08   5A   C3   010C    641          SUBL3   R10,#8,R3                       ;GET # BITS AFTER CLR BIT
              53 61   53   5A   EA   0110    642          FFS     R10,R3,(R1),R3                  ;IS THERE A SET BIT?
                     05   13   0115    643          BEQL    NO_BIT_SET                      ;BR ON NO, GO USE NEXT BYTE
                52   51   D0   0117    644          MOVL    R1,R2                           ;SAVE BYTE ADR OF SET BIT
                     8E   11   011A    645          BRB     FIND_PIECE_END                  ;GO FIND THE END OF THIS PIECE
                               011C    646 NO_BIT_SET:
                     51   D6   011C    647          INCL    R1                              ;POINT TO NEXT BYTE OF BITMAP
                     50   D7   011E    648          DECL    R0                              ;ONE LESS BYTE TO SEARCH
                     03   15   0120    649          BLEQ    END_OF_BITMAP                   ;BR ON NO MORE BITMAP TO SEARCH
              FF77   31   0122    650          BRW     NXT_PIECE                       ;GO FIND NEXT PIECE
                               0125    651 ;
                               0125    652 ; NO ONE CONTIGUOUS PIECE WAS LARGE ENOUGH TO HOLD THIS GLOBAL SECTION.
                               0125    653 ; R8 CONTAINS THE NUMBER OF PAGES STILL NEEDED TO HOLD THE GLOBAL SECTION.  IF
                               0125    654 ; IT IS EQUAL TO ZERO, THEN THE SECTION WAS EXACTLY CONTAINED IN SOME NUMBER
                               0125    655 ; OF PIECES OF SHARED MEMORY.  IF IT IS LESS THAN ZERO, THEN THE LAST PIECE OF
                               0125    656 ; SHARED MEMORY USED, WAS LARGER THAN NEEDED FOR THE SECTION.  IF IT IS GREATER
                               0125    657 ; THAN ZERO, THEN THE FIRST N PIECES FOUND WERE NOT LARGE ENOUGH TO HOLD ALL OF
                               0125    658 ; THE GLOBAL SECTION (WHERE N IS THE NUMBER OF PFN BASES IN THE GSD).
                               0125    659 ;
                               0125    660 END_OF_BITMAP:
```

```
                58   D5  0125   661          TSTL    R8                              ;MORE PAGES NEEDED?
                2E   14  0127   662          BGTR    INSF_MEM                        ;BR ON YES, FRAGMENTED MEMORY
          FC AB     58   C0  0129   663          ADDL2   R8,-4(R11)                      ;SET ACTUAL SIZE OF PIECE NEEDED
                14   11  012D   664          BRB     CLR_BITMAP                      ;BR AS GOT PAGES IN PIECES
                          012F   665 FOUND_1_PIECE:
          50    54 A6   9E  012F   666          MOVAB   GSD$L_BASPFN1(R6),R0            ;ADR OF FIRST PFN BASE IN GSD
             51    04   9A  0133   667          MOVZBL  #GSD$C_PFNBASMAX,R1             ;COUNT OF PFN BASES ALLOWED
                          0136   668
                          0136   669          ASSUME GSD$L_BASCNT1 EQ <GSD$L_BASPFN1+4>
                          0136   670
             80   52   D0  0136   671          MOVL    R2,(R0)+                        ;SET BASE PFN IN GSD
             80   57   D0  0139   672          MOVL    R7,(R0)+                        ;SET SIZE OF SECTION IN GSD
                51   D7  013C   673          DECL    R1                              ;ANY MORE BASES TO SET?
                80   7C  013E   674 50$:     CLRQ    (R0)+                           ;CLEAR BASE AND COUNT
             FB 51   F5  0140   675          SOBGTR  R1,50$                          ;REPEAT TILL ALL BASES CLEAR
                          0143   676
                          0143   677 CLR_BITMAP:
             FEC3   30  0143   678          BSBW    MMG$CLR_BITMAP                  ;CLEAR CORRESPONDING BITMAP BITS
                          0146   679          ASSUME  SS$_NORMAL LT <^X100>
             50   01   9A  0146   680 90$:     MOVZBL  #SS$_NORMAL,R0                  ;REPORT SUCCESS
       00 009F C5   01   E7  0149   681          BBCCI   #SHD$V_BITMAPLCK,SHD$B_FLAGS(R5),98$ ;RELEASE BITMAP LOCK
                05FF   30  014F   682 98$:     BSBW    MMG$SHMTXULK                    ;RELEASE SHM MUTEX
             0F3E 8F   BA  0152   683 100$:    POPR    #^M<R1,R2,R3,R4,R5,R8,R9,R10,R11> ;RESTORE REGISTERS
                05   0156   684          RSB
                          0157   685
                          0157   686 INSF_MEM:
       00 009F C5   01   E7  0157   687          BBCCI   #SHD$V_BITMAPLCK,SHD$B_FLAGS(R5),200$ ;RELEASE BITMAP LOCK
                05F1   30  015D   688 200$:    BSBW    MMG$SHMTXULK                    ;RELEASE SHM MUTEX
          54    0C AE   D0  0160   689          MOVL    <3*4>(SP),R4                    ;GET ADDRESS OF SHB
                0098   30  0164   690          BSBW    MMG$FREEGSD                     ;FREE UNOWNED PAGES AND GSD'S
             03 50   E9  0167   691          BLBC    R0,210$                         ;BR IF NOTHING WAS FREED
                FEFF   31  016A   692          BRW     3$                              ;TRY AGAIN TO ALLOCATE PAGES
                          016D   693          ASSUME  SS$_INSFMEM LT <^X10000>
    50    0124 8F   3C  016D   694 210$:    MOVZWL  #SS$_INSFMEM,R0                 ;REPORT INSUFICIENT MEMORY
                DE   11  0172   695          BRB     100$                            ;RETURN TO USER
                          0174   696          .DSABL LSB
```

```
                     0174   698          .SBTTL  ALOSHMGSD - ALLOCATE SHARED MEMORY GLOBAL SECTION DESCRIPTOR
                     0174   699 ;++
                     0174   700 ; FUNCTIONAL DESCRIPTION:
                     0174   701 ;
                     0174   702 ;       THIS ROUTINE ALLOCATES A GLOBAL SECTION DESCRIPTOR BLOCK FROM THE
                     0174   703 ;       TABLE OF GSD'S IN A SPECIFIC SHARED MEMORY.  IT ACCEPTS AS INPUT THE
                     0174   704 ;       ADDRESS OF THE SHARED MEMORY CONTROL BLOCK.  IT OUTPUTS THE ADDRESS
                     0174   705 ;       OF THE GSD ALLOCATED AND A SUCCESS CODE OR IF NO GSD IS AVAILABLE,
                     0174   706 ;       AN ERROR CODE.  THE GSD IS LOCKED FOR MODIFICATION.
                     0174   707 ;
                     0174   708 ; CALLING SEQUENCE:
                     0174   709 ;
                     0174   710 ;       BSBW    MMG$ALOSHMGSD
                     0174   711 ;
                     0174   712 ; INPUT PARAMETERS:
                     0174   713 ;
                     0174   714 ;       R4 - ADDRESS OF THE SHARED MEMORY CONTROL BLOCK
                     0174   715 ;
                     0174   716 ; IMPLICIT INPUTS:
                     0174   717 ;
                     0174   718 ;       THE TABLE OF GLOBAL SECTION DESCRIPTORS IN SHARED MEMORY HAS BEEN
                     0174   719 ;       INITIALIZED.  THE CONSTANT FIELDS IN THESE DESCRIPTORS ARE ALREADY
                     0174   720 ;       INITIALIZED, ALSO.  THE SHARED MEMORY CONTROL BLOCK AND COMMON DATA
                     0174   721 ;       PAGE HAVE BEEN INITIALIZED BY CONNECTING TO THE SHARED MEMORY.
                     0174   722 ;
                     0174   723 ; OUTPUT PARAMETERS:
                     0174   724 ;
                     0174   725 ;       R0 - RETURN STATUS CODE
                     0174   726 ;       R6 - ADDRESS OF THE GLOBAL SECTION DESCRIPTOR ALLOCATED, IF SUCCESSFUL
                     0174   727 ;
                     0174   728 ; IMPLICIT OUTPUTS:
                     0174   729 ;
                     0174   730 ;       THE CONSTANT GSD FIELDS ARE ALREADY INITIALIZED AND THE GSD IS LOCKED
                     0174   731 ;       BY THE ALLOCATING PROCESSOR.
                     0174   732 ;
                     0174   733 ; COMPLETION CODES:
                     0174   734 ;
                     0174   735 ;       SS$_NORMAL - ALL PAGES FOR SECTION SUCCESSFULLY ALLOCATED
                     0174   736 ;       SS$_GSDFULL - NO GSD AVAILABLE FOR ALLOCATION
                     0174   737 ;       SS$_EXPORTQUOTA - PORT QUOTA EXCEEDED
                     0174   738 ;
                     0174   739 ; SIDE EFFECTS:
                     0174   740 ;
                     0174   741 ;       THE GSD IS LOCKED AND NO OTHER PROCESS ON ANY PROCESSOR MAY ACCESS IT.
                     0174   742 ;
                     0174   743 ;       IF NO GSD CAN BE FOUND, FREEGSD IS CALLED TO SCAN FOR GSD'S AND DATA
                     0174   744 ;       PAGES THAT CAN BE FREED.
                     0174   745 ;
                     0174   746 ;--
                     0174   747
                     0174   748 MMG$ALOSHMGSD::
                     0174   749          .ENABLE LSB
              26 BB  0174   750          PUSHR   #^M<R1,R2,R5>                   ;SAVE REGISTERS
        55 04 A4 D0  0176   751 3$:       MOVL    SHB$L_DATAPAGE(R4),R5           ;GET ADR OF COMMON DATA PAGE
        52 15 A4 9A  017A   752          MOVZBL  SHB$B_PORT(R4),R2               ;GET PORT NUMBER
3C A542  FFFF 8F 58  017E   753          ADAWI   #-1,SHD$W_GSDQUOTA(R5)[R2]      ;ALLOC QUOTA FOR 1 CREATE
              6C 19  0185   754          BLSS    NO_QUOTA                        ;BR IF NO QUOTA AVAILABLE
```

```
56   55   04 A5  C1  0187   755          ADDL3   SHD$L_GSDPTR(R5),R5,R6          ;ADR OF FIRST GSD
               23  11  018C   756          BRB     20$                             ;GO SEE IF GSD IS UNUSED
          50   01  9A  018E   757 10$:     MOVZBL  #1,R0                           ;ONE REF COUNT TO LOCK ENTRY
       00000076'EF  16  0191   758          JSB     MMG$DECSHMREF                   ;RELEASE LOCK ON GSD ENTRY
     50   08 A6  3C  0197   759          MOVZWL  GSD$W_SIZE(R6),R0               ;GET SIZE OF ONE GSD
          56   50  C0  019B   760          ADDL2   R0,R6                           ;GET ADR OF NEXT GSD
     51   18 A5  3C  019E   761          MOVZWL  SHD$W_GSDMAX(R5),R1             ;GET MAX # OF GSD'S IN TABLE
          50   51  C4  01A2   762          MULL2   R1,R0                           ;GET SIZE OF GSD TABLE IN BYTES
          50   55  C0  01A5   763          ADDL2   R5,R0                           ;ADD IN BASE VA FOR DATA PAGE
     50   04 A5  C0  01A8   764          ADDL2   SHD$L_GSDPTR(R5),R0             ;ADD ADR OF START OF GSD TABLE
          50   56  D1  01AC   765          CMPL    R6,R0                           ;PAST END OF GSD TABLE?
               37  1E  01AF   766          BGEQU   NO_FREE_GSD                     ;BR IF PAST END OF TABLE
          50   01  9A  01B1   767 20$:     MOVZBL  #1,R0                           ;ONE REF COUNT TO LOCK ENTRY
       00000079'EF  16  01B4   768          JSB     MMG$INCSHMREF                   ;LOCK ENTRY IN SHM GSD TBL
       D0 66   01  E0  01BA   769          BBS     #GSD$V_LOCKED,GSD$L_GSDFL(R6),10$ ;BR IF GSD BEING MODIFIED
       CC 66   00  E0  01BE   770          BBS     #GSD$V_VALID,GSD$L_GSDFL(R6),10$ ;BR IF GSD IS IN USE
       C8 66   01  E6  01C2   771          BBSSI   #GSD$V_LOCKED,GSD$L_GSDFL(R6),10$ ;BR IF GSD BEING MODIFED
            0C A4  D6  01C6   772          INCL    SHB$L_REFCNT(R4)                ;ONE FOR GSD OWNED BY THIS PORT
     50   54 A6  9E  01C9   773          MOVAB   GSD$L_BASPFN1(R6),R0            ;ADR OF 1ST BASE PFN & CNT PAIR
          51   04  9A  01CD   774          MOVZBL  #GSD$C_PFNBASMAX,R1             ;# BASE PFN'S ALLOWED IN GSD
               80  7C  01D0   775 30$:     CLRQ    (R0)+                           ;CLEAR ONE BASE PFN & CNT PAIR
          FB   51  F5  01D2   776          SOBGTR  R1,30$                          ;REPEAT FOR ALL BASES
            53 A6  94  01D5   777          CLRB    GSD$B_DELETPORT(R6)             ;CLEAR THE DELETOR PORT #
   52 A6   15 A4  90  01D8   778          MOVB    SHB$B_PORT(R4),GSD$B_CREATPORT(R6) ;SET CREATOR PROCESSOR PORT #
   50 A6   15 A4  90  01DD   779          MOVB    SHB$B_PORT(R4),GSD$B_LOCK(R6)   ;SET # OF PORT HOLDING GSD LOCK
                      01E2   780          ASSUME  SS$_NORMAL LT <^X100>
          50   01  9A  01E2   781          MOVZBL  #SS$_NORMAL,R0                  ;REPORT SUCCESSFUL ALLOCATION
               26  BA  01E5   782 50$:     POPR    #^M<R1,R2,R5>                   ;RESTORE REGISTERS
               05      01E7   783          RSB
                      01E8   784
                      01E8   785 NO_FREE_GSD:
          15   10      01E8   786          BSBB    MMG$FREEGSD                     ;FREE ABANDONED GSD'S AND PAGES
          89   50  E8  01EA   787          BLBS    R0,3$                           ;BR IF RESOURCES WERE FREED
                      01ED   788          ASSUME  SS$_GSDFULL LT <^X100>
     50   CC 8F  9A  01ED   789          MOVZBL  #SS$_GSDFULL,R0                 ;REPORT NO GSD TO BE ALLOCATED
               05  11  01F1   790          BRB     60$                             ;GO RETURN QUOTA ALLOCATED
                      01F3   791
                      01F3   792 NO_QUOTA:
   50    03AC 8F  3C  01F3   793          MOVZWL  #SS$_EXPORTQUOTA,R0             ;REPORT NO QUOTA AVAILABLE
   3C A542   01  58  01F8   794 60$:     ADAWI   #1,SHD$W_GSDQUOTA(R5)[R2]       ;RETURN QUOTA ALLOCATED
               E6  11  01FD   795          BRB     50$                             ;RETURN ERROR CODE TO CALLER
                      01FF   796          .DSABL LSB
```

```
                     01FF    798          .SBTTL  FREEGSD - FREE LOST SHARED MEMORY GLOBAL SECTION DESCRIPTORS
                     01FF    799 ;++
                     01FF    800 ; FUNCTIONAL DESCRIPTION:
                     01FF    801 ;
                     01FF    802 ;       THIS ROUTINE SCANS THE GLOBAL SECTION DESCRIPTOR BLOCKS IN THE
                     01FF    803 ;       TABLE OF GSD'S IN A SPECIFIC SHARED MEMORY.  IT FREES ANY BLOCKS
                     01FF    804 ;       THAT WERE CREATED BY A PROCESSOR THAT HAS BEEN REBOOTED AND ARE
                     01FF    805 ;       NO LONGER ACCESSED BY ANY PROCESSOR.
                     01FF    806 ;
                     01FF    807 ; CALLING SEQUENCE:
                     01FF    808 ;
                     01FF    809 ;       BSBW    MMG$FREEGSD
                     01FF    810 ;
                     01FF    811 ; INPUT PARAMETERS:
                     01FF    812 ;
                     01FF    813 ;       R4 - ADDRESS OF THE SHARED MEMORY CONTROL BLOCK
                     01FF    814 ;       R5 - ADDRESS OF THE SHARED MEMORY COMMON DATA PAGE
                     01FF    815 ;
                     01FF    816 ; IMPLICIT INPUTS:
                     01FF    817 ;
                     01FF    818 ;       THE TABLE OF GLOBAL SECTION DESCRIPTORS IN SHARED MEMORY HAS BEEN
                     01FF    819 ;       INITIALIZED.  THE CONSTANT FIELDS IN THESE DESCRIPTORS ARE ALREADY
                     01FF    820 ;       INITIALIZED.  THE SHARED MEMORY CONTROL BLOCK AND COMMON DATA
                     01FF    821 ;       PAGE HAVE BEEN INITIALIZED BY CONNECTING TO THE SHARED MEMORY.
                     01FF    822 ;
                     01FF    823 ; OUTPUT PARAMETERS:
                     01FF    824 ;
                     01FF    825 ;       NONE
                     01FF    826 ;
                     01FF    827 ; IMPLICIT OUTPUTS:
                     01FF    828 ;
                     01FF    829 ;       NONE
                     01FF    830 ;
                     01FF    831 ; COMPLETION CODES:
                     01FF    832 ;
                     01FF    833 ;       R0 - RETURN STATUS CODE
                     01FF    834 ;                1 IF RESOURCES WERE MADE AVAILABLE
                     01FF    835 ;                0 OTHERWISE
                     01FF    836 ;
                     01FF    837 ; SIDE EFFECTS:
                     01FF    838 ;
                     01FF    839 ;       GSD'S MAY BE MADE AVAILABLE.  THE FREE PAGE BITMAP IS UPDATED.
                     01FF    840 ;       R1,R2,R3 ARE DESTROYED.
                     01FF    841 ;
                     01FF    842 ;--
                     01FF    843
                     01FF    844 MMG$FREEGSD::
                     01FF    845          .ENABLE LSB
                  56 DD 01FF    846          PUSHL   R6                              ;SAVE REGISTERS
                  7E D4 0201    847          CLRL    -(SP)                           ;ANTICIPATE FINDING NOTHING
56    55    04 A5 C1 0203    848          ADDL3   SHD$L_GSDPTR(R5),R5,R6          ;ADR OF FIRST GSD
      51    18 A5 3C 0208    849          MOVZWL  SHD$W_GSDMAX(R5),R1             ;GET # OF GSD'S IN TABLE
               55 11 020C    850          BRB     70$                             ;BEGIN GSD SCAN
      4A 66    00 E1 020E    851 10$:     BBC     #GSD$V_VALID,GSD$L_GSDFL(R6),60$ ;BR IF GSD IS NOT IN USE
      46 66    01 E0 0212    852          BBS     #GSD$V_LOCKED,GSD$L_GSDFL(R6),60$ ;BR IF GSD BEING MODIFIED
      42 66    02 E1 0216    853          BBC     #GSD$V_DELPEND,GSD$L_GSDFL(R6),60$ ;BR IF DELETE NOT PENDING
            52 A6 95 021A    854          TSTB    GSD$B_CREATPRT(R6)              ;NON-EXISTENT CREATOR?
```

