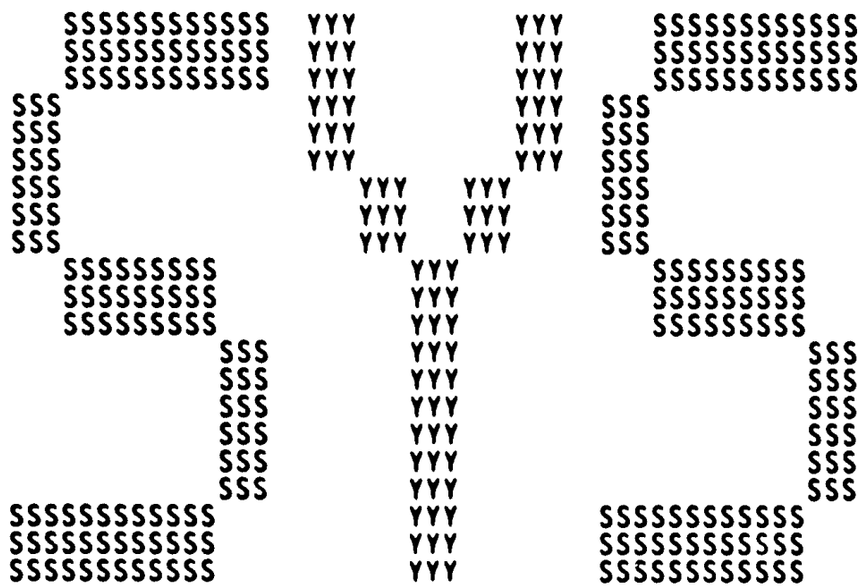

FILEID**IMGMAPISD

```
 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
```

