

[illegible]

3-

Sy

LI

LI
LILI
LI
LILI
LI

LI

LI
LILI
LI
LILI
LI

LI

LI
LI

LI

LI
LI

LI

LI
LI

LI

11

LI

LI
LI

LI

LI
LI

21

LI

LI
LI

22

11

LI
LI

LI

LI
LI

LI

```

LL      IIIIII  BBBB BBBB  FFFFFFFF  NN      NN  DDDDDDDD  IIIIII  MM      MM  GGGGGGGG
LL      IIIIII  BBBB BBBB  FFFFFFFF  NN      NN  DDDDDDDD  IIIIII  MM      MM  GGGGGGGG
LL      II      BB      BB  FF      FF  NN      NN  DD      DD  II      II  MM      MM  GG      GG
LL      II      BB      BB  FF      FF  NN      NN  DD      DD  II      II  MM      MM  GG      GG
LL      II      BB      BB  FF      FF  NNNN      NN  DD      DD  II      II  MM      MM  GG      GG
LL      II      BBBB BBBB  FFFFFFFF  NN      NN  DD      DD  II      II  MM      MM  GG      GG
LL      II      BBBB BBBB  FFFFFFFF  NN      NN  DD      DD  II      II  MM      MM  GG      GG
LL      II      BB      BB  FF      FF  NN      NN  DD      DD  II      II  MM      MM  GG      GG
LL      II      BB      BB  FF      FF  NN      NN  DD      DD  II      II  MM      MM  GG      GG
LL      II      BB      BB  FF      FF  NN      NN  DD      DD  II      II  MM      MM  GG      GG
LL      II      BB      BB  FF      FF  NN      NN  DD      DD  II      II  MM      MM  GG      GG
LLLLLLLL  IIIIII  BBBB BBBB  FF      FF  NN      NN  DDDDDDDD  IIIIII  MM      MM  GGGGGG  GG
LLLLLLLL  IIIIII  BBBB BBBB  FF      FF  NN      NN  DDDDDDDD  IIIIII  MM      MM  GGGGGG  GG

```



```

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
LLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLL  IIIIII  SSSSSSSS

```



```
1 0001 0 MODULE lib$find_image( ! File: LIBFNDIMG.B32
2 0002 0 LANGUAGE (BLISS32),
3 0003 0 IDENT = 'V03-006'
4 0004 0 ) =
5 0005 1 BEGIN
6 0006 1 %TITLE 'Dynamically activate shareable image';
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
12 0012 1 * ALL RIGHTS RESERVED. *
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
19 0019 1 * TRANSFERRED. *
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
23 0023 1 * CORPORATION. *
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1 ++
32 0032 1
33 0033 1 FACILITY: Run time library
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1 Implements dynamic linking to shareable images
38 0038 1
39 0039 1 ENVIRONMENT:
40 0040 1
41 0041 1 VAX native, user mode.
42 0042 1
43 0043 1 --
44 0044 1
45 0045 1
46 0046 1 AUTHOR: Benn Schreiber
47 0047 1
48 0048 1 CREATION DATE: 5-Feb-1983
49 0049 1
50 0050 1 MODIFIED BY:
51 0051 1
52 0052 1 V03-006 BLS0298 Benn Schreiber 9-APR-1984
53 0053 1 Modify method of looking up file to not rely on
54 0054 1 string returned from image activator.
55 0055 1
56 0056 1 V03-005 BLS0277 Benn Schreiber 7-FEB-1984
57 0057 1 Convert to RTL routine.
```

LIB\$FIND_IMAGE Dynamically activate shareable image
V03-006

J 4
16-Sep-1984 00:56:42
14-Sep-1984 12:38:55

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]LIBFNDIMG.B32;1

Page 2
(1)

:	58	0058	1	:	
:	59	0059	1	:	
:	60	0060	1	:	
:	61	0061	1	:	
:	62	0062	1	:	
:	63	0063	1	:	
:	64	0064	1	:	
:	65	0065	1	:	
:	66	0066	1	:	
:	67	0067	1	:	
:	68	0068	1	:	
:	69	0069	1	:	
:	70	0070	1	:	
:	71	0071	1	:	
:	72	0072	1	:	--

V03-004	BLS0225	Benn Schreiber	16-Jun-1983
	Add flags argument to call to util\$read_object. Upcase		
	routine name before looking up.		
V03-003	BLS0222	Benn Schreiber	20-May-1983
	Copy file spec to hdrbuf if error before signalling.		
	spec goes away if message file image activated		
V03-002	PDG0002	Peter D Gilbert	22-Mar-1983
	Remove .ADDRESS reference in UTIL\$FIND_SYMBOL		
V03-001	BLS0208	Benn Schreiber	18-Feb-1983
	Report full file spec on failed activation if available		

LIB
V03

; R


```
74 0073 1 %SBTTL 'Declarations';
75 0074 1
76 0075 1 BLISS Libraries
77 0076 1
78 0077 1 SWITCHES
79 0078 1 ADDRESSING_MODE (EXTERNAL = GENERAL,
80 0079 1 NONEXTERNAL = WORD_RELATIVE);
81 0080 1
82 0081 1 LIBRARY
83 0082 1 'SYSS$LIBRARY:LIB'; !Need to get IFD definitions from LIB
84 0083 1
85 0084 1 REQUIRE
86 0085 1 'RTLIN:RTPSECT'; !RTL psect definitions
87 0180 1
88 0181 1 DECLARE_PSECTS (LIB); !Declare psepts for LIB facility
89 0182 1
90 0183 1 FIELD
91 0184 1
92 0185 1 Define the Module information descriptor. There is one for each image
93 0186 1 mapped.
94 0187 1
95 0188 1 mid_fields =
96 0189 1 SET
97 0190 1 mid_l_next = [0,0,32,0], !Next in list or 0 if end
98 0191 1 mid_l_symlst = [4,0,32,0], !Listhead of symbols this image
99 0192 1 mid_l_base = [8,0,32,0], !Base of image in memory
100 0193 1 mid_l_vbn = [12,0,32,0], !VBN of GST
101 0194 1 mid_b_namlen = [16,0,8,0], !Length of name
102 0195 1 mid_t_name = [17,0,0,0] !Start of image name
103 0196 1 TES;
104 0197 1
105 0198 1 Define a symbol descriptor
106 0199 1
107 0200 1 FIELD
108 0201 1 msd_fields =
109 0202 1 SET
110 0203 1 msd_l_left = [0,0,32,0], !Left subtree pointer
111 0204 1 msd_l_right = [4,0,32,0], !Right subtree pointer
112 0205 1 msd_w_bal = [8,0,16,0], !Balance this node
113 0206 1 msd_l_value = [10,0,32,0], !Value of this symbol
114 0207 1 msd_b_namlen = [14,0,8,0], !Length of name
115 0208 1 msd_t_name = [15,0,0,0] !Start of symbol name
116 0209 1 TES;
117 0210 1
118 0211 1 LITERAL
119 0212 1 msd_c_size = 15; !Length of msd without name
120 0213 1
121 0214 1 GLOBAL
122 0215 1 lib$$gl_imagelist : REF $BBLOCK FIELD(mid_fields); !Mapped image listhead
123 0216 1
124 0217 1 EXTERNAL LITERAL
125 0218 1 lib$m_lnk_1mod : UNSIGNED(8); !Only read one module
126 0219 1
127 0220 1 EXTERNAL ROUTINE
128 0221 1 lib$get_ef, !Allocate event flag
129 0222 1 lib$free_ef, !Free event flag
130 0223 1 lib$get_vm, !Allocate virtual memory
```