```
                3D   18   021D   855          BGEQ    60$                                     ;BR IF CREATOR VALID
          39 66    01   E6   021F   856          BBSSI   #GSD$V_LOCKED,GSD$L_GSDFL(R6),60$ ;BR IF GSD BEING MODIFIED
          52   51  A6   9A   0223   857          MOVZBL  GSD$B_PROCCNT(R6),R2                ;NUMBER OF REF COUNTS TO CHECK
          50   74  A6   DE   0227   858          MOVAL   GSD$L_PTECNT1(R6),R0                ;ADDRESS OF FIRST REF COUNT
                80   D5   022B   859 20$:     TSTL    (R0)+                               ;GSD STILL IN USE?
                29   12   022D   860          BNEQ    40$                                 ;BR IF STILL IN USE
             F9 52   F5   022F   861          SOBGTR  R2,20$                              ;ITERATE OVER ALL PORTS
          50    01   9A   0232   862          MOVZBL  #SHD$V_BITMAPLCK,R0                 ;NUMBER OF BIT TO LOCK
            04DC   30   0235   863          BSBW    MMG$SHMTXLK                         ;ACQUIRE MUTEX AND LOCK BIT
           1D 50   E9   0238   864          BLBC    R0,40$                              ;BR IF CAN'T LOCK BIT
009E C5    15 A4   90   023B   865          MOVB    SHD$B_PORT(R4),SHD$B_BITMAPLCK(R5) ;IDENTIFY HOLDER OF LOCK
             FDBC   30   0241   866          BSBW    MMG$SET_BITMAP                      ;FREE THE PAGES OF THE SECTION
00 009F C5    01   E7   0244   867          BBCCI   #SHD$V_BITMAPLCK,SHD$B_FLAGS(R5),30$ ;RELEASE BITMAP LOCK
            0504   30   024A   868 30$:     BSBW    MMG$SHMTXULK                        ;RELEASE MUTEX
       00 66    02   E7   024D   869          BBCCI   #GSD$V_DELPEND,GSD$L_GSDFL(R6),35$ ;CLEAR DELETE PENDING
       00 66    00   E7   0251   870 35$:     BBCCI   #GSD$V_VALID,GSD$L_GSDFL(R6),37$ ;CLEAR VALID BIT
          6E    01   D0   0255   871 37$:     MOVL    #1,(SP)                             ;FREED SOMETHING
       00 66    01   E7   0258   872 40$:     BBCCI   #GSD$V_LOCKED,GSD$L_GSDFL(R6),60$ ;UNLOCK GSD
       50    08 A6   3C   025C   873 60$:     MOVZWL  GSD$W_SIZE(R6),R0                   ;GET SIZE OF ONE GSD
          56   50   C0   0260   874          ADDL2   R0,R6                               ;GET ADR OF NEXT GSD
             A8 51   F4   0263   875 70$:     SOBGEQ  R1,10$                              ;ITERATE OVER ALL GSD'S
          0041 8F   BA   0266   876          POPR    #^M<R0,R6>                          ;RESTORE REGISTERS AND GET STATUS
                05   026A   877          RSB
                     026B   878          .DSABL LSB
```

```
                     026B   880          .SBTTL  FIND1STGSD - FIND THE FIRST GLOBAL SECTION TO SEARCH
                     026B   881 ;++
                     026B   882 ; FUNCTIONAL DESCRIPTION:
                     026B   883 ;
                     026B   884 ;       THIS ROUTINE TAKES AN INPUT STRING, BREAKS IT INTO SHARED MEMORY
                     026B   885 ;       AND GLOBAL SECTION NAMES WITH THE APPROPRIATE TRANSLATION, AND
                     026B   886 ;       RETURNS THE ADDRESS OF THE FIRST GLOBAL SECTION IN THE SEARCH PATH.
                     026B   887 ;
                     026B   888 ; CALLING SEQUENCE:
                     026B   889 ;
                     026B   890 ;       BSBW    MMG$FIND1STGSD
                     026B   891 ;
                     026B   892 ; INPUT PARAMETERS:
                     026B   893 ;
                     026B   894 ;         R6    - SYSTEM OR GROUP GLOBAL INDICATOR (1=SYSTEM, 0=GROUP)
                     026B   895 ;       (R10)   - SIZE OF SHARED MEMORY NAME (0 IF NO SH MEM NAME SPECIFIED)
                     026B   896 ;      4(R10)   - ADDRESS OF ASCIC SHARED MEMORY NAME
                     026B   897 ;
                     026B   898 ; IMPLICIT INPUTS:
                     026B   899 ;
                     026B   900 ;       NONE
                     026B   901 ;
                     026B   902 ; OUTPUT PARAMETERS:
                     026B   903 ;
                     026B   904 ;       IF A SHARED MEMORY IS BEING SEARCHED:
                     026B   905 ;               R4 - ADR OF SHARED MEMORY CONTROL BLOCK
                     026B   906 ;               R5 - ADR OF SHARED MEMORY COMMON DATA PAGE
                     026B   907 ;               R6 - ADR OF FIRST GSD OR 0 IF THERE IS NONE
                     026B   908 ;       IF LOCAL MEMORY IS BEING SEARCHED:
                     026B   909 ;               R4 - ADR OF LOCAL MEMORY GSD LISTHEAD
                     026B   910 ;               R6 - ADR OF FIRST LOCAL MEMORY GSD FROM LISTHEAD
                     026B   911 ;
                     026B   912 ; IMPLICIT OUTPUTS:
                     026B   913 ;
                     026B   914 ;       NONE
                     026B   915 ;
                     026B   916 ; COMPLETION CODES:
                     026B   917 ;
                     026B   918 ;       SS$_NORMAL - SUCCESS RETURN CODE
                     026B   919 ;       SS$_SHMNOTCNCT - SHARED MEMORY NOT CONNECTED
                     026B   920 ;
                     026B   921 ; SIDE EFFECTS:
                     026B   922 ;
                     026B   923 ;       NONE
                     026B   924 ;
                     026B   925 ;--
                     026B   926
                     026B   927 MMG$FIND1STGSD::
                 26  10   026B   928          BSBB    MMG$FINDSHB                     ;GET GS AND SHMEM NAMES
             22  50  E9   026D   929          BLBC    R0,20$                          ;BR ON ERROR FINDING SH MEM
                 54  D5   0270   930          TSTL    R4                              ;WAS SH MEM CONTROL BLK FOUND?
                 13  12   0272   931          BNEQ    10$                             ;BR ON YES
54   00000000'GF  46  7E   0274   932          MOVAQ   G^EXE$GL_GSDGRPFL[R6],R4        ;GET LISTHEAD FOR LOCAL MEM
                 56  54   D0   027C   933          MOVL    R4,R6                           ;SET UP TO FIND FIRST GSD
       0000009C'EF  16   027F   934          JSB     MMG$GETNXTGSD                   ;GET ADR OF FIRST LOCAL MEM GSD
                 0B  11   0285   935          BRB     20$                             ;RETURN
56   55   04  A5  C1   0287   936 10$:     ADDL3   SHD$L_GSDPTR(R5),R5,R6          ;GET ADR OF FIRST SH MEM GSD
```

```
                00000098'EF   16  028C   937          JSB     MMG$VALIDATEGSD                 ;CHECK IF GSD IS VALID, IF NOT
                              0292   938                                                      ;RETURN ADDRESS OF FIRST VALID
                              0292   939                                                      ;GSD OR 0 IF NONE IN R6
                              05  0292   940 20$:     RSB
```

```
                     0293   942          .SBTTL  FINDSHB - FIND SPECIFIC SHARED MEMORY CONTROL BLOCK
                     0293   943 ;++
                     0293   944 ; FUNCTIONAL DESCRIPTION:
                     0293   945 ;
                     0293   946 ;       THIS ROUTINE SEARCHED THE SHARED MEMORY CONTROL BLOCK LIST FOR
                     0293   947 ;       A SPECIFIC SHARED MEMORY.  IF FOUND, THE ADDRESSES FOR THE CONTROL
                     0293   948 ;       BLOCK AND THE COMMON DATA PAGE FOR THAT SHARED MEMORY ARE RETURNED.
                     0293   949 ;
                     0293   950 ; CALLING SEQUENCE:
                     0293   951 ;
                     0293   952 ;       BSBW    MMG$FINDSHB
                     0293   953 ;
                     0293   954 ; INPUT PARAMETERS:
                     0293   955 ;
                     0293   956 ;        (R10) - SIZE OF SHARED MEMORY NAME (0 IF NO SH MEM NAME SPECIFIED)
                     0293   957 ;       4(R10) - ADDRESS OF ASCIC SHARED MEMORY NAME
                     0293   958 ;
                     0293   959 ; IMPLICIT INPUTS:
                     0293   960 ;
                     0293   961 ;       NONE
                     0293   962 ;
                     0293   963 ; OUTPUT PARAMETERS:
                     0293   964 ;
                     0293   965 ;       R4 - CONTAINS THE ADR OF THE SHARED MEMORY CONTROL BLOCK OR
                     0293   966 ;            ZERO IF NONE FOUND
                     0293   967 ;       R5 - CONTAINS THE ADR OF THE COMMON DATA PAGE FOR THE SHARED
                     0293   968 ;            MEMORY IF R4 IS NOT ZERO, OTHERWISE JUNK
                     0293   969 ;
                     0293   970 ; IMPLICIT OUTPUTS:
                     0293   971 ;
                     0293   972 ;       NONE
                     0293   973 ;
                     0293   974 ; COMPLETION CODES:
                     0293   975 ;
                     0293   976 ;       SS$_NORMAL - SUCCESS RETURN CODE
                     0293   977 ;       SS$_SHMNOTCNCT - SHARED MEMORY NOT CONNECTED
                     0293   978 ;
                     0293   979 ; SIDE EFFECTS:
                     0293   980 ;
                     0293   981 ;       NONE
                     0293   982 ;
                     0293   983 ;--
                     0293   984
                     0293   985 MMG$FINDSHB::
                0E   BB 0293   986          PUSHR   #^M<R1,R2,R3>                   ;SAVE REGISTERS
                     0295   987          ASSUME  SS$_NORMAL LT <^X100>
             7E      01 9A  0295   988          MOVZBL  #SS$_NORMAL,-(SP)               ;ASSUME SUCCESS
                6A   D5 0298   989          TSTL    (R10)                           ;IS SHARED MEM NAME SPECIFIED?
                24   13 029A   990          BEQL    30$                             ;BR ON NO NAME
          54 00000000'GF   D0 029C   991          MOVL    G^EXE$GL_SHBLIST,R4             ;GET FIRST SH MEM CONTROL BLK
                16   13 02A3   992          BEQL    25$                             ;BR ON NO CONTROL BLK
       0C 0B A4    00   E1 02A5   993 10$:     BBC     #SHB$V_CONNECT,SHB$B_FLAGS(R4),20$ ;BR ON MEMORY NOT CONNECTED
          55    04 A4   D0 02AA   994          MOVL    SHB$L_DATAPAGE(R4),R5           ;GET COMMON DATA PAGE ADR
    20 A5    04 BA   10   29 02AE   995          CMPC3   #16,@4(R10),SHD$T_NAME(R5)      ;IS NAME STRING THE SAME?
                0C   13 02B4   996          BEQL    40$                             ;RETURN SHB FOUND
             54 64   D0 02B6   997 20$:     MOVL    SHB$L_LINK(R4),R4               ;GET NEXT SHB
                EA   12 02B9   998          BNEQ    10$                             ;GO TRY TO MATCH SH MEM NAME
```

```
                        02BB   999                 ASSUME  SS$_SHMNOTCNCT LT <^X10000>
6E   037C 8F    3C      02BB  1000 25$:            MOVZWL  #SS$_SHMNOTCNCT,(SP)        ;REPORT SH MEM SHB NOT FOUND
              54    D4  02C0  1001 30$:            CLRL    R4                          ;INDICATE SH MEM NOT FOUND
              50 8ED0   02C2  1002 40$:            POPL    R0                          ;GET RETURN STATUS CODE
              0E    BA  02C5  1003                 POPR    #^M<R1,R2,R3>               ;RESTORE REGISTERS
                    05  02C7  1004                 RSB                                 ;RETURN SHB ADR
                        02C8  1005
```

```
02C8  1007          .SBTTL  GETNXT/VALIDATEGSD - GET NEXT VALID GLOBAL SECTION DESCRIPTOR
02C8  1008 ;++
02C8  1009 ; FUNCTIONAL DESCRIPTION:
02C8  1010 ;
02C8  1011 ;       THIS ROUTINE FINDS THE NEXT SEQUENTIAL GLOBAL SECTION DESCRIPTOR.
02C8  1012 ;       IF LOCAL MEMORY GSD'S ARE BEING SEARCHED, THEN THE "NEXT" GSD IS
02C8  1013 ;       FOUND BY THE FORWARD LINK, GSD$L_GSDFL.  IF THERE ARE NO MORE
02C8  1014 ;       LOCAL MEMORY GSD'S, THEN THE SHARED MEMORIES ARE SEARCHED FOR
02C8  1015 ;       THE NEXT GSD.  IF A SPECIFIC SHARED MEMORY IS BEING SEARCHED, I.E.,
02C8  1016 ;       THE SHARED MEMORY NAME DESCRIPTOR HAS A COUNT GREATER THAN ZERO,
02C8  1017 ;       THEN THE NEXT PHYSICALLY CONSECUTIVE GSD IS TESTED TO SEE IF IT
02C8  1018 ;       IS VALID.  IF THERE ARE NO MORE VALID GSD'S IN THE SPECIFIC
02C8  1019 ;       SHARED MEMORY REQUESTED, THE OTHER SHARED MEMORIES ARE NOT SEARCHED.
02C8  1020 ;       INSTEAD, AN ERROR CODE INDICATING NO MORE GSD'S IS RETURNED.
02C8  1021 ;
02C8  1022 ;       THE SHARED MEMORY NAME DESCRIPTOR COUNT IS SET TO MINUS ONE IF
02C8  1023 ;       THE END OF THE GSD LIST IN LOCAL MEMORY WAS REACHED AND THE SEARCH
02C8  1024 ;       IS NOW BEING EXTENDED INTO THE SHARED MEMORIES.
02C8  1025 ;
02C8  1026 ;       THE SECOND ENTRY POINT, MMG$VALIDATEGSD, IS CALLED WHEN THE FIRST
02C8  1027 ;       GSD HAS BEEN LOCATED IN THE SHARED MEMORY GSD TABLE.  IT IS USED
02C8  1028 ;       TO VALIDATE THAT THE GSD "IN HAND" IS A VALID GSD.  IF IT IS NOT
02C8  1029 ;       A VALID GSD, THEN THE ROUTINE PROCEEDS TO FIND THE FIRST VALID
02C8  1030 ;       GSD IN THE SHARED MEMORY TABLE JUST AS DESCRIBED ABOVE.
02C8  1031 ;
02C8  1032 ; CALLING SEQUENCE:
02C8  1033 ;
02C8  1034 ;       BSBW    MMG$GETNXTGSD
02C8  1035 ;       BSBW    MMG$VALIDATEGSD
02C8  1036 ;
02C8  1037 ; INPUT PARAMETERS:
02C8  1038 ;
02C8  1039 ;       R6 - ADR OF LAST GSD FOUND WITH THIS SCAN
02C8  1040 ;       R10 - ADR OF STRING DESCRIPTOR FOR SHARED MEMORY NAME
02C8  1041 ;             STRING SIZE IS ZERO IF NO SHARED MEMORY NAME SPECIFIED
02C8  1042 ;             STRING SIZE IS -1 IF LOCAL MEMORY SEARCH HAS EXTENDED INTO
02C8  1043 ;               SEARCHING A SHARED MEMORY.
02C8  1044 ;       IF SHARED MEMORY SEARCH:
02C8  1045 ;             R4 - ADR OF SHARED MEMORY CONTROL BLOCK
02C8  1046 ;             R5 - ADR OF SHARED MEMORY COMMON DATA PAGE
02C8  1047 ;       IF LOCAL MEMORY SEARCH:
02C8  1048 ;             R4 - ADR OF LOCAL MEMORY GSD LISTHEAD
02C8  1049 ;
02C8  1050 ; IMPLICIT INPUTS:
02C8  1051 ;
02C8  1052 ;       NONE
02C8  1053 ;
02C8  1054 ; OUTPUT PARAMETERS:
02C8  1055 ;
02C8  1056 ;       R6 - ADR OF NEXT SEQUENTIAL GSD OR ZERO IF NO NEXT GSD
02C8  1057 ;
02C8  1058 ; IMPLICIT OUTPUTS:
02C8  1059 ;
02C8  1060 ;       IF LOCAL MEMORY SEARCH EXTENDS INTO SHARED MEMORY:
02C8  1061 ;             R4 - ADR OF SHARED MEMORY CONTROL BLOCK
02C8  1062 ;             R5 - ADR OF SHARED MEMORY COMMON DATA PAGE
02C8  1063 ;             4(R10) - SHARED MEMORY NAME SIZE IS SET TO -1
```

```
                        02C8  1064 ;
                        02C8  1065 ; COMPLETION CODES:
                        02C8  1066 ;
                        02C8  1067 ;       NONE
                        02C8  1068 ;
                        02C8  1069 ; SIDE EFFECTS:
                        02C8  1070 ;
                        02C8  1071 ;       NONE
                        02C8  1072 ;
                        02C8  1073 ;--
                        02C8  1074
                        02C8  1075 ; ******************************************************************************
                        02C8  1076 ;
                        02C8  1077 ; ************** THE FOLLOWING CODE MUST BE RESIDENT ********************
                        02C8  1078 ;
                  00000098  1079       .PSECT  $MMGCOD
                        0098  1080 ;
                        0098  1081 ; ******************************************************************************
                        0098  1082
                        0098  1083       .ENABL  LSB
                        0098  1084 MMG$VALIDATEGSD::
          03    BB      0098  1085       PUSHR   #^M<R0,R1>                      ;REMEMBER REGISTER
          28    11      009A  1086       BRB     15$                             ;GO VALIDATE GSD "IN HAND"
                        009C  1087
                        009C  1088 MMG$GETNXTGSD::
          03    BB      009C  1089       PUSHR   #^M<R0,R1>                      ;REMEMBER REGISTER
          6A    D5      009E  1090       TSTL    (R10)                           ;IS THIS A SHARED MEM SEARCH?
          28    12      00A0  1091       BNEQ    20$                             ;BR IF SEARCHING SHARED MEMORY
     56   66    D0      00A2  1092       MOVL    GSD$L_GSDFL(R6),R6              ;GET NEXT LOCAL MEMORY GSD
     56   54    D1      00A5  1093       CMPL    R4,R6                           ;IS THIS BACK TO LISTHEAD?
          5D    12      00A8  1094       BNEQ    70$                             ;NO, BR TO RETURN NEXT GSD
                        00AA  1095
                        00AA  1096 ;
                        00AA  1097 ; DEFAULT SEARCH OVERFLOW FROM LOCAL MEMORY INTO SHARED MEMORY.
                        00AA  1098 ;
     54   00000000'GF   D0    00AA  1099       MOVL    G^EXE$GL_SHBLIST,R4             ;GET PTR TO SH MEM CONTROL BLK
                52       13  00B1  1100 10$:    BEQL    60$                             ;BR ON NO SHARED MEMORY
        4D 0B A4    00    E1  00B3  1101       BBC     #SHB$V_CONNECT,SHB$B_FLAGS(R4),60$ ;BR IF SH MEM NOT CONNECTED
          55   04 A4    D0  00B8  1102       MOVL    SHB$L_DATAPAGE(R4),R5           ;GET ADR OF COMMON DATA PAGE
             6A    01    CE  00BC  1103       MNEGL   #1,(R10)                        ;INDICATE DEFAULT SH MEM SEARCH
     56   55    04 A5    C1  00BF  1104       ADDL3   SHD$L_GSDPTR(R5),R5,R6          ;GET FIRST GSD ADR
          50    08 A6    3C  00C4  1105 15$:    MOVZWL  GSD$W_SIZE(R6),R0               ;GET SIZE OF SHMEM GSD
                0D       11  00C8  1106       BRB     30$                             ;GO CHECK VALIDITY OF GSD
                        00CA  1107
                        00CA  1108 ;
                        00CA  1109 ; FIND NEXT SHARED MEMORY GSD IN TABLE.  SHARED MEMORY GSD'S ARE CONTAINED
                        00CA  1110 ; IN A TABLE AND ARE NOT LINKED VIA FORWARD AND BACKWARD LINKS.
                        00CA  1111 ;
             50   01    9A  00CA  1112 20$:    MOVZBL  #1,R0                           ;ONE REF COUNT FOR A LOCK
                FFA6    30  00CD  1113       BSBW    MMG$DECSHMREF                   ;RELEASE THE PREVIOUS GSD LOCK
          50    08 A6    3C  00D0  1114       MOVZWL  GSD$W_SIZE(R6),R0               ;GET SIZE OF SHMEM GSD
             56   50    C0  00D4  1115       ADDL2   R0,R6                           ;POINT TO NEXT GSD
          51    18 A5    3C  00D7  1116 30$:    MOVZWL  SHD$W_GSDMAX(R5),R1             ;GET MAX # GSD'S IN TABLE
             51   50    C4  00DB  1117       MULL2   R0,R1                           ;FIND SIZE OF GSD TABLE
             51   55    C0  00DE  1118       ADDL2   R5,R1                           ;ADD IN BASE VA
          51    04 A5    C0  00E1  1119       ADDL2   SHD$L_GSDPTR(R5),R1             ;COMPUTE ADR OF END OF TABLE
                07       11  00E5  1120       BRB     50$                             ;SKIP OFFSETING TO NEXT GSD
```

