

[illegible][illegible]

[illegible]

```

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 LIB_PROCMD ( ! Process LIBRARY command
2 0002 0 LANGUAGE (BLISS32),
3 0003 0 IDENT = 'V04-000'
4 0004 0 ) =
5 0005 1 BEGIN
6 0006 1
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: Library command processor
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1 The VAX/VMS librarian is invoked by DCL to process the LIBRARY
38 0038 1 command. It utilizes the librarian procedure set to perform
39 0039 1 the actual modifications to the library.
40 0040 1
41 0041 1 ENVIRONMENT:
42 0042 1
43 0043 1 VAX native, user mode.
44 0044 1
45 0045 1 --
46 0046 1
47 0047 1
48 0048 1 AUTHOR: Benn Schreiber, CREATION DATE: 13-June-1979
49 0049 1
50 0050 1 MODIFIED BY:
51 0051 1
52 0052 1 V03-003 MCN0145 Maria del C. Nasr 07-Feb-1984
53 0053 1 Call LIB_COMPRS_LIB too for the /DATA qualifier.
54 0054 1 The impact this has, is that whenever the /DATA
55 0055 1 qualifier is used, a /COMPRESS is done by default.
56 0056 1
57 0057 1 V03-002 JWT0068 Jim Teague 24-Nov-1982
```

LIB PROCMD
V04=000

B 5
16-Sep-1984 02:03:14
14-Sep-1984 12:38:09

VAX-11 Bliss-32 V4.0-742
[LIBRAR.SRC]PROCMD.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 !

Passed related name block for creating compressed
libraries. This ensures, along with an edit in
[lbr.src]openclase.b32, that compressed libraries
will end up in the same directory that the originals
reside in.