LIB\$FIND_IMAGE
V03-006

Dynamically activate shareable image
Declarations

L 4
16-Sep-1984 00:56:42
14-Sep-1984 12:38:55

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]LIBFNDIMG.B32;1

Page 4
(2)

```
: 131      0224 1      lib$free_vm,      !Deallocate it
: 132      0225 1      lib$insert_tree, !Insert element into binary tree
: 133      0226 1      lib$lookup_tree, !Lookup element in binary tree
: 134      0227 1      lib$scopy_r_dx,   !Dynamic string copy
: 135      0228 1      str$free1_dx,    !Free dynamic string
: 136      0229 1      str$upcase,      !Convert string to upper case
: 137      0230 1      lib$$getfilename, !Get filename descr. from fab/nam
: 138      0231 1      lib$$report_io_error, !Report I/O error
: 139      0232 1      lib$$read_object; !Read object file
: 140      0233 1
: 141      0234 1      BIND
: 142      0235 1          sysshr_cstring = UPLIT(BYTE(%CHARCOUNT('SYSS$SHARE:.EXE'),
: 143      0236 1          'SYSS$SHARE:.EXE')) : VECTOR[,BYTE];
: 144      0237 1      !
: 145      0238 1      ! Define facility-specific names for shared messages
: 146      0239 1      !
: 147      P 0240 1      $SHR_MSGDEF(LIB,21,LOCAL,
: 148      P 0241 1          (actimage,error), !Error activating image
: 149      P 0242 1          (insvirmem,error), !Insufficient virtual memory
: 150      P 0243 1          (openin,error), !Error opening input file
: 151      P 0244 1          (closein,warning), !Error closing input file
: 152      0245 1          (readerr,error)); !Read error
```



```
154 0246 1 %SBTTL 'get_rec - Read GST record';
155 0247 1 ROUTINE get_rec(datavector,recdesc) =
156 0248 2 BEGIN
157 0249 2 ++
158 0250 2 FUNCTIONAL DESCRIPTION:
159 0251 2
160 0252 2 Read the next record of the GST. This routine called by
161 0253 2 LIB$$READ_OBJECT
162 0254 2
163 0255 2 INPUTS:
164 0256 2
165 0257 2 datavector = address of data vector passed by read_gst
166 0258 2 1st longword is current MID block
167 0259 2 2nd longword is file RAB
168 0260 2 recdesc = address of dynamic string descriptor to return record
169 0261 2
170 0262 2 --
171 0263 2 MAP
172 0264 2 datavector : REF VECTOR[,LONG];
173 0265 2
174 0266 2 BIND
175 0267 2 midptr = .datavector[0] : $BBLOCK FIELD(mid_fields),
176 0268 2 irab = .datavector[1] : $BBLOCK;
177 0269 2
178 0270 2 LOCAL
179 0271 2 status;
180 0272 2
181 0273 2 |
182 0274 2 | Read the record. If successful read, copy record to dynamic string
183 0275 2 |
184 0276 2 status = $GET(RAB=irab,ERR=lib$$report_io_error);
185 0277 2 IF NOT .status
186 0278 2 THEN RETURN .status;
187 0279 2
188 0280 2 status = lib$scopy_r_dx(irab[rab$w_rsz],.irab[rab$l_rbf],.recdesc);
189 0281 2 IF NOT .status
190 0282 2 THEN BEGIN
191 0283 2 SIGNAL(lib$ insvirmem,0,.status);
192 0284 2 RETURN .status
193 0285 2 END;
194 0286 2
195 0287 2 RETURN ss$_normal
196 0288 1 END;
```

```
.TITLE LIB$FIND_IMAGE Dynamically activate shareable i
                                mage
```

```
.IDENT \V03-006\
```

```
.PSECT _LIB$DATA,NOEXE, PIC,2
```

```
00000 LIB$$GL_IMAGELIST::
```

```
.BLKB 4
```

```
.PSECT _LIB$CODE,NOWRT, SHR, PIC,2
```

```
OE 00000 P.AAA: .BYTE 14
```

45 58 45 2E 3A 45 52 41 48 53 24 53 59 53 00001

.ASCII \SYSS\$SHARE:.EXE\

SYSSHR_CSTRING=

P.AAA

```
.EXTRN LIB$M_LNK_1MOD, LIB$GET_EF
.EXTRN LIB$FREE_EF, LIB$GET_VM
.EXTRN LIB$FREE_VM, LIB$INSERT_TREE
.EXTRN LIB$LOOKUP_TREE
.EXTRN LIB$SCOPY_R_DX, STR$FREE1_DX
.EXTRN STR$UPCASE, LIB$GETFILENAME
.EXTRN LIB$$REPORT_IO_ERROR
.EXTRN LIB$$READ_OBJECT
.EXTRN SYSS$GET
```

		000C 00000	GET_REC: .WORD	Save R2,R3	: 0247
50	04	AC D0 00002	MOVL	DATAVECTOR, R0	: 0267
52	04	A0 D0 00006	MOVL	4(R0), R2	: 0268
	00000000G	00 9F 0000A	PUSHAB	LIB\$\$REPORT_IO_ERROR	: 0276
		52 DD 00010	PUSHL	R2	
00000000G	00	02 FB 00012	CALLS	#2, SYSS\$GET	
53		50 D0 00019	MOVL	R0, STATUS	
27		53 E9 0001C	BLBC	STATUS, 1\$	: 0277
	08	AC DD 0001F	PUSHL	RECDISC	: 0280
	28	A2 DD 00022	PUSHL	40(R2)	
	22	A2 9F 00025	PUSHAB	34(R2)	
00000000G	00	03 FB 00028	CALLS	#3, LIB\$SCOPY_R_DX	
53		50 D0 0002F	MOVL	R0, STATUS	
15		53 E8 00032	BLBS	STATUS, 2\$	: 0281
		53 DD 00035	PUSHL	STATUS	: 0283
		7E D4 00037	CLRL	-(SP)	
	001512F2	8F DD 00039	PUSHL	#1381106	
00000000G	00	03 FB 0003F	CALLS	#3, LIB\$SIGNAL	
50		53 D0 00046	MOVL	STATUS, R0	: 0284
		04 00049	RET		
50		01 D0 0004A	MOVL	#1, R0	: 0287
		04 0004D	RET		: 0288

; Routine Size: 78 bytes, Routine Base: _LIB\$CODE + 000F


```

198      0289 1 %SBTTL 'compare_symbols - Compare new symbol to node';
199      0290 1 ROUTINE compare_symbols(desc,node) =
200      0291 2 BEGIN
201      0292 2 !++
202      0293 2 FUNCTIONAL DESCRIPTION:
203      0294 2
204      0295 2 Compare symbol against symbol in current node
205      0296 2
206      0297 2 INPUTS:
207      0298 2
208      0299 2 desc = address of descriptor of new symbol
209      0300 2 node = pointer to an MSD block
210      0301 2
211      0302 2 --
212      0303 2 MAP
213      0304 2 desc : REF $BBLOCK,
214      0305 2 node : REF $BBLOCK FIELD(msd_fields);
215      0306 2
216      0307 2 LOCAL
217      0308 2 retval;
218      0309 2
219      0310 3 IF (retval = CH$COMPARE(.desc[dsc$w_length],.desc[dsc$a_pointer],
220      0311 2 .node[msd_b_namlen],node[msd_t_name],0)) NEQ 0
221      0312 2 THEN RETURN .retval
222      0313 2 ELSE RETURN SIGN(.desc[dsc$w_length] - .node[msd_b_namlen])
223      0314 1 END;

