L 12

```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
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
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
```

FILEID**IMGMAPISD

N 4

```
IIIIII   MM        MM   GGGGGGGG  MM        MM    AAAAAA    PPPPPPPP    IIIIII      SSSSSSSS  DDDDDDDD
IIIIII   MM        MM   GGGGGGGG  MM        MM    AAAAAA    PPPPPPPP    IIIIII      SSSSSSSS  DDDDDDDD
  II     MMMM    MMMM  GG         MMMM    MMMM  AA      AA  PP      PP    II      SS          DD      DD
  II     MMMM    MMMM  GG         MMMM    MMMM  AA      AA  PP      PP    II      SS          DD      DD
  II     MM  MM  MM    GG         MM  MM  MM    AA      AA  PP      PP    II      SS          DD      DD
  II     MM  MM  MM    GG         MM  MM  MM    AA      AA  PP      PP    II      SS          DD      DD
  II     MM        MM  GG         MM        MM  AA      AA  PPPPPPPP      II        SSSSSS    DD      DD
  II     MM        MM  GG         MM        MM  AA      AA  PPPPPPPP      II        SSSSSS    DD      DD
  II     MM        MM  GG  GGGGGG MM        MM  AAAAAAAAAA  PP            II              SS  DD      DD
  II     MM        MM  GG  GGGGGG MM        MM  AAAAAAAAAA  PP            II              SS  DD      DD
  II     MM        MM  GG      GG MM        MM  AA      AA  PP            II              SS  DD      DD   ....
  II     MM        MM  GG      GG MM        MM  AA      AA  PP            II              SS  DD      DD   ....
IIIIII   MM        MM    GGGGGG   MM        MM  AA      AA  PP          IIIIII      SSSSSSSS  DDDDDDDD     ....
IIIIII   MM        MM    GGGGGG   MM        MM  AA      AA  PP          IIIIII      SSSSSSSS  DDDDDDDD     ....

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
```

IMG$
V04-

```
0 %TITLE 'MAP_ISDS - Convert ISDs to Mapping Requests'
0 MODULE IMG$MAP_ISDS (
0                 IDENT = 'V04-001'                  ! File: SRC$:IMGMAPISD.B32
0                 ) =
1 BEGIN
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! Facility:
1 !
1 !       Executive, Support Routines Used by Image Activator System Service
1 !
1 ! Abstract:
1 !
1 !       This module contains the routines that transform each image section
1 !       descriptor into a mapping request for the process address space.
1 !
1 ! Environment:
1 !
1 !       The code in this module executes in executive mode.
1 !
1 ! Author:
1 !
1 !       Lawrence J. Kenah
1 !
1 ! Creation Date:
1 !
1 !       6 May 1983
1 !
1 ! Modified By:
1 !
1 !       V04-001 MSH0074         Michael S. Harvey       6-Sep-1984
1 !               Don't open image file for write for ISDs that are
1 !               global, writeable, and demand-zero.
```

```
1 !
1 !       V03B-017  MSH0045        Michael S. Harvey        10-May-1984
1 !                 Correctly set page protection for indirect message
1 !                 sections.
1 !
1 !       V03B-016  MSH0044        Michael S. Harvey        9-May-1984
1 !                 Correct page protection for protected writable CRF
1 !                 sections.
1 !
1 !       V03B-015  LJK0281        Lawrence J. Kenah        9-May-1984
1 !                 Still more cleanup. Make sure that all unused ICBs are
1 !                 deallocated when an error occurs in the middle of a
1 !                 complicated activation.
1 !
1 !       V03B-014  LJK0269        Lawrence J. Kenah        31-Mar-1984
1 !                 Miscellaneous cleanup.
1 !                 Do not perform fixups or load message or change mode vectors
1 !                 if this is the P0 part of a P1 merge activation.
1 !                 Use flag in own storage to detect that situation.
1 !                 Pages in privileged shareable image should be owned by
1 !                 exec mode.
1 !                 Add privileged vectors in exec mode instead of kernel mode.
1 !                 Store alternate RMS address as part of kernel mode completion.
1 !
1 !       V03B-013  LJK0267        Lawrence J. Kenah        28-Mar-1984
1 !                 Do not use name in ICB as global section name. Instead, use
1 !                 name in CTX storage that was loaded from the KFE.
1 !
1 !       V03B-012  LJK0265        Lawrence J. Kenah        25-Mar-1984
1 !                 Add support for variable size SHL elements.
1 !
1 !       V03B-011  JWT0152        Jim Teague               20-Feb-1984
1 !                 Enlarge ISD_BUFFER to accomodate longer global ISDs.
1 !
1 !       V03B-010  WMC0001        Wayne Cardoza            23-Jan-1984
1 !                 Add loading of sequential image files.
1 !                 Fix bug in based shareable images.
1 !
1 !       V03B-009  LJK0245        Lawrence J. Kenah        23-Aug-1983
1 !                 Make user stack size a cumulative number.
1 !
1 !       V03B-008  LJK0242        Lawrence J. Kenah        2-Aug-1983
1 !                 Add support for writable global sections.
1 !
1 !       V03B-007  LJK0236        Lawrence J. Kenah        26-Jul-1983
1 !                 Add concept of image base address, different from first address
1 !                 that is mapped. Continue fixing bugs as they are found.
1 !
1 !       V03B-006  LJK0233        Lawrence J. Kenah        21-Jul-1983
1 !                 Try again to make the logic for P1 merges into P0 space
1 !                 to work correctly.
1 !
1 !       V03B-005  LJK0231        Lawrence J. Kenah        19-Jul-1983
1 !                 Make corrections for global section mapping
1 !
1 !       V03B-004  LJK0227        Lawrence J. Kenah        12-Jul-1983
1 !                 Continue with cleanup efforts and testing.
```

```
1 !
1 !       V03B-003  LJK0220          Lawrence J. Kenah          29-Jun-1983
1 !                 Handle case where input range is specified. Perform minor
1 !                 cleanup. Fix the many bugs discovered during testing.
1 !
1 !       V03B-002  LJK0216          Lawrence J. Kenah          25-Jun-1983
1 !                 Make new IHD generation constant conform to standard names
1 !
1 !       V03B-001  LJK0200          Lawrence J. Kenah          6-May-1983
1 !                 The original version of this module is a portion of a
1 !                 rewrite of the image activator.
1 !--
1
```

```
%SBTTL 'Declarations'

SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);

PSECT
        CODE    = YF$$SYSIMGACT (WRITE),
        PLIT    = YF$$SYSIMGACT (WRITE, EXECUTE);

LIBRARY 'SYS$LIBRARY:LIB.L32';              ! Define system data structures

REQUIRE 'LIB$:IMGMSGDEF.R32';               ! Get status code definitions
REQUIRE 'LIB$:IMGACTCTX.R32';               ! Define internal structures

! Machine dependent features

BUILTIN
    PROBEW,
    INSQUE,
    REMQUE;

! Internal references

FORWARD ROUTINE
    PROCESS_ISD_LIST,
    ADD_PRIVILEGED_VECTOR,
    ADD_FIXUP_VECTOR,
    NEXT_GBL_SEC_NAME           : NOVALUE,
    LOAD_SEQ_IMAGE;

! Linkage declaration for JSB routines

LINKAGE
    EXE_ALOP1PROC     = JSB (REGISTER = 1; REGISTER = 1,REGISTER = 2) :
                        NOPRESERVE (3)
                        NOTUSED (4,5,6,7,8,9,10,11),
    EXE_DEAP1         = JSB (REGISTER = 0, REGISTER = 1) :
                        NOPRESERVE (2,3)
                        NOTUSED (4,5,6,7,8,9,10,11),
    IMG_IS_IT_MAPPED = JSB (REGISTER = 0; REGISTER = 1) :
                        PRESERVE (2,3,4,5,6,7)
                        NOTUSED (8,9,10,11),
    IMG_PRVSHRIMG     = JSB (REGISTER = 0, REGISTER = 1) :
                        PRESERVE (2,3,4,5)
                        NOTUSED (6,7,8,9,10,11);

! External routines with special linkages

EXTERNAL ROUTINE
    EXE$ALOP1PROC     : EXE_ALOP1PROC,
    EXE$DEAP1         : EXE_DEAP1,
    IMG$IS_IT_MAPPED : IMG_IS_IT_MAPPED,
    IMG$PRVSHRIMG     : IMG_PRVSHRIMG;

! Constants to make life simple

LITERAL
    TRUE = 1,
```

IMG$
V04-

```
1       FALSE = 0,
1       BYTES_PER_PAGE         = 512,
1       END_OF_P1_SPACE        = %X'7FFFFFFF',
1       ISD_M_DISPATCH_FLAGS = (ISD$M_GBL OR ISD$M_CRF OR ISD$M_DZRO OR ISD$M_WRT);
1
1 !  External routine references
1
1 EXTERNAL ROUTINE
1       IMG$OPEN_IMAGE,
1       IMG$GET_HEADER,
1       IMG$ALLOCATE_ICB,
1       IMG$DEALLOCATE_ICB,
1       IMG$GET_NEXT_ISD;
1
1 ! References to external data cells
1
1 EXTERNAL
1       CTL$A_DISPVEC,
1       CTL$GL_FIXUPLNK,
1       CTL$GL_PHD,
1       IAC$GL_IMAGCTX                  : $BBLOCK,
1       IAC$AL_IMGACTBUF,
1       IAC$GL_WORK_LIST                : VECTOR [2],
1       IAC$GL_IMAGE_LIST               : VECTOR [2];
1
1 ! Miscellaneous constants defined elsewhere
1
1 EXTERNAL LITERAL
1       EXE$C_SYSEFN            : UNSIGNED (6),
1       SYS$K_VERSION;
1
1 ! Some miscellaneous address definitions
1
1 ! The first part of the image activator scratch area is divided up into two
1 ! large pieces, each of which is further subdivided into a FAB, a NAM block,
1 ! a 512-byte block into which each succeeding block of the image header will
1 ! be read, and a buffer that will receive the decoded image header. The
1 ! decoded image header is assumed to be smaller than a page. The area that
1 ! follows these buffers is used as OWN storage by the image activator.
1
1 LITERAL
1       INPUT_BUFFER_SIZE = BYTES_PER_PAGE,
1       IHD_BUFFER_SIZE   = BYTES_PER_PAGE;
1
1 BIND
1       INPUT_BUFFER = IAC$AL_IMGACTBUF,
1       PRIMARY_IHD  = INPUT_BUFFER + 512,
1       AUX_BUFFER   = PRIMARY_IHD  + 512,
1       AUX_IHD      = AUX_BUFFER   + 512,
1       PRIMARY_FAB  = AUX_IHD      + 512,
1       PRIMARY_NAM  = PRIMARY_FAB  + FAB$K_BLN,
1       AUX_FAB      = PRIMARY_NAM  + NAM$K_BLN,
1       AUX_NAM      = AUX_FAB      + FAB$K_BLN,
1       RESULT_NAME  = AUX_NAM      + NAM$K_BLN,
1       OWN_STORAGE  = RESULT_NAME  + NAM$C_MAXRSS  : $BBLOCK;
1
1 ! There are eight pages set aside in P1 space (in module SHELL) for
```

```
! the image activator scratch area. The following assumption guarantees
! that the scratch area that is defined here fits into eight pages.

    $ASSUME ( OWN_STORAGE_SIZE + (2 * (INPUT_BUFFER_SIZE +
                                       IHD_BUFFER_SIZE +
                                       FAB$K_BLN +
                                       NAM$K_BLN) ),
            LEQU,
            8 * BYTES_PER_PAGE );
```

```
%SBTTL 'PROCESS_WORK_LIST - Process Each Work List Item'

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine removes each work list item in turn from the work list,
!       opens the file if it is not already opened, and calls the inner routine
!       that processes the ISD list for this image.
!
! CALLING SEQUENCE:
!
!       IMG$DO_WORK_LIST ()
!
! FORMAL PARAMETERS:
!
!       TBS
!
! STATUS CODES:
!
!       TBS
!--

GLOBAL ROUTINE IMG$DO_WORK_LIST =

BEGIN

LOCAL
    ICB         : REF $BBLOCK,
    OLD_ICB     : REF $BBLOCK,
    NAME_DESC   : $BBLOCK [DSC$K_S_BLN],
    IHD_CTX     : $BBLOCK [CTX_K_LENGTH],
    NAME_STRING : VECTOR [NAM$C_MAXRSS],
    SIZE,
    STATUS;

WHILE NOT REMQUE (.IAC$GL_WORK_LIST , ICB) DO
    BEGIN

    STATUS = IMG$IS_IT_MAPPED (ICB [ICB$T_IMAGE_NAME]; OLD_ICB);
    IF .STATUS EQL SS$_NORMAL
    THEN
        IMG$DEALLOCATE_ICB (.ICB)
    ELSE
        BEGIN

        ! The ICB is immediately placed into the done list. If an error occurs
        ! later in activation, the error cleanup routine will remove it,
        ! deassign the channel, and deallocate the ICB.

        IF NOT (.OWN_STORAGE [P1_MERGE_PO])
        THEN
            INSQUE (.ICB , .IAC$GL_IMAGE_LIST [1]);     ! Insert at tail of list

        ! A nonzero channel number implies that the ICB represents the image
        ! whose name was passed to $IMGACT. This image file was already opened
        ! by the caller and its header read and verified.

