

[illegible]

```
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 EE NN NN PP PP UU UU TT
EE XX XX EE EE NNNN NN PP PP UU UU TT
EE XX XX EE EE NNNN NN PP PP UU UU TT
EEEEEEEE XX XX EEEEEEEEE IIIIII NN NN PPPPPPPP UU UU TT
EEEEEEEE XX XX EEEEEEEEE IIIIII NN NN PPPPPPPP UU UU TT
EE XX XX EE EE NN NN NNNN PP PP UU UU TT
EE XX XX EE EE NN NN NNNN PP PP UU UU TT
EE XX XX EE EE NN NN NNNN PP PP UU UU TT
EEEEEEEEEE XX XX EEEEEEEEE IIIIII NN NN PP PP UUUUUUUUUU TT
EEEEEEEEEE XX XX EEEEEEEEE IIIIII NN NN PP 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
```

```
0001 0 %title 'EXEINPUT - Handle Image Files & Libraries'
0002 0      module exeinput (
0003 1          ident='V04-000') = begin
0004 1
0005 1
0006 1 *****
0007 1 *
0008 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0009 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0010 1 *  ALL RIGHTS RESERVED.
0011 1 *
0012 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0013 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0014 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0015 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0016 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0017 1 *  TRANSFERRED.
0018 1 *
0019 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0020 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0021 1 *  CORPORATION.
0022 1 *
0023 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0024 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0025 1 *
0026 1 *
0027 1 *****
0028 1
0029 1
0030 1 ++
0031 1 Facility:      VAX/VMS Analyze Facility, Handle Image Files
0032 1
0033 1 Abstract:      This module is responsible for handling file specs from
0034 1                  the command line, and reading data from image files.
0035 1
0036 1
0037 1 Environment:
0038 1
0039 1 Author: Paul C. Anagnostopoulos, Creation Date: 31 March 1981
0040 1
0041 1 Modified By:
0042 1
0043 1 V03-007 ROP0009      Robert Posniak      15-JUN-1984
0044 1                  Clear image buffer valid flag when analyzing
0045 1                  more than one image.
0046 1
0047 1 V03-006 BLS0286      Benn Schreiber      20-MAR-1984
0048 1                  Correct 005.
0049 1
0050 1 V03-005 LJA0112      Laurie J. Anderson  21-Feb-1984
0051 1                  Add new related file parsing arguments to LIB$FIND_FILE
0052 1                  to make search lists behave properly.
0053 1
0054 1 V03-004 LJA0105      Laurie J. Anderson  25-Jan-1984
0055 1                  Changes due because now using the new image activator
0056 1                  decode routines. Now, open the image file through RMS
0057 1                  with the UFO option bit set, and do not bother to connect
```