V03-001 JWT0056 Jim Teague 16-Sep-1982
Equipped with DCX interface for /COMPRESS=REDUCE.

```
: 71      0070 1 LIBRARY
: 72      0071 1      'SYS$LIBRARY:STARLET.L32';      !System data structures
: 73      0072 1 REQUIRE
: 74      0073 1      'PREFIX';                      !Librarian macros
: 75      0257 1 REQUIRE
: 76      0258 1      'LIBDEF';                      !Librarian data structures
: 77      0546 1 REQUIRE
: 78      0547 1      'LBRDEF';                      !Library processor definitions
: 79      1138 1
: 80      1139 1 FORWARD ROUTINE
: 81      1140 1      add_del_mod,                  !Add new module to list of modules to delete
: 82      1141 1      add_rem_sym,                  !Add new symbol to list of symbols to removed
: 83      1142 1      getkeysize,                    !Get maximum keysize from library header
: 84      1143 1      lib_log_op,                    !Log operation on console
: 85      1144 1      lib_create_lib,                 !Create library
: 86      1145 1      lib_remove_syms,                !Remove global symbols from object library
: 87      1146 1      lib_delete_mods,                !Delete modules from library
: 88      1147 1      lib_insert_mods,                !Insert modules into library
: 89      1148 1      lib_log_upd,                    !Build updated modules name list
: 90      1149 1      lib_put_luh;                    !Put library update history record
: 91      1150 1
: 92      1151 1 EXTERNAL ROUTINE
: 93      1152 1      next_input_file,
: 94      1153 1      lib_extnct_mods,
: 95      1154 1      lbr$ini_control : ADDRESSING MODE (GENERAL), !Initialize library control table
: 96      1155 1      lbr$open : ADDRESSING MODE (GENERAL),      !Open library
: 97      1156 1      lbr$close : ADDRESSING MODE (GENERAL),     !Close library
: 98      1157 1      lbr$set_index : ADDRESSING MODE (GENERAL),  !Set index number
: 99      1158 1      lbr$set_locate : ADDRESSING MODE (GENERAL), !Set locate mode
: 100     1159 1      lbr$get_header : ADDRESSING MODE (GENERAL), !Get header info
: 101     1160 1      lbr$get_index : ADDRESSING MODE (GENERAL),  !Return contents of index
: 102     1161 1      lbr$delete_key : ADDRESSING MODE (GENERAL), !Delete symbol from index
: 103     1162 1      lbr$put_history : ADDRESSING MODE (GENERAL), !Put a LUH record
: 104     1163 1      lbr$ret_rmsstv : ADDRESSING MODE (GENERAL), !Return RMS status
: 105     1164 1      sys$fao : ADDRESSING MODE (GENERAL),        !Format ASCII
: 106     1165 1      lib$put_output : ADDRESSING_MODE (GENERAL), !Write output to terminal
: 107     1166 1      lib_get_mem,                    !allocate virtual memory
: 108     1167 1      lib_free_mem,                   !deallocate virtual memory
: 109     1168 1      lib_comprs_lib,                  !Compress the library
: 110     1169 1      lib_cross_obj,                  !Cross reference of obj lib
: 111     1170 1      lib_list_lib;                    !List contents of library
: 112     1171 1
: 113     1172 1 EXTERNAL
: 114     1173 1      lbr$gl_control : REF BBLOCK ADDRESSING_MODE (GENERAL), !Librarian control block
: 115     1174 1      lib$gl_keysize,                  !Max length of keys in library
: 116     1175 1      lib$gl_allblks,                  !Number of blocks to allocate
: 117     1176 1      lib$gl_allgbls,                  !Number of globals to allocate
: 118     1177 1      lib$gl_allmods,                  !Number of modules
: 119     1178 1      lib$gl_allksz,                   !Size of keys in index from command
: 120     1179 1      lib$gl_allhis,                   !number of Library Update history records
: 121     1180 1      lib$gl_allver,                   !Version of library to create
: 122     1181 1      lib$gl_cre8flags : BITVECTOR,      !Flags for create/compress options set by command
: 123     1182 1      lib$al_deldspat : VECTOR [LONG],    !Delete module dispatch vector
: 124     1183 1      lib$al_inpdspat : VECTOR [LONG],    !Input processing dispatch vector
: 125     1184 1      lib$al_tynames : VECTOR [lbr$c_typ_decmx+1, LONG], !Library type names
: 126     1185 1      lib$al_hdrlen : VECTOR [lbr$c_typ_decmx+1, LONG], !Length of extra header bytes
: 127     1186 1      lib$al_ascbinf : VECTOR [lbr$c_typ_decmx+1, LONG], !ASCII/binary flags
```

```

: 128      1187 1      lib$al_numidx : VECTOR [lbr$c_typ_decmx+1, LONG], !Number of indices
: 129      1188 1      lib$al_idxopt : VECTOR [lbr$c_typ_decmx+1, LONG], !Number of indices
: 130      1189 1      lib$gl_modnamix,      !Module name index number
: 131      1190 1      lib$gl_objgsdix,      !index number for object lib gsd index
: 132      1191 1      lib$gl_type,          !Type of library requested
: 133      1192 1      lib$gl_libctl,        !Control index for library
: 134      1193 1      lib$gl_objsyml : VECTOR [2],      !listhead for deleted gsd symbols
: 135      1194 1      lib$gl_delmodl : VECTOR [2],      !listhead for deleted module names
: 136      1195 1      lib$gl_modupdl : VECTOR [2],      !listhead for updated module names
: 137      1196 1      lib$gl_ctlmsk : BLOCK [2],        !Control mask
: 138      1197 1      lib$gl_tmprfdb : REF BBLOCK,      !Pointer to temporary FDB
: 139      1198 1      lib$gl_outfdb : REF BBLOCK,      !Pointer to output FDB
: 140      1199 1      lib$gl_libfdb : REF BBLOCK;       !Pointer to library FDB
: 141      1200 1
: 142      1201 1      EXTERNAL LITERAL
: 143      1202 1      lbr$_errclose,
: 144      1203 1      lbr$_nomtchfou,
: 145      1204 1      lbr$_oldlibrary,
: 146      1205 1      lbr$_oldmismch,
: 147      1206 1      lbr$_typmismch,
: 148      1207 1
: 149      1208 1      lib$_brknlbr,          !Library improperly close
: 150      1209 1      lib$_delkeyerr,       !Error deleting a key
: 151      1210 1      lib$_diftyp,          !Different type than expected
: 152      1211 1      lib$_histerr,         !error in update history
: 153      1212 1      lib$_indexerr,        !Index error
: 154      1213 1      lib$_initerr,         !Initization error
: 155      1214 1      lib$_nomtchfou,       !No match found
: 156      1215 1      lib$_notobjlib,       !Not object library
: 157      1216 1      lib$_removed;         !Removed a symbol
: 158      1217 1
: 159      1218 1      OWN
: 160      1219 1      textrfa : VECTOR [rfa$c_length, BYTE];
```

```
162 1220 1 GLOBAL ROUTINE lib_process_cmd =
163 1221 2 BEGIN
164 1222 2
165 1223 2 ++
166 1224 2
167 1225 2 FUNCTIONAL DESCRIPTION:
168 1226 2
169 1227 2 This routine processes the command line parsed by lib_get_command.
170 1228 2
171 1229 2 CALLING SEQUENCE:
172 1230 2
173 1231 2 INPUT PARAMETERS:
174 1232 2 NONE
175 1233 2
176 1234 2 IMPLICIT INPUTS:
177 1235 2
178 1236 2 Command has been parsed, data base set up
179 1237 2
180 1238 2 OUTPUT PARAMETERS:
181 1239 2 NONE
182 1240 2
183 1241 2 IMPLICIT OUTPUTS:
184 1242 2
185 1243 2 Command processed
186 1244 2
187 1245 2 --
188 1246 2
189 1247 2 BIND
190 1248 2 libnamblk = lib$gl_libfdb [fdb$t_nam] : BBLOCK; !Point to NAM block for library
191 1249 2
192 1250 2 LOCAL
193 1251 2 status,
194 1252 2 func;
195 1253 2
196 1254 2 func = lbr$c_read; !Assume function is read
197 1255 2 IF .lib$gl_ctlmsk [lib$v_create] !If function is create
198 1256 2 THEN func = lbr$c_create
199 1257 2 ELSE IF .lib$gl_ctlmsk [lib$v_delete] !If function is delete
200 1258 2 OR .lib$gl_ctlmsk [lib$v_insert] !or insert or replace
201 1259 2 OR .lib$gl_ctlmsk [lib$v_remove] !or remove
202 1260 2 THEN func = lbr$c_update; !then function is update
203 1261 2
204 1262 2 initialize the library control table
205 1263 2
206 P 1264 2 perform (lbr$ini_control (lib$gl_libctl, func, lib$gl_type, libnamblk),
207 1265 2 [lib$_initerr, 1, lib$gl_libfdb [fdb$l_namdesc]]);
208 1266 2 IF .lib$gl_ctlmsk [lib$v_create] !Creating?
209 1267 2 THEN BEGIN
210 1268 2 perform (lib_create_lib (.lib$gl_libfdb, lib$gl_libctl));
211 1269 2 lib$gl_ctlmsk [lib$v_libopen] = true; !flag library open
212 1270 2 END
213 1271 2 ELSE BEGIN !Not create--open library
214 1272 2 status = lbr$open (lib$gl_libctl, lib$gl_libfdb [fdb$l_namdesc]); !Then open it
215 1273 2 IF .status EQL lbr$_oldlibrary
216 1274 2 OR .status EQL lbr$_oldmismch
217 1275 2 THEN BEGIN
218 1276 2 lib$gl_ctlmsk [lib$v_oldlib] = true;
```

```
219 1277 3      END;
220 1278 3      IF .status EQL lbr$_typmismch      !If library type mismatch
221 1279 3      OR .status EQL lbr$_oldmismch      ! or old format library mismatch
222 1280 4      THEN BEGIN
223 1281 4          IF .lib$gl_type EQL lbr$_type obj
224 1282 4              AND lbr$_ret_rmsstv() EQL lbr$_type_shstb
225 1283 4              THEN lib$gl_ctlmsk [lib$_shrstb] = true
226 1284 5          ELSE BEGIN
227 1285 5              SIGNAL (lib$_diftyp, 3, .lib$_typnames [lbr$_ret_rmsstv()],
228 1286 5                  lib$gl_libfdb [fdb$_namdesc],
229 1287 5                  lib$_typnames [.lib$gl_type]);
230 1288 5              lib$gl_ctlmsk [lib$_shrstb] = false;      !Guaranteed not one of these
231 1289 4          END;
232 1290 4      END
233 1291 3      ELSE IF NOT .status
234 1292 3      THEN
235 1293 4          BEGIN
236 1294 4              IF .status EQL lbr$_errclose
237 1295 4              THEN
238 1296 5                  BEGIN
239 1297 5                      IF .func EQL lbr$_update
240 1298 5                      THEN SIGNAL_STOP (lib$_brknlbr, 1, lib$gl_libfdb [fdb$_namdesc], lbr$_errclose)
241 1299 5                      ELSE SIGNAL (lib$_brknlbr, 1, lib$gl_libfdb [fdb$_namdesc], lbr$_errclose);
242 1300 5                  END
243 1301 4              ELSE
244 1302 4                  SIGNAL_STOP (lib$_openin, 1, lib$gl_libfdb [fdb$_namdesc],
245 1303 4                      .status);
246 1304 3              END;
247 1305 3          lib$gl_type = lbr$_ret_rmsstv();      !Set type of library opened
248 1306 3          getkeysize ();      !Get key size from library header
249 1307 3          lib$gl_ctlmsk [lib$_libopen] = true;      !flag library open
250 1308 3      END;
251 1309 2      lbr$_set_locate (lib$gl_libctl);      !Set locate mode
252 1310 2      IF .func EQL lbr$_update
253 1311 2      THEN
254 1312 2          BEGIN
255 1313 2              LOCAL
256 1314 2              headary : BBLOCK [lbr$_pagesize];
257 1315 2              perform ( lbr$_get_header (lib$gl_libctl, headary),
258 1316 2                  lib$_indexerr, 1, lib$gl_libfdb [fdb$_namdesc]);
259 1317 2              IF .headary [lhi$_maxluhrec] GTR 0
260 1318 2              THEN lib$gl_ctlmsk [lib$_recordluh] = true;
261 1319 2          END;
262 1320 2      !
263 1321 2      ! If going to modify old library, then convert to new format first
264 1322 2      !
265 1323 2      IF .lib$gl_ctlmsk [lib$_oldlib]
266 1324 2      AND (.lib$gl_ctlmsk [lib$_delete]
267 1325 2          OR .lib$gl_ctlmsk [lib$_insert]
268 1326 2          OR .lib$gl_ctlmsk [lib$_remove])
269 1327 2      THEN BEGIN
270 1328 2          !then compress to a new library
271 1329 2          LOCAL
272 1330 2          saveoutfdb;
273 1331 2          saveoutfdb = .lib$gl_outfdb;      !Save old output FDB
274 1332 2          lib$gl_outfdb = .lib$gl_tmpfdb;      !And use temporary FDB
275 1333 2      END
```

```
276 1334 lib$gl_ctlmsk [lib$v_convert] = true; !Flag to issue convert message
277 1335 perform (lib_comprs_lib (lbr$c_update)); !Compress to new format library
278 1336 getkeysize (); !Get keysize from library header
279 1337 lib$gl_ctlmsk [lib$v_convert] = false; !No longer converting
280 1338 lib$gl_outfdb = .saveoutfdb;
281 1339 lbr$set_locate (lib$gl_libctl); !Set locate mode
282 1340 END;
283 1341
284 1342 ! Execute the requested operations
285 1343
286 1344 IF .lib$gl_ctlmsk [lib$v_delete] !Delete modules?
287 1345 THEN
288 1346 BEGIN
289 1347 lib_delete_mods ();
290 P 1348 perform (lib_put_luh (lhe$c_deleted),
291 1349 lib$_histerr, 1, lib$gl_libfdb [fdb$_namdesc]);
292 1350 END;
293 1351 IF .lib$gl_ctlmsk [lib$v_insert] !Insert or replace
294 1352 THEN
295 1353 BEGIN
296 1354 lib_insert_mods ();
297 P 1355 perform (lib_put_luh (lhe$c_inserted),
298 1356 lib$_histerr, 1, lib$gl_libfdb [fdb$_namdesc]);
299 P 1357 perform (lib_put_luh (lhe$c_replaced),
300 1358 lib$_histerr, 1, lib$gl_libfdb [fdb$_namdesc]);
301 1359 END;
302 1360 IF .lib$gl_ctlmsk [lib$v_remove]
303 1361 THEN lib_remove_syms ();
304 1362
305 1363 IF .lib$gl_ctlmsk [lib$v_compress]
306 1364 OR .lib$gl_ctlmsk [lib$v_data]
307 1365 THEN
308 1366 BEGIN
309 1367 lib_comprs_lib (lbr$c_read);
310 1368 lbr$set_locate (lib$gl_libctl); !Reset locate mode after compress
311 1369 getkeysize ();
312 1370 END;
313 1371
314 1372 IF .lib$gl_ctlmsk [lib$v_extract]
315 1373 THEN lib_extact_mods ();
316 1374 IF .lib$gl_ctlmsk [lib$v_cross]
317 1375 THEN lib_cross_obj ();
318 1376 IF .lib$gl_ctlmsk [lib$v_list]
319 1377 THEN lib_list_lib ();
320 1378 IF .lib$gl_ctlmsk [lib$v_libopen] !If library is open now
321 1379 THEN BEGIN
322 P 1380 rms_perform (lbr$close (lib$gl_libctl), !then close it
323 P 1381 lib$_closein, lbr$_ret_rmsstg(),
324 1382 1, lib$gl_libfdb [fdb$_namdesc]);
325 1383 lib$gl_ctlmsk [lib$v_libopen] = false; !mark it closed
326 1384 END;
327 1385 RETURN true
328 1386 END;
```

