



```

LL      NN      NN      KK      KK      PPPPPPP      RRRRRRR      000000      LL      IIIIII      88888888
LL      NN      NN      KK      KK      PPPPPPP      RRRRRRR      000000      LL      IIIIII      88888888
LL      NN      NN      KK      KK      PP      PP      RR      RR      00      00      LL      II      88      88
LL      NN      NN      KK      KK      PP      PP      RR      RR      00      00      LL      II      88      88
LL      NNNN     NN      KK      KK      PP      PP      RR      RR      00      00      LL      II      88      88
LL      NNNN     NN      KK      KK      PP      PP      RR      RR      00      00      LL      II      88      88
LL      NN      NN      NNNN     KK      KK      PPPPPPP      RRRRRRR      00      00      LL      II      88888888
LL      NN      NN      NNNN     KK      KK      PPPPPPP      RRRRRRR      00      00      LL      II      88888888
LL      NN      NNNN     KK      KK      PP      RR      RR      RR      00      00      LL      II      88      88
LL      NN      NNNN     KK      KK      PP      RR      RR      RR      00      00      LL      II      88      88
LL      NN      NN      KK      KK      PP      RR      RR      RR      00      00      LL      II      88      88
LL      NN      NN      KK      KK      PP      RR      RR      RR      00      00      LL      II      88      88
LLLLLLLLLLLL      NN      NN      KK      KK      PP      RR      RR      RR      000000      LLLLLLLLLL      IIIIII      88888888
LLLLLLLLLLLL      NN      NN      KK      KK      PP      RR      RR      RR      000000      LLLLLLLLLL      IIIIII      88888888

```

```

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
LLLLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLLLL      IIIIII      SSSSSSSS

```

```

1 0001 0 module lnk_proclib ( ! OBJECT LIBRARY PROCESSING
2 0002 0 ident = 'V04-000',
3 0003 0 addressing_mode (external = general, nonexternal = long_relative)
4 0004 0 ) =
5 0005 1 begin
6 0006 1
7 0007 1 *****
8 0008 1 *
9 0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
10 0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
11 0011 1 * ALL RIGHTS RESERVED. *
12 0012 1 *
13 0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
14 0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
15 0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
16 0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
17 0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
18 0018 1 * TRANSFERRED. *
19 0019 1 *
20 0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
21 0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
22 0022 1 * CORPORATION. *
23 0023 1 *
24 0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
25 0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
26 0026 1 *
27 0027 1 *
28 0028 1 *****
29 0029 1
30 0030 1 ++
31 0031 1 FACILITY: LINKER
32 0032 1
33 0033 1 ABSTRACT: ROUTINES TO DO ALL PASS 1 OBJECT LIBRARY PROCESSING
34 0034 1
35 0035 1
36 0036 1 ENVIRONMENT: VMS NATIVE MODE
37 0037 1
38 0038 1 AUTHOR: T.J. PORTER, CREATION DATE: 16-MAY-77
39 0039 1
40 0040 1 MODIFIED BY:
41 0041 1
42 0042 1 V03-011 JWT0168 Jim Teague 21-Mar-1984
43 0043 1 LBR$SEARCH will now return a status other than true,
44 0044 1 so when the Linker returns a 0 from LNK$ADDIMAGE to
45 0045 1 stop the library search, it must be prepared to see
46 0046 1 that 0 propagated all the way back through the
47 0047 1 LBR$SEARCH call.
48 0048 1
49 0049 1 V03-010 JWT0099 Jim Teague 14-Mar-1983
50 0050 1 New CLI interface.
51 0051 1
52 0052 1 V03-009 JWT0063 Jim Teague 26-Oct-1982
53 0053 1 Correct bug in shareable image name manipulation.
54 0054 1
55 0055 1 V03-008 JWT0044 Jim Teague 30-Jul-1982
56 0056 1 Open file performance boost. Also correct weak
57 0057 1 shr-img-symbol bug.

```

```

58 0058 1 |
59 0059 1 |      V03-007 BLS0170      Benn Schreiber      13-Apr-1982
60 0060 1 |      Beef up error handling from lbr$ calls
61 0061 1 |
62 0062 1 |      V03-006 BLS0159      Benn Schreiber      17-Mar-1982
63 0063 1 |      Also check for angles in directory spec
64 0064 1 |
65 0065 1 |      --
66 0066 1 |
67 0067 1 |      INCLUDE FILES:
68 0068 1 |
69 0069 1 |
70 0070 1 |      library 'STARLETL32':      ! STARLET DATA STRUCTURES
71 0071 1 |
72 0072 1 |      require 'PREFIX';      ! GENERAL DEFINITIONS
73 0187 1 |
74 0188 1 |      library 'DATBAS';      ! INTERNAL DATA BASE
75 0189 1 |
76 0190 1 |      forward routine
77 0191 1 |          lnk$bintim,      ! CONVERT TIME TO BINARY
78 0192 1 |          lnk$addimage;      ! ADD SHAREABLE IMAGE TO CLUSTER LIST
79 0193 1 |
80 0194 1 |
81 0195 1 |      EQUATED SYMBOLS:
82 0196 1 |
83 0197 1 |      global literal
84 0198 1 |          lnk$_libblocks = 10 : short;      ! NUMBER OF BLOCKS IN A WINDOW
85 0199 1 |          ! OF A LIBRARY
86 0200 1 |
87 0201 1 |      EXTERNAL REFERENCES:
88 0202 1 |
89 0203 1 |      external literal
90 0204 1 |          lbr$_keynotfd,      ! KEY NOT FOUND
91 0205 1 |          lns$_format,      ! FORMAT BAD
92 0206 1 |          lns$_libfind,      ! FIND FAILURE IN LIBRARY
93 0207 1 |          lns$_libnamlng,      ! ILLEGAL MODULE NAME LENGTH
94 0208 1 |          lns$_nosuchmod,      ! MODULE NOT IN LIBRARY ERROR
95 0209 1 |          lns$_readerr;      ! READ ERROR
96 0210 1 |
97 0211 1 |      external
98 0212 1 |          lbr$_gl_rmsstv,      ! STV RETURNED BY LIBRARIAN
99 0213 1 |          lnk$_gl_ctlmsk : block [, byte],      ! LINKER CONTROL FLAGS
100 0214 1 |          lnk$_gl_curfil : ref block [, byte],      ! POINTER TO CURRENT (LIBRARY) FILE DESCRIPTOR
101 0215 1 |          lnk$_gl_curclu : ref block [, byte],      ! POINTER TO CURRENT CLUSTER DESCRIPTOR
102 0216 1 |          lnk$_gl_clulst,      ! HEAD OF CLUSTER DESCRIPTOR LIST
103 0217 1 |          lnk$_gl_clutree,      ! TREE HEAD OF CLUSTER TREE
104 0218 1 |          lnk$_gl_lastclu : ref block [, byte],      ! POINTER TO LAST CLUSTER DESCRIPTOR
105 0219 1 |          lnk$_gl_udflst,      ! UNDEFINED SYMBOL LISTHEAD
106 0220 1 |          lnk$_gw_nudfsyms : word,      ! NUMBER OF UNDEFINED SYMBOLS
107 0221 1 |          lnk$_gl_objrecs,      ! NUMBER OF RECORDS PROCESSED
108 0222 1 |          lnk$_gb_pass : byte,      ! LINKER PASS
109 0223 1 |          lnk$_al_rab : block [rab$_c_bln, byte];      ! RAB TO USE FOR READS
110 0224 1 |
111 0225 1 |      external routine
112 0226 1 |          lib$_lookup_tree,      ! LOOKUP ITEM IN TREE
113 0227 1 |          lbr$_find,      ! POINT TO MODULE
114 0228 1 |          lbr$_set_module,      ! READ MODULE HEADER

```

```

: 115 0229 1 lbr$get_record, : READ RECORD OF MODULE
: 116 0230 1 lbr$lookup_key, : LOOKUP KEY IN LIBRARY
: 117 0231 1 lbr$set_index, : SET INDEX NUMBER
: 118 0232 1 lbr$search, : SEARCH INDEX FOR ENTRIES
: 119 0233 1 lnk$alloblk, : DYNAMIC MEMORY ALLOCATOR
: 120 0234 1 lnk$dealblk, : AND DEALLOCATOR
: 121 0235 1 lnk$allocluster, : ALLOCATE CLUSTER DESCRIPTOR
: 122 0236 1 lnk$insert_clu, : INSERT CLUSTER INTO CLUSTER TREE
: 123 0237 1 lnk$allofdb, : ALLOCATE FILE DESCRIPTOR BLOCK
: 124 0238 1 lnk$procsobj, : PROCESS OBJ FILES
: 125 0239 1 lnk$pointobj; : POINT TO OBJ IN A LIBRARY
: 126 0240 1 :
: 127 0241 1 : MODULE OWN STORAGE:
: 128 0242 1 :
: 129 0243 1 own
: 130 0244 1 shrdefext : quadvector [1] initial (stringdesc ('SYS$LIBRARY:.EXE')),
: 131 0245 1 : DEFAULT NAME STRING FOR SHR IMAGES
: 132 0246 1 savedrecount, : RECORD COUNT A BEGINNING OF LIBRARY MODULE
: 133 0247 1 modnamindex : initial (1), : MODULE NAME INDEX IS INDEX 1
: 134 0248 1 gstnamindex : initial (2), : GLOBAL SYMBOL INDEX IS INDEX 2
: 135 0249 1 gstmisscnt; : NUMBER OF UNSUCCESSFUL GST SEARCHS THIS CALL
: 136 0250 1
: 137 0251 1 global
: 138 0252 1 lnk$gl_futlsrch, : ACCUMULATED FUTILE SEARCHES
: 139 0253 1 lnk$gl_librecs, : NUMBER OF RECORDS PROCESSED IN LIBRARIES
: 140 0254 1 lnk$gl_libsym : ref block [, byte], : POINTER TO THE SYMBOL THAT CAUSED
: 141 0255 1 lnk$gl_nmodsexp, : NUMBER OF EXPLICITLY EXTRACTED MODULES
: 142 0256 1 lnk$gl_nmodsrch; : " EXTRACTED BECAUSE THEY RESOLVE SYMBOLS
: 143 0257 1 : A MODULE TO LOAD FROM LIBRARY.
: 144 0258 1 literal
: 145 0259 1 lnk$k_stopsearch = 0; : Flag to stop library search
: 146 0260 1