IMGS
V04-

```
;  1     0001  0 %TITLE 'MAP_ISDS - Convert ISDs to Mapping Requests'
;  2     0002  0 MODULE IMG$MAP_ISDS (
;  3     0003  0                      IDENT = 'V04-001'                ! File: SRC$:IMGMAPISD.B32
;  4     0004  0                      ) =
;  5     0005  1 BEGIN
;  6     0006  1 !
;  7     0007  1 !****************************************************************************
;  8     0008  1 !*                                                                          *
;  9     0009  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
; 10     0010  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
; 11     0011  1 !*  ALL RIGHTS RESERVED.                                                    *
; 12     0012  1 !*                                                                          *
; 13     0013  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
; 14     0014  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
; 15     0015  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
; 16     0016  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
; 17     0017  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
; 18     0018  1 !*  TRANSFERRED.                                                            *
; 19     0019  1 !*                                                                          *
; 20     0020  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
; 21     0021  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
; 22     0022  1 !*  CORPORATION.                                                            *
; 23     0023  1 !*                                                                          *
; 24     0024  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
; 25     0025  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
; 26     0026  1 !*                                                                          *
; 27     0027  1 !*                                                                          *
; 28     0028  1 !****************************************************************************
; 29     0029  1 !
; 30     0030  1
; 31     0031  1 !++
; 32     0032  1 ! Facility:
; 33     0033  1 !
; 34     0034  1 !       Executive, Support Routines Used by Image Activator System Service
; 35     0035  1 !
; 36     0036  1 ! Abstract:
; 37     0037  1 !
; 38     0038  1 !       This module contains the routines that transform each image section
; 39     0039  1 !       descriptor into a mapping request for the process address space.
; 40     0040  1 !
; 41     0041  1 ! Environment:
; 42     0042  1 !
; 43     0043  1 !       The code in this module executes in executive mode.
; 44     0044  1 !
; 45     0045  1 ! Author:
; 46     0046  1 !
; 47     0047  1 !       Lawrence J. Kenah
; 48     0048  1 !
; 49     0049  1 ! Creation Date:
; 50     0050  1 !
; 51     0051  1 !       6 May 1983
; 52     0052  1 !
; 53     0053  1 ! Modified By:
; 54     0054  1 !
; 55     0055  1 !       V04-001 MSH0074         Michael S. Harvey       6-Sep-1984
; 56     0056  1 !                               Don't open image file for write for ISDs that are
; 57     0057  1 !                               global, writeable, and demand-zero.
```

```
;    58          0058  1 !
;    59          0059  1 !       V03B-017  MSH0045        Michael S. Harvey        10-May-1984
;    60          0060  1 !                 Correctly set page protection for indirect message
;    61          0061  1 !                 sections.
;    62          0062  1 !
;    63          0063  1 !       V03B-016  MSH0044        Michael S. Harvey        9-May-1984
;    64          0064  1 !                 Correct page protection for protected writable CRF
;    65          0065  1 !                 sections.
;    66          0066  1 !
;    67          0067  1 !       V03B-015  LJK0281        Lawrence J. Kenah        9-May-1984
;    68          0068  1 !                 Still more cleanup. Make sure that all unused ICBs are
;    69          0069  1 !                 deallocated when an error occurs in the middle of a
;    70          0070  1 !                 complicated activation.
;    71          0071  1 !
;    72          0072  1 !       V03B-014  LJK0269        Lawrence J. Kenah        31-Mar-1984
;    73          0073  1 !                 Miscellaneous cleanup.
;    74          0074  1 !                 Do not perform fixups or load message or change mode vectors
;    75          0075  1 !                 if this is the P0 part of a P1 merge activation.
;    76          0076  1 !                 Use flag in own storage to detect that situation.
;    77          0077  1 !                 Pages in privileged shareable image should be owned by
;    78          0078  1 !                 exec mode.
;    79          0079  1 !                 Add privileged vectors in exec mode instead of kernel mode.
;    80          0080  1 !                 Store alternate RMS address as part of kernel mode completion.
;    81          0081  1 !
;    82          0082  1 !       V03B-013  LJK0267        Lawrence J. Kenah        28-Mar-1984
;    83          0083  1 !                 Do not use name in ICB as global section name. Instead, use
;    84          0084  1 !                 name in CTX storage that was loaded from the KFE.
;    85          0085  1 !
;    86          0086  1 !       V03B-012  LJK0265        Lawrence J. Kenah        25-Mar-1984
;    87          0087  1 !                 Add support for variable size SHL elements.
;    88          0088  1 !
;    89          0089  1 !       V03B-011  JWT0152        Jim Teague               20-Feb-1984
;    90          0090  1 !                 Enlarge ISD_BUFFER to accomodate longer global ISDs.
;    91          0091  1 !
;    92          0092  1 !       V03B-010  WMC0001        Wayne Cardoza            23-Jan-1984
;    93          0093  1 !                 Add loading of sequential image files.
;    94          0094  1 !                 Fix bug in based shareable images.
;    95          0095  1 !
;    96          0096  1 !       V03B-009  LJK0245        Lawrence J. Kenah        23-Aug-1983
;    97          0097  1 !                 Make user stack size a cumulative number.
;    98          0098  1 !
;    99          0099  1 !       V03B-008  LJK0242        Lawrence J. Kenah        2-Aug-1983
;   100          0100  1 !                 Add support for writable global sections.
;   101          0101  1 !
;   102          0102  1 !       V03B-007  LJK0236        Lawrence J. Kenah        26-Jul-1983
;   103          0103  1 !                 Add concept of image base address, different from first address
;   104          0104  1 !                 that is mapped. Continue fixing bugs as they are found.
;   105          0105  1 !
;   106          0106  1 !       V03B-006  LJK0233        Lawrence J. Kenah        21-Jul-1983
;   107          0107  1 !                 Try again to make the logic for P1 merges into P0 space
;   108          0108  1 !                 to work correctly.
;   109          0109  1 !
;   110          0110  1 !       V03B-005  LJK0231        Lawrence J. Kenah        19-Jul-1983
;   111          0111  1 !                 Make corrections for global section mapping
;   112          0112  1 !
;   113          0113  1 !       V03B-004  LJK0227        Lawrence J. Kenah        12-Jul-1983
;   114          0114  1 !                 Continue with cleanup efforts and testing.
```

```
;  115          0115  1 !
;  116          0116  1 !       V03B-003  LJK0220         Lawrence J. Kenah       29-Jun-1983
;  117          0117  1 !                 Handle case where input range is specified. Perform minor
;  118          0118  1 !                 cleanup. Fix the many bugs discovered during testing.
;  119          0119  1 !
;  120          0120  1 !       V03B-002  LJK0216         Lawrence J. Kenah       25-Jun-1983
;  121          0121  1 !                 Make new IHD generation constant conform to standard names
;  122          0122  1 !
;  123          0123  1 !       V03B-001  LJK0200         Lawrence J. Kenah       6-May-1983
;  124          0124  1 !                 The original version of this module is a portion of a
;  125          0125  1 !                 rewrite of the image activator.
;  126          0126  1 !--
;  127          0127  1
```

```
;  129          0128  1 %SBTTL 'Declarations'
;  130          0129  1
;  131          0130  1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;  132          0131  1
;  133          0132  1 PSECT
;  134          0133  1         CODE    = YF$$SYSIMGACT (WRITE),
;  135          0134  1         PLIT    = YF$$SYSIMGACT (WRITE, EXECUTE);
;  136          0135  1
;  137          0136  1 LIBRARY 'SYS$LIBRARY:LIB.L32';          ! Define system data structures
;  138          0137  1
;  139          0138  1 REQUIRE 'LIB$:IMGMSGDEF.R32';           ! Get status code definitions
;  140          0224  1 REQUIRE 'LIB$:IMGACTCTX.R32';           ! Define internal structures
;  141          0371  1
;  142          0372  1 ! Machine dependent features
;  143          0373  1
;  144          0374  1 BUILTIN
;  145          0375  1     PROBEW,
;  146          0376  1     INSQUE,
;  147          0377  1     REMQUE;
;  148          0378  1
;  149          0379  1 ! Internal references
;  150          0380  1
;  151          0381  1 FORWARD ROUTINE
;  152          0382  1     PROCESS_ISD_LIST,
;  153          0383  1     ADD_PRIVILEGED_VECTOR,
;  154          0384  1     ADD_FIXUP_VECTOR,
;  155          0385  1     NEXT_GBL_SEC_NAME           : NOVALUE,
;  156          0386  1     LOAD_SEQ_IMAGE;
;  157          0387  1
;  158          0388  1 ! Linkage declaration for JSB routines
;  159          0389  1
;  160          0390  1 LINKAGE
;  161          0391  1     EXE_ALOP1PROC     = JSB (REGISTER = 1; REGISTER = 1,REGISTER = 2) :
;  162          0392  1                         NOPRESERVE (3)
;  163          0393  1                         NOTUSED (4,5,6,7,8,9,10,11),
;  164          0394  1     EXE_DEAP1         = JSB (REGISTER = 0, REGISTER = 1) :
;  165          0395  1                         NOPRESERVE (2,3)
;  166          0396  1                         NOTUSED (4,5,6,7,8,9,10,11),
;  167          0397  1     IMG_IS_IT_MAPPED  = JSB (REGISTER = 0; REGISTER = 1) :
;  168          0398  1                         PRESERVE (2,3,4,5,6,7)
;  169          0399  1                         NOTUSED (8,9,10,11),
;  170          0400  1     IMG_PRVSHRIMG     = JSB (REGISTER = 0, REGISTER = 1) :
;  171          0401  1                         PRESERVE (2,3,4,5)
;  172          0402  1                         NOTUSED (6,7,8,9,10,11);
;  173          0403  1
;  174          0404  1 ! External routines with special linkages
;  175          0405  1
;  176          0406  1 EXTERNAL ROUTINE
;  177          0407  1     EXE$ALOP1PROC     : EXE_ALOP1PROC,
;  178          0408  1     EXE$DEAP1         : EXE_DEAP1,
;  179          0409  1     IMG$IS_IT_MAPPED  : IMG_IS_IT_MAPPED,
;  180          0410  1     IMG$PRVSHRIMG     : IMG_PRVSHRIMG;
;  181          0411  1
;  182          0412  1 ! Constants to make life simple
;  183          0413  1
;  184          0414  1 LITERAL
;  185          0415  1     TRUE = 1,
```

```
;  186          0416  1     FALSE = 0,
;  187          0417  1     BYTES_PER_PAGE       = 512,
;  188          0418  1     END_OF_P1_SPACE      = %X'7FFFFFFF',
;  189          0419  1     ISD_M_DISPATCH_FLAGS = (ISD$M_GBL OR ISD$M_CRF OR ISD$M_DZRO OR ISD$M_WRT);
;  190          0420  1
;  191          0421  1 !  External routine references
;  192          0422  1
;  193          0423  1 EXTERNAL ROUTINE
;  194          0424  1     IMG$OPEN_IMAGE,
;  195          0425  1     IMG$GET_HEADER,
;  196          0426  1     IMG$ALLOCATE_ICB,
;  197          0427  1     IMG$DEALLOCATE_ICB,
;  198          0428  1     IMG$GET_NEXT_ISD;
;  199          0429  1
;  200          0430  1 ! References to external data cells
;  201          0431  1
;  202          0432  1 EXTERNAL
;  203          0433  1     CTL$A_DISPVEC,
;  204          0434  1     CTL$GL_FIXUPLNK,
;  205          0435  1     CTL$GL_PHD,
;  206          0436  1     IAC$GL_IMAGCTX              : $BBLOCK,
;  207          0437  1     IAC$AL_IMGACTBUF,
;  208          0438  1     IAC$GL_WORK_LIST            : VECTOR [2],
;  209          0439  1     IAC$GL_IMAGE_LIST           : VECTOR [2];
;  210          0440  1
;  211          0441  1 ! Miscellaneous constants defined elsewhere
;  212          0442  1
;  213          0443  1 EXTERNAL LITERAL
;  214          0444  1     EXE$C_SYSEFN        : UNSIGNED (6),
;  215          0445  1     SYS$K_VERSION;
;  216          0446  1
;  217          0447  1 ! Some miscellaneous address definitions
;  218          0448  1
;  219          0449  1 ! The first part of the image activator scratch area is divided up into two
;  220          0450  1 ! large pieces, each of which is further subdivided into a FAB, a NAM block,
;  221          0451  1 ! a 512-byte block into which each succeeding block of the image header will
;  222          0452  1 ! be read, and a buffer that will receive the decoded image header. The
;  223          0453  1 ! decoded image header is assumed to be smaller than a page. The area that
;  224          0454  1 ! follows these buffers is used as OWN storage by the image activator.
;  225          0455  1
;  226          0456  1 LITERAL
;  227          0457  1     INPUT_BUFFER_SIZE = BYTES_PER_PAGE,
;  228          0458  1     IHD_BUFFER_SIZE   = BYTES_PER_PAGE;
;  229          0459  1
;  230          0460  1 BIND
;  231          0461  1     INPUT_BUFFER = IAC$AL_IMGACTBUF,
;  232          0462  1     PRIMARY_IHD  = INPUT_BUFFER + 512,
;  233          0463  1     AUX_BUFFER   = PRIMARY_IHD  + 512,
;  234          0464  1     AUX_IHD      = AUX_BUFFER   + 512,
;  235          0465  1     PRIMARY_FAB  = AUX_IHD      + 512,
;  236          0466  1     PRIMARY_NAM  = PRIMARY_FAB  + FAB$K_BLN,
;  237          0467  1     AUX_FAB      = PRIMARY_NAM  + NAM$K_BLN,
;  238          0468  1     AUX_NAM      = AUX_FAB      + FAB$K_BLN,
;  239          0469  1     RESULT_NAME  = AUX_NAM      + NAM$K_BLN,
;  240          0470  1     OWN_STORAGE  = RESULT_NAME  + NAM$C_MAXRSS  : $BBLOCK;
;  241          0471  1
;  242          0472  1 ! There are eight pages set aside in P1 space (in module SHELL) for
```

```
;  243          0473  1 ! the image activator scratch area. The following assumption guarantees
;  244          0474  1 ! that the scratch area that is defined here fits into eight pages.
;  245          0475  1
;  246        P 0476  1     $ASSUME ( OWN_STORAGE_SIZE + (2 * (INPUT_BUFFER_SIZE +
;  247        P 0477  1                                        IMD_BUFFER_SIZE +
;  248        P 0478  1                                        FAB$K_BLN +
;  249        P 0479  1                                        NAM$K_BLN) ),
;  250        P 0480  1                 LEQU,
;  251          0481  1                 8 * BYTES_PER_PAGE );
```

```
;  253          0482  1 %SBTTL 'PROCESS_WORK_LIST - Process Each Work List Item'
;  254          0483  1
;  255          0484  1 !++
;  256          0485  1 ! FUNCTIONAL DESCRIPTION:
;  257          0486  1 !
;  258          0487  1 !       This routine removes each work list item in turn from the work list,
;  259          0488  1 !       opens the file if it is not already opened, and calls the inner routine
;  260          0489  1 !       that processes the ISD list for this image.
;  261          0490  1 !
;  262          0491  1 ! CALLING SEQUENCE:
;  263          0492  1 !
;  264          0493  1 !       IMG$DO_WORK_LIST ()
;  265          0494  1 !
;  266          0495  1 ! FORMAL PARAMETERS:
;  267          0496  1 !
;  268          0497  1 !       TBS
;  269          0498  1 !
;  270          0499  1 ! STATUS CODES:
;  271          0500  1 !
;  272          0501  1 !       TBS
;  273          0502  1 !--
;  274          0503  1
;  275          0504  1 GLOBAL ROUTINE IMG$DO_WORK_LIST =
;  276          0505  1
;  277          0506  2 BEGIN
;  278          0507  2
;  279          0508  2 LOCAL
;  280          0509  2     ICB         : REF $BBLOCK,
;  281          0510  2     OLD_ICB     : REF $BBLOCK,
;  282          0511  2     NAME_DESC   : $BBLOCK [DSC$K_S_BLN],
;  283          0512  2     IHD_CTX     : $BBLOCK [CTX_K_LENGTH],
;  284          0513  2     NAME_STRING : VECTOR [NAM$C_MAXRSS],
;  285          0514  2     SIZE,
;  286          0515  2     STATUS;
;  287          0516  2
;  288          0517  2 WHILE NOT REMQUE (.IAC$GL_WORK_LIST , ICB) DO
;  289          0518  3     BEGIN
;  290          0519  3
;  291          0520  3     STATUS = IMG$IS_IT_MAPPED (ICB [ICB$T_IMAGE_NAME]; OLD_ICB);
;  292          0521  3     IF .STATUS EQL SS$_NORMAL
;  293          0522  3     THEN
;  294          0523  3         IMG$DEALLOCATE_ICB (.ICB)
;  295          0524  3     ELSE
;  296          0525  4         BEGIN
;  297          0526  4
;  298          0527  4         ! The ICB is immediately placed into the done list. If an error occurs
;  299          0528  4         ! later in activation, the error cleanup routine will remove it,
;  300          0529  4         ! deassign the channel, and deallocate the ICB.
;  301          0530  4
;  302          0531  5         IF NOT (.OWN_STORAGE [P1_MERGE_PO])
;  303          0532  4         THEN
;  304          0533  4             INSQUE (.ICB , .IAC$GL_IMAGE_LIST [1]);     ! Insert at tail of list
;  305          0534  4
;  306          0535  4         ! A nonzero channel number implies that the ICB represents the image
;  307          0536  4         ! whose name was passed to $IMGACT. This image file was already opened
;  308          0537  4         ! by the caller and its header read and verified.
;  309          0538  4
```

```
;  310     0539  4          IF .ICB [ICB$W_CHAN] EQL 0
;  311     0540  4          THEN
;  312     0541  5              BEGIN
;  313     0542  5
;  314     0543  5              BIND
;  315     0544  5                  ICB_NAME = ICB [ICB$T_IMAGE_NAME]       : VECTOR [, BYTE];
;  316     0545  5
;  317     0546  5              ICB [ICB$L_CONTEXT] = IHD_CTX;
;  318     0547  5              NAME_DESC [DSC$W_LENGTH] = .ICB_NAME [0];
;  319     0548  5              NAME_DESC [DSC$A_POINTER] = ICB_NAME [1];
;  320     0549  5              STATUS = IMG$OPEN_IMAGE (
;  321     0550  5                  NAME_DESC,
;  322     0551  5                  $DESCRIPTOR('SYS$SHARE:.EXE'),
;  323     0552  5                  AUX_FAB,
;  324     0553  5                  AUX_NAM,
;  325     0554  5                  NAME_STRING,
;  326     0555  5                  .ICB);
;  327     0556  5              IF NOT .STATUS THEN RETURN .STATUS;
;  328     0557  5
;  329     0558  5              ! Use the auxiliary buffers when decoding the image header
;  330     0559  5              ! from this point.
;  331     0560  5
;  332     0561  5              IHD_CTX [CTX_L_BUFFER] = AUX_BUFFER;
;  333     0562  5              IHD_CTX [CTX_L_IHDBUF] = AUX_IHD;
;  334     0563  5
;  335     0564  5              STATUS = IMG$GET_HEADER (.ICB);
;  336     0565  5              IF NOT .STATUS THEN RETURN .STATUS;
;  337     0566  4              END;
;  338     0567  4
;  339     0568  4          ! Reserve ISD storage if image is loaded from sequential device
;  340     0569  4          ! The allocation is for the worst case (all local ISDs)
;  341     0570  4
;  342     0571  4          IF .ICB [ICB$V_LOAD_IMAGE]
;  343     0572  4          THEN
;  344     0573  5              BEGIN
;  345     0574  5              BIND
;  346     0575  5                  IHD_CTX = .ICB [ICB$L_CONTEXT]: $BBLOCK,
;  347     0576  5                  IHD = .IHD_CTX [CTX_L_IHDBUF] : $BBLOCK;
;  348     0577  5              SIZE = 12 * (.IHD [IHD$B_ADRBLKCNT] * BYTES_PER_PAGE) / ISD$K_LENPRIV;
;  349     0578  5              IF NOT EXE$ALOP1PROC ( .SIZE; SIZE, OWN_STORAGE [SEQ_LOAD_ISDS])
;  350     0579  5              THEN RETURN SS$_INSFMEM;
;  351     0580  5              CH$FILL ( 0, .SIZE, .OWN_STORAGE [SEQ_LOAD_ISDS] );
;  352     0581  4              END;
;  353     0582  4
;  354     0583  4          STATUS = PROCESS_ISD_LIST (.ICB);
;  355     0584  4          IF NOT .STATUS
;  356     0585  4          THEN
;  357     0586  5              BEGIN
;  358     0587  5
;  359     0588  5              ! Check for sequential load device and release ISD storage
;  360     0589  5
;  361     0590  5              IF .ICB [ICB$V_LOAD_IMAGE]
;  362     0591  5              THEN
;  363     0592  5                  EXE$DEAP1 (.OWN_STORAGE [SEQ_LOAD_ISDS], .SIZE);
;  364     0593  5
;  365     0594  5              RETURN .STATUS;
;  366     0595  4              END;
```

```
;  367          0596  4
;  368          0597  4          ! See if image should be loaded now
;  369          0598  4
;  370          0599  4          IF .ICB [ICB$V_LOAD_IMAGE]
;  371          0600  4          THEN
;  372          0601  5              BEGIN
;  373          0602  5              STATUS = LOAD_SEQ_IMAGE ( .ICB );
;  374          0603  5              EXE$DEAP1 (.OWN_STORAGE [SEQ_LOAD_ISDS], .SIZE);
;  375          0604  5              IF NOT .STATUS THEN RETURN .STATUS;
;  376          0605  4              END;
;  377          0606  4
;  378          0607  3          END;                                ! End of "it is not yet mapped" block
;  379          0608  3
;  380          0609  2      END;                                    ! End of WHILE loop
;  381          0610  2
;  382          0611  2  RETURN SS$_NORMAL;
;  383          0612  2
;  384          0613  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    84       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  DO 0010D          MOVL    R0, STATUS                                  ;
                     51      58  DO 00110          MOVL    SIZE, R1                                    ; 0603
                     50      6A  DO 00113          MOVL    OWN_STORAGE+40, R0                          ;
                             6B  16 00116          JSB     EXE$DEAP1                                   ;
                     E8      57  E8 00118          BLBS    STATUS, 10$                                 ; 0604
                     50      57  DO 0011B 12$:     MOVL    STATUS, R0                                  ;
                                 04 0011E          RET                                                 ;
                     50      01  DO 0011F 13$:     MOVL    #1, R0                                      ; 0611
                                 04 00122          RET                                                 ; 0613

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