```

				007C 00000 COMPARE_SYMBOLS:				
			56	04	AC D0 00002	.WORD	Save R2,R3,R4,R5,R6	: 0290
			54	08	AC D0 00006	MOVL	DESC, R6	: 0310
			50	0E	A4 9A 0000A	MOVL	NODE, R4	: 0311
			55		01 D0 0000E	MOVZBL	14(R4), R0	:
50	00	04	B6		66 2D 00011	MOVL	#1, R5	:
				0F	A4 00017	CMPC5	(R6), @4(R6), #0, R0, 15(R4)	:
					03 1A 00019	BGTRU	1\$	:
			55		01 D9 0001B	SBWC	#1, R5	:
			50		55 D0 0001E 1\$:	MOVL	R5, RETVAL	:
					15 12 00021	BNEQ	2\$	:
			54	0E	A4 9A 00023	MOVZBL	14(R4), R4	: 0313
			50		66 3C 00027	MOVZWL	(R6), R0	:
			54		50 C2 0002A	SUBL2	R0, R4	:
					54 D5 0002D	TSTL	R4	:
			50		50 DC 0002F	MOVPSL	R0	:
50	50	02			02 EF 00031	EXTZV	#2, #2, R0, R0	:
					50 D7 00036	DECL	R0	:
					04 00038 2\$:	RET		: 0314

```
; Routine Size: 57 bytes,    Routine Base: _LIB$CODE + 005D
```

```
225 0315 1 %SBTTL 'alloc_symbol - Allocate MSD block for new symbol';
226 0316 1 ROUTINE alloc_symbol(desc,retadr) =
227 0317 2 BEGIN
228 0318 2 ++
229 0319 2 FUNCTIONAL DESCRIPTION:
230 0320 2
231 0321 2 Allocate an MSD block for new symbol
232 0322 2
233 0323 2 INPUTS:
234 0324 2
235 0325 2 desc = address of string descriptor of new symbol
236 0326 2 retadr = address of longword to return address in
237 0327 2
238 0328 2 --
239 0329 2 MAP
240 0330 2 desc : REF $BBLOCK,
241 0331 2 retadr : REF VECTOR[,LONG];
242 0332 2
243 0333 2 LOCAL
244 0334 2 status,
245 0335 2 msdptr : REF $BBLOCK FIELD (msd_fields);
246 0336 2
247 0337 2 status = lib$get_vm(%REF(msd_c_size+.desc[dsc$w_length]),msdptr);
248 0338 2 IF NOT .status
249 0339 2 THEN BEGIN
250 0340 2 SIGNAL(lib$insvirmem,0,.status);
251 0341 2 RETURN .status
252 0342 2 END;
253 0343 2
254 0344 2 msdptr[msd_l_value] = 0;
255 0345 2 msdptr[msd_b_namlen] = .desc[dsc$w_length];
256 0346 2 CH$MOVE(.msdptr[msd_b_namlen],.desc[dsc$a_pointer],msdptr[msd_t_name]);
257 0347 2 retadr[0] = .msdptr;
258 0348 2
259 0349 2 RETURN ss$_normal
260 0350 1 END;
```

007C 00000 ALLOC_SYMBOL:						
	5E		08 C2 00002	.WORD	Save R2,R3,R4,R5,R6	: 0316
		04	AE 9F 00005	SUBL2	#8, SP	
		04	AC D0 00008	PUSHAB	MSDPTR	: 0337
	04 AE		62 3C 0000C	MOVL	DESC, R2	
	04 AE		0F C0 00010	MOVZWL	(R2), 4(SP)	
		04	AE 9F 00014	ADDL2	#15, 4(SP)	
00000000G	00		02 FB 00017	PUSHAB	4(SP)	
	53		50 D0 0001E	CALLS	#2, LIB\$GET_VM	
	15		53 E8 00021	MOVL	R0, STATUS	
			53 DD 00024	BLBS	STATUS, 1\$	: 0338
			53 DD 00024	PUSHL	STATUS	: 0340
			7E D4 00026	CLRL	-(SP)	
00000000G	00	001512F2	8F DD 00028	PUSHL	#1381106	
	50		03 FB 0002E	CALLS	#3, LIB\$SIGNAL	
			53 D0 00035	MOVL	STATUS, R0	: 0341

LIB\$FIND_IMAGE
V03-006

Dynamically activate shareable image
alloc_symbol - Allocate MSD block for new symbo

D 5
16-Sep-1984 00:56:42
14-Sep-1984 12:38:55

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]LIBFNDIMG.B32;1

Page 9
(5)

		56	04	AE	D0	00038	1\$:	RET		:	
			0A	A6	D4	0003D		MOVL	MSDPTR, R6	:	0344
		0E	A6	62	90	00040		CLRL	10(R6)	:	
								MOVB	(R2), 14(R6)	:	0345
		04	B2	0E	A6	9A	00044	MOVZBL	14(R6), R0	:	0346
OF	A6				50	28	00048	MOVC3	R0, @4(R2), 15(R6)	:	
		08	BC		56	D0	0004E	MOVL	R6, @RETADR	:	0347
			50		01	D0	00052	MOVL	#1, R0	:	0349
					04	00055		RET		:	0350

; Routine Size: 86 bytes, Routine Base: _LIB\$CODE + 0096

```
262 0351 1 %SBTTL 'global_routine - Process global symbol from GST';
263 0352 1 ROUTINE global_routine(symbol_desc,symbol_value,symbol_flags,
264 0353 1 entry_mask,datavector,gsd_desc) =
265 0354 2 BEGIN
266 0355 2 ++
267 0356 2 FUNCTIONAL DESCRIPTION:
268 0357 2
269 0358 2 This routine called with global symbol from shareable image. A
270 0359 2 block describing the symbol is allocated and put in the symbols
271 0360 2 list for this image.
272 0361 2
273 0362 2 --
274 0363 2 MAP
275 0364 2 symbol_desc : REF $BBLOCK,
276 0365 2 symbol_value : REF VECTOR[.LONG],
277 0366 2 symbol_flags : REF VECTOR[.LONG],
278 0367 2 datavector : REF VECTOR[.LONG];
279 0368 2
280 0369 2 BIND
281 0370 2 midptr = .datavector[0] : $BBLOCK FIELD(mid_fields);
282 0371 2
283 0372 2 LOCAL
284 0373 2 msdptr : REF $BBLOCK FIELD(msd_fields),
285 0374 2 status;
286 0375 2
287 0376 2 status = lib$insert_tree(midptr[mid_l_symlst],.symbol_desc,%REF(1),
288 0377 2 compare_symbols,alloc_symbol,msdptr);
289 0378 2 IF NOT .status
290 0379 2 THEN RETURN .status;
291 0380 2
292 0381 2 ! Set symbol value. Relocate if needed
293 0382 2 !
294 0383 2 msdptr[msd_l_value] = .symbol_value[0];
295 0384 2 IF (.symbol_flags[0] AND gsy$m_rel) NEQ 0
296 0385 2 THEN msdptr[msd_l_value] = .msdptr[msd_l_value] + .midptr[mid_l_base];
297 0386 2
298 0387 2 RETURN ss$_normal
299 0388 1 END;
```

0004 00000 GLOBAL_ROUTINE:

5E	08	C2	00002	.WORD	Save R2	0352
52	14	BC	D0 00005	SUBL2	#8, SP	0370
	04	AE	9F 00009	MOVL	@DATAVECTOR, R2	0376
	9B	AF	9F 0000C	PUSHAB	MSDPTR	
	FF5E	CF	9F 0000F	PUSHAB	ALLOC SYMBOL	
0C	AE	01	D0 00013	PUSHAB	COMPARE SYMBOLS	
	0C	AE	9F 00017	MOVL	#1, 12(SP)	
	04	AC	DD 0001A	PUSHAB	12(SP)	
	04	A2	9F 0001D	PUSHL	SYMBOL_DESC	
00000000G	00	06	FB 00020	PUSHAB	4(R2)	
	16	55	E9 00027	CALLS	#6, LIB\$INSERT_TREE	0378
	50	AE	D0 0002A	BLBC	STATUS, 2\$	0383
				MOVL	MSDPTR, R0	

LIB\$FIND_IMAGE	Dynamically activate shareable image	F 5	16-Sep-1984 00:56:42	VAX-11 Bliss-32 V4.0-742	Page 11
V03-006	global_routine - Process global symbol from GST	14-Sep-1984 12:38:55	[LIBRTL.SRC]LIBFNDIMG.B32;1	(6)	

05	0A	A0	08	BC	D0	0002E	MOVL	@SYMBOL VALUE, 10(R0)	:	
	0C	BC		03	E1	00033	BBC	#3, @SYMBOL FLAGS, 1\$	:	0384
	0A	A0	08	A2	C0	00038	ADDL2	8(R2), 10(R0)	:	0385
		50		01	D0	0003D	MOVL	#1, R0	:	0387
				04	00040	2\$:	RET		:	0388

; Routine Size: 65 bytes, Routine Base: _LIB\$CODE + 00EC

```
301 0389 1 %SBTTL 'read_gst - Read GST from image';
302 0390 1 ROUTINE read_gst (ifdptr,midptr) =
303 0391 2 BEGIN
304 0392 2 ++
305 0393 2 FUNCTIONAL DESCRIPTION:
306 0394 2
307 0395 2 Read the GST of the image just loaded
308 0396 2
309 0397 2 INPUTS:
310 0398 2
311 0399 2 ifdptr = address of the image IFD (image file descriptor)
312 0400 2 midptr = address of MID block for image
313 0401 2 --
314 0402 2 MAP
315 0403 2 ifdptr : REF $BBLOCK,
316 0404 2 midptr : REF $BBLOCK FIELD(mid_fields);
317 0405 2
318 0406 2 LOCAL
319 0407 2 status,
320 0408 2 datavector : VECTOR[2, LONG],
321 0409 2 ubuf : $BBLOCK[obj$sc_maxrecsiz],
322 0410 2 ifab : $FAB_DECL,
323 0411 2 irab : $RAB_DECL,
324 0412 2 inam : $NAM_DECL,
325 0413 2 esbuf : $BBLOCK[nam$sc_maxrss];
326 0414 2
327 P 0415 2 $FAB_INIT(FAB=ifab,
328 P 0416 2 FNS=.midptr[mid_b_namlen],
329 P 0417 2 FNA=midptr[mid_f_name],
330 P 0418 2 DNS=.sysshr_cstring[0],
331 P 0419 2 DNA=sysshr_cstring[1],
332 P 0420 2 FAC=(BRO,GET),
333 P 0421 2 NAM=inam,
334 0422 2 SHR=(GET,PUT,UPI));
335 0423 2
336 0424 2 IF .ifdptr[ifd$sc_priv]
337 0425 2 THEN ifab[ifab$sc_lnm_mode] = psl$sc_exec;
338 0426 2
339 P 0427 2 $NAM_INIT(NAM=inam,
340 P 0428 2 RSS=nam$sc_maxrss,
341 P 0429 2 RSA=esbuf,
342 P 0430 2 ESS=nam$sc_maxrss,
343 0431 2 ESA=esbuf);
344 0432 2
345 0433 2 ifab[ifab$sc_ctx] = lib$openin;
346 0434 2 status = $OPEN(FAB=ifab,ERR=lib$$report_io_error);
347 0435 2 IF NOT .status
348 0436 2 THEN RETURN .status;
349 0437 2
350 P 0438 2 $RAB_INIT(RAB=irab,
351 P 0439 2 FAB=ifab,
352 P 0440 2 UBF=ubuf,
353 P 0441 2 USZ=obj$sc_maxrecsiz,
354 0442 2 ROP=LOC);
355 0443 2
356 0444 2 irab[irab$sc_ctx] = lib$openin;
357 0445 2 status = $CONNECT(RAB=irab,ERR=lib$$report_io_error);
```



```
358 0446 2 IF NOT .status
359 0447 2 THEN BEGIN
360 0448 2     $CLOSE(FAB=ifab);
361 0449 2     RETURN .status
362 0450 2 END;
363 0451 2
364 0452 2 Tell RMS that records are variable length, so we can read the GST
365 0453 2
366 0454 2 ifab[fab$v_esc] = 1;
367 0455 2 ifab[fab$l_ctx] = rme$c_setrfm;
368 0456 2 ifab[fab$b_rfm] = fab$c_var;
369 0457 2 status = $MODIFY(FAB=ifab,ERR=lib$$report_io_error);
370 0458 2 IF NOT .status
371 0459 2 THEN BEGIN
372 0460 2     $CLOSE(FAB=ifab);
373 0461 2     RETURN .status
374 0462 2 END;
375 0463 2
376 0464 2 Point to and read the GST
377 0465 2
378 0466 2 irab[rab$l_rfa0] = .midptr[mid_l_vbn];
379 0467 2 irab[rab$b_rfa4] = 0;
380 0468 2 irab[rab$b_rac] = rab$c_rfa;
381 0469 2 irab[rab$l_ctx] = lib$_readerr;
382 0470 2 status = $FIND(RAB=irab,ERR=lib$$report_io_error);
383 0471 2 IF NOT .status
384 0472 2 THEN BEGIN
385 0473 2     $CLOSE(FAB=ifab);
386 0474 2     RETURN .status
387 0475 2 END;
388 0476 2 irab[rab$b_rac] = rab$c_seq;
389 0477 2
390 0478 2 datavector[0] = .midptr;
391 0479 2 datavector[1] = irab;
392 0480 2
393 0481 2 status = lib$$read_object(get_rec,%REF(lib$m_lnk_1mod),
394 0482 2                               datavector,globa[ routine]);
395 0483 2 ifab[fab$l_ctx] = lib$_closein;
396 0484 2 $CLOSE(FAB=ifab,ERR=lib$$report_io_error);
397 0485 2
398 0486 2 RETURN .status
399 0487 1 END;
```

```
.EXTRN SYSSOPEN, SYSSCONNECT
.EXTRN SYSSCLOSE, SYSSMODIFY
.EXTRN SYSSFIND
```

07FC 00000 READ_GST:

5A	FECE	CF	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10	: 0390
59	00000000G	00	9E	00007	MOVAB	SYSSHR CSTRING+1, R10	
58	00000000G	00	9E	0000E	MOVAB	SYSSCLOSE, R9	
5E	F600	CE	9E	00015	MOVAB	LIB\$\$REPORT_IO_ERROR, R8	
6E	01A8	CE	2C	0001A	MOVAB	-2560(SP), SP	
				00021	MOVCS	#0, (SP), #0, #80, \$RMS_PTR	: 0422