```

```
148 0261 1 global routine lnk$procslib (arglist) = ! PROCESS LIBRARY
149 0262 1
150 0263 1 ++
151 0264 1 FUNCTIONAL DESCRIPTION:
152 0265 1 THIS ROUTINE IS CALLED DURING PASS 1 OF
153 0266 1 LINKING TO PROCESS A RELOCATABLE OBJECT MODULE LIBRARY
154 0267 1 WHICH HAS ALREADY BEEN OPENED. THERE ARE TWO FUNCTIONS
155 0268 1 PERFORMED, (IN ORDER IF BOTH SPECIFIED):
156 0269 1 (1) IF EXPLICIT MODULE INCLUSION HAS BEEN SPECIFIED,
157 0270 1 THE NAMED MODULES ARE SEARCHED FOR IN THE
158 0271 1 LIBRARY'S MODULE NAME TABLE AND, IF FOUND,
159 0272 1 PROCESSED SEQUENTIALLY BY CALLING LNK$PROCSOBJ FOR EACH.
160 0273 1 (2) IF SEARCH FOR UNRESOLVED SYMBOLS IS SPECIFIED, AND THERE
161 0274 1 EXIST CURRENTLY UNDEFINED SYMBOLS ON THE UNDEFINED LIST,
162 0275 1 SEARCH THE LIBRARY GLOBAL SYMBOL TABLE FOR EACH SYMBOL.
163 0276 1 WHEN ONE IS FOUND, PROCESS THE DEFINING MODULE BY
164 0277 1 CALLING LNK$PROCSOBJ.
165 0278 1
166 0279 1 FORMAL PARAMETERS:
167 0280 1 ARGLIST IS THE ADDRESS OF THE ORIGINAL ARGUMENT LIST FROM
168 0281 1 THE IMAGE ACTIVATOR. AT OFFSET CLISA UTILSERV IS THE
169 0282 1 ADDRESS AT WHICH TO RE-CALL CLI TO PROVIDE THE MODULE
170 0283 1 NAMES ON AN EXPLICIT MODULE EXTRACTION FROM LIBRARY
171 0284 1
172 0285 1 IMPLICIT INPUTS:
173 0286 1
174 0287 1 LNK$GL_CURFIL - POINTS TO CURRENT OBJ FILE (IN THIS
175 0288 1 CASE A LIBRARY) DESCRIPTOR BLOCK.
176 0289 1 FLAG BITS IN THE DESCRIPTOR SPECIFY THE KIND OF
177 0290 1 LIBRARY SEARCH (MODULE OR SYMBOL OR BOTH). IF
178 0291 1 MODULE SEARCH IS SPECIFIED, THE FILE DESCRIPTOR CONTAINS
179 0292 1 THE POINTERS TO THE CLI DATA WHICH DESCRIBES MODULES TO
180 0293 1 BE INCLUDED.
181 0294 1 LNK$GW_NUDFSYMS - NUMBER OF UNDEFINED (STRONGLY REFERENCED)
182 0295 1 SYMBOLS
183 0296 1 LNK$GL_UDFLST - LISTHEAD FOR DOUBLY LINKED LIST OF
184 0297 1 UNDEFINED SYMBOLS.
185 0298 1
186 0299 1 IMPLICIT OUTPUTS:
187 0300 1
188 0301 1 THE MODULES SELECTED FOR PROCESSING ARE PROCESSED BY LNK$PROCSOBJ
189 0302 1 IN ADDITION:
190 0303 1 LNK$GL_LIBSYM RECEIVES THE ADDRESS OF THE
191 0304 1 ENTRY IN THE SYMBOL TABLE WHEN
192 0305 1 A SYMBOL SEARCH IS SUCCESSFUL.
193 0306 1
194 0307 1 ROUTINE VALUE:
195 0308 1
196 0309 1 COMPLETION CODES:
197 0310 1
198 0311 1 NONE
199 0312 1
200 0313 1 SIDE EFFECTS:
201 0314 1
202 0315 1 AS PERFORMED BY LNK$PROCSOBJ
203 0316 1
204 0317 1 --
```

```
205 0318 2 begin
206 0319 2 local
207 0320 2 moduleptr,
208 0321 2 nextptr,
209 0322 2 status,
210 0323 2 keydesc : block [dsc$_s_bln, byte], ! STRING DESCRIPTOR
211 0324 2 nextsym : ref block [, byte], ! NEXT UNDEFINED SYMBOL IN LIST
212 0325 2 modulerfa : block [6, byte]; ! FILE ADDRESS OF FIRST RECORD OF
213 0326 2 ! THE ASSOCIATED MODULE. FIRST 4
214 0327 2 ! BYTES ARE VBN, FOLLOWED BY THE
215 0328 2 ! OFFSET INTO BLOCK
216 0329 2
217 0330 2 map
218 0331 2 arglist : ref block [, byte];
219 0332 2 bind
220 0333 2 auxfnb = lnk$gl_curfil [fdb$_auxfnb] : block [nam$_bln, byte];
221 0334 2 ! AUXILLIARY FILE NAME BLOCK IN FDB
222 0335 2 if not .lnk$gl_curfil [fdb$_libextr] ! IF NOT EXTRACTING SPECIFIC MODULES AND
223 0336 2 then
224 0337 2 if .lnk$gw_nudfsyms eql 0 then return true; ! THERE ARE NO UNDEFINED SYMBOLS
225 0338 2
226 0339 2 if .lnk$gl_curfil [fdb$_libextr] ! IF THIS IS INCLUSION OF EXPLICITLY
227 0340 2 then
228 0341 2 begin
229 0342 2 lnk$gl_libsym = 0; ! INVALIDATE LIBRARY SYMBOL
230 0343 2 status = lbr$_set_index (%ref (.lnk$gl_curfil [fdb$_ifi]), modnamindex);
231 0344 2 ! SET TO LOOK AT MODULE NAME INDEX
232 0345 2 moduleptr = .lnk$gl_curfil [fdb$_omdlst];
233 0346 2 lnk$gl_curfil [fdb$_omdlst] = 0;
234 0347 2
235 0348 2 if not .status
236 0349 2 then
237 0350 2 begin
238 0351 2 signal (lin$_readerr, 1, lnk$gl_curfil [fdb$_q_filename], .status);
239 0352 2 return true; ! DON'T ABORT THE LINK, THO
240 0353 2 end;
241 0354 2 !
242 0355 2 NOW LOOP, GRABBING THE NEXT MODULE NAME IN THE LINKED
243 0356 2 LIST, SEARCHING MODULE NAME TABLE FOR THAT MODULE THEN,
244 0357 2 IF FOUND PROCESSING THE MODULE
245 0358 2
246 0359 2 while .moduleptr neq 0 ! THAT IS WHILE THERE
247 0360 2 do
248 0361 2 begin ! REMAINS MORE TEXT ON THE
249 0362 2 nextptr = (.moduleptr);
250 0363 2 keydesc [dsc$_length] = (.moduleptr+4)<0,8>;
251 0364 2 keydesc [dsc$_pointer] = .moduleptr + 5;
252 0365 2 if .keydesc [dsc$_length] eql 0 ! GO GET NEXT NAME (ALLOWING
253 0366 2 or .keydesc [dsc$_length] gtru sym$_maxlng ! CLI TO USE THE LIBRARY HEADER BUFFER)
254 0367 2 then
255 0368 2 begin
256 0369 2 signal (lin$_libnamlng, 2, ! CHECK A VALID NAME
257 0370 2 keydesc [dsc$_length], .keydesc [dsc$_length]); ! AND ISSUE ERROR IF AN
258 0371 2 keydesc [dsc$_length] = sym$_maxlng; ! ILLEGAL LENGTH, SET TO MAXIMUM
259 0372 2 end;
260 0373 2 if not (status = lbr$_lookup_key (%ref (.lnk$gl_curfil [fdb$_ifi]), keydesc, modulerfa))
261 0374 2 ! LOOKUP THE MODULE NAME
```

```
262 0375 4
263 0376 5
264 0377 5
265 0378 5
266 0379 5
267 0380 5
268 0381 5
269 0382 5
270 0383 5
271 0384 4
272 0385 5
273 0386 5
274 0387 5
275 0388 5
276 0389 5
277 0390 6
278 0391 6
279 0392 6
280 0393 6
281 0394 6
282 0395 6
283 0396 6
284 0397 6
285 0398 6
286 0399 5
287 0400 4
288 0401 4
289 0402 4
290 0403 3
291 0404 2
292 0405 2
293 0406 2
294 0407 2
295 0408 2
296 0409 2
297 0410 2
298 0411 3
299 0412 3
300 0413 3
301 0414 3
302 0415 3
303 0416 3
304 0417 3
305 0418 3
306 0419 4
307 0420 4
308 0421 4
309 0422 3
310 0423 3
311 0424 3
312 0425 3
313 0426 4
314 0427 4
315 0428 5
316 0429 5
317 0430 5
318 0431 5

then
begin
if .status eql lbr$_keynotfnd
then
signal (lin$_nosuchmod, 2, keydesc [dsc$_length],
lnk$_gl_curfil [fdb$_q_filename])
else
signal (lin$_readerr, 1, lnk$_gl_curfil [fdb$_q_filename], .status, .lbr$_gl_rmsstv);
end
else
begin
if .lnk$_gl_curfil [fdb$_v_imglib] ! IF THIS IS SHR IMG STB LIBRARY
then
lnk$_addimage (keydesc, modulerfa) ! THEN JUST ADD TO THE CLUSTER LIST
else
begin
savedrecount = .lnk$_gl_objrecs; ! SAVE CURRENT RECORD COUNT
lnk$_gl_nmodsexp = .lnk$_gl_nmodsexp + 1; ! COUNT ONE MORE EXPLICITLY EXTRACTED
lnk$_pointobj (modulerfa); ! FOUND IT SO GO POINT TO

if not lnk$_procsobj (modulerfa) then return false; ! THE MODULE IN THE LIBRARY
lnk$_gl_librecs = .lnk$_gl_librecs + .lnk$_gl_objrecs -
! ACCUMULATE THE NUMBER OF RECORDS
.savedrecount; ! FOUND IN LIBRARIES
end;
end;
lnk$_dealblk(.keydesc[dsc$_length]+5, .moduleptr);
moduleptr = .nextptr;
end;
! AND PROCESS IT

NOW CHECK WHETHER THIS LIBRARY IS TO BE SEARCHED FOR
CURRENTLY UNDEFINED SYMBOLS. EXIT NOW IF NOT

if .lnk$_gl_curfil [fdb$_v_libsrch] ! IF A SYMBOL SEARCH REQUIRED
then
begin
lnk$_gl_curfil [fdb$_v_newudf] = false; ! RESET UNDEFINED SYMBOLS CONTRIBUTED
gstmisscnt = 0; ! RESET COUNT OF SYMBOLS NOT FOUND
nextsym = .lnk$_gl_udflst; ! START AT TOP OF LIST, AND
status = lbr$_set_index (%ref (.lnk$_gl_curfil [fdb$_w_ifi]), gstnamindex); ! LOOK IN GLOBAL SYMBOL INDEX

if not .status
then
begin
signal (lin$_readerr, 1, lnk$_gl_curfil [fdb$_q_filename], .status);
return true; ! DON'T ABORT THE LINK, THO
end;

if .lnk$_gl_curfil [fdb$_v_imglib] ! IF THIS IS SHR IMG STB LIBRARY
then
begin
while .nextsym neq lnk$_gl_udflst do
begin
bind
nextsymnam = .nextsym - .nextsym [sym$_b_namlng] - snb$_c_fxdlen : block [, byte];
if not .nextsym [sym$_v_weak]
```



```

319 0432 5
320 0433 6
321 0434 6
322 0435 6
323 0436 6
324 0437 7
325 0438 7
326 0439 6
327 0440 7
328 0441 7
329 0442 7
330 0443 7
331 0444 8
332 0445 7
333 0446 7
334 0447 7
335 0448 6
336 0449 6
337 0450 6
338 0451 6
339 0452 6
340 0453 6
341 0454 5
342 0455 5
343 0456 5
344 0457 5
345 0458 4
346 0459 3
347 0460 3
348 0461 3
349 0462 3
350 0463 4
351 0464 4
352 0465 4
353 0466 4
354 0467 4
355 0468 5
356 0469 5
357 0470 5
358 0471 5
359 0472 5
360 0473 4
361 0474 3
362 0475 4
363 0476 4
364 0477 4
365 0478 4
366 0479 4
367 0480 4
368 0481 4
369 0482 4
370 0483 4
371 0484 4
372 0485 4
373 0486 4
374 0487 4
375 0488 4

then
begin
keydesc [dsc$w_length] = .nextsym [sym$b_namlng];
keydesc [dsc$a_pointer] = nextsymnam [snb$t_name];

if (status = lbr$lookup_key (%ref (.lnk$gl_curfil [fdb$w_ifi]), keydesc, modulerfa))
! IF SYMBOL IS IN LIBRARY
then
begin
status = lbr$search (%ref (.lnk$gl_curfil [fdb$w_ifi]), modnamindex,
! FIND THE MODULE NAME
modulerfa, lnk$addimage);
if (not .status) and (.status neq lnk$k_stopsearch)
then
signal (lin$_readerr, 1, lnk$gl_curfil [fdb$q_filename], .status);
end
else
if .status neq lbr$_keynotfnd
then
signal (lin$_readerr, 1,
lnk$gl_curfil [fdb$q_filename], .status, .lbr$gl_rmsstv);
end;

nextsym = .nextsym [sym$l_udflink]; ! LINK TO NEXT UNDEFINED SYMBOL
end
else
while .lnk$gw_nudfsyms neq 0 ! WHILE IT CONTAINS SOME UN-
! DEFINED SYMBOLS, GET
and (if (lnk$gl_libsym = .nextsym) neq lnk$gl_udf[st ! NEXT ENTRY. HOWEVER
then true ! IF BACK AT THE LISTHEAD
else if not .lnk$gl_curfil [fdb$v_newudf] ! AND THIS FILE DID NOT ADD
then false ! MORE UNDEFINED SYMBOLS-WE ARE DONE
else
begin ! IF IT DID ADD MORE, GET
lnk$gl_libsym = .lnk$gl_libsym [sym$l_udflink]; ! TOP ENTRY IN LIST
lnk$gl_curfil [fdb$v_newudf] = false; ! RESET THE UNDEFINED SYMBOLS CONTRIBUTED FL
true ! AND CONTINUE THE
end
) ! SEARCH
do
begin ! FOR A SYMBOL ON THE
bind
libsymnam = .lnk$gl_libsym - .lnk$gl_libsym [sym$b_namlng] - snb$c_fxdlen : block [, byt
; ! POINT TO NAME PART
keydesc [dsc$w_length] = .lnk$gl_libsym [sym$b_namlng]; ! MAKE STRING DESCRIPTOR FOR NAME
keydesc [dsc$a_pointer] = libsymnam [snb$t_name];
nextsym = .lnk$gl_libsym [sym$l_udflink]; ! UNDEFINED LIST AND
if (.lnk$gl_libsym [sym$w_flags] and gsy$m_weak) eql 0 ! PROVIDED IT IS NOT A WEAK
and not .lnk$gl_libsym [sym$v_gstmiss] ! REFERENCE AND THAT WE
then
! HAVE NOT BEFORE FAILED TO
! FIND IT IN THIS LIBRARY.