```
                50    08 A6    3C   00E7  1121 40$:    MOVZWL  GSD$W_SIZE(R6),R0                       ;GET SIZE OF ONE SHMEM GSD
                   56    50    C0   00EB  1122          ADDL2   R0,R6                                   ;GET ADR OF NEXT GSD
                   51    56    D1   00EE  1123 50$:     CMPL    R6,R1                                   ;PAST END OF GSD TABLE?
                         17    1E   00F1  1124          BGEQU   80$                                     ;BR IF YES, PAST LAST GSD
                   50    01    9A   00F3  1125          MOVZBL  #1,R0                                   ;ONE REF COUNT FOR A LOCK
                      FF80    30   00F6  1126          BSBW    MMG$INCSHMREF                           ;LOCK THE GSD
                0A 66    00    E0   00F9  1127          BBS     #GSD$V_VALID,GSD$L_GSDFL(R6),70$        ;BR IF CAN READ GSD, RETURN IT
                   50    01    9A   00FD  1128          MOVZBL  #1,R0                                   ;ONE REF COUNT FOR A LOCK
                      FF73    30   0100  1129          BSBW    MMG$DECSHMREF                           ;RELEASE THIS GSD LOCK
                         E2    11   0103  1130          BRB     40$                                     ;BR TO FIND NEXT GSD
                         56    D4   0105  1131 60$:     CLRL    R6                                      ;INDICATE NO MORE GSD'S
                         03    BA   0107  1132 70$:     POPR    #^M<R0,R1>                              ;RESTORE REGISTER
                               05   0109  1133          RSB                                             ;RETURN WITH NEXT GSD ADR
                         6A    D5   010A  1134 80$:     TSTL    (R10)                                   ;SEARCHING SPECIFIC SH MEM?
                         F7    18   010C  1135          BGEQ    60$                                     ;BR ON YES, DON'T SEARCH OTHERS
                   54    64    D0   010E  1136          MOVL    SHB$L_LINK(R4),R4                       ;GET NEXT SH MEM CONTROL BLK
                         9E    11   0111  1137          BRB     10$                                     ;GO SHECK SHB VALIDITY
                                    0113  1138
                                    0113  1139          .DSABL  LSB
```

```
                        .SBTTL  GETGSNAM - GET GLOBAL SECTION NAME AND SHARED MEMORY NAME
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ; THIS ROUTINE TAKES AN INPUT STRING WHICH MAY BE A GLOBAL SECTION NAME, A
                ; LOGICAL NAME, OR A SHARED MEMORY NAME AND A GLOBAL SECTION NAME.  IF THE
                ; STRING IS SUFFIXED WITH "_nnn" (AN UNDERSCORE FOLLOWED BY THREE DIGITS)
                ; THE SUFFIX IS REMOVED.  THEN THE STRING IS SUBMITTED FOR LOGICAL NAME
                ; TRANSLATION AND SEPARATION INTO GLOBAL SECTION NAME AND SHARED MEMORY NAME.
                ; THE SUFFIX IS APPENDED ONTO THE RESULTANT GLOBAL SECTION NAME.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       BSBW    MMG$GETGSNAM
                ;
                ; INPUT PARAMETERS:
                ;
                ;       R9 - ADR OF STRING DESCRIPTOR FOR INPUT STRING FROM USER
                ;       R10 - ADR OF STRING DESCRIPTOR FOR RETURNED SHARED MEMORY NAME
                ;       R11 - ADR OF STRING DESCRIPTOR FOR RETURNED GLOBAL SECTION NAME
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       THE INPUT STRING DESCRIPTOR POINTS TO THE STRING TO BE TRANSLATED.
                ;       THE OUTPUT STRING DESCRIPTORS ARE SET TO DESCRIBE THE SIZE AND
                ;       ADDRESS OF THE OUTPUT BUFFERS.
                ;
                ; OUTPUT PARAMETERS:
                ;
                ;       R0 CONTAINS THE STATUS CODE FOR THE TRANSLATION.
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       THE SHARED MEMORY AND GLOBAL SECTION NAMES ARE ENTERED IN THE
                ;       BUFFERS DESCRIBED BY THE INPUT STRING DESCRIPTORS.  THE DESCRIPTORS
                ;       ARE UPDATED.  IF AN ERROR CODE IS RETURNED, THE DESCRIPTORS ARE
                ;       NOT VALID.
                ;
                ; COMPLETION CODES:
                ;
                ;       SS$_NORMAL - SUCCESSFUL COMPLETION
                ;       SS$_IVLOGNAM - NAME TOO LARGE FOR USER BUFFER
                ;       SS$_TOOMANYLNAM - TOO MANY LOGICAL NAME TRANSLATIONS
                ;
                ; SIDE EFFECTS:
                ;
                ;       NONE
                ;
                ;--
                ;*****************************************************************************
                ;
                ;****************** THE FOLLOWING CODE MAY BE PAGED ********************
                ;
000002C8        .PSECT  Y$EXEPAGED
                ;
                ;*****************************************************************************
```

```
                        02C8  1198
                        02C8  1199 MMG$GETGSNAM::
        0202 8F   BB    02C8  1200          PUSHR   #^M<R1,R9>                ;SAVE REGISTERS
          04 A9   DD    02CC  1201          PUSHL   4(R9)                    ;BUILD AN INPUT NAME STRING
        7E   69   3C    02CF  1202          MOVZWL  (R9),-(SP)               ;DESCRIPTOR THAT CAN BE MODIFIED
        59   5E   D0    02D2  1203          MOVL    SP,R9                    ;SET ADR OF INPUT NAME STR DSC
     50 69   04   C3    02D5  1204          SUBL3   #4,(R9),R0               ;GET STR SIZE MINUS SUFFIX
           23   15    02D9  1205          BLEQ    10$                      ;BR IF STRING HAS NO SUFFIX
     50   04 A9   C0    02DB  1206          ADDL2   4(R9),R0                 ;GET ADR OF SUFFIX
     60   5F 8F   91    02DF  1207          CMPB    #^A/_/,(R0)              ;IS THIS A SUFFIX?
           19   12    02E3  1208          BNEQ    10$                      ;BR ON NO
        51   03   9A    02E5  1209          MOVZBL  #3,R1                    ;SIZE OF SUFFIX
     30   6041 91    02E8  1210 5$:       CMPB    (R0)[R1],#^A/0/          ;IS CHARACTER LESS THAN "0"?
           10   1F    02EC  1211          BLSSU   10$                      ;BR ON SUFFIX NOT NUMERIC
     39   6041 91    02EE  1212          CMPB    (R0)[R1],#^A/9/          ;IS CHARACTER GREATER THAN "9"?
           0A   1A    02F2  1213          BGTRU   10$                      ;BR ON SUFFIX NOT NUMERIC
        F1 51   F5    02F4  1214          SOBGTR  R1,5$                    ;REPEAT TO CHECK ALL OF SUFFIX
           60   DD    02F7  1215          PUSHL   (R0)                     ;REMEMBER THE SUFFIX
        69   04   C2    02F9  1216          SUBL2   #4,(R9)                  ;SUBTRACT OFF THE SUFFIX
           02   11    02FC  1217          BRB     20$                      ;GO TRANSLATE NAME
           00   DD    02FE  1218 10$:      PUSHL   #0                       ;INDICATE NO SUFFIX
           6B   DD    0300  1219 20$:      PUSHL   (R11)                    ;REMEMBER SIZE OF GS BUFFER
           5D   10    0302  1220          BSBB    MMG$GSDTRNLOG            ;TRANSLATE LOGICAL NAME
        2F 50   E9    0304  1221          BLBC    R0,50$                   ;BR IF ERR TRANSLATING NAME
        04 AE   D5    0307  1222          TSTL    4(SP)                    ;WAS THERE A SUFFIX?
           2A   13    030A  1223          BEQL    50$                      ;BR IF NONE TO APPEND
     51 6B   04   C1    030C  1224          ADDL3   #4,(R11),R1              ;GET NEW SIZE OF GS
        51   8E   D1    0310  1225          CMPL    (SP)+,R1                 ;IS BUFFER TOO SMALL FOR SUFFIX?
           18   19    0313  1226          BLSS    40$                      ;BR ON YES
  51   04 AB   6B   C1    0315  1227          ADDL3   (R11),4(R11),R1          ;GET ADR FOR SUFFIX
        61   8E   D0    031A  1228          MOVL    (SP)+,(R1)               ;PUT SUFFIX ON END OF STRING
           06   13    031D  1229          BEQL    30$                      ;BR IF NO SUFFIX
        69   04   C0    031F  1230          ADDL2   #4,(R9)                  ;ADD IN LENGTH OF SUFFIX
        6B   04   C0    0322  1231          ADDL2   #4,(R11)                 ;ADD IN LENGTH OF SUFFIX
        5E   08   C0    0325  1232 30$:      ADDL2   #<4*2>,SP                ;CLEAN STR DSC OFF STACK
        0202 8F   BA    0328  1233          POPR    #^M<R1,R9>               ;RESTORE REGISTERS
           05   05    032C  1234          RSB                              ;RETURN
                        032D  1235          ASSUME  SS$_IVLOGNAM LT <^X10000>
     50 0154 8F   3C    032D  1236 40$:      MOVZWL  #SS$_IVLOGNAM,R0         ;REPORT BUFFER TOO SMALL
           8E   D5    0332  1237          TSTL    (SP)+                    ;CLEAN OFF SUFFIX
           EF   11    0334  1238          BRB     30$                      ;GO RETURN
        5E   08   C0    0336  1239 50$:      ADDL2   #<4*2>,SP                ;CLEAN SUFFIX AND CNT OFF
           EA   11    0339  1240          BRB     30$                      ;JOIN COMMON CODE
```

```
                        .SBTTL  GSDTRNLOG  - GLOBAL SECTION LOGICAL NAME TRANSLATION
                        .SBTTL  MBXTRNLOG  - MAILBOX LOGICAL NAME TRANSLATION
                        .SBTTL  CEFTRNLOG  - COMMON EVENT FLAG CLUSTER LOGICAL NAME TRANSLATION

                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ; MMG$GSDTRNLOG - TRANSLATE LOGICAL NAMES FOR GLOBAL SECTIONS.
                ; MMG$MBXTRNLOG - TRANSLATE LOGICAL NAMES FOR MAILBOXES.
                ; MMG$CEFTRNLOG - TRANSLATE LOGICAL NAMES FOR COMMON EVENT FLAG CLUSTERS.
                ;
                ; THE ONLY DIFFERENCE BETWEEN THESE THREE TRANSLATION ROUTINES IS THE PREFIX
                ; ADDED TO THE NAME STRING BEFORE EACH ITERATIVE TRANSLATION.  THE PREFIX FOR
                ; GLOBAL SECTIONS IS "GBL$", FOR MAILBOXES IT IS "MBX$", AND FOR COMMON EVENT
                ; FLAG CLUSTERS IT IS "CEF$".
                ;
                ; EACH ROUTINE IS CAPABLE OF ITERATIVELY TRANSLATING NAME STRINGS FOR BOTH
                ; SHARED AND LOCAL MEMORY OBJECTS.  SHARED MEMORY OBJECTS HAVE THE FOLLOWING
                ; SPECIAL FORMAT:
                ;
                ;                       SHARED-MEMORY-NAME:OBJECT-NAME
                ;
                ; AS SOON AS A COLON IS ENCOUNTERED WITHIN ( AND NOT AT THE END OF ) THE CURRENT
                ; INPUT STRING THE OBJECT IS ASSUMED TO BE LOCATED IN SHARED MEMORY.  ITERATIVE
                ; NAME STRING TRANSLATION FOR SHARED MEMORY OBJECTS PROCEEDS AS FOLLOWS:
                ;
                ; 1. THE CURRENT INPUT STRING IS SEARCHED FOR A COLON.
                ; 2. EVERYTHING TO THE RIGHT OF THE COLON IS PLACED IN THE GLOBAL SECTION /
                ;    MAILBOX / COMMON EVENT FLAG CLUSTER NAME BUFFER IN FRONT OF WHATEVER STRING
                ;    IS ALREADY PRESENT IN THE BUFFER.
                ; 3. EVERYTHING TO THE LEFT OF THE COLON ( OR THE ENTIRE CURRENT INPUT STRING
                ;    IF THERE IS NO COLON ) BECOMES THE CURRENT NAME STRING.
                ; 4. IF THE CURRENT NAME STRING CONTAINS A LEADING UNDERSCORE THEN THE
                ;    UNDERSCORE IS STRIPPED FROM THE CURRENT NAME STRING, ITERATIVE LOGICAL
                ;    NAME TRANSLATION TERMINATES, AND THE CURRENT NAME STRING BECOMES THE SHARED
                ;    MEMORY NAME.  GO TO STEP 9.
                ; 5. IF THE CURRENT NAME STRING IS ITSELF THE RESULTANT OF A LOGICAL NAME
                ;    TRANSLATION THEN IT IS CHECKED FOR POSSESSION OF THE "TERMINAL" ATTRIBUTE.
                ;    IF THE CURRENT TRANSLATION IS MARKED "TERMINAL" THEN ITERATIVE LOGICAL NAME
                ;    TRANSLATION TERMINATES, AND THE CURRENT NAME STRING BECOMES THE SHARED
                ;    MEMORY NAME.  GO TO STEP 9.
                ; 6. THE CURRENT NAME STRING IS PREFIXED WITH "GBL$" / "MBX$" / "CEF$",
                ;    SUBMITTED FOR LOGICAL NAME TRANSLATION, AND THE RESULTANT STRING BECOMES
                ;    THE CURRENT INPUT STRING.
                ; 7. THESE SIX STEPS ARE REPEATED UP TO LNM$C_MAXDEPTH TIMES.
                ; 8. WHEN THE CURRENT LOGICAL NAME TRANSLATION FAILS, THE CURRENT NAME STRING,
                ;    THE NAME THAT COULD NOT BE TRANSLATED, MINUS ITS UNIQUE OBJECT PREFIX,
                ;    BECOMES THE SHARED MEMORY NAME.
                ; 9. THE OBJECT NAME IS THE STRING THAT HAD BEEN CONSTRUCTED DURING STEP 2
                ;    OF THE ITERATIVE PROCESS FROM PIECES TO THE RIGHT OF COLONS.
                ;
                ; LOGICAL NAME TRANSLATION FOR OBJECTS IN LOCAL MEMORY PROCEEDS AS FOLLOWS:
                ;
                ; 1. IF THE CURRENT NAME STRING CONTAINS A LEADING UNDERSCORE THEN THE
                ;    UNDERSCORE IS STRIPPED FROM THE CURRENT NAME STRING AND ITERATIVE LOGICAL
                ;    NAME TRANSLATION TERMINATES.  GO TO STEP 5.
                ; 2. IF THE CURRENT NAME STRING IS ITSELF THE RESULTANT OF A LOGICAL NAME
```

```
                     ;    TRANSLATION THEN IT IS CHECKED FOR POSSESSION OF THE "TERMINAL" ATTRIBUTE.
                     ;    IF THE CURRENT TRANSLATION IS MARKED "TERMINAL" THEN ITERATIVE LOGICAL NAME
                     ;    TRANSLATION TERMINATES.  GO TO STEP 5.
                     ; 3. THE CURRENT NAME STRING IS PREFIXED WITH "GBL$" / "MBX$" / "CEF$", D
                     ;    SUBMITTED FOR LOGICAL NAME TRANSLATION, AND THE RESULTANT STRING BECOMES
                     ;    THE CURRENT NAME STRING.
                     ; 4. THESE THREE STEPS ARE REPEATED UP TO LNM$C_MAXDEPTH TIMES OR UNTIL
                     ;    TRANSLATION OF THE CURRENT NAME STRING FAILS.
                     ; 5. WHEN THE ITERATIVE LOGICAL NAME TRANSLATION TERMINATES, THE CURRENT NAME
                     ;    STRING, MINUS ITS UNIQUE OBJECT PREFIX, BECOMES THE OBJECT NAME.
                     ;
                     ; THE UNIQUE OBJECT PREFIX STRING "GBL$" / "MBX$" / "CEF$" IS NEVER RETURNED TO
                     ; THE USER AS PART OF EITHER THE SHARED MEMORY OR OBJECT NAME ALTHOUGH IT IS
                     ; PREFIXED TO EACH STRING SUBMITTED FOR LOGICAL NAME TRANSLATION.
                     ;
                     ;
                     ; CALLING SEQUENCE:
                     ;
                     ;       BSBW    MMG$GSDTRNLOG
                     ;       BSBW    MMG$MBXTRNLOG
                     ;       BSBW    MMG$CEFTRNLOG
                     ;
                     ; INPUT PARAMETERS:
                     ;
                     ;       R9      - ADDRESS OF STRING DESCRIPTOR FOR INPUT STRING FROM USER
                     ;       R10     - ADDRESS OF STRING DESCRIPTOR FOR RETURNED SHARED MEMORY NAME
                     ;       R11     - ADDRESS OF STRING DESCRIPTOR FOR RETURNED OBJECT NAME
                     ;
                     ; IMPLICIT INPUTS:
                     ;
                     ;       THE INPUT STRING DESCRIPTOR POINTS TO THE STRING TO BE TRANSLATED.
                     ;       THE OUTPUT STRING DESCRIPTORS ARE SET TO DESCRIBE THE SIZE AND
                     ;           ADDRESS OF THE OUTPUT BUFFERS.
                     ;
                     ; OUTPUT PARAMETERS:
                     ;       NONE
                     ;
                     ; IMPLICIT OUTPUTS:
                     ;
                     ;       THE SHARED MEMORY AND OBJECT NAMES ARE ENTERED IN THE BUFFERS DESCRIBED
                     ;       BY THE INPUT STRING DESCRIPTORS.  THE DESCRIPTORS ARE UPDATED.  IF AN
                     ;       ERROR CODE IS RETURNED, THE DESCRIPTORS ARE NOT VALID.  IF EITHER NAME
                     ;       IS NOT FOUND, THE APPROPRIATE DESCRIPTOR'S SIZE FIELD  IS SET TO ZERO.
                     ;
                     ; COMPLETION CODES:
                     ;
                     ;       SS$_NORMAL      - SUCCESSFUL COMPLETION OF THE ROUTINE
                     ;       SS$_NOPRIV      - INSUFFICIENT PRIVILEGE TO ACCESS A LOGICAL NAME TABLE
                     ;       SS$_IVLOGNAM    - EITHER THE OBJECT NAME OR SHARED MEMORY BUFFER IS TOO
                     ;                         SMALL TO HOLD THE CORRESPONDING NAME
                     ;                         OR INPUT STRING ITERATIVELY TRANSLATES INTO A ZERO
                     ;                         LENGTH OBJECT NAME
                     ;       SS$_TOOMANYLNAM - ITERATIVE LOGICAL NAME TRANSLATION DEPTH EXCEEDED
                     ;                         LNM$C_MAXDEPTH.
                     ;
                     ; SIDE EFFECTS:
                     ;
```

```
033B  1356 ;       THIS ROUTINE ASSUMES THE UPPER WORD IN RETURN STRING DESCRIPTORS IS 0.
033B  1357 ;
033B  1358 ;--
```