```
;  386          0614  1 %SBTTL 'PROCESS_ISD_LIST - Convert Each ISD into Mapping Request'
;  387          0615  1
;  388          0616  1 ROUTINE PROCESS_ISD_LIST (ICB_PTR) =
;  389          0617  1
;  390          0618  1 !++
;  391          0619  1 ! FUNCTIONAL DESCRIPTION:
;  392          0620  1 !
;  393          0621  1 !       This routine converts each ISD in the current image into mapping
;  394          0622  1 !       requests to one of the memory management system services. Global
;  395          0623  1 !       ISDs are converted into new work list items to be processes at a
;  396          0624  1 !       later time.
;  397          0625  1 !
;  398          0626  1 ! CALLING SEQUENCE:
;  399          0627  1 !
;  400          0628  1 !       PROCESS_ISD_LIST (ICB pointer)
;  401          0629  1 !
;  402          0630  1 ! FORMAL PARAMETERS:
;  403          0631  1 !
;  404          0632  1 !       TBS
;  405          0633  1 !
;  406          0634  1 ! STATUS CODES:
;  407          0635  1 !
;  408          0636  1 !       TBS
;  409          0637  1 !--
;  410          0638  1
;  411          0639  2 BEGIN
;  412          0640  2
;  413          0641  2 BIND
;  414          0642  2     ICB     = .ICB_PTR                  : $BBLOCK,
;  415          0643  2     KFE     = .ICB [ICB$L_KFE]          : $BBLOCK,
;  416          0644  2     IHD_CTX = .ICB [ICB$L_CONTEXT]      : $BBLOCK,
;  417          0645  2     FLAGS   = OWN_STORAGE [INPUT_FLAGS] : $BBLOCK;
;  418          0646  2
;  419          0647  2 LOCAL
;  420          0648  2     SEQ_ISD                     : REF VECTOR [],
;  421          0649  2     GBL_ICB                     : REF $BBLOCK
;  422          0650  2                                   INITIAL (0),
;  423          0651  2     ISD_BUFFER                  : $BBLOCK [ISD$K_MAXLENGLBL],
;  424          0652  2     ISD                         : REF $BBLOCK
;  425          0653  2                                   INITIAL (0),
;  426          0654  2     GLOBAL_SEC_NAME_DESC        : $BBLOCK [DSC$K_S_BLN],
;  427          0655  2     MAP_BASE_ADDRESS,
;  428          0656  2     ISD_BASE_ADDRESS,
;  429          0657  2     PAGE_COUNT,
;  430          0658  2     INPUT_RANGE                 : VECTOR [2],
;  431          0659  2     RETURN_RANGE                : VECTOR [2],
;  432          0660  2     ACCUMULATED_RANGE           : VECTOR [2]
;  433          0661  2                                   INITIAL (0,0),
;  434          0662  2     SECTION_FLAGS,
;  435          0663  2     BASE_ADDRESS_STORED         : INITIAL (FALSE),
;  436          0664  2     FIRST_MAPPING               : INITIAL (TRUE),
;  437          0665  2     STATUS;
;  438          0666  2
;  439          0667  2 ! A base address is selected that will determine the actual range passed to
;  440          0668  2 ! the various mapping requests. If the ICB contains an explicit base address,
;  441          0669  2 ! that address is used. Otherwise, the address following the last mapping
;  442          0670  2 ! request is used. Note that this latter situation includes the case where
```

```
;  443          0671  2 ! no input range is specified for the activation of a main program. Because
;  444          0672  2 ! the CHECK_PARAMS routine sets RETURN_END_RANGE to FFFFFFFF, this logic
;  445          0673  2 ! chooses 0 as MAP_BASE_ADDRESS.
;  446          0674  2
;  447          0675  3 MAP_BASE_ADDRESS = (
;  448          0676  3     IF .ICB [ICB$V_EXPREG]
;  449          0677  3     THEN
;  450          0678  3         .OWN_STORAGE [RETURN_END_ADDRESS] + 1
;  451          0679  3     ELSE
;  452          0680  2         .ICB [ICB$L_STARTING_ADDRESS]);
;  453          0681  2
;  454          0682  2 ! Remember where ISD storage is for a sequential device load
;  455          0683  2 ! The storage location is meaningless in all other case
;  456          0684  2
;  457          0685  2 SEQ_ISD = .OWN_STORAGE [SEQ_LOAD_ISDS];
;  458          0686  2
;  459          0687  2 WHILE TRUE DO                              ! Idiom for DO FOREVER
;  460          0688  2
;  461          0689  3     BEGIN
;  462          0690  3
;  463          0691  3     ! If the header is resident, then it is not necessary to decode the ISDs
;  464          0692  3     ! as that has already been done. For images where ISD decoding is
;  465          0693  3     ! required, the context passed from IMG$DECODE_IHD to IMG$GET_NEXT_ISD is
;  466          0694  3     ! contained in a context block located through an ICB pointer.
;  467          0695  3
;  468          0696  3     IF .ICB [ICB$V_RES_HEADER]
;  469          0697  3     THEN
;  470          0698  4         BEGIN
;  471          0699  4
;  472          0700  4         BIND
;  473          0701  4             IHD = .KFE [KFE$L_IMGHDR]   : $BBLOCK;
;  474          0702  4
;  475          0703  5         ISD = (IF .ISD EQL 0
;  476          0704  5               THEN IHD + .IHD [IHD$W_SIZE]
;  477          0705  4               ELSE .ISD + .ISD [ISD$W_SIZE]);
;  478          0706  4
;  479          0707  4         ! The next test is the loop breaker for the resident header case.
;  480          0708  4         ! An ISD size of zero indicates the end of the ISD list.
;  481          0709  4
;  482          0710  4         IF .ISD [ISD$W_SIZE] EQL 0
;  483          0711  4             THEN
;  484          0712  5                 BEGIN
;  485          0713  5
;  486          0714  5                 ! Update the address range in the ICB with the accumulated
;  487          0715  5                 ! range into which the image was mapped.
;  488          0716  5
;  489          0717  5                 ICB [ICB$L_STARTING_ADDRESS] = .ACCUMULATED_RANGE [0];
;  490          0718  5                 ICB [ICB$L_END_ADDRESS] = .ACCUMULATED_RANGE [1];
;  491          0719  5
;  492          0720  5                 RETURN SS$_NORMAL
;  493          0721  4                 END;
;  494          0722  4         END
;  495          0723  3     ELSE
;  496          0724  4         BEGIN
;  497          0725  4
;  498          0726  4         STATUS = IMG$GET_NEXT_ISD (
;  499          0727  4             .ICB [ICB$W_CHAN],
```

```
;  500          0728  4              .IHD_CTX [CTX_L_BUFFER],
;  501          0729  4              .IHD_CTX [CTX_L_IHDBUF],
;  502          0730  4              IHD_CTX [CTX_L_VBN],
;  503          0731  4              IHD_CTX [CTX_W_ISD_OFFSET],
;  504          0732  4              ISD_BUFFER);
;  505          0733  4
;  506          0734  4          ! The following test is the loop breaker in the case where we are
;  507          0735  4          ! making validity checks on the ISD contents as we go along. The
;  508          0736  4          ! routine IMG$GET_NEXT_ISD returns a status of IMG$_ENDOFHDR when it
;  509          0737  4          ! reaches the end of the ISD list.
;  510          0738  4
;  511          0739  4          IF NOT .STATUS THEN
;  512          0740  4              IF .STATUS EQL IMG$_ENDOFHDR
;  513          0741  4              THEN
;  514          0742  5                  BEGIN
;  515          0743  5
;  516          0744  5                  ! Update the address range in the ICB with the accumulated
;  517          0745  5                  ! range into which the image was mapped.
;  518          0746  5
;  519          0747  5                  ICB [ICB$L_STARTING_ADDRESS] = .ACCUMULATED_RANGE [0];
;  520          0748  5                  ICB [ICB$L_END_ADDRESS] = .ACCUMULATED_RANGE [1];
;  521          0749  5
;  522          0750  5                  RETURN SS$_NORMAL
;  523          0751  5                  END
;  524          0752  4              ELSE
;  525          0753  4                  RETURN .STATUS;
;  526          0754  4
;  527          0755  4          ISD = ISD_BUFFER;
;  528          0756  4
;  529          0757  3          END;
;  530          0758  3
;  531          0759  3      ! The common mapping parameters are computed from the ISD contents and
;  532          0760  3      ! previous mapping context and stored in a suitable place for use by the
;  533          0761  3      ! various memory management system services.
;  534          0762  3      !
;  535          0763  3      ! The set of addresses that are passed to the various memory
;  536          0764  3      ! management system services depend on the kind of image that is
;  537          0765  3      ! being activated, and the kind of ISDs that exist within  these
;  538          0766  3      ! images. The various cases are described in an approximate order
;  539          0767  3      ! of occurrence.
;  540          0768  3      !
;  541          0769  3      ! 1.  New shareable images that are not based (that is, they
;  542          0770  3      !     are PIC and based at zero) have the ISD addresses added
;  543          0771  3      !     to the highest address previously mapped.
;  544          0772  3      !
;  545          0773  3      ! 2.  New based shareable images use the addresses contained
;  546          0774  3      !     in the ISDs.
;  547          0775  3      !
;  548          0776  3      ! 3.  When processing old shareable images, the addresses
;  549          0777  3      !     contained in the ISDs of the main program are taken at
;  550          0778  3      !     face value. If the "last cluster" flag is set, the end
;  551          0779  3      !     address is set to 7FFFFFFF to disable all upper limit
;  552          0780  3      !     address checks. Note that the last cluster flag is
;  553          0781  3      !     ignored in new shareable images.
;  554          0782  3      !
;  555          0783  3      !!! NOW THAT I HAVE THIS WONDERFUL COMMENT, I OUGHT TO MAKE THE CODE
;  556          0784  3      !!! BEHAVE THE WAY THAT IT'S SUPPOSED TO.
```