.TITLE LIB PROCMD
.IDENT \V04-000\

.PSECT \$OWNS,NOEXE,2
00000 TEXTRFA: .BLKB 6

.EXTRN NEXT_INPUT_FILE
.EXTRN LIB_EXTRCT_MODS
.EXTRN LBR\$INI_CONTROL
.EXTRN LBR\$OPEN, LBR\$CLOSE
.EXTRN LBR\$SET_INDEX, LBR\$SET_LOCATE
.EXTRN LBR\$GET_HEADER, LBR\$GET_INDEX
.EXTRN LBR\$DELETE_KEY, LBR\$PUT_HISTORY
.EXTRN LBR\$RET_RMSSTV, SYSS\$FAO
.EXTRN LIB\$PUT_OUTPUT, LIB_GET_MEM
.EXTRN LIB_FREE_MEM, LIB_COMPRS_LIB
.EXTRN LIB_CROSS_OBJ, LIB_LIST_LIB
.EXTRN LBR\$GL_CONTROL, LIB\$GL_KEYSIZE
.EXTRN LIB\$GL_ALLBLKS, LIB\$GL_ALLGBLS
.EXTRN LIB\$GL_ALLMODS, LIB\$GL_ALLKSZ
.EXTRN LIB\$GL_ALLHIS, LIB\$GL_ALLVER
.EXTRN LIB\$GL_CRE8FLAGS
.EXTRN LIB\$AL_DELDSPAT
.EXTRN LIB\$AL_INPDSPAT
.EXTRN LIB\$AL_TYPNAMES
.EXTRN LIB\$AL_HDRLEN, LIB\$AL_ASCBINF
.EXTRN LIB\$AL_NUMIDX, LIB\$AL_IDXOPT
.EXTRN LIB\$GL_MODNAMIX
.EXTRN LIB\$GL_OBJGSDIX
.EXTRN LIB\$GL_TYPE, LIB\$GL_LIBCTL
.EXTRN LIB\$GL_OBJSYRML
.EXTRN LIB\$GL_DELMODL, LIB\$GL_MODUPDL
.EXTRN LIB\$GL_CTLMSK, LIB\$GL_TMPFDB
.EXTRN LIB\$GL_OUTFDB, LIB\$GL_LIBFDB
.EXTRN LBR\$_ERRCLOSE, LBR\$_NOMTCHFOU
.EXTRN LBR\$_OLDLIBRARY
.EXTRN LBR\$_OLDMISMCH, LBR\$_TYPMISMCH
.EXTRN LIB\$_BRKNLIB, LIB\$_DELKEYERR
.EXTRN LIB\$_DIFTYP, LIB\$_RISTERR
.EXTRN LIB\$_INDEXERR, LIB\$_INITERR
.EXTRN LIB\$_NOMTCHFOU, LIB\$_NOTOBJLIB
.EXTRN LIB\$_REMOVED