```
                                033B  1360
                                033B  1361 ;
                                033B  1362 ; LOGICAL NAME TRANSLATION WORK AREA OFFSETS INTO KERNEL REQUEST PACKET
                                033B  1363 ; AND LOGICAL NAME STORAGE.
                                033B  1364 ;
                                033B  1365
                                033B  1366          ASSUME  LNMX$T_XLATION+1,GE,4
                                033B  1367
                   00000000     033B  1368 LWA_PREFIX      =   0                       ;LOGICAL NAME PREFIX
                   00000004     033B  1369 LWA_INPUT_DESC  =   4                       ;CURRENT INPUT STRING DESCRIPTOR
                   0000000C     033B  1370 LWA_COLON       =  12                       ;COLON INDICATOR CELL
                   0000000D     033B  1371 LWA_XLATION     =  13                       ;BUFFER TO HOLD TRANSLATION BLOCKS
                   00000012     033B  1372 LWA_INPUT       =  13+LNMX$T_XLATION+1      ;CURRENT INPUT STRING ADDRESS
                   00000111     033B  1373 LWA_END         =  LWA_INPUT+LNM$C_NAMLENGTH
                                033B  1374
                                033B  1375          ASSUME  LWA_END,LE,512
                                033B  1376
                                033B  1377 FILE_DEV_DESC:                           ;DESCRIPTOR OF LOGICAL NAME TABLE NAME
                   0000000C'    033B  1378          .LONG   FILE_DEV_SIZE
                   00000343'    033F  1379          .ADDRESS FILE_DEV
                                0343  1380
                                0343  1381 FILE_DEV:                               ;LOGICAL NAME TABLE NAME BUFFER
56 45 44 5F 45 4C 49 46 24 4D 4E 4C  0343  1382          .ASCII  /LNM$FILE_DEV/
                   0000000C     034F  1383 FILE_DEV_SIZE = . - FILE_DEV
                                034F  1384
                                034F  1385          .ENABLE LSB
                                034F  1386 MMG$CEFTRNLOG::
            50   24464543 8F    D0  034F  1387          MOVL    #^A/CEF$/,R0            ;SET INDICATOR TO USE ''CEF$''
                          10    11  0356  1388          BRB     10$                     ;SKIP OTHER PREFIXES
                                0358  1389
                                0358  1390 MMG$MBXTRNLOG::
50   2458424D 8F    D0  0358  1391          MOVL    #^A/MBX$/,R0            ;SET INDICATOR TO USE ''MBX$''
                   07    11  035F  1392          BRB     10$                     ;SKIP OTHER PREFIXES
                                0361  1393
                                0361  1394 MMG$GSDTRNLOG::
50   244C4247 8F    D0  0361  1395          MOVL    #^A/GBL$/,R0            ;SET INDICATOR TO USE ''GBL$''
                                0368  1396
        0FFE 8F    BB  0368  1397 10$:     PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;SAVE REGISTERS
                                036C  1398
                                036C  1399 ;
                                036C  1400 ; ALLOCATE AND INITIALIZE A KERNEL REQUEST PACKET TO PROVIDE A WORK AREA.
                                036C  1401 ;
                                036C  1402
57   00000000'GF    9E  036C  1403          MOVAB   G^CTL$GL_KRPFL,R7       ;RETRIEVE ADDRESS OF KRP QUEUE LISTHEAD
      57   04 B7    0F  0373  1404          REMQUE  @4(R7),R7               ;RETRIEVE KRP FROM LIST
                   04    1C  0377  1405          BVC     20$                     ;CONTINUE IF GOT ONE
                                0379  1406          BUG_CHECK       KRPEMPTY,FATAL  ;OTHERWISE BUGCHECK
                                037D  1407
        67   50    D0  037D  1408 20$:     MOVL    R0,LWA_PREFIX(R7)       ;STORE UNIQUE PREFIX IN WORD AREA
                                0380  1409
        50   69    3C  0380  1410          MOVZWL  (R9),R0                 ;RETRIEVE SIZE OF INPUT STRING FROM USER
50   000000FF 8F    D1  0383  1411          CMPL    #LNM$C_NAMLENGTH,R0     ;IS INPUT STRING OF VALID SIZE?
                   03    1E  038A  1412          BGEQU   25$                     ;CONTINUE IF IT IS; ELSE,
                 0104    31  038C  1413          BRW     INVALID_LOGNAM          ;RETURN ERROR IF INPUT STRING TOO LARGE
                                038F  1414
                                038F  1415          ASSUME  LNMX$T_XLATION,LE,8
                  OD A7    7C  038F  1416 25$:     CLRQ    LWA_XLATION(R7)         ;CREATE ''TRANSLATION BLOCK'' FOR USER
```

```
                11 A7   50   90  0392  1417          MOVB    R0,LWA_INPUT-1(R7)      ;SUPPLIED INPUT STRING
        12 A7   04 B9   50   28  0396  1418          MOVC3   R0,@4(R9),LWA_INPUT(R7)
                                 039C  1419
                   12 A7   9E  039C  1420          MOVAB   LWA_INPUT(R7),-         ;INITIALIZE CURRENT INPUT STRING
                   08 A7       039F  1421                  LWA_INPUT_DESC+4(R7)    ;DESCRIPTOR BUFFER ADDRESS
```

```
                                03A1
                                03A1    ;
                                03A1    ; SETUP TO PERFORM THE ITERATIVE LOGICAL NAME TRANSLATIONS, AND THEN BEGIN BY
                                03A1    ; PROCESSING THE USER SUPPLIED INPUT STRING AS IF IT WERE THE RESULT OF A
                                03A1    ; LOGICAL NAME TRANSLATION.  IN OTHER WORDS, CHECK THE INPUT STRING FOR A COLON
                                03A1    ; INDICATIVE OF A SHARED MEMORY OBJECT, AND THEN DETERMINE WHETHER OR NOT THE
                                03A1    ; ITERATIVE LOGICAL NAME TRANSLATIONS SHOULD BE TERMINATED.
                                03A1    ;
                                03A1    ; R6  - ADDRESS OF BUFFER TO RECEIVE RESULTANT TRANSLATION BLOCKS
                                03A1    ; R7  - ADDRESS OF KRP
                                03A1    ; R8  - SIZE OF OBJECT NAME BUFFER REMAINING
                                03A1    ; R9  - ITERATIVE LOGICAL NAME TRANSLATION COUNTER
                                03A1    ; R10 - ADDRESS OF RETURNED SHARED NAME BUFFER DESCRIPTOR
                                03A1    ; R11 - ADDRESS OF RETURNED OBJECT NAME BUFFER DESCRIPTOR
                                03A1    ;
                                03A1
           56   0D A7     9E    03A1            MOVAB   LWA_XLATION(R7),R6            ;RETRIEVE ADDRESS OF XLATION BUFFER
                58   6B   D0    03A5            MOVL    (R11),R8                      ;RETRIEVE OBJECT NAME BUFFER SIZE
                     6B   D4    03A8            CLRL    (R11)                         ;ZERO CURRENT OBJECT NAME SIZE
 04 BB   58   00     6B   00 2C 03AA            MOVC5   #0,(R11),#0,R8,@4(R11)        ;ZERO BUFFER (SOURCE SPEC IS MEANINGLESS)
                59   0A   D0    03B1            MOVL    #LNM$C_MAXDEPTH,R9            ;MAXIMUM NUMBER TRANSLATION ITERATIONS
                     3A   11    03B4            BRB     CHECK_XLATION                 ;GO CHECK USERS INPUT STRING FOR COLON
                                03B6                                                  ;AND WHETHER ITERATIONS SHOULD TERMINATE
                                03B6
                                03B6    ;
                                03B6    ; APPEND THE CURRENT NAME STRING TO THE OBJECT'S UNIQUE PREFIX AND THEN
                                03B6    ; TRANSLATE THE RESULTING NAME STRING UTILIZING A FAST INTERNAL INTERFACE.
                                03B6    ; NOTE THAT THE ROUTINE LNM$SEARCH_ONE WILL ONLY RETURN THE TRANSLATION BLOCKS
                                03B6    ; FOR TRANSLATIONS WITH INDEXES OF 0; OTHERWISE, AN ERROR OF SS$_NOLOGNAM IS
                                03B6    ; RETURNED.  THIS ROUTINE EXPECTS THE FOLLOWING REGISTERS AS INPUT:
                                03B6    ;
                                03B6    ; R0  - SIZE OF NAME STRING TO BE TRANSLATED
                                03B6    ; R1  - ADDRESS OF NAME STRING TO BE TRANSLATED
                                03B6    ; R2  - SIZE OF TABLE NAME STRING
                                03B6    ; R3  - ADDRESS OF TABLE NAME STRING
                                03B6    ; R4  - ADDRESS OF PCB
                                03B6    ; R5  - HIGH-ORDER WORD 0; CASE-INSENSITIVE FLAG; ACCESS MODE OF TRANSLATION
                                03B6    ; R6  - ADDRESS OF BUFFER TO RECEIVE RESULTANT TRANSLATION BLOCKS
                                03B6    ;
                                03B6    ; IF THE LOGICAL NAME TOGETHER WITH ITS PREFIX EXCEEDS THE MAXIMUM SIZE OF A
                                03B6    ; LOGICAL NAME THEN IMMEDIATELY TERMINATE THE ITERATIVE TRANSLATIONS.
                                03B6    ;
                                03B6
                                03B6    TRANSLATE_LOOP:                                       ;LOOP TO PERFORM ITERATIVE TRANSLATIONS
                     04   C3    03B6            SUBL3   #4,-                          ;SETUP DESCRIPTOR OF LOGICAL NAME TO
           51   08 A7           03B8                    LWA_INPUT_DESC+4(R7),R1       ;BE TRANSLATED
      50   04 A7     04   C1    03BB            ADDL3   #4,LWA_INPUT_DESC(R7),R0
   000000FF 8F       50   D1    03C0            CMPL    R0,#LNM$C_NAMLENGTH           ;IS RESULTING NAME TOO LARGE?
                     03   1B    03C7            BLEQU   27$                           ;IF SO THEN TERMINATE TRANSLATIONS
                   0094   31    03C9            BRW     STOP_TRANSLATION
                                03CC
                61   67   D0    03CC    27$:    MOVL    LWA_PREFIX(R7),(R1)           ;PREFIX CURRENT INPUT STRING WITH
                                03CF                                                  ;OBJECT'S UNIQUE PREFIX
      52   FF68 CF        7D    03CF            MOVQ    FILE_DEV_DESC,R2              ;LOGICAL NAME TABLE NAME DESCRIPTOR
   54   00000000'9F       D0    03D4            MOVL    @#CTL$GL_PCB,R4               ;RETRIEVE PCB ADDRESS
      55   0103 8F        3C    03DB            MOVZWL  #<1@8 + PSL$C_USER>,R5        ;ALL TRANSLATIONS ARE DONE CASE
                                03E0                                                  ;INSENSITIVE AND FROM USER MODE
```

```
                FC1D'  30  03E0  1480          BSBW    LNM$SEARCH_ONE          ;TRANSLATE THE CURRENT NAME STRING
                 0A 50 E8  03E3  1481          BLBS    R0,CHECK_XLATION        ;GO CHECK TRANSLATION IF SUCCESSFUL
          50    01BC 8F  B1  03E6  1482       CMPW    #SS$_NOLOGNAM,R0        ;IF FAILED TO TRANSLATE CURRENT NAME
                   73  13  03EB  1483          BEQL    STOP_TRANSLATION        ;STRING THEN TERMINATE TRANSLATIONS
                 00A8  31  03ED  1484          BRW     RETURN                  ;OTHERWISE GO RETURN AN ERROR
```

```
                                C 14
                                03F0  1486
                                03F0  1487 ;
                                03F0  1488 ; DETERMINE WHETHER OR NOT THE CURRENT INPUT STRING CONTAINS A COLON.  IF SO
                                03F0  1489 ; THEN EVERYTHING TO THE RIGHT OF THE COLON BECOMES THE CURRENT OBJECT NAME
                                03F0  1490 ; PIECE.  IF THE COLON IS THE FIRST CHARACTER IN THE CURRENT INPUT STRING THEN
                                03F0  1491 ; THE COLON IS INCLUDED AS THE FIRST CHARACTER WITHIN THE CURRENT OBJECT NAME
                                03F0  1492 ; PIECE.  EVERYTHING TO THE LEFT OF THE COLON BECOMES THE CURRENT NAME STRING.
                                03F0  1493 ;
                                03F0  1494 ; THE CURRENT OBJECT NAME PIECE IS MOVED INTO THE OBJECT NAME BUFFER IN FRONT OF
                                03F0  1495 ; ANY PART OF THE OBJECT NAME UNDER CONSTRUCTION WHICH ALREADY RESIDES THERE.
                                03F0  1496 ; THE CURRENT NAME STRING IS SUBJECTED TO A SET OF TESTS TO DETERMINE WHETHER OR
                                03F0  1497 ; NOT ANOTHER ROUND OF LOGICAL NAME TRANSLATION IS REQUIRED.
                                03F0  1498 ;
                                03F0  1499
                                03F0  1500 CHECK_XLATION:
                             9A 03F0  1501          MOVZBL  LNMX$T_XLATION+-        ;INITIALIZE CURRENT INPUT STRING
                                03F1  1502                  LWA_XLATION(R7),-        ;DESCRIPTOR LENGTH FIELD
                 04 A7    11 A7     03F1  1503                  LWA_INPUT_DESC(R7)
                          3A 3A     03F5  1504          LOCC    #^A7:/,-                ;IS THERE A COLON PRESENT IN THE
                       04 A7        03F7  1505                  LWA_INPUT_DESC(R7),-    ;CURRENT INPUT STRING?
                       12 A7        03F9  1506                  LWA_INPUT(R7)
                    0C A7    50 90 03FB  1507          MOVB    R0,LWA_COLON(R7)                ;SAVE WHETHER OR NOT A COLON WAS FOU
                                03FF  1508                                          ;DURING THIS ITERATION AND
                          30 13     03FF  1509          BEQL    40$                     ;BRANCH IF NO COLON WAS FOUND
                    04 A7    50 C2 0401  1510          SUBL2   R0,LWA_INPUT_DESC(R7)   ;ELSE COMPUTE SIZE OF REMAINING NAME
                          04 12     0405  1511          BNEQ    30$                     ;IF THE VERY FIRST CHARACTER IS A COLON
                          50 D6     0407  1512          INCL    R0                      ;THEN SETUP SO THAT THE COLON WILL BE
                          51 D7     0409  1513          DECL    R1                      ;TREATED AS PART OF THE NEW OBJECT NAME
                                040B  1514                                          ;PIECE
                                040B  1515
                          50 D7     040B  1516 30$:     DECL    R0                      ;COMPUTE SIZE OF NEW OBJECT NAME PIECE
                          22 13     040D  1517          BEQL    40$                     ;NO NEED TO MOVE IT IF SIZE IS ZERO
                       58 50 C2     040F  1518          SUBL2   R0,R8                   ;IS OBJECT NAME BUFFER LARGE ENOUGH?
                          7F 19     0412  1519          BLSS    INVALID_LOGNAM          ;RETURN AN ERROR IF IT ISN'T
                                0414  1520
                          6B D5     0414  1521          TSTL    (R11)                   ;ANY PART OF THE OBJECT NAME TO MOVE?
                          10 13     0416  1522          BEQL    35$                     ;BRANCH IF NOTHING TO MOVE
                 52 50    04 AB C1  0418  1523          ADDL3   4(R11),R0,R2            ;COMPUTE ADDRESS OF WHERE TO MOVE
                                041D  1524                                          ;CURRENT OBJECT NAME TO
                       7E 50 7D     041D  1525          MOVQ    R0,-(SP)                ;SAVE SIZE OF NEW OBJECT NAME PIECE
                                0420  1526                                          ;AND ADDRESS OF COLON
                 62 04 BB 50 28     0420  1527          MOVC3   R0,@4(R11),(R2)         ;SHIFT CURRENT OBJECT NAME TO MAKE ROOM
                          50 8E 7D  0425  1528          MOVQ    (SP)+,R0                ;RESTORE SAVED INFORMATION
                                0428  1529
                          6B 50 C0  0428  1530 35$:     ADDL2   R0,(R11)                ;UPDATE CURRENT OBJECT NAME SIZE
              04 BB    01 A1 50 28 042B  1531          MOVC3   R0,1(R1),@4(R11)        ;MOVE NEW OBJECT NAME PIECE INTO BUFFER
```

```
                     0431  1533
                     0431  1534 ;
                     0431  1535 ; WHEN ONE OF THE FOLLOWING CONDITIONS IS MET, ITERATIVE LOGICAL NAME
                     0431  1536 ; TRANSLATION IS TERMINATED WITHOUT ATTEMPTING TO PERFORM ANOTHER TRANSLATION.
                     0431  1537 ;
                     0431  1538 ; 1. THE SIZE OF THE CURRENT RESULTANT STRING, AFTER REMOVAL OF THE CURRENT
                     0431  1539 ;    OBJECT NAME PIECE, IS ZERO.  IN THIS CASE THERE IS NO SHARED MEMORY NAME
                     0431  1540 ;    TO BE RETURNED.  IF THERE IS ALSO NO OBJECT NAME TO BE RETURNED, THEN
                     0431  1541 ;    RETURN AN ERROR STATUS.
                     0431  1542 ;
                     0431  1543 ; 2. THE CURRENT RESULTANT STRING BEGINS WITH AN UNDERSCORE.  REMOVE THE
                     0431  1544 ;    UNDERSCORE.  IN THIS CASE THERE IS ALSO NO SHARED MEMORY NAME TO BE
                     0431  1545 ;    RETURNED.  IF THERE IS ALSO NO OBJECT NAME TO BE RETURNED, THEN RETURN AN
                     0431  1546 ;    ERROR STATUS.
                     0431  1547 ;
                     0431  1548 ; 3. THE CURRENT RESULTANT TRANSLATION IS MARKED WITH THE TERMINAL ATTRIBUTE.
                     0431  1549 ;    IN THIS CASE RETURN AN OBJECT NAME, AND IF APPROPRIATE A SHARED MEMORY
                     0431  1550 ;    NAME.
                     0431  1551 ;
                     0431  1552 ; 1. MAXIMUM LEVEL OF ITERATION HAS BEEN REACHED.  IN THIS CASE AN ERROR WILL
                     0431  1553 ;    BE RETURNED.
                     0431  1554 ;
                     0431  1555 ; IF ONE OF THE ABOVE CONDITIONS IS NOT MET, THE REMAINING RESULTANT NAME STRING
                     0431  1556 ; BECOMES THE CURRENT NAME STRING AND IS SUBJECTED TO FURTHER TRANSLATION.
                     0431  1557 ;
                     0431  1558
          04 A7  D5  0431  1559 40$:    TSTL    LWA_INPUT_DESC(R7)      ;ANY NAME AT ALL REMAINING?
             0F  13  0434  1560         BEQL    50$                     ;IF NOT THEN GO DETERMINE IF THERE IS
                     0436  1561                                         ;ANY OBJECT NAME TO BE RETURNED
                     0436  1562
12 A7  5F 8F 91  0436  1563         CMPB    #^A/_/,LWA_INPUT(R7)    ;BRANCH IF CURRENT RESULTANT NAME STRING
             10  12  043B  1564         BNEQ    60$                     ;DOESN'T BEGIN WITH AN UNDERSCORE
          08 A7  D6  043D  1565         INCL    LWA_INPUT_DESC+4(R7)    ;ELSE REMOVE " " FROM CURRENT RESULTANT
          04 A7  D7  0440  1566         DECL    LWA_INPUT_DESC(R7)      ;NAME STRING AND TERMINATE TRANSLATION
             1B  1A  0443  1567         BGTRU   STOP_TRANSLATION        ;IF THERE IS SOMETHING LEFT
                     0445  1568
             6B  D5  0445  1569 50$:    TSTL    (R11)                   ;ANY OBJECT NAME TO BE RETURNED?
             4A  13  0447  1570         BEQL    INVALID_LOGNAM          ;IF NOT THEN GO RETURN AN ERROR
             6A  D4  0449  1571         CLRL    (R10)                   ;ELSE NO SHARED MEMORY NAME TO BE
             41  11  044B  1572         BRB     TRANSLATION_DONE        ;RETURNED AND WE ARE DONE
                     044D  1573
             01  E0  044D  1574 60$:    BBS     #LNMX$V_TERMINAL,-      ;IF THE CURRENT RESULTANT TRANSLATION
                     044F  1575                 LNMX$B_FLAGS+-          ;IS MARKED WITH THE TERMINAL ATTRIBUTE
                     044F  1576                 LWA_XLATION(R7),-        ;THEN STOP THE ITERATIVE TRANSLATIONS
       0E 0D A7      044F  1577                 STOP_TRANSLATION
                     0452  1578
             59  D7  0452  1579         DECL    R9                      ;DECREMENT TRANSLATION ITERATION COUNT
             03  19  0454  1580         BLSS    65$                     ;GO RETURN ERROR IF EXCEEDED MAX DEPTH
          FF5D  31  0456  1581         BRW     TRANSLATE_LOOP          ;ELSE CONTINUE WITH CURRENT ITERATION
50   0374 8F  3C  0459  1582 65$:    MOVZWL  #SS$_TOOMANYLNAM,R0     ;MAXIMUM ITERATION DEPTH EXCEEDED
             38  11  045E  1583         BRB     RETURN                  ;GO RETURN THE APPROPRIATE ERROR
```