```
;  557          0785  3
;  558          0786  3      PAGE_COUNT = .ISD [ISD$W_PAGCNT];
;  559          0787  3      IF .PAGE_COUNT EQL 0 THEN RETURN SS$_BADISD;
;  560          0788  3
;  561          0789  3      ISD_BASE_ADDRESS = ((.ISD [ISD$V_VPG]) * BYTES_PER_PAGE);
;  562          0790  3
;  563          0791  3      IF NOT .BASE_ADDRESS_STORED
;  564          0792  3      THEN
;  565          0793  4          BEGIN
;  566          0794  4
;  567          0795  4          ! Renormalize the base address for merged activations and for all
;  568          0796  4          ! ICBs that represent implicitly referenced shareable images.
;  569          0797  4
;  570          0798  4          IF .ICB [ICB$B_ACT_CODE] NEQ ICB$K_MAIN_PROGRAM
;  571          0799  4          THEN
;  572          0800  4              MAP_BASE_ADDRESS = .MAP_BASE_ADDRESS - .ISD_BASE_ADDRESS;
;  573          0801  4
;  574          0802  4          ! The base address stored in the ICB is used by $IMGFIX to perform
;  575          0803  4          ! address relocation fixups. The transfer address bias is used to
;  576          0804  4          ! adjust transfer addresses that lie within the bounds of an image's
;  577          0805  4          ! address space.
;  578          0806  4
;  579          0807  4          ICB [ICB$L_BASE_ADDRESS] = .MAP_BASE_ADDRESS + .ISD_BASE_ADDRESS;
;  580          0808  4          BASE_ADDRESS_STORED = TRUE;
;  581          0809  4
;  582          0810  4          IF NOT .OWN_STORAGE [TRANSFER_BIAS_STORED]
;  583          0811  4          THEN
;  584          0812  5              BEGIN
;  585          0813  5              OWN_STORAGE [TRANSFER_ARRAY_BIAS] = .MAP_BASE_ADDRESS;
;  586          0814  5              OWN_STORAGE [TRANSFER_BIAS_STORED] = TRUE;
;  587          0815  4              END;
;  588          0816  3          END;
;  589          0817  3
;  590          0818  3      INPUT_RANGE [0] = .MAP_BASE_ADDRESS + .ISD_BASE_ADDRESS;
;  591          0819  3      INPUT_RANGE [1] = .INPUT_RANGE [0] + ((.PAGE_COUNT * BYTES_PER_PAGE) - 1);
;  592          0820  3
;  593          0821  3      CASE .ISD [ISD$L_FLAGS]                 ! Form case index from low four
;  594          0822  3           AND                                !  bits of the ISD flags longword
;  595          0823  3           ISD_M_DISPATCH_FLAGS
;  596          0824  3      FROM 0 TO ISD_M_DISPATCH_FLAGS OF
;  597          0825  3
;  598          0826  3      SET
```

```
;  600          0827  3 %SBTTL 'Create Address Space for Demand-Zero ISD'
;  601          0828  3
;  602          0829  3     [ISD$M_DZRO OR ISD$M_WRT , ISD$M_DZRO OR ISD$M_WRT OR ISD$M_CRF]:
;  603          0830  3
;  604          0831  3     ! The section is a demand-zero section. The pages are created at the
;  605          0832  3     ! current end of the image.
;  606          0833  3
;  607          0834  4         BEGIN
;  608          0835  4
;  609          0836  4         IF .ISD [ISD$B_TYPE] NEQ !SD$K_USRSTACK
;  610          0837  4         THEN
;  611          0838  5             BEGIN
;  612        P 0839  5             STATUS = $CRETVA (
;  613        P 0840  5                 INADR = INPUT_RANGE,
;  614        P 0841  5                 RETADR = RETURN_RANGE,
;  615          0842  5                 ACMODE = .OWN_STORAGE [ACCESS_MODE] );
;  616          0843  5             IF NOT .STATUS THEN RETURN .STATUS;
;  617          0844  5             END
;  618          0845  4         ELSE
;  619          0846  4             OWN_STORAGE [USER_STACK_SIZE] =
;  620          0847  4                 .OWN_STORAGE [USER_STACK_SIZE] + .PAGE_COUNT;
;  621          0848  3         END;
;  622          0849  3
```

```
;  624          0850  3 %SBTTL 'Create or Map Address Space for Private ISD'
;  625          0851  3
;  626          0852  3     [ISD$M_CRF OR ISD$M_WRT , ISD$M_WRT , 0]:
;  627          0853  3
;  628          0854  3     ! The section is a private section that will be mapped into the process
;  629          0855  3     ! address space. The section may be read only or writable. Read-only
;  630          0856  3     ! sections may either be global sections or private sections. Writable CRF
;  631          0857  3     ! sections are always created as private sections. Writable sections that
;  632          0858  3     ! are not also copy on reference are always mapped as global global
;  633          0859  3     ! sections.
;  634          0860  3
;  635          0861  4         BEGIN
;  636          0862  4         LITERAL SEC_M_EXEC_OWNED = PSL$C_EXEC ^ $BITPOSITION (SEC$V_WRTMOD);
;  637          0863  4
;  638          0864  4         ! If the SHAREABLE bit is set in the ICB, then global sections were
;  639          0865  4         ! created for the read-only sections of this image. These global
;  640          0866  4         ! sections can be mapped. In other cases, private sections are
;  641          0867  4         ! created.
;  642          0868  4
;  643          0869  4         SECTION_FLAGS = (.ISD [ISD$L_FLAGS] AND ISD_M_DISPATCH_FLAGS);
;  644          0870  4
;  645          0871  4         ! If the section is marked protected, then the ownership and
;  646          0872  4         ! writability of the section pages is restricted to exec mode.
;  647          0873  4
;  648          0874  4         IF .ISD [ISD$V_PROTECT]
;  649          0875  4         THEN
;  650          0876  4             SECTION_FLAGS = .SECTION_FLAGS
;  651          0877  4                             OR
;  652          0878  4                             SEC$M_PROTECT
;  653          0879  4                             OR
;  654          0880  4                             SEC_M_EXEC_OWNED;
;  655          0881  4
;  656          0882  4         IF
;  657          0883  4             .ICB [ICB$V_SHAREABLE]
;  658          0884  4             AND
;  659          0885  6                 ((NOT .ISD [ISD$V_WRT])
;  660          0886  5                 OR
;  661          0887  5                 (.ISD [ISD$V_WRT] AND NOT .ISD [ISD$V_CRF]))
;  662          0888  4         THEN
;  663          0889  5             BEGIN
;  664          0890  5
;  665          0891  5             BIND
;  666          0892  5                 GSD_NAME = IHD_CTX [CTX_T_GSD_NAME] : VECTOR [, BYTE];
;  667          0893  5
;  668          0894  5             GLOBAL_SEC_NAME_DESC [DSC$W_LENGTH] = .GSD_NAME [0] + 4;
;  669          0895  5             GLOBAL_SEC_NAME_DESC [DSC$A_POINTER] = GSD_NAME [1];
;  670          0896  5
;  671          0897  5             NEXT_GBL_SEC_NAME (GSD_NAME);
;  672          0898  5
;  673        P 0899  5             STATUS = $MGBLSC (
;  674        P 0900  5                 INADR  = INPUT_RANGE ,
;  675        P 0901  5                 RETADR = RETURN_RANGE ,
;  676        P 0902  5                 ACMODE = .OWN_STORAGE [ACCESS_MODE] ,
;  677        P 0903  5                 FLAGS  = (.SECTION_FLAGS OR SEC$M_GBL OR SEC$M_SYSGBL) ,
;  678        P 0904  5                 GSDNAM = GLOBAL_SEC_NAME_DESC ,
;  679          0905  5                 IDENT  = ICB [ICB$Q_IDENT] );
;  680          0906  5             IF NOT .STATUS THEN RETURN .STATUS;
```

```
;  681          0907  5              END
;  682          0908  4          ELSE
;  683          0909  5              BEGIN
;  684          0910  5
;  685          0911  5              ! Private copies of writable sections that are not also copy on
;  686          0912  5              ! reference are prohibited.
;  687          0913  5
;  688          0914  5              IF .ISD [ISD$V_WRT] AND NOT .ISD [ISD$V_CRF]
;  689          0915  5              THEN
;  690          0916  5                  RETURN SS$_NOTINSTALL;
;  691          0917  5
;  692          0918  5              ! Check for loading from a sequential device before deciding
;  693          0919  5              ! whether to map the section or simply record the load data
;  694          0920  5
;  695          0921  5              IF NOT .ICB [ICB$V_LOAD_IMAGE]
;  696          0922  5              THEN
;  697          0923  6                  BEGIN
;  698        P 0924  6                  STATUS = $CRMPSC (
;  699        P 0925  6                      INADR  = INPUT_RANGE ,
;  700        P 0926  6                      RETADR = RETURN_RANGE ,
;  701        P 0927  6                      ACMODE = .OWN_STORAGE [ACCESS_MODE] ,
;  702        P 0928  6                      FLAGS  = .SECTION_FLAGS ,
;  703        P 0929  6                      CHAN   = .ICB [ICB$W_CHAN] ,
;  704        P 0930  6                      PAGCNT = .PAGE_COUNT ,
;  705        P 0931  6                      VBN    = .ISD [ISD$L_VBN] ,
;  706          0932  6                      PFC    = .ISD [ISD$B_PFC] );
;  707          0933  6                  IF NOT .STATUS THEN RETURN .STATUS;
;  708          0934  6                  END
;  709          0935  5              ELSE
;  710          0936  5
;  711          0937  5                  ! Record size of image section, starting address, and flags
;  712          0938  5                  ! Advance pointer for the next ISD
;  713          0939  5
;  714          0940  6                  BEGIN
;  715          0941  6                  SEQ_ISD [0] = .PAGE_COUNT;
;  716          0942  6                  SEQ_ISD [1] = .INPUT_RANGE[0];
;  717          0943  6                  SEQ_ISD [2] = .ISD [ISD$L_FLAGS];
;  718          0944  6                  SEQ_ISD = SEQ_ISD [3];
;  719          0945  5                  END;
;  720          0946  5
;  721          0947  5              ! If the section just mapped is a fixup vector, then it must be
;  722          0948  5              ! verified and added to the list of fixup vectors. Note that the
;  723          0949  5              ! placement of this code assumes that fixup sections can never be
;  724          0950  5              ! shared. That is, they are always writable CRF sections.
;  725          0951  5
;  726          0952  6              IF (
;  727          0953  7                  (.ISD [ISD$V_FIXUPVEC])
;  728          0954  6                  AND
;  729          0955  7                  (NOT (.OWN_STORAGE [P1_MERGE_P0]))
;  730          0956  6                 )
;  731          0957  5              THEN
;  732          0958  5
;  733          0959  5                  ! If the image being mapped is a privileged shareable image,
;  734          0960  5                  ! then perform the fixups immediately in exec mode. Otherwise,
;  735          0961  5                  ! add the fixup vector to the list of fixups to be done.
;  736          0962  5                  ! This work is delayed for images loaded from sequential devices
;  737          0963  5
```