```

```

376 0489 4
377 0490 5
378 0491 5
379 0492 4
380 0493 5
381 0494 5
382 0495 5
383 0496 5
384 0497 5
385 0498 5
386 0499 5
387 0500 5
388 0501 5
389 0502 5
390 0503 5
391 0504 5
392 0505 4
393 0506 5
394 0507 5
395 0508 5
396 0509 5
397 0510 5
398 0511 5
399 0512 5
400 0513 5
401 0514 5
402 0515 5
403 0516 4
404 0517 3
405 0518 3
406 0519 3
407 0520 3
408 0521 3
409 0522 3
410 0523 3
411 0524 3
412 0525 3
413 0526 3
414 0527 3
415 0528 3
416 0529 3
417 0530 3
418 0531 3
419 0532 2
420 0533 2
421 0534 2
422 0535 2
423 0536 1

if (status = lbr$lookup_key (%ref (.lnk$gl_curfil [fdb$w_ifi]), keydesc, modulerfa))
    ! GO LOOK FOR THE SYMBOL
then
begin
    ! RETURN RECORD'S FILE ADDRESS
    lnk$gl_nmodsrch = .lnk$gl_nmodsrch + 1; ! COUNT THE NUMBER OF MODULES
    savedrccount = .lnk$gl_objrecs; ! SAVE CURRENT RECORD COUNT
    lnk$pointobj (modulerfa); ! TO POINT TO THE MODULE

    if not lnk$procsobj (modulerfa) then return false; ! AND GO PROCESS IT
    lnk$gl_librecs = .lnk$gl_librecs + .lnk$gl_objrecs - ! ACCUMULATE THE NUMBER OF
    .savedrccount; ! RECORDS FROM LIBRARIES
    nextsym = .lnk$gl_libsym; ! RETRIEVE NEXT IN LIST (SINCE THE
    ! ONE WE HAD MAY HAVE JUST BEEN
    ! DEFINED BY THAT MODULE)
end
else
begin
    ! IF THE SYMBOL WAS NOT

    if .status neq lbr$_keynotfnd
    then
        signal (lin$_readerr, 1,
            lnk$gl_curfil [fdb$q_filename], .status, .lbr$gl_rmsstv);

        gstmisscnt = .gstmisscnt + 1; ! FOUND IN LIBRARY, COUNT
        lnk$gl_libsym [sym$v_gstmiss] = true; ! ANOTHER MISS AND SUPPRESS
        ! ANY MORE SEARCHES FOR IT
    end;
    ! END OF SYMBOL LIST LOOP

    end;
    lnk$gl_libsym = 0; ! INVALIDATE THE SYMBOL POINTER

    NOW FINISHED LOOKING FOR UNDEFINED SYMBOLS IN THE CURRENT LIBRARY
    MUST NOW GO DOWN WHAT IS LEFT OF THE UNDEFINED SYMBOL LIST, TURNING
    OFF THE GST MISS FLAG IN EACH SYMBOL DESCRIPTOR.

    nextsym = lnk$gl_udflst;
    if .gstmisscnt neq 0
    then
        ! IF THERE WERE NO MISSES
        while (nextsym = .nextsym [sym$_udflink]) neq lnk$gl_udflst
        ! FORGET IT
        do
            nextsym [sym$v_gstmiss] = false; ! TURN OFF FLAG

        lnk$gl_futlsrch = .lnk$gl_futlsrch + .gstmisscnt; ! ACCUMULATE FUTILE SEARCH COUNT
    end;

    lnk$gl_curfil [fdb$v_selser] = false; ! RESET THE POSSIBLE SELECTIVE SEARCH FLAG
    return true; ! AND ALL DONE
end;
! END OF ROUTINE

```

```

.TITLE LNK_PROCSLIB
.IDENT \V04-000\
.PSECT $PLITS,NOWRT,NOEXE,2

```

```

58 45 2E 3A 59 52 41 52 42 49 4C 24 53 59 53 0000 P.AAA: .ASCII \SYSS$LIBRARY:.EXE\
45 000F

```

.PSECT \$OWNS,NOEXE,2

```

00000010 0000 SHRDEFEXT:
                .LONG 16
00000000' 00004 .ADDRESS P.AAA
00008 SAVEDRECOUNT:
                .BLKB 4
00000001 0000C MODNAMINDEX:
                .LONG 1
00000002 00010 GSTNAMINDEX:
                .LONG 2
00014 GSTMISSCNT:
                .BLKB 4

```

.PSECT \$GLOBALS,NOEXE,2

```

0000 LNK$GL_FUTLSRCH::
                .BLKB 4
00004 LNK$GL_LIBRECS::
                .BLKB 4
00008 LNK$GL_LIBSYM::
                .BLKB 4
0000C LNK$GL_NMODSEXP::
                .BLKB 4
00010 LNK$GL_NMODSRCH::
                .BLKB 4