```
                                0460  1585
                                0460  1586 ;
                                0460  1587 ; WHEN THE ITERATIVE LOGICAL NAME TRANSLATION OF THE USER SUPPLIED INPUT STRING
                                0460  1588 ; TERMINATES THE LEFTOVER NAME STRING BECOMES THE SHARED MEMORY NAME RETURNED
                                0460  1589 ; TO THE CALLER IF AN OBJECT NAME HAD BEEN CONSTRUCTED DURING THE ITERATIVE
                                0460  1590 ; LOGICAL NAME TRANSLATION PROCESS.  OTHERWISE, THE LEFTOVER NAME STRING IS
                                0460  1591 ; RETURNED TO THE CALLER AS THE OBJECT NAME, AND THERE IS NO SHARED MEMORY NAME
                                0460  1592 ; TO BE RETURNED.
                                0460  1593 ;
                                0460  1594
                                0460  1595 STOP_TRANSLATION:                                ;STOP THE ITERATIVE TRANSLATIONS
                          6B D5 0460  1596         TSTL    (R11)                           ;DOES AN OBJECT NAME ALREADY EXIST?
                          15 12 0462  1597         BNEQ    70$                             ;IF SO THEN LEFTOVER BECOMES THE
                                0464  1598                                                 ;SHARED MEMORY NAME
                                0464  1599
                          6A D4 0464  1600         CLRL    (R10)                           ;INDICATE NO SHARED MEMORY NAME
                          5B DD 0466  1601         PUSHL   R11                             ;SWITCH THE OBJECT AND SHARED MEMORY
                       5B 5A D0 0468  1602         MOVL    R10,R11                         ;NAME POINTERS SO THAT THE LEFTOVER
                          5A 8ED0 046B  1603       POPL    R10                             ;GETS SAVED AS THE OBJECT NAME
                       6A 58 D0 046E  1604         MOVL    R8,(R10)                        ;RESTORE OBJECT NAME BUFFER SIZE TO
                                0471  1605                                                 ;THE SIZE FIELD OF ITS DESCRIPTOR
                                0471  1606
                    0C A7 95    0471  1607         TSTB    LWA_COLON(R7)                   ;COLON SEEN IN LAST RESULTANT STRING?
                          03 13 0474  1608         BEQL    70$                             ;BRANCH IF IT WASN'T; ELSE RETURN COLON
                    04 A7 D6    0476  1609         INCL    LWA_INPUT_DESC(R7)              ;AS PART OF OBJECT NAME STRING
                                0479  1610
                 50 04 A7 D0    0479  1611 70$:     MOVL    LWA_INPUT_DESC(R7),R0           ;SIZE OF STRING TO BE RETURNED
                    6A 50 D1    047D  1612         CMPL    R0,(R10)                        ;DOES STRING SIZE EXCEED BUFFER SIZE?
                          11 1A 0480  1613         BGTRU   INVALID_LOGNAM                  ;RETURN ERROR IF SO
                          50 2C 0482  1614         MOVC5   R0,-
                                0484  1615                 @LWA_INPUT_DESC+4(R7),-
04 BA 6A 00 08 B7               0484  1616                 #0,(R10),@4(R10)                ;MOVE NAME STRING, ZERO FILLED
                    6A 04 A7 D0 048A  1617         MOVL    LWA_INPUT_DESC(R7),(R10);STORE STRING'S LENGTH
                                048E  1618
                                048E  1619 ;
                                048E  1620 ; SETUP THE APPROPRIATE RETURN STATUS, AND RETURN TO THE CALLER AFTER
                                048E  1621 ; DEALLOCATING THE KRP BACK TO THE KRP LOOKASIDE LIST.
                                048E  1622 ;
                                048E  1623
                                048E  1624 TRANSLATION_DONE:                               ;TRANSLATIONS HAVE COMPLETED
                       50 01 D0 048E  1625         MOVL    #SS$_NORMAL,R0                  ;SET APPROPRIATE STATUS
                          05 11 0491  1626         BRB     RETURN                          ;RETURN STATUS
                                0493  1627
                                0493  1628 INVALID_LOGNAM:                                 ;REPORT AN INVALID LOGICAL NAME
                    50 0154 8F 3C 0493  1629         MOVZWL  #SS$_IVLOGNAM,R0                ;SET APPROPRIATE ERROR CODE
                                0498  1630
                56 00000000'GF 9E 0498  1631 RETURN: MOVAB   G^CTL$GL_KRPFL,R6               ;RETRIEVE ADDRESS OF KRP QUEUE LISTHEAD
                    04 B6 67 0E 049F  1632         INSQUE  (R7),@4(R6)                     ;INSERT KRP INTO LIST
                                04A3  1633
                       0FFE 8F BA 04A3  1634         POPR    #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;RESTORE REGISTERS
                          05 04A7  1635         RSB                                     ;RETURN STATUS
                                04A8  1636         .DSABL  LSB
```

```
                     04A8  1638          .SBTTL  MMG$READ_GSD/MMG$WRITE_GSD - READ/WRITE SHARED MEM GBL SECTION
                     04A8  1639
                     04A8  1640 ;++
                     04A8  1641 ; FUNCTIONAL DESCRIPTION:
                     04A8  1642 ;
                     04A8  1643 ; THIS ROUTINE READS THE PAGES OF A GLOBAL SECTION BEING CREATED INTO
                     04A8  1644 ; SHARED MEMORY OR WRITES THE PAGES BACK TO A DISK FILE.
                     04A8  1645 ;
                     04A8  1646 ; CALLING SEQUENCE:
                     04A8  1647 ;
                     04A8  1648 ;       BSBW    MMG$READ_GSD
                     04A8  1649 ;       BSBW    MMG$WRITE_GSD
                     04A8  1650 ;
                     04A8  1651 ; INPUT PARAMETERS:
                     04A8  1652 ;
                     04A8  1653 ;       R6 = GLOBAL SECTION DESCRIPTOR ADDRESS
                     04A8  1654 ;       R2 = STARTING VIRTUAL ADDRESS INTO WHICH SECTION IS MAPPED
                     04A8  1655 ;            (MMG$READ_GSD ONLY)
                     04A8  1656 ;       R3 = ENDING VIRTUAL ADDRESS INTO WHICH SECTION IS MAPPED
                     04A8  1657 ;            (MMG$READ_GSD ONLY)
                     04A8  1658 ;       4(SP) = RETURN STATUS CODE SO FAR FOR $CRMPSC SYSTEM SERVICE
                     04A8  1659 ;                    (MMG$READ_GSD ONLY)
                     04A8  1660 ;
                     04A8  1661 ; IMPLICIT INPUTS:
                     04A8  1662 ;
                     04A8  1663 ;       THE GSD IS FULLY INITIALIZED AS WELL THE SECTION TABLE ENTRY (IF
                     04A8  1664 ;       THERE IS ONE).
                     04A8  1665 ;
                     04A8  1666 ; OUTPUT PARAMETERS:
                     04A8  1667 ;
                     04A8  1668 ;       R0 CONTAINS THE STATUS CODE FOR THE I/O TRANSFER.
                     04A8  1669 ;
                     04A8  1670 ; IMPLICIT OUTPUTS:
                     04A8  1671 ;
                     04A8  1672 ;       THE GLOBAL SECTION IS READ/WRITTEN.
                     04A8  1673 ;
                     04A8  1674 ; COMPLETION CODES:
                     04A8  1675 ;
                     04A8  1676 ;       SS$_NORMAL - SUCCESSFUL COMPLETION
                     04A8  1677 ;       VARIOUS SYSTEM SERVICE FAILURE CODES.
                     04A8  1678 ;
                     04A8  1679 ; SIDE EFFECTS:
                     04A8  1680 ;
                     04A8  1681 ;       NONE
                     04A8  1682 ;
                     04A8  1683 ;--
                     04A8  1684
           00000020  04A8  1685          MAXIO = 32                              ;MAXIMUM # PAGES IN ONE I/O
                     04A8  1686
                     04A8  1687
                     04A8  1688 MMG$WRITE_GSD::
                     04A8  1689          .ENABL  LSB
OFFE 8F    BB       04A8  1690          PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;SAVE REGISTERS
57   01    D0       04AC  1691          MOVL    #1,R7                           ;INDICATE GS IS BEING WRITTEN
     0D    11       04AF  1692          BRB     5$                              ;JOIN COMMON CODE
                     04B1  1693
                     04B1  1694 MMG$READ_GSD::
```

```
                50   04 AE   D0  04B1  1695          MOVL    4(SP),R0                        ;GET RETURN CODE SO FAR
                   60 50   E9  04B5  1696          BLBC    R0,50$                          ;BR IF ERROR CREATING SECTION
                  0FFE 8F   BB  04B8  1697          PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;SAVE REGISTERS
                      57    D4  04BC  1698          CLRL    R7                              ;INDICATE GS IS BEING READ
                   5B   04   9A  04BE  1699 5$:      MOVZBL  #GSD$C_PFNBASMAX,R11            ;SET COUNT OF PFN BASES IN GSD
        5A   56  00000054 8F   C1  04C1  1700          ADDL3   #GSD$L_BASPFN1,R6,R10           ;GET ADR OF 1ST PFN BASE IN GSD
             59 20 A6    02   E1  04C9  1701          BBC     #SEC$V_DZRO,GSD$W_FLAGS(R6),100$ ;BR IF SECTION MUST BE READ IN
                   56 57   E8  04CE  1702          BLBS    R7,100$                         ;BR IF WRITING SECTION TO DISK
                               04D1  1703 ;
                               04D1  1704 ; THE SECTION IS DEMAND-ZERO.  INITIALIZE THE PAGES TO ALL ZEROS.
                               04D1  1705 ;
                               04D1  1706 ; R10 = ADDRESS OF NEXT PFN BASE IN GSD
                               04D1  1707 ; R11 = NUMBER OF PFN BASES IN GSD
                               04D1  1708 ;
                      57   52  7D  04D1  1709          MOVQ    R2,R7                           ;GET START AND END VA
                7E    0200 8F   3C  04D4  1710          MOVZWL  #^X200,-(SP)                    ;SET VA INCREMENT
                7E    58   57   C3  04D9  1711          SUBL3   R7,R8,-(SP)                     ;GET # BYTES MAPPED
                           06   18  04DD  1712          BGEQ    6$                              ;BR IF RANGE MAPPED FORWARDS
                      6E   6E   CE  04DF  1713          MNEGL   (SP),(SP)                       ;CONVERT TO POSITIVE BYTE COUNT
                      57   58   D0  04E2  1714          MOVL    R8,R7                           ;REVERSE STARTING ADR FOR MOVC
             6E       6E     F7 8F   78  04E5  1715 6$:      ASHL    #-9,(SP),(SP)           ;CONVERT FROM BYTE TO PAGE COUNT
                           6E   D6  04EA  1716          INCL    (SP)                            ;ACTUAL # OF PAGES MAPPED
                               04EC  1717          ASSUME  GSD$L_BASCNT1 EQ <GSD$L_BASPFN1 + 4>
                      59   8A   D0  04EC  1718 10$:     MOVL    (R10)+,R9                       ;NEXT PFN BASE IN GSD
                      59   8A   D0  04EF  1719          MOVL    (R10)+,R9                       ;NEXT BASE CNT IN GSD
                           17   13  04F2  1720          BEQL    25$                             ;BR ON NO MORE PAGES TO INIT
                      6E   59   C2  04F4  1721          SUBL    R9,(SP)                         ;IS THIS PIECE MAPPED?
                           20   19  04F7  1722          BLSS    NOT_MAPPED                      ;BR ON ERROR, NOT MAPPED
    67   0200 8F   00    66   00   2C  04F9  1723 20$:     MOVC5   #0,(R6),#0,#^X200,(R7)          ;ZERO-FILL A PAGE
                   57   04 AE   C0  0501  1724          ADDL2   4(SP),R7                        ;GET VA OF NEXT PAGE TO INIT
                     F1 59   F5  0505  1725          SOBGTR  R9,20$                          ;REPEAT FOR EACH PAGE IN PIECE
                     E1 5B   F5  0508  1726          SOBGTR  R11,10$                         ;REPEAT FOR EACH PIECE OF GS
                   5E   04   C0  050B  1727 25$:     ADDL2   #4,SP                           ;CLEAN OFF # PAGES MAPPED
                   50   01   9A  050E  1728 30$:     MOVZBL  #SS$_NORMAL,R0                  ;REPORT SUCCESS
                   5E   04   C0  0511  1729 35$:     ADDL2   #4,SP                           ;CLEAN OFF INCREMENT
                   0FFE 8F   BA  0514  1730 40$:     POPR    #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ;RESTORE REGISTERS
                         05    05  0518  1731 50$:     RSB
                               0519  1732
                               0519  1733 NOT_MAPPED:
          50   036C 8F   3C  0519  1734          MOVZWL  #SS$_SHMGSNOTMAP,R0             ;DZRO SECTION MUST BE MAPPED, TO
            38 AE   50   D0  051E  1735          MOVL    R0,<T4*4>(SP)                   ;ERROR CODE TO RETURN TO CALLER
               5E   04   C0  0522  1736          ADDL2   #4,SP                           ;CLEAN OFF # PAGES MAPPED
                    EA   11  0525  1737          BRB     35$                             ;ALLOW INIT. DURING CREATION
                               0527  1738
                               0527  1739 ;
                               0527  1740 ; THE SECTION WAS NOT DEMAND-ZERO, THEREFORE IT MUST BE MAPPED TO A FILE.
                               0527  1741 ; (PFN MAPPED SECTIONS ARE NEVER INITIALIZED AND THUS NEVER REACH THIS CODE.)
                               0527  1742 ; THE PAGES MUST BE READ FROM THE FILE INTO SHARED MEMORY BEFORE A STATUS
                               0527  1743 ; CODE CAN BE RETURNED TO THE CALLER OF $CRMPSC.
                               0527  1744 ;
                               0527  1745 ;
                               0527  1746 ; FIRST GET THE NEEDED PARAMETERS FROM THE SECTION TABLE ENTRY. (ALL GLOBAL
                               0527  1747 ; SECTIONS MAPPED TO A FILE, HAVE A SECTION TABLE ENTRY IN THE SYSTEM PROCESS
                               0527  1748 ; HEADER.)  THESE PARAMETERS INCLUDE THE WINDOW ADDRESS, VIRTUAL BLOCK NUMBER,
                               0527  1749 ; PAGE FAULT CLUSTER SIZE FOR THE SECTION.
                               0527  1750 ;
    00000113'EF                16  0527  1751 100$:    JSB     MMG$FINDSHD                     ;GET SHD AND SHB ADDRS
```

```
                10 A4   DD  052D  1752          PUSHL   SHB$L_BASGSPFN(R4)                ;REMEMBER BASE PFN OF SHM
          51    16 A6   32  0530  1753          CVTWL   GSD$W_GSTX(R6),R1                 ;GET SECTION TABLE INDEX
     50   00000000'GF   D0  0534  1754          MOVL    G^MMG$GL_SYSPHD,R0                ;GET SYSTEM PROCESS HEADER
          50    20 A0   C0  053B  1755          ADDL2   PHD$L_PSTBASOFF(R0),R0            ;GET BASE ADR OF PROC SEC TBL
          51     6041   DE  053F  1756          MOVAL   (R0)[R1],R1                       ;GET ADR OF SECTION TABLE ENTRY
          52    0C A1   D0  0543  1757          MOVL    SEC$L_WINDOW(R1),R2               ;GET ADR OF WINDOW
          50    10 A1   D0  0547  1758          MOVL    SEC$L_VBN(R1),R0                  ;GET FIRST VBN MAPPED
          56    0B A1   9A  054B  1759          MOVZBL  SEC$B_PFC(R1),R6                  ;GET PAGE FAULT CLUSTER FOR GS
                        054F  1760 ;
                        054F  1761 ; NOW COMPUTE THE SIZE OF THE I/O REQUEST TO BE MADE.  THIS IS LIMITED BY
                        054F  1762 ; (1) THE SIZE OF THE PIECE OF SECTION BEING INITIALIZED, (2) THE PAGE FAULT
                        054F  1763 ; CLUSTER SIZE OF THE SECTION, AND (3) THE MAXIMUM I/O REQUEST ALLOWED BY THE
                        054F  1764 ; SYSTEM.  THE LARGEST I/O POSSIBLE IS ALLOWED.  (REMEMBER THAT SHARED MEMORY
                        054F  1765 ; SECTIONS MAY BE MAPPED IN UP TO #GSD$C_PFNBASMAX PIECES OF CONSECUTIVE PAGES.)
                        054F  1766 ;
                        054F  1767          ASSUME  GSD$L_BASCNT1 EQ <GSD$L_BASPFN1 + 4>
     58    6E    8A   C1  054F  1768 110$:   ADDL3   (R10)+,(SP),R8                     ;BASE PFN OF NEXT PIECE
           59    8A   D0  0553  1769          MOVL    (R10)+,R9                         ;CNT OF PAGES IN NEXT PIECE
                 B6   13  0556  1770          BEQL    30$                               ;BR IF NO MORE PAGES TO READ/WRT
           51    56   D0  0558  1771 120$:   MOVL    R6,R1                             ;ASSUME READ SIZE IS PFC SIZE
                 05   13  055B  1772          BEQL    130$                              ;BR IF NO PFC SPECIFIED
           51    59   D1  055D  1773          CMPL    R9,R1                             ;IS PIECE > CLUSTER SIZE?
                 03   14  0560  1774          BGTR    140$                              ;BR IF PIECE GREATER
           51    59   D0  0562  1775 130$:   MOVL    R9,R1                             ;PIECE IS SMALLER, USE PIECE SIZ
           20    51   D1  0565  1776 140$:   CMPL    R1,#MAXIO                         ;IS READ SIZE > MAXIMUM I/O?
                 03   19  0568  1777          BLSS    150$                              ;BR IF READ SIZE IS OK
           51    20   3C  056A  1778          MOVZWL  #MAXIO,R1                         ;READ MAXIMUM SIZE I/O ALLOWED
     51    51    09   78  056D  1779 150$:   ASHL    #9,R1,R1                          ;CONVERT PAGES TO BYTES
     54   00000000'GF   D0  0571  1780          MOVL    G^SCH$GL_CURPCB,R4                ;CURRENT PROCESS CONTROL BLOCK
          55    6C A4   D0  0578  1781          MOVL    PCB$L_PHD(R4),R5                  ;CURRENT PROCESS HEADER ADR
                        057C  1782 ;
                        057C  1783 ; NOW ALLOCATE ONE PACKET THAT WILL CONTAIN AN IRP AND A LIST OF PAGE
                        057C  1784 ; TABLE ENTRIES, DESCRIBING THE RANGE OF PHYSICAL PAGES TO BE READ/WRITTEN.
                        057C  1785 ; THE PTE'S MUST BE CREATED AS THE PAGES MAY NOT BE MAPPED TO VIRTUAL
                        057C  1786 ; ADDRESSES.  THE PTE'S MUST BE IN THE SAME BLOCK OF NON-PAGED POOL AS
                        057C  1787 ; THE IRP, OTHERWISE THE PROCESS MIGHT BE DELETED AND THE POOL SPACE FOR
                        057C  1788 ; THE PTE'S LOST.  THE I/O SYSTEM WILL RELEASE THE IRP IF THE PROCESS IS
                        057C  1789 ; DELETED.
                        057C  1790 ;
                 07   BB  057C  1791          PUSHR   #^M<R0,R1,R2>                     ;SAVE WINDOW ADDRESS, CNT & VBN
     51    51    F9 8F   78  057E  1792          ASHL    #-7,R1,R1                         ;# OF BYTES OF PTE NEEDED
     7E    51    FE 8F   78  0583  1793          ASHL    #-2,R1,-(SP)                      ;# OF PTE'S TO BE CREATED
     51   000000C4 8F   C0  0588  1794          ADDL2   #IRP$C_LENGTH,R1                  ;ADD IN SIZE OF I/O PACKET
          00000000'GF   16  058F  1795          JSB     G^EXE$ALONONPAGED                 ;ALLOCATE NONPAGED PACKET
                 03   50  E8  0595  1796          BLBS    R0,155$                           ;SUCCESSFUL
                 0084 31  0598  1797          BRW     NO_IRP                            ;BR IF UNABLE TO GET PACKET
           55    52   D0  059B  1798 155$:   MOVL    R2,R5                             ;SET PACKET ADR FOR EXE$BLDPKT
     0A    A5    0A   90  059E  1799          MOVB    #DYN$C_IRP,IRP$B_TYPE(R5)         ;INDICATE THAT IT IS IRP
     08    A5    51   B0  05A2  1800          MOVW    R1,IRP$W_SIZE(R5)                 ;SET SIZE OF PACKET ALLOCATED
                 51 8ED0  05A6  1801          POPL    R1                                ;GET # OF PTE'S TO CREATE
     52   00C4   C2   9E  05A9  1802          MOVAB   <IRP$C_LENGTH+^X3>&<^C<^X3>>(R2),R2 ;LONGWORD ALIGN ADR FOR PTE
          53    52   D0  05AE  1803          MOVL    R2,R3                             ;REMEMBER FIRST SVAPTE
                        05B1  1804 ;
                        05B1  1805 ; R1 = SIZE OF PACKET ALLOCATED IN BYTES
                        05B1  1806 ; R2 = LONGWORD ALIGNED ADDRESS FOR FIRST SVAPTE TO BE CREATED
                        05B1  1807 ; R5 = ADDRESS OF PACKET ALLOCATED
                        05B1  1808 ; R8 = NEXT PFN TO BE READ/WRITTEN
```