```

```
        IF .ICB [ICB$W_CHAN] EQL 0
        THEN
            BEGIN

            BIND
                ICB_NAME = ICB [ICB$T_IMAGE_NAME]       : VECTOR [, BYTE];

            ICB [ICB$L_CONTEXT] = IHD_CTX;
            NAME_DESC [DSC$W_LENGTH] = .ICB_NAME [0];
            NAME_DESC [DSC$A_POINTER] = ICB_NAME [1];
            STATUS = IMG$OPEN_IMAGE (
                NAME_DESC,
                $DESCRIPTOR('SYS$SHARE:.EXE'),
                AUX_FAB,
                AUX_NAM,
                NAME_STRING,
                .ICB);
            IF NOT .STATUS THEN RETURN .STATUS;

            ! Use the auxiliary buffers when decoding the image header
            ! from this point.

            IHD_CTX [CTX_L_BUFFER] = AUX_BUFFER;
            IHD_CTX [CTX_L_IHDBUF] = AUX_IHD;

            STATUS = IMG$GET_HEADER (.ICB);
            IF NOT .STATUS THEN RETURN .STATUS;
            END;

        ! Reserve ISD storage if image is loaded from sequential device
        ! The allocation is for the worst case (all local ISDs)

        IF .ICB [ICB$V_LOAD_IMAGE]
        THEN
            BEGIN
            BIND
                IHD_CTX = .ICB [ICB$L_CONTEXT]: $BBLOCK,
                IHD = .IHD_CTX [CTX_L_IHDBUF] : $BBLOCK;
            SIZE = 12 * (.IHD [IHD$B_HDRBLKCNT] * BYTES_PER_PAGE) / ISD$K_LENPRIV;
            IF NOT EXE$ALOP1PROC ( .SIZE; SIZE, OWN_STORAGE [SEQ_LOAD_ISDS])
            THEN RETURN SS$_INSFMEM;
            CH$FILL ( 0, .SIZE, .OWN_STORAGE [SEQ_LOAD_ISDS] );
            END;

        STATUS = PROCESS_ISD_LIST (.ICB);
        IF NOT .STATUS
        THEN
            BEGIN

            ! Check for sequential load device and release ISD storage

            IF .ICB [ICB$V_LOAD_IMAGE]
            THEN
                EXE$DEAP1 (.OWN_STORAGE [SEQ_LOAD_ISDS], .SIZE);

            RETURN .STATUS;
            END;
```

```
;       4
;       4          ! See if image should be loaded now
;       4
;       4          IF .ICB [ICB$V_LOAD_IMAGE]
;       4          THEN
;       5              BEGIN
;       5              STATUS = LOAD_SEQ_IMAGE ( .ICB );
;       5              EXE$DEAP1 (.OWN_STORAGE [SEQ_LOAD_ISDS], .SIZE);
;       5              IF NOT .STATUS THEN RETURN .STATUS;
;       4              END;
;       4
;       3          END;                          ! End of "it is not yet mapped" block
;       3
;       2      END;                              ! End of WHILE loop
;       2
;       2  RETURN SS$_NORMAL;
;       2
;       1  END;                                  ! End of routine PROCESS_WORK_LIST

                                                          .TITLE  IMG$MAP_ISDS MAP_ISDS - Convert ISDs to Mapping
;                                                                                 Requests
                                                          .IDENT  \V04-001\

                                                          .PSECT  YF$$SYSIMGACT,2

 45 58 45 2E 3A 45 52 41 48 53 24 53 59 53  00000 P.AAB:  .ASCII  \SYS$SHARE:.EXE\                                    ;
                                            0000E         .BLKB   2
                                  0000000E  00010 P.AAA:  .LONG   14                                                 ;
                                  00000000' 00014         .ADDRESS P.AAB                                             ;

                                                          .EXTRN  EXE$ALOP1PROC, EXE$DEAP1
                                                          .EXTRN  IMG$IS_IT_MAPPED
                                                          .EXTRN  IMG$PRVSHRIMG, IMG$OPEN_IMAGE
                                                          .EXTRN  IMG$GET_HEADER, IMG$ALLOCATE_ICB
                                                          .EXTRN  IMG$DEALLOCATE_ICB
                                                          .EXTRN  IMG$GET_NEXT_ISD
                                                          .EXTRN  CTL$A_DISPVEC, CTL$GL_FIXUPLNK
                                                          .EXTRN  CTL$GL_PHD, IAC$GL_IMAGCTX
                                                          .EXTRN  IAC$AL_IMGACTBUF
                                                          .EXTRN  IAC$GL_WORK_LIST
                                                          .EXTRN  IAC$GL_IMAGE_LIST
                                                          .EXTRN  EXE$C_SYSEFN, SYS$K_VERSION

                                      0FFC  00000         .ENTRY  IMG$DO_WORK_LIST, Save R2,R3,R4,R5,R6,R7,-     ; 0504
                                                                  R8,R9,R10,R11                                      ;
                  5B 00000000G   00   9E  00002         MOVAB   EXE$DEAP1, R11                                     ;
                  5A 00000000G   00   9E  00009         MOVAB   OWN_STORAGE+40, R10                                ;
                  5E     FBB8    CE   9E  00010         MOVAB   -1096(SP), SP                                      ;
                  50 00000000G   00   9E  00015 1$:     MOVAB   IAC$GL_WORK_LIST, R0                               ; 0517
                  56       00    B0   0F  0001C         REMQUE  @0(R0), ICB                                        ;
                                 03   1C  00020         BVC     2$                                                 ;
                               00FA   31  00022         BRW     13$                                                ;
                  50       14    A6   9E  00025 2$:     MOVAB   20(ICB), R0                                        ; 0520
                     00000000G   00   16  00029         JSB     IMG$IS_IT_MAPPED                                   ;
                  57             50   D0  0002F         MOVL    R0, STATUS                                         ;
                  59             51   D0  00032         MOVL    R1, R9                                             ;
```

```
                              01         57 D1 00035        CMPL    STATUS, #1                          ; 0521
                                         0B 12 00038        BNEQ    3$                                  ;
                                         56 DD 0003A        PUSHL   ICB                                 ; 0523
                     00000000G  00       01 FB 0003C        CALLS   #1, IMG$DEALLOCATE_ICB              ;
                                         D0 11 00043        BRB     1$                                  ;
              0B        D8      AA       05 E0 00045 3$:    BBS     #5, OWN_STORAGE, 4$                 ; 0531
                                50 00000000G  00 9E 0004A   MOVAB   IAC$GL_IMAGE_LIST+4, R0             ; 0533
                        00      B0       66 0E 00051        INSQUE  (ICB), @0(R0)                       ;
                                    0E   A6 B5 00055 4$:    TSTW    14(ICB)                             ; 0539
                                         4B 12 00058        BNEQ    5$                                  ;
                        58      A6   B4  AD 9E 0005A        MOVAB   IHD_CTX, 88(ICB)                    ; 0546
                        F8      AD   14  A6 9B 0005F        MOVZBW  20(ICB), NAME_DESC                  ; 0547
                        FC      AD   15  A6 9E 00064        MOVAB   21(ICB), NAME_DESC+4                ; 0548
                                         56 DD 00069        PUSHL   ICB                                 ; 0555
                                     04  AE 9F 0006B        PUSHAB  NAME_STRING                         ; 0549
                                   FE79  CA 9F 0006E        PUSHAB  AUX_NAM                             ;
                                   FE29  CA 9F 00072        PUSHAB  AUX_FAB                             ;
                                   FF7E  CF 9F 00076        PUSHAB  P.AAA                               ; 0551
                                     F8  AD 9F 0007A        PUSHAB  NAME_DESC                           ; 0549
                     00000000G  00       06 FB 0007D        CALLS   #6, IMG$OPEN_IMAGE                  ;
                                57       50 D0 00084        MOVL    R0, STATUS                          ;
                                72       57 E9 00087        BLBC    STATUS, 8$                          ; 0556
                        B4      AD F979  CA 9E 0008A        MOVAB   AUX_BUFFER, IHD_CTX                 ; 0561
                        B8      AD FB79  CA 9E 00090        MOVAB   AUX_IHD, IHD_CTX+4                  ; 0562
                                         56 DD 00096        PUSHL   ICB                                 ; 0564
                     00000000G  00       01 FB 00098        CALLS   #1, IMG$GET_HEADER                  ;
                                57       50 D0 0009F        MOVL    R0, STATUS                          ;
                                76       57 E9 000A2        BLBC    STATUS, 12$                         ; 0565
              38        10      A6       04 E1 000A5 5$:    BBC     #4, 16(ICB), 7$                     ; 0571
                                50   58  A6 D0 000AA        MOVL    88(ICB), R0                         ; 0575
                                50   04  A0 D0 000AE        MOVL    4(R0), R0                           ; 0576
                                50   10  A0 9A 000B2        MOVZBL  16(R0), R0                          ; 0577
                                50 00001800 8F C4 000B6     MULL2   #6144, R0                           ;
              58                50       10 C7 000BD        DIVL3   #16, R0, SIZE                       ;
                                51       58 D0 000C1        MOVL    SIZE, R1                            ; 0578
                                00000000G 00 16 000C4       JSB     EXE$ALOP1PROC                       ;
                                58       51 D0 000CA        MOVL    R1, R8                              ;
                                6A       52 D0 000CD        MOVL    R2, OWN_STORAGE+40                  ;
                                06       50 E8 000D0        BLBS    R0, 6$                              ;
                                50 0124  8F 3C 000D3        MOVZWL  #292, R0                            ; 0579
                                         04 000D8           RET                                         ;
                                50       6A D0 000D9 6$:    MOVL    OWN_STORAGE+40, R0                  ; 0580
     58       00                6E       00 2C 000DC        MOVC5   #0, (SP), #0, SIZE, (R0)            ;
                                         60    000E1                                                    ;
                                         56 DD 000E2 7$:    PUSHL   ICB                                 ; 0583
                        0000V   CF       01 FB 000E4        CALLS   #1, PROCESS_ISD_LIST                ;
                                57       50 D0 000E9        MOVL    R0, STATUS                          ;
                                0F       57 E8 000EC        BLBS    STATUS, 9$                          ; 0584
              27        10      A6       04 E1 000EF        BBC     #4, 16(ICB), 12$                    ; 0590
                                51       58 D0 000F4        MOVL    SIZE, R1                            ; 0592
                                50       6A D0 000F7        MOVL    OWN_STORAGE+40, R0                  ;
                                         6B 16 000FA        JSB     EXE$DEAP1                           ;
                                         1D 11 000FC 8$:    BRB     12$                                 ; 0594
              03        10      A6       04 E0 000FE 9$:    BBS     #4, 16(ICB), 11$                    ; 0599
                                   FF0F  31 00103 10$:      BRW     1$                                  ;
                                         56 DD 00106 11$:   PUSHL   ICB                                 ; 0602
                        0000V   CF       01 FB 00108        CALLS   #1, LOAD_SEQ_IMAGE                  ;
```

```
     57        50 D0 0010D          MOVL    R0, STATUS                 ;
     51        58 D0 00110          MOVL    SIZE, R1                   ; 0603
     50        6A D0 00113          MOVL    OWN_STORAGE+40, R0         ;
               6B 16 00116          JSB     EXE$DEAP1                  ;
     E8        57 E8 00118          BLBS    STATUS, 10$                ; 0604
     50        57 D0 0011B  12$:    MOVL    STATUS, R0                 ;
                  04 0011E          RET                                ;
     50        01 D0 0011F  13$:    MOVL    #1, R0                     ; 0611
                  04 00122          RET                                ; 0613

; Routine Size:  291 bytes,    Routine Base:  YF$$SYSIMGACT + 0018
```

```
1 %SBTTL 'PROCESS_ISD_LIST - Convert Each ISD into Mapping Request'
1
1 ROUTINE PROCESS_ISD_LIST (ICB_PTR) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine converts each ISD in the current image into mapping
1 !       requests to one of the memory management system services. Global
1 !       ISDs are converted into new work list items to be processes at a
1 !       later time.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       PROCESS_ISD_LIST (ICB pointer)
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       TBS
1 !
1 ! STATUS CODES:
1 !
1 !       TBS
1 !--
1
2 BEGIN
2
2 BIND
2     ICB     = .ICB_PTR                    : $BBLOCK,
2     KFE     = .ICB [ICB$L_KFE]            : $BBLOCK,
