J 4

```
   AAAAAAAAA     NNN         NNN     AAAAAAAAA     LLL              YYY         YYY  ZZZZZZZZZZZZZZZ
   AAAAAAAAA     NNN         NNN     AAAAAAAAA     LLL              YYY         YYY  ZZZZZZZZZZZZZZZ
   AAAAAAAAA     NNN         NNN     AAAAAAAAA     LLL              YYY         YYY  ZZZZZZZZZZZZZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLL              YYY         YYY              ZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLL              YYY         YYY              ZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLL              YYY         YYY              ZZZ
AAA         AAA  NNNNNN      NNN  AAA         AAA  LLL                 YYY   YYY              ZZZ
AAA         AAA  NNNNNN      NNN  AAA         AAA  LLL                 YYY   YYY              ZZZ
AAA         AAA  NNNNNN      NNN  AAA         AAA  LLL                 YYY   YYY              ZZZ
AAA         AAA  NNN   NNN   NNN  AAA         AAA  LLL                    YYY              ZZZ
AAA         AAA  NNN   NNN   NNN  AAA         AAA  LLL                    YYY              ZZZ
AAA         AAA  NNN   NNN   NNN  AAA         AAA  LLL                    YYY              ZZZ
AAAAAAAAAAAAAAA  NNN      NNNNNN  AAAAAAAAAAAAAAA  LLL                    YYY           ZZZ
AAAAAAAAAAAAAAA  NNN      NNNNNN  AAAAAAAAAAAAAAA  LLL                    YYY           ZZZ
AAAAAAAAAAAAAAA  NNN      NNNNNN  AAAAAAAAAAAAAAA  LLL                    YYY           ZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLL                    YYY        ZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLL                    YYY        ZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLL                    YYY        ZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLLLLLLLLLLLLLL        YYY        ZZZZZZZZZZZZZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLLLLLLLLLLLLLL        YYY        ZZZZZZZZZZZZZZZ
AAA         AAA  NNN         NNN  AAA         AAA  LLLLLLLLLLLLLLL        YYY        ZZZZZZZZZZZZZZZ
```

```
**FILE**ID**EXEINPUT                                J 14
```

```
EEEEEEEEEE  XX      XX  EEEEEEEEEE    IIIIII    NN      NN  PPPPPPPP    UU      UU  TTTTTTTTTT
EEEEEEEEEE  XX      XX  EEEEEEEEEE    IIIIII    NN      NN  PPPPPPPP    UU      UU  TTTTTTTTTT
EE          XX      XX  EE              II      NN      NN  PP      PP  UU      UU      TT
EE          XX      XX  EE              II      NN      NN  PP      PP  UU      UU      TT
EE            XX  XX    EE              II      NNNN    NN  PP      PP  UU      UU      TT
EE            XX  XX    EE              II      NNNN    NN  PP      PP  UU      UU      TT
EEEEEEEE        XX      EEEEEEEE        II      NN  NN  NN  PPPPPPPP    UU      UU      TT
EEEEEEEE        XX      EEEEEEEE        II      NN  NN  NN  PPPPPPPP    UU      UU      TT
EE            XX  XX    EE              II      NN    NNNN  PP          UU      UU      TT
EE            XX  XX    EE              II      NN    NNNN  PP          UU      UU      TT
EE          XX      XX  EE              II      NN      NN  PP          UU      UU      TT          ....
EE          XX      XX  EE              II      NN      NN  PP          UU      UU      TT          ....
EEEEEEEEEE  XX      XX  EEEEEEEEEE    IIIIII    NN      NN  PP          UUUUUUUUUU      TT          ....
EEEEEEEEEE  XX      XX  EEEEEEEEEE    IIIIII    NN      NN  PP          UUUUUUUUUU      TT          ....

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
```

```
0 %title 'EXEINPUT - Handle Image Files & Libraries'
0       module exeinput (
1               ident='V04-000') = begin
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! Facility:     VAX/VMS Analyze Facility, Handle Image Files
1 !
1 ! Abstract:     This module is responsible for handling file specs from
1 !               the command line, and reading data from image files.
1 !
1 !
1 ! Environment:
1 !
1 ! Author: Paul C. Anagnostopoulos, Creation Date: 31 March 1981
1 !
1 ! Modified By:
1 !
1 !       V03-007 ROP0009         Robert Posniak          15-JUN-1984
1 !               Clear image buffer valid flag when analyzing
1 !               more than one image.
1 !
1 !       V03-006 BLS0286         Benn Schreiber          20-MAR-1984
1 !               Correct 005.
1 !
1 !       V03-005 LJA0112         Laurie J. Anderson      21-Feb-1984
1 !               Add new related file parsing arguments to LIB$FIND_FILE
1 !               to make search lists behave properly.
1 !
1 !       V03-004 LJA0105         Laurie J. Anderson      25-Jan-1984
1 !               Changes due because now using the new image activator
1 !               decode routines.  Now, open the image file through RMS
1 !               with the UFO option bit set, and do not bother to connect
```

```
1 !                 stream.  Will be using QIO's, because the decode routines
1 !                 do, too.  Add routines to read the image header and isd's.
1 !
1 !         V03-003 PCA1011         Paul C. Anagnostopoulos  1-Apr-1983
1 !                 Change the message prefix to ANLOBJ$_ to ensure that
1 !                 message symbols are unique across all ANALYZEs.  This
1 !                 is necessitated by the new merged message files.
1 !
1 !         V03-002 PCA0020         Paul Anagnostopoulos    24-Mar-1982
1 !                 Signal errors using the correct STV value.
1 !
1 !         V03-001 PCA0014         Paul Anagnostopoulos    22-Mar-1982
1 !                 Use the resultant spec rather than the wildcard spec
1 !                 when complaining about a file to be analyzed.
1 !--
```

```
%sbttl 'Module Declarations'
!
! Libraries and Requires:
!

library 'starlet';
require 'objexereq';

!
! Table of Contents:
!

forward routine
        anl$open_next_image_file,
        anl$get_image_block,
        anl$map_fixup_section,
        anl$get_image_header,
        anl$get_isd ;

!
! External References:
!

external routine
        anl$image_positionals,
        cli$get_value: addressing_mode(general),
        img$decode_ihd: addressing_mode(general),
        img$get_next_isd: addressing_mode(general),
        lib$find_file: addressing_mode(general),
        lib$free_vm: addressing_mode(general),
        lib$get_vm: addressing_mode(general),
        str$trim: addressing_mode(general);

!
! Own Variables:
!
! The following data structures are used to access and read blocks from
! the file we are to analyze.