EXEINPUT
V04-000

EXEINPUT - Handle Image Files & Libraries

L 14
15-Sep-1984 23:48:19
14-Sep-1984 11:52:44

VAX-11 Bliss-32 V4.0-742
[ANALYZ.SRC]EXEINPUT.B32;1

Page 2
(1)

```
.. 58      0058 1 1      stream. Will be using QIO's, because the decode routines
.. 59      0059 1 1      do, too. Add routines to read the image header and isd's.
.. 60      0060 1 1
.. 61      0061 1 1      V03-003 PCA1011      Paul C. Anagnostopoulos 1-Apr-1983
.. 62      0062 1 1      Change the message prefix to ANLOBJ$ to ensure that
.. 63      0063 1 1      message symbols are unique across all ANALYZEs. This
.. 64      0064 1 1      is necessitated by the new merged message files.
.. 65      0065 1 1
.. 66      0066 1 1      V03-002 PCA0020      Paul Anagnostopoulos 24-Mar-1982
.. 67      0067 1 1      Signal errors using the correct STV value.
.. 68      0068 1 1
.. 69      0069 1 1      V03-001 PCA0014      Paul Anagnostopoulos 22-Mar-1982
.. 70      0070 1 1      Use the resultant spec rather than the wildcard spec
.. 71      0071 1 1      when complaining about a file to be analyzed.
.. 72      0072 1 1  --
```

```

74 0073 1 %sbtll 'Module Declarations'
75 0074 1
76 0075 1   Libraries and Requires:
77 0076 1
78 0077 1
79 0078 1   library 'starlet';
80 0079 1   require 'objexereq';
81 0515 1
82 0516 1
83 0517 1   Table of Contents:
84 0518 1
85 0519 1
86 0520 1   forward routine
87 0521 1       anl$open_next_image_file,
88 0522 1       anl$get_image_block,
89 0523 1       anl$map_fixup_section,
90 0524 1       anl$get_image_header,
91 0525 1       anl$get_isd ;
92 0526 1
93 0527 1
94 0528 1   External References:
95 0529 1
96 0530 1
97 0531 1   external routine
98 0532 1       anl$image_positionals,
99 0533 1       cli$get_value: addressing_mode(general),
100 0534 1       img$decode_ihd: addressing_mode(general),
101 0535 1       img$get_next_isd: addressing_mode(general),
102 0536 1       lib$find_file: addressing_mode(general),
103 0537 1       lib$free_vm: addressing_mode(general),
104 0538 1       lib$get_vm: addressing_mode(general),
105 0539 1       str$trim: addressing_mode(general);
106 0540 1
107 0541 1
108 0542 1   Own Variables:
109 0543 1
110 0544 1   The following data structures are used to access and read blocks from
111 0545 1   the file we are to analyze.
112 0546 1
113 0547 1   own
114 0548 1       own_described_buffer(resultant_spec,nam$c_maxrss),
115 0549 1       image_fab: $fab(fop=ufo),      FAB for RMS UFO $OPEN of image file
116 0550 1       image_buffer: block[512,byte], contains decoded IHD
117 0551 1       isd_buf: block[512,byte],      Contains ISD block
118 0552 1       hdr_blk: block[512,byte],      contains block of image header
119 0553 1       blk_buf: block[512,byte],      contains block read by get_image_block
120 0554 1       hdrver,                        Header version
121 0555 1       vbn: long initial(1),           VBN of image found in hdr_blk
122 0556 1       offset,                        offset to the ISD's returned here
123 0557 1       chan : long initial( 0 ),        Channel assigned to the image file
124 0558 1       image_buf_valid : long initial(0); ! indicates if the image buffer is valid
125 0559 1
```

```
127 0560 1 %sbttl 'ANL$OPEN_NEXT_IMAGE_FILE - Right'
128 0561 1 ++
129 0562 1 Functional Description:
130 0563 1 This routine is called to open the next image file we are to analyze.
131 0564 1 It handles multiple file specs and wildcarding. It does not handle
132 0565 1 sharable image libraries.
133 0566 1
134 0567 1 Formal Parameters:
135 0568 1 opened_spec Address of descriptor of buffer in which to return
136 0569 1 the spec of the file we open. We set the length.
137 0570 1
138 0571 1 Implicit Inputs:
139 0572 1 global data
140 0573 1
141 0574 1 Implicit Outputs:
142 0575 1 global data
143 0576 1
144 0577 1 Returned Value:
145 0578 1 True if there is another image file, false otherwise.
146 0579 1
147 0580 1 Side Effects:
148 0581 1
149 0582 1 --
150 0583 1
151 0584 1
152 0585 2 global routine anl$open_next_image_file(opened_spec) = begin
153 0586 2
154 0587 2 bind
155 0588 2 opened_spec_dsc = .opened_spec: descriptor;
156 0589 2
157 0590 2 own
158 0591 2 own_described_buffer(wildcard_spec,nam$c_maxrss),
159 0592 2 wildcard_context: long initial(0),
160 0593 2 get_new_spec: long initial(true);
161 0594 2
162 0595 2 local
163 0596 2 stv: long,
164 0597 2 status: long;
165 0598 2
166 0599 2
167 0600 2 First clear flag to invalidate old image header
168 0601 2
169 0602 2
170 0603 2 image_buf_valid = 0;
171 0604 2
172 0605 2 ! If the wildcard context is zero (it means this is the first call), or
173 0606 2 ! have we finished with a file spec on the previous call, we must obtain
174 0607 2 ! the next file spec from the command line.
175 0608 2
176 0609 2 if .get_new_spec then (
177 0610 3 wildcard_spec[len] = nam$c_maxrss;
178 0611 3 status = cli$get_value(describe('file_specs'),wildcard_spec);
179 0612 3
180 0613 3 ! If there are no more specs, we are all done.
181 0614 3
182 0615 3 if not .status then
183 0616 3 return false;
```