2     IHD_CTX = .ICB [ICB$L_CONTEXT]        : $BBLOCK,
2     FLAGS   = OWN_STORAGE [INPUT_FLAGS]   : $BBLOCK;
2
2 LOCAL
2     SEQ_ISD                        : REF VECTOR [],
2     GBL_ICB                        : REF $BBLOCK
2                                      INITIAL (0),
2     ISD_BUFFER                     : $BBLOCK [ISD$K_MAXLENGLBL],
2     ISD                            : REF $BBLOCK
2                                      INITIAL (0),
2     GLOBAL_SEC_NAME_DESC           : $BBLOCK [DSC$K_S_BLN],
2     MAP_BASE_ADDRESS,
2     ISD_BASE_ADDRESS,
2     PAGE_COUNT,
2     INPUT_RANGE                    : VECTOR [2],
2     RETURN_RANGE                   : VECTOR [2],
2     ACCUMULATED_RANGE              : VECTOR [2]
2                                      INITIAL (0,0),
2     SECTION_FLAGS,
2     BASE_ADDRESS_STORED            : INITIAL (FALSE),
2     FIRST_MAPPING                  : INITIAL (TRUE),
2     STATUS;
2
2 ! A base address is selected that will determine the actual range passed to
2 ! the various mapping requests. If the ICB contains an explicit base address,
2 ! that address is used. Otherwise, the address following the last mapping
2 ! request is used. Note that this latter situation includes the case where
```

```
! no input range is specified for the activation of a main program. Because
! the CHECK_PARAMS routine sets RETURN_END_RANGE to FFFFFFFF, this logic
! chooses 0 as MAP_BASE_ADDRESS.

MAP_BASE_ADDRESS = (
    IF .ICB [ICB$V_EXPREG]
    THEN
        .OWN_STORAGE [RETURN_END_ADDRESS] + 1
    ELSE
        .ICB [ICB$L_STARTING_ADDRESS]);

! Remember where ISD storage is for a sequential device load
! The storage location is meaningless in all other case

SEQ_ISD = .OWN_STORAGE [SEQ_LOAD_ISDS];

WHILE TRUE DO                               ! Idiom for DO FOREVER

    BEGIN

    ! If the header is resident, then it is not necessary to decode the ISDs
    ! as that has already been done. For images where ISD decoding is
    ! required, the context passed from IMG$DECODE_IHD to IMG$GET_NEXT_ISD is
    ! contained in a context block located through an ICB pointer.

    IF .ICB [ICB$V_RES_HEADER]
    THEN
        BEGIN

        BIND
            IHD = .KFE [KFE$L_IMGHDR]   : $BBLOCK;

        ISD = (IF .ISD EQL 0
              THEN IHD + .IHD [IHD$W_SIZE]
              ELSE .ISD + .ISD [ISD$W_SIZE]);

        ! The next test is the loop breaker for the resident header case.
        ! An ISD size of zero indicates the end of the ISD list.

        IF .ISD [ISD$W_SIZE] EQL 0
            THEN
                BEGIN

                ! Update the address range in the ICB with the accumulated
                ! range into which the image was mapped.

                ICB [ICB$L_STARTING_ADDRESS] = .ACCUMULATED_RANGE [0];
                ICB [ICB$L_END_ADDRESS] = .ACCUMULATED_RANGE [1];

                RETURN SS$_NORMAL
                END;
        END
    ELSE
        BEGIN

        STATUS = IMG$GET_NEXT_ISD (
            .ICB [ICB$W_CHAN],
```

```
                .IHD_CTX [CTX_L_BUFFER],
                .IHD_CTX [CTX_L_IHDBUF],
                IHD_CTX [CTX_L_VBN],
                IHD_CTX [CTX_W_ISD_OFFSET],
                ISD_BUFFER);

            ! The following test is the loop breaker in the case where we are
            ! making validity checks on the ISD contents as we go along. The
            ! routine IMG$GET_NEXT_ISD returns a status of IMG$_ENDOFHDR when it
            ! reaches the end of the ISD list.

        IF NOT .STATUS THEN
            IF .STATUS EQL IMG$_ENDOFHDR
            THEN
                BEGIN

                ! Update the address range in the ICB with the accumulated
                ! range into which the image was mapped.

                ICB [ICB$L_STARTING_ADDRESS] = .ACCUMULATED_RANGE [0];
                ICB [ICB$L_END_ADDRESS] = .ACCUMULATED_RANGE [1];

                RETURN SS$_NORMAL
                END
            ELSE
                RETURN .STATUS;

        ISD = ISD_BUFFER;

        END;

    ! The common mapping parameters are computed from the ISD contents and
    ! previous mapping context and stored in a suitable place for use by the
    ! various memory management system services.
    !
    ! The set of addresses that are passed to the various memory
    ! management system services depend on the kind of image that is
    ! being activated, and the kind of ISDs that exist within  these
    ! images. The various cases are described in an approximate order
    ! of occurrence.
    !
    ! 1.  New shareable images that are not based (that is, they
    !     are PIC and based at zero) have the ISD addresses added
    !     to the highest address previously mapped.
    !
    ! 2.  New based shareable images use the addresses contained
    !     in the ISDs.
    !
    ! 3.  When processing old shareable images, the addresses
    !     contained in the ISDs of the main program are taken at
    !     face value. If the "last cluster" flag is set, the end
    !     address is set to 7FFFFFFF to disable all upper limit
    !     address checks. Note that the last cluster flag is
    !     ignored in new shareable images.

    !!! NOW THAT I HAVE THIS WONDERFUL COMMENT, I OUGHT TO MAKE THE CODE
    !!! BEHAVE THE WAY THAT IT'S SUPPOSED TO.
```

```
    PAGE_COUNT = .ISD [ISD$W_PAGCNT];
    IF .PAGE_COUNT EQL 0 THEN RETURN SS$_BADISD;

    ISD_BASE_ADDRESS = ((.ISD [ISD$V_VPG]) * BYTES_PER_PAGE);

    IF NOT .BASE_ADDRESS_STORED
    THEN
        BEGIN

        ! Renormalize the base address for merged activations and for all
        ! ICBs that represent implicitly referenced shareable images.

        IF .ICB [ICB$B_ACT_CODE] NEQ ICB$K_MAIN_PROGRAM
        THEN
            MAP_BASE_ADDRESS = .MAP_BASE_ADDRESS - .ISD_BASE_ADDRESS;

        ! The base address stored in the ICB is used by $IMGFIX to perform
        ! address relocation fixups. The transfer address bias is used to
        ! adjust transfer addresses that lie within the bounds of an image's
        ! address space.

        ICB [ICB$L_BASE_ADDRESS] = .MAP_BASE_ADDRESS + .ISD_BASE_ADDRESS;
        BASE_ADDRESS_STORED = TRUE;

        IF NOT .OWN_STORAGE [TRANSFER_BIAS_STORED]
        THEN
            BEGIN
            OWN_STORAGE [TRANSFER_ARRAY_BIAS] = .MAP_BASE_ADDRESS;
            OWN_STORAGE [TRANSFER_BIAS_STORED] = TRUE;
            END;
        END;

    INPUT_RANGE [0] = .MAP_BASE_ADDRESS + .ISD_BASE_ADDRESS;
    INPUT_RANGE [1] = .INPUT_RANGE [0] + ((.PAGE_COUNT * BYTES_PER_PAGE) - 1);

    CASE .ISD [ISD$L_FLAGS]                 ! Form case index from low four
            AND                             !  bits of the ISD flags longword
        ISD_M_DISPATCH_FLAGS
    FROM 0 TO ISD_M_DISPATCH_FLAGS OF

    SET
```

```
  3 %SBTTL 'Create Address Space for Demand-Zero ISD'
  3
  3       [ISD$M_DZRO OR ISD$M_WRT , ISD$M_DZRO OR ISD$M_WRT OR ISD$M_CRF]:
  3
  3       ! The section is a demand-zero section. The pages are created at the
  3       ! current end of the image.
  3
  4           BEGIN
  4
  4           IF .ISD [ISD$B_TYPE] NEQ ISD$K_USRSTACK
  4           THEN
  5               BEGIN
P 5               STATUS = $CRETVA (
P 5                   INADR = INPUT_RANGE ,
P 5                   RETADR = RETURN_RANGE ,
  5                   ACMODE = .OWN_STORAGE [ACCESS_MODE] );
  5               IF NOT .STATUS THEN RETURN .STATUS;
  5               END
  4           ELSE
  4               OWN_STORAGE [USER_STACK_SIZE] =
  4                   .OWN_STORAGE [USER_STACK_SIZE] + .PAGE_COUNT;
  3           END;
  3
```

```
  3 %SBTTL 'Create or Map Address Space for Private ISD'
  3
  3      [ISD$M_CRF OR ISD$M_WRT , ISD$M_WRT , 0]:
  3
  3      ! The section is a private section that will be mapped into the process
  3      ! address space. The section may be read only or writable. Read-only
  3      ! sections may either be global sections or private sections. Writable CRF
  3      ! sections are always created as private sections. Writable sections that
  3      ! are not also copy on reference are always mapped as global global
  3      ! sections.
  3
  4          BEGIN
  4          LITERAL SEC_M_EXEC_OWNED = PSL$C_EXEC ^ $BITPOSITION (SEC$V_WRTMOD);
  4
  4          ! If the SHAREABLE bit is set in the ICB, then global sections were
  4          ! created for the read-only sections of this image. These global
  4          ! sections can be mapped. In other cases, private sections are
  4          ! created.
  4
  4          SECTION_FLAGS = (.ISD [ISD$L_FLAGS] AND ISD_M_DISPATCH_FLAGS);
  4
  4          ! If the section is marked protected, then the ownership and
  4          ! writability of the section pages is restricted to exec mode.
  4
  4          IF .ISD [ISD$V_PROTECT]
  4          THEN
  4              SECTION_FLAGS = .SECTION_FLAGS
  4                              OR
  4                              SEC$M_PROTECT
  4                              OR
  4                              SEC_M_EXEC_OWNED;
  4
  4          IF
  4              .ICB [ICB$V_SHAREABLE]
  4              AND
  6                  ((NOT .ISD [ISD$V_WRT])
  5                  OR
  5                  (.ISD [ISD$V_WRT] AND NOT .ISD [ISD$V_CRF]))
  4          THEN
  5              BEGIN
  5
  5              BIND
  5                  GSD_NAME = IHD_CTX [CTX_T_GSD_NAME] : VECTOR [, BYTE];
  5
  5              GLOBAL_SEC_NAME_DESC [DSC$W_LENGTH] = .GSD_NAME [0] + 4;
  5              GLOBAL_SEC_NAME_DESC [DSC$A_POINTER] = GSD_NAME [1];
  5
  5              NEXT_GBL_SEC_NAME (GSD_NAME);
  5
P 5              STATUS = $MGBLSC (
P 5                  INADR  = INPUT_RANGE ,
P 5                  RETADR = RETURN_RANGE ,
P 5                  ACMODE = .OWN_STORAGE [ACCESS_MODE] ,
P 5                  FLAGS  = (.SECTION_FLAGS OR SEC$M_GBL OR SEC$M_SYSGBL) ,
P 5                  GSDNAM = GLOBAL_SEC_NAME_DESC ,
  5                  IDENT  = ICB [ICB$Q_IDENT] );
  5              IF NOT .STATUS THEN RETURN .STATUS;
```

```
                END
            ELSE
                BEGIN

                ! Private copies of writable sections that are not also copy on
                ! reference are prohibited.

                IF .ISD [ISD$V_WRT] AND NOT .ISD [ISD$V_CRF]
                THEN
                    RETURN SS$_NOTINSTALL;

                ! Check for loading from a sequential device before deciding
                ! whether to map the section or simply record the load data

                IF NOT .ICB [ICB$V_LOAD_IMAGE]
                THEN
                    BEGIN
                    STATUS = $CRMPSC (
                        INADR  = INPUT_RANGE ,
                        RETADR = RETURN_RANGE ,
                        ACMODE = .OWN_STORAGE [ACCESS_MODE] ,
                        FLAGS  = .SECTION_FLAGS ,
                        CHAN   = .ICB [ICB$W_CHAN] ,
                        PAGCNT = .PAGE_COUNT ,
                        VBN    = .ISD [ISD$L_VBN] ,
                        PFC    = .ISD [ISD$B_PFC] );
                    IF NOT .STATUS THEN RETURN .STATUS;
                    END
                ELSE

                    ! Record size of image section, starting address, and flags
                    ! Advance pointer for the next ISD

                    BEGIN
                    SEQ_ISD [0] = .PAGE_COUNT;
                    SEQ_ISD [1] = .INPUT_RANGE[0];
                    SEQ_ISD [2] = .ISD [ISD$L_FLAGS];
                    SEQ_ISD = SEQ_ISD [3];
                    END;

                ! If the section just mapped is a fixup vector, then it must be
                ! verified and added to the list of fixup vectors. Note that the
                ! placement of this code assumes that fixup sections can never be
                ! shared. That is, they are always writable CRF sections.

                IF (
                    (.ISD [ISD$V_FIXUPVEC])
                    AND
                    (NOT (.OWN_STORAGE [P1_MERGE_P0]))
                   )
                THEN

                    ! If the image being mapped is a privileged shareable image,
                    ! then perform the fixups immediately in exec mode. Otherwise,
                    ! add the fixup vector to the list of fixups to be done.
                    ! This work is delayed for images loaded from sequential devices
```

```
                IF .ISD [ISD$V_PROTECT]
                THEN
                    STATUS = (
                        IF
                            BEGIN
                            IF (KFE EQL 0)
                            THEN
                                TRUE
                            ELSE
                                NOT (.KFE [KFE$V_PROTECT] AND .KFE [KFE$V_SHARED])
                            END
                        THEN
                            SS$_PROTINSTALL
                        ELSE
                            IMG$PRVSHRIMG (
                                .RETURN_RANGE,
                                .ICB [ICB$L_BASE_ADDRESS]))
                ELSE
                    IF NOT .ICB [ICB$V_LOAD_IMAGE]
                    THEN
                        STATUS = ADD_FIXUP_VECTOR (
                            RETURN_RANGE,
                            .ICB [ICB$L_BASE_ADDRESS]);
            IF NOT .STATUS THEN RETURN .STATUS
            END;

        ! At this point, we need to check whether the section just mapped
        ! is a message section or a vector section. We need to put the check
        ! here because, while vector sections can only be global sections,
        ! message sections can be either private or global.