own
        own_described_buffer(resultant_spec,nam$c_maxrss),
        image_fab: $fab(fop=ufo),       ! FAB for RMS UFO $OPEN of image file
        image_buffer: block[512,byte],  ! contains decoded IHD
        isd_buf: block[512,byte],       ! Contains ISD block
        hdr_blk: block[512,byte],       ! contains block of image header
        blk_buf: block[512,byte],       ! contains block read by get_image_block
        hdrver,                         ! Header version
        vbn: long initial(1),           ! VBN of image fourd in hdr_blk
        offset,                         ! offset to the ISD's returned here
        chan : long initial( 0 ),       ! Channel assigned to the image file
        image_buf_valid : long initial(0); ! indicates if the image buffer is valid
```

```
1 %sbttl 'ANL$OPEN_NEXT_IMAGE_FILE - Right'
1 !++
1 ! Functional Description:
1 !       This routine is called to open the next image file we are to analyze.
1 !       It handles multiple file specs and wildcarding.  It does not handle
1 !       sharable image libraries.
1 !
1 ! Formal Parameters:
1 !       opened_spec     Address of descriptor of buffer in which to return
1 !                       the spec of the file we open.  We set the length.
1 !
1 ! Implicit Inputs:
1 !       global data
1 !
1 ! Implicit Outputs:
1 !       global data
1 !
1 ! Returned Value:
1 !       True if there is another image file, false otherwise.
1 !
1 ! Side Effects:
1 !
1 !--
1
1
2 global routine anl$open_next_image_file(opened_spec) = begin
2
2 bind
2       opened_spec_dsc = .opened_spec: descriptor;
2
2 own
2       own_described_buffer(wildcard_spec,nam$c_maxrss),
2       wildcard_context: long initial(0),
2       get_new_spec: long initial(true);
2
2 local
2       stv: long,
2       status: long;
2
2 !
2 ! First clear flag to invalidate old image header
2 !
2
2 image_buf_valid = 0;
2
2 ! If the wildcard context is zero (it means this is the first call), or
2 ! have we finished with a file spec on the previous call, we must obtain
2 ! the next file spec from the command line.
2
3 if .get_new_spec then (
3       wildcard_spec[len] = nam$c_maxrss;
3       status = cli$get_value(describe('file_specs'),wildcard_spec);
3
3       ! If there are no more specs, we are all done.
3
3       if not .status then
3               return false;
3          str$trim(wildcard_spec,wildcard_spec,wildcard_spec);
3
3          ! Call a routine to process any positional qualifiers for this spec.
3          ! We don't know how to do that.
3
3          anl$image_positionals();
2 );
2
2 ! On the other hand, if the previous call done is true, we may have just
2 ! finished processing a file.  Better deassign channel.
2
3 if .chan nequ 0 then (
3          status = $dassgn(chan=.chan);
3          check (.status, anlobj$_closein,1,resultant_spec,.status);
3          chan = 0;
2 );
```

```
;  201          0633  2 ! We have obtained a wildcard spec from the file parameter.  We also have
;  202          0634  2 ! all the positional qualifiers associated with it.
;  203          0635  2
;  204          0636  2 ! Now we need to find the next file that matches the current wildcard spec.
;  205          0637  2
;  206          0638  2 resultant_spec[len] = nam$c_maxrss;
;  207          0639  2 status = lib$find_file(wildcard_spec,resultant_spec,
;  208          0640  2                 wildcard_context,describe('.EXE'),
;  209          0641  2                 0,stv,%ref(2));
;  210          0642  2 str$trim(resultant_spec,resultant_spec,resultant_spec);
;  211          0643  2
;  212          0644  2 ! If we failed to find a file, then reset the wildcard context and call
;  213          0645  2 ! ourselves recursively to process the next file spec.  Also give an
;  214          0646  2 ! error, unless we just plain ran out of files.
;  215          0647  2
;  216          0648  3 if not .status then (
;  217          0649  3         if .status nequ rms$_nmf then
;  218          0650  3                 signal (anlobj$_openin,1,resultant_spec,.stv);
;  219          0651  3         get_new_spec = true;
;  220          0652  3         return anl$open_next_image_file(opened_spec_dsc);
;  221          0653  2 );
;  222          0654  2
;  223          0655  2 ! Hey, we got a file spec.  Open the file and connect the RAB.
;  224          0656  2
;  225          0657  2 get_new_spec = false;
;  226          0658  2
;  227          0659  2 image_fab[fab$b_fns] = .resultant_spec[len];
;  228          0660  2 image_fab[fab$l_fna] = .resultant_spec[ptr];
;  229          0661  2 status = $open(fab=image_fab);
;  230          0662  2 check (.status, anlobj$_openin,1,resultant_spec,.status,.image_fab[fab$l_stv]);
;  231          0663  2
;  232          0664  2 ! If the open failed, then we need to recurse to try the next file.
;  233          0665  2
;  234          0666  2 if not .status then
;  235          0667  2         return anl$open_next_image_file(opened_spec_dsc);
;  236          0668  2
;  237          0669  2 ! Finally, we have to return the resultant file spec to the caller.
;  238          0670  2
;  239          0671  2 opened_spec_dsc[len] = .resultant_spec[len];
;  240          0672  2 ch$move(.resultant_spec[len],.resultant_spec[ptr], .opened_spec_dsc[ptr]);
;  241          0673  2
;  242          0674  2 chan = .image_fab[fab$l_stv];
;  243          0675  2
;  244          0676  2 return true;
;  245          0677  2
;  246          0678  1 end;


                                                                          .TITLE  EXEINPUT EXEINPUT - Handle Image Files & Librar
;                                                                                          ies
                                                                          .IDENT  \V04-000\

                                                                          .PSECT  $PLIT$,NOWRT,NOEXE,2

                  73  63  65  70  73  5F  65  6C  69  66  00000 P.AAB:   .ASCII  \file_specs\                                      ;
                                                          0000A           .BLKB   2
                                                0000000A  0000C P.AAA:    .LONG   10                                                ;
```

```
             00000000  00010           .ADDRESS P.AAB                      ;
      45  58  45  2E   00014 P.AAD:    .ASCII   \.EXE\                     ;
             00000004  00018 P.AAC:    .LONG    4                          ;