```
;  738     0964  5                 IF .ISD [ISD$V_PROTECT]
;  739     0965  5                 THEN
;  740     0966  6                     STATUS = (
;  741     0967  6                         IF
;  742     0968  7                             BEGIN
;  743     0969  8                             IF (KFE EQL 0)
;  744     0970  7                             THEN
;  745     0971  7                                 TRUE
;  746     0972  7                             ELSE
;  747     0973  8                                 NOT (.KFE [KFE$V_PROTECT] AND .KFE [KFE$V_SHARED])
;  748     0974  7                             END
;  749     0975  6                         THEN
;  750     0976  6                             SS$_PROTINSTALL
;  751     0977  6                         ELSE
;  752     0978  6                             IMG$PRVSHRIMG (
;  753     0979  6                                 .RETURN_RANGE,
;  754     0980  6                                 .ICB [ICB$L_BASE_ADDRESS]))
;  755     0981  5                 ELSE
;  756     0982  5                     IF NOT .ICB [ICB$V_LOAD_IMAGE]
;  757     0983  5                     THEN
;  758     0984  5                         STATUS = ADD_FIXUP_VECTOR (
;  759     0985  5                             RETURN_RANGE,
;  760     0986  5                             .ICB [ICB$L_BASE_ADDRESS]);
;  761     0987  5             IF NOT .STATUS THEN RETURN .STATUS
;  762     0988  4             END;
;  763     0989  4
;  764     0990  4         ! At this point, we need to check whether the section just mapped
;  765     0991  4         ! is a message section or a vector section. We need to put the check
;  766     0992  4         ! here because, while vector sections can only be global sections,
;  767     0993  4         ! message sections can be either private or global.
;  768     0994  4         ! This work is delayed for images loaded from sequential devices
;  769     0995  4
;  770     0996  5         IF (
;  771     0997  6             (.ISD [ISD$V_VECTOR])
;  772     0998  5             AND
;  773     0999  6             (NOT (.OWN_STORAGE [P1_MERGE_PO]))
;  774     1000  5            )
;  775     1001  4         THEN
;  776     1002  4             IF .ISD [ISD$V_PROTECT]
;  777     1003  4             THEN
;  778     1004  5                 BEGIN
;  779     1005  5                 IF NOT .ICB [ICB$V_LOAD_IMAGE]
;  780     1006  5                 THEN
;  781     1007  6                     BEGIN
;  782     1008  6                     STATUS = ADD_PRIVILEGED_VECTOR (RETURN_RANGE [0], ICB);
;  783     1009  6                     IF NOT .STATUS THEN RETURN .STATUS
;  784     1010  5                     END;
;  785     1011  5                 END
;  786     1012  4             ELSE
;  787     1013  4                 RETURN SS$_BADISD;
;  788     1014  4
;  789     1015  3         END;
;  790     1016  3
```

```
;   792          1017  3 %SBTTL 'Add ICB to Work List for Global ISD'
;   793          1018  3
;   794          1019  3 [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$M_DZR
;   795          1020  3
;   796          1021  3     ! This is a global ISD. An entry will be made in the work list and the
;   797          1022  3     ! actual mapping will occur in a later call to this routine. A check is
;   798          1023  3     ! made to determine whether we are processing a list of global ISDs of
;   799          1024  3     ! the same name. This is done by comparing the name in the global ISD
;   800          1025  3     ! with the name in the most recently allocated ICB.
;   801          1026  3
;   802          1027  4         BEGIN
;   803          1028  4
;   804          1029  4         BIND
;   805          1030  4             ISD_NAM = ISD [ISD$T_GBLNAM]        : VECTOR [,BYTE];
;   806          1031  4
;   807          1032  4         LOCAL
;   808          1033  4             ICB_NAM                             : REF VECTOR [,BYTE];
;   809          1034  4
;   810          1035  4         IF
;   811          1036  5             BEGIN
;   812          1037  6                 IF (.GBL_ICB EQL 0)
;   813          1038  5                 THEN TRUE
;   814          1039  5                 ELSE
;   815          1040  6                     BEGIN
;   816          1041  6                     ICB_NAM = GBL_ICB [ICB$T_IMAGE_NAME];
;   817          1042  6                     CH$NEQ (
;   818          1043  6                         .ISD_NAM [0] - 4, ISD_NAM [1],
;   819          1044  6                         .ICB_NAM [0], ICB_NAM [1], 0)
;   820          1045  6                     END
;   821          1046  5             END
;   822          1047  4         THEN
;   823          1048  5             BEGIN
;   824          1049  5
;   825          1050  5             STATUS = IMG$ALLOCATE_ICB (GBL_ICB);
;   826          1051  5             IF NOT .STATUS THEN RETURN .STATUS;
;   827          1052  5
;   828          1053  5             GBL_ICB [ICB$B_ACT_CODE] = ICB$K_GLOBAL_IMAGE_SECTION;
;   829          1054  5
;   830          1055  5             ICB_NAM = GBL_ICB [ICB$T_IMAGE_NAME];
;   831          1056  5
;   832          1057  5             ! Move the section name including the count and the
;   833          1058  5             ! trailing _00n into the ICB. Insert the ICB at the tail of the
;   834          1059  5             ! work list. Subtract four from the count so that the _00n prefix,
;   835          1060  5             ! while present in the ICB, is not represented in the count.
;   836          1061  5
;   837          1062  5             ICB_NAM [0] = .ISD_NAM [0] - 4;
;   838          1063  5             CH$MOVE (.ISD_NAM [0], ISD_NAM [1], ICB_NAM [1]);
;   839          1064  5
;   840          1065  5             GBL_ICB [ICB$L_MATCH_CONTROL] = .ISD [ISD$V_MATCHCTL];
;   841          1066  5             GBL_ICB [ICB$L_VERSION] = .ISD [ISD$L_IDENT];
;   842          1067  5             IF
;   843          1068  6                 (.IHD_CTX [CTX_W_GENERATION] GEQU IHD$C_GEN_FIXUP)
;   844          1069  5                 AND
;   845          1070  6                 NOT (.ISD [ISD$V_BASED])
;   846          1071  5             THEN
;   847          1072  6                 BEGIN
;   848          1073  6                 GBL_ICB [ICB$V_EXPREG] = TRUE;
```

```
;  849          1074  6                  GBL_ICB [ICB$L_STARTING_ADDRESS] = 0;
;  850          1075  6                  GBL_ICB [ICB$L_END_ADDRESS] = END_OF_P1_SPACE;
;  851          1076  6                  END
;  852          1077  5              ELSE
;  853          1078  6                  BEGIN
;  854          1079  6                  GBL_ICB [ICB$L_STARTING_ADDRESS] = .INPUT_RANGE [0];
;  855          1080  7                  GBL_ICB [ICB$L_END_ADDRESS] = (
;  856          1081  7                      IF .ISD [ISD$V_LASTCLU]
;  857          1082  7                      THEN END_OF_P1_SPACE
;  858          1083  6                      ELSE .INPUT_RANGE [1] );
;  859          1084  5                  END;
;  860          1085  5
;  861          1086  5              INSQUE (.GBL_ICB , .IAC$GL_WORK_LIST [1]);
;  862          1087  5              END
;  863          1088  4          ELSE
;  864          1089  5              BEGIN
;  865          1090  5
;  866          1091  5              ! This is another global ISD of the same name. Some consistency
;  867          1092  5              ! checks are performed on the address range. The address range
;  868          1093  5              ! stored in the ICB may change.
;  869          1094  5
;  870          1095  6              IF (.INPUT_RANGE [0] LEQU .GBL_ICB [ICB$L_STARTING_ADDRESS])
;  871          1096  5              THEN RETURN SS$_BADISD;
;  872          1097  5
;  873          1098  5              IF .GBL_ICB [ICB$L_END_ADDRESS] NEQ END_OF_P1_SPACE
;  874          1099  5              THEN
;  875          1100  6                  BEGIN
;  876          1101  6
;  877          1102  7                  IF (.INPUT_RANGE [0] LEQU .GBL_ICB [ICB$L_END_ADDRESS])
;  878          1103  6                  THEN RETURN SS$_BADISD;
;  879          1104  6
;  880          1105  6                  GBL_ICB [ICB$L_END_ADDRESS] = .INPUT_RANGE [1];
;  881          1106  5                  END;
;  882          1107  4              END;
;  883          1108  4
;  884          1109  4          ! If a global ISD has the WRT flag set but the CRF and DZRO bits clear,
;  885          1110  4          ! this indicates a writable global section. A flag is set to indicate
;  886          1111  4          ! that the image file should be opened for write access.
;  887          1112  4
;  888          1113  5          IF .ISD [ISD$V_WRT] AND NOT (.ISD [ISD$V_CRF] OR .ISD [ISD$V_DZRO])
;  889          1114  4          THEN
;  890          1115  4              GBL_ICB [ICB$V_OPEN_FOR_WRITE] = TRUE;
;  891          1116  4
;  892          1117  3          END;
```