```
                              05B1  ;
58   B0000000 8F        C8    05B1          BISL2   #<PTE$C_ERKW ! PTE$M_VALID>,R8  ;SET OWNER AND VALID IN PTE
              82  88    9E    05B8  160$:   MOVAB   (R8)+,(R2)+                     ;SET ONE PTE
              FA  51    F5    05BB          SOBGTR  R1,160$                         ;LOOP FOR SIZE OF TRANSFER
58   B0000000 8F        CA    05BE          BICL2   #<PTE$C_ERKW ! PTE$M_VALID>,R8  ;CLEAR OWNER AND VALID BITS
                              05C5  ;
                              05C5  ; FINALLY, INITIALIZE THE I/O REQUEST PACKET (IRP) ITSELF.  A LOCATION ON
                              05C5  ; THE STACK IS ALLOCATED TO HOLD THE I/O COMPLETION STATUS CODE.  THE I/O
                              05C5  ; COMPLETION AST ROUTINE WILL MOVE THE STATUS CODE INTO THIS LOCATION AND
                              05C5  ; DELETE THE IRP.
                              05C5  ;
                  07    BA    05C5          POPR    #^M<R0,R1,R2>                   ;GET WINDOW ADR, CNT & VBN
                  07    BB    05C7          PUSHR   #^M<R0,R1,R2>                   ;SAVE BYTE CNT & WINDOW ADR
                  7E    7C    05C9          CLRQ    -(SP)                           ;INITIALIZE I/O RETURN STATUS
          24 A5   5E    D0    05CB          MOVL    SP,IRP$L_IOSB(R5)               ;SET ADR FOR RETURN STATUS
     14 A5   40'AF      9E    05CF          MOVAB   B^SHMIODONE,IRP$L_ASTPRM(R5)    ;SET AST ROUTINE ADR
     23 A5   2F A4      90    05D4          MOVB    PCB$B_PRIB(R4),IRP$B_PRI(R5)    ;SET PRIORITY FOR I/O
                              05D9  ;
                              05D9  ; THE INPUTS FOR EXE$BLDPKTGSR/EXE$BLDPKTGSW ARE:
                              05D9  ;       R0 = VBN
                              05D9  ;       R1 = NUMBER OF BYTES TO TRANSFER
                              05D9  ;       R2 = WINDOW ADDRESS
                              05D9  ;       R3 = SVAPTE
                              05D9  ;       R4 = PCB ADDRESS
                              05D9  ;       R5 = IRP ADDRESS
                              05D9  ;
                              05D9  ; IT DESTROYS R0, R1, R2, R3, R4 AND R5.
                              05D9  ;
               08 57    E9    05D9          BLBC    R7,185$                         ;BR IF READING SHM PAGES
         00000000'GF    16    05DC          JSB     G^EXE$BLDPKTGSW                 ;GO BUILD & SUBMIT WRITE REQUEST
                  06    11    05E2          BRB     190$                            ;JOIN COMMON CODE
         00000000'GF    16    05E4  185$:   JSB     G^EXE$BLDPKTGSR                 ;GO BUILD & SUBMIT READ REQUEST
                              05EA  ;
                              05EA  ; NOW WAIT FOR THE I/O REQUEST TO COMPLETE.  THIS IS ACCOMPLISHED BY WAITING
                              05EA  ; FOR AN I/O COMPLETION STATUS CODE TO BE SET BY THE AST ROUTINE.  THIS CODE
                              05EA  ; MAY OR MAY NOT BE SET BEFORE THE WAIT STATE IS ENTERED.  THE WAIT STATE
                              05EA  ; MAY ALSO BE LEFT FOR THE WRONG REASON.  THEREFORE, THE STATUS CODE MUST BE
                              05EA  ; CHECKED BEFORE WAITING AND UPON AWAKENING.  THE WAIT STATE IS PAGE FAULT WAIT.
                              05EA  ;
                              05EA  ;
                              05EA  ; ******* THERE IS A PROBLEM HERE.  LOWERING IPL SO AS TO RECEIVE THE AST
                              05EA  ; ******* WILL ALLOW THE PROCESS CREATING THE SHM GS TO BE DELETED WHILE
                              05EA  ; ******* IT HOLDS AN UNFINISHED GSD.
                              05EA  ;
                  00    DD    05EA  190$:   PUSHL   #0                              ;LOWER IPL TO RECEIVE AST'S
                              05EC  195$:   SETIPL  SYNCHIPL                        ;RAISE IPL TO SYNCH AND INSURE
                              05F3                                                  ;THAT CODE IS FAULTED INTO MEM
               04 AE    D5    05F3          TSTL    4(SP)                           ;CHECK IF I/O STATUS CODE IS SET
                  17    12    05F6          BNEQ    200$                            ;BR IF I/O REQUEST IS COMPLETE
52       00000000'GF    7E    05F8          MOVAQ   G^SCH$GQ_PFWQ,R2                ;SET ADR OF PAGE FAULT WAIT QUE
54       00000000'GF    D0    05FF          MOVL    G^SCH$GL_CURPCB,R4              ;SET ADR OF CURRENT PROC CTL BLK
         00000000'GF    16    0606          JSB     G^SCH$WAITK                     ;WAIT ON A KERNEL AST
                  DC    11    060C          BRB     190$                            ;CHECK IF AST WAS FOR THIS I/O
                              060E  REI_RTN1:
                  02          060E          REI                                     ;SET NEW PSL AND PC FROM STACK
                  FD    10    060F  200$:   BSBB    REI_RTN1                        ;RESTORE TO PSL BEFORE WAIT
          50      8E    D0    0611          MOVL    (SP)+,R0                        ;GET I/O COMPLETION CODE
```

```
          08 50  E9 0614            BLBC    R0,IO_FAIL                          ;BR IF I/O FAILED
             8E  D5 0617            TSTL    (SP)+                               ;CLEAN OFF STACK
             0A  11 0619            BRB     220$                                ;CONTINUE
                    061B  ;
                    061B  ; PLACING THE SYNCH IPL IN A LONGWORD AT THIS LOCATION WILL FORCE THE ABOVE
                    061B  ; SETIPL INSTRUCTION TO FAULT INTO MEMORY ALL INSTRUCTIONS BETWEEN IT AND THIS
                    061B  ; LONGWORD.  THIS IS NECESSARY BECAUSE THIS CODE RESIDES IN A PAGEABLE PSECT
                    061B  ; RUNS AT RAISED IPL, AND PAGE FAULTS CANNOT BE ALLOWED AT RAISED IPL.
                    061B  ; THE ASSUME MACRO GUARANTEES THAT THE SETIPL INSTRUCTION AND THE IPL
                    061B  ; LONGWORD ARE ON ADJACENT PAGES.
                    061B  ;
                    061B  SYNCHIPL:
          00000008  061B            .LONG   IPL$_SYNCH                          ;SYNCH IPL
                    061F  205$:
                    061F            ASSUME  <205$ - 195$> LE 512                ;GUARANTEE PAGE ADJACENCY
                    061F  ;
                    061F  ; THE I/O TO INITIALIZE THE GLOBAL SECTION FAILED.
                    061F  ;
                    061F  IO_FAIL:
                    061F  ;
                    061F  ; UNABLE TO ALLOCATE AN IRP.  RETURN ERROR STATUS CODE.
                    061F  ;
                    061F  NO_IRP:
       5E    10  C0 061F            ADDL2   #<4*4>,SP                           ;WIND, CNT, VBN, PTE, & BAS PFN
           FEEC  31 0622            BRW     35$                                 ;ERROR EXIT
                    0625  ;
                    0625  ; I/O REQUEST COMPLETED SUCCESSFULLY.  NOW SET UP TO DO THE NEXT
                    0625  ; PAGES OF THE GLOBAL SECTION.  THESE PAGES MAY BE IN THE SAME PIECE, (I.E.,
                    0625  ; HAVE THE SAME BASE PFN) OR THEY MAY BE PART OF THE NEXT PIECE OF THE SECTION.
                    0625  ; THE ENTIRE SECTION MAY NOW BE MAPPED, TOO.  THE PARAMETERS TO BE INITIALIZED
                    0625  ; ARE:  (1) PFN, (2) VIRTUAL BLOCK NUMBER, AND (3) NUMBER OF PAGES
                    0625  ; LEFT TO MAP IN THIS PIECE.
                    0625  ;
             07  BA 0625  220$:     POPR    #^M<R0,R1,R2>                       ;RESTORE REGISTERS
    51   51 F7 8F  78 0627          ASHL    #-9,R1,R1                           ;GET # OF PAGES READ/WRITTEN
          50 51  C0 062C            ADDL2   R1,R0                               ;GET NEXT VBN TO BE READ/WRITTEN
          59 51  C2 062F            SUBL2   R1,R9                               ;GET # PAGES IN PIECE TO XFER
             03  13 0632            BEQL    250$                                ;BR IF ALL OF THIS PIECE IS DONE
           FF21  31 0634            BRW     120$                                ;BR IF MORE OF PIECE TO READ
          03 5B  F5 0637  250$:     SOBGTR  R11,260$                            ;BR TO GET NEXT PIECE OF GS
           FED1  31 063A            BRW     30$                                 ;BR IF NO MORE PIECES TO READ
           FF0F  31 063D  260$:     BRW     110$                                ;GO GET NEXT BASE PFN/CNT
                    0640  ;
                    0640  ; THIS IS THE AST ROUTINE CALLED WHEN I/O IS COMPLETED TO SHARED MEMORY.
                    0640  ; IT SETS THE COMPLETION STAUS CODE INTO A STACK ADDRESS FOR THE I/O
                    0640  ; REQUESTOR TO CHECK.  THE IRP IS THEN DELETED.
                    0640  ;
                    0640  SHMIODONE:
                    0640  270$:     DSBINT  NEWIPL                              ;DISABLE INTERRUPTS & PAGEFAULTS
    24 B5    38 A5  D0 064A         MOVL    IRP$L_IOST1(R5),@IRP$L_IOSB(R5) ;SET I/O COMPLETION STATUS CODE
          50 55  D0 064F            MOVL    R5,R0                               ;SET ADR OF IRP
    00000000'GF  16 0652            JSB     G^EXE$DEANONPAGED                   ;DEALLOCATE THE IRP
 54 00000000'GF  D0 0658            MOVL    G^SCH$GL_CURPCB,R4                  ;GET ADR OF CURRENT PCB
          52 01  9A 065F            MOVZBL  #PRI$_IOCOM,R2                      ;SET I/O COMPLETION STATE CODE
                    0662            RPTEVT  PFCOM,CALL_TYPE=JSB                 ;REPORT PAGEFAULT COMPLETE EVENT
                    0669            ENBINT                                      ;ENABLE INTERRUPTS
             05     066C            RSB                                         ;RETURN FROM AST
```

```
                     066D  1923 ;
                     066D  1924 ; PLACING THE SYNCH IPL IN A LONGWORD AT THIS LOCATION WILL FORCE THE ABOVE
                     066D  1925 ; SETIPL INSTRUCTION TO FAULT INTO MEMORY ALL INSTRUCTIONS BETWEEN IT AND THIS
                     066D  1926 ; LONGWORD.  THIS IS NECESSARY BECAUSE THIS CODE RESIDES IN A PAGEABLE PSECT
                     066D  1927 ; AND PAGE FAULTS CANNOT BE ALLOWED AT RAISED IPL.
                     066D  1928 ;
                     066D  1929 NEWIPL:
00000008             066D  1930          .LONG   IPL$_SYNCH                      ;SYNCH IPL
                     0671  1931 300$:
                     0671  1932          ASSUME  <300$ - 270$> LE 512            ;GUARANTEE PAGE ADJACENCY
                     0671  1933          .DSABL  LSB
```

```
0671  1935          .SBTTL  MMG$FINDGSNOTRN - FIND GSD WITHOUT LOGICAL NAME TRANSLATION
0671  1936
0671  1937  ;++
0671  1938  ; FUNCTIONAL DESCRIPTION:
0671  1939  ;
0671  1940  ; THIS ROUTINE IS CALLED BY $MGBLSC AND $DGBLSC WHEN THEY CANNOT FIND A GLOBAL
0671  1941  ; SECTION VIA THE NORMAL SEARCH PATH.  IF A SPECIFIC SHARED MEMORY WAS BEING
0671  1942  ; SEARCHED, THE SECTION MIGHT NOT BE IN THAT MEMORY.  IF IT IS A COPY-ON-
0671  1943  ; REFERENCE SECTION, IT WILL HAVE BEEN PLACED IN LOCAL MEMORY.  THIS ROUTINE
0671  1944  ; CHECKS TO SEE IF THIS HAS OCCURRED.  IF THE SEARCH WAS IN A SPECIFIC SHARED
0671  1945  ; MEMORY, THE RESULTANT GLOBAL SECTION NAME PREFIXED BY AN UNDERSCORE (CAUSING
0671  1946  ; NO FURTHER LOGICAL NAME TRANSLATION) IS USED IN A SECOND SEARCH; THIS SEARCH
0671  1947  ; STARTING IN LOCAL MEMORY.
0671  1948  ;
0671  1949  ; CALLING SEQUENCE:
0671  1950  ;
0671  1951  ;       BSBW    MMG$FINDGSNOTRN
0671  1952  ;
0671  1953  ; INPUT PARAMETERS:
0671  1954  ;
0671  1955  ;       R7 - ADDRESS OF A SCRATCH AREA CONTAINING THE RESULTANT ASCIC GLOBAL
0671  1956  ;            SECTION NAME FOLLOWED BY THE IDENT QUADWORD
0671  1957  ;       R9 - SECTION FLAGS SPECIFIED BY USER
0671  1958  ;       R10 - 0 IF THE GSD WAS FOUND IN LOCAL MEMORY
0671  1959  ;             -1 IF THE LOCAL MEMORY SEARCH EXTENDED INTO SHARED MEMORY TABLES
0671  1960  ;             >0 IF A SPECIFIC SHARED MEMORY NAME WAS SPECIFIED
0671  1961  ;
0671  1962  ; IMPLICIT INPUTS:
0671  1963  ;
0671  1964  ;       NONE
0671  1965  ;
0671  1966  ; OUTPUT PARAMETERS:
0671  1967  ;
0671  1968  ;       R0 - RETURN STATUS CODE
0671  1969  ;       R6 - GSD ADDRESS, IF FOUND
0671  1970  ;       R10 - 0 IF THE GSD WAS FOUND IN LOCAL MEMORY
0671  1971  ;             -1 IF THE LOCAL MEMORY SEARCH EXTENDED INTO SHARED MEMORY TABLES
0671  1972  ;             >0 IF A SPECIFIC SHARED MEMORY NAME WAS SPECIFIED
0671  1973  ;
0671  1974  ; IMPLICIT OUTPUTS:
0671  1975  ;
0671  1976  ;       THE PREVIOUS MODE IS SET TO THE CURRENT MODE TO ALLOW THE DESCRIPTORS
0671  1977  ;       AND BUFFERS WHICH ARE ON THE STACK TO BE PROBED.
0671  1978  ;
0671  1979  ; COMPLETION CODES:
0671  1980  ;
0671  1981  ;       SS$_NORMAL - SUCCESSFUL COMPLETION
0671  1982  ;       SS$_NOSUCHSEC - NO SUCH GLOBAL SECTION
0671  1983  ;       SS$_IVLOGNAM - INVALID LOGICAL NAME
0671  1984  ;       SS$_ACCVIO - ACCESS VIOLATION
0671  1985  ;
0671  1986  ; SIDE EFFECTS:
0671  1987  ;
0671  1988  ;       NONE
0671  1989  ;
0671  1990  ;--
0671  1991
```

```
                                0671  1992
                                0671  1993 MMG$FINDGSNOTRN::
                          5A    D5  0671  1994          TSTL    R10                             ;SPECIFIC SHARED MEMORY SEARCH?
                          3A    15  0673  1995          BLEQ    10$                             ;BR IF NOT SPEC MEM SEARCH
                                0675  1996 ;
                                0675  1997 ; THE ROUTINE THAT DOES A GSD TABLE SCAN PROBES THE NAME BUFFER AND THE IDENT
                                0675  1998 ; QUADWORD FROM THE PREVIOUS MODE.  SINCE THESE AREAS ARE NOW ON THE KERNEL
                                0675  1999 ; STACK AND THE PREVIOUS MODE IS PROBABLY USER, IT IS NECESSARY TO MAKE THE
                                0675  2000 ; PREVIOUS MODE BE KERNEL.  NOTE:  NO OTHER PROBES OF USER PROVIDED DATA ARE
                                0675  2001 ; DONE BY $CRMPSC OR $DGBLSC SYSTEM SERVICES FROM HERE ON.
                                0675  2002 ;
                          7E    DC  0675  2003          MOVPSL  -(SP)                           ;GET CURRENT PSL
                          7E    DC  0677  2004          MOVPSL  -(SP)                           ;GET CURRENT PSL, AGAIN !!!
                50  6E  02    18  EF  0679  2005          EXTZV   #PSL$V_CURMOD,#PSL$S_CURMOD,(SP),R0 ;EXTRACT CURRENT MODE
                6E  02  16    50  F0  067E  2006          INSV    R0,#PSL$V_PRVMOD,#PSL$S_PRVMOD,(SP) ;SET PREV MODE TO CUR MODE
                              2B  10  0683  2007          BSBB    REI_ROUTINE                     ;FORCE PREV MODE TO BE CUR MODE
                                      0685  2008
                              3E  BB  0685  2009          PUSHR   #^M<R1,R2,R3,R4,R5>             ;SAVE REGISTERS
                          5E  2C  C2  0687  2010          SUBL    #<11*4>,SP                      ;BUFFER FOR NEW NAME STRING
                01 AE     01  A7  2B  28  068A  2011          MOVC3   #43,1(R7),1(SP)             ;COPY RESULTANT NAME STRING
                        6E    5F  8F  90  0690  2012          MOVB    #^A/_/,(SP)                 ;FORCE NO LOG NAM TRANS
                          5E      DD  0694  2013          PUSHL   SP                              ;SET ADR IN STR DESC
                        7E    67  9A  0696  2014          MOVZBL  (R7),-(SP)                      ;SET SIZ IN STR DESC
                          6E      D6  0699  2015          INCL    (SP)                            ;ADD IN ONE UNDERSCORE CHAR
                        50    5E  D0  069B  2016          MOVL    SP,R0                           ;SET ADR OF STR DESC
                        56    59  D0  069E  2017          MOVL    R9,R6                           ;SET SECTION FLAGS
                      51    2C  A7  9E  06A1  2018          MOVAB   44(R7),R1                   ;SET MATCH CONTROL INFO
                              F958'  30  06A5  2019          BSBW    MMG$GSDSCN                  ;GO SEARCH AGAIN
                          5E  34  C0  06A8  2020          ADDL2   #<11*4>+8,SP                    ;RELEASE BUFFER AND STR DESC
                              3E  BA  06AB  2021          POPR    #^M<R1,R2,R3,R4,R5>             ;RESTORE REGISTERS
                              01  10  06AD  2022          BSBB    REI_ROUTINE                     ;RESTORE ORIGINAL PREVIOUS MODE
                              05  06AF  2023 10$:        RSB                                     ;RETURN STATUS OF SEARCH
                                  06B0  2024
                                  06B0  2025 REI_ROUTINE:
                              02  06B0  2026          REI                                     ;THIS WILL ALLOW A NEW MODE
                                  06B1  2027                                                  ;TO BE SET FROM THE STACK
```