            00000000'  0001C           .ADDRESS P.AAD                      ;

                               .PSECT  $OWN$,NOEXE,2

             000000FF  00000 RESULTANT_SPEC:
                               .LONG    255
            00000000'  00004           .ADDRESS RESULTANT_SPEC+8           ;
                       00008           .BLKB    255
                       00107           .BLKB    1
                   03  00108 IMAGE_FAB:
                               .BYTE    3                                  ;
                   50  00109           .BYTE    80                         ;
                 0000  0010A           .WORD    0                          ;
             00020000  0010C           .LONG    131072                     ;
             00000000  00110           .LONG    0                          ;
             00000000  00114           .LONG    0                          ;
             00000000  00118           .LONG    0                          ;
                 0000  0011C           .WORD    0                          ;
                   02  0011E           .BYTE    2                          ;
                   00  0011F           .BYTE    0                          ;
             00000000  00120           .LONG    0                          ;
                   00  00124           .BYTE    0                          ;
                   00  00125           .BYTE    0                          ;
                   00  00126           .BYTE    0                          ;
                   02  00127           .BYTE    2                          ;
             00000000  00128           .LONG    0                          ;
             00000000  0012C           .LONG    0                          ;
             00000000  00130           .LONG    0                          ;
             00000000  00134           .LONG    0                          ;
             00000000  00138           .LONG    0                          ;
                   00  0013C           .BYTE    0                          ;
                   00  0013D           .BYTE    0                          ;
                 0000  0013E           .WORD    0                          ;
             00000000  00140           .LONG    0                          ;
                 0000  00144           .WORD    0                          ;
                   00  00146           .BYTE    0                          ;
                   00  00147           .BYTE    0                          ;
             00000000  00148           .LONG    0                          ;
             00000000  0014C           .LONG    0                          ;
                 0000  00150           .WORD    0                          ;
                   00  00152           .BYTE    0                          ;
                   00  00153           .BYTE    0                          ;
             00000000  00154           .LONG    0                          ;
                       00158 IMAGE_BUFFER:
                               .BLKB    512
                       00358 ISD_BUF:.BLKB    512
                       00558 HDR_BLK:.BLKB    512
                       00758 BLK_BUF:.BLKB    512
                       00958 HDRVER: .BLKB    4
             00000001  0095C VBN:    .LONG    1                            ;
                       00960 OFFSET: .BLKB    4
             00000000  00964 CHAN:   .LONG    0                            ;
             00000000  00968 IMAGE_BUF_VALID:
                               .LONG    0                                  ;
```

```
000000FF  0096C WILDCARD_SPEC:
                        .LONG    255                                            ;
00000000' 00970         .ADDRESS WILDCARD_SPEC+8                                ;
          00974         .BLKB    255
          00A73         .BLKB    1
00000000  00A74 WILDCARD_CONTEXT:
                        .LONG    0                                              ;
00000001  00A78 GET_NEW_SPEC:
                        .LONG    1                                              ;

                        .EXTRN   ANLOBJ$_OK, ANLOBJ$_ANYTHING
                        .EXTRN   ANLOBJ$_DATATYPE
                        .EXTRN   ANLOBJ$_ERRORCOUNT
                        .EXTRN   ANLOBJ$_ERRORNONE
                        .EXTRN   ANLOBJ$_ERRORS, ANLOBJ$_EXEFIXA
                        .EXTRN   ANLOBJ$_EXEFIXAIMAGE
                        .EXTRN   ANLOBJ$_EXEFIXALINE
                        .EXTRN   ANLOBJ$_EXEFIXCOUNT
                        .EXTRN   ANLOBJ$_EXEFIXEXTRA
                        .EXTRN   ANLOBJ$_EXEFIXFIXED
                        .EXTRN   ANLOBJ$_EXEFIXFLAGS
                        .EXTRN   ANLOBJ$_EXEFIXG
                        .EXTRN   ANLOBJ$_EXEFIXGIMAGE
                        .EXTRN   ANLOBJ$_EXEFIXGLINE
                        .EXTRN   ANLOBJ$_EXEFIXLIST
                        .EXTRN   ANLOBJ$_EXEFIXNAME
                        .EXTRN   ANLOBJ$_EXEFIXNAMEO
                        .EXTRN   ANLOBJ$_EXEFIXP
                        .EXTRN   ANLOBJ$_EXEFIXPSECT
                        .EXTRN   ANLOBJ$_EXEFIXUP
                        .EXTRN   ANLOBJ$_EXEFIXUPNONE
                        .EXTRN   ANLOBJ$_EXEGST, ANLOBJ$_EXEHDR
                        .EXTRN   ANLOBJ$_EXEHDRACTIVE
                        .EXTRN   ANLOBJ$_EXEHDRBLKCOUNT
                        .EXTRN   ANLOBJ$_EXEHDRCHANCOUNT
                        .EXTRN   ANLOBJ$_EXEHDRCHANDEF
                        .EXTRN   ANLOBJ$_EXEHDRDECECO
                        .EXTRN   ANLOBJ$_EXEHDRDMT
                        .EXTRN   ANLOBJ$_EXEHDRDST
                        .EXTRN   ANLOBJ$_EXEHDRFILEID
                        .EXTRN   ANLOBJ$_EXEHDRFIXED
                        .EXTRN   ANLOBJ$_EXEHDRFLAGS
                        .EXTRN   ANLOBJ$_EXEHDRGBLIDENT
                        .EXTRN   ANLOBJ$_EXEHDRGST
                        .EXTRN   ANLOBJ$_EXEHDRIDENT
                        .EXTRN   ANLOBJ$_EXEHDRIMAGEID
                        .EXTRN   ANLOBJ$_EXEHDRISD
                        .EXTRN   ANLOBJ$_EXEHDRISDBASE
                        .EXTRN   ANLOBJ$_EXEHDRISDCOUNT
                        .EXTRN   ANLOBJ$_EXEHDRISDFLAGS
                        .EXTRN   ANLOBJ$_EXEHDRISDGBLNAM
                        .EXTRN   ANLOBJ$_EXEHDRISDNUM
                        .EXTRN   ANLOBJ$_EXEHDRISDPFCDEF
                        .EXTRN   ANLOBJ$_EXEHDRISDPFCSIZ
                        .EXTRN   ANLOBJ$_EXEHDRISDTYPE