        ! This work is delayed for images loaded from sequential devices

        IF (
            (.ISD [ISD$V_VECTOR])
            AND
            (NOT (.OWN_STORAGE [P1_MERGE_P0]))
           )
        THEN
            IF .ISD [ISD$V_PROTECT]
            THEN
                BEGIN
                IF NOT .ICB [ICB$V_LOAD_IMAGE]
                THEN
                    BEGIN
                    STATUS = ADD_PRIVILEGED_VECTOR (RETURN_RANGE [0], ICB);
                    IF NOT .STATUS THEN RETURN .STATUS
                    END;
                END
            ELSE
                RETURN SS$_BADISD;

        END;
```

```
%SBTTL 'Add ICB to Work List for Global ISD'

[ISD$M_GBL, ISD$M_GBL OR ISD$M_WRT, ISD$M_GBL OR ISD$M_WRT OR ISD$M_CRF, ISD$M_GBL OR ISD$M_WRT OR ISD$M_DZR

        ! This is a global ISD. An entry will be made in the work list and the
        ! actual mapping will occur in a later call to this routine. A check is
        ! made to determine whether we are processing a list of global ISDs of
        ! the same name. This is done by comparing the name in the global ISD
        ! with the name in the most recently allocated ICB.

            BEGIN

            BIND
                ISD_NAM = ISD [ISD$T_GBLNAM]            : VECTOR [,BYTE];

            LOCAL
                ICB_NAM                                 : REF VECTOR [,BYTE];

            IF
                BEGIN
                    IF (.GBL_ICB EQL 0)
                    THEN TRUE
                    ELSE
                        BEGIN
                        ICB_NAM = GBL_ICB [ICB$T_IMAGE_NAME];
                        CH$NEQ (
                            .ISD_NAM [0] - 4, ISD_NAM [1],
                            .ICB_NAM [0], ICB_NAM [1], 0)
                        END
                END
            THEN
                BEGIN

                STATUS = IMG$ALLOCATE_ICB (GBL_ICB);
                IF NOT .STATUS THEN RETURN .STATUS;

                GBL_ICB [ICB$B_ACT_CODE] = ICB$K_GLOBAL_IMAGE_SECTION;

                ICB_NAM = GBL_ICB [ICB$T_IMAGE_NAME];

                ! Move the section name including the count and the
                ! trailing _00n into the ICB. Insert the ICB at the tail of the
                ! work list. Subtract four from the count so that the _00n prefix,
                ! while present in the ICB, is not represented in the count.

                ICB_NAM [0] = .ISD_NAM [0] - 4;
                CH$MOVE (.ISD_NAM [0], ISD_NAM [1], ICB_NAM [1]);

                GBL_ICB [ICB$L_MATCH_CONTROL] = .ISD [ISD$V_MATCHCTL];
                GBL_ICB [ICB$L_VERSION] = .ISD [ISD$L_IDENT];
                IF
                    (.IHD_CTX [CTX_W_GENERATION] GEQU IHD$C_GEN_FIXUP)
                    AND
                    NOT (.ISD [ISD$V_BASED])
                THEN
                    BEGIN
                    GBL_ICB [ICB$V_EXPREG] = TRUE;
```

```
                        GBL_ICB [ICB$L_STARTING_ADDRESS] = 0;
                        GBL_ICB [ICB$L_END_ADDRESS] = END_OF_P1_SPACE;
                        END
                    ELSE
                        BEGIN
                        GBL_ICB [ICB$L_STARTING_ADDRESS] = .INPUT_RANGE [0];
                        GBL_ICB [ICB$L_END_ADDRESS] = (
                            IF .ISD [ISD$V_LASTCLU]
                            THEN END_OF_P1_SPACE
                            ELSE .INPUT_RANGE [1] );
                        END;

                    INSQUE (.GBL_ICB , .IAC$GL_WORK_LIST [1]);
                    END
                ELSE
                    BEGIN

                    ! This is another global ISD of the same name. Some consistency
                    ! checks are performed on the address range. The address range
                    ! stored in the ICB may change.

                    IF (.INPUT_RANGE [0] LEQU .GBL_ICB [ICB$L_STARTING_ADDRESS])
                    THEN RETURN SS$_BADISD;

                    IF .GBL_ICB [ICB$L_END_ADDRESS] NEQ END_OF_P1_SPACE
                    THEN
                        BEGIN

                        IF (.INPUT_RANGE [0] LEQU .GBL_ICB [ICB$L_END_ADDRESS])
                        THEN RETURN SS$_BADISD;

                        GBL_ICB [ICB$L_END_ADDRESS] = .INPUT_RANGE [1];
                        END;
                    END;

                ! If a global ISD has the WRT flag set but the CRF and DZRO bits clear,
                ! this indicates a writable global section. A flag is set to indicate
                ! that the image file should be opened for write access.

                IF .ISD [ISD$V_WRT] AND NOT (.ISD [ISD$V_CRF] OR .ISD [ISD$V_DZRO])
                THEN
                    GBL_ICB [ICB$V_OPEN_FOR_WRITE] = TRUE;

                END;
```

```
3 %SBTTL 'Update return address arrays'
3
3       [INRANGE]:
3
3       ! An unrecognized combination of flags exists in this ISD. Because the
3       ! linker would never produce such a combination, this is either not an
3       ! image file or some tampering has occurred.
3
3           RETURN SS$_BADISD;
3
3       TES;
3
3   ! Update the address range arrays in the ICB and in impure storage.
3   ! The latter may be returned to the caller and represents the total
3   ! address space mapped in this activation.
3   !
3   ! Note that the following conditional update works correctly because
3   ! ALL of the nonglobal ISDs and NONE of the global ISDs result in
3   ! mapping requests.
3
3   IF NOT .ISD [ISD$V_GBL]
3   THEN
4       BEGIN
4       IF .FIRST_MAPPING
4       THEN
5           BEGIN
5           FIRST_MAPPING = FALSE;
5           ACCUMULATED_RANGE [0] = .RETURN_RANGE [0];
4           END;
4
4       IF .OWN_STORAGE [RETURN_START_ADDRESS] EQL -1
4       THEN
4           OWN_STORAGE [RETURN_START_ADDRESS] = .RETURN_RANGE [0];
4
4       ACCUMULATED_RANGE [1] = .RETURN_RANGE [1];
4
4       IF .RETURN_RANGE [1]  GTR  .OWN_STORAGE [RETURN_END_ADDRESS]
4       THEN
4           OWN_STORAGE [RETURN_END_ADDRESS] = .RETURN_RANGE [1];
4
3       END;
3
3   END                                         ! End of DO FOREVER loop
3
1   END;                                        ! End of routine PROCESS_ISD_LIST


                                                .EXTRN  SYS$CRETVA, SYS$MGBLSC
                                                .EXTRN  SYS$CRMPSC

                                OFFC 00000 PROCESS_ISD_LIST:
                                                .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11   ; 0616
            5E      FF78    CE  9E 00002        MOVAB   -136(SP), SP                           ;
            58        04    AC  D0 00007        MOVL    ICB_PTR, R8                            ; 0642
       10   AE        54    A8  D0 0000B        MOVL    84(R8), 16(SP)                         ; 0643
            5A        58    A8  D0 00010        MOVL    88(R8), R10                            ; 0644
```

```
                                   59  D4 00014          CLRL    ISD                                              ;
                            24     AE  7C 00016          CLRQ    GBL_ICB                                          ;
                            2C     AE  D4 00019          CLRL    ACCUMULATED_RANGE+4                              ;
              08     AE            01  7D 0001C          MOVQ    #1, FIRST_MAPPING                                ;
              04     AE     10     A8  9E 00020          MOVAB   16(R8), 4(SP)                                    ; 0676
                     0A     04     BE  E9 00025          BLBC    @4(SP), 1$                                       ;
       6E 00000000G  00            01  C1 00029          ADDL3   #1, OWN_STORAGE+88, MAP_BASE_ADDRESS             ; 0678
                                   04  11 00031          BRB     2$                                               ;
                     6E     48     A8  D0 00033 1$:      MOVL    72(R8), MAP_BASE_ADDRESS                         ; 0680
              18     AE 00000000G  00  D0 00037 2$:      MOVL    OWN_STORAGE+40, SEQ_ISD                          ; 0685
       23            04     BE     03  E1 0003F 3$:      BBC     #3, @4(SP), 6$                                   ; 0696
                                   59  D5 00044          TSTL    ISD                                              ; 0703
                                   13  12 00046          BNEQ    4$                                               ;
       51     10     AE            1C  C1 00048          ADDL3   #28, 16(SP), R1                                  ; 0704
                     50            91  3C 0004D          MOVZWL  @(R1)+, R0                                       ;
       52     10     AE            1C  C1 00050          ADDL3   #28, 16(SP), R2                                  ;
       59            62            50  C1 00055          ADDL3   R0, (R2), ISD                                    ;
                                   06  11 00059          BRB     5$                                               ;
                     50            69  3C 0005B 4$:      MOVZWL  (ISD), R0                                        ; 0705
                     59            50  C0 0005E          ADDL2   R0, ISD                                          ;
                                   69  B5 00061 5$:      TSTW    (ISD)                                            ; 0710
                                   3B  12 00063          BNEQ    9$                                               ;
                                   2C  11 00065          BRB     7$                                               ; 0717
                            48     AE  9F 00067 6$:      PUSHAB  ISD_BUFFER                                       ; 0731
                            0C     AA  9F 0006A          PUSHAB  12(R10)                                          ;
                            08     AA  9F 0006D          PUSHAB  8(R10)                                           ; 0730
                     7E            6A  7D 00070          MOVQ    (R10), -(SP)                                     ; 0731
                     7E     0E     A8  3C 00073          MOVZWL  14(R8), -(SP)                                    ;
          00000000G  00            06  FB 00077          CALLS   #6, IMG$GET_NEXT_ISD                             ;
                 14  AE            50  D0 0007E          MOVL    R0, STATUS                                       ;
                     16     14     AE  E8 00082          BLBS    STATUS, 8$                                       ; 0739
           084D8640  8F     14     AE  D1 00086          CMPL    STATUS, #139298368                               ; 0740
                                   03  13 0008E          BEQL    7$                                               ;
                                 0232  31 00090          BRW     38$                                              ;
              48     A8     28     AE  7D 00093 7$:      MOVQ    ACCUMULATED_RANGE, 72(R8)                        ; 0747
                     50            01  D0 00098          MOVL    #1, R0                                           ; 0753
                                       04 0009B          RET                                                      ;
                     59     48     AE  9E 0009C 8$:      MOVAB   ISD_BUFFER, ISD                                  ; 0755
                     56     02     A9  3C 000A0 9$:      MOVZWL  2(ISD), PAGE_COUNT                               ; 0786
                                   03  12 000A4          BNEQ    10$                                              ; 0787
                                 02BD  31 000A6          BRW     46$                                              ;
 50           04     A9            17  00  EF 000A9 10$: EXTZV   #0, #23, 4(ISD), R0                              ; 0789
              20     AE            50  09  78 000AF      ASHL    #9, R0, ISD_BASE_ADDRESS                         ;
                     2A     0C     AE  E8 000B4          BLBS    BASE_ADDRESS_STORED, 12$                         ; 0791
                     01     0D     A8  91 000B8          CMPB    13(R8), #1                                       ; 0798
                                   04  13 000BC          BEQL    11$                                              ;
                     6E     20     AE  C2 000BE          SUBL2   ISD_BASE_ADDRESS, MAP_BASE_ADDRESS               ; 0800
       5C     A8     6E     20     AE  C1 000C2 11$:     ADDL3   ISD_BASE_ADDRESS, MAP_BASE_ADDRESS, 92(R8)       ; 0807
              0C     AE            01  D0 000C8          MOVL    #1, BASE_ADDRESS_STORED                          ; 0808
       0E 00000000G  00            04  E0 000CC          BBS     #4, OWN_STORAGE, 12$                             ; 0810
          00000000G  00            6E  D0 000D4          MOVL    MAP_BASE_ADDRESS, OWN_STORAGE+24                 ; 0813
          00000000G  00            10  88 000DB          BISB2   #16, OWN_STORAGE                                 ; 0814
       38     AE     6E     20     AE  C1 000E2 12$:     ADDL3   ISD_BASE_ADDRESS, MAP_BASE_ADDRESS, -            ; 0818
                                                                 INPUT_RANGE                                      ;
       50            56            09  78 000E8          ASHL    #9, PAGE_COUNT, R0                               ; 0819
                     50     38     AE  C0 000EC          ADDL2   INPUT_RANGE, R0                                  ;
              3C     AE     FF     A0  9E 000F0          MOVAB   -1(R0), INPUT_RANGE+4                            ;
```