```
;   894          1118  3 %SBTTL 'Update return address arrays'
;   895          1119  3
;   896          1120  3     [INRANGE]:
;   897          1121  3
;   898          1122  3     ! An unrecognized combination of flags exists in this ISD. Because the
;   899          1123  3     ! linker would never produce such a combination, this is either not an
;   900          1124  3     ! image file or some tampering has occurred.
;   901          1125  3
;   902          1126  3         RETURN SS$_BADISD;
;   903          1127  3
;   904          1128  3     TES;
;   905          1129  3
;   906          1130  3 ! Update the address range arrays in the ICB and in impure storage.
;   907          1131  3 ! The latter may be returned to the caller and represents the total
;   908          1132  3 ! address space mapped in this activation.
;   909          1133  3 !
;   910          1134  3 ! Note that the following conditional update works correctly because
;   911          1135  3 ! ALL of the nonglobal ISDs and NONE of the global ISDs result in
;   912          1136  3 ! mapping requests.
;   913          1137  3
;   914          1138  3 IF NOT .ISD [ISD$V_GBL]
;   915          1139  3 THEN
;   916          1140  4     BEGIN
;   917          1141  4     IF .FIRST_MAPPING
;   918          1142  4     THEN
;   919          1143  5         BEGIN
;   920          1144  5         FIRST_MAPPING = FALSE;
;   921          1145  5         ACCUMULATED_RANGE [0] = .RETURN_RANGE [0];
;   922          1146  4         END;
;   923          1147  4
;   924          1148  4     IF .OWN_STORAGE [RETURN_START_ADDRESS] EQL -1
;   925          1149  4     THEN
;   926          1150  4         OWN_STORAGE [RETURN_START_ADDRESS] = .RETURN_RANGE [0];
;   927          1151  4
;   928          1152  4     ACCUMULATED_RANGE [1] = .RETURN_RANGE [1];
;   929          1153  4
;   930          1154  4     IF .RETURN_RANGE [1]  GTR  .OWN_STORAGE [RETURN_END_ADDRESS]
;   931          1155  4     THEN
;   932          1156  4         OWN_STORAGE [RETURN_END_ADDRESS] = .RETURN_RANGE [1];
;   933          1157  4
;   934          1158  3     END;
;   935          1159  3
;   936          1160  3 END                                     ! End of DO FOREVER loop
;   937          1161  3
;   938          1162  1 END;                                    ! End of routine PROCESS_ISD_LIST


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

                                                    0FFC 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                       ;
```

```
                                  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$:      MOVL    STATUS, R0                              ;
                                                                04  C9 002C9           RET     ;                                       ;
                                                57      24      AE  D0 002CA 39$:      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$:      MOVZWL  #8196, R0                               ;  1126
                                                                04  00368             RET                                             ;
                                                38      08      A9  E8 0036C 47$:      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
```

```
;  940          1163  1 %SBTTL 'ADD_PRIVILEGED_VECTOR - Install User-Written Change Mode Vectors'
;  941          1164  1
;  942          1165  1 ROUTINE ADD_PRIVILEGED_VECTOR (SECTION_ADDRESS, ICB_ADDRESS) =
;  943          1166  1
;  944          1167  1 !+
;  945          1168  1 ! Functional Description:
;  946          1169  1 !
;  947          1170  1 ! This routine is called when the section just mapped contains
;  948          1171  1 ! user-written change mode or message vectors. After verification, the
;  949          1172  1 ! vectors are added to the list of privileged routines called in
;  950          1173  1 ! response to an unfielded change mode call.
;  951          1174  1 !
;  952          1175  1 ! Calling Sequence:
;  953          1176  1 !
;  954          1177  1 !    ADD_PRIVILEGED_VECTOR (SECTION_ADDRESS, KFE_ADDRESS, ICB_ADDRESS)
;  955          1178  1 !
;  956          1179  1 ! Input Parameters:
;  957          1180  1 !
;  958          1181  1 !    SECTION_ADDRESS - Address of section just mapped. The initial portion
;  959          1182  1 !        of this section contains the privileged library vector (PLV).
;  960          1183  1 !
;  961          1184  1 !    ICB_ADDRESS - Address of image control block that describes image.
;  962          1185  1 !
;  963          1186  1 ! Implicit Input:
;  964          1187  1 !
;  965          1188  1 !    .ICB [ICB$L_KFE] - Address of known file entry for this image. (A change
;  966          1189  1 !        mode vector must be installed with a PROTECT option. Message
;  967          1190  1 !        vectors do not have to be installed.
;  968          1191  1 !-
;  969          1192  1
;  970          1193  2 BEGIN
;  971          1194  2
;  972          1195  2 BIND
;  973          1196  2     KERNEL_VECTOR  = CTL$A_DISPVEC,
;  974          1197  2     EXEC_VECTOR    = CTL$A_DISPVEC + (1*256),
;  975          1198  2     RUNDOWN_VECTOR = CTL$A_DISPVEC + (2*256),
;  976          1199  2     MESSAGE_VECTOR = CTL$A_DISPVEC + (3*256);
;  977          1200  2
;  978          1201  2 BIND
;  979          1202  2     PLV = .SECTION_ADDRESS : REF $BBLOCK,
;  980          1203  2     ICB = .ICB_ADDRESS     : $BBLOCK,
;  981          1204  2     KFE = .ICB[ICB$L_KFE] : $BBLOCK;
;  982          1205  2
;  983          1206  2 LITERAL
;  984          1207  2     VECTOR_SIZE = 256,                    ! One half page for each vector
;  985          1208  2     ABSOLUTE_MODE = %X'9F',
;  986          1209  2     AT_R5_MODE = %X'65',
;  987          1210  2     RSB_ABSOLUTE = (ABSOLUTE_MODE ^ 8) OR OP$_RSB : UNSIGNED (16);
;  988          1211  2
;  989          1212  2 LOCAL
;  990          1213  2     NEW_VECTOR_LOC;
;  991          1214  2
;  992          1215  2 CASE .PLV [PLV$L_TYPE]
;  993          1216  2     FROM PLV$C_TYP_CMOD TO PLV$C_TYP_MSG OF
;  994          1217  2     SET
;  995          1218  2
;  996          1219  2     [PLV$C_TYP_CMOD]:
```

```
;  997          1220  2
;  998          1221  2      ! The section contains a user-written change-mode dispatcher. The beginning
;  999          1222  2      ! of the section contains a privileged library vector, laid out as follows.
; 1000          1223  2      !
; 1001          1224  2      !     .PLV [PLV$L_TYPE]             ! Vector type code (PLV$C_TYP_CMOD)
; 1002          1225  2      !
; 1003          1226  2      !     .PLV [PLV$L_VERSION]          ! System version number (SYS$K_VERSION)
; 1004          1227  2      !
; 1005          1228  2      !     .PLV [PLV$L_KERNEL]           ! Offset to kernel mode dispatcher
; 1006          1229  2      !
; 1007          1230  2      !     .PLV [PLV$L_EXEC]             ! Offset to exec mode dispatcher
; 1008          1231  2      !
; 1009          1232  2      !     .PLV [PLV$L_USRUNDWN]         ! Offset to rundown routine
; 1010          1233  2      !
; 1011          1234  2      !     .PLV_L_CMOD_RFU               ! Reserved longword
; 1012          1235  2      !
; 1013          1236  2      !     .PLV [PLV$L_RMS]              ! Offset to alternate RMS dispatcher
; 1014          1237  2      !
; 1015          1238  2      !     .PLV [PLV$L_CHECK]            ! Address check
; 1016          1239  2      !
; 1017          1240  2      ! If this last longword contains nonzero, its contents must be equal to
; 1018          1241  2      ! the base address of the section.
; 1019          1242  2
; 1020          1243  3      BEGIN
; 1021          1244  3
; 1022          1245  3      BIND
; 1023          1246  3          PLV_L_CMOD_RFU = PLV [PLV$L_USRUNDWN] + 4,
; 1024          1247  3          DISP_VEC = PLV [PLV$L_KERNE[] : VECTOR [3];
; 1025          1248  3
; 1026          1249  3      ! Check that the privileged shareable image is linked against the current
; 1027          1250  3      ! system and make other sanity checks on the vector contents. The CHECK
; 1028          1251  3      ! field insures that a position dependent privileged shareable image is
; 1029          1252  3      ! mapped at the correct address. If the contents of the PLV$L_CHECK are
; 1030          1253  3      ! not zero, the contents must be equal to the address of the cell.
; 1031          1254  3
; 1032          1255  3      IF
; 1033          1256  4          (
; 1034          1257  5              (.PLV [PLV$L_VERSION] EQL 0)
; 1035          1258  4              AND
; 1036          1259  4              .ICB [ICB$V_SYS_STB]
; 1037          1260  4          )
; 1038          1261  3          OR
; 1039          1262  4          (
; 1040          1263  5              (.PLV [PLV$L_VERSION] NEQ 0)
; 1041          1264  4              AND
; 1042          1265  5              (.PLV [PLV$L_VERSION] NEQ SYS$K_VERSION)
; 1043          1266  4          )
; 1044          1267  3      THEN
; 1045          1268  3          RETURN SS$_BADVEC;
; 1046          1269  3
; 1047          1270  3      IF
; 1048          1271  4          (.PLV_L_CMOD_RFU NEQ 0)
; 1049          1272  3          OR
; 1050          1273  4          (
; 1051          1274  5              (.PLV [PLV$L_CHECK] NEQ 0)
; 1052          1275  4              AND
; 1053          1276  5              (.PLV [PLV$L_CHECK] NEQ PLV [PLV$L_CHECK])
```

```
;  1054          1277  4          )
;  1055          1278  3      THEN
;  1056          1279  3          RETURN SS$_BADVEC;
;  1057          1280  3
;  1058          1281  3      ! A privileged shareable image must be installed (KFE must exist) with
;  1059          1282  3      ! both the PROTECT and SHARED qualifiers.
;  1060          1283  3
;  1061          1284  3      IF
;  1062          1285  4          BEGIN
;  1063          1286  5          IF (KFE EQL 0)
;  1064          1287  4          THEN
;  1065          1288  4              TRUE
;  1066          1289  4          ELSE
;  1067          1290  5              NOT (.KFE [KFE$V_PROTECT] AND .KFE [KFE$V_SHARED])
;  1068          1291  4          END
;  1069          1292  3      THEN
;  1070          1293  3          RETURN SS$_PROTINSTALL;
;  1071          1294  3
;  1072          1295  3      ! The same operations are performed for all three of kernel, exec,
;  1073          1296  3      ! and rundown routines.
;  1074          1297  3
;  1075          1298  3      INCR DISP_INDEX FROM 0 TO 2
;  1076          1299  3          DO
;  1077          1300  3
;  1078          1301  3          IF .DISP_VEC [.DISP_INDEX] NEQ 0
;  1079          1302  3          THEN
;  1080          1303  4              BEGIN
;  1081          1304  4
;  1082          1305  4              BIND
;  1083          1306  4                  SPECIAL_VECTOR = KERNEL_VECTOR + (VECTOR_SIZE * .DISP_INDEX);
;  1084          1307  4
;  1085          1308  4              ! There must be enough room left in the vector to accommodate the
;  1086          1309  4              ! new JSB instruction (six bytes) and an RSB instruction (four
;  1087          1310  4              ! bytes to allow room for padding with zeros).
;  1088          1311  4
;  1089          1312  5              IF (VECTOR_SIZE - (.SPECIAL_VECTOR) ) LSSU (6 + 4)
;  1090          1313  4              THEN RETURN SS$_VECFULL;
;  1091          1314  4
;  1092          1315  4              ! A special instruction sequence (RSB followed by absolute
;  1093          1316  4              ! addressing) is inserted at the current end of the vector. After
;  1094          1317  4              ! the address fixups have been completed, the RSB will be replaced
;  1095          1318  4              ! with a JSB. The destination of the JSB is computed from the
;  1096          1319  4              ! contents and address of the current dispatch vector cell.
;  1097          1320  4
;  1098          1321  4              NEW_VECTOR_LOC = SPECIAL_VECTOR + .SPECIAL_VECTOR;
;  1099          1322  4              .NEW_VECTOR_LOC = RSB_ABSOLUTE;
;  1100          1323  4              (.NEW_VECTOR_LOC) + 2 = DISP_VEC [.DISP_INDEX] + .DISP_VEC [.DISP_INDEX];
;  1101          1324  4              (.NEW_VECTOR_LOC) + 6 = OP$_RSB;
;  1102          1325  4
;  1103          1326  4              ! Store the offset to the new location of the RSB
;  1104          1327  4
;  1105          1328  4              SPECIAL_VECTOR = ((.NEW_VECTOR_LOC) + 6) - SPECIAL_VECTOR;
;  1106          1329  4
;  1107          1330  4              IAC$GL_IMAGCTX [IMAGCTX$V_SETVECTOR] = TRUE;
;  1108          1331  4
;  1109          1332  3              END;
;  1110          1333  3
```