                        .EXTRN   ANLOBJ$_EXEHDRISDVBN
                        .EXTRN   ANLOBJ$_EXEHDRLINKID
```

```
        .EXTRN  ANLOBJ$_EXEHDRMATCH
        .EXTRN  ANLOBJ$_EXEHDRNAME
        .EXTRN  ANLOBJ$_EXEHDRNOPATCH
        .EXTRN  ANLOBJ$_EXEHDRPAGECOUNT
        .EXTRN  ANLOBJ$_EXEHDRPAGEDEF
        .EXTRN  ANLOBJ$_EXEHDRPATCH
        .EXTRN  ANLOBJ$_EXEHDRPATCHDATE
        .EXTRN  ANLOBJ$_EXEHDRPRIV
        .EXTRN  ANLOBJ$_EXEHDRROPATCH
        .EXTRN  ANLOBJ$_EXEHDRRWPATCH
        .EXTRN  ANLOBJ$_EXEHDRSYMDBG
        .EXTRN  ANLOBJ$_EXEHDRSYSVER
        .EXTRN  ANLOBJ$_EXEHDRTEXTVBN
        .EXTRN  ANLOBJ$_EXEHDRTIME
        .EXTRN  ANLOBJ$_EXEHDRTYPEEXE
        .EXTRN  ANLOBJ$_EXEHDRTYPELIM
        .EXTRN  ANLOBJ$_EXEHDRUSERECO
        .EXTRN  ANLOBJ$_EXEHDRXFER1
        .EXTRN  ANLOBJ$_EXEHDRXFER2
        .EXTRN  ANLOBJ$_EXEHDRXFER3
        .EXTRN  ANLOBJ$_EXEHEADING
        .EXTRN  ANLOBJ$_EXEPATCH
        .EXTRN  ANLOBJ$_FLAG, ANLOBJ$_HEXDATA
        .EXTRN  ANLOBJ$_HEXHEADING1
        .EXTRN  ANLOBJ$_HEXHEADING2
        .EXTRN  ANLOBJ$_INDMSGSEC
        .EXTRN  ANLOBJ$_INTERACT
        .EXTRN  ANLOBJ$_MASK, ANLOBJ$_OBJCPRREC
        .EXTRN  ANLOBJ$_OBJDBGREC
        .EXTRN  ANLOBJ$_OBJENV, ANLOBJ$_OBJEOMFLAGS
        .EXTRN  ANLOBJ$_OBJEOMREC
        .EXTRN  ANLOBJ$_OBJEOMSEVABT
        .EXTRN  ANLOBJ$_OBJEOMSEVERR
        .EXTRN  ANLOBJ$_OBJEOMSEVIGN
        .EXTRN  ANLOBJ$_OBJEOMSEVRES
        .EXTRN  ANLOBJ$_OBJEOMSEVSUC
        .EXTRN  ANLOBJ$_OBJEOMSEVWRN
        .EXTRN  ANLOBJ$_OBJEOMWREC
        .EXTRN  ANLOBJ$_OBJFADPASSMECH
        .EXTRN  ANLOBJ$_OBJGSDENV
        .EXTRN  ANLOBJ$_OBJGSDENVFLAGS
        .EXTRN  ANLOBJ$_OBJGSDENVPAR
        .EXTRN  ANLOBJ$_OBJGSDEPM
        .EXTRN  ANLOBJ$_OBJGSDEPMW
        .EXTRN  ANLOBJ$_OBJGSDIDC
        .EXTRN  ANLOBJ$_OBJGSDIDCENT
        .EXTRN  ANLOBJ$_OBJGSDIDCFLAGS
        .EXTRN  ANLOBJ$_OBJGSDIDCMATCH
        .EXTRN  ANLOBJ$_OBJGSDIDCOBJ
        .EXTRN  ANLOBJ$_OBJGSDIDCVALA
        .EXTRN  ANLOBJ$_OBJGSDIDCVALB
        .EXTRN  ANLOBJ$_OBJGSDLEPM
        .EXTRN  ANLOBJ$_OBJGSDLPRO
        .EXTRN  ANLOBJ$_OBJGSDLSY
        .EXTRN  ANLOBJ$_OBJGSDPRO
        .EXTRN  ANLOBJ$_OBJGSDPROW
        .EXTRN  ANLOBJ$_OBJGSDPSC
```

```
.EXTRN  ANLOBJ$_OBJGSDPSCALIGN
.EXTRN  ANLOBJ$_OBJGSDPSCALLOC
.EXTRN  ANLOBJ$_OBJGSDPSCBASE
.EXTRN  ANLOBJ$_OBJGSDPSCFLAGS
.EXTRN  ANLOBJ$_OBJGSDREC
.EXTRN  ANLOBJ$_OBJGSDPSC
.EXTRN  ANLOBJ$_OBJGSDSYM
.EXTRN  ANLOBJ$_OBJGSDSYMW
.EXTRN  ANLOBJ$_OBJGTXREC
.EXTRN  ANLOBJ$_OBJHDRIGNREC
.EXTRN  ANLOBJ$_OBJHEADING
.EXTRN  ANLOBJ$_OBJLITINDEX
.EXTRN  ANLOBJ$_OBJLNKREC
.EXTRN  ANLOBJ$_OBJLNMREC
.EXTRN  ANLOBJ$_OBJMHDCREATE
.EXTRN  ANLOBJ$_OBJMHDNAME
.EXTRN  ANLOBJ$_OBJMHDPATCH
.EXTRN  ANLOBJ$_OBJMHDREC
.EXTRN  ANLOBJ$_OBJMHDRECSIZ
.EXTRN  ANLOBJ$_OBJMHDSTRLVL
.EXTRN  ANLOBJ$_OBJMHDVERSION
.EXTRN  ANLOBJ$_OBJMTCCORRECT
.EXTRN  ANLOBJ$_OBJMTCINPUT
.EXTRN  ANLOBJ$_OBJMTCNAME
.EXTRN  ANLOBJ$_OBJMTCREC
.EXTRN  ANLOBJ$_OBJMTCSEQNUM
.EXTRN  ANLOBJ$_OBJMTCUIC
.EXTRN  ANLOBJ$_OBJMTCVERSION
.EXTRN  ANLOBJ$_OBJMTCWHEN
.EXTRN  ANLOBJ$_OBJPROARGCOUNT
.EXTRN  ANLOBJ$_OBJPROARGNUM
.EXTRN  ANLOBJ$_OBJPSECT
.EXTRN  ANLOBJ$_OBJSRCREC
.EXTRN  ANLOBJ$_OBJSTATHEADING1
.EXTRN  ANLOBJ$_OBJSTATHEADING2
.EXTRN  ANLOBJ$_OBJSTATLINE
.EXTRN  ANLOBJ$_OBJSTATTOTAL
.EXTRN  ANLOBJ$_OBJSYMBOL
.EXTRN  ANLOBJ$_OBJSYMFLAGS
.EXTRN  ANLOBJ$_OBJTIRARGINDEX
.EXTRN  ANLOBJ$_OBJTIRCMD
.EXTRN  ANLOBJ$_OBJTIRCMDSTK
.EXTRN  ANLOBJ$_OBJTBTREC
.EXTRN  ANLOBJ$_OBJTIRREC
.EXTRN  ANLOBJ$_OBJTIRSTOIM
.EXTRN  ANLOBJ$_OBJTIRVIELD
.EXTRN  ANLOBJ$_OBJTTLREC
.EXTRN  ANLOBJ$_OBJVALUE
.EXTRN  ANLOBJ$_OBJUVALUE
.EXTRN  ANLOBJ$_PROTECTION
.EXTRN  ANLOBJ$_SEVERITY
.EXTRN  ANLOBJ$_TEXT, ANLOBJ$_TEXTHDR
.EXTRN  ANLOBJ$_NOSUCHMOD
.EXTRN  ANLOBJ$_BADDATE
.EXTRN  ANLOBJ$_BADHDRBLKCOUNT
.EXTRN  ANLOBJ$_BADSEVERITY
.EXTRN  ANLOBJ$_BADSYM1ST
```

```
        .EXTRN  ANLOBJ$_BADSYMCHAR
        .EXTRN  ANLOBJ$_BADSYMLEN
        .EXTRN  ANLOBJ$_EXEBADFIXUPEND
        .EXTRN  ANLOBJ$_EXEBADFIXUPISD
        .EXTRN  ANLOBJ$_EXEBADFIXUPVBN
        .EXTRN  ANLOBJ$_EXEBADISDS1
        .EXTRN  ANLOBJ$_EXEBADISDTYPE
        .EXTRN  ANLOBJ$_EXEBADMATCH
        .EXTRN  ANLOBJ$_EXEBADPATCHLEN
        .EXTRN  ANLOBJ$_EXEBADOBJ
        .EXTRN  ANLOBJ$_EXEBADTYPE
        .EXTRN  ANLOBJ$_EXEBADXFERO
        .EXTRN  ANLOBJ$_EXEHDRISDLONG
        .EXTRN  ANLOBJ$_EXEHDRLONG
        .EXTRN  ANLOBJ$_EXEISDLENDZRO
        .EXTRN  ANLOBJ$_EXEISDLENGBL
        .EXTRN  ANLOBJ$_EXEISDLENPRIV
        .EXTRN  ANLOBJ$_EXENOTNATIVE
        .EXTRN  ANLOBJ$_EXTRABYTES
        .EXTRN  ANLOBJ$_FIELDFIT
        .EXTRN  ANLOBJ$_FLAGERROR
        .EXTRN  ANLOBJ$_NOTOK, ANLOBJ$_OBJBADIDCMATCH
        .EXTRN  ANLOBJ$_OBJBADNUM
        .EXTRN  ANLOBJ$_OBJBADPOP
        .EXTRN  ANLOBJ$_OBJBADPUSH
        .EXTRN  ANLOBJ$_OBJBADTYPE
        .EXTRN  ANLOBJ$_OBJBADVIELD
        .EXTRN  ANLOBJ$_OBJEOMBADSEV
        .EXTRN  ANLOBJ$_OBJEOMMISSING
        .EXTRN  ANLOBJ$_OBJFADBADAVC
        .EXTRN  ANLOBJ$_OBJFADBADRBC
        .EXTRN  ANLOBJ$_OBJGSDBADALIGN
        .EXTRN  ANLOBJ$_OBJGSDBADSUBTYP
        .EXTRN  ANLOBJ$_OBJHDRRES
        .EXTRN  ANLOBJ$_OBJMHDBADRECSIZ
        .EXTRN  ANLOBJ$_OBJMHDBADSTRLVL
        .EXTRN  ANLOBJ$_OBJMHDMISSING
        .EXTRN  ANLOBJ$_OBJNONTIRCMD
        .EXTRN  ANLOBJ$_OBJNOPSC
        .EXTRN  ANLOBJ$_OBJNULLREC
        .EXTRN  ANLOBJ$_OBJPOSPACE
        .EXTRN  ANLOBJ$_OBJPROMINMAX
        .EXTRN  ANLOBJ$_OBJPSCABSLEN
        .EXTRN  ANLOBJ$_OBJRECTOOBIG
        .EXTRN  ANLOBJ$_OBJTIRRES
        .EXTRN  ANLOBJ$_OBJUNDEFENV
        .EXTRN  ANLOBJ$_OBJUNDEFLIT
        .EXTRN  ANLOBJ$_OBJUNDEFPSC
        .EXTRN  ANALYZE$_FACILITY
        .EXTRN  ANL$IMAGE_POSITIONALS
        .EXTRN  CLI$GET_VALUE, IMG$DECODE_IHD
        .EXTRN  IMG$GET_NEXT_ISD
        .EXTRN  LIB$FIND_FILE, LIB$FREE_VM
        .EXTRN  LIB$GET_VM, STR$TRIM
        .EXTRN  SYS$DASSGN, SYS$OPEN

        .PSECT  $CODE$,NOWRT,2
```

```
                                   01FC 00000          .ENTRY  ANL$OPEN_NEXT_IMAGE_FILE, Save R2,R3,R4,R5,-; 0585
                                                               R6,R7,R8                                    ;
                   58 00000000G  00   9E 00002          MOVAB   STR$TRIM, R8                                ;
                   57 00000000G  00   9E 00009          MOVAB   LIB$SIGNAL, R7                              ;
                   56     0000'  CF   9E 00010          MOVAB   RESULTANT_SPEC, R6                          ;