```
                     06B1  2029          .SBTTL  MMG$UNIQUEGSD - CHECK THAT SH MEM GSD IS UNIQUE
                     06B1  2030
                     06B1  2031 ;++
                     06B1  2032 ; FUNCTIONAL DESCRIPTION:
                     06B1  2033 ;
                     06B1  2034 ; THIS ROUTINE IS CALLED BY $CRMPSC AFTER IT HAS INITIALIZED A SHARED MEMORY
                     06B1  2035 ; GLOBAL SECTION.  A SEARCH OF THE SPECIFIC SHARED MEMORY'S GSD TABLE IS MADE
                     06B1  2036 ; TO ASCERTAIN IF A GLOBAL SECTION OF THE SAME NAME WAS CREATED DURING THE
                     06B1  2037 ; TIME THAT $CRMPSC WAS CREATING THE SECTION.
                     06B1  2038 ;
                     06B1  2039 ; TWO LOCKS MUST BE ACQUIRED BEFORE THE SHARED MEMORY GSD TABLE MAY BE SEARCHED
                     06B1  2040 ; TO VERIFY A SECTION IS UNIQUE.  THE FIRST IS THE SHARED MEMORY GSD MUTEX
                     06B1  2041 ; WHICH INTERLOCKS PROCESSES ON ONE PROCESSOR.  THE SECOND IS THE SHARED MEMORY
                     06B1  2042 ; GSD TABLE LOCK CONTAINED IN THE SHARED MEMORY COMMON DATA PAGE, WHICH
                     06B1  2043 ; INTERLOCKS BETWEEN PROCESSORS.
                     06B1  2044 ;
                     06B1  2045 ; CALLING SEQUENCE:
                     06B1  2046 ;
                     06B1  2047 ;       BSBW    MMG$UNIQUEGSD
                     06B1  2048 ;
                     06B1  2049 ; INPUT PARAMETERS:
                     06B1  2050 ;
                     06B1  2051 ;       R4 - ADDRESS OF SHARED MEMORY CONTROL BLOCK
                     06B1  2052 ;       R11 - ADDRESS OF GLOBAL SECTION DESCRIPTOR TO BE VERIFIED AS UNIQUE
                     06B1  2053 ;
                     06B1  2054 ; IMPLICIT INPUTS:
                     06B1  2055 ;
                     06B1  2056 ;       NONE
                     06B1  2057 ;
                     06B1  2058 ; OUTPUT PARAMETERS:
                     06B1  2059 ;
                     06B1  2060 ;       R5 - ADDRESS OF SHARED MEMORY COMMON DATA PAGE
                     06B1  2061 ;       R6 - 0 IF THE GSD IS UNIQUE
                     06B1  2062 ;            OTHERWISE, ADDRESS OF DUPLICATE GSD
                     06B1  2063 ;
                     06B1  2064 ; IMPLICIT OUTPUTS:                                  ,
                     06B1  2065 ;
                     06B1  2066 ;       NONE
                     06B1  2067 ;
                     06B1  2068 ; COMPLETION CODES:
                     06B1  2069 ;
                     06B1  2070 ;       NONE
                     06B1  2071 ;
                     06B1  2072 ; SIDE EFFECTS:
                     06B1  2073 ;
                     06B1  2074 ;       NONE
                     06B1  2075 ;
                     06B1  2076 ;--
                     06B1  2077
                     06B1  2078 MMG$UNIQUEGSD::
                     06B1  2079          .ENABL  LSB
           041F 8F   BB 06B1  2080          PUSHR   #^M<R0,R1,R2,R3,R4,R10>          ;SAVE REGISTERS
           50   02   9A 06B5  2081          MOVZBL  #SHD$V_GSDLCK,R0                 ;BIT NUMBER OF LOCK REQUESTED
             0059   30 06B8  2082          BSBW    MMG$SHMTXLK                      ;GET SHM MUTEX AND BIT LOCK
          52 50     E9 06BB  2083          BLBC    R0,ERROR_EXIT                    ;REPORT UNABLE TO GET BIT LOCK
00A0 C5   15 A4     90 06BE  2084          MOVB    SHB$B_PORT(R4),SHD$B_GSDLOCK(R5) ;SET OWNER OF GSD TBL LOCK
             01     DD 06C4  2085          PUSHL   #1                               ;INDICATE TO MMG$VALIDATE AND
```

```
                5A   5E    D0  06C6  2086              MOVL    SP,R10                                  ;MMG$GETNXTGSD NOT TO USE ALL
                                 06C9  2087                                                          ;SHARED MEMORIES IN SEARCH
                                 06C9  2088                                                          ;JUST THE ONE PASSED IN R4,R5
          56    55    04 A5    C1  06C9  2089              ADDL3   SHD$L_GSDPTR(R5),R5,R6           ;GET ADR OF FIRST GSD IN SH MEM
                00000098'EF    16  06CE  2090              JSB     MMG$VALIDATEGSD                 ;FIND FIRST VALID GSD
                          08   11  06D4  2091              BRB     30$                             ;BR TO CHECK IF GSD FOUND
                          30   BA  06D6  2092  20$:        POPR    #^M<R4,R5>                      ;RESTORE SHB, SHD ADRS
                0000009C'EF    16  06D8  2093              JSB     MMG$GETNXTGSD                   ;FIND NEXT VALID GSD
                          56   D5  06DE  2094  30$:        TSTL    R6                              ;IS THERE A GSD ADR?
                          1E   13  06E0  2095              BEQL    NO_DUP_GSD                      ;BR ON NO MORE VALID GSD'S
                          30   BB  06E2  2096              PUSHR   #^M<R4,R5>                      ;REMEMBER SHB, SHD ADRS
             EE 66        01   E0  06E4  2097              BBS     #GSD$V_LOCKED,GSD$L_GSDFL(R6),20$ ;BR IF GSD LOCKED FOR READING
             EA 66        02   E0  06E8  2098              BBS     #GSD$V_DELPEND,GSD$L_GSDFL(R6),20$ ;BR IF GSD BEING DELETION
             54     22    A6   9B  06EC  2099              MOVZBW  GSD$T_GSDNAM(R6),R4             ;GET GLOBAL SECTION NAME LENGTH
             22 AB        54   91  06F0  2100              CMPB    R4,GSD$T_GSDNAM(R11)            ;DO LENGTHS MATCH?
                          E0   12  06F4  2101              BNEQ    20$                             ;IF NEQ NO, TRY AGAIN
       23 AB    23 A6       54   29  06F6  2102              CMPC3   R4,GSD$T_GSDNAM+1(R6),GSD$T_GSDNAM+1(R11) ;COMPARE NAME STRINGS
                          D8   12  06FC  2103              BNEQ    20$                             ;IF NEQ, DIFFERENT NAMES
                          30   BA  06FE  2104              POPR    #^M<R4,R5>                      ;RESTORE SHB,SHD ADRS
                                 0700  2105  NO_DUP_GSD:
          00 009F C5     02   E7  0700  2106              BBCCI   #SHD$V_GSDLCK,SHD$B_FLAGS(R5),50$ ;RELEASE SHM GSD TBL LOCK
                        0048   30  0706  2107  50$:        BSBW    MMG$SHMTXULK                    ;RELEASE SHM MUTEX
                          8E   D5  0709  2108              TSTL    (SP)+                           ;CLEAN OFF DUMMY SHMEM NAM CNT
                    041F  8F    BA  070B  2109  60$:        POPR    #^M<R0,R1,R2,R3,R4,R10>         ;RESTORE REGISTERS
                          05   070F  2110              RSB                                     ;RETURN TO $CRMPSC
                                 0710  2111
                                 0710  2112  ; ************
                                 0710  2113  ; AT SOME LATER DATE, THIS SHOULD SEND AN ERROR MESSAGE TO THE ERROR LOGGER.
                                 0710  2114  ; ************
                                 0710  2115  ERROR_EXIT:
                          56   D4  0710  2116              CLRL    R6                              ;FAILURE TO ACQUIRE BIT LOCK
                          F7   11  0712  2117              BRB     60$                             ;RETURN ERROR STATUS
                                 0714  2118              .DSABL  LSB
```

```
                                0714  2120          .SBTTL  MMG$SHMTXLK/MMG$SHMTXULK - GET/RELEASE SHARED MEMORY MUTEX
                                0714  2121
                                0714  2122 ;++
                                0714  2123 ; FUNCTIONAL DESCRIPTION:
                                0714  2124 ;
                                0714  2125 ; THE ROUTINE MMG$SHMTXLK IS CALLED TO ACQUIRE EXCLUSIVE USE OF A SHARED
                                0714  2126 ; MEMORY GLOBAL SECTION DATA STRUCTURE.  THIS IS DONE BY ACQUIRING A LOCAL
                                0714  2127 ; MEMORY MUTEX AND THEN A SHARED MEMORY BIT LOCK.  A WAIT IS DONE FOR THE MUTEX
                                0714  2128 ; AND A LOOP IS EXECUTED TO ACQUIRE THE BIT LOCK.  THE STATUS CODE FOR
                                0714  2129 ; ACQUIRING THE LOCK, IS RETURNED.  IF THE BIT LOCK COULD NOT BE ACQUIRED, THEN
                                0714  2130 ; AN ERROR CODE IS RETURNED.
                                0714  2131 ;
                                0714  2132 ; THE ROUTINE MMG$SHMTXULK RELEASES THE SHARED MEMORY GLOBAL SECTION DATA
                                0714  2133 ; STRUCTURE MUTEX.
                                0714  2134 ;
                                0714  2135 ; CALLING SEQUENCE:
                                0714  2136 ;
                                0714  2137 ;       BSBW    MMG$SHMTXLK
                                0714  2138 ;       BSBW    MMG$SHMTXULK
                                0714  2139 ;
                                0714  2140 ; INPUT PARAMETERS:
                                0714  2141 ;
                                0714  2142 ;       R0 - BIT NUMBER OF LOCK BEING REQUESTED, FOR MMG$SHMTXLK ONLY
                                0714  2143 ;       R4 - ADDRESS OF SHARED MEMORY CONTROL BLOCK
                                0714  2144 ;
                                0714  2145 ; IMPLICIT INPUTS:
                                0714  2146 ;
                                0714  2147 ;       NONE
                                0714  2148 ;
                                0714  2149 ; OUTPUT PARAMETERS:
                                0714  2150 ;
                                0714  2151 ;       R0 - STATUS CODE, FOR MMG$SHMTXLK ONLY
                                0714  2152 ;       R5 - ADR OF SHARED MEMORY COMMON DATA PAGE, FOR MMG$SHMTXLK ONLY.
                                0714  2153 ;
                                0714  2154 ; IMPLICIT OUTPUTS:
                                0714  2155 ;
                                0714  2156 ;       THE SHARED MEMORY MUTEX AND BIT LOCK MAY BE ACQUIRED BY MMG$SHMTXLK.
                                0714  2157 ;       THE SHARED MEMORY MUTEX MAY BE RELEASED BY MMG$SHMTXULK.
                                0714  2158 ;
                                0714  2159 ; COMPLETION CODES:
                                0714  2160 ;
                                0714  2161 ;       SS$_NORMAL - SUCCESSFULLY ACQUIRED LOCKS, FOR MMG$SHMTXLK ONLY.
                                0714  2162 ;       SS$_INTERLOCK  - UNABLE TO ACQUIRE LOCK, FOR MMG$SHMTXLK ONLY.
                                0714  2163 ;       NONE FOR MMG$SHMTXULK.
                                0714  2164 ;
                                0714  2165 ; SIDE EFFECTS:
                                0714  2166 ;
                                0714  2167 ;       NONE
                                0714  2168 ;
                                0714  2169 ;--
                                0714  2170
                                0714  2171 MMG$SHMTXLK::
                     51   DD    0714  2172          PUSHL   R1                              ;SAVE REGISTER
                     11   BB    0716  2173          PUSHR   #^M<R0,R4>                      ;REMEMBER SHB AND BIT LOCK #
50   00000000'GF        DE    0718  2174          MOVAL   G^EXE$GL_SHMGSMTX,R0            ;ADR OF SH MEM GSD MUTEX
54   00000000'GF        D0    071F  2175          MOVL    G^SCH$GL_CURPCB,R4              ;ADR OF CURRENT PCB
     00000000'GF        16    0726  2176          JSB     G^SCH$LOCKW                     ;GET UNIQUE ACCESS TO MUTEX
```

```
                   11   BA  072C              POPR     #^M<R0,R4>                  ;RESTORE SHB AND BIT LOCK #
51       00000000'GF    D0  072E              MOVL     G^EXE$GL_LOCKRTRY,R1        ;GET LOOP COUNT FOR BIT LOCK
           55     04 A4 D0  0735              MOVL     SHB$L_DATAPAGE(R4),R5       ;GET ADR OF COMMON DATA PAGE
   07 009F C5     50    E6  0739   10$:       BBSSI    R0,SHD$B_FLAGS(R5),20$      ;TRY TO ACQUIRE BIT LOCK
           50     01    9A  073F              MOVZBL   #SS$_NORMAL,R0              ;REPORT LOCK SUCCESSFULLY ACQUIR
                  51  8ED0  0742              POPL     R1                          ;RESTORE REGISTER
                        05  0745              RSB                                  ;RETURN SUCCESS CODE
              F0  51    F5  0746   20$:       SOBGTR   R1,10$                      ;TRY AGAIN TO ACQUIRE BIT LOCK
                  51  8ED0  0749              POPL     R1                          ;RESTORE REGISTER
                            074C                                                   ;R0 CONTAINS 0 TO REPORT FAILURE
     50    038C 8F      3C  074C              MOVZWL   #SS$_INTERLOCK,R0           ;REPORT ERROR STATUS
                            0751                                                   ;FALL THRU TO RELEASE SHM MUTEX
                            0751   MMG$SHMTXULK::
                   13   BB  0751              PUSHR    #^M<R0,R1,R4>               ;SAVE REGISTERS
50       00000000'GF    DE  0753              MOVAL    G^EXE$GL_SHMGSMTX,R0        ;ADR OF SH MEM GSD MUTEX
54       00000000'GF    D0  075A              MOVL     G^SCH$GL_CURPCB,R4          ;ADR OF CURRENT PCB
         00000000'GF    16  0761              JSB      G^SCH$UNLOCK                ;GET UNIQUE ACCESS TO MUTEX
                   13   BA  0767              POPR     #^M<R0,R1,R4>               ;RESTORE REGISTERS
                        05  0769              RSB                                  ;RETURN TO CALLER
```

```
                     076A  2197          .SBTTL  MMG$DELSHMGS - DELETE SHARED MEMORY GLOBAL SECTION
                     076A  2198
                     076A  2199 ;++
                     076A  2200 ; FUNCTIONAL DESCRIPTION:
                     076A  2201 ;
                     076A  2202 ; THIS ROUTINE IS CALLED DURING A SCAN OF THE SECTION TABLE FOR SECTIONS READY
                     076A  2203 ; TO BE DELETED.  IT CHECKS THE PTE REFERENCE COUNTS FOR THE PARTICULAR GLOBAL
                     076A  2204 ; SECTION, DETERMINING WHETHER OR NOT THE SECTION IS READY TO BE DELETED.  IF
                     076A  2205 ; IT CAN BE DELETED, THEN THE PAGES ALLOCATED ARE RELEASED, THE GSD IS RELEASED,
                     076A  2206 ; AND LASTLY, THE SECTION TABLE ENTRY IS RELEASED.
                     076A  2207 ;
                     076A  2208 ; CALLING SEQUENCE:
                     076A  2209 ;
                     076A  2210 ;       BSBW    MMG$DELSHMGS
                     076A  2211 ;
                     076A  2212 ; INPUT PARAMETERS:
                     076A  2213 ;
                     076A  2214 ;       R1 - SECTION TABLE OFFSET
                     076A  2215 ;       R3 - ADDRESS OF SECTION TABLE ENTRY TO BE DELETED
                     076A  2216 ;       R5 - SYSTEM PROCESS HEADER ADDRESS
                     076A  2217 ;
                     076A  2218 ; IMPLICIT INPUTS:
                     076A  2219 ;
                     076A  2220 ;       THE SHARED MEMORY GLOBAL SECTION PAGE BITMAP MUST HAVE BEEN INITIALIZED.
                     076A  2221 ;
                     076A  2222 ; OUTPUT PARAMETERS:
                     076A  2223 ;
                     076A  2224 ;       NONE
                     076A  2225 ;
                     076A  2226 ; IMPLICIT OUTPUTS:
                     076A  2227 ;
                     076A  2228 ;       THE GLOBAL SECTION PAGES, GLOBAL SECTION DESCRIPTOR, AND SECTION TABLE
                     076A  2229 ;       ENTRY ARE RELEASED, IF ALL REFERENCE COUNTS ARE ZERO.
                     076A  2230 ;
                     076A  2231 ; COMPLETION CODES:
                     076A  2232 ;
                     076A  2233 ;       NONE
                     076A  2234 ;
                     076A  2235 ; SIDE EFFECTS:
                     076A  2236 ;
                     076A  2237 ;       NONE
                     076A  2238 ;
                     076A  2239 ;--
                     076A  2240          .ENABL  LSB
                     076A  2241 ;
                     076A  2242 ; SET INDICATOR TO CHECK LATER TO DELETE THIS SECTION.  THERE IS STILL A PROCESS
                     076A  2243 ; MAPPED TO IT AT PRESENT.
                     076A  2244 ;
                     076A  2245 RETRY_DEL:
00 36 A5   01   E6   076A  2246          BBSSI   #PHD$V_DALCSTX,PHD$W_FLAGS(R5),NO_DEL ;SECTION STILL TO BE DEALLOC
         009A   31   076F  2247 NO_DEL: BRW     100$                                  ;BRANCH TO EXIT
                     0772  2248
                     0772  2249 MMG$DELSHMGS::
     007E 8F   BB    0772  2250          PUSHR   #^M<R1,R2,R3,R4,R5,R6>            ;SAVE REGISTERS
     56   63   D0    0776  2251          MOVL    SEC$L_GSD(R3),R6                  ;GET ADR OF GSD
          2F   13    0779  2252          BEQL    18$                               ;BR IF PARTIALLY CREATED GS
                     077B  2253 ;
```