```

LNK\$K\_LIBBLOCKS== 10

```

.EXTRN LBR$ KEYNOTFND, LINS FORMAT
.EXTRN LINS_LIBFIND, LINS LIBNAMLNG
.EXTRN LINS_NOSUCHMOD, LINS_READERR
.EXTRN LBR$GL_RMSSTV, LNK$GL_CTLMSK
.EXTRN LNK$GL_CURFIL, LNK$GL_CURCLU
.EXTRN LNK$GL_CLULST, LNK$GL_CLUTREE
.EXTRN LNK$GL_LASTCLU, LNK$GL_UDFLST
.EXTRN LNK$GW_NUDFSYMS
.EXTRN LNK$GL_OBJRECS, LNK$GB_PASS
.EXTRN LNK$AL_RAB, LIB$LOOKUP_TREE
.EXTRN LBR$FIND, LBR$SET_MODUCE
.EXTRN LBR$GET_RECORD, LBR$LOOKUP_KEY
.EXTRN LBR$SET_INDEX, LBR$SEARCH
.EXTRN LNK$ALLOBLK, LNK$DEALBLK
.EXTRN LNK$ALLOCLUSTER
.EXTRN LNK$INSERT_CLU, LNK$ALLOFDB
.EXTRN LNK$PROCSOBJ, LNK$POINTOBJ

```

.PSECT \$CODE\$,NOWRT,2

```

                                OFFC 00000
00000000' 00 9E 00002 .ENTRY LNK$PROCSLIB, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 0261
5B 00000000G 00 9E 00002 MOVAB LNK$GL_OBJRECS, R11
5A 00000000G 00 9E 00009 MOVAB LNK$GL_UDFLST, R10
59 00000000G 8F D0 00010 MOVL #LINS_READERR, R9
58 00000000G 00 9E 00017 MOVAB LIB$SIGNAL, R8
57 00000000' EF 9E 0001E MOVAB SAVEDRECOUNT, R7
56 00000000' EF 9E 00025 MOVAB LNK$GL_LIBSYM, R6

```

13	0A	55 0000000G	00 9E 0002C	MOVAB	LNK\$GL_CURFIL, R5	
		5E	14 C2 00033	SUBL2	#20, SP	
		50	65 D0 00036	MOVL	LNK\$GL_CURFIL, R0	0332
		A0	06 E0 00039	BBS	#6, 10(R0), 3\$	0334
		00000000G	00 B5 0003E	TSTW	LNK\$GW_NUDFSYMS	0336
			03 12 00044	BNEQ	1\$	
03	0A	A0	02C1 31 00046	BRW	31\$	
			06 E0 00049	BBS	#6, 10(R0), 3\$	0338
			0100 31 0004E	BRW	12\$	
			66 D4 00051	CLRL	LNK\$GL_LIBSYM	0341
			04 A7 9F 00053	PUSHAB	MODNAMINDEX	0342
	04	AE	24 A0 3C 00056	MOVZWL	36(R0), 4(SP)	
			04 AE 9F 0005B	PUSHAB	4(SP)	
	00000000G	00	02 FB 0005E	CALLS	#2, LBR\$SET_INDEX	
		53	50 D0 00065	MOVL	R0, STATUS	
		50	65 D0 00068	MOVL	LNK\$GL_CURFIL, R0	0344
		52	04 A0 D0 0006B	MOVL	4(R0), -MODULEPTR	
			04 A0 D4 0006F	CLRL	4(R0)	0345
		08	53 E8 00072	BLBS	STATUS, 4\$	0347
			53 DD 00075	PUSHL	STATUS	0350
			14 A0 9F 00077	PUSHAB	20(R0)	
			0107 31 0007A	BRW	14\$	
			52 D5 0007D	TSTL	MODULEPTR	0358
			CD 13 0007F	BEQL	2\$	
		54	62 D0 00081	MOVL	(MODULEPTR), NEXTPTR	0361
	0C	AE	04 A2 9B 00084	MOVZBW	4(MODULEPTR), KEYDESC	0362
	10	AE	05 A2 9E 00089	MOVAB	5(R2), KEYDESC+4	0363
		50	0C AE 3C 0008E	MOVZWL	KEYDESC, R0	0364
			05 13 00092	BEQL	5\$	
		1F	50 B1 00094	CMPW	R0, #31	0365
			14 1B 00097	BLEQU	6\$	
			50 DD 00099	PUSHL	R0	0370
			10 AE 9F 0009B	PUSHAB	KEYDESC	
			02 DD 0009E	PUSHL	#2	
		00000000G	8F DD 000A0	PUSHL	#LINS LIBNAMLNG	
			04 FB 000A6	CALLS	#4, LIB\$SIGNAL	
	0C	AE	1F B0 000A9	MOVW	#31, KEYDESC	0371
			04 AE 9F 000AD	PUSHAB	MODULERFA	0373
			10 AE 9F 000B0	PUSHAB	KEYDESC	
		50	65 D0 000B3	MOVL	LNK\$GL_CURFIL, R0	
	08	AE	24 A0 3C 000B6	MOVZWL	36(R0), 8(SP)	
			08 AE 9F 000BB	PUSHAB	8(SP)	
	00000000G	00	03 FB 000BE	CALLS	#3, LBR\$LOOKUP_KEY	
		53	50 D0 000C5	MOVL	R0, STATUS	
		30	53 E8 000C8	BLBS	STATUS, 8\$	
50		65	14 C1 000CB	ADDL3	#20, LNK\$GL_CURFIL, R0	0380
	00000000G	8F	53 D1 000CF	CMPL	STATUS, #LBR\$_KEYNOTFND	0377
			12 12 000D6	BNEQ	7\$	
			50 DD 000D8	PUSHL	R0	0380
			10 AE 9F 000DA	PUSHAB	KEYDESC	0379
			02 DD 000DD	PUSHL	#2	0380
		00000000G	8F DD 000DF	PUSHL	#LINS NOSUCHMOD	
		68	04 FB 000E5	CALLS	#4, LIB\$SIGNAL	
			51 11 000E8	BRB	11\$	
		00000000G	00 DD 000EA	PUSHL	LBR\$GL_RMSSTV	0382
			09 BB 000F0	PUSHR	#*M<R0,R3>	
			01 DD 000F2	PUSHL	#1	

			59	DD	000F4		PUSHL	R9			
	68		05	FB	000F6		CALLS	#5, LIBSSIGNAL			0373
			40	11	000F9		BRB	11\$			0386
	50		65	DO	000FB	8\$:	MOVL	LNK\$GL_CURFIL, R0			
	OF		0B	A0	E9	000FE	BLBC	11(R0), 9\$			
			04	AE	9F	00102	PUSHAB	MODULERFA			0388
			10	AE	9F	00105	PUSHAB	KEYDESC			
	00000000V	EF	02	FB	00108		CALLS	#2, LNK\$ADDIMAGE			
			2A	11	0010F		BRB	11\$			
		67	04	6B	DO	00111	9\$:	MOVL	LNK\$GL_OBJRECS, SAVEDRECOUNT		0391
			04	A6	D6	00114	INCL	LNK\$GL_NMODSEXP			0392
	00000000G	00	04	AE	9F	00117	PUSHAB	MODULERFA			0393
			04	01	FB	0011A	CALLS	#1, LNK\$POINTOBJ			
	00000000G	00	04	AE	9F	00121	PUSHAB	MODULERFA			0395
		03	01	FB	00124		CALLS	#1, LNK\$PROCSOBJ			
			50	E8	0012B		BLBS	R0, 10\$			
			01DD	31	0012E		BRW	32\$			
FC	50	FC	68	C1	00131	10\$:	ADDL3	LNK\$GL_OBJRECS, LNK\$GL_LIBRECS, R0			0396
	A6		50	C3	00136		SUBL3	SAVEDRECOUNT, R0, LNK\$GL_LIBRECS			0398
			52	DD	0013B	11\$:	PUSHL	MODULEPTR			0401
			7E	AE	3C	0013D	MOVZWL	KEYDESC, -(SP)			
			6E	05	C0	00141	ADDL2	#5, (SP)			
	00000000G	00	02	FB	00144		CALLS	#2, LNK\$DEALBLK			
		52	54	DO	0014B		MOVL	NEXTPTR, MODULEPTR			0402
			FF2C	31	0014E		BRW	4\$			0358
		50	65	DO	00151	12\$:	MOVL	LNK\$GL_CURFIL, R0			0409
			0A	A0	95	00154	TSTB	10(R0)			
			03	19	00157		BLSS	13\$			
			01A7	31	00159		BRW	30\$			
	0A	A0	01	8A	0015C	13\$:	BICB2	#1, 10(R0)			0412
			0C	A7	D4	00160	CLRL	GSTMISSCNT			0413
		52	6A	DO	00163		MOVL	LNK\$GL_UDFLST, NEXTSYM			0414
			08	A7	9F	00166	PUSHAB	GSTNAMINDEX			0415
	04	AE	24	A0	3C	00169	MOVZWL	36(R0), 4(SP)			
			04	AE	9F	0016E	PUSHAB	4(SP)			
	00000000G	00	02	FB	00171		CALLS	#2, LBR\$SET_INDEX			
		53	50	DO	00178		MOVL	R0, STATUS			
		10	53	E8	0017B		BLBS	STATUS, 15\$			0417
			53	DD	0017E		PUSHL	STATUS			0420
7E		65	14	C1	00180		ADDL3	#20, LNK\$GL_CURFIL, -(SP)			
			01	DD	00184	14\$:	PUSHL	#1			
			59	DD	00186		PUSHL	R9			
		68	04	FB	00188		CALLS	#4, LIBSSIGNAL			
			017C	31	0018B		BRW	31\$			0421
		50	65	DO	0018E	15\$:	MOVL	LNK\$GL_CURFIL, R0			0424
		03	0B	A0	E8	00191	BLBS	11(R0), 16\$			
			0095	31	00195		BRW	20\$			
		50	6A	9E	00198	16\$:	MOVAB	LNK\$GL_UDFLST, R0			0427
		50	52	D1	0019B		CPL	NEXTSYM, R0			
			03	12	0019E		BNEQ	17\$			
			0140	31	001A0		BRW	27\$			
		50	0F	A2	9A	001A3	17\$:	MOVZBL	15(NEXTSYM), R0		0430
		52	50	C3	001A7		SUBL3	R0, NEXTSYM, R0			
		78	0A	A2	E8	001AB	BLBS	10(NEXTSYM), 19\$			0431
			0F	A2	9B	001AF	MOVZBW	15(NEXTSYM), KEYDESC			0434
		OC	AE	60	9E	001B4	MOVAB	(R0), KEYDESC+4			0435
		10	AE	9F	001B8		PUSHAB	MODULERFA			0437