EXEINPUT
V04-000

EXEINPUT - Handle Image Files & Libraries
ANL\$OPEN_NEXT_IMAGE_FILE - Right

B 15
15-Sep-1984 23:48:19
14-Sep-1984 11:52:44

VAX-11 Bliss-32 V4.0-742
[ANALYZ.SRC]EXEINPUT.B32;1

Page 5
(3)

```
: 184      0617 3      str$trim(wildcard_spec,wildcard_spec,wildcard_spec);
: 185      0618 3
: 186      0619 3      ! Call a routine to process any positional qualifiers for this spec.
: 187      0620 3      ! We don't know how to do that.
: 188      0621 3
: 189      0622 3      anl$image_positionals();
: 190      0623 2 );
: 191      0624 2
: 192      0625 2 ! On the other hand, if the previous call done is true, we may have just
: 193      0626 2 ! finished processing a file. Better deassign channel.
: 194      0627 2
: 195      0628 3 if .chan nequ 0 then (
: 196      0629 3      status = $dassgn(chan=.chan);
: 197      0630 3      check (.status, anlobj$_closein,1,resultant_spec,.status);
: 198      0631 3      chan = 0;
: 199      0632 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$trim(resultant_spec,resultant_spec,resultant_spec);
: 211 0643 2
: 212 0644 2 ! If we failed to find a file, then reset the wildcard context and call
: 213 0645 2 ! ourselves recursively to process the next file spec. Also give an
: 214 0646 2 ! error, unless we just plain ran out of files.
: 215 0647 2
: 216 0648 2 if not .status then (
: 217 0649 2     if .status nequ rms$ nmf then
: 218 0650 2         signal (anlobj$_openin,1,resultant_spec,.stv);
: 219 0651 2         get_new_spec = true;
: 220 0652 2         return anl$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 \$SPLIT\$,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 $OWNS$,NOEXE,2

000000FF 00000 RESULTANT_SPEC:
00000000' 00004 .LONG 255
00008 .ADDRESS RESULTANT_SPEC+8
00107 .BLKB 255
03 00108 IMAGE_FAB:
50 00109 .BYTE 3
0000 0010A .BYTE 80
00020000 0010C .WORD 0
00000000 00110 .LONG 131072
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:
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 ANLOBS$OK, ANLOBS$ANYTHING
.EXTRN ANLOBS$DATATYPE
.EXTRN ANLOBS$ERRORCOUNT
.EXTRN ANLOBS$ERRORNONE
.EXTRN ANLOBS$ERRORS, ANLOBS$EXEFIXA
.EXTRN ANLOBS$EXEFIXAIMAGE
.EXTRN ANLOBS$EXEFIXALINE
.EXTRN ANLOBS$EXEFIXCOUNT
.EXTRN ANLOBS$EXEFIXEXTRA
.EXTRN ANLOBS$EXEFIXFIXED
.EXTRN ANLOBS$EXEFIXFLAGS
.EXTRN ANLOBS$EXEFIXG
.EXTRN ANLOBS$EXEFIXGIMAGE
.EXTRN ANLOBS$EXEFIXGLINE
.EXTRN ANLOBS$EXEFIXLIST
.EXTRN ANLOBS$EXEFIXNAME
.EXTRN ANLOBS$EXEFIXNAME0
.EXTRN ANLOBS$EXEFIXP
.EXTRN ANLOBS$EXEFIXPSECT
.EXTRN ANLOBS$EXEFIXUP
.EXTRN ANLOBS$EXEFIXUPNONE
.EXTRN ANLOBS$EXEGST, ANLOBS$EXEHDR
.EXTRN ANLOBS$EXEHDRACTIVE
.EXTRN ANLOBS$EXEHDRBLKCOUNT
.EXTRN ANLOBS$EXEHDRCHANCOUNT
.EXTRN ANLOBS$EXEHDRCHANDEF
.EXTRN ANLOBS$EXEHDRDECECO
.EXTRN ANLOBS$EXEHDRDMT
.EXTRN ANLOBS$EXEHDRDST
.EXTRN ANLOBS$EXEHDRFILEID
.EXTRN ANLOBS$EXEHDRFIXED
.EXTRN ANLOBS$EXEHDRFLAGS
.EXTRN ANLOBS$EXEHDRGBLIDENT
.EXTRN ANLOBS$EXEHDRGST
.EXTRN ANLOBS$EXEHDRIDENT
.EXTRN ANLOBS$EXEHDRIMAGEID
.EXTRN ANLOBS$EXEHDRISD
.EXTRN ANLOBS$EXEHDRISDBASE
.EXTRN ANLOBS$EXEHDRISDCOUNT
.EXTRN ANLOBS$EXEHDRISDFLAGS
.EXTRN ANLOBS$EXEHDRISDGBLNAME
.EXTRN ANLOBS$EXEHDRISDNUM
.EXTRN ANLOBS$EXEHDRISDPFCDEF
.EXTRN ANLOBS$EXEHDRISDPFCISZ
.EXTRN ANLOBS$EXEHDRISDTYPE
.EXTRN ANLOBS$EXEHDRISDVBN
.EXTRN ANLOBS$EXEHDRLINKID
```

```

EXEINPUT - Handle Image Files & Libraries
ANLSOPEN_NEXT_IMAGE_FILE - Right