```
;  1111          1334  3      ! An alternate RMS dispatcher address is saved in OWN storage. The
;  1112          1335  3      ! completion routine that executes in kernel mode will store this
;  1113          1336  3      ! address into the cell used by the change-mode-to-exec handler.
;  1114          1337  3
;  1115          1338  3      IF .PLV [PLV$L_RMS] NEQ 0
;  1116          1339  3      THEN OWN_STORAGE [RMS_BASE] = PLV [PLV$L_RMS] + .PLV [PLV$L_RMS];
;  1117          1340  3
;  1118          1341  3      RETURN SS$_NORMAL;                      ! All done
;  1119          1342  3
;  1120          1343  2      END;                                    ! End of change mode section
;  1121          1344  2
;  1122          1345  2      [PLV$C_TYP_MSG]:
;  1123          1346  2
;  1124          1347  2      ! The section contains an image-specific message section, laid out in the
;  1125          1348  2      ! following way.
;  1126          1349  2      !
;  1127          1350  2      !     .PLV [PLV$L_TYPE]                 ! Vector type code (PLV$C_TYP_MSG)
;  1128          1351  2      !
;  1129          1352  2      !     .PLV_L_MSG_RFU                    ! Reserved longword (MBZ)
;  1130          1353  2      !
;  1131          1354  2      !     .PLV [PLV$L_MSGDSP]               ! Offset to message dispatcher (6)
;  1132          1355  2      !
;  1133          1356  2      ! This offset locates the JSB (R5) instruction.
;  1134          1357  2      !
;  1135          1358  2      ! The rest of the message section header contains instructions that are
;  1136          1359  2      ! interpreted (not executed) by the Get Message system service.
;  1137          1360  2      !
;  1138          1361  2      !     NOP                               ! The next two bytes could serve
;  1139          1362  2      !     NOP                               !   as an entry mask
;  1140          1363  2      !     JSB    (R5)                       ! The offset above locates this
;  1141          1364  2
;  1142          1365  3      BEGIN
;  1143          1366  3
;  1144          1367  3      BIND
;  1145          1368  3          PLV_L_MSG_RFU = PLV [PLV$L_VERSION],
;  1146          1369  3          PLV_L_MSG_ENTRY = PLV [PLV$L_EXEC],
;  1147          1370  3          PLV_L_MSC_OFFSET = PLV [PLV$[_USRUNDWN],
;  1148          1371  3          MSC = PLV_L_MSC_OFFSET + .PLV_L_MSC_OFFSET : BYTE ;
;  1149          1372  3
;  1150          1373  3      LITERAL
;  1151          1374  3          JSB_AT_R5 =
;  1152          1375  3              (AT_R5_MODE ^ 24) OR
;  1153          1376  3              (OP$_JSB ^ 16) OR
;  1154          1377  3              (OP$_NOP ^ 8) OR
;  1155          1378  3              (OP$_NOP);
;  1156          1379  3
;  1157          1380  3      LOCAL
;  1158          1381  3          PROT_DSC : VECTOR [2];
;  1159          1382  3
;  1160          1383  3      ! Verify the contents of the next three longwords in the vector
;  1161          1384  3
;  1162          1385  4      IF (
;  1163          1386  4          (.PLV_L_MSG_RFU NEQ 0) OR
;  1164          1387  4          (.PLV [PLV$[_MSGDSP] NEQ 6) OR
;  1165          1388  4          (.PLV_L_MSG_ENTRY NEQ JSB_AT_R5) )
;  1166          1389  3      THEN
;  1167          1390  3          RETURN SS$_BADVEC;
```

```
;  1168          1391  3
;  1169          1392  3      ! There must be enough room left in the vector to accommodate the new
;  1170          1393  3      ! JSB instruction (six bytes) and an RSB instruction (four bytes to allow
;  1171          1394  3      ! room for padding with zeros).
;  1172          1395  3
;  1173          1396  4      IF (VECTOR_SIZE - (.MESSAGE_VECTOR) ) LSSU (6 + 4)
;  1174          1397  3      THEN RETURN SS$_VECFULL;
;  1175          1398  3
;  1176          1399  3      ! Find the current end of the vector and store a JSB @# instruction
;  1177          1400  3      ! there, pointing to the JSB (R5) instruction in the message section.
;  1178          1401  3
;  1179          1402  3      NEW_VECTOR_LOC = MESSAGE_VECTOR + .MESSAGE_VECTOR;
;  1180          1403  3      .NEW_VECTOR_LOC = RSB_ABSOLUTE;
;  1181          1404  3      (.NEW_VECTOR_LOC) + 2 = PLV_L_MSG_ENTRY + 2;
;  1182          1405  3      (.NEW_VECTOR_LOC) + 6 = OP$_RSB;
;  1183          1406  3
;  1184          1407  3      ! Store the offset to the new location of the RSB
;  1185          1408  3
;  1186          1409  3      MESSAGE_VECTOR = ((.NEW_VECTOR_LOC) + 6) - MESSAGE_VECTOR;
;  1187          1410  3
;  1188          1411  3      IACSGL_IMAGCTX [IMAGCTX$V_SETVECTOR] = TRUE;
;  1189          1412  3
;  1190          1413  3      ! Finally, if this is a pointer (indirect) message section, the section
;  1191          1414  3      ! must be writable from user mode.
;  1192          1415  3
;  1193          1416  3      PROT_DSC [0] = ..SECTION_ADDRESS;   ! Only need to change first page
;  1194          1417  3      PROT_DSC [1] = .PROT_DSC [0] + 511;
;  1195          1418  3
;  1196          1419  3      IF .MSC EQL MSC$C_IND               ! Indirect message section?
;  1197          1420  3      THEN
;  1198        P 1421  3          $SETPRT (INADR = PROT_DSC,      ! Yes, set page protection
;  1199        P 1422  3                   PROT = PRT$C_UW,
;  1200          1423  3                   ACMODE = PSL$C_EXEC);
;  1201          1424  3
;  1202          1425  3      RETURN SS$_NORMAL;
;  1203          1426  3
;  1204          1427  2      END;                                ! End of message section
;  1205          1428  2
;  1206          1429  2      [OUTRANGE]:
;  1207          1430  2          RETURN SS$_BADVEC
;  1208          1431  2
;  1209          1432  2      YES
;  1210          1433  2
;  1211          1434  1 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                     62  CF 00018        CASEL   (R2), #1, #1            ; 1371
```

```
                       00A3            0007      0001C 1$:    .WORD    3$-1$,-                          :
                                                                       12$-1$                           :
                                    00B8  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$:   MOVZWL  #8244, R0                               ; 1397
                                                04  00F6            RET                                             ;
                51              53              56  C1 000F7 16$:   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
```

```
;  1213          1435  1 %SBTTL 'ADD_FIXUP_VECTOR - Add Fixup Vector to Work List'
;  1214          1436  1
;  1215          1437  1 ROUTINE ADD_FIXUP_VECTOR (SECTION_ADDRESS, BASE_ADDRESS) =
;  1216          1438  1
;  1217          1439  1 !+
;  1218          1440  1 ! Functional Description:
;  1219          1441  1 !
;  1220          1442  1 !     This routine processes an image section that contains a fixup vector. In
;  1221          1443  1 !     addition to some validation of the contents of the fixup vector, two
;  1222          1444  1 !     operations are performed.
;  1223          1445  1 !
;  1224          1446  1 !         The fixup vector is added to the front of the list of fixups that must
;  1225          1447  1 !         be done after the image is activated.
;  1226          1448  1 !
;  1227          1449  1 !         The base address of the image currently being activated is stored in
;  1228          1450  1 !         the SHL entry at index 0. Note that the linker did not store a name
;  1229          1451  1 !         here so the name field for SHL entry 0 is always left blank.
;  1230          1452  1 !
;  1231          1453  1 !     If the image being activated represents a privileged shareable image, then
;  1232          1454  1 !     this routine is not even called. Rather, a check is made to insure that
;  1233          1455  1 !     the image contains no outbound calls. Then the fixups are performed
;  1234          1456  1 !     immediately, rather than being postponed for later processing in user
;  1235          1457  1 !     mode.
;  1236          1458  1 !
;  1237          1459  1 ! Calling Sequence:
;  1238          1460  1 !
;  1239          1461  1 !     ADD_FIXUP_VECTOR ()
;  1240          1462  1 !
;  1241          1463  1 ! Formal Parameters:
;  1242          1464  1 !
;  1243          1465  1 !     SECTION_ADDRESS - Base address of image section that has been identified
;  1244          1466  1 !         as a fixup vector.
;  1245          1467  1 !
;  1246          1468  1 !     BASE_ADDRESS - Base address of image that contains the current fixup
;  1247          1469  1 !         vector.
;  1248          1470  1 !-
;  1249          1471  1
;  1250          1472  2 BEGIN
;  1251          1473  2
;  1252          1474  2 BIND
;  1253          1475  2     RANGE = .SECTION_ADDRESS              : VECTOR,
;  1254          1476  2     IAF   = .RANGE [0]                    : $BBLOCK,
;  1255          1477  2     SHL = IAF + .IAF [IAF$L_SHLSTOFF]     : $BBLOCK;
;  1256          1478  2
;  1257          1479  2 ! Calculate the address of the shareable image list, insure that it lies
;  1258          1480  2 ! within the bounds of the section just mapped, insure further that the
;  1259          1481  2 ! caller can write into the SHL, and store the base address of the image that
;  1260          1482  2 ! contains this SHL. If this SHL does not include a size, store a size that
;  1261          1483  2 ! reflects the size of an old SHL.
;  1262          1484  2
;  1263          1485  3 IF (SHL LSSU .RANGE [0]) OR ((SHL+SHL$C_LENGTH) GTRU .RANGE [1])
;  1264          1486  2 THEN RETURN IMG$_BAD_FIXUPVEC;
;  1265          1487  2
;  1266          1488  2 ! The following $ASSUME allows the BASEVA longword and SHL_SIZE byte
;  1267          1489  2 ! to have their writability checked with a single PROBE instruction.
;  1268          1490  2
;  1269          1491  2 $ASSUME ( $BYTEOFFSET (SHL$L_BASEVA) EQL 0);
```