                   5E            08   C2 00015          SUBL2   #8, SP                                      ;
                   52        04  AC   D0 00018          MOVL    OPENED_SPEC, R2                             ; 0588
                           0968  C6   D4 0001C          CLRL    IMAGE_BUF_VALID                             ; 0603
                   32      0A78  C6   E9 00020          BLBC    GET_NEW_SPEC, 2$                            ; 0609
          096C     C6        FF  8F   9B 00025          MOVZBW  #255, WILDCARD_SPEC                         ; 0610
                           096C  C6   9F 0002B          PUSHAB  WILDCARD_SPEC                               ; 0611
                           0000' CF   9F 0002F          PUSHAB  P.AAA                                       ;
     00000000G     00            02   FB 00033          CALLS   #2, CLI$GET_VALUE                           ;
                   53            50   D0 0003A          MOVL    R0, STATUS                                  ;
                   03            53   E8 0003D          BLBS    STATUS, 1$                                  ; 0615
                               00E1   31 00040          BRW     9$                                          ;
                           096C  C6   9F 00043 1$:      PUSHAB  WILDCARD_SPEC                               ; 0617
                           096C  C6   9F 00047          PUSHAB  WILDCARD_SPEC                               ;
                           096C  C6   9F 0004B          PUSHAB  WILDCARD_SPEC                               ;
                   68            03   FB 0004F          CALLS   #3, STR$TRIM                                ;
         0000G     CF            00   FB 00052          CALLS   #0, ANL$IMAGE_POSITIONALS                   ; 0622
                   50      0964  C6   D0 00057 2$:      MOVL    CHAN, R0                                    ; 0628
                                 22   13 0005C          BEQL    4$                                          ;
                                 50   DD 0005E          PUSHL   R0                                          ; 0629
     00000000G     00            01   FB 00060          CALLS   #1, SYS$DASSGN                              ;
                   53            50   D0 00067          MOVL    R0, STATUS                                  ;
                   0F            53   E8 0006A          BLBS    STATUS, 3$                                  ; 0630
                                 53   DD 0006D          PUSHL   STATUS                                      ;
                                 56   DD 0006F          PUSHL   R6                                          ;
                                 01   DD 00071          PUSHL   #1                                          ;
                       00B11052  8F   DD 00073          PUSHL   #11604050                                   ;
                   67            04   FB 00079          CALLS   #4, LIB$SIGNAL                              ;
                           0964  C6   D4 0007C 3$:      CLRL    CHAN                                        ; 0631
                   66        FF  8F   9B 00080 4$:      MOVZBW  #255, RESULTANT_SPEC                        ; 0638
                   6E            02   D0 00084          MOVL    #2, (SP)                                    ; 0641
                                 5E   DD 00087          PUSHL   SP                                          ;
                             08  AE   9F 00089          PUSHAB  STV                                         ; 0639
                                 7E   D4 0008C          CLRL    -(SP)                                       ;
                           0000' CF   9F 0008E          PUSHAB  P.AAC                                       ; 0640
                           0A74  C6   9F 00092          PUSHAB  WILDCARD_CONTEXT                            ; 0639
                                 56   DD 00096          PUSHL   R6                                          ;
                           096C  C6   9F 00098          PUSHAB  WILDCARD_SPEC                               ;
     00000000G     00            07   FB 0009C          CALLS   #7, LIB$FIND_FILE                           ;
                   53            50   D0 000A3          MOVL    R0, STATUS                                  ;
                                 56   DD 000A6          PUSHL   R6                                          ; 0642
                                 56   DD 000A8          PUSHL   R6                                          ;
                                 56   DD 000AA          PUSHL   R6                                          ;
                   68            03   FB 000AC          CALLS   #3, STR$TRIM                                ;
                   20            53   E8 000AF          BLBS    STATUS, 6$                                  ; 0648
     000182CA      8F            53   D1 000B2          CMPL    STATUS, #99018                              ; 0649
                                 10   13 000B9          BEQL    5$                                          ;
                             04  AE   DD 000BB          PUSHL   STV                                         ; 0650
                                 56   DD 000BE          PUSHL   R6                                          ;
                                 01   DD 000C0          PUSHL   #1                                          ;
                       00B1109A  8F   DD 000C2          PUSHL   #11604122                                   ;
                   67            04   FB 000C8          CALLS   #4, LIB$SIGNAL                              ;
```

```
                      0A78    C6              01  D0 000CB 5$:    MOVL    #1, GET_NEW_SPEC                          ; 0651
                                              36  11 000D0        BRB     7$                                        ; 0652
                                      0A78    C6  D4 000D2 6$:    CLRL    GET_NEW_SPEC                              ; 0657
                      013C    C6              66  90 000D6        MOVB    RESULTANT_SPEC, IMAGE_FAB+52              ; 0659
                      0134    C6        04    A6  D0 000DB        MOVL    RESULTANT_SPEC+4, IMAGE_FAB+44            ; 0660
                                      0108    C6  9F 000E1        PUSHAB  IMAGE_FAB                                 ; 0661
                 00000000G    00              01  FB 000E5        CALLS   #1, SYS$OPEN                              ;