01F2	CE	07	10	01A8	CE	5003	8F	B0	00024	MOVW	#20483, \$RMS_PTR	...
0060	8F	02		01BE	CE	4342	8F	B0	0002B	MOVW	#17218, \$RMS_PTR+22	...
		00		01C7	CE		02	90	00032	MOVB	#2, \$RMS_PTR+31	...
				01D0	CE	0104	CE	9E	00037	MOVAB	INAM, \$RMS_PTR+40	...
					57	08	AC	DO	0003E	MOVL	MIDPTR, R7	...
				01D4	CE	11	A7	9E	00042	MOVAB	17(R7), \$RMS_PTR+44	...
				01D8	CE		6A	9E	00048	MOVAB	SYSSHR_CSTRING+1, \$RMS_PTR+48	...
				01DC	CE	10	A7	90	0004D	MOVB	16(R7), \$RMS_PTR+52	...
				01DD	CE	FF	AA	90	00053	MOVB	SYSSHR_CSTRING, \$RMS_PTR+53	...
					50	04	AC	DO	00059	MOVL	IFDPT, R0	0424
					A0		01	E1	0005D	BBC	#1, 16(R0), 1\$	...
					00		01	F0	00062	INSV	#1, #0, #2, IFAB+74	0425
					6E		00	2C	00069	MOVCS	#0, (SP), #0, #96, \$RMS_PTR	0431
						0104	CE		00070			...
						6002	8F	B0	00073	MOVW	#24578, \$RMS_PTR	...
				0104	CE		01	8E	0007A	MNEGB	#1, \$RMS_PTR+2	...
				0106	CE		AE	9E	0007F	MOVAB	ESBUF, \$RMS_PTR+4	...
				0108	CE	04	01	8E	00085	MNEGB	#1, \$RMS_PTR+10	...
				010E	CE		AE	9E	0008A	MOVAB	ESBUF, \$RMS_PTR+12	...
				0110	CE	04	8F	DO	00090	MOVL	#1380506, IFAB+24	0433
				01C0	CE	0015109A	58	DD	00099	PUSHL	R8	0434
						01AC	CE	9F	0009B	PUSHAB	IFAB	...
				00000000G	00		02	FB	0009F	CALLS	#2, SYSS\$OPEN	...
					56		50	DO	000A6	MOVL	R0, STATUS	...
					03		56	E8	000A9	BLBS	STATUS, 2\$	0435
							00DE	31	000AC	BRW	5\$	...
0044	8F	00			6E		00	2C	000AF	MOVCS	#0, (SP), #0, #68, \$RMS_PTR	0442
						0164	CE		000B6			...
				0164	CE	4401	8F	B0	000B9	MOVW	#17409, \$RMS_PTR	...
				0168	CE	00010000	8F	DO	000C0	MOVL	#65536, \$RMS_PTR+4	...
				0184	CE	0800	8F	B0	000C9	MOVW	#2048, \$RMS_PTR+32	...
				0188	CE	01F8	CE	9E	000D0	MOVAB	UBUF, \$RMS_PTR+36	...
				01A0	CE	01A8	CE	9E	000D7	MOVAB	IFAB, \$RMS_PTR+60	...
				017C	CE	0015109A	8F	DO	000DE	MOVL	#1380506, IRAB+24	0444
						0168	58	DD	000E7	PUSHL	R8	0445
							CE	9F	000E9	PUSHAB	IRAB	...
				00000000G	00		02	FB	000ED	CALLS	#2, SYSS\$CONNECT	...
					56		50	DO	000F4	MOVL	R0, STATUS	...
					4D		56	E9	000F7	BLBC	STATUS, 3\$	0446
				01AF	CE		08	88	000FA	BISB2	#8, IFAB+7	0454
				01C0	CE		01	DO	000FF	MOVL	#1, IFAB+24	0455
				01C7	CE		02	90	00104	MOVB	#2, IFAB+31	0456
						01AC	58	DD	00109	PUSHL	R8	0457
							CE	9F	0010B	PUSHAB	IFAB	...
				00000000G	00		02	FB	0010F	CALLS	#2, SYSS\$MODIFY	...
					56		50	DO	00116	MOVL	R0, STATUS	...
					2B		56	E9	00119	BLBC	STATUS, 3\$	0458
				0174	CE	0C	A7	DO	0011C	MOVL	12(R7), IRAB+16	0466
						0178	CE	B4	00122	CLRW	IRAB+20	0467
				0182	CE		02	90	00126	MOVB	#2, IRAB+30	0468
				017C	CE	001510B2	8F	DO	0012B	MOVL	#1380530, IRAB+24	0469
						0168	58	DD	00134	PUSHL	R8	0470
							CE	9F	00136	PUSHAB	IRAB	...
				00000000G	00		02	FB	0013A	CALLS	#2, SYSS\$FIND	...
					56		50	DO	00141	MOVL	R0, STATUS	...
					09		56	E8	00144	BLBS	STATUS, 4\$	0471
						01A8	CE	9F	00147	PUSHAB	IFAB	0473

LIB\$FIND_IMAGE Dynamically activate shareable image
V03-006 read_gst - Read GST from image

J 5
16-Sep-1984 00:56:42
14-Sep-1984 12:38:55

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]LIBFNDIMG.B32;1

Page 15
(7)

69	01	FB	0014B	CALLS	#1, SYSS\$CLOSE	:	
	3D	11	0014E	BRB	5\$	:	0474
	CE	94	00150	CLRB	IRAB+30	:	0476
F8	AD	57	D0 00154	MOVL	R7, DATAVECTOR	:	0478
FC	AD	CE	9E 00158	MOVAB	IRAB, DATAVECTOR+4	:	0479
	FE5D	CF	9F 0015E	PUSHAB	GLOBAL ROUTINE	:	0481
	F8	AD	9F 00162	PUSHAB	DATAVECTOR	:	
08	AE	8F	9A 00165	MOVZBL	#LIB\$M_LNK_1MOD, 8(SP)	:	
	08	AE	9F 0016A	PUSHAB	8(SP)	:	
	FD71	CF	9F 0016D	PUSHAB	GET_REC	:	
00000000G	00	04	FB 00171	CALLS	#4, LIB\$\$READ_OBJECT	:	
	56	50	D0 00178	MOVL	R0, STATUS	:	
01C0	CE	8F	D0 0017B	MOVL	#1380432, IFAB+24	:	0483
		58	DD 00184	PUSHL	R8	:	0484
	01AC	CE	9F 00186	PUSHAB	IFAB	:	
69		02	FB 0018A	CALLS	#2, SYSS\$CLOSE	:	
50		56	D0 0018D	MOVL	STATUS, R0	:	0486
		04	00190	RET		:	0487

; Routine Size: 401 bytes, Routine Base: _LIB\$CODE + 012D

```

: 401 0488 1 %SBTTL 'find_image - Find image in mapped images list';
: 402 0489 1 ROUTINE find_image(desc,midptr) =
: 403 0490 2 BEGIN
: 404 0491 1 ++
: 405 0492 2 FUNCTIONAL DESCRIPTION:
: 406 0493 2
: 407 0494 2 Look up image in mapped image list
: 408 0495 2
: 409 0496 2 INPUTS:
: 410 0497 2
: 411 0498 2 desc = address of descriptor of image name
: 412 0499 2 midptr = address of longword to store MID block address if found
: 413 0500 2 --
: 414 0501 2 MAP
: 415 0502 2 desc : REF $BBLOCK,
: 416 0503 2 midptr : REF VECTOR[,LONG];
: 417 0504 2
: 418 0505 2 LOCAL
: 419 0506 2 ptr : REF $BBLOCK FIELD(mid_fields);
: 420 0507 2
: 421 0508 2 ptr = lib$$gl_imagelist;
: 422 0509 2
: 423 0510 2 WHILE (ptr = .ptr[mid_l_next]) NEQ 0
: 424 0511 2 DO IF CH$EQL(.desc[dsc$w_length],.desc[dsc$a_pointer],
: 425 0512 2 .ptr[mid_b_namlen],ptr[mid_t_name],0)
: 426 0513 3 THEN BEGIN
: 427 0514 3 midptr[0] = .ptr;
: 428 0515 3 RETURN ss$_normal
: 429 0516 2 END;
: 430 0517 2
: 431 0518 2 RETURN 0
: 432 0519 1 END;
```