```

VAX-11 Bliss-32 V4.0-742
[ANALYZ.SRC]EXEINPUT.B32:1

Page 9
(4)

```

. 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$_EXEHEDING
. EXTRN  ANLOBJ$_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$_OBJGSDENVPAR
. EXTRN  ANLOBJ$_OBJGSDPEPM
. EXTRN  ANLOBJ$_OBJGSDPEPMW
. EXTRN  ANLOBJ$_OBJGSDIDC
. EXTRN  ANLOBJ$_OBJGSDIDCENT
. EXTRN  ANLOBJ$_OBJGSDIDCFLAGS
. EXTRN  ANLOBJ$_OBJGSDIDCMATCH
. EXTRN  ANLOBJ$_OBJGSDIDCOBJ
. EXTRN  ANLOBJ$_OBJGSDIDCVALA
. EXTRN  ANLOBJ$_OBJGSDIDCVALB
. EXTRN  ANLOBJ$_OBJGSDLPEM
. EXTRN  ANLOBJ$_OBJGSDLPRO
. EXTRN  ANLOBJ$_OBJGSDLSY
. EXTRN  ANLOBJ$_OBJGSDPRO
. EXTRN  ANLOBJ$_OBJGSDPROW
. EXTRN  ANLOBJ$_OBJGSDPSC