.PSECT \$CODE\$,NOWRT,2

OFFC 00000

5B 0000G CF 9E 00002
5A 00000000G 8F D0 00007
59 00000000G 00 9E 0000E
58 00000000G 8F D0 00015
57 00000000G 00 9E 0001C
56 0000G CF 9E 00023
55 0000G CF 9E 00028
54 00000000G 00 9E 0002D
53 0000G CF 9E 00034
5E FE00 CE 9E 00039
65 00000040 8F C1 0003E

.ENTRY LIB_PROCESS_CMD, Save R2,R3,R4,R5,R6,R7,R8,-: 1220
R9,R10,R11
MOVAB LIB\$GL_TYPE, R11
MOVL #LIB\$_RISTERR, R10
MOVAB LBR\$SET_LOCATE, R9
MOVL #LBR\$_ERRCLOSE, R8
MOVAB LBR\$RET_RMSSTV, R7
MOVAB LIB\$GL_LIBCTL, R6
MOVAB LIB\$GL_LIBFDB, R5
MOVAB LIB\$SIGNAL, R4
MOVAB LIB\$GL_CTLMSK, R3
MOVAB -512(SP), SP
ADDL3 #64, LIB\$GL_LIBFDB, R0 : 1248

				01	DD	00046	PUSHL	#1	:	1254
				63	95	00048	TSTB	LIB\$GL_CTLMSK	:	1255
				04	18	0004A	BGEQ	1\$	:	
				6E	D4	0004C	CLRL	FUNC	:	1256
				11	11	0004E	BRB	3\$	:	
		0A	01	A3	E8	00050	1\$:	LIB\$GL_CTLMSK+1, 2\$	:	1257
05	01	A3		02	E0	00054	BBS	#2, LIB\$GL_CTLMSK+1, 2\$	:	1258
03	01	A3		04	E1	00059	BBC	#4, LIB\$GL_CTLMSK+1, 3\$	:	1259
		6E		02	D0	0005E	2\$:	MOVL	:	1260
				50	DD	00061	3\$:	PUSHL	:	1265
				5B	DD	00063		R0	:	
			08	AE	9F	00065	PUSHL	R11	:	
				56	DD	00068	PUSHAB	FUNC	:	
	00000000G	00		04	FB	0006A	PUSHL	R6	:	
		11		50	E8	00071	CALLS	#4, LBR\$INI_CONTROL	:	
				50	DD	00074	BLBS	STATUS, 4\$	:	
7E		65		10	C1	00076	PUSHL	STATUS	:	
				01	DD	0007A	ADDL3	#16, LIB\$GL_LIBFDB, -(SP)	:	
			00000000G	8F	DD	0007C	PUSHL	#1	:	
		64		04	FB	00082	PUSHL	#LIB\$ INITERR	:	
				63	95	00085	CALLS	#4, LIB\$SIGNAL	:	
				10	18	00087	TSTB	LIB\$GL_CTLMSK	:	1266
				56	DD	00089	BGEQ	6\$	:	
				65	DD	0008B	PUSHL	R6	:	1268
	0000V	CF		02	FB	0008D	PUSHL	LIB\$GL_LIBFDB	:	
		03		50	E9	00092	CALLS	#2, LIB_CREATE_LIB	:	
			00BC	31	00095	BLBC	STATUS, 5\$		:	
				04	00098	BRW	16\$		:	
7E		65		10	C1	00099	5\$:	RET	:	
				56	DD	0009D	6\$:	ADDL3	:	1272
	00000000G	00		02	FB	0009F	PUSHL	R6	:	
		52		50	D0	000A6	CALLS	#2, LBR\$OPEN	:	
00000000G		8F		52	D1	000A9	MOVL	R0, STATUS	:	
				09	13	000B0	CMPL	STATUS, #LBR\$_OLDLIBRARY	:	1273
00000000G		8F		52	D1	000B2	BEQL	7\$	:	
				05	12	000B9	CMPL	STATUS, #LBR\$_OLDMISMCH	:	1274
	02	A3	80	8F	88	000BB	BNEQ	8\$	:	
00000000G		8F		52	D1	000C0	7\$:	BISB2	:	1276
				09	13	000C7	8\$:	CMPL	:	1278
00000000G		8F		52	D1	000C9	BEQL	9\$	:	
				36	12	000D0	CMPL	STATUS, #LBR\$_OLDMISMCH	:	1279
		01		6B	D1	000D2	BNEQ	11\$	:	
				0D	12	000D5	CMPL	LIB\$GL_TYPE, #1	:	1281
		67		00	FB	000D7	BNEQ	10\$	:	
		05		50	D1	000DA	CALLS	#0, LBR\$RET_RMSSTV	:	1282
				05	12	000DD	CMPL	R0, #5	:	
		63		20	88	000DF	BNEQ	10\$	:	
				65	11	000E2	BISB2	#32, LIB\$GL_CTLMSK	:	1283
		50		6B	D0	000E4	BRB	15\$	:	
			0000GCF	40	DD	000E7	10\$:	MOVL	:	1287
				10	C1	000EC	PUSHL	LIB\$GL_TYPE, R0	:	
7E		65		00	FB	000F0	ADDL3	LIB\$AL_TYPPNAMES[R0]	:	
		67		40	DD	000F3	CALLS	#16, LIB\$GL_LIBFDB, -(SP)	:	1286
			0000GCF	03	DD	000F8	PUSHL	#0, LBR\$RET_RMSSTV	:	1285
				8F	DD	000FA	PUSHL	LIB\$AL_TYPPNAMES[R0]	:	1286
		64	00000000G	05	FB	00100	PUSHL	#3	:	
		63		20	8A	00103	PUSHL	#LIB\$ DIFTYP	:	
							CALLS	#5, LIB\$SIGNAL	:	
							BICB2	#32, LIB\$GL_CTLMSK	:	1288