IMG
V04

```
; 1
; 1
; 1
; 1
; 1
; 1
; 1
; 1
; 1
; 1
; 1
; 1
; 1
; 1
; 1
; 1
; 1
; 1
; 1
; 1
; 1
; 1
; 1
; 1
```

```
                              5B      08  A9  9E 000F5        MOVAB   8(ISD), R11                           ; 0821
      7E          6B          04          00  EF 000F9        EXTZV   #0, #4, (R11), -(SP)                  ; 0822
                  0F          00          8E  CF 000FE        CASEL   (SP)+, #0, #15                        ;
    0264        0264        0190        004F     00102 13$:   .WORD   18$-13$,-                             ;
    0264        0264        0264        0264     0010A                36$-13$,-                             ;
    0190        004F        0190        004F     00112                46$-13$,-                             ;
    0264        0020        0190        0020     0011A                46$-13$,-                             ;
                                                                      46$-13$,-                             ;
                                                                      46$-13$,-                             ;
                                                                      46$-13$,-                             ;
                                                                      46$-13$,-                             ;
                                                                      18$-13$,-                             ;
                                                                      36$-13$,-                             ;
                                                                      18$-13$,-                             ;
                                                                      36$-13$,-                             ;
                                                                      14$-13$,-                             ;
                                                                      36$-13$,-                             ;
                                                                      14$-13$,-                             ;
                                                                      46$-13$                               ;
                          FD  8F      0B  A9  91 00122 14$:   CMPB    11(ISD), #253                         ; 0836
                                          1E  13 00127        BEQL    16$                                   ;
                              00000000G   00  DD 00129        PUSHL   OWN_STORAGE+100                       ; 0842
                                      34  AE  9F 0012F        PUSHAB  RETURN_RANGE                          ;
                                      40  AE  9F 00132        PUSHAB  INPUT_RANGE                           ;
                   00000000G  00          03  FB 00135        CALLS   #3, SYS$CRETVA                        ;
                          14  AE          50  D0 0013C 15$:   MOVL    R0, STATUS                            ;
                              0A      14  AE  E8 00140        BLBS    STATUS, 17$                           ; 0843
                                        017E  31 00144        BRW     38$                                   ;
                   00000000G  00          56  C0 00147 16$:   ADDL2   PAGE_COUNT, OWN_STORAGE+28            ; 0847
                                        021B  31 0014E 17$:   BRW     47$                                   ; 0821
 1C    AE         6B          04          00  EF 00151 18$:   EXTZV   #0, #4, (R11), SECTION_FLAGS          ; 0869
                  08          6B          12  E1 00157        BBC     #18, (R11), 19$                       ; 0874
                          1C  AE 00040040 8F  C8 0015B        BISL2   #262208, SECTION_FLAGS                ; 0879
                  4D      04  BE          01  E1 00163 19$:   BBC     #1, @4(SP), 22$                       ; 0883
                  04          6B          03  E1 00168        BBC     #3, (R11), 20$                        ; 0885
                  45          6B          01  E0 0016C        BBS     #1, (R11), 22$                        ; 0887
                          40  AE      14  AA  9B 00170 20$:   MOVZBW  20(R10), GLOBAL_SEC_NAME_DESC         ; 0894
                          40  AE          04  A0 00175        ADDW2   #4, GLOBAL_SEC_NAME_DESC              ;
                          44  AE      15  AA  9E 00179        MOVAB   21(R10), GLOBAL_SEC_NAME_DESC+4       ; 0895
                                      14  AA  9F 0017E        PUSHAB  20(R10)                               ; 0897
                       0000V  CF          01  FB 00181        CALLS   #1, NEXT_GBL_SEC_NAME                 ;
                                          7E  D4 00186        CLRL    -(SP)                                 ; 0905
                                      40  A8  9F 00188        PUSHAB  64(R8)                                ;
                                      48  AE  9F 0018B        PUSHAB  GLOBAL_SEC_NAME_DESC                  ;
                  7E      28  AE 00008001 8F  C9 0018E        BISL3   #32769, SECTION_FLAGS, -(SP)          ;
                              00000000G   00  DD 00197        PUSHL   OWN_STORAGE+100                       ;
                                      44  AE  9F 0019D        PUSHAB  RETURN_RANGE                          ;
                                      50  AE  9F 001A0        PUSHAB  INPUT_RANGE                           ;
                   00000000G  00          07  FB 001A3        CALLS   #7, SYS$MGBLSC                        ;
                          14  AE          50  D0 001AA 21$:   MOVL    R0, STATUS                            ;
                              47      14  AE  E9 001AE        BLBC    STATUS, 24$                           ; 0906
                                        00B5  31 001B2        BRW     32$                                   ;
                  0A          6B          03  E1 001B5 22$:   BBC     #3, (R11), 23$                        ; 0914
                  06          6B          01  E0 001B9        BBS     #1, (R11), 23$                        ;
                              50    2014  8F  3C 001BD        MOVZWL  #8212, R0                             ; 0916
                                          04     001C2        RET                                           ;
                  34      04  BE          04  E0 001C3 23$:   BBS     #4, @4(SP), 25$                       ; 0921
```

```
                     7E         07   A9  9A 001C8         MOVZBL  7(ISD), -(SP)                        ; 0932
                                     7E  D4 001CC         CLRL    -(SP)                                ;
                                0C   A9  DD 001CE         PUSHL   12(ISD)                              ;
                                     56  DD 001D1         PUSHL   PAGE_COUNT                           ;
                     7E         0E   A8  3C 001D3         MOVZWL  14(R8), -(SP)                        ;
                                     7E  7C 001D7         CLRQ    -(SP)                                ;
                                     7E  D4 001D9         CLRL    -(SP)                                ;
                                3C   AE  DD 001DB         PUSHL   SECTION_FLAGS                        ;
                         00000000G   00  DD 001DE         PUSHL   OWN_STORAGE+100                      ;
                                58   AE  9F 001E4         PUSHAB  RETURN_RANGE                         ;
                                64   AE  9F 001E7         PUSHAB  INPUT_RANGE                          ;
         00000000G   00              0C  FB 001EA         CALLS   #12, SYS$CRMPSC                      ;
                14   AE              50  D0 001F1         MOVL    R0, STATUS                           ;
                     1C         14   AE  E8 001F5         BLBS    STATUS, 26$                          ; 0933
                                   00C9  31 001F9 24$:    BRW     38$                                  ;
                18   BE              56  D0 001FC 25$:    MOVL    PAGE_COUNT, @SEQ_ISD                 ; 0941
     50         18   AE              04  C1 00200         ADDL3   #4, SEQ_ISD, R0                      ; 0942
                     60         38   AE  D0 00205         MOVL    INPUT_RANGE, (R0)                    ;
     50         18   AE              08  C1 00209         ADDL3   #8, SEQ_ISD, R0                      ; 0943
                     60              6B  D0 0020E         MOVL    (R11), (R0)                          ;
                18   AE              0C  C0 00211         ADDL2   #12, SEQ_ISD                         ; 0944
     4D              6B              0A  E1 00215 26$:    BBC     #10, (R11), 31$                      ; 0953
     45  00000000G   00              05  E0 00219         BBS     #5, OWN_STORAGE, 31$                 ; 0955
     2E              6B              12  E1 00221         BBC     #18, (R11), 29$                      ; 0964
                                10   AE  D5 00225         TSTL    16(SP)                               ; 0969
                                     11  13 00228         BEQL    27$                                  ;
     50         10   AE              10  C1 0022A         ADDL3   #16, 16(SP), R0                      ; 0973
                     09              60  E9 0022F         BLBC    (R0), 27$                            ;
     51         10   AE              10  C1 00232         ADDL3   #16, 16(SP), R1                      ;
     08              61              05  E0 00237         BBS     #5, (R1), 28$                        ;
                14   AE       205C   8F  3C 0023B 27$:    MOVZWL  #8284, STATUS                        ; 0967
                                     23  11 00241         BRB     31$                                  ;
                     51         5C   A8  D0 00243 28$:    MOVL    92(R8), R1                           ; 0979
                     50         30   AE  D0 00247         MOVL    RETURN_RANGE, R0                     ;
                         00000000G   00  16 0024B         JSB     IMG$PRVSHRIMG                        ;
                                     10  11 00251         BRB     30$                                  ;
     0E         04   BE              04  E0 00253 29$:    BBS     #4, @4(SP), 31$                      ; 0982
                                5C   A8  DD 00258         PUSHL   92(R8)                               ; 0986
                                34   AE  9F 0025B         PUSHAB  RETURN_RANGE                         ; 0984
             0000V   CF              02  FB 0025E         CALLS   #2, ADD_FIXUP_VECTOR                 ;
                                   FF44  31 00263 30$:    BRW     21$                                  ;
                     5B         14   AE  E9 00266 31$:    BLBC    STATUS, 38$                          ; 0987
     03              6B              11  E0 0026A 32$:    BBS     #17, (R11), 34$                      ; 0997
                                   00FB  31 0026E 33$:    BRW     47$                                  ;
     F5  00000000G   00              05  E0 00271 34$:    BBS     #5, OWN_STORAGE, 33$                 ; 0999
     03              6B              12  E0 00279         BBS     #18, (R11), 35$                      ; 1002
                                   00E6  31 0027D         BRW     46$                                  ;
     E9         04   BE              04  E0 00280 35$:    BBS     #4, @4(SP), 33$                      ; 1005
                                     58  DD 00285         PUSHL   R8                                   ; 1008
                                34   AE  9F 00287         PUSHAB  RETURN_RANGE                         ;
             0000V   CF              02  FB 0028A         CALLS   #2, ADD_PRIVILEGED_VECTOR            ;
                                   FEAA  31 0028F         BRW     15$                                  ;
                     55         14   A9  9E 00292 36$:    MOVAB   20(ISD), R5                          ; 1030
                     57         24   AE  D0 00296         MOVL    GBL_ICB, R7                          ; 1037
                                     17  13 0029A         BEQL    37$                                  ;
                     54         14   A7  9E 0029C         MOVAB   20(R7), ICB_NAM                      ; 1041
                     51              65  9A 002A0         MOVZBL  (R5), R1                             ; 1043
```

```
                                  51            04  C2 002A3           SUBL2   #4, R1                                ;
                                  50            64  9A 002A6           MOVZBL  (ICB_NAM), R0                         ; 1044
         50          00       01  A5            51  2D 002A9           CMPC5   R1, T(R5), #0, R0, 1(ICB_NAM)         ;
                                            01  A4     002AF                                                         ;
                                                7D  13 002B1           BEQL    44$                                   ;
                                      24        AE  9F 002B3 37$:      PUSHAB  GBL_ICB                               ; 1050
                         00000000G   00         01  FB 002B6           CALLS   #1, IMG$ALLOCATE_ICB                  ;
                                14   AE         50  D0 002BD           MOVL    R0, STATUS                            ;
                                     05   14    AE  E8 002C1           BLBS    STATUS, 39$                           ; 1051
                                     50   14    AE  D0 002C5 38$:      MOVL    STATUS, R0                            ;
                                                04     002C9           RET                                           ;
                                     57   24    AE  D0 002CA 39$:      MOVL    GBL_ICB, R7                           ; 1053
                                0D   A7         03  90 002CE           MOVB    #3, 13(R7)                            ;
                                     54   14    A7  9E 002D2           MOVAB   20(R7), ICB_NAM                       ; 1055
                          64         65         04  83 002D6           SUBB3   #4, (R5), (ICB_NAM)                   ; 1062
                                     50         65  9A 002DA           MOVZBL  (R5), R0                              ; 1063
                   01     A4    01   A5         50  28 002DD           MOVC3   R0, 1(R5), 1(ICB_NAM)                 ;
   40   A7                6B         03         04  EF 002E3           EXTZV   #4, #3, (R11), 64(R7)                 ; 1065
                          44         A7   10    A9  D0 002E9           MOVL    16(ISD), 68(R7)                       ; 1066