```
                                077B  2254 ; NOW CHECK IF THE GLOBAL SECTION IS MARKED FOR DELETION.  IF SO, CHECK THE
                                077B  2255 ; PROCESSOR PTE REFERENCE COUNTS TO SEE IF THE SECTION CAN BE DELETED NOW.
                                077B  2256 ;
          F0 66    02    E1   077B  2257          BBC     #GSD$V_DELPEND,GSD$L_GSDFL(R6),NO_DEL ;BR IF NOT MARK FOR DEL
          52    51 A6    9A   077F  2258          MOVZBL  GSD$B_PROCCNT(R6),R2          ;GET # OF PROC REF CNTS TO CHECK
          53    74 A6    9E   0783  2259          MOVAB   GSD$L_PTECNT1(R6),R3          ;GET ADR OF FIRT PROC REF CNT
                   83    D5   0787  2260 10$:     TSTL    (R3)+                         ;ARE THERE OUTSTANDING REFS?
                   DF    12   0789  2261          BNEQ    RETRY_DEL                     ;BR IF REF EXISTS, CAN'T DEL YET
                F9 52    F5   078B  2262          SOBGTR  R2,10$                        ;LOOP TO CHECK ALL REF CNTS
                              078E  2263 ;
                              078E  2264 ; THE GSD IS MARKED FOR DELETE AND ALL THE PROCESSOR REFERENCE COUNTS HAVE
                              078E  2265 ; DROPPED TO ZERO.  THEREFORE, THE SHARED MEMORY GLOBAL PAGES MAY BE RELEASED
                              078E  2266 ; AND THE GSD MARKED AS UNUSED.  IF THE SECTION IS WRITABLE, NOT COPY-ON-
                              078E  2267 ; REFERENCE, AND MAPPED TO A FILE THEN THE SECTION MUST BE WRITTEN BACK TO
                              078E  2268 ; THE FILE BEFORE IT CAN BE DELETED.
                              078E  2269 ;
          53    16 A6    32   078E  2270          CVTWL   GSD$W_GSTX(R6),R3             ;IS IT MAPPED TO A FILE?
                   2A    13   0792  2271          BEQL    20$                           ;NO, BR AS NO OUTPUT NEEDED
       08 20 A6    03    E1   0794  2272          BBC     #SEC$V_WRT,GSD$W_FLAGS(R6),15$ ;BR IF NOT WRITABLE
       03 20 A6    01    E0   0799  2273          BBS     #SEC$V_CRF,GSD$W_FLAGS(R6),15$ ;DON'T WRITE OUT CRF SEC EITHER
             FD07    30   079E  2274          BSBW    MMG$WRITE_GSD                 ;WRITE SECTION INTO DISK FILE
    52    55    20 A5    C1   07A1  2275 15$:     ADDL3   PHD$L_PSTBASOFF(R5),R5,R2     ;GET BASE OF SECTION TABLE
          53    6243    DE   07A6  2276          MOVAL   (R2)[R3],R3                   ;GET ADR OF SECTION TABLE ENTRY
          53    0C A3    D0   07AA  2277 18$:     MOVL    SEC$L_WINDOW(R3),R3           ;GET WCB ADDRESS FOR SECTION
                0E A3    B7   07AE  2278          DECW    WCB$W_REFCNT(R3)              ;LAST REF ON SHARED WINDOW?
                   0B    14   07B1  2279          BGTR    20$                           ;BRANCH IF NOT LAST REF
                16 A3    B4   07B3  2280          CLRW    WCB$W_NMAP(R3)                ;NO RETRIEVAL POINTERS
                              07B6  2281          ASSUME  WCB$W_P1_COUNT&3 EQ 0         ;STARTS AT LONG WORD OFFSET
00000000'GF     30 A3    0E   07B6  2282          INSQUE  WCB$W_PT_COUNT(R3),G^EXE$GL_WCBDELFL ;QUE WINDOW ON DELETE LIST
                56    D5   07BE  2283 20$:     TSTL    R6                            ;IS THIS A PARTIALLY CREATED GS?
                   3D    13   07C0  2284          BEQL    70$                           ;BR ON YES, NO GSD TO DELETE
          00000113'EF    16   07C2  2285          JSB     MMG$FINDSHD                   ;FIND THE SHB AND SHD FOR GSD
                50    01    9A   07C8  2286          MOVZBL  #SHD$V_BITMAPLCK,R0           ;NUMBER OF BIT TO LOCK
                   FF46    30   07CB  2287          BSBW    MMG$SHMTXLK                   ;ACQUIRE MUTEX AND LOCK BIT
                40 50    E9   07CE  2288          BLBC    R0,300$                       ;BR IF CAN'T LOCK BIT
 009E C5        15 A4    90   07D1  2289          MOVB    SHB$B_PORT(R4),SHD$B_BITMAPLCK(R5) ;IDENTIFY HOLDER OF LOCK
                F826    30   07D7  2290          BSBW    MMG$SET_BITMAP                ;FREE THE PAGES OF THE SECTION
 00 009F C5     01    E7   07DA  2291          BBCCI   #SHD$V_BITMAPLCK,SHD$B_FLAGS(R5),25$ ;RELEASE BITMAP LOCK
                   FF6E    30   07E0  2292 25$:     BSBW    MMG$SHMTXULK                  ;RELEASE MUTEX
             00 66    01    E6   07E3  2293          BBSSI   #GSD$V_LOCKED,GSD$L_GSDFL(R6),30$ ;LOCK THE GSD TO RELEASE IT
             00 66    02    E7   07E7  2294 30$:     BBCCI   #GSD$V_DELPEND,GSD$L_GSDFL(R6),40$ ;INDICATE NO MORE DELETE PEND
             00 66    00    E7   07EB  2295 40$:     BBCCI   #GSD$V_VALID,GSD$L_GSDFL(R6),50$ ;RELEASE THE GSD
             00 66    01    E7   07EF  2296 50$:     BBCCI   #GSD$V_LOCKED,GSD$L_GSDFL(R6),60$ ;UNLOCK THE GSD FOR RE-USE
             56    15 A4    9A   07F3  2297 60$:     MOVZBL  SHB$B_PORT(R4),R6             ;GET PORT NUMBER
                0C A4    D7   07F7  2298          DECL    SHB$L_REFCNT(R4)              ;ONE LESS PORT REFCNT
          3C A546    01    58   07FA  2299          ADAWI   #1,SHD$W_GSDQUOTA(R5)[R6]     ;RETURN QUOTA FOR GSD
    55     00000000'FF    DE   07FF  2300 70$:     MOVAL   @L^MMG$GL_SYSPHD,R5           ;GET SYSTEM HEADER ADDRESS
                51    6E    D0   0806  2301          MOVL    (SP),R1                       ;GET SECTION TABLE OFFSET
                   F7F4'    30   0809  2302          BSBW    MMG$DALCSTX                   ;GO RELEASE THE SEC TBL ENTRY
                              080C  2303 ;
                              080C  2304 ; RETURN SUCCESSFULLY HERE.
                              080C  2305 ;
                   007E 8F    BA   080C  2306 100$:    POPR    #^M<R1,R2,R3,R4,R5,R6>        ;RESTORE REGISTERS
                              05   0810  2307          RSB                           ;RETURN TO CALLER
                              0811  2308
                              0811  2309 ;
                              0811  2310 ; CAN'T LOCK BITMAP.  MAKE THE GSD LOOK UNOWNED AND CONTINUE CLEANING UP
```

```
                     0811  2311 ; THE SECTION TABLE ENTRY.  EVENTUALLY, MMG$FREEGSD WILL FIND AND FREE
                     0811  2312 ; UP THE GSD AND BITMAP.
                     0811  2313 ;
      16 A6   B4     0811  2314 300$:    CLRW    GSD$W_GSTX(R6)                  ;NULL SECTION TABLE INDEX
52 A6    00   92     0814  2315           MCOMB   #0,GSD$B_CREATPORT(R6)          ;MAKE CREATOR INVALID
         D9   11     0818  2316           BRB     60$                             ;REJOIN MAIN FLOW
                     081A  2317           .DSABL  LSB
```

```
                        081A         .SBTTL  MMG$FINDSHD - FIND THE SHARED MEMORY CONTAINING THIS GSD
                        081A
                        081A  ;++
                        081A  ; FUNCTIONAL DESCRIPTION:
                        081A  ;
                        081A  ; THIS ROUTINE FINDS THE SHARED MEMORY CONTROL BLOCK ADDRESS AND COMMON
                        081A  ; DATA PAGE ADDRESS FOR A SHARED MEMORY THAT CONTAINS A PARTICULAR GLOBAL
                        081A  ; SECTION.
                        081A  ;
                        081A  ; CALLING SEQUENCE:
                        081A  ;
                        081A  ;      BSBW    MMG$FINDSHD
                        081A  ;
                        081A  ; INPUT PARAMETERS:
                        081A  ;
                        081A  ;      R6 - ADDRESS OF GLOBAL SECTION DESCRIPTOR
                        081A  ;
                        081A  ; IMPLICIT INPUTS:
                        081A  ;
                        081A  ;      THE SHARED MEMORY DATA STRUCTURES ARE AVAILABLE (NOT DISCONNECTED).
                        081A  ;
                        081A  ; OUTPUT PARAMETERS:
                        081A  ;
                        081A  ;      R4 - ADDRESS OF SHARED MEMORY CONTROL BLOCK
                        081A  ;      R5 - ADDRESS OF SHARED MEMORY COMMON DATA PAGE
                        081A  ;
                        081A  ; IMPLICIT OUTPUTS:
                        081A  ;
                        081A  ;      NONE
                        081A  ;
                        081A  ; COMPLETION CODES:
                        081A  ;
                        081A  ;      NONE
                        081A  ;
                        081A  ; SIDE EFFECTS:
                        081A  ;
                        081A  ;      NONE
                        081A  ;
                        081A  ;--
                        081A  ; ****************************************************************************
                        081A  ;
                        081A  ; *********************** THE FOLLOWING CODE MUST BE RESIDENT *****************
                        081A  ;
                    00000113         .PSECT  $MMGCOD
                        0113  ;
                        0113  ; ****************************************************************************
                        0113  ;
                        0113
                        0113  MMG$FINDSHD::
              07   BB   0113         PUSHR   #^M<R0,R1,R2>                       ;SAVE REGISTERS
 54    00000000'GF   D0   0115         MOVL    G^EXE$GL_SHBLIST,R4                 ;GET ADR OF FIRST SHB
              03   11   011C         BRB     20$                                 ;JOIN COMMON CODE
         54   64   D0   011E  10$:   MOVL    SHB$L_LINK(R4),R4                   ;GET ADR OF NEXT SHB
              29   13   0121  20$:   BEQL    NO_SHD_FOUND                        ;IF NO NEXT SHB, FATAL ERROR
   F6 0B A4   00   E1   0123         BBC     #SHB$V_CONNECT,SHB$B_FLAGS(R4),10$  ;IF DISCONNECTED, TRY NXT SHB
      55   04 A4   D0   0128         MOVL    SHB$L_DATAPAGE(R4),R5               ;GET ADR OF COMMON DATA PAGE
 51   55   04 A5   C1   012C         ADDL3   SHD$L_GSDPTR(R5),R5,R1              ;FIND START OF GSD TABLE
```

```
                51    56    D1   0131  2376          CMPL    R6,R1                           ;IS GSD WITHIN THIS TABLE?
                      E8    1F   0134  2377          BLSSU   10$                             ;NO, GO FIND NEXT SHB
          50    08  A1    3C   0136  2378          MOVZWL  GSD$W_SIZE(R1),R0               ;GET SIZE OF ONE GSD
          52    18  A5    3C   013A  2379          MOVZWL  SHD$W_GSDMAX(R5),R2             ;GET NUMBER OF GSD'S IN TABLE
                52    50    C4   013E  2380          MULL2   R0,R2                           ;GET # OF BYTES IN TABLE
                52    51    C0   0141  2381          ADDL2   R1,R2                           ;GET ADR PAST END OF GSD TABLE
                52    56    D1   0144  2382          CMPL    R6,R2                           ;IS GSD IN THIS TABLE?
                      D5    1E   0147  2383          BGEQU   10$                             ;NO, GO FIND NEXT SHB
                      07    BA   0149  2384          POPR    #^M<R0,R1,R2>                   ;RESTORE REGISTERS
                      05    05   014B  2385          RSB                                     ;RETURN TO CALLER
                            014C  2386
                            014C  2387 ;
                            014C  2388 ; THE GSD IS NOT IN A CONNECTED SHARED MEMORY.  THIS IS AN INCONSISTENCY IN
                            014C  2389 ; IN THE DATA BASE.  FOR NOW, BUGCHECK.
                            014C  2390 ;
                            014C  2391 NO_SHD_FOUND:
                            014C  2392          BUG_CHECK       NOSHMGSD,FATAL          ;FATAL ERROR
                            0150  2393          .END
```

```
ALL_DONE                        00000063 R      02
ALL_REST_SET                    000000CC R      02
BUG$_KRPEMPTY                   ********   X    02
BUG$_NEGSHBREF                  ********   X    03
BUG$_NOSHMGSD                   ********   X    03
BUG$_REFCNTNEG                  ********   X    03
CHECK_XLATION                   000003F0 R      02
CLR_BITMAP                      00000143 R      02
CTL$GL_KRPFL                    ********   X    02
CTL$GL_PCB                      ********   X    02
DYN$C_IRP                     = 0000000A
END_OF_BITMAP                   00000125 R      02
ERROR_EXIT                      00000710 R      02
EVT$_PFCOM                      ********   X    02
EXE$ALONONPAGED                 ********   X    02
EXE$BLDPKTGSR                   ********   X    02
EXE$BLDPKTGSW                   ********   X    02
EXE$DEANONPAGED                 ********   X    02
EXE$GL_GSDGRPFL                 ********   X    02
EXE$GL_LOCKRTRY                 ********   X    02
EXE$GL_SHBLIST                  ********   X    03
EXE$GL_SHMGSMTX                 ********   X    02
EXE$GL_WCBDELFL                 ********   X    02
FILE_DEV                        00000343 R      02
FILE_DEV_DESC                   0000033B R      02
FILE_DEV_SIZE                 = 0000000C
FIND_PIECE                      0000001B R      02
FIND_PIECE_END                  000000AA R      02
FOUND_1_PIECE                   0000012F R      02
FOUND_IT                        00000057 R      03
GET_NXT_SHM                     0000006A R      03
GOT_PIECE                       000000D1 R      02
GSD$B_CREATPORT               = 00000052
GSD$B_DELETPORT               = 00000053
GSD$B_LOCK                    = 00000050
GSD$B_PROCCNT                 = 00000051
GSD$C_PFNBASMAX               = 00000004
GSD$L_BASCNT1                 = 00000058
GSD$L_BASPFN1                 = 00000054
GSD$L_[illegible]DFL          = 00000000
GSD$L_[illegible]ECNT1        = 00000074
GSD$T_GSDNAM                  = 00000022
GSD$V_DELPEND                 = 00000002
GSD$V_LOCKED                  = 00000001
GSD$V_VALID                   = 00000000
GSD$W_FLAGS                   = 00000020
GSD$W_GSTX                    = 00000016
GSD$W_SIZE                    = 00000008
INSF_MEM                        00000157 R      02
INVALID_LOGNAM                  00000493 R      02
IO_FAIL                         0000061F R      02
IPL$_SYNCH                    = 00000008
IRP$B_PRI                     = 00000023
IRP$B_TYPE                    = 0000000A
IRP$C_LENGTH                  = 000000C4
IRP$L_ASTPRM                  = 00000014
IRP$L_IOSB                    = 00000024
IRP$L_IOST1                   = 00000038
IRP$W_SIZE                    = 00000008
LNM$C_MAXDEPTH                = 0000000A
LNM$C_NAMLENGTH               = 000000FF
LNM$SEARCH_ONE                  ********   X    02
LNMX$B_FLAGS                  = 00000000
LNMX$T_XLATION                = 00000004
LNMX$V_TERMINAL               = 00000001
LOCK_ERR                        00000099 R      02
LWA_COLON                     = 0000000C
LWA_END                       = 00000111
LWA_INPUT                     = 00000012
LWA_INPUT_DESC                = 00000004
LWA_PREFIX                    = 00000000
LWA_XLATION                   = 0000000D
MAXIO                         = 00000020
MMG$ALOSHMGSD                   00000174 RG     02
MMG$ALOSHMPAG                   00000068 RG     02
MMG$CEFTRNLOG                   0000034F RG     02
MMG$CLR_BITMAP                  00000009 RG     02
MMG$DALCSTX                     ********   X    02
MMG$DECSHMREF                   00000076 RG     03
MMG$DELSHMGS                    00000772 RG     02
MMG$FIND1STGSD                  0000026B RG     02
MMG$FINDGSDPFN                  00000000 RG     03
MMG$FINDGSNOTRN                 00000671 RG     02
MMG$FINDSHB                     00000293 RG     02
MMG$FINDSHD                     00000113 RG     03
MMG$FREEGSD                     000001FF RG     02
MMG$GETGSNAM                    000002C8 RG     02
MMG$GETNXTGSD                   0000009C RG     03
MMG$GL_SYSPHD                   ********   X    02
MMG$GSDSCN                      ********   X    02
MMG$GSDTRNLOG                   00000361 RG     02
MMG$INCSHMREF                   00000079 RG     03
MMG$MBXTRNLOG                   00000358 RG     02
MMG$READ_GSD                    000004B1 RG     02
MMG$SET_BITMAP                  00000000 RG     02
MMG$SHMTXLK                     00000714 RG     02
MMG$SHMTXULK                    00000751 RG     02
MMG$UNIQUEGSD                   000005B1 RG     02
MMG$VALIDATEGSD                 0000093[illegible] RG     03
MMG$WRITE_GSD                   000004A8 RG     02
NEWIPL                          0000066D R      02
NEXT_PIECE                      00000060 R      02
NOT_FOUND                       0000006F R      03
NOT_MAPPED                      00000519 R      02
NO_BIT_SET                      0000011C R      02
NO_DEL                          0000076F R      02
NO_DUP_GSD                      00000700 R      02
NO_FREE_GSD                     000001E8 R      02
NO_IRP                          0000061F R      02
NO_QUOTA                        000001F3 R      02
NO_SHD_FOUND                    0000014C R      03
NXT_PIECE                       0000009C R      02
PCB$B_PRIB                    = 0000002F
PCB$L_PHD                     = 0000006C
```

```
PHD$L_PSTBASOFF                 = 00000020
PHD$V_DALCSTX                   = 00000001
PHD$W_FLAGS                     = 00000036
PR$_IPL                         = 00000012
PRI$_IOCOM                      = 00000001
PSL$C_USER                      = 00000003
PSL$S_CURMOD                    = 00000002
PSL$S_PRVMOD                    = 00000002
PSL$V_CURMOD                    = 00000018
PSL$V_PRVMOD                    = 00000016
PTE$C_ERKW                      = 30000000
PTE$M_VALID                     = 80000000
REI_ROUTINE                       000006B0 R      02
REI_RTN1                          0000060E R      02
RETRY_DEL                         0000076A R      02
RETURN                            00000498 R      02
RSB_HERE                          00000062 R      03
SCH$GL_CURPCB                     ******** X    02
SCH$GQ_PFWQ                       ******** X    02
SCH$LOCKW                         ******** X    02
SCH$RSE                           ******** X    02
SCH$UNLOCK                        ******** X    02
SCH$WAITK                         ******** X    02
SEC$B_PFC                       = 0000000B
SEC$L_GSD                       = 00000000
SEC$L_VBN                       = 00000010
SEC$L_WINDOW                    = 0000000C
SEC$V_CRF                       = 00000001
SEC$V_DZRO                      = 00000002
SEC$V_WRT                       = 00000003
SHB$B_FLAGS                     = 0000000B
SHB$B_PORT                      = 00000015
SHB$L_BASGSPFN                  = 00000010
SHB$L_DATAPAGE                  = 00000004
SHB$L_LINK                      = 00000000
SHB$L_REFCNT                    = 0000000C
SHB$V_CONNECT                   = 00000000
SHD$B_BITMAPLCK                 = 0000009E
SHD$B_FLAGS                     = 0000009F
SHD$B_GSDLOCK                   = 000000A0
SHD$L_GSBITMAP                  = 0000000C
SHD$L_GSDPTR                    = 00000004
SHD$L_GSPAGCNT                  = 00000010
SHD$T_NAME                      = 00000020
SHD$V_BITMAPLCK                 = 00000001
SHD$V_GSDLCK                    = 00000002
SHD$W_GSDMAX                    = 00000018
SHD$W_GSDQUOTA                  = 0000003C
SHMIODONE                         00000640 R      02
SS$_EXPORTQUOTA                 = 000003AC
SS$_GSDFULL                     = 000000CC
SS$_INSFMEM                     = 00000124
SS$_INTERLOCK                   = 0000038C
SS$_IVLOGNAM                    = 00000154
SS$_NOLOGNAM                    = 000001BC
SS$_NORMAL                      = 00000001
SS$_NOSUCHSEC                   = 00000978
SS$_SHMGSNOTMAP                 = 0000036C
SS$_SHMNOTCNCT                  = 0000037C
SS$_TOOMANYLNAM                 = 00000374
STOP_TRANSLATION                  00000460 R      02
SYNCHIPL                          0000061B R      02
TRANSLATE_LOOP                    000003B6 R      02
TRANSLATION_DONE                  0000048E R      02
WCB$W_NMAP                      = 00000016
WCB$W_P1_COUNT                  = 00000030
WCB$W_REFCNT                    = 0000000E
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
Y$EXEPAGED	0000081A (2074.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$MMGCOD	00000150 (336.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.04	00:00:01.29
Command processing	113	00:00:00.52	00:00:05.67
Pass 1	476	00:00:18.43	00:01:14.95
Symbol table sort	0	00:00:02.67	00:00:07.35
Pass 2	400	00:00:06.04	00:00:21.42
Symbol table output	1	00:00:00.17	00:00:00.58
Psect synopsis output	0	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1021	00:00:27.90	00:01:51.30

The working set limit was 2100 pages.
113259 bytes (222 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1630 non-local and 118 local symbols.
2393 source lines were read in Pass 1, producing 23 object records in Pass 2.
32 pages of virtual memory were used to define 31 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	19
_255DUA28:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	28

1711 GETS were required to define 28 macros.

There were no errors, warnings or information messages.

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

SHELL
LIS

RSE
LIS

SCBVECTOR
LIS

SDAT
LIS

SHMGSDRTN
LIS

SCSVEC
LIS

SCHED
LIS

SECAUDIT
LIS

SPTSKEL
LIS

RUFSYSVEC
LIS