                              53              50  D0 000EC        MOVL    R0, STATUS                                ;
                              1E              53  E8 000EF        BLBS    STATUS, 8$                                ; 0662
                                      0114    C6  DD 000F2        PUSHL   IMAGE_FAB+12                              ;
                                              53  DD 000F6        PUSHL   STATUS                                    ;
                                              56  DD 000F8        PUSHL   R6                                        ;
                                              01  DD 000FA        PUSHL   #1                                        ;
                                  00B1109A    8F  DD 000FC        PUSHL   #11604122                                 ;
                              67              05  FB 00102        CALLS   #5, LIB$SIGNAL                            ;
                              08              53  E8 00105        BLBS    STATUS, 8$                                ; 0666
                                              52  DD 00108 7$:    PUSHL   R2                                        ; 0667
                      FEF1    CF              01  FB 0010A        CALLS   #1, ANL$OPEN_NEXT_IMAGE_FILE              ;
                                                  04 0010F        RET                                               ;
                              62              66  B0 00110 8$:    MOVW    RESULTANT_SPEC, (R2)                      ; 0671
          04    B2      04    B6              66  28 00113        MOVC3   RESULTANT_SPEC, @RESULTANT_SPEC+4, @4(R2) ; 0672
                      0964    C6      0114    C6  D0 00119        MOVL    IMAGE_FAB+12, CHAN                        ; 0674
                              50              01  D0 00120        MOVL    #1, R0                                    ; 0676
                                                  04 00123        RET                                               ;
                                              50  D4 00124 9$:    CLRL    R0                                        ; 0678
                                                  04 00126        RET                                               ;

; Routine Size:  295 bytes,    Routine Base:  $CODE$ + 0000
```

```
  1 %sbttl 'ANL$GET_IMAGE_BLOCK - Get Block from Image'
  1 !++
  1 ! Functional Description:
  1 !       This routine is called to read a block from the current image
  1 !       file, which is assumed to be open.
  1 !
  1 ! Formal Parameters:
  1 !       vbn             Virtual block number of desired block.
  1 !       buffer          Address of buffer pointer to fill in with the
  1 !                       address of our buffer.
  1 !
  1 ! Implicit Inputs:
  1 !       global data
  1 !
  1 ! Implicit Outputs:
  1 !       global data
  1 !
  1 ! Returned Value:
  1 !       $QIOW status
  1 !
  1 ! Side Effects:
  1 !
  1 !--
  1
  1
  2 global routine anl$get_image_block(vbn,buffer) = begin
  2
  2 bind
  2         buffer_ptr = .buffer: ref block[,byte];
  2
  2 local
  2         status: long,
  2         iosb : vector[ 4, word ] ;
  2
  2
  2 ! Read in the desired block to the static buffer.
  2
P 2 status = $qiow(
P 2                 efn = 7,
P 2                 chan = .chan,
P 2                 func = io$_readvblk,     ! Read a virtual block
P 2                 iosb = iosb,             ! I/O Status block
P 2                 p1 = blk_buf,            ! Buffer to read in to
P 2                 p2 = 512,                ! number of bytes to read
P 2                 p3 = .vbn                ! Virtual block number to read
  2               );
  2
  2 if not .status
  2 then
  2         return (.status);
  2
  2 ! Point the caller's pointer at our buffer.  Then return the $QIOW status
  2
  2 buffer_ptr = blk_buf;
  2 return (.iosb[0]);
  2
  1 end;
```

```
                                                              .EXTRN  SYS$QIOW

                                          0000 00000          .ENTRY  ANL$GET_IMAGE_BLOCK, Save nothing             ; 0704
                    5E                 08   C2 00002          SUBL2   #8, SP                                        ;
                                       7E   7C 00005          CLRQ    -(SP)                                         ; 0724
                                       7E   D4 00007          CLRL    -(SP)                                         ;
                               04      AC   DD 00009          PUSHL   VBN                                           ;
                    7E       0200      8F   3C 0000C          MOVZWL  #512, -(SP)                                   ;
                             0000'     CF   9F 00011          PUSHAB  BLK_BUF                                       ;
                                       7E   7C 00015          CLRQ    -(SP)                                         ;
                               20      AE   9F 00017          PUSHAB  IOSB                                          ;
                                       31   DD 0001A          PUSHL   #49                                           ;
                             0000'     CF   DD 0001C          PUSHL   CHAN                                          ;
                                       07   DD 00020          PUSHL   #7                                            ;
          00000000G 00                 0C   FB 00022          CALLS   #12, SYS$QIOW                                 ;
                    09                 50   E9 00029          BLBC    STATUS, 1$                                    ; 0726
               08   BC       0000'     CF   9E 0002C          MOVAB   BLK_BUF, @BUFFER                              ; 0732
                    50                 6E   3C 00032          MOVZWL  IOSB, R0                                      ; 0733
                                            04 00035 1$:      RET                                                   ; 0735

; Routine Size:  54 bytes,    Routine Base:  $CODE$ + 0127
```

```
%sbttl 'ANL$MAP_FIXUP_SECTION - Map Fixup Section into Memory'
!++
! Functional Description:
!       This routine is called to map a shareable image's fixup section
!       into memory so we can analyze it.  It is also called to free up
!       the mapping.  We read the section blocks, as opposed to actually
!       mapping the section, so that we can analyze remote images.