50	3E	41	11	00106	BRB	15\$	:	1278
	65	52	E8	00108	BLBS	STATUS, 15\$	:	1291
	58	10	C1	0010B	ADDL3	#16, LIB\$GL_LIBFDB, R0	:	1298
		52	D1	0010F	CMPL	STATUS, R8	:	1294
		24	12	00112	BN:Q	13\$	:	
	02	6E	D1	00114	CMPL	FUNC, #2	:	1297
		0E	12	00117	BNEQ	12\$	:	
		8F	BB	00119	PUSHR	#*M<R0,R8>	:	1298
		01	DD	0011D	PUSHL	#1	:	
		8F	DD	0011F	PUSHL	#LIB\$_BRKNLIB	:	
		1B	11	00125	BRB	14\$	:	
		8F	BB	00127	PUSHR	#*M<R0,R8>	:	1299
		01	DD	0012B	PUSHL	#1	:	
		8F	DD	0012D	PUSHL	#LIB\$_BRKNLIB	:	
	64	04	FB	00133	CALLS	#4, LIB\$SIGNAL	:	
		11	11	00136	BRB	15\$	:	1294
		05	BB	00138	PUSHR	#*M<R0,R2>	:	1302
		01	DD	0013A	PUSHL	#1	:	
		8F	DD	0013C	PUSHL	#8786072	:	
	00000000G	00	04	FB	00142	CALLS	#4, LIB\$STOP	
		67	00	FB	00149	CALLS	#0, LBR\$RET_RMSSTV	1306
		6B	50	DD	0014C	MOVL	R0, LIB\$GL_TYPE	
	0000V	CF	00	FB	0014F	CALLS	#0, GETKEYSIZE	1307
	02	A3	10	88	00154	BISB2	#16, LIB\$GL_CTLMSK+2	1269
			56	DD	00158	PUSHL	R6	1310
		69	01	FB	0015A	CALLS	#1, LBR\$SET_LOCATE	
		02	6E	D1	0015D	CMPL	FUNC, #2	1311
			2A	12	00160	BNEQ	18\$	
		04	AE	9F	00162	PUSHAB	HEADARY	1317
			56	DD	00165	PUSHL	R6	
	00000000G	00	02	FB	00167	CALLS	#2, LBR\$GET_HEADER	
		11	50	E8	0016E	BLBS	STATUS, 17\$	
			50	DD	00171	PUSHL	STATUS	
7E		65	10	C1	00173	ADDL3	#16, LIB\$GL_LIBFDB, -(SP)	
			01	DD	00177	PUSHL	#1	
			8F	DD	00179	PUSHL	#LIB\$_INDEXERR	
		64	04	FB	0017F	CALLS	#4, LIB\$SIGNAL	
			AE	D5	00182	TSTL	HEADARY+116	1318
		78	05	15	00185	BLEQ	18\$	
	03	A3	8F	88	00187	BISB2	#64, LIB\$GL_CTLMSK+3	1319
			02	A3	95	0018C	LIB\$GL_CTLMSK+2	1324
			3C	18	0018F	BGEQ	21\$	
		01	A3	E8	00191	BLBS	LIB\$GL_CTLMSK+1, 19\$	1325
05	01	0A	02	E0	00195	BBS	#2, LIB\$GL_CTLMSK+1, 19\$	1326
2E	01	A3	04	E1	0019A	BBC	#4, LIB\$GL_CTLMSK+1, 21\$	1327
		52	CF	DD	0019F	MOVL	LIB\$GL_OUTFDB, SAVEOUTFDB	1332
	0000G	CF	CF	DD	001A4	MOVL	LIB\$GL_TMPFDB, LIB\$GL_OUTFDB	1333
	03	A3	01	88	001AB	BISB2	#1, LIB\$GL_CTLMSK+3	1334
			02	DD	001AF	PUSHL	#2	1335
	0000G	CF	01	FB	001B1	CALLS	#1, LIB_COMPRS_LIB	
		01	50	E8	001B6	BLBS	STATUS, 20\$	
			04	001B9	RET			
	0000V	CF	00	FB	001BA	CALLS	#0, GETKEYSIZE	1336
	03	A3	01	8A	001BF	BICB2	#1, LIB\$GL_CTLMSK+3	1337
	0000G	CF	52	DD	001C3	MOVL	SAVEOUTFDB, LIB\$GL_OUTFDB	1338
			56	DD	001C8	PUSHL	R6	1339
		69	01	FB	001CA	CALLS	#1, LBR\$SET_LOCATE	

		1C	01	A3	E9	001CD	21\$:	BLBC	LIB\$GL_CTLMSK+1, 22\$	1344
	0000V	CF		00	FB	001D1		CALLS	#0, LIB_DELETE_MODS	1347
				01	DD	001D6		PUSHL	#1	1349
	0000V	CF		01	FB	001D8		CALLS	#1, LIB_PUT_LUH	
		OD		50	E8	001DD		BLBS	STATUS, 22\$	
7E		65		50	DD	001E0		PUSHL	STATUS	
				10	C1	001E2		ADDL3	#16, LIB\$GL_LIBFDB, -(SP)	
				01	DD	001E6		PUSHL	#1	
				5A	DD	001E8		PUSHL	R10	
		64		04	FB	001EA		CALLS	#4, LIB\$SIGNAL	
33	01	A3		02	E1	001ED	22\$:	BBC	#2, LIB\$GL_CTLMSK+1, 24\$	1351
	0000V	CF		00	FB	001F2		CALLS	#0, LIB_INSERT_MODS	1354
				02	DD	001F7		PUSHL	#2	1356
	0000V	CF		01	FB	001F9		CALLS	#1, LIB_PUT_LUH	
		OD		50	E8	001FE		BLBS	STATUS, 23\$	
				50	DD	00201		PUSHL	STATUS	
7E		65		10	C1	00203		ADDL3	#16, LIB\$GL_LIBFDB, -(SP)	
				01	DD	00207		PUSHL	#1	
				5A	DD	00209		PUSHL	R10	
		64		04	FB	0020B		CALLS	#4, LIB\$SIGNAL	
	0000V	CF		03	DD	0020E	23\$:	PUSHL	#3	1358
		OD		01	FB	00210		CALLS	#1, LIB_PUT_LUH	
				50	E8	00215		BLBS	STATUS, 24\$	
				50	DD	00218		PUSHL	STATUS	
7E		65		10	C1	0021A		ADDL3	#16, LIB\$GL_LIBFDB, -(SP)	
				01	DD	0021E		PUSHL	#1	
				5A	DD	00220		PUSHL	R10	
		64		04	FB	00222		CALLS	#4, LIB\$SIGNAL	
05	01	A3		04	E1	00225	24\$:	BBC	#4, LIB\$GL_CTLMSK+1, 25\$	1360
	0000V	CF		00	FB	0022A		CALLS	#0, LIB_REMOVE_SYMS	1361
05		63		06	E0	0022F	25\$:	BBS	#6, LIB\$GL_CTLMSK, 26\$	1363
11	04	A3		03	E1	00233		BBC	#3, LIB\$GL_CTLMSK+4, 27\$	1364
				01	DD	00238	26\$:	PUSHL	#1	1367
	0000G	CF		01	FB	0023A		CALLS	#1, LIB_COMPRS_LIB	
				56	DD	0023F		PUSHL	R6	1368
		69		01	FB	00241		CALLS	#1, LBR\$SET LOCATE	
	0000V	CF		00	FB	00244		CALLS	#0, GETKEYSIZE	1369
05	01	A3		01	E1	00249	27\$:	BBC	#1, LIB\$GL_CTLMSK+1, 28\$	1372
	0000G	CF		00	FB	0024E		CALLS	#0, LIB_EXTRCT_MODS	1373
05	03	A3		04	E1	00253	28\$:	BBC	#4, LIB\$GL_CTLMSK+3, 29\$	1374
	0000G	CF		00	FB	00258		CALLS	#0, LIB_CROSS_OBJ	1375
05	01	A3		03	E1	0025D	29\$:	BBC	#3, LIB\$GL_CTLMSK+1, 30\$	1376
	0000G	CF		00	FB	00262		CALLS	#0, LIB_LIST_LIB	1377
29	02	A3		04	E1	00267	30\$:	BBC	#4, LIB\$GL_CTLMSK+2, 32\$	1378
				56	DD	0026C		PUSHL	R6	1382
	00000000G	00		01	FB	0026E		CALLS	#1, LBR\$CLOSE	
		52		50	DD	00275		MOVL	R0, STATUS	
		16		52	E8	00278		BLBS	STATUS, 31\$	
		67		00	FB	0027B		CALLS	#0, LBR\$RET_RMSSTV	
				50	DD	0027E		PUSHL	R0	
				52	DD	00280		PUSHL	STATUS	
7E		65		10	C1	00282		ADDL3	#16, LIB\$GL_LIBFDB, -(SP)	
				01	DD	00286		PUSHL	#1	
			00861050	8F	DD	00288		PUSHL	#8786000	
		64		05	FB	0028E		CALLS	#5, LIB\$SIGNAL	
	02	A3		10	8A	00291	31\$:	BICB2	#16, LIB\$GL_CTLMSK+2	1383
		50		01	DD	00295	32\$:	MOVL	#1, R0	1385