003C 00000 FIND_IMAGE:										
			54	00000000'	EF	9E	00002	.WORD	Save R2,R3,R4,R5	0489
			55	04	AC	D0	00009	MOVAB	LIB\$\$GL_IMAGELIST, PTR	0508
			54		64	D0	0000D	MOVL	DESC, R5	0511
					17	13	00010	MOVL	(PTR), PTR	0510
					17	13	00010	BEQL	2\$	
			50	10	A4	9A	00012	MOVZBL	16(PTR), R0	0512
50		00	04	B5	04	BC	2D	CMPC5	@DESC, @4(R5), #0, R0, 17(PTR)	
				11	A4		0001D			
					EC	12	0001F	BNEQ	1\$	
		08	BC		54	D0	00021	MOVL	PTR, @MIDPTR	0514
			50		01	D0	00025	MOVL	#1, R0	0515
						04	00028	RET		
					50	D4	00029	CLRL	R0	0518
						04	0002B	RET		0519

; Routine Size: 44 bytes, Routine Base: _LIB\$CODE + 02BE


```

434 0520 1 %SBTTL 'lib$find_image_symbol - Find symbol by string name';
435 0521 1 GLOBAL ROUTINE lib$find_image_symbol (image_desc,routine_desc,retadr) =
436 0522 2 BEGIN
437 0523 2 ++
438 0524 2 FUNCTIONAL DESCRIPTION:
439 0525 2
440 0526 2     Return the address of the named routine.
441 0527 2
442 0528 2 INPUTS:
443 0529 2
444 0530 2     image_desc = Address of string descriptor for image name. A
445 0531 2     default file specification of SYSS$SHARE:.EXE is applied.
446 0532 2     routine_desc = Address of string descriptor of symbol to find
447 0533 2     retadr = Address to return symbol value in
448 0534 2
449 0535 2 OUTPUTS:
450 0536 2
451 0537 2     If the image can be found and the symbol is found in the image, the
452 0538 2     relocated (if needed) symbol value will be returned in the longword
453 0539 2     pointed to by retadr.
454 0540 2
455 0541 2 ROUTINE VALUE:
456 0542 2
457 0543 2     ss$_normal = found, value returned ok
458 0544 2     any errors from services called
459 0545 2
460 0546 2 NOTES:
461 0547 2
462 0548 2     Use of these routines requires READ access to the shareable image file,
463 0549 2     because the file must be open to read the global symbols contained within.
464 0550 2
465 0551 2     LIB$FIND_IMAGE_SYMBOL signals all errors detected, such as image file not found,
466 0552 2     and format errors in the global symbol table of the image.
467 0553 2
468 0554 2     After the first call to LIB$FIND IMAGE SYMBOL for a particular image,
469 0555 2     successive calls for that image will be quite fast, as an in-memory database
470 0556 2     is maintained so that the image is only activated once.
471 0557 2
472 0558 2     If the calling image is installed with privileges, then the image being
473 0559 2     activated must be installed. Further, only logical names defined
474 0560 2     /SYSTEM /EXEC will be considered while activating the image.
475 0561 2
476 0562 2     There is no way to deallocate this database, nor is there any supported
477 0563 2     method to remove an activated image from the address space. All activated
478 0564 2     images are activated into P0 space.
479 0565 2
480 0566 2 EXAMPLES:
481 0567 2
482 0568 2 a) Using LIB$FIND_IMAGE_SYMBOL from BLISS
483 0569 2
484 0570 2     STATUS = LIB$FIND_IMAGE_SYMBOL($DESCRIPTOR('EDTSHR'),$DESCRIPTOR('EDT$EDIT'),EDTADR);
485 0571 2     (.EDTADR)($DESCRIPTOR('FOO.INP'),$DESCRIPTOR('FOO.OUT'));
486 0572 2
487 0573 2 b) Using LIB$FIND_IMAGE_SYMBOL from FORTRAN
488 0574 2
489 0575 2     CALL LIB$FIND_IMAGE_SYMBOL('EDTSHR','EDT$EDIT',EDTADR)
490 0576 2     CALL USER_CALC_SYMBOL(EDTADR,%VAL(#args),arg1, arg2, ...)
```

```
491 0577 2 |
492 0578 2 | where USER_CALL_SYMBOL is the following short MACRO routine
493 0579 2 |
494 0580 2 | .ENTRY USER_CALL_SYMBOL,0
495 0581 2 |
496 0582 2 |     MOVL    @4(AP),R0                ;Get address to call
497 0583 2 |     CALLG   8(AP),(R0)              ;Call the routine
498 0584 2 |     RET                                ;Return with status in R0
499 0585 2 |
500 0586 2 |
501 0587 2 | MAP
502 0588 2 |     image_desc : REF $BBLOCK,
503 0589 2 |     routine_desc : REF $BBLOCK,
504 0590 2 |     retradr : REF VECTOR[.LONG];
505 0591 2 |
506 0592 2 | LOCAL
507 0593 2 |     status,
508 0594 2 |     tempvec : REF VECTOR[.LONG],
509 0595 2 |     d_desc : $BBLOCK[dsc$sc_s_bln],
510 0596 2 |     d_desc : VECTOR[2,.LONG],
511 0597 2 |     ptr : REF $BBLOCK,
512 0598 2 |     ifdptr : REF $BBLOCK,
513 0599 2 |     midptr : REF $BBLOCK FIELD(mid_fields),
514 0600 2 |     msdptr : REF $BBLOCK FIELD(msd_fields),
515 0601 2 |     dflnam : VECTOR[2,.LONG],
516 0602 2 |     adrvec : VECTOR[2,.LONG];
517 0603 2 |
518 0604 2 | IF NOT find_image(.image_desc,midptr)
519 0605 2 | THEN BEGIN
520 0606 2 |
521 0607 2 |     Image not yet mapped. Attempt to map it and then read in the GST
522 0608 2 |
523 0609 4 |     IF NOT (status = lib$get_vm(%REF(512),midptr))
524 0610 4 |     THEN BEGIN
525 0611 4 |         SIGNAL(lib$insvirmem,0,.status);
526 0612 4 |         RETURN .status
527 0613 4 |     END;
528 0614 4 |
529 0615 4 |     adrvec[0] = 1;
530 0616 4 |     adrvec[1] = 0;
531 0617 4 |     dflnam[0] = .sysshr_cstring[0];
532 0618 4 |     dflnam[1] = sysshr_cstring[1];
533 0619 4 |     status = $IMGACT(NAME=.image_desc,
534 0620 4 |         DFLNAM=dflnam,
535 0621 4 |         HDRBUF=.midptr,
536 0622 4 |         IMGCTL=iac$m_merge OR iac$m_expreg,
537 0623 4 |         INADR=adrvec,
538 0624 4 |         RETADR=adrvec);
539 0625 4 |
540 0626 4 |     IF .status
541 0627 4 |     THEN status = $IMGFIX;
542 0628 4 |     IF NOT .status
543 0629 4 |     THEN BEGIN
544 0630 4 |         tempvec = .midptr;
545 0631 4 |         ptr = .tempvec[2];                !Get FAB address
546 0632 4 |         IF .ptr NEQ 0
547 0633 4 |         THEN CH$MOVE(dsc$sc_s_bln,lib$$getfilename(.ptr),desc);
548 0634 4 |         IF .desc[0] EQL 0
```



```

548      0634 4      THEN CH$MOVE(dsc$c_s_bln,.image_desc,desc);
549      0635 4      CH$MOVE(.desc[0],.desc[1],.midptr);      !Copy the file spec
550      0636 4      desc[1] = .midptr;
551      0637 4      SIGNAL(lib$_actimage,1,desc,.status);
552      0638 4      lib$free_vm(%REF(512),midptr);
553      0639 4      RETURN .status
554      0640 4      END;
555      0641 4      |
556      0642 4      | Image is now mapped. Insert into the mapped image list ad
557      0643 4      | then read the GST
558      0644 4      |
559      0645 4      | tempvec = .midptr;      !Get IFD address to get filespec
560      0646 4      | ptr = .tempvec[0];
561      0647 4      | ifdptr = .tempvec[1];
562      0648 4      | midptr[mid_l_next] = .lib$$gl_imagelist;
563      0649 4      | lib$$gl_imagelist = .midptr;
564      0650 4      | midptr[mid_l_symlst] = 0;
565      0651 4      | midptr[mid_l_base] = .adrvec[0];
566      0652 4      |
567      0653 4      | If image activator did not return image header, do a qio
568      0654 4      | and read it for ourselves
569      0655 4      |
570      0656 4      IF .ptr NEQ 0
571      0657 4      THEN BEGIN
572      0658 4          ptr = .ptr + .ptr[ihd$w_syndbgoff];
573      0659 4          midptr[mid_l_vbn] = .ptr[ihs$_gstvbn];
574      0660 4          END
575      0661 4      ELSE BEGIN
576      0662 4          LOCAL
577      0663 4              efnun,
578      0664 4              ihdbuf : $BBLOCK[512],
579      0665 4              qiosb : VECTOR[4,WORD];
580      0666 4
581      0667 4          status = lib$get_ef(efnun);
582      0668 4          IF NOT .status
583      0669 5              THEN BEGIN
584      0670 5                  SIGNAL(.status);
585      0671 5                  RETURN .status
586      0672 4              END;
587      0673 4          status = $QIOW(chan=.ifdptr[ifd$w_chan],
588      0674 4                      efn = .efnun,
589      0675 4                      func=IO$_READVBLK,
590      0676 4                      iosb=qiosb,
591      0677 4                      p1=ihdbuf,
592      0678 4                      p2=512,
593      0679 4                      p3=1);
594      0680 4          IF .status
595      0681 4              THEN status = .qiosb[0];
596      0682 4
597      0683 4          lib$free_ef(efnun);
598      0684 4          IF NOT .status
599      0685 5              THEN BEGIN
600      0686 5                  SIGNAL(lib$_readerr,1,.image_desc,.status);
601      0687 5                  RETURN .status
602      0688 4              END;
603      0689 4          ptr = ihdbuf + .ihdbuf[ihd$w_syndbgoff];
604      0690 4          midptr[mid_l_vbn] = .ptr[ihs$_gstvbn];
```

```

605 0691      END;
606 0692
607 0693      Save the user-specified name in the mid-block
608 0694
609 0695      midptr[mid_b_namlen] = .image_desc[dsc$w_length];
610 0696      CH$MOVE(.image_desc[dsc$w_length],.image_desc[dsc$a_pointer],
611 0697      midptr[mid_t_name]);
612 0698      read_gst(.ifdptr,.midptr);
613 0699      END;
614 0700
615 0701      Image is mapped, lookup and return symbol value
616 0702
617 0703      d_desc[dsc$w_length] = 0;
618 0704      d_desc[dsc$b_class] = dsc$k_class_d;
619 0705      d_desc[dsc$b_dtype] = dsc$k_dtype_t;
620 0706      d_desc[dsc$a_pointer] = 0;
621 0707      str$upcase(d_desc,.routine_desc);
622 0708      status = lib$lookup_tree(midptr[mid_l_symlst],d_desc,
623 0709      compare_symbols,s,msdptr);
624 0710      str$free1_dx(d_desc);
625 0711      IF .status
626 0712      THEN BEGIN
627 0713          retradr[0] = .msdptr[msd_l_value];
628 0714          RETURN ss$_normal
629 0715      END
630 0716      ELSE BEGIN
631 0717          SIGNAL((.status AND NOT sts$m_severity) OR sts$k_error);
632 0718          retradr[0] = 0;
633 0719          RETURN .status
634 0720      END;
635 0721
636 0722 1 END;
```