		10	AE	9F	001BB	PUSHAB	KEYDESC		
	50		65	DO	001BE	MOVL	LNK\$GL_CURFIL, R0		
08	AE	24	A0	3C	001C1	MOVZWL	36(R0), 8(SP)		
		08	AE	9F	001C6	PUSHAB	8(SP)		
00000000G	00		03	FB	001C9	CALLS	#3, LBR\$LOOKUP_KEY		
	53		50	DO	001D0	MOVL	R0, STATUS		
	35		53	E9	001D3	BLBC	STATUS, 18\$		
		00000000V	EF	9F	001D6	PUSHAB	LNK\$ADDIMAGE		0441
		08	AE	9F	001DC	PUSHAB	MODULERFA		
		04	A7	9F	001DF	PUSHAB	MODNAMINDEX		
	50		65	DO	001E2	MOVL	LNK\$GL_CURFIL, R0		
0C	AE	24	A0	3C	001E5	MOVZWL	36(R0), 12(SP)		
		0C	AE	9F	001EA	PUSHAB	12(SP)		
00000000G	00		04	FB	001ED	CALLS	#4, LBR\$SEARCH		
	53		50	DO	001F4	MOVL	R0, STATUS		
	2D		53	E8	001F7	BLBS	STATUS, 19\$		0444
			2B	13	001FA	BEQL	19\$		
7E	65		53	DD	001FC	PUSHL	STATUS		0446
			14	C1	001FE	ADDL3	#20, LNK\$GL_CURFIL, -(SP)		
			01	DD	00202	PUSHL	#1		
			59	DD	00204	PUSHL	R9		
	68		04	FB	00206	CALLS	#4, LIB\$SIGNAL		
			1C	11	00209	BRB	19\$		0437
00000000G	8F		53	D1	0020B	18\$:	STATUS, #LBR\$_KEYNOTFND		0450
			13	13	00212	BEQL	19\$		
		00000000G	00	DD	00214	PUSHL	LBR\$GL_RMSSTV		0453
			53	DD	0021A	PUSHL	STATUS		
7E	65		14	C1	0021C	ADDL3	#20, LNK\$GL_CURFIL, -(SP)		
			01	DD	00220	PUSHL	#1		
			59	DD	00222	PUSHL	R9		
	68		05	FB	00224	CALLS	#5, LIB\$SIGNAL		
	52		62	DO	00227	19\$:	(NEXTSYM), NEXTSYM		0456
			FF6B	31	0022A	BRW	16\$		0426
		00000000G	00	B5	0022D	20\$:	LNK\$GW_NUDFSYMS		0461
			03	12	00233	BNEQ	22\$		
			00AB	31	00235	21\$:	BRW		0463
	66		52	DO	00238	22\$:	MOVL	NEXTSYM, LNK\$GL_LIBSYM	
	50		6A	9E	0023B	MOVAB	LNK\$GL_UDFLST, R0		
	50		52	D1	0023E	CMPL	NEXTSYM, R0		
			0E	12	00241	BNEQ	23\$		
	50		65	DO	00243	MOVL	LNK\$GL_CURFIL, R0		0465
	EB	0A	A0	E9	00246	BLBC	10(R0), 21\$		
	76		96	DO	0024A	MOVL	@LNK\$GL_LIBSYM, LNK\$GL_LIBSYM		0469
0A	A0		01	8A	0024D	BICB2	#1, 10(R0)		0470
	50		66	DO	00251	23\$:	MOVL	LNK\$GL_LIBSYM, R0	0478
	51	OF	A0	9A	00254	MOVZBL	15(R0), R1		
51	50		51	C3	00258	SUBL3	R1, R0, R1		
	OC	OF	A0	9B	0025C	MOVZBW	15(R0), KEYDESC		0481
	10		61	9E	00261	MOVAB	(R1), KEYDESC+4		0482
	52		60	DO	00265	MOVL	(R0), NEXTSYM		0483
	C1	0A	A0	E8	00268	BLBS	10(R0), 20\$		0485
	BD	0C	A0	E8	0026C	BLBS	12(R0), 20\$		0486
		04	AE	9F	00270	PUSHAB	MODULERFA		0490
		10	AE	9F	00273	PUSHAB	KEYDESC		
	50		65	DO	00276	MOVL	LNK\$GL_CURFIL, R0		
08	AE	24	A0	3C	00279	MOVZWL	36(R0), 8(SP)		
		08	AE	9F	0027E	PUSHAB	8(SP)		

	00000000G	00	03	FB	00281	CALLS	#3, LBR\$LOOKUP_KEY		
		53	50	DO	00288	MOVL	R0, STATUS		
		2C	53	E9	0028B	BLBC	STATUS, 24\$		
			08	A6	D6	0028E	INCL	LNK\$GL_NMODSRCH	0494
		67	6B	DO	00291	MOVL	LNK\$GL_OBJRECS, SAVEDRECOUNT		0495
			04	AE	9F	00294	PUSHAB	MODULERFA	0496
	00000000G	00	01	FB	00297	CALLS	#1, LNK\$POINTOBJ		
			04	AE	9F	0029E	PUSHAB	MODULERFA	0498
	00000000G	0C	01	FB	002A1	CALLS	#1, LNK\$PROCSOBJ		
		63	50	E9	002AB	BLBC	R0, 32\$		
FC	50	FC	6B	C1	002AB	ADDL3	LNK\$GL_OBJRECS, LNK\$GL_LIBRECS, R0		0499
	A6		50	C3	002B0	SUBL3	SAVEDRECOUNT, R0, LNK\$GL_LIBRECS		0500
			52	D0	002B5	MOVL	LNK\$GL_LIBSYM, NEXTSYM		0501
			26	11	002B8	BRB	26\$		0490
	00000000G	8F	53	D1	002BA	24\$:	CMPL	STATUS, #LBR\$_KEYNOTFND	0508
			13	13	002C1		BEQL	25\$	
			00	DD	002C3		PUSHL	LBR\$GL_RMSSTV	0511
			53	DD	002C9		PUSHL	STATUS	
	7E		14	C1	002CB		ADDL3	#20, LNK\$GL_CURFIL, -(SP)	
			01	DD	002CF		PUSHL	#1	
			59	DD	002D1		PUSHL	R9	
			68	05	FB	002D3	CALLS	#5, LIB\$SIGNAL	
			0C	A7	D6	002D6	25\$:	INCL	GSTMISSCNT
			50	D0	002D9		MOVL	LNK\$GL_LIBSYM, R0	0513
			0C	A0	01	88	002DC	BISB2	#1, 12(R0)
				FF4A	31	002E0	26\$:	BRW	20\$
			66	D4	002E3	27\$:	CLRL	LNK\$GL_LIBSYM	0518
			52	6A	9E	002E5	MOVAB	LNK\$GL_UDFLST, NEXTSYM	0524
			51	0C	A7	D0	002E8	MOVL	GSTMISSCNT, R1
					11	13	002EC	BEQL	29\$
			52	62	D0	002EE	28\$:	MOVL	(NEXTSYM), NEXTSYM
			50	6A	9E	002F1	MOVAB	LNK\$GL_UDFLST, R0	0527
			50	52	D1	002F4	CMPL	NEXTSYM, R0	
				06	13	002F7	BEQL	29\$	
			0C	A2	01	8A	002F9	BICB2	#1, 12(NEXTSYM)
					EF	11	002FD	BRB	28\$
			F8	A6	51	C0	002FF	29\$:	ADDL2
			50	D0	00303	30\$:	MOVL	LNK\$GL_CURFIL, R0	0531
			0A	A0	08	8A	00306	BICB2	#8, 10(R0)
			50	01	C0	0030A	31\$:	MOVL	#1, R0
					04	0030D		RET	0535
					50	D4	0030E	32\$:	CLRL
					04	00310		RET	0536

; Routine Size: 785 bytes, Routine Base: \$CODE\$ + 0000

; 424 0537 1

```

: 426 0538 1 global routine lnk$bintim (asctim, bintim) =
: 427 0539 2   begin
: 428 0540 2   :
: 429 0541 2       THIS ROUTINE CONVERTS A DATE/TIME STRING FROM A MODULE
: 430 0542 2       HEADER TO BINARY.
: 431 0543 2   :
: 432 0544 2   INPUTS:
: 433 0545 2   :
: 434 0546 2       ASCTIM          ADDRESS OF 17-BYTE ASCII DATE/TIME
: 435 0547 2       BINTIM          ADDRESS OF QUADWORD TO STORE BINARY TIME
: 436 0548 2   :
: 437 0549 2   --
: 438 0550 2   local
: 439 0551 2       timedesc : block [dsc$w_s bln, byte],
: 440 0552 2       timestring : vector [23, byte];
: 441 0553 2   :
: 442 0554 2       ch$move (17, .asctim, timestring);          ! COPY ASCII STRING FOR DATE/TIME
: 443 0555 2       ch$fill (%c'0', 6, timestring [17]);        ! FILL REST OF STRING WITH 0'S
: 444 0556 2       timestring [17] = %c'.';                   ! FIX PUNCTUATION AS REQUIRED
: 445 0557 2       timestring [20] = %c'.';
: 446 0558 2       timedesc [dsc$w_length] = 23;
: 447 0559 2       timedesc [dsc$a_pointer] = timestring;
: 448 0560 2       $bintim (timbuf = timedesc, timadr = .bintim);
: 449 0561 2       return true
: 450 0562 1       end;

```

				.EXTRN SYSSBINTIM			
				.ENTRY LNK\$BINTIM, Save R2,R3,R4,R5		:	0538
				SUBL2	#32, SP	:	
06	6E	04	BC	11	28	00005	0554
	30		6E	00	2C	0000A	0555
				11	AE	0000F	
		11	AE	3A	90	00011	0556
		14	AE	2E	90	00015	0557
		18	AE	17	B0	00019	0558
		1C	AE	6E	9E	0001D	0559
				08	AC	DD	00021
				1C	AE	9F	00024
	00000000G	00		02	FB	00027	0560
		50		01	D0	0002E	0561
				04	00	00031	0562