LIB PROCMD
V04=000

L 5
16-Sep-1984 02:03:14
14-Sep-1984 12:38:09

VAX-11 Bliss-32 V4.0-742
[LIBRAR.SRC]PROCMD.B32;1

Page 12
(3)

04 00298

RET

; 1386

; Routine Size: 665 bytes, Routine Base: \$CODE\$ + 0000

```

: 330      1387 1 ROUTINE getkeysize =
: 331      1388 2 BEGIN
: 332      1389 2 |
: 333      1390 2 | This routine gets the maximum keysize from the library header
: 334      1391 2 |
: 335      1392 2 BIND
: 336      1393 2     libheader = .lbr$gl_control [lbr$l_hdrptr] : BBLOCK; !Name the header
: 337      1394 2
: 338      1395 2 IF .lib$gl_ctlmsk [lib$v_oldlib]           !If old format library
: 339      1396 2     THEN lib$gl_keysize = lib$c_shortsym
: 340      1397 2     ELSE BEGIN
: 341      1398 2         BIND
: 342      1399 2         indexdesc = libheader + lhd$c_idxdesc : BBLOCK; !First index descriptor
: 343      1400 2         lib$gl_keysize = .indexdesc [idd$w_keylen] - 1;
: 344      1401 2     END;
: 345      1402 2 RETURN true
: 346      1403 2
: 347      1404 1 END;

```

!Of getkeysize

```

                                0004 00000 GETKEYSIZE:
                                .WORD      Save R2
                                52          0000G CF 9E 00002      MOVAB      LIB$GL_KEYSIZE, R2
                                50 00000000G 00 D0 00007      MOVL       LBR$GL_CONTROL, R0
                                0000G      CF 95 0000E      TSTB       LIB$GL_CTLMSK+2
                                05 18 00012      BGEQ       1$
                                62          0F D0 00014      MOVL       #15, LIB$GL_KEYSIZE
                                0F 11 00017      BRB        2$
                                50          0A A0 00000C4 8F C1 00019 1$: ADDL3      #196, 10(R0), R0
                                62          02 A0 3C 00022      MOVZWL     2(R0), LIB$GL_KEYSIZE
                                62 D7 00026      DECL      LIB$GL_KEYSIZE
                                50          01 D0 00028 2$: MOVL       #1, R0
                                04 0002B      RET

```

; Routine Size: 44 bytes, Routine Base: \$CODE\$ + 0299

; 348 1405 1

```

: 1387
: 1393
: 1395
: 1396
: 1399
: 1400
: 1402
: 1404

```

```
1406 1 GLOBAL ROUTINE lib_create_lib (fdbadr, control, map_desc) =
1407 2 BEGIN
1408 2
1409 2 | This routine calls the library access procedures
1410 2 | to create the library.
1411 2
1412 2 BUILTIN
1413 2 NULLPARAMETER;
1414 2 MAP
1415 2 fdbadr : REF BBLOCK;
1416 2 BIND
1417 2 namblk = fdbadr [fdb$t_nam] : BBLOCK,
1418 2 fildesc = fdbadr [fdb$_namdesc] : BBLOCK;
1419 2 LOCAL
1420 2 create_opts : BBLOCK [cre$c_length],
1421 2 status;
1422 2
1423 2 CH$FILL (0, cre$c_length, create_opts); !Zero create options array
1424 2 create_opts [cre$_type] = .lib$gl_type; !Set the library type
1425 3 create_opts [cre$_keylen] = (IF .lib$gl_cre8flags [lib$c_cpt_ksz]
1426 3 THEN .lib$gl_allksz
1427 2 ELSE .lib$al_ascbinf [.lib$gl_type]); !Set ASCII/binary flag
1428 2 lib$gl_keysize = .create_opts [cre$_keylen]; !Set max key length of library
1429 2 create_opts [cre$_alloc] = .lib$gl_allblks;
1430 2 create_opts [cre$_idxmax] = .lib$al_numidx [.lib$gl_type]; !Set number of indices
1431 2 create_opts [cre$_uhdmax] = .lib$al_hdrlen [.lib$gl_type]; !Set length of extra header info
1432 2 IF .lib$gl_type EQ[ lbr$c_typ_obj
1433 2 OR .lib$gl_type EQ[ lbr$c_typ_shstb
1434 2 THEN create_opts [cre$_entall] = .lib$gl_allgbls + .lib$gl_allmods
1435 2 ELSE create_opts [cre$_entall] = .lib$gl_allmods;
1436 2 create_opts [cre$_luhmax] = .lib$gl_allhis;
1437 2 create_opts [cre$_vertyp] = .lib$gl_allver; ! library version format
1438 2 create_opts [cre$_idxopt] = .lib$al_idxopt [.lib$gl_type]; ! index options
1439 2
1440 2 IF NOT NULLPARAMETER(3)
1441 2 THEN
1442 2 status = lbr$open (.control, fildesc, create_opts,
1443 2 fdbadr [fdb$_defext], .namblk[nam$_r[f]], 0, 0, .map_desc)
1444 2 ELSE
1445 2 status = lbr$open (.control, fildesc, create_opts,
1446 2 fdbadr [fdb$_defext], .namblk[nam$_r[f]]);
1447 2
1448 2 | Get the string descriptor for the filename correct
1449 2
1450 2 IF .namblk [nam$b_rsl] NEQ 0
1451 2 THEN BEGIN
1452 3 fildesc [dsc$_w_length] = .namblk [nam$b_rsl];
1453 3 fildesc [dsc$_a_pointer] = .namblk [nam$_rsl];
1454 3 END
1455 2 ELSE IF .namblk [nam$b_esl] NEQ 0
1456 2 THEN BEGIN
1457 3 fildesc [dsc$_w_length] = .namblk [nam$b_esl];
1458 3 fildesc [dsc$_a_pointer] = .namblk [nam$_esa];
1459 3 END;
1460 2 IF NOT .status
1461 2 THEN SIGNAL_STOP (lib$_openout, 1, fildesc, .status, lbr$ret_rmsstv());
1462 2 RETURN true
```