```
1492  2 $ASSUME ( $BYTEOFFSET (SHL$B_SHL_SIZE) LEQU 512);
1493  2
1494  2 IF NOT PROBEW
1495  2     (OWN_STORAGE [ACCESS_MODE],
1496  2      %REF ($BYTEOFFSET (SHL$B_SHL_SIZE) ),
1497  2      SHL [SHL$L_BASEVA])
1498  2 THEN RETURN SS$_ACCVIO;
1499  2
1500  2 SHL [SHL$L_BASEVA] = .BASE_ADDRESS;
1501  2 IF .SHL [SHL$B_SHL_SIZE] EQL 0
1502  2 THEN SHL [SHL$B_SHL_SIZE] = SHL$C_OLD_SHL_SIZE;
1503  2
1504  2 ! Insure that the caller can write into the link longword in the fixup
1505  2 ! vector and then add this fixup vector to the front of the list.
1506  2
1507  2 IF NOT PROBEW (OWN_STORAGE [ACCESS_MODE], %REF (4), IAF [IAF$L_FIXUPLNK])
1508  2 THEN RETURN SS$_ACCVIO;
1509  2
1510  2 IAF [IAF$L_FIXUPLNK] = .CTL$GL_FIXUPLNK;
1511  2 CTL$GL_FIXUPLNK = IAF;
1512  2 RETURN SS$_NORMAL
1513  2
1514  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

```
;  1294          1515  1 %SBTTL 'NEXT_GBL_SEC_NAME - Update Global Section Name Suffix'
;  1295          1516  1
;  1296          1517  1 ROUTINE NEXT_GBL_SEC_NAME (STRING_POINTER) : NOVALUE =
;  1297          1518  1
;  1298          1519  1 !+
;  1299          1520  1 ! Functional Description:
;  1300          1521  1 !
;  1301          1522  1 !     This routine performs string arithmetic on a global section name suffix of
;  1302          1523  1 !     the form _00n by adding one to the suffix and performing the appropriate
;  1303          1524  1 !     carries. No error checking for illegal string contents or carries beyond
;  1304          1525  1 !     _999 is performed. (The suffix _999 is transformed to _000.)
;  1305          1526  1 !
;  1306          1527  1 ! Calling Sequence:
;  1307          1528  1 !
;  1308          1529  1 !     NEXT_GBL_SEC_NAME (STRING_POINTER)
;  1309          1530  1 !
;  1310          1531  1 ! Formal Parameters:
;  1311          1532  1 !
;  1312          1533  1 !     STRING_POINTER - Address of counted ASCII string containing global
;  1313          1534  1 !         section name
;  1314          1535  1 !-
;  1315          1536  1
;  1316          1537  2 BEGIN
;  1317          1538  2
;  1318          1539  2 ! Two synonyms are created. The string called NAME represents the entire
;  1319          1540  2 ! counted ASCII string for the global section name. Recall that the count field
;  1320          1541  2 ! does not contain the four character suffix. The substring SUFFIX represents
;  1321          1542  2 ! the final three characters in the global section name, usually 00n.
;  1322          1543  2
;  1323          1544  2 BIND
;  1324          1545  2     NAME   = .STRING_POINTER            : VECTOR [,BYTE],
;  1325          1546  2     SUFFIX = NAME [.NAME [0] + 2]       : VECTOR [3,BYTE];
;  1326          1547  2
;  1327          1548  2 LOCAL
;  1328          1549  2     DIGIT : INITIAL (2),
;  1329          1550  2     CARRY : INITIAL (TRUE);
;  1330          1551  2
;  1331          1552  2 WHILE .CARRY DO
;  1332          1553  2     IF .SUFFIX [.DIGIT] NEQ %C'9'
;  1333          1554  2     THEN
;  1334          1555  3         BEGIN
;  1335          1556  3         SUFFIX [.DIGIT] = .SUFFIX [.DIGIT] + 1;
;  1336          1557  3         CARRY = FALSE;
;  1337          1558  3         END
;  1338          1559  2     ELSE
;  1339          1560  3         BEGIN
;  1340          1561  3         SUFFIX [.DIGIT] = %C'0';
;  1341          1562  3         DIGIT = .DIGIT - 1;
;  1342          1563  3         IF .DIGIT LSS 0
;  1343          1564  3         THEN CARRY = FALSE;
;  1344          1565  2         END;
;  1345          1566  2
;  1346          1567  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
```

```
;  1348          1568  1 %SBTTL 'LOAD_SEQ_IMAGE - Load an image from a sequential device'
;  1349          1569  1
;  1350          1570  1 ROUTINE LOAD_SEQ_IMAGE ( ICB : REF $BBLOCK ) =
;  1351          1571  1
;  1352          1572  1 !+
;  1353          1573  1 ! Functional Description:
;  1354          1574  1 !
;  1355          1575  1 ! This routine loads all private non-dzro sections of an image from a
;  1356          1576  1 ! sequential device.
;  1357          1577  1 !
;  1358          1578  1 ! Calling Sequence:
;  1359          1579  1 !
;  1360          1580  1 !    LOAD_SEQ_IMAGE (ICB pointer)
;  1361          1581  1 !
;  1362          1582  1 ! Formal Parameters:
;  1363          1583  1 !
;  1364          1584  1 !       ICB - pointer to the current ICB
;  1365          1585  1 !-
;  1366          1586  1
;  1367          1587  2 BEGIN
;  1368          1588  2
;  1369          1589  2 ! This routine expects a vector of image section entries.  Each entry consists
;  1370          1590  2 ! of two longwords.  The first is a page count.  The second is a starting
;  1371          1591  2 ! address with the low order bit used to indicate writeable address space.
;  1372          1592  2 ! The read is done one page at a time under the assumption of one page records.
;  1373          1593  2 !
;  1374          1594  2 ! This routine also zeros the debug symbol table entry in the image header to
;  1375          1595  2 ! prevent an attempt by DEBUG to read this table.
;  1376          1596  2
;  1377          1597  2 LOCAL
;  1378          1598  2     SEQ_ISDS    : REF VECTOR [],
;  1379          1599  2     INADR       : VECTOR [2],
;  1380          1600  2     IHD         : REF $BBLOCK,
;  1381          1601  2     IOSB        : VECTOR [4,WORD],
;  1382          1602  2     NEXT_ADDRESS,
;  1383          1603  2     STATUS;
;  1384          1604  2
;  1385          1605  2 BIND
;  1386          1606  2     IHD_CTX = .ICB [ICB$L_CONTEXT] : $BBLOCK;
;  1387          1607  2
;  1388          1608  2 SEQ_ISDS = .OWN_STORAGE [SEQ_LOAD_ISDS];
;  1389          1609  2
;  1390          1610  2 IHD = .IHD_CTX [CTX_L_IHDBUF];
;  1391          1611  2 IHD [IHD$W_SYMDBGOFF] = 0;
;  1392          1612  2
;  1393          1613  2 ! Process ISDs until done
;  1394          1614  2
;  1395          1615  2 UNTIL .SEQ_ISDS [0] EQL 0
;  1396          1616  2 DO
;  1397          1617  3     BEGIN
;  1398          1618  3     INADR [0] = .SEQ_ISDS [1];
;  1399          1619  3     INADR [1] = .INADR [0] + (.SEQ_ISDS [0] * BYTES_PER_PAGE) - 1;
;  1400          1620  3
;  1401          1621  3     ! Create adrress space
;  1402          1622  3
;  1403        P 1623  3     STATUS = $CRETVA (
;  1404        P 1624  3         INADR = INADR,
```

```
; 1405          1625  3          ACMODE = PSL$C_USER );
; 1406          1626  3      IF NOT .STATUS THEN RETURN .STATUS;
; 1407          1627  3
; 1408          1628  3      ! Read the image one page at a t4me
; 1409          1629  3
; 1410          1630  3      NEXT_ADDRESS = .INADR [0];
; 1411          1631  3      INCR I FROM 1 TO .SEQ_ISDS [0]
; 1412          1632  3      DO
; 1413          1633  4          BEGIN
; 1414        P 1634  4          STATUS = $QIOW (
; 1415        P 1635  4              EFN = EXE$C_SYSEFN,
; 1416        P 1636  4              CHAN = .ICB [ICB$W_CHAN],
; 1417        P 1637  4              FUNC = IO$_READVBLK,
; 1418        P 1638  4              IOSB = IOSB,
; 1419        P 1639  4              P1 = .NEXT_ADDRESS,
; 1420          1640  4              P2 = BYTES_PER_PAGE );
; 1421          1641  4          IF NOT .STATUS THEN RETURN .STATUS;
; 1422          1642  4          IF NOT .IOSB [0] THEN RETURN .IOSB [0];
; 1423          1643  4          NEXT_ADDRESS = .NEXT_ADDRESS + BYTES_PER_PAGE;
; 1424          1644  3          END;
; 1425          1645  3
; 1426          1646  3      ! Is it a message section (can't be vector since not installed)
; 1427          1647  3
; 1428          1648  3      IF (.SEQ_ISDS [2] AND ISD$M_VECTOR) NEQ 0
; 1429          1649  3      THEN
; 1430          1650  4          BEGIN
; 1431          1651  4          STATUS = ADD_PRIVILEGED_VECTOR (INADR, ICB);
; 1432          1652  4          IF NOT .STATUS THEN RETURN .STATUS
; 1433          1653  3          END;
; 1434          1654  3
; 1435          1655  3      ! Is it a fixup section
; 1436          1656  3
; 1437          1657  3      IF (.SEQ_ISDS [2] AND ISD$M_FIXUPVEC) NEQ 0
; 1438          1658  3      THEN
; 1439          1659  4          BEGIN
; 1440          1660  4          STATUS = ADD_FIXUP_VECTOR (
; 1441          1661  4              INADR,
; 1442          1662  4              .ICB [ICB$L_BASE_ADDRESS]);
; 1443          1663  4          IF NOT .STATUS THEN RETURN .STATUS
; 1444          1664  3          END;
; 1445          1665  3
; 1446          1666  3      ! Check to see if it should be left writeable
; 1447          1667  3
; 1448          1668  3      IF (.SEQ_ISDS [2] AND ISD$M_WRT) EQL 0
; 1449          1669  3      THEN
; 1450          1670  4          BEGIN
; 1451        P 1671  4          STATUS = $SETPRT (
; 1452        P 1672  4              INADR = INADR,
; 1453        P 1673  4              ACMODE = PSL$C_USER,
; 1454          1674  4              PROT = PRT$C_UR );
; 1455          1675  4          IF NOT .STATUS THEN RETURN .STATUS;
; 1456          1676  3          END;
; 1457          1677  3
; 1458          1678  3      ! Advance pointer to next ISD description
; 1459          1679  3
; 1460          1680  3      SEQ_ISDS = SEQ_ISDS [3];
; 1461          1681  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 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
```

```
; 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