; Routine Size: 50 bytes, Routine Base: \$CODE\$ + 0311



```

: 452 0563 1 global routine lnk$addimage (moduledesc, modulerfa, retcludesc, foundflag) =
: 453 0564 2 begin
: 454 0565 2
: 455 0566 2 THIS ROUTINE IS CALLED BY THE LIBRARIAN WHEN IT FINDS A MODULE
: 456 0567 2 NAME WITH THE SAME RFA AS THE GLOBAL SYMBOL JUST LOCATED. WE
: 457 0568 2 CHECK TO SEE IF THIS SHAREABLE IMAGE HAS ALREADY BEEN REQUESTED.
: 458 0569 2 IF NOT, THEN A CLUSTER DESCRIPTOR AND FDB ARE ALLOCATED.
: 459 0570 2
: 460 0571 2 IF MODULERFA IS NOT PRESENT (NULLPARAMETER), THEN NO LIBRARY READING
: 461 0572 2 IS DONE, THE CLUSTER DESCRIPTOR AND FILE DESCRIPTOR BLOCKS ARE CREATED,
: 462 0573 2 HOWEVER.
: 463 0574 2
: 464 0575 2 IF RETCLUDESC IS PASSED, IT IS THE ADDRESS OF A LONGWORD TO STORE
: 465 0576 2 THE ALLOCATED CLUSTER DESCRIPTOR ADDRESS. NOTE THAT THE ONLY WAY
: 466 0577 2 TO DETERMINE IF AN IMAGE WAS REQUESTED IS TO CHECK FOR RETCLUDESC
: 467 0578 2 BEING NON-0, SINCE THIS ROUTINE ALWAYS RETURNS FALSE TO STOP LBR SEARCH
: 468 0579 2
: 469 0580 2 IF FOUNDFLAG IS PASSED, IT IS THE ADDRESS OF A LONGWORD TO STORE
: 470 0581 2 A 1 (FOUND) OR 0 (INSERTED)
: 471 0582 2
: 472 0583 2 routine compareclu (keydesc, clunode) =
: 473 0584 2 begin
: 474 0585 2
: 475 0586 2 LOCAL ROUTINE TO COMPARE A NAME OF NODE WITH ANOTHER NAME
: 476 0587 2
: 477 0588 2 map
: 478 0589 2 keydesc : ref block [, byte], ! POINTER TO STRING DESCRIPTOR
: 479 0590 2 clunode : ref block [, byte]; ! NODE FOR DESCRIPTOR BEING EXAMINED
: 480 0591 2 local
: 481 0592 2 clu : ref block [, byte];
: 482 0593 2 clu = .clunode [node$_ptr]; ! POINT TO CLUSTER DESCRIPTOR
: 483 0594 2 return ch$compare (.keydesc [dsc$_length], .keydesc [dsc$_pointer], .clu [clu$_namlng],
: 484 0595 2 clu [clu$_name])
: 485 0596 2 end;

```

```

                                001C 0000 COMPARECLU:
                                .WORD Save R2,R3,R4
                                MOVL CLUNODE, R0 ; 0583
                                MOVL 10(R0), CLU ; 0593
                                MOVL KEYDESC, R1 ; 0594
                                MOVZBL 92(CLU), R2
                                MOVL #1, R4 ; 0595
                                CMPC5 (R1), @4(R1), #0, R2, 93(CLU)
                                BGTRU 1$
                                SBWC #1, R4
                                MOVL R4, R0 ; 0596
                                RET

```

: Routine Size: 38 bytes, Routine Base: \$CODE\$ + 0343

: 486 0597 2 !

```

487 0598 2 : MAIN BODY OF LNK$ADDIMAGE
488 0599 2 :
489 0600 2 :
490 0601 2 :   map
491 0602 2 :     moduledesc : ref block [, byte],
492 0603 2 :     modulerfa : ref block [, byte],
493 0604 2 :     retcludesc : ref vector [, long],
494 0605 2 :     foundflag : ref vector [, long];
495 0606 2 :   builtin
496 0607 2 :     nullparameter;
497 0608 2 :   local
498 0609 2 :     status,
499 0610 2 :     read_library,
500 0611 2 :     mhdbuf : block [lbr$C_maxhdrsiz, byte], ! BUFFER TO READ MODULE HEADER
501 0612 2 :     mhdbufdesc : block [dsc$C_s_bln, byte], ! STRING DESCRIPTOR OF MHDBUF
502 0613 2 :     bufdesc : block [dsc$C_s_bln, byte],
503 0614 2 :     prevclu : ref block [, byte], ! POINTER TO PREVIOUS CLUSTER
504 0615 2 :     nextclu : ref block [, byte],
505 0616 2 :     lastclu : ref block [, byte],
506 0617 2 :     fdb : ref block [, byte], ! POINTER TO CREATED FDB
507 0618 2 :     clu : ref block [, byte]; ! AND CLUSTER
508 0619 2 :
509 0620 2 : SEARCH THE CLUSTER LIST TO SEE IF THIS SHAREABLE IMAGE ALREADY REQUESTED
510 0621 2 :
511 0622 2 :   if not nullparameter (3) then retcludesc [0] = 0;
512 0623 2 :   if not nullparameter (4) then foundflag [0] = 0;
513 0624 2 :   if lib$lookup_tree (lnk$gl_clutree, .moduledesc, compareclu, clu) ! LOOK IT UP
514 0625 2 :   then
515 0626 2 :     begin
516 0627 2 :       if not nullparameter (4) then foundflag [0] = 1;
517 0628 2 :       if not nullparameter (3) then retcludesc [0] = .clu [node$l_ptr];
518 0629 2 :       return lnk$k_stopsearch; ! RETURN FALSE TO STOP SEARCH IF FOUND
519 0630 2 :     end;
520 0631 2 :
521 0632 2 : IMAGE NOT REQUESTED. READ AND VERIFY MODULE HEADER. THEN CREATE CLUSTER DESCRIPTOR
522 0633 2 :
523 0634 2 :   if (read_library = not nullparameter (2)) ! SET FLAG IF PARAMETER SPECIFIED
524 0635 2 :   then
525 0636 2 :     begin
526 0637 2 :
527 0638 2 :       if not (status = lbr$find (%ref (.lnk$gl_curfil [fdb$w_ifi]), .modulerfa)) ! POINT TO THE MODUL
528 0639 2 :       then
529 0640 2 :         begin
530 0641 2 :           signal ((lin$ libfind and not sts$m_severity) or sts$k_error, 4, ! REPORT ERROR
531 0642 2 :             .modulerfa [rfa$l_vbn], .modulerfa [rfa$w_offset], .moduledesc [dsc$a_pointer] - 1,
532 0643 2 :             lnk$gl_curfil [fdb$q_filename], lin$_format, 0, .status, .lbr$gl_rmsstv);
533 0644 2 :           return lnk$k_stopsearch; ! RETURN TO STOP SEARCH
534 0645 2 :         end;
535 0646 2 :
536 0647 2 :       bufdesc [dsc$w_length] = .lnk$al_rab [rab$w_usz];
537 0648 2 :       ! SET UP BUFFER DESCRIPTOR TO READ OBJ MODULE HEADER REC
538 0649 2 :       bufdesc [dsc$a_pointer] = .lnk$al_rab [rab$l_ubf];
539 0650 2 :       if not (status = lbr$get_record (%ref (.lnk$gl_curfil [fdb$w_ifi]), bufdesc, bufdesc))
540 0651 2 :       ! READ FIRST RECORD OF OBJ MODULE
541 0652 2 :       then
542 0653 2 :         begin
543 0654 2 :           signal (lin$_readerr, 1, lnk$gl_curfil [fdb$q_filename], .status, .lbr$gl_rmsstv);

```

```

544      return lnk$_stopsearch;
545      end;
546      mhdbufdesc [dsc$_length] = lbr$_maxhdrsiz;
547      ! READ LIBRARY MODULE HEADER...SET UP BUFFER DESCRIPTOR
548      mhdbufdesc [dsc$_pointer] = mhdbuf;
549      if not (status = [br$_set_module (%ref (.lnk$_gl_curfil [fdb$_w_ifi]), .modulerfa, ! READ IT NO
550      mhdbufdesc, mhdbufdesc))
551      then
552      begin
553      signal (lin$_readerr, 1, lnk$_gl_curfil [fdb$_q_filename], .status, .lbr$_gl_rmsstv);
554      return lnk$_stopsearch;
555      end;
556      begin
557      bind
558      hdbrec = .bufdesc [dsc$_pointer] : block [, byte], ! NAME THE HEADER RECORD
559      mhdid = hdbrec [mhd$_name] + .hdbrec [mhd$_namlng] : vector [, byte];
560      ! AND THE MODULE ID PART OF HEADER
561      if .hdbrec [obj$_b_rectyp] neq obj$_c_hdr ! MAKE SURE IT LOOKS LIKE AN OBJ MODULE HEADER
562      or .hdbrec [obj$_b_subtyp] neq obj$_c_hdr_mhd
563      then
564      begin
565      signal (lin$_readerr, 1, lnk$_gl_curfil [fdb$_q_filename], lin$_format, 0);
566      return lnk$_stopsearch;
567      end;
568      end;
569      end;
570      end;
571      ! OF READ_LIBRARY
572      NOW ALLOCATE A CLUSTER DESCRIPTOR FOR THE NEW SHAREABLE IMAGE
573      lnk$_alloccluster (clu, 1);
574      ! CREATE CLUSTER DESCRIPTOR, DON'T LINK INTO LIST
575      if not nullparameter (3)
576      ! IF CALLER WANTS DESCRIPTOR ADDRESS
577      then
578      retcludesc [0] = .clu;
579      ! THEN RETURN IT
580      lastclu = .lnk$_gl_curclu [clu$_l_lastclu];
581      ! GET POINTER TO LAST IMAGE CONTAINED IN THIS ONE
582      if .lastclu neq 0
583      ! IF THERE IS ONE, INSERT AFTER IT
584      then
585      begin
586      nextclu = .lastclu [clu$_l_nextclu];
587      lastclu [clu$_l_nextclu] = .clu;
588      clu [clu$_l_prevclu] = .lastclu;
589      end
590      else
591      begin
592      ! THIS IS THE FIRST, INSERT AFTER CURRENT CLUSTER
593      nextclu = .lnk$_gl_curclu [clu$_l_nextclu];
594      lnk$_gl_curclu [clu$_l_nextclu] = .clu;
595      clu [clu$_l_prevclu] = .lnk$_gl_curclu;
596      end;
597      if (clu [clu$_l_nextclu] = .nextclu) neq 0
598      ! SET PREVCLU IN NEXT CLUSTER
599      then
600      nextclu [clu$_l_prevclu] = .clu
601      else
602      lnk$_gl_lastclu = .clu;
603      ! OR MAKE THIS THE LAST CLUSTER IF IT IS