                                     05   10    AA  B1 002EE           CMPW    16(R10), #5                           ; 1068
                                                15  1F 002F2           BLSSU   40$                                   ;
                          11         6B         09  E0 002F4           BBS     #9, (R11), 40$                        ; 1070
                                10   A7         01  88 002F8           BISB2   #1, 16(R7)                            ; 1073
                                          48    A7  D4 002FC           CLRL    72(R7)                                ; 1074
                                4C   A7 7FFFFFFF 8F D0 002FF           MOVL    #2147483647, 76(R7)                   ; 1075
                                                1A  11 00307           BRB     43$                                   ; 1067
                          48   A7         38    AE  D0 00309 40$:      MOVL    INPUT_RANGE, 72(R7)                   ; 1079
                                                6B  95 0030E           TSTB    (R11)                                 ; 1081
                                                09  18 00310           BGEQ    41$                                   ;
                                50 7FFFFFFF     8F  D0 00312           MOVL    #2147483647, R0                       ;
                                                04  11 00319           BRB     42$                                   ;
                                     50   3C    AE  D0 0031B 41$:      MOVL    INPUT_RANGE+4, R0                     ; 1083
                                4C   A7         50  D0 0031F 42$:      MOVL    R0, 76(R7)                            ; 1080
                                50 00000000G    00  9E 00323 43$:      MOVAB   IAC$GL_WORK_LIST+4, R0                ; 1086
                                00   B0         67  0E 0032A           INSQUE  (R7), @0(R0)                          ;
                                                1D  11 0032E           BRB     45$                                   ; 1035
                          48   A7         38    AE  D1 00330 44$:      CMPL    INPUT_RANGE, 72(R7)                   ; 1095
                                                2F  1B 00335           BLEQU   46$                                   ;
                    7FFFFFFF   8F         4C    A7  D1 00337           CMPL    76(R7), #2147483647                   ; 1098
                                                0C  13 0033F           BEQL    45$                                   ;
                          4C   A7         38    AE  D1 00341           CMPL    INPUT_RANGE, 76(R7)                   ; 1102
                                                1E  1B 00346           BLEQU   46$                                   ;
                          4C   A7         3C    AE  D0 00348           MOVL    INPUT_RANGE+4, 76(R7)                 ; 1105
                  1A      08   A9               03  E1 0034D 45$:      BBC     #3, 8(ISD), 47$                       ; 1113
                  15      08   A9               01  E0 00352           BBS     #1, 8(ISD), 47$                       ;
                  10      08   A9               02  E0 00357           BBS     #2, 8(ISD), 47$                       ;
                               50         24    AE  D0 0035C           MOVL    GBL_ICB, R0                           ; 1115
                          10   A0               04  88 00360           BISB2   #4, 16(R0)                            ;
                                                06  11 00364           BRB     47$                                   ; 0821
                               50       2004    8F  3C 00366 46$:      MOVZWL  #8196, R0                             ; 1126
                                                04     0036B           RET                                           ;
                               38         08    A9  E8 0036C 47$:      BLBS    8(ISD), 50$                           ; 1138
                               08         08    AE  E9 00370           BLBC    FIRST_MAPPING, 48$                    ; 1141
                                          08    AE  D4 00374           CLRL    FIRST_MAPPING                         ; 1144
                          28   AE         30    AE  D0 00377           MOVL    RETURN_RANGE, ACCUMULATED_RANGE       ; 1145
                    FFFFFFFF   8F 00000000G     00  D1 0037C 48$:      CMPL    OWN_STORAGE+84, #-1                   ; 1148
                                                08  12 00387           BNEQ    49$                                   ;
```

```
00000000G  00      30   AE  D0 00389       MOVL    RETURN_RANGE, OWN_STORAGE+84               ; 1150
       2C  AE      34   AE  D0 00391 49$:  MOVL    RETURN_RANGE+4, ACCUMULATED_RANGE+4        ; 1152
00000000G  00      34   AE  D1 00396       CMPL    RETURN_RANGE+4, OWN_STORAGE+88             ; 1154
                        08  15 0039E       BLEQ    50$                                        ;
00000000G  00      34   AE  D0 003A0       MOVL    RETURN_RANGE+4, OWN_STORAGE+88             ; 1156
                      FC94  31 003A8 50$:  BRW     3$                                         ; 0687

; Routine Size:  939 bytes,    Routine Base:  YF$$SYSIMGACT + 013B
```

```
%SBTTL 'ADD_PRIVILEGED_VECTOR - Install User-Written Change Mode Vectors'

ROUTINE ADD_PRIVILEGED_VECTOR (SECTION_ADDRESS, ICB_ADDRESS) =

!+
! Functional Description:
!
! This routine is called when the section just mapped contains
! user-written change mode or message vectors. After verification, the
! vectors are added to the list of privileged routines called in
! response to an unfielded change mode call.
!
! Calling Sequence:
!
!    ADD_PRIVILEGED_VECTOR (SECTION_ADDRESS, KFE_ADDRESS, ICB_ADDRESS)
!
! Input Parameters:
!
!    SECTION_ADDRESS - Address of section just mapped. The initial portion
!        of this section contains the privileged library vector (PLV).
!
!    ICB_ADDRESS - Address of image control block that describes image.
!
! Implicit Input:
!
!    .ICB [ICB$L_KFE] - Address of known file entry for this image. (A change
!        mode vector must be installed with a PROTECT option. Message
!        vectors do not have to be installed.
!-

BEGIN

BIND
    KERNEL_VECTOR  = CTL$A_DISPVEC,
    EXEC_VECTOR    = CTL$A_DISPVEC + (1*256),
    RUNDOWN_VECTOR = CTL$A_DISPVEC + (2*256),
    MESSAGE_VECTOR = CTL$A_DISPVEC + (3*256);

BIND
    PLV = .SECTION_ADDRESS : REF $BBLOCK,
    ICB = .ICB_ADDRESS     : $BBLOCK,
    KFE = .ICB [ICB$L_KFE] : $BBLOCK;

LITERAL
    VECTOR_SIZE = 256,                  ! One half page for each vector
    ABSOLUTE_MODE = %X'9F',
    AT_R5_MODE = %X'65',
    RSB_ABSOLUTE = (ABSOLUTE_MODE ^ 8) OR OP$_RSB : UNSIGNED (16);

LOCAL
    NEW_VECTOR_LOC;

CASE .PLV [PLV$L_TYPE]
    FROM PLV$C_TYP_CMOD TO PLV$C_TYP_MSG OF
    SET

    [PLV$C_TYP_CMOD]:
```

```
! The section contains a user-written change-mode dispatcher. The beginning
! of the section contains a privileged library vector, laid out as follows.
!
!    .PLV [PLV$L_TYPE]                  ! Vector type code (PLV$C_TYP_CMOD)
!
!    .PLV [PLV$L_VERSION]               ! System version number (SYS$K_VERSION)
!
!    .PLV [PLV$L_KERNEL]                ! Offset to kernel mode dispatcher
!
!    .PLV [PLV$L_EXEC]                  ! Offset to exec mode dispatcher
!
!    .PLV [PLV$L_USRUNDWN]              ! Offset to rundown routine
!
!    .PLV_L_CMOD_RFU                    ! Reserved longword
!
!    .PLV [PLV$L_RMS]                   ! Offset to alternate RMS dispatcher
!
!    .PLV [PLV$L_CHECK]                 ! Address check
!
! If this last longword contains nonzero, its contents must be equal to
! the base address of the section.

BEGIN

BIND
    PLV_L_CMOD_RFU = PLV [PLV$L_USRUNDWN] + 4,
    DISP_VEC = PLV [PLV$L_KERNEL] : VECTOR [3];

! Check that the privileged shareable image is linked against the current
! system and make other sanity checks on the vector contents. The CHECK
! field insures that a position dependent privileged shareable image is
! mapped at the correct address. If the contents of the PLV$L_CHECK are
! not zero, the contents must be equal to the address of the cell.

IF
    (
        (.PLV [PLV$L_VERSION] EQL 0)
        AND
        .ICB [ICB$V_SYS_STB]
    )
    OR
    (
        (.PLV [PLV$L_VERSION] NEQ 0)
        AND
        (.PLV [PLV$L_VERSION] NEQ SYS$K_VERSION)
    )
THEN
    RETURN SS$_BADVEC;

IF
    (.PLV_L_CMOD_RFU NEQ 0)
    OR
    (
        (.PLV [PLV$L_CHECK] NEQ 0)
        AND
        (.PLV [PLV$L_CHECK] NEQ PLV [PLV$L_CHECK])
```

```
4          )
3      THEN
3          RETURN SS$_BADVEC;
3
3      ! A privileged shareable image must be installed (KFE must exist) with
3      ! both the PROTECT and SHARED qualifiers.
3
3      IF
4          BEGIN
5          IF (KFE EQL 0)
4          THEN
4              TRUE
4          ELSE
5              NOT (.KFE [KFE$V_PROTECT] AND .KFE [KFE$V_SHARED])
4          END
3      THEN
3          RETURN SS$_PROTINSTALL;
3
3      ! The same operations are performed for all three of kernel, exec,
3      ! and rundown routines.
3
3      INCR DISP_INDEX FROM 0 TO 2
3          DO
3
3          IF .DISP_VEC [.DISP_INDEX] NEQ 0
3          THEN
4              BEGIN
4
4              BIND
4                  SPECIAL_VECTOR = KERNEL_VECTOR + (VECTOR_SIZE * .DISP_INDEX);
4
4              ! There must be enough room left in the vector to accommodate the
4              ! new JSB instruction (six bytes) and an RSB instruction (four
4              ! bytes to allow room for padding with zeros).
4
5              IF (VECTOR_SIZE - (.SPECIAL_VECTOR) ) LSSU (6 + 4)
4              THEN RETURN SS$_VECFULL;
4
4              ! A special instruction sequence (RSB followed by absolute
4              ! addressing) is inserted at the current end of the vector. After
4              ! the address fixups have been completed, the RSB will be replaced
4              ! with a JSB. The destination of the JSB is computed from the
4              ! contents and address of the current dispatch vector cell.
4
4              NEW_VECTOR_LOC = SPECIAL_VECTOR + .SPECIAL_VECTOR;
4              .NEW_VECTOR_LOC = RSB_ABSOLUTE;
4              (.NEW_VECTOR_LOC) + 2 = DISP_VEC [.DISP_INDEX] + .DISP_VEC [.DISP_INDEX];
4              (.NEW_VECTOR_LOC) + 6 = OP$_RSB;
4
4              ! Store the offset to the new location of the RSB
4
4              SPECIAL_VECTOR = ((.NEW_VECTOR_LOC) + 6) - SPECIAL_VECTOR;
4
4              IAC$GL_IMAGCTX [IMAGCTX$V_SETVECTOR] = TRUE;
4
3              END;
3
```

```
3         ! An alternate RMS dispatcher address is saved in OWN storage. The
3         ! completion routine that executes in kernel mode will store this
3         ! address into the cell used by the change-mode-to-exec handler.
3
3         IF .PLV [PLV$L_RMS] NEQ 0
3         THEN OWN_STORAGE [RMS_BASE] = PLV [PLV$L_RMS] + .PLV [PLV$L_RMS];
3
3         RETURN SS$_NORMAL;                        ! All done
3
2         END;                                      ! End of change mode section
2
2         [PLV$C_TYP_MSG]:
2
2         ! The section contains an image-specific message section, laid out in the
2         ! following way.
2         !
2         !    .PLV [PLV$L_TYPE]                     ! Vector type code (PLV$C_TYP_MSG)
2         !
2         !    .PLV_L_MSG_RFU                        ! Reserved longword (MBZ)
2         !
2         !    .PLV [PLV$L_MSGDSP]                   ! Offset to message dispatcher (6)
2         !
2         ! This offset locates the JSB (R5) instruction.
2         !
2         ! The rest of the message section header contains instructions that are
2         ! interpreted (not executed) by the Get Message system service.
2         !
2         !    NOP                                   ! The next two bytes could serve
2         !    NOP                                   !    as an entry mask
2         !    JSB     (R5)                          ! The offset above locates this
2
3         BEGIN
3
3         BIND
3             PLV_L_MSG_RFU = PLV [PLV$L_VERSION],
3             PLV_L_MSG_ENTRY = PLV [PLV$L_EXEC],
3             PLV_L_MSC_OFFSET = PLV [PLV$L_USRUNDWN],
3             MSC = PLV_L_MSC_OFFSET + .PLV_L_MSC_OFFSET : BYTE ;
3
3         LITERAL
3             JSB_AT_R5 =
3                 (AT_R5_MODE ^ 24) OR
3                 (OP$_JSB ^ 16) OR
3                 (OP$_NOP ^ 8) OR
3                 (OP$_NOP);
3
3         LOCAL
3             PROT_DSC : VECTOR [2];
3
3         ! Verify the contents of the next three longwords in the vector
3
4         IF (
4             (.PLV_L_MSG_RFU NEQ 0) OR
4             (.PLV [PLV$L_MSGDSP] NEQ 6) OR
4             (.PLV_L_MSG_ENTRY NEQ JSB_AT_R5) )
3         THEN
3             RETURN SS$_BADVEC;
```

```
    ! There must be enough room left in the vector to accommodate the new
    ! JSB instruction (six bytes) and an RSB instruction (four bytes to allow
    ! room for padding with zeros).

    IF (VECTOR_SIZE - (.MESSAGE_VECTOR) ) LSSU (6 + 4)
    THEN RETURN SS$_VECFULL;

    ! Find the current end of the vector and store a JSB @# instruction
    ! there, pointing to the JSB (R5) instruction in the message section.

    NEW_VECTOR_LOC = MESSAGE_VECTOR + .MESSAGE_VECTOR;
    .NEW_VECTOR_LOC = RSB_ABSOLUTE;
    (.NEW_VECTOR_LOC) + 2 = PLV_L_MSG_ENTRY + 2;
    (.NEW_VECTOR_LOC) + 6 = OP$_RSB;

    ! Store the offset to the new location of the RSB

    MESSAGE_VECTOR = ((.NEW_VECTOR_LOC) + 6) - MESSAGE_VECTOR;

    IAC$GL_IMAGCTX [IMAGCTX$V_SETVECTOR] = TRUE;

    ! Finally, if this is a pointer (indirect) message section, the section
    ! must be writable from user mode.

    PROT_DSC [0] = ..SECTION_ADDRESS;     ! Only need to change first page
    PROT_DSC [1] = .PROT_DSC [0] + 511;

    IF .MSC EQL MSC$C_IND                 ! Indirect message section?
    THEN
        $SETPRT (INADR = PROT_DSC,        ! Yes, set page protection
                 PROT = PRT$C_UW,
                 ACMODE = PSL$C_EXEC);

    RETURN SS$_NORMAL;

    END;                                  ! End of message section

    [OUTRANGE]:
        RETURN SS$_BADVEC

    TES
END;                                      ! End of routine ADD_PRIVILEGED_VECTOR


                                        .EXTRN  SYS$SETPRT

                      007C 00000 ADD_PRIVILEGED_VECTOR:
                                        .WORD   Save R2,R3,R4,R5,R6                 ; 1165
     56 00000000G    00  9E 00002       MOVAB   MESSAGE_VECTOR, R6                  ;
     5E              08  C2 00009       SUBL2   #8, SP                              ;
     50        08    AC  D0 0000C       MOVL    ICB_ADDRESS, R0                     ; 1203
     53        54    A0  D0 00010       MOVL    84(R0), R3                          ; 1204
     52        04    BC  D0 00014       MOVL    @SECTION_ADDRESS, R2                ; 1215