0050	8F	56	04	58	00000000G	00	01FC	00000	.ENTRY	LIB CREATE LIB, Save R2,R3,R4,R5,R6,R7,R8	1406
		57	04	5E	B0	AE	9E	00002	MOVAB	LBR\$OPEN, R8	
		00		AC	00000040	8F	C1	00009	MOVAB	-80(SP), SP	
				AC		10	C1	0000D	ADDL3	#64, FDBADR, R6	1417
				6E		00	2C	0001B	ADDL3	#16, FDBADR, R7	1418
						6E		00022	MOVCS	#0, (SP), #0, #80, CREATE_OPTS	1423
				50	0000G	CF	D0	00023	MOVL	LIB\$GL_TYPE, R0	1424
		07	0000G	6E		50	D0	00028	MOVL	R0, CREATE_OPTS	
				CF		03	E1	0002B	BBC	#3, LIB\$GL-CRE8FLAGS, 1\$	1425
				51	0000G	CF	D0	00031	MOVL	LIB\$GL_ALLRSZ, R1	1426
						06	11	00036	BRB	2\$	
				51	0000GCF	40	D0	00038	MOVL	LIB\$AL_ASCBINFER0], R1	1427
				AE		51	D0	0003E	MOVL	R1, CREATE_OPTS+4	1425
			04	CF	04	AE	D0	00042	MOVL	CREATE_OPTS+4, LIB\$GL_KEYSZ	1428
			08	AE	0000G	CF	D0	00048	MOVL	LIB\$GL_ALLBLKS, CREATE_OPTS+8	1429
			0C	AE	0000GCF	40	D0	0004E	MOVL	LIB\$AL_NUMIDX[R0], CREATE_OPTS+12	1430
			10	AE	0000GCF	40	D0	00055	MOVL	LIB\$AL_HDRLEN[R0], CREATE_OPTS+16	1431
				01		50	D1	0005C	CMPL	R0, #1	1432
						05	13	0005F	BEQL	3\$	
				05		50	D1	00061	CMPL	R0, #5	1433
						0B	12	00064	BNEQ	4\$	
14	AE		0000G	CF	0000G	CF	C1	00066	ADDL3	LIB\$GL_ALLMODS, LIB\$GL_ALLGBLS, -	1434
						06	11	0006F	BRB	CREATE_OPTS+20	
		14	AE	0000G	CF	D0	00071	4\$:	MOVL	LIB\$GL_ALLMODS, CREATE_OPTS+20	1435
		18	AE	0000G	CF	D0	00077	5\$:	MOVL	LIB\$GL_ALLHIS, CREATE_OPTS+24	1436
		1C	AE	0000G	CF	D0	0007D		MOVL	LIB\$GL_ALLVER, CREATE_OPTS+28	1437
		20	AE	0000GCF	40	D0	00083		MOVL	LIB\$AL_IDXOPT[R0], CREATE_OPTS+32	1438
50		04	AC		08	C1	0008A		ADDL3	#8, FDBADR, R0	1443
			03		6C	91	0008F		CMPL	(AP), #3	1440
					1C	1F	00092		BLSSU	6\$	
				0C	AC	D5	00094		TSTL	12(AP)	
					17	13	00097		BEQL	6\$	
				0C	AC	DD	00099		PUSHL	MAP DESC	1443
					7E	7C	0009C		CLRQ	-(SP)	
				10	A6	DD	0009E		PUSHL	16(R6)	
					50	DD	000A1		PUSHL	R0	
				14	AE	9F	000A3		PUSHAB	CREATE_OPTS	1442
					57	DD	000A6		PUSHL	R7	1443
				08	AC	DD	000A8		PUSHL	CONTROL	
			68		08	FB	000AB		CALLS	#8, LBR\$OPEN	
					10	11	000AE		BRB	7\$	
					10	A6	DD	000B0	PUSHL	16(R6)	1446
					50	DD	000B3		PUSHL	R0	
				08	AE	9F	000B5		PUSHAB	CREATE_OPTS	1445
					57	DD	000B8		PUSHL	R7	1446
				08	AC	DD	000BA		PUSHL	CONTROL	
			68		05	FB	000BD		CALLS	#5, LBR\$OPEN	
			52		50	D0	000C0	7\$:	MOVL	R0, STATUS	
					03	A6	95	000C3	TSTB	3(R6)	1450

LIB_PROCMD
V04=000

C 6
16-Sep-1984 02:03:14
14-Sep-1984 12:38:09

VAX-11 Bliss-32 V4.0-742
[LIBRAR.SRC]PROCMD.B32;1

Page 16
(5)

04	67	03	0B	13	000C6	BEQL	8\$	:	1452
	A7	04	A6	9B	000C8	MOVZBW	3(R6), (R7)	:	1453
			A6	D0	000CC	MOVL	4(R6), 4(R7)	:	1450
			0E	11	000D1	BRB	9\$	:	1455
		0B	A6	95	000D3	TSTB	11(R6)	:	1457
			09	13	000D6	BEQL	9\$	:	1458
04	67	0B	A6	9B	000D8	MOVZBW	11(R6), (R7)	:	1460
	A7	0C	A6	D0	000DC	MOVL	12(R6), 4(R7)	:	1461
	1C		52	E8	000E1	BLBS	STATUS, 10\$	:	
00000000G	00		00	FB	000E4	CALLS	#0, LBR\$RET_RMSSTV	:	
			50	DD	000EB	PUSHL	R0	:	
			52	DD	000ED	PUSHL	STATUS	:	
			57	DD	000EF	PUSHL	R7	:	
			01	DD	000F1	PUSHL	#1	:	
		008610A4	8F	DD	000F3	PUSHL	#8786084	:	
00000000G	00		05	FB	000F9	CALLS	#5, LIB\$STOP	:	1462
	50		01	D0	00100	MOVL	#1, R0	:	1463
			04	00	103	RET		:	