```

```
601 0712 2 lnk$gl_curclu [clu$l_lastclu] = .clu; ! MAKE THIS THE NEW LAST CLUSTER
602 0713 2 clu [clu$sv_shring] = true; ! FLAG CLUSTER AS SHAREABLE IMAGE
603 0714 2 clu [clu$sv_intclu] = true; ! FLAG AS INTERNALLY CREATED
604 0715 2 ch$move ((clu [clu$b_namng] = .moduledesc [dsc$w_length]), ! SET MODULE NAME INTO CLUSTER DESCRIPTOR
605 0716 2 .moduledesc [dsc$a_pointer], clu [clu$t_name]);
606 0717 2 lnk$insert_clu (.clu); ! INSERT CLUSTER INTO CLUSTER TREE
607 0718 2
608 0719 2
609 0720 2 if .read_library ! IF READING LIBRARY, SET MORE INFO INTO CLUSTER DESCRIPTOR
610 0721 2 then
611 0722 2 begin
612 0723 2 bind
613 0724 2 hdrec = .bufdesc [dsc$a_pointer] : block [, byte], ! NAME THE HEADER RECORD
614 0725 2 mhdid = hdrec [mhd$t_name] + .hdrec [mhd$b_namng] : vector [, byte],
615 0726 2 ! AND THE MODULE ID PART OF HEADER
616 0727 2 mhdgsmatch = mhdid [1] : long,
617 0728 2 mhdcredat = mhdid [0] + .mhdid [0] + 1 : vector [, byte], ! AND THE CREATE DATE/TIME
618 0729 2 modheader = .mhdbufdesc [dsc$a_pointer] : block [, byte], ! THE LIBRARY MODULE HEADER
619 0730 2 modgsmatch = modheader [mhd$t_objident] : long; ! THE GSMATCH STORED IN LIBRARY MODULE HEADE
620 0731 2
621 0732 2 lnk$bintim (mhdcredat, clu [clu$q_creat]); ! CONVERT CREATION DATE/TIME FOR LATER
622 0733 2 clu [clu$l_gsmatch] = .modgsmatch; ! SAVE THE GSMATCH FOUND IN THE LIBRARY
623 0734 2 end;
624 0735 2
625 0736 2 ALLOCATE AN FDB
626 0737 2
627 0738 2 lnk$allofdb (fdb);
628 0739 2 clu [clu$l_fstfdb] = clu [clu$l_lstfdb] = .fdb;
629 0740 2 lnk$alloblk ((fdb [fdb$w_usrnamlen] = .moduledesc [dsc$w_length]), fdb [fdb$l_usrnamadr]);
630 0741 2 ch$move (.fdb [fdb$w_usrnamlen], .moduledesc [dsc$a_pointer], .fdb [fdb$l_usrnamadr]);
631 0742 2
632 0743 2 if .lnk$gl_ctlmsk [lnk$sv_intfil]
633 0744 2 then
634 0745 2 ch$move (dsc$c_s_bln, shrdefext, fdb [fdb$w_defnamlen]) ! SET DEFAULT FILENAME STRING
635 0746 2 else
636 0747 2 begin
637 0748 2 local
638 0749 2 ptr
639 0750 2 ptr1;
640 0751 2
641 0752 2 THE DEFAULT FILENAME STRING CONSISTS OF THE RESULTANT
642 0753 2 FILENAME OF THE CURRENT FILE WITH THE EXTENSION SET TO ".EXE"
643 0754 2
644 0755 2 if (ptr = ch$find_ch (.lnk$gl_curfil [fdb$w_defnamlen], ! FIND END OF DIRECTORY
645 0756 2 .lnk$gl_curfil [fdb$l_defnamadr], %ascii'')) eql 0
646 0757 2 then
647 0758 2 ptr = ch$find_ch (.lnk$gl_curfil [fdb$w_defnamlen], .lnk$gl_curfil [fdb$l_defnamadr], %ascii'>')
648 0759 2
649 0760 2 ptr1 = ch$find_ch (.lnk$gl_curfil [fdb$w_defnamlen] - (.ptr - .lnk$gl_curfil [fdb$l_defnamadr]),
650 0761 2 .ptr, %ascii''); ! THEN FIND START OF EXTENSION
651 0762 2 lnk$alloblk (.ptr1 - .lnk$gl_curfil [fdb$l_defnamadr] + 4, fdb [fdb$l_defnamadr]);
652 0763 2 ! ALLOCATE BLOCK TO HOLD MODIFIED NAME(+4 FOR .EXE)
653 0764 2 ptr = ch$move (.ptr1 - .lnk$gl_curfil [fdb$l_defnamadr], .lnk$gl_curfil [fdb$l_defnamadr],
654 0765 2 ! MOVE IN FIRST PART OF NAME
655 0766 2 .fdb [fdb$l_defnamadr]);
656 0767 2 ptr1 = ch$move 74, uplit ('.EXE'), .ptr; ! SET THE EXTENSION
657 0768 2 fdb [fdb$w_defnamlen] = .ptr1 - .fdb [fdb$l_defnamadr]; ! COMPUTE LENGTH OF DEFAULT NAME
```

: 658  
: 659  
: 660  
: 661  
: 662  
: 663  
0769  
0770  
0771  
0772  
0773  
0774  
2  
2  
2  
2  
2  
1

```
end;  
ch$move (dsc$c_s_bln, lnk$gl_curfil [fdb$q_filename], fdb [fdb$q_libnamdsc]);  
fdb [fdb$v_shr] = true;  
return lnk$k_stopsearch  
end;
```

```
! COPY LIBRARY FILE DESCRIPTOR  
! FLAG FILE AS SHAREABLE IMAGE  
! RETURN FALSE TO STOP SEARCH  
! OF ADDIMAGE
```

.PSECT \$SPLITS,NOWRT,NOEXE,2

45 58 45 2E 00010 P.AAB: .ASCII \.EXE\ ;

.PSECT \$CODE\$,NOWRT,2

OFFC 00000

.ENTRY LNK\$ADDIMAGE, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 0563

5B	00000000G	8F	D0	00002	MOVL	R10,R11	
5A	00000000G	00	9E	00009	MOVAB	#LINS FORMAT, R11	
59	00000000G	00	9E	00010	MOVAB	LBR\$GC_RMSSTV, R10	
5E	FF64	CE	9E	00017	MOVAB	LNK\$GL_CURFIL, R9	
03		6C	91	0001C	MOVAB	-156(SP), SP	
		08	1F	0001F	CMPB	(AP), #3	0622
		0C	AC	D5 00021	BLSSU	1\$	
		03	13	00024	TSTL	12(AP)	
		0C	BC	D4 00026	BEQL	1\$	
04		6C	91	00029 1\$:	CLRL	@RETCLUDESC	
		08	1F	0002C	CMPB	(AP), #4	0623
		10	AC	D5 0002E	BLSSU	2\$	
		03	13	00031	TSTL	16(AP)	
		10	BC	D4 00033	BEQL	2\$	
		04	AE	9F 00036 2\$:	CLRL	@FOUNDFLAG	
		9E	AF	9F 00039	PUSHAB	CLU	0624
58		04	AC	D0 0003C	PUSHAB	COMPARECLU	
		58	DD	00040	MOVL	MODULEDESC, R8	
		00	9F	00042	PUSHL	R8	
00000000G		00	FB	00048	PUSHAB	LNK\$GL CLUTREE	
		04	FB	00048	CALLS	#4, LIB\$LOOKUP_TREE	
		23	50	E9 0004F	BLBC	R0, 4\$	
		04	6C	91 00052	CMPB	(AP), #4	0627
			09	1F 00055	BLSSU	3\$	
		10	AC	D5 00057	TSTL	16(AP)	
			04	13 0005A	BEQL	3\$	
10		BC	01	D0 0005C	MOVL	#1, @FOUNDFLAG	
		03	6C	91 00060 3\$:	CMPB	(AP), #3	0628
			70	1F 00063	BLSSU	8\$	
		0C	AC	D5 00065	TSTL	12(AP)	
			6B	13 00068	BEQL	8\$	
		50	AE	D0 0006A	MOVL	CLU, R0	
0C		BC	0A	D0 0006E	MOVL	10(R0), @RETCLUDESC	
			60	11 00073	BRB	8\$	0629
		02	6C	91 00075 4\$:	CMPB	(AP), #2	0634
			05	1E 00078	BGEQU	5\$	
		56	01	D0 0007A	MOVL	#1, R6	
			09	11 0007D	BRB	6\$	
			56	D4 0007F 5\$:	CLRL	R6	

		08	AC	D5	00081	TSTL	8(AP)	
			02	12	00084	BNEQ	6\$	
			56	D6	00086	INCL	R6	
	56		56	D2	00088	6\$: MCOML	R6, READ_LIBRARY	
	03		56	E8	0008B	BLBS	READ_LIBRARY, 7\$	
			00CC	31	0008E	BRW	14\$	
		52	08	AC	D0	7\$: MOVL	MODULERFA, R2	0638
			52	DD	00095	PUSHL	R2	
	50		69	D0	00097	MOVL	LNK\$GL_CURFIL, R0	
	04	AE	24	A0	3C	MOVZWL	36(R0), 4(SP)	
			04	AE	9F	PUSHAB	4(SP)	
00000000G	00		02	FB	000A2	CALLS	#2, LBR\$FIND	
	53		50	D0	000A9	MOVL	R0, STATUS	
	29		53	E8	000AC	BLBS	STATUS, 9\$	0643
			6A	DD	000AF	PUSHL	LBR\$GL_RMSSTV	
			53	DD	000B1	PUSHL	STATUS	
			7E	D4	000B3	CLRL	-(SP)	
			5B	DD	000B5	PUSHL	R11	
7E		69	14	C1	000B7	ADDL3	#20, LNK\$GL_CURFIL, -(SP)	
7E	04	AB	01	C3	000BB	SUBL3	#1, 4(R8), -(SP)	0642
		7E	04	A2	3C	MOVZWL	4(R2), -(SP)	0643
				62	DD	PUSHL	(R2)	
			04	DD	000C6	PUSHL	#4	
			00000000*	8F	DD	PUSHL	#<<LINS LIBFIND&-8>!2>	
00000000G	00		0A	FB	000CE	CALLS	#10, LIB\$SIGNAL	
			01DA	31	000D5	8\$: BRW	27\$	0644
			00	B0	000D8	9\$: MOVW	LNK\$AL_RAB+32, BUFDESC	0647
0C	AE	00000000G	00	D0	000E0	MOVL	LNK\$AL_RAB+36, BUFDESC+4	0649
10	AE	00000000G	0C	AE	9F	PUSHAB	BUFDESC	0650
			10	AE	9F	PUSHAB	BUFDESC	
			50	D0	000EE	MOVL	LNK\$GL_CURFIL, R0	
	08	AE	24	A0	3C	MOVZWL	36(R0), 8(SP)	
			08	AE	9F	PUSHAB	8(SP)	
00000000G	00		03	FB	000F9	CALLS	#3, LBR\$GET_RECORD	
	53		50	D0	00100	MOVL	R0, STATUS	
	2A		53	E9	00103	BLBC	STATUS, 10\$	
	14	AE	80	8F	9B	MOVZBW	#128, MHDBUFDESC	0657
	18	AE	1C	AE	9E	MOVAB	MHDBUF, MHDBUFDESC+4	0659
			14	AE	9F	PUSHAB	MHDBUFDESC	0660
			18	AE	9F	PUSHAB	MHDBUFDESC	
			52	DD	00116	PUSHL	R2	
	50		69	D0	00118	MOVL	LNK\$GL_CURFIL, R0	
	0C	AE	24	A0	3C	MOVZWL	36(R0), 12(SP)	
			0C	AE	9F	PUSHAB	12(SP)	
00000000G	00		04	FB	00123	CALLS	#4, LBR\$SET_MODULE	
	53		50	D0	0012A	MOVL	R0, STATUS	
	06		53	E8	0012D	BLBS	STATUS, 11\$	
			6A	DD	00130	10\$: PUSHL	LBR\$GL_RMSSTV	0664
			53	DD	00132	PUSHL	STATUS	
			11	11	00134	BRB	13\$	
	50		10	AE	D0	11\$: MOVL	BUFDESC+4, R0	0669
			60	95	0013A	TSTB	(R0)	0672
			05	12	0013C	BNEQ	12\$	
			01	A0	95	TSTB	1(R0)	0673
			1A	13	00141	BEQL	14\$	
			7E	D4	00143	12\$: CLRL	-(SP)	0676
			5B	DD	00145	PUSHL	R11	