!
! Formal Parameters:
!       fixup_size      Number of blocks in fixup section.
!       fixup_vbn       Starting VBN of fixup section.
!
! Implicit Inputs:
!       global data
!
! Implicit Outputs:
!       global data
!
! Returned Value:
!       We return the address of the mapped memory, or zero if we couldn't
!       map it.
!
! Side Effects:
!
!--


global routine anl$map_fixup_section(fixup_size,fixup_vbn) = begin

own
        saved_size: long,
        saved_address: long;

local
        status: long,
        iosb : vector[ 4, word ] ;

builtin
        nullparameter;


! If we are called with both parameters, then we must map the fixup
! section into main memory and return its address.

if not nullparameter(1) and not nullparameter(2) then (

        saved_size = .fixup_size * 512;
        status = lib$get_vm(saved_size,saved_address);
        check (.status, .status);
        status = $qiow(
                efn = 7,
                chan = .chan,
                func = io$_readvblk,    ! Read a virtual block
                iosb = iosb,            ! I/O Status block
                p1 = .saved_address,    ! Buffer to read in to
                p2 = .saved_size,       ! number of bytes to read
                p3 = .fixup_vbn         ! Virtual block number to read
;  363          0793  3                 );
;  364          0794  3
;  365          0795  4         if not .iosb[0] or .iosb[1] nequ .saved_size then (
;  366          0796  4
;  367          0797  4                 ! Oops, we couldn't read it in correctly.
;  368          0798  4
;  369          0799  4                 anl$map_fixup_section();
;  370          0800  4                 return 0;
;  371          0801  3         ) else
;  372          0802  3                 return .saved_address;
;  373          0803  3
;  374          0804  3 ) else (
;  375          0805  3
;  376          0806  3         ! We are to free up the mapped memory.
;  377          0807  3
;  378          0808  3         status = lib$free_vm(saved_size,saved_address);
;  379          0809  3         check (.status, .status);
;  380          0810  3         return (ss$_normal);
;  381          0811  2 );
;  382          0812  2
;  383          0813  1 end;


                                                      .PSECT  $OWN$,NOEXE,2

                                         00A7C SAVED_SIZE:
                                                      .BLKB   4
                                         00A80 SAVED_ADDRESS:
                                                      .BLKB   4


                                                      .PSECT  $CODE$,NOWRT,2

                                    001C 00000        .ENTRY  ANL$MAP_FIXUP_SECTION, Save R2,R3,R4           ; 0763
                 54 00000000G   00    9E 00002        MOVAB   LIB$SIGNAL, R4                                  ;
                 53     0000'   CF    9E 00009        MOVAB   SAVED_SIZE, R3                                  ;
                 5E             08    C2 0000E        SUBL2   #8, SP                                          ;
                                6C    95 00011        TSTB    (AP)                                            ; 0780
                                65    13 00013        BEQL    4$                                              ;
                       04       AC    D5 00015        TSTL    4(AP)                                           ;
                                60    13 00018        BEQL    4$                                              ;
                 02             6C    91 0001A        CMPB    (AP), #2                                        ;
                                5B    1F 0001D        BLSSU   4$                                              ;
                       08       AC    D5 0001F        TSTL    8(AP)                                           ;
                                56    13 00022        BEQL    4$                                              ;
       63        04     AC      09    78 00024        ASHL    #9, FIXUP_SIZE, SAVED_SIZE                      ; 0782
                       04       A3    9F 00029        PUSHAB  SAVED_ADDRESS                                   ; 0783
                                53    DD 0002C        PUSHL   R3                                              ;
          00000000G    00       02    FB 0002E        CALLS   #2, LIB$GET_VM                                  ;
                       52       50    D0 00035        MOVL    R0, STATUS                                      ;
                       05       52    E8 00038        BLBS    STATUS, 1$                                      ; 0784
                                52    DD 0003B        PUSHL   STATUS                                          ;
                       64       01    FB 0003D        CALLS   #1, LIB$SIGNAL                                  ;
                                7E    7C 00040 1$:    CLRQ    -(SP)                                           ; 0793
                                7E    D4 00042        CLRL    -(SP)                                           ;
```

```
                                         08     AC  DD 00044          PUSHL   FIXUP_VBN                      ;
                                                63  DD 00047          PUSHL   SAVED_SIZE                     ;
                                         04     A3  DD 00049          PUSHL   SAVED_ADDRESS                  ;
                                                7E  7C 0004C          CLRQ    -(SP)                          ;
                                         20     AE  9F 0004E          PUSHAB  IOSB                           ;
                                                31  DD 00051          PUSHL   #49                            ;
                                       FEE8     C3  DD 00053          PUSHL   CHAN                           ;
                                                07  DD 00057          PUSHL   #7                             ;
                         00000000G  00          0C  FB 00059          CALLS   #12, SYS$QIOW                  ;
                                    52          50  D0 00060          MOVL    R0, STATUS                     ;
                                    08          6E  E9 00063          BLBC    IOSB, 2$                       ; 0795
        63          02    AE        10          00  ED 00066          CMPZV   #0, #16, IOSB+2, SAVED_SIZE    ;
                                                07  13 0006C          BEQL    3$                             ;
                              8E    AF          00  FB 0006E 2$:      CALLS   #0, ANL$MAP_FIXUP_SECTION      ; 0799
                                                50  D4 00072          CLRL    R0                             ; 0802
                                                    04 00074          RET                                    ;
                                    50   04     A3  D0 00075 3$:      MOVL    SAVED_ADDRESS, R0              ;
                                                    04 00079          RET                                    ; 0804
                                         04     A3  9F 0007A 4$:      PUSHAB  SAVED_ADDRESS                  ; 0808
                                                53  DD 0007D          PUSHL   R3                             ;
                         00000000G  00          02  FB 0007F          CALLS   #2, LIB$FREE_VM                ;
                                    52          50  D0 00086          MOVL    R0, STATUS                     ;
                                    05          52  E8 00089          BLBS    STATUS, 5$                     ; 0809
                                                52  DD 0008C          PUSHL   STATUS                         ;
                                    64          01  FB 0008E          CALLS   #1, LIB$SIGNAL                 ;
                                    50          01  D0 00091 5$:      MOVL    #1, R0                         ; 0810
                                                    04 00094          RET                                    ; 0813

; Routine Size:  149 bytes,    Routine Base:  $CODE$ + 015D
```

```
1 %sbttl 'ANL$GET_IMAGE_HEADER - get image header from opened image file'
1 !++
1 ! Functional Description:
1 !       Get the image header.  Call the image decode routine in the image
1 !       activator to return the image header already decoded into an
1 !       understantable form.
1 !