01   01              62  CF 00018       CASEL   (R2), #1, #1                        ; 1371
```

```
                  00A3          0007     0001C 1$:     .WORD   3$-1$,-                                 ;
                                                               12$-1$                                  ;
                                0088  31 00020 2$:     BRW     13$                                     ; 1430
                    51      04    A2  D0 00023 3$:     MOVL    4(R2), R1                               ; 1257
                                  05  12 00027         BNEQ    4$                                      ;
                            10    A0  95 00029         TSTB    16(R0)                                  ; 1259
                                  F2  19 0002C         BLSS    2$                                      ;
                                  51  D5 0002E 4$:     TSTL    R1                                      ; 1263
                                  09  13 00030         BEQL    5$                                      ;
          00000000G 8F            51  D1 00032         CMPL    R1, #SYS$K_VERSION                      ; 1265
                                  E5  12 00039         BNEQ    2$                                      ;
                            14    A2  D5 0003B 5$:     TSTL    20(R2)                                  ; 1271
                                  E0  12 0003E         BNEQ    2$                                      ;
                            1C    A2  D5 00040         TSTL    28(R2)                                  ; 1274
                                  0A  13 00043         BEQL    6$                                      ;
                    50      1C    A2  9E 00045         MOVAB   28(R2), R0                              ; 1276
                    50      1C    A2  D1 00049         CMPL    28(R2), R0                              ;
                                  D1  12 0004D         BNEQ    2$                                      ;
                                  53  D5 0004F 6$:     TSTL    R3                                      ; 1286
                                  09  13 00051         BEQL    7$                                      ;
                    05      10    A3  E9 00053         BLBC    16(R3), 7$                              ; 1290
      06       10   A3            05  E0 00057         BBS     #5, 16(R3), 8$                          ;
                    50    205C    8F  3C 0005C 7$:     MOVZWL  #8284, R0                               ; 1293
                                      04 00061         RET                                             ;
                                  50  D4 00062 8$:     CLRL    DISP_INDEX                              ; 1298
                    55      08  A240  DE 00064 9$:     MOVAL   8(R2)[DISP_INDEX], R5                   ; 1301
                                  65  D5 00069         TSTL    (R5)                                    ;
                                  3B  13 0006B         BEQL    10$                                     ;
      53            50            08  78 0006D         ASHL    #8, DISP_INDEX, R3                      ; 1306
                    53    FD00  C643  9E 00071         MOVAB   KERNEL_VECTOR[R3], R3                   ;
      54            63 00000100   8F  C3 00077         SUBL3   #256, (R3), R4                          ; 1312
                    54            54  CE 0007F         MNEGL   R4, R4                                  ;
                    0A            54  D1 00082         CMPL    R4, #10                                 ;
                                  6A  1F 00085         BLSSU   15$                                     ;
      51            53            63  C1 00087         ADDL3   (R3), R3, NEW_VECTOR_LOC                ; 1321
                    61    9F05    8F  3C 0008B         MOVZWL  #40709, (NEW_VECTOR_LOC)                ; 1322
 02   A1            55            65  C1 00090         ADDL3   (R5), R5, 2(NEW_VECTOR_LOC)             ; 1323
               06   A1            05  D0 00095         MOVL    #5, 6(NEW_VECTOR_LOC)                   ; 1324
                    54      06    A1  9E 00099         MOVAB   6(NEW_VECTOR_LOC), R4                   ; 1328
      63            54            53  C3 0009D         SUBL3   R3, R4, (R3)                            ;
          00000000G 00            01  88 000A1         BISB2   #1, IAC$GL_IMAGCTX+2                    ; 1330
      B8            50            02  F3 000A8 10$:    AOBLEQ  #2, DISP_INDEX, 9$                      ; 1301
                            18    A2  D5 000AC         TSTL    24(R2)                                  ; 1338
                                  0C  13 000AF         BEQL    11$                                     ;
                    52      18    A2  C0 000B1         ADDL2   24(R2), R2                              ; 1339
          00000000G 00      18    A2  9E 000B5         MOVAB   24(R2), OWN_STORAGE+44                  ;
                                  7D  11 000BD 11$:    BRB     17$                                     ; 1371
                    50      10    A2  9E 000BF 12$:    MOVAB   16(R2), R0                              ; 1370
                    50            60  C0 000C3         ADDL2   (R0), R0                                ; 1371
                            04    A2  D5 000C6         TSTL    4(R2)                                   ; 1386
                                  10  12 000C9         BNEQ    13$                                     ;
                    06      08    A2  D1 000CB         CMPL    8(R2), #6                               ; 1387
                                  0A  12 000CF         BNEQ    13$                                     ;
           65160101 8F      0C    A2  D1 000D1         CMPL    12(R2), #1695940865                     ; 1388
                                  06  13 000D9         BEQL    14$                                     ;
                    50    2064    8F  3C 000DB 13$:    MOVZWL  #8292, R0                               ; 1390
                                      04 000E0         RET                                             ;
```

```
                        53           66 D0 000E1 14$:   MOVL    MESSAGE_VECTOR, R3                     ; 1396
                        51     FF00  C3 9E 000E4        MOVAB   -256(R3), R1                           ;
                        51           51 CE 000E9        MNEGL   R1, R1                                 ;
                        0A           51 D1 000EC        CMPL    R1, #10                                ;
                                     06 1E 000EF        BGEQU   16$                                    ;
                        50     2034  8F 3C 000F1 15$:   MOVZWL  #8244, R0                              ; 1397
                                     04 000F6           RET                                            ;
            51          53           56 C1 000F7 16$:   ADDL3   R6, R3, NEW_VECTOR_LOC                 ; 1402
                        81     9F05  8F 3C 000FB        MOVZWL  #40709, (NEW_VECTOR_LOC)+              ; 1403
                FE      A1       0E  A2 9E 00100        MOVAB   14(R2), -2(NEW_VECTOR_LOC)             ; 1404
                        51           02 C0 00105        ADDL2   #2, R1                                 ; 1405
                        61           05 D0 00108        MOVL    #5, (R1)                               ;
                        52           66 9E 0010B        MOVAB   MESSAGE_VECTOR, R2                     ; 1409
            66          51           52 C3 0010E        SUBL3   R2, R1, MESSAGE_VECTOR                 ;
                00000000G 00         01 88 00112        BISB2   #1, IAC$GL_IMAGCTX+2                   ; 1411
                        6E       04  BC D0 00119        MOVL    @SECTION_ADDRESS, PROT_DSC             ; 1416
    04      AE          6E 000001FF  8F C1 0011D        ADDL3   #511, PROT_DSC, PROT_DSC+4             ; 1417
                        01           60 91 00126        CMPB    (R0), #1                               ; 1419
                                     11 12 00129        BNEQ    17$                                    ;
                        7E           04 7D 0012B        MOVQ    #4, -(SP)                              ; 1423
                                     01 DD 0012E        PUSHL   #1                                     ;
                                     7E D4 00130        CLRL    -(SP)                                  ;
                               10    AE 9F 00132        PUSHAB  PROT_DSC                               ;
                00000000G 00         05 FB 00135        CALLS   #5, SYS$SETPRT                         ;
                        50           01 D0 0013C 17$:   MOVL    #1, R0                                 ; 1425
                                     04 0013F           RET                                            ; 1434

; Routine Size:  320 bytes,    Routine Base:  YF$$SYSIMGACT + 04E6
```

```
%SBTTL 'ADD_FIXUP_VECTOR - Add Fixup Vector to Work List'

ROUTINE ADD_FIXUP_VECTOR (SECTION_ADDRESS, BASE_ADDRESS) =

!+
! Functional Description:
!
!       This routine processes an image section that contains a fixup vector. In
!       addition to some validation of the contents of the fixup vector, two
!       operations are performed.
!
!           The fixup vector is added to the front of the list of fixups that must
!           be done after the image is activated.
!
!           The base address of the image currently being activated is stored in
!           the SHL entry at index 0. Note that the linker did not store a name
!           here so the name field for SHL entry 0 is always left blank.
!
!       If the image being activated represents a privileged shareable image, then
!       this routine is not even called. Rather, a check is made to insure that
!       the image contains no outbound calls. Then the fixups are performed
!       immediately, rather than being postponed for later processing in user
!       mode.
!
! Calling Sequence:
!
!       ADD_FIXUP_VECTOR ()
!
! Formal Parameters:
!
!       SECTION_ADDRESS - Base address of image section that has been identified
!           as a fixup vector.
!
!       BASE_ADDRESS - Base address of image that contains the current fixup
!           vector.
!-

BEGIN

BIND
    RANGE = .SECTION_ADDRESS            : VECTOR,
    IAF   = .RANGE [0]                  : $BBLOCK,
    SHL = IAF + .IAF [IAF$L_SHLSTOFF]   : $BBLOCK;

! Calculate the address of the shareable image list, insure that it lies
! within the bounds of the section just mapped, insure further that the
! caller can write into the SHL, and store the base address of the image that
! contains this SHL. If this SHL does not include a size, store a size that
! reflects the size of an old SHL.

IF (SHL LSSU .RANGE [0]) OR ((SHL+SHL$C_LENGTH) GTRU .RANGE [1])
THEN RETURN IMG$_BAD_FIXUPVEC;

! The following $ASSUME allows the BASEVA longword and SHL_SIZE byte
! to have their writability checked with a single PROBE instruction.

$ASSUME ( $BYTEOFFSET (SHL$L_BASEVA) EQL 0);
```

```
2 $ASSUME ( $BYTEOFFSET (SHL$B_SHL_SIZE) LEQU 512);
2
2 IF NOT PROBEW
2     (OWN_STORAGE [ACCESS_MODE],
2      %REF ($BYTEOFFSET (SHL$B_SHL_SIZE) ),
2      SHL [SHL$L_BASEVA])
2 THEN RETURN SS$_ACCVIO;
2
2 SHL [SHL$L_BASEVA] = .BASE_ADDRESS;
2 IF .SHL [SHL$B_SHL_SIZE] EQL 0
2 THEN SHL [SHL$B_SHL_SIZE] = SHL$C_OLD_SHL_SIZE;
2
2 ! Insure that the caller can write into the link longword in the fixup
2 ! vector and then add this fixup vector to the front of the list.
2
2 IF NOT PROBEW (OWN_STORAGE [ACCESS_MODE], %REF (4), IAF [IAF$L_FIXUPLNK])
2 THEN RETURN SS$_ACCVIO;
2
2 IAF [IAF$L_FIXUPLNK] = .CTL$GL_FIXUPLNK;
2 CTL$GL_FIXUPLNK = IAF;
2 RETURN SS$_NORMAL
2
1 END;                                        ! End of routine ADD_FIXUP_VECTOR
```

```
                                        003C 00000 ADD_FIXUP_VECTOR:
                                                          .WORD   Save R2,R3,R4,R5                    ; 1437
                 55 00000000G    00  9E 00002             MOVAB   OWN_STORAGE+100, R5                 ;
                 54 00000000G    00  9E 00009             MOVAB   CTL$GL_FIXUPLNK, R4                 ;
                 51        04    AC  D0 00010             MOVL    SECTION_ADDRESS, R1                 ; 1475
                 52              61  D0 00014             MOVL    (R1), R2                            ; 1476
   50            52        18    A2  C1 00017             ADDL3   24(R2), R2, R0                      ; 1477
                 61              50  D1 0001C             CMPL    R0, (R1)                            ; 1485
                                 0A  1F 0001F             BLSSU   1$                                  ;
                 53        40    A0  9E 00021             MOVAB   64(R0), R3                          ;
           04    A1              53  D1 00025             CMPL    R3, 4(R1)                           ;
                                 08  1B 00029             BLEQU   2$                                  ;
                 50 084D8C8C     8F  D0 0002B 1$:         MOVL    #139299980, R0                      ; 1486
                                     04 00032             RET                                         ;
   60            10              65  0D 00033 2$:         PROBEW  OWN_STORAGE+100, #16, (R0)          ; 1497
                                 14  13 00037             BEQL    4$                                  ;
                 60        08    AC  D0 00039             MOVL    BASE_ADDRESS, (R0)                  ; 1500
                           10    A0  95 0003D             TSTB    16(R0)                              ; 1501
                                 04  12 00040             BNEQ    3$                                  ;
           10    A0              38  90 00042             MOVB    #56, 16(R0)                         ; 1502
   04      A2    04              65  0D 00046 3$:         PROBEW  OWN_STORAGE+100, #4, 4(R2)          ; 1507
                                 04  12 0004B             BNEQ    5$                                  ;
                 50              0C  D0 0004D 4$:         MOVL    #12, R0                             ; 1508
                                     04 00050             RET                                         ;
           04    A2              64  D0 00051 5$:         MOVL    CTL$GL_FIXUPLNK, 4(R2)              ; 1510
                 64              52  D0 00055             MOVL    R2, CTL$GL_FIXUPLNK                 ; 1511
                 50              01  D0 00058             MOVL    #1, R0                              ; 1512
                                     04 0005B             RET                                         ; 1514
```