7E	69		14	C1	00147	13\$:	ADDL3	#20, LNK\$GL_CURFIL, -(SP)	
			01	DD	0014B		PUSHL	#1	
00000000G	00	00000000G	8F	DD	0014D		PUSHL	#LINS READERR	
			05	FB	00153		CALLS	#5, LIB\$SIGNAL	
			0155	31	0015A		BRW	27\$	0677
			01	DD	0015D	14\$:	PUSHL	#1	0685
			08	AE	9F 0015F		PUSHAB	CLU	
00000000G	00		02	FB	00162		CALLS	#2, LNK\$ALLOCLUSTER	
	03		6C	91	00169		CMPB	(AP), #3	0686
			0A	1F	0016C		BLSSU	15\$	
			0C	AC	D5 0016E		TSTL	12(AP)	
			G5	13	00171		BEQL	15\$	
0C	BC	00000000G	04	AE	D0 00173		MOVL	CLU, @RETCLUDESC	0688
	50		00	D0	00178	15\$:	MOVL	LNK\$GL_CURCLU, R0	0690
	52		A0	D0	0017F		MOVL	36(R0) - LASTCLU	
	57		04	AE	D0 00183		MOVL	CLU, R7	0696
			52	D5	00187		TSTL	LASTCLU	0692
			0C	13	00189		BEQL	16\$	
	51		62	D0	0018B		MOVL	(LASTCLU), NEXTCLU	0695
	62		57	D0	0018E		MOVL	R7, (LASTCLU)	0696
04	A7		52	D0	00191		MOVL	LASTCLU, 4(R7)	0697
			0A	11	00195		BRB	17\$	0692
	51		60	D0	00197	16\$:	MOVL	(R0), NEXTCLU	0701
	60		57	D0	0019A		MOVL	R7, (R0)	0702
04	A7		50	D0	0019D		MOVL	R0, 4(R7)	0703
	67		51	D0	001A1	17\$:	MOVL	NEXTCLU, (R7)	0706
			06	13	001A4		BEQL	18\$	
04	A1		57	D0	001A6		MOVL	R7, 4(NEXTCLU)	0708
			07	11	001AA		BRB	19\$	
00000000G	00		57	D0	001AC	18\$:	MOVL	R7, LNK\$GL_LASTCLU	0710
	24		57	D0	001B3	19\$:	MOVL	R7, 36(R0)	0712
	58		8F	A8	001B7		BISW2	#516, 88(R7)	0714
	50	0204	68	3C	001BD		MOVZWL	(R8), R0	0715
	5C		50	90	001C0		MOVB	R0, 92(R7)	
SD	A7		50	28	001C4		MOV3	R0, @4(R8), 93(R7)	0717
	04		57	DD	001CA		PUSHL	R7	0718
00000000G	00		01	FB	001CC		CALLS	#1, LNK\$INSERT_CLU	
	26		56	E9	001D3		BLBC	READ LIBRARY, 20\$	0720
	51		10	AE	D0 001D6		MOVL	BUFDESC+4, R1	0724
	50		05	A1	9A 001DA		MOVZBL	5(R1), R0	0725
	50		06	A140	9E 001DE		MOVAB	6(R1)[R0], R0	
	51		60	9A	001E3		MOVZBL	(R0), R1	0728
52	18		12	C1	001E6		ADDL3	#18, MHDBUFDESC+4, R2	0730
			A7	9F	001EB		PUSHAB	48(R7)	0732
			01	A14C	9F 001EE		PUSHAB	1(R1)[R0]	
	FDB1	CF	02	FB	001F2		CALLS	#2, LNK\$BINTIM	
	0084	C7	62	D0	001F7		MOVL	(R2), 132(R7)	0733
			08	AE	9F 001FC	20\$:	PUSHAB	FDB	0738
00000000G	00		01	FB	001FF		CALLS	#1, LNK\$ALLOFDB	
	56		08	AE	D0 00206		MOVL	FDB, R6	0739
	0C		56	D0	0020A		MOVL	R6, 12(R7)	
	08		56	D0	0020E		MOVL	R6, 8(R7)	
			10	A6	9F 00212		PUSHAB	16(R6)	0740
			68	3C	00215		MOVZWL	(R8), -(SP)	
	0C		6E	B0	00218		MOVW	(SP), 12(R6)	
00000000G	00		02	FB	0021C		CALLS	#2, LNK\$ALLOBLK	
10	B6		0C	A6	28 00223		MOV3	12(R6), @4(R8), @16(R6)	0741

14	OB	00000000G	00	03	E1	0022A	BBC	#3, LNK\$GL CTLMSK+1, 21\$	0743	
	A6	00000000'	EF	08	28	00232	MOV C3	#8, SHRDEFEXT, 20(R6)	0745	
			52	68	11	0023B	BRB	26\$		
18	B2	14	A2	5D	69	D0 0023D	21\$:	MOVL	LNK\$GL CURFIL, R2	0755
					8F	3A 00240		LOCC	#93, 20(R2), @24(R2)	
			53		02	12 00247		BNEQ	22\$	
					51	D4 00249		CLRL	R1	
					51	D0 0024B	22\$:	MOVL	R1, PTR	
18	B2	14	A2		0D	12 0024E		BNEQ	24\$	0756
					3E	3A 00250		LOCC	#62, 20(R2), @24(R2)	0758
			53		02	12 00256		BNEQ	23\$	
					51	D4 00258		CLRL	R1	
			53		51	D0 0025A	23\$:	MOVL	R1, PTR	
	50	18	A2		53	C3 0025D	24\$:	SUBL3	PTR, 24(R2), R0	0760
			51	14	A2	3C 00262		MOVZWL	20(R2), R1	
			50		51	C0 00266		ADDL2	R1, R0	
	63		50		2E	3A 00269		LOCC	#46, R0, (PTR)	
					02	12 0026D		BNEQ	25\$	
			57		51	D4 0026F		CLRL	R1	
					51	D0 00271	25\$:	MOVL	R1, PTR1	
			57	18	A6	9F 00274		PUSHAB	24(R6)	0762
	51		57	18	A2	C3 00277		SUBL3	24(R2), PTR1, R1	
				04	A1	9F 0027C		PUSHAB	4(R1)	
		00000000G	00		02	FB 0027F		CALLS	#2, LNK\$ALLOBLK	
			50		69	D0 00286		MOVL	LNK\$GL CURFIL, R0	0764
	51		57	18	A0	C3 00289		SUBL3	24(R0), PTR1, R1	
18	B6	18	B0		51	28 0028E		MOV C3	R1, @24(R0), @24(R6)	0766
			63		EF	D0 00294		MOVL	P.AAB, (PTR)	0767
			57	04	A3	9E 0029B		MOVAB	4(R3), PTR1	
14	A6		57	18	A6	A3 0029F		SUBW3	24(R6), PTR1, 20(R6)	0768
			50		69	D0 002A5	26\$:	MOVL	LNK\$GL CURFIL, R0	0770
1C	A6	14	A0		08	28 002AB		MOV C3	#8, 20(R0), 28(R6)	
		0A	A6		04	88 002AE		BISB2	#4, 10(R6)	0772
					50	D4 002B2	27\$:	CLRL	R0	0774
					04	002B4		RET		

: Routine Size: 693 bytes, Routine Base: \$CODE\$ + 0369

: 664 0775 1  
: 665 0776 1 end  
: 666 0777 0 eludom

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
\$PLITS	20	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$OWNS	24	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$GLOBALS	20	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODE\$	1566	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
. ABS .	0	NOVEC, NOWRT, NORD, NOEXE, NOSHR, LCL, ABS, CON, NOPIC, ALIGN(0)



Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	24	0	581	00:01.0
\$_\$255\$DUA28:[LINKER.OBJ]DATBAS.L32;1	538	38	7	28	00:00.5

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:LNKPROLIB/OBJ=OBJ\$:LNKPROLIB MSRC\$:LNKPROLIB/UPDATE=(ENH\$:LNKPROLIB)  
; 667 0778 0 ! End of module  
; Size: 1566 code + 64 data bytes  
; Run Time: 00:28.7  
; Elapsed Time: 01:01.8  
; Lines/CPU Min: 1624  
; Lexemes/CPU-Min: 17402  
; Memory Used: 242 pages  
; Compilation Complete



LNKPROLTB  
LIS

LNKSYMTBL  
LIS

LNKSYMOUT  
LIS

LNKUMALLO  
LIS

LNKPSCTBL  
LIS

LNKPROSHR  
LIS

LNKSTATSD  
LIS