```

```
.EXTRN ANLOBS$_OBJGSDPSCALIGN
.EXTRN ANLOBS$_OBJGSDPSCALLOC
.EXTRN ANLOBS$_OBJGSDPSCBASE
.EXTRN ANLOBS$_OBJGSDPSCFLAGS
.EXTRN ANLOBS$_OBJGSDREC
.EXTRN ANLOBS$_OBJGSDSPSC
.EXTRN ANLOBS$_OBJGSDSYM
.EXTRN ANLOBS$_OBJGSDSYMW
.EXTRN ANLOBS$_OBJGTXREC
.EXTRN ANLOBS$_OBJHDRIGNREC
.EXTRN ANLOBS$_OBJHEADING
.EXTRN ANLOBS$_OBJLITINDEX
.EXTRN ANLOBS$_OBJLNKREC
.EXTRN ANLOBS$_OBJLNMREC
.EXTRN ANLOBS$_OBJMHDCREATE
.EXTRN ANLOBS$_OBJMHDNAME
.EXTRN ANLOBS$_OBJMHDPATCH
.EXTRN ANLOBS$_OBJMHDREC
.EXTRN ANLOBS$_OBJMHDRECSIZ
.EXTRN ANLOBS$_OBJMHDSTRLVL
.EXTRN ANLOBS$_OBJMHDVERSION
.EXTRN ANLOBS$_OBJMTCORRECT
.EXTRN ANLOBS$_OBJMTCINPUT
.EXTRN ANLOBS$_OBJMTCNAME
.EXTRN ANLOBS$_OBJMTCREC
.EXTRN ANLOBS$_OBJMTCSEQNUM
.EXTRN ANLOBS$_OBJMTCUIC
.EXTRN ANLOBS$_OBJMTCVERSION
.EXTRN ANLOBS$_OBJMTCWHEN
.EXTRN ANLOBS$_OBJPROARGCOUNT
.EXTRN ANLOBS$_OBJPROARGNUM
.EXTRN ANLOBS$_OBJPSECT
.EXTRN ANLOBS$_OBJSRCREC
.EXTRN ANLOBS$_OBJSTATHEADING1
.EXTRN ANLOBS$_OBJSTATHEADING2
.EXTRN ANLOBS$_OBJSTATLINE
.EXTRN ANLOBS$_OBJSTATTOTAL
.EXTRN ANLOBS$_OBJSYMBOL
.EXTRN ANLOBS$_OBJSYMFLAGS
.EXTRN ANLOBS$_OBJTIRARGINDEX
.EXTRN ANLOBS$_OBJTIRCMD
.EXTRN ANLOBS$_OBJTIRCMDSTK
.EXTRN ANLOBS$_OBJTBTRC
.EXTRN ANLOBS$_OBJTIRREC
.EXTRN ANLOBS$_OBJTIRSTOIM
.EXTRN ANLOBS$_OBJTIRVIELD
.EXTRN ANLOBS$_OBJTTLREC
.EXTRN ANLOBS$_OBJVALUE
.EXTRN ANLOBS$_OBJUVALUE
.EXTRN ANLOBS$_PROTECTION
.EXTRN ANLOBS$_SEVERITY
.EXTRN ANLOBS$_TEXT, ANLOBS$_TEXTHDR
.EXTRN ANLOBS$_NOSUCHMOD
.EXTRN ANLOBS$_BADDATE
.EXTRN ANLOBS$_BADHDRBLKCOUNT
.EXTRN ANLOBS$_BADSEVERITY
.EXTRN ANLOBS$_BADSYS1ST
```

```
.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$_OBJMHDBADSTRVL
.EXTRN ANLOBJ$_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$_OBJUNDEFPS
.EXTRN ANALYZ$FACILITY
.EXTRN ANL$IMAGE_POSITIONALS
.EXTRN CLIS$GET_VALUE, IMG$DECODE_IHD
.EXTRN IMG$GET_NEXT_ISD
.EXTRN LIB$FIND_FILE, LIB$FREE_VM
.EXTRN LIB$GET_VM, STR$TRIM
.EXTRN SYSS$ASSGN, SYSS$OPEN

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

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

EXEINPUT
V04-000

EXEINPUT - Handle Image Files & Libraries
ANLSOPEN_NEXT_IMAGE_FILE - Right

J 15
15-Sep-1984 23:48:19
14-Sep-1984 11:52:44

VAX-11 Bliss-32 V4.0-742
[ANALYZ.SRC]EXEINPUT.B32;1

Page 13
(4)

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		MOVW	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, SYSSOPEN	:	
	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, ANLSOPEN_NEXT_IMAGE_FILE	:	
			04	0010F		RET		:		
	62		66	B0	00110	8\$:	MOVW	RESULTANT_SPEC, (R2)	:	0671
04	B2	04	66	28	00113		MOVW	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