; Routine Size: 260 bytes, Routine Base: \$CODE\$ + 02C5

```
409 1464 1 ROUTINE lib_delete_mods =
410 1465 2 BEGIN
411 1466 3
412 1467 2 This routine deletes modules from the library
413 1468 2
414 1469 2 LOCAL
415 1470 2     modnb : REF BBLOCK,
416 1471 2     status,
417 1472 2     modesc : BBLOCK [dsc$c_s_bln];
418 1473 2 BUILTIN
419 1474 2     REMQUE;
420 1475 2
421 1476 2 WHILE NOT REMQUE (.lib$gl_delmodl, modnb)           !get next module to delete
422 1477 3 DO BEGIN
423 1478 3     modesc [dsc$w_length] = .modnb [lnb$b_namlng];      !Set up name descriptor
424 1479 3     modesc [dsc$a_pointer] = modnb [lnb$t_name];
425 1480 3
426 1481 3     Check for wild cards in name.
427 1482 3
428 1483 3     IF NOT CH$FAIL (CH$FIND_CH (.modnb [lnb$b_namlng], modnb [lnb$t_name],
429 1484 3         %ASCII '%'))
430 1485 3     OR NOT CH$FAIL (CH$FIND_CH (.modnb [lnb$b_namlng], modnb [lnb$t_name],
431 1486 3         %ASCII '*'))
432 1487 4     THEN BEGIN
433 1488 5         IF (status = lbr$get_index (lib$gl_libctl, lib$gl_modnamix,
434 1489 4             add_de[mod, modesc]) EQL lbr$nomtchfou
435 1490 4         THEN SIGNAL (lib$nomtchfou, 1, modesc)
436 1491 4         ELSE IF NOT .status
437 1492 4             THEN SIGNAL (lib$indexerr, 1, lib$gl_libfdb [fdb$l_namdesc],
438 1493 4                 .status, [lbr$ret_rmsstv()]);
439 1494 4     END
440 1495 3     ELSE (.lib$al_deldspat [.lib$gl_type]) (modesc);      !Delete the module
441 1496 3     lib_free_mem (lnb$c_fixedsize+.modnb [lnb$b_namlng], .modnb); !Deallocate the block
442 1497 2 END;
443 1498 2 RETURN true
444 1499 1 END;
```

```
001C 00000 LIB_DELETE_MODS:
      .WORD Save R2,R3,R4
54 00000000G 00 9E 00002 MOVAB LIB$SIGNAL, R4
5E 08 C2 00009 SUBL2 #8, SP
52 0000G DF 0F 0000C 1$: REMQUE @LIB$GL_DELMODL, MODNB
      03 1C 00011 BVC 2$
      009D 31 00013 BRW 9$
04 6E 09 A2 9B 00016 2$: MOVZBW 9(MODNB), MODESC
      AE 0A A2 9E 0001A MOVAB 10(MODNB), MODESC+4
      50 09 A2 9A 0001F MOVZBL 9(MODNB), R0
      50 25 3A 00023 LOCC #37, R0, 10(MODNB)
      02 12 00028 BNEQ 3$
      51 D4 0002A CLRL R1
      51 D5 0002C 3$: TSTL R1
      11 12 0002E BNEQ 5$
      50 09 A2 9A 00030 MOVZBL 9(MODNB), R0
```

1464
1476
1478
1479
1483
1484
1485

0A	A2	50	2A	3A	00034	LOCC	#42, R0, 10(MODNB)	:	
			02	12	00039	BNEQ	4\$	:	
			51	D4	0003B	CLRL	R1	:	
			51	D5	0003D	TSTL	R1	:	1486
			51	13	0003F	BEQL	7\$	:	
			5E	DD	00041	PUSHL	SP	:	1488
		0000V	CF	9F	00043	PUSHAB	ADD_DEL_MOD	:	
		0000G	CF	9F	00047	PUSHAB	LIB\$GL_MODNAMIX	:	
		0000G	CF	9F	0004B	PUSHAB	LIB\$GL_LIBCTL	:	
	00000000G	00	04	FB	0004F	CALLS	#4, LBR\$GET_INDEX	:	
		53	50	D0	00056	MOVL	R0, STATUS	:	
	00000000G	8F	53	D1	00059	CMPL	STATUS, #LBR\$_NOMTCHFOU	:	1489
			0F	12	00060	BNEQ	6\$	:	
			5E	DD	00062	PUSHL	SP	:	1490
			01	DD	00064	PUSHL	#1	:	
		00000000G	8F	DD	00066	PUSHL	#LIB\$_NOMTCHFOU	:	
	64		03	FB	0006C	CALLS	#3, LIB\$SIGNAL	:	
			31	11	0006F	BRB	8\$	:	
		2E	53	E8	00071	BLBS	STATUS, 8\$	:	1491
	00000000G	00	00	FB	00074	CALLS	#0, LBR\$RET_RMSSTV	:	1493
			50	DD	0007B	PUSHL	R0	:	
			53	DD	0007D	PUSHL	STATUS	:	
7E	0000G	CF	10	C1	0007F	ADDL3	#16, LIB\$GL_LIBFDB, -(SP)	:	1492
			01	DD	00085	PUSHL	#1	:	
		00000000G	8F	DD	00087	PUSHL	#LIB\$_INDEXERR	:	
	64		05	FB	0008D	CALLS	#5, LIB\$SIGNAL	:	
			10	11	00090	BRB	8\$	:	1483
	50	0000G	CF	D0	00092	MOVL	LIB\$GL_TYPE, R0	:	1495
	50	0000G	CF	D0	00097	MOVL	LIB\$AL_DELDSPAT[R0], R0	:	
			5E	DD	0009D	PUSHL	SP	:	
	60		01	FB	0009F	CALLS	#1, (R0)	:	
			52	DD	000A2	PUSHL	MODNB	:	1496
	7E	09	A2	9A	000A4	MOVZBL	9(MODNB), -(SP)	:	
	6E		0A	C0	000A8	ADDL2	#10, (SP)	:	
	0000G	CF	02	FB	000AB	CALLS	#2, LIB_FREE_MEM	:	
			FF	59	31	BRW	1\$	:	1476
	50		01	D0	000B3	MOVL	#1, R0	:	1498
			04	000B6	RET			:	1499

; Routine Size: 183 bytes, Routine Base: \$CODE\$ + 03C9

```
445 1500 1 ROUTINE add_del_mod (modesc) =
446 1501 2 BEGIN
447 1502 2
448 1503 2 This routine is called by lbr$get_index when looking for module names to delete
449 1504 2 The name is inserted into the list of modules to delete
450 1505 2
451 1506 2 MAP
452 1507 2 modesc : REF BBLOCK;
453 1508 2
454 1509 2 LOCAL
455 1510 2 namblk : REF BBLOCK;
456 1511 2
457 1512 2 BUILTIN
458 1513 2 INSQUE;
459 1514 2
```

```
: 460      1515 2 perform (lib get_mem (lnb$fixedsize + .modesc [dsc$w_length], namblk));
: 461      1516 2 INSQUE (.namblk, .lib$gl_delmodl [1]);
: 462      1517 2 namblk [lnb$b_naming] = .modesc [dsc$w_length];
: 463      1518 2 CH$MOVE (.modesc [dsc$w_length], .modesc [dsc$a_pointer], namblk [lnb$t_name]);
: 464      1519 2 RETURN true
: 465      1520 1 END;
```

!Of add_del_mod

				003C 00000 ADD_DEL_MOD:			
		5E	04	C2 00002	.WORD	Save R2,R3,R4,R5	: 1500
			5E	DD 00005	SUBL2	#4, SP	
		7E	04	BC 3C 00007	PUSHL	SP	: 1515
		6E		0A C0 0000B	MOVZWL	@MODESC, -(SP)	
	0000G	CF		02 FB 0000E	ADDL2	#10, (SP)	
		1A		50 E9 00013	CALLS	#2, LIB_GET_MEM	
					BLBC	STATUS, 1\$	