; Routine Size: 92 bytes, Routine Base: YF$$SYSIMGACT + 0626

```
1 %SBTTL 'NEXT_GBL_SEC_NAME - Update Global Section Name Suffix'
1
1 ROUTINE NEXT_GBL_SEC_NAME (STRING_POINTER) : NOVALUE =
1
1 !+
1 ! Functional Description:
1 !
1 !    This routine performs string arithmetic on a global section name suffix of
1 !    the form _00n by adding one to the suffix and performing the appropriate
1 !    carries. No error checking for illegal string contents or carries beyond
1 !    _999 is performed. (The suffix _999 is transformed to _000.)
1 !
1 ! Calling Sequence:
1 !
1 !    NEXT_GBL_SEC_NAME (STRING_POINTER)
1 !
1 ! Formal Parameters:
1 !
1 !    STRING_POINTER - Address of counted ASCII string containing global
1 !        section name
1 !-
1
2 BEGIN
2
2 ! Two synonyms are created. The string called NAME represents the entire
2 ! counted ASCII string for the global section name. Recall that the count field
2 ! does not contain the four character suffix. The substring SUFFIX represents
2 ! the final three characters in the global section name, usually 00n.
2
2 BIND
2     NAME   = .STRING_POINTER               : VECTOR [,BYTE],
2     SUFFIX = NAME [.NAME [0] + 2]          : VECTOR [3,BYTE];
2
2 LOCAL
2     DIGIT : INITIAL (2),
2     CARRY : INITIAL (TRUE);
2
2 WHILE .CARRY DO
2     IF .SUFFIX [.DIGIT] NEQ %C'9'
2     THEN
3         BEGIN
3         SUFFIX [.DIGIT] = .SUFFIX [.DIGIT] + 1;
3         CARRY = FALSE;
3         END
2     ELSE
3         BEGIN
3         SUFFIX [.DIGIT] = %C'0';
3         DIGIT = .DIGIT - 1;
3         IF .DIGIT LSS 0
3         THEN CARRY = FALSE;
2         END;
2
1 END;                                       ! End of routine NEXT_GBL_SEC_NAME
```

```
                                   0004 00000 NEXT_GBL_SEC_NAME:
                                                     .WORD   Save R2                              ; 1517
      50          04   BC   9A 00002                 MOVZBL  @STRING_POINTER, R0                  ; 1546
      50          04   AC   C0 00006                 ADDL2   STRING_POINTER, R0                   ;
      50               02   C0 0000A                 ADDL2   #2, R0                               ;
      51               02   D0 0000D                 MOVL    #2, DIGIT                            ;
      52               01   D0 00010                 MOVL    #1, CARRY                            ;
      16               52   E9 00013 1$:             BLBC    CARRY, 4$                            ; 1552
      39             6140   91 00016                 CMPB    (DIGIT)[R0], #57                     ; 1553
                       05   13 0001A                 BEQL    2$                                   ;
                     6140   96 0001C                 INCB    (DIGIT)[R0]                          ; 1556
                       07   11 0001F                 BRB     3$                                   ; 1557
    6140               30   90 00021 2$:             MOVB    #48, (DIGIT)[R0]                     ; 1561
      EB               51   F4 00025                 SOBGEQ  DIGIT, 1$                            ; 1562
                       52   D4 00028 3$:             CLRL    CARRY                                ; 1564
                       E7   11 0002A                 BRB     1$                                   ; 1553
                            04 0002C 4$:             RET                                          ; 1567

; Routine Size:  45 bytes,    Routine Base:  YF$$SYSIMGACT + 0682
```

```
  1 %SBTTL 'LOAD_SEQ_IMAGE - Load an image from a sequential device'
  1
  1 ROUTINE LOAD_SEQ_IMAGE ( ICB : REF $BBLOCK ) =
  1
  1 !+
  1 ! Functional Description:
  1 !
  1 ! This routine loads all private non-dzro sections of an image from a
  1 ! sequential device.
  1 !
  1 ! Calling Sequence:
  1 !
  1 !    LOAD_SEQ_IMAGE (ICB pointer)
  1 !
  1 ! Formal Parameters:
  1 !
  1 !        ICB - pointer to the current ICB
  1 !-
  1
  2 BEGIN
  2
  2 ! This routine expects a vector of image section entries.  Each entry consists
  2 ! of two longwords.  The first is a page count.  The second is a starting
  2 ! address with the low order bit used to indicate writeable address space.
  2 ! The read is done one page at a time under the assumption of one page records.
  2 !
  2 ! This routine also zeros the debug symbol table entry in the image header to
  2 ! prevent an attempt by DEBUG to read this table.
  2
  2 LOCAL
  2     SEQ_ISDS      : REF VECTOR [],
  2     INADR         : VECTOR [2],
  2     IHD           : REF $BBLOCK,
  2     IOSB          : VECTOR [4,WORD],
  2     NEXT_ADDRESS,
  2     STATUS;
  2
  2 BIND
  2     IHD_CTX = .ICB [ICB$L_CONTEXT] : $BBLOCK;
  2
  2 SEQ_ISDS = .OWN_STORAGE [SEQ_LOAD_ISDS];
  2
  2 IHD = .IHD_CTX [CTX_L_IHDBUF];
  2 IHD [IHD$W_SYMDBGOFF] = 0;
  2
  2 ! Process ISDs until done
  2
  2 UNTIL .SEQ_ISDS [0] EQL 0
  2 DO
  3     BEGIN
  3     INADR [0] = .SEQ_ISDS [1];
  3     INADR [1] = .INADR [0] + (.SEQ_ISDS [0] * BYTES_PER_PAGE) - 1;
  3
  3     ! Create adrress space
  3
P 3     STATUS = $CRETVA (
P 3         INADR = INADR,
```

```
                ACMODE = PSL$C_USER );
        IF NOT .STATUS THEN RETURN .STATUS;

        ! Read the image one page at a t4me

        NEXT_ADDRESS = .INADR [0];
        INCR I FROM 1 TO .SEQ_ISDS [0]
        DO
            BEGIN
            STATUS = $QIOW (
                EFN = EXE$C_SYSEFN,
                CHAN = .ICB [ICB$W_CHAN],
                FUNC = IO$_READVBLK,
                IOSB = IOSB,
                P1 = .NEXT_ADDRESS,
                P2 = BYTES_PER_PAGE );
            IF NOT .STATUS THEN RETURN .STATUS;
            IF NOT .IOSB [0] THEN RETURN .IOSB [0];
            NEXT_ADDRESS = .NEXT_ADDRESS + BYTES_PER_PAGE;
            END;

        ! Is it a message section (can't be vector since not installed)

        IF (.SEQ_ISDS [2] AND ISD$M_VECTOR) NEQ 0
        THEN
            BEGIN
            STATUS = ADD_PRIVILEGED_VECTOR (INADR, ICB);
            IF NOT .STATUS THEN RETURN .STATUS
            END;

        ! Is it a fixup section

        IF (.SEQ_ISDS [2] AND ISD$M_FIXUPVEC) NEQ 0
        THEN
            BEGIN
            STATUS = ADD_FIXUP_VECTOR (
                INADR,
                .ICB [ICB$L_BASE_ADDRESS]);
            IF NOT .STATUS THEN RETURN .STATUS
            END;

        ! Check to see if it should be left writeable

        IF (.SEQ_ISDS [2] AND ISD$M_WRT) EQL 0
        THEN
            BEGIN
            STATUS = $SETPRT (
                INADR = INADR,
                ACMODE = PSL$C_USER,
                PROT = PRT$C_UR );
            IF NOT .STATUS THEN RETURN .STATUS;
            END;

        ! Advance pointer to next ISD description

        SEQ_ISDS = SEQ_ISDS [3];
```

```
; 1462          1682  2        END;
; 1463          1683  2
; 1464          1684  2  RETURN SS$_NORMAL;
; 1465          1685  2
; 1466          1686  1  END;


                                                                .EXTRN  SYS$QIOW

                                         003C 00000 LOAD_SEQ_IMAGE:
                                                                .WORD   Save R2,R3,R4,R5                              ; 1570
                    5E           10    C2 00002                SUBL2   #16, SP
                    50     04    AC    D0 00005                MOVL    ICB, R0                                       ; 1606
                    50     58    A0    D0 00009                MOVL    88(R0), R0
                    52 00000000G 00    D0 0000D                MOVL    OWN_STORAGE+40, SEQ_ISDS                      ; 1608
                    50     04    A0    D0 00014                MOVL    4(R0), IHD                                    ; 1610
                           04    A0    B4 00018                CLRW    4(IHD)                                        ; 1611
                                 62    D5 0001B 1$:            TSTL    (SEQ_ISDS)                                    ; 1615
                                 03    12 0001D                BNEQ    2$
                               00AB    31 0001F                BRW     10$
            08      AE     04    A2    D0 00022 2$:            MOVL    4(SEQ_ISDS), INADR                            ; 1618
       51           62           09    78 00027                ASHL    #9, (SEQ_ISDS), R1                            ; 1619
                    51     08    AE    C0 0002B                ADDL2   INADR, R1
            0C      AE     FF    A1    9E 0002F                MOVAB   -1(R1), INADR+4
                                 03    DD 00034                PUSHL   #3                                            ; 1625
                                 7E    D4 00036                CLRL    -(SP)
                           10    AE    9F 00038                PUSHAB  INADR
         00000000G  00           03    FB 0003B                CALLS   #3, SYS$CRETVA
                    7F           50    E9 00042                BLBC    STATUS, 8$                                    ; 1626
                    55     08    AE    D0 00045                MOVL    INADR, NEXT_ADDRESS                           ; 1630
                    53     04    AC    D0 00049                MOVL    ICB, R3                                       ; 1640
                                 54    D4 0004D                CLRL    I
                                 2F    11 0004F                BRB     5$
                                 7E    7C 00051 3$:            CLRQ    -(SP)
                                 7E    7C 00053                CLRQ    -(SP)
                    7E   0200    8F    3C 00055                MOVZWL  #512, -(SP)
                                 55    DD 0005A                PUSHL   NEXT_ADDRESS
                                 7E    7C 0005C                CLRQ    -(SP)
                           20    AE    9F 0005E                PUSHAB  IOSB
                                 31    DD 00061                PUSHL   #49
                    7E     0E    A3    3C 00063                MOVZWL  14(R3), -(SP)
                    7E          00G    9A 00067                MOVZBL  S^EXE$C_SYSEFN, -(SP)
         00000000G  00           0C    FB 0006A                CALLS   #12, SYS$QIOW
                    5C           50    E9 00071                BLBC    STATUS, 11$                                   ; 1641
                    04           6E    E8 00074                BLBS    IOSB, 4$                                      ; 1642
                    50           6E    3C 00077                MOVZWL  IOSB, R0
                                       04 0007A                RET
                    55   0200    C5    9E 0007B 4$:            MOVAB   512(R5), NEXT_ADDRESS                         ; 1643
       CD           54           62    F3 00080 5$:            AOBLEQ  (SEQ_ISDS), I, 3$                             ; 1631
       0E    08     A2           11    E1 00084                BBC     #17, 8(SEQ_ISDS), 6$                          ; 1648
                           04    AC    9F 00089                PUSHAB  ICB                                           ; 1651
                           0C    AE    9F 0008C                PUSHAB  INADR
              FDA3  CF           02    FB 0008F                CALLS   #2, ADD_PRIVILEGED_VECTOR
                    39           50    E9 00094                BLBC    STATUS, 11$                                   ; 1652
       12    08     A2           0A    E1 00097 6$:            BBC     #10, 8(SEQ_ISDS), 7$                          ; 1657
                    51     04    AC    D0 0009C                MOVL    ICB, R1                                       ; 1662
```

```
                                 5C  A1  DD 000A0        PUSHL   92(R1)                          ;
                                 0C  AE  9F 000A3        PUSHAB  INADR                           ; 1660
              FECC   CF              02  FB 000A6        CALLS   #2, ADD_FIXUP_VECTOR            ;
                     22              50  E9 000AB        BLBC    STATUS, 11$                     ; 1663
       14      08    A2              03  E0 000AE  7$:   BBS     #3, 8(SEQ_ISDS), 9$             ; 1668
                     7E              0F  7D 000B3        MOVQ    #15, -(SP)                      ; 1674
                                     03  DD 000B6        PUSHL   #3                              ;
                                     7E  D4 000B8        CLRL    -(SP)                           ;
                                 18  AE  9F 000BA        PUSHAB  INADR                           ;
         00000000G   00              05  FB 000BD        CALLS   #5, SYS$SETPRT                  ;
                     09              50  E9 000C4  8$:   BLBC    STATUS, 11$                     ; 1675
                     52              0C  C0 000C7  9$:   ADDL2   #12, SEQ_ISDS                   ; 1680
                               FF4E  31  000CA           BRW     1$                              ; 1615
                     50              01  D0 000CD  10$:  MOVL    #1, R0                          ; 1684
                                     04  000D0     11$:  RET                                     ; 1686

; Routine Size:  209 bytes,    Routine Base:  YF$$SYSIMGACT + 06AF
```

INI
V04

2D
65
64

```
; 1468         1687  1 END                                  ! End of module IMGACT_MAP_ISDS
; 1469         1688  1
; 1470         1689  0 ELUDOM
```

```
;                                   PSECT SUMMARY
;
;       Name                      Bytes                       Attributes
;
;  YF$$SYSIMGACT                     1920  NOVEC,  WRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
```

```
;                            Library Statistics
;
;                                          -------- Symbols --------         Pages        Processing
;       File                               Total     Loaded   Percent        Mapped       Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1           18619        90         0         1000          00:01.8
```

```
;                                   COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:IMGMAPISD/OBJ=OBJ$:IMGMAPISD MSRC$:IMGMAPISD/UPDATE=(ENH$:IMGMAPISD)

; Size:            1896 code + 24 data bytes
; Run Time:           00:41.2
; Elapsed Time:       00:46.9
; Lines/CPU Min:        2461
; Lexemes/CPU-Min: 20722
; Memory Used:  325 pages
; Compilation Complete
```

FORKCNTRL
LIS

FILINIWCB
LIS

IMGDECODE
LIS

INIT
LIS

IOLOCK
LIS

IODEF
LIS

IOCIOPOST
LIS

GLOBALS
LIS

IMGMAPISD
LIS