```
: 248 0679 1 %sbttl 'ANL$GET_IMAGE_BLOCK - Get Block from Image'
: 249 0680 1 ++
: 250 0681 1 Functional Description:
: 251 0682 1 This routine is called to read a block from the current image
: 252 0683 1 file, which is assumed to be open.
: 253 0684 1
: 254 0685 1 Formal Parameters:
: 255 0686 1 vbn Virtual block number of desired block.
: 256 0687 1 buffer Address of buffer pointer to fill in with the
: 257 0688 1 address of our buffer.
: 258 0689 1
: 259 0690 1 Implicit Inputs:
: 260 0691 1 global data
: 261 0692 1
: 262 0693 1 Implicit Outputs:
: 263 0694 1 global data
: 264 0695 1
: 265 0696 1 Returned Value:
: 266 0697 1 $QIOW status
: 267 0698 1
: 268 0699 1 Side Effects:
: 269 0700 1
: 270 0701 1 --
: 271 0702 1
: 272 0703 1
: 273 0704 2 global routine anl$get_image_block(vbn,buffer) = begin
: 274 0705 2
: 275 0706 2 bind
: 276 0707 2 buffer_ptr = .buffer: ref block[,byte];
: 277 0708 2
: 278 0709 2 local
: 279 0710 2 status: long,
: 280 0711 2 iosb : vector[ 4, word ] ;
: 281 0712 2
: 282 0713 2
: 283 0714 2 ! Read in the desired block to the static buffer.
: 284 0715 2
: 285 P 0716 2 status = $qiow(
: 286 P 0717 2 efn = 7,
: 287 P 0718 2 chan = .chan,
: 288 P 0719 2 func = io$readvblk, ! Read a virtual block
: 289 P 0720 2 iosb = iosb, ! I/O Status block
: 290 P 0721 2 p1 = blk_buf, ! Buffer to read in to
: 291 P 0722 2 p2 = 512, ! number of bytes to read
: 292 P 0723 2 p3 = .vbn ! Virtual block number to read
: 293 0724 2 );
: 294 0725 2
: 295 0726 2 if not .status
: 296 0727 2 then
: 297 0728 2 return (.status);
: 298 0729 2
: 299 0730 2 ! Point the caller's pointer at our buffer. Then return the $QIOW status
: 300 0731 2
: 301 0732 2 buffer_ptr = blk_buf;
: 302 0733 2 return(.iosb[0]);
: 303 0734 2
: 304 0735 1 end;
```

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

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