				OFFC 00000		.EXTRN SYSS\$IMGACT, SYSS\$IMGFIX	
						.EXTRN SYSS\$QIOW	
						.ENTRY LIB\$FIND_IMAGE_SYMBOL, Save R2,R3,R4,R5,R6,-	0521
						R7,R8,R9,R10,R11	
						MOVAB LIB\$SIGNAL, R11	
						MOVAB -568(SP), SP	
						PUSHAB MIDPTR	0604
						MOVL IMAGE_DESC, R10	
						PUSHL R10	
						CALLS #2, FIND_IMAGE	
						BLBC R0, 1\$	
						BRW 14\$	
						PUSHAB MIDPTR	0609
						MOVZWL #512, 4(SP)	
						PUSHAB 4(SP)	
						CALLS #2, LIB\$GET_VM	
						MOVL R0, STATUS	
						BLBS STATUS, 2\$	
						PUSHL STATUS	0611
						CLRL -(SP)	
						PUSHL #1381106	

				00FC 00000	
5B	00000000G	00	9E	00002	
5E	FDC8	CE	9E	00009	
	04	AE	9F	0000E	
5A	04	AC	D0	00011	
		5A	DD	00015	
B9	AF	02	FB	00017	
03		50	E9	0001B	
		0166	31	0001E	
	04	AE	9F	00021	1\$:
04	AE	0200	8F	3C	00024
	04	AE	9F	0002A	
00000000G	00	02	FB	0002D	
	59	50	D0	00034	
	10	59	E8	00037	
		59	DD	0003A	
		7E	D4	0003C	
	001512F2	8F	DD	0003E	