1 ! Formal Parameters:
1 !       ihdp            Longword address to return buffer address of
1 !                       decoded image header.
1 !       alias           Longword address to return the last longword of
1 !                       the image header.
1 !
1 ! Implicit Inputs:
1 !       global data
1 !
1 ! Implicit Outputs:
1 !       global data
1 !
1 ! Returned Value:
1 !       none.
1 !
1 ! Side Effects:
1 !
1 !--
1
1
2 global routine anl$get_image_header(ihdp,alias) = begin
2
2 bind
2       buffer_ptr = .ihdp: ref block[,byte];
2
2 local
2       status;
2
2 ! Point the caller's pointer to our buffer.  If we have already read the
2 ! image header in, it is still in our buffer.
2 !
2 buffer_ptr = image_buffer;
2
2 if .image_buf_valid
2 then
3     return( ss$_normal)
2 else
3     begin
3     status = img$decode_ihd( .chan,
3                             hdr_blk,
3                             image_buffer,
3                             vbn,
3                             offset,
3                             hdrver,
3                             .alias
3                             );
3     if .status
3     then
3         image_buf_valid = 1;
3     return (.status);
```

```
;  442          0871  2      end;
;  443          0872  2
;  444          0873  1 end;



                                             0000 00000          .ENTRY  ANL$GET_IMAGE_HEADER, Save nothing            ; 0841
                    04  BC     0000'  CF   9E 00002          MOVAB   IMAGE_BUFFER, @IHDP                           ; 0852
                        04     0000'  CF   E9 00008          BLBC    IMAGE_BUF_VALID, 1$                           ; 0854
                        50            01   D0 0000D          MOVL    #1, R0                                        ; 0858
                                           04 00010          RET                                                   ;
                                 08   AC   DD 00011 1$:      PUSHL   ALIAS                                         ; 0865
                               0000'  CF   9F 00014          PUSHAB  HDRVER                                        ; 0859
                               0000'  CF   9F 00018          PUSHAB  OFFSET                                        ;
                               0000'  CF   9F 0001C          PUSHAB  VBN                                           ;
                               0000'  CF   9F 00020          PUSHAB  IMAGE_BUFFER                                  ;
                               0000'  CF   9F 00024          PUSHAB  HDR_BLK                                       ;
                               0000'  CF   DD 00028          PUSHL   CHAN                                          ;
             00000000G  00            07   FB 0002C          CALLS   #7, IMG$DECODE_IHD                            ;
                        05            50   E9 00033          BLBC    STATUS, 2$                                    ; 0867
                 0000'  CF            01   D0 00036          MOVL    #1, IMAGE_BUF_VALID                           ; 0869
                                           04 0003B 2$:      RET                                                   ; 0873

; Routine Size:  60 bytes,    Routine Base:  $CODE$ + 01F2


;  445          0874  1
;  447          0875  1 %sbttl 'ANL$GET_ISD - Return the next ISD into buffer'
;  448          0876  1 !++
;  449          0877  1 ! Functional Description:
;  450          0878  1 !       Get the image section descriptor. Call the image decode routine
;  451          0879  1 !       in the image activator to return the image section descriptor.
;  452          0880  1 !
;  453          0881  1 ! Formal Parameters:
;  454          0882  1 !       isdp            Longword address to return buffer address of
;  455          0883  1 !                       image section descriptor.
;  456          0884  1 !
;  457          0885  1 ! Implicit Inputs:
;  458          0886  1 !       global data
;  459          0887  1 !
;  460          0888  1 ! Implicit Outputs:
;  461          0889  1 !       global data
;  462          0890  1 !
;  463          0891  1 ! Returned Value:
;  464          0892  1 !       Status from img$get_next_isd - IMG$_ENDOFHDR at EOF
;  465          0893  1 !
;  466          0894  1 ! Side Effects:
;  467          0895  1 !
;  468          0896  1 !--
;  469          0897  1
;  470          0898  1
;  471          0899  2 global routine anl$get_isd(isdp) = begin
;  472          0900  2
;  473          0901  2
;  474          0902  2 local
;  475          0903  2         status;
;  476          0904  2
;  477          0905  2 bind
;  478          0906  2         buffer_ptr = .isdp: ref block[,byte];
;  479          0907  2
;  480          0908  2 status = img$get_next_isd(
;  481          0909  2                         .chan,
;  482          0910  2                         hdr_blk,
;  483          0911  2                         image_buffer,
;  484          0912  2                         vbn,
;  485          0913  2                         offset,
;  486          0914  2                         isd_buf
;  487          0915  2                         );
;  488          0916  2
;  489          0917  2 buffer_ptr = isd_buf;
;  490          0918  2 return( .status);
;  491          0919  2
;  492          0920  1 end;



                                  0000 00000          .ENTRY  ANL$GET_ISD, Save nothing                    ; 0899
                        0000'  CF   9F 00002          PUSHAB  ISD_BUF                                      ; 0908
                        0000'  CF   9F 00006          PUSHAB  OFFSET                                       ;
                        0000'  CF   9F 0000A          PUSHAB  VBN                                          ;
                        0000'  CF   9F 0000E          PUSHAB  IMAGE_BUFFER                                 ;
                        0000'  CF   9F 00012          PUSHAB  HDR_BLK                                      ;
```

```
                                       0000'  CF   DD 00016          PUSHL   CHAN                                         ; 0909
                        L0000000G  00          06   FB 0001A          CALLS   #6, IMG$GET_NEXT_ISD                         ;
                               04  BC  0000'  CF   9E 00021          MOVAB   ISD_BUF, @ISDP                               ; 0917
                                                    04 00027          RET                                                  ; 0920

; Routine Size:  40 bytes,    Routine Base:  $CODE$ + 022E
;  494          0921  0 end eludom



                                                                   .EXTRN  LIB$SIGNAL

;                                    PSECT SUMMARY
;
;      Name                        Bytes                      Attributes
;
;  $OWN$                             2692  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                              32  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                             598  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



;                              Library Statistics
;
;                                                  -------- Symbols --------      Pages      Processing
;      File                                        Total    Loaded   Percent      Mapped     Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1                9776        40         0        581        00:01.0




;                                     COMMAND QUALIFIERS

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

; Size:           598 code + 2724 data bytes
; Run Time:          00:15.5
; Elapsed Time:      00:41.5
; Lines/CPU Min:      3562
; Lexemes/CPU-Min: 19725
; Memory Used:  177 pages
; Compilation Complete
```

0005 AH-BT13A-SE DIGITAL EQUIPMENT CORPORATION
VAX/VMS V4.0 CONFIDENTIAL AND PROPRIETARY

SETSHOACL
LIS

OBJEXEREQ
REQ

EXEFIXUP
LIS

ANALYZRMS
MAP

SHOWACL
LIS

EXESTUFF
LIS

ANALYZ

EXEINPUT
LIS

ANALYZOBJ
MAP

EXEDRIVE
LIS

RMSREQ
REQ