```
306 0736 1 %sbttl 'ANL$MAP_FIXUP_SECTION - Map Fixup Section into Memory'
307 0737 1 ++
308 0738 1 Functional Description:
309 0739 1 This routine is called to map a shareable image's fixup section
310 0740 1 into memory so we can analyze it. It is also called to free up
311 0741 1 the mapping. We read the section blocks, as opposed to actually
312 0742 1 mapping the section, so that we can analyze remote images.
313 0743 1
314 0744 1 Formal Parameters:
315 0745 1 fixup_size      Number of blocks in fixup section.
316 0746 1 fixup_vbn       Starting VBN of fixup section.
317 0747 1
318 0748 1 Implicit Inputs:
319 0749 1 global data
320 0750 1
321 0751 1 Implicit Outputs:
322 0752 1 global data
323 0753 1
324 0754 1 Returned Value:
325 0755 1 We return the address of the mapped memory, or zero if we couldn't
326 0756 1 map it.
327 0757 1
328 0758 1 Side Effects:
329 0759 1
330 0760 1 --
331 0761 1
332 0762 1
333 0763 2 global routine anl$map_fixup_section(fixup_size,fixup_vbn) = begin
334 0764 2
335 0765 2 own
336 0766 2 saved_size: long,
337 0767 2 saved_address: long;
338 0768 2
339 0769 2 local
340 0770 2 status: long,
341 0771 2 iosb : vector[ 4, word ] ;
342 0772 2
343 0773 2 builtin
344 0774 2 nullparameter;
345 0775 2
346 0776 2
347 0777 2 ! If we are called with both parameters, then we must map the fixup
348 0778 2 ! section into main memory and return its address.
349 0779 2
350 0780 2 if not nullparameter(1) and not nullparameter(2) then (
351 0781 3
352 0782 3 saved_size = .fixup_size * 512;
353 0783 3 status = lib$get_vm(saved_size,saved_address);
354 0784 3 check (.status, .status);
355 P 0785 3 status = $qiow(
356 P 0786 3 efn = 7,
357 P 0787 3 chan = .chan,
358 P 0788 3 func = io$_readvblk, ! Read a virtual block
359 P 0789 3 iosb = iosb, ! I/O Status block
360 P 0790 3 p1 = .saved_address, ! Buffer to read in to
361 P 0791 3 p2 = .saved_size, ! number of bytes to read
362 P 0792 3 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 ! Ops, 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 \$OWNS\$,NOEXE,2

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

.PSECT \$CODE\$,NOWRT,2

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

EXEINPUT
V04-000

EXEINPUT - Handle Image Files & Libraries
ANL\$MAP_FIXUP_SECTION - Map Fixup Section into

B 16
15-Sep-1984 23:48:19
14-Sep-1984 11:52:44

VAX-11 Bliss-32 V4.0-742
[ANALYZ.SRC]EXEINPUT.B32;1

Page 18
(6)

			08	AC	DD	00044	PUSHL	FIXUP_VBN	
			63	DD	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	DC	00057	PUSHL	#7	
		00000000G	00	0C	FC	00059	CALLS	#12, SYSSQIOW	
			52	50	DD	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	CALLS	#0, ANL\$MAP_FIXUP_SECTION	0799
				50	D4	00072	CLRL	R0	0802
					04	00074	RET		
		50		04	A3	DD	00075	3\$: MOVL	SAVED_ADDRESS, R0
					04	00079	RET		0804
				04	A3	9F	0007A	4\$: PUSHAB	SAVED_ADDRESS
					53	DD	0007D	PUSHL	R3
		00000000G	00	02	FB	0007F	CALLS	#2, LIB\$FREE_VM	
			52	50	DD	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	DD	00091	MOVL	#1, R0	0810
				04	00094	RET			0813

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

```

385 0814 1 %sbttl 'ANL$GET_IMAGE_HEADER - get image header from opened image file'
386 0815 1 ++
387 0816 1 Functional Description:
388 0817 1     Get the image header. Call the image decode routine in the image
389 0818 1     activator to return the image header already decoded into an
390 0819 1     understandable form.
391 0820 1
392 0821 1 Formal Parameters:
393 0822 1     ihdp           Longword address to return buffer address of
394 0823 1                 decoded image header.
395 0824 1     alias        Longword address to return the last longword of
396 0825 1                 the image header.
397 0826 1
398 0827 1 Implicit Inputs:
399 0828 1     global data
400 0829 1
401 0830 1 Implicit Outputs:
402 0831 1     global data
403 0832 1
404 0833 1 Returned Value:
405 0834 1     none.
406 0835 1
407 0836 1 Side Effects:
408 0837 1
409 0838 1 --
410 0839 1
411 0840 1
412 0841 2 global routine anl$get_image_header(ihdp,alias) = begin
413 0842 2
414 0843 2 bind
415 0844 2     buffer_ptr = .ihdp: ref block[,byte];
416 0845 2
417 0846 2 local
418 0847 2     status;
419 0848 2
420 0849 2 ! Point the caller's pointer to our buffer. If we have already read the
421 0850 2 ! image header in, it is still in our buffer.
422 0851 2
423 0852 2 buffer_ptr = image_buffer;
424 0853 2
425 0854 2 if .image_buf_valid
426 0855 2 then
427 0856 3     return( ss$normal)
428 0857 2 else
429 0858 3     begin
430 0859 3         status = img$decode_ihd( .chan,
431 0860 3             hdr_blk,
432 0861 3             image_buffer,
433 0862 3             vbn,
434 0863 3             offset,
435 0864 3             hdrver,
436 0865 3             .alias
437 0866 3             );
438 0867 3     if .status
439 0868 3     then
440 0869 3         image_buf_valid = 1;
441 0870 3     return (.status);