	6B		03	FB	00044	CALLS	#3, LIB\$SIGNAL		
			0196	31	00047	BRW	16\$		0612
E0	AD		01	7D	0004A	MOVQ	#1, ADRVEC		0615
E8	AD	FCC4	CF	9A	0004E	MOVZBL	SYSSHR_CSTRING, DFLNAM		0617
EC	AD	FCBF	CF	9E	00054	MOVAB	SYSSHR_CSTRING+1, DFLNAM+4		0618
			7E	7C	0005A	CLRQ	-(SP)		0624
		E0	AD	9F	0005C	PUSHAB	ADRVEC		
		E0	AD	9F	0005F	PUSHAB	ADRVEC		
			30	DD	00062	PUSHL	#48		
		18	AE	DD	00064	PUSHL	MIDPTR		
		E8	AD	9F	00067	PUSHAB	DFLNAM		
			5A	DD	0006A	PUSHL	R10		
00000000G	00		08	FB	0006C	CALLS	#8, SYSS\$IMGACT		
	59		50	D0	00073	MOVL	R0, STATUS		0625
00000000G	00		59	E9	00076	BLBC	STATUS, 3\$		0626
	0D		00	FB	00079	CALLS	#0, SYSS\$IMGFIX		
	59		50	D0	00080	MOVL	R0, STATUS		
	53		59	E8	00083	BLBS	STATUS, 6\$		0627
	58	04	AE	D0	00086	MOVL	MIDPTR, TEMPVEC		0629
	57	08	A8	D0	0008A	MOVL	8(TEMPVEC), PTR		0630
			0E	13	0008E	BEQL	4\$		0631
			57	DD	00090	PUSHL	PTR		0632
00000000G	00		01	FB	00092	CALLS	#1, LIB\$\$GETFILENAME		
F0	AD		08	28	00099	MOVC3	#8, (R0), DESC		0633
	60	F0	AD	D5	0009E	TSTL	DESC		
			05	12	000A1	BNEQ	5\$		
			08	28	000A3	MOVC3	#8, (R10), DESC		0634
F0	AD		AD	28	000A8	MOVC3	DESC, @DESC+4, @MIDPTR		0635
04	BE	F4	AD	D0	000AF	MOVL	MIDPTR, DESC+4		0636
		F4	AD	DD	000B4	PUSHL	STATUS		0637
			AD	9F	000B6	PUSHAB	DESC		
			01	DD	000B9	PUSHL	#1		
		001512BA	8F	DD	000BB	PUSHL	#1381050		
	6B		04	FB	000C1	CALLS	#4, LIB\$SIGNAL		0638
		04	AE	9F	000C4	PUSHAB	MIDPTR		
04	AE	0200	8F	3C	000C7	MOVZWL	#512, 4(SP)		
		04	AE	9F	000CD	PUSHAB	4(SP)		
00000000G	00		02	FB	000D0	CALLS	#2, LIB\$FREE_VM		0639
			3F	11	000D7	BRB	8\$		0645
	56	04	AE	D0	000D9	MOVL	MIDPTR, R6		
	58		56	D0	000DD	MOVL	R6, TEMPVEC		
	57		68	7D	000E0	MOVQ	(TEMPVEC), PTR		0646
	66	00000000'	EF	D0	000E3	MOVL	LIB\$\$GL_IMAGELIST, (R6)		0648
00000000'	EF		56	D0	000EA	MOVL	R6, LIB\$\$GL_IMAGELIST		0649
		04	A6	D4	000F1	CLRL	4(R6)		0650
08	A6	E0	AD	D0	000F4	MOVL	ADRVEC, 8(R6)		0651
			57	D5	000F9	TSTL	PTR		0656
			06	13	000FB	BEQL	7\$		
	50	04	A7	3C	000FD	MOVZWL	4(PTR), R0		0658
			69	11	00101	BRB	13\$		
		08	AE	9F	00103	PUSHAB	EFNUM		0667
00000000G	00		01	FB	00106	CALLS	#1, LIB\$GET_EF		
	59		50	D0	0010D	MOVL	R0, STATUS		
	07		59	E8	00110	BLBS	STATUS, 9\$		0668
			59	DD	00113	PUSHL	STATUS		0670
	6B		01	FB	00115	CALLS	#1, LIB\$SIGNAL		
			48	11	00118	BRB	11\$		0671

			7E	7C	0011A	9\$:	CLRQ	-(SP)		0679	
			01	7D	0011C		MOVQ	#1, -(SP)			
			7E	8F	3C	0011F	MOVZWL	#512, -(SP)			
			2C	AE	9F	00124	PUSHAB	IHDBUF			
				7E	7C	00127	CLRQ	-(SP)			
			30	AE	9F	00129	PUSHAB	QIOSB			
				31	DD	0012C	PUSHL	#49			
			7E	08	A8	3C	0012E	MOVZWL	8(IFDPTR), -(SP)		
			34	AE	DD	00132	PUSHL	EFNUM			
00000000G	00			0C	FB	00135	CALLS	#12, SYSSQ10W			
	59			50	DD	0013C	MOVL	R0, STATUS		0680	
	04			59	E9	0013F	BLBC	STATUS, 10\$		0681	
	59			10	AE	3C	00142	MOVZWL	QIOSB, STATUS	0683	
00000000G	00			08	AE	9F	00146	PUSHAB	EFNUM		
	11				01	FB	00149	CALLS	#1, LIB\$FREE_EF	0684	
					59	E8	00150	BLBS	STATUS, 12\$	0686	
					59	DD	00153	PUSHL	STATUS		
					5A	DD	00155	PUSHL	R10		
					01	DD	00157	PUSHL	#1		
					8F	DD	00159	PUSHL	#1380530		
					04	FB	0015F	CALLS	#4, LIB\$SIGNAL		
					7C	11	00162	BRB	16\$	0687	
					AE	9E	00164	MOVAB	IHDBUF, R0	0689	
					AE	3C	00168	MOVZWL	IHDBUF+4, PTR		
					50	C0	0016C	ADDL2	R0, PTR		
					A7	D0	0016F	MOVL	4(PTR), 12(R6)	0690	
					6A	90	00174	MOVB	(R10), 16(R6)	0695	
11	A6				6A	28	00178	MOV3	(R10), @4(R10), 17(R6)	0697	
					56	DD	0017E	PUSHL	R6	0698	
					58	DD	00180	PUSHL	IFDPTR		
					02	FB	00182	CALLS	#2, READ GST		
					8F	D0	00187	MOVL	#34471938, D_DESC	0703	
					AD	D4	0018F	CLRL	D_DESC+4	0706	
					08	AC	DD	00192	PUSHL	ROUTINE_DESC	0707
					F8	AD	9F	00195	PUSHAB	D_DESC	
00000000G	00				02	FB	00198	CALLS	#2, STR\$UPCASE		
					OC	AE	9F	0019F	PUSHAB	MSDPTR	0708
					FBCD	CF	9F	001A2	PUSHAB	COMPARE_SYMBOLS	
					F8	AD	9F	001A6	PUSHAB	D_DESC	
					04	C1	001A9	ADDL3	#4, MIDPTR, -(SP)		
7E	10				04	FB	001AE	CALLS	#4, LIB\$LOOKUP_TREE		
00000000G	00				50	D0	001B5	MOVL	R0, STATUS		
					F8	AD	9F	001B8	PUSHAB	D_DESC	0710
00000000G	00				01	FB	001BB	CALLS	#1, STR\$FREE1_DX		
	0D				59	E9	001C2	BLBC	STATUS, 15\$	0711	
	50				AE	D0	001C5	MOVL	MSDPTR, R0	0713	
	OC				0A	D0	001C9	MOVL	10(R0), @RETADR		
	50				01	D0	001CE	MOVL	#1, R0	0716	
					04	001D1	RET				
50	59				07	CB	001D2	BICL3	#7, STATUS, R0	0717	
7E	50				02	C9	001D6	BISL3	#2, R0, -(SP)		
	6B				01	FB	001DA	CALLS	#1, LIB\$SIGNAL		
					BC	D4	001DD	CLRL	@RETADR	0718	
	50				59	D0	001E0	MOVL	STATUS, R0	0719	
					04	001E3	RET			0722	

; Routine Size: 484 bytes, Routine Base: _LIB\$CODE + 02EA

LIB\$FIND_IMAGE Dynamically activate shareable image
V03-006 lib\$find_image_symbol - Find symbol by string n

E 6
16-Sep-1984 00:56:42
14-Sep-1984 12:38:55

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]LIBFNDIMG.B32;1

Page 23
(9)

**

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
LIB\$DATA	4	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
LIB\$CODE	1230	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	125	0	1000	00:01.3

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:LIBFNDIMG/OBJ=OBJ\$:LIBFNDIMG MSRC\$:LIBFNDIMG/UPDATE=(ENH\$:LIBFNDIMG)

Size: 1215 code + 19 data bytes

Run Time: 00:17.3

Elapsed Time: 01:22.4

Lines/CPU Min: 2503

Lexemes/CPU-Min: 40857

Memory Used: 211 pages

Compilation Complete

0207 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