```

EXEINPUT
V04-000

EXEINPUT - Handle Image Files & Libraries
ANL\$GET_IMAGE_HEADER - get image header from op

D 16
15-Sep-1984 23:48:19
14-Sep-1984 11:52:44

VAX-11 Bliss-32 V4.0-742
[ANALYZ.SRC]EXEINPUT.B32;1

Page 20
(7)

: 442
: 443
: 444

0871 2 end;
0872 2
0873 1 end;

04 BC 0000' CF 0000 00000
04 0000' CF E9 00002
50 01 D0 0000D
04 00010
08 AC DD 00011 1\$:
0000' CF 9F 00014
0000' CF 9F 00018
0000' CF 9F 0001C
0000' CF 9F 00020
0000' CF 9F 00024
0000' CF DD 00028
00000000G 00 07 FB 0002C
05 50 E9 00033
0000' CF 01 D0 00036
04 0003B 2\$:

.ENTRY ANL\$GET_IMAGE_HEADER, Save nothing
MOVAB IMAGE_BUFFER, @IHP
BLBC IMAGE_BUF_VALID, 1\$
MOVL #1, R0
RET
PUSHL ALIAS
PUSHAB HDRVER
PUSHAB OFFSET
PUSHAB VBN
PUSHAB IMAGE_BUFFER
PUSHAB HDR_BCK
PUSHL CHAN
CALLS #7, IMG\$DECODE_IHD
BLBC STATUS, 2\$
MOVL #1, IMAGE_BUF_VALID
RET

: 0841
: 0852
: 0854
: 0858
: 0865
: 0859
: 0867
: 0869
: 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 = .isd: 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 0000 0000
0000: CF 9F 00002
0000: CF 9F 00006
0000: CF 9F 0000A
0000: CF 9F 0000E
0000: CF 9F 00012
```

```

.ENTRY ANL$GET_ISD, Save nothing
PUSHAB ISD_BUF
PUSHAB OFFSET
PUSHAB VBN
PUSHAB IMAGE_BUFFER
PUSHAB HDR_BLK
```

```

: 0899
: 0908
:
:
:
:
:
```

EXEINPUT
V04-000

EXEINPUT - Handle Image Files & Libraries
ANL\$GET_ISD - Return the next ISD into buffer

F 16
15-Sep-1984 23:48:19
14-Sep-1984 11:52:44

VAX-11 Bliss-32 V4.0-742
[ANALYZ.SRC]EXEINPUT.B32;1

Page 22
(8)

00000000G 00 0000' CF DD 00016
04 BC 0000' CF 06 FB 0001A
9E 00021
04 00027

PUSHL CHAN
CALLS #6, IMG\$GET_NEXT_ISD
MOVAB ISD_BUF, @ISDP
RET

: 0909
:
: 0917
: 0920

; Routine Size: 40 bytes, Routine Base: \$CODE\$ + 022E

EXEINPUT
V04-000

EXEINPUT - Handle Image Files & Libraries
ANL\$GET_ISD - Return the next ISD into buffer

G 16
15-Sep-1984 23:48:19
14-Sep-1984 11:52:44

VAX-11 Bliss-32 V4.0-742
[ANALYZ.SRC]EXEINPUT.B32;1

Page 23
(9)

: 494

0921 0 end eludom

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	2692	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	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

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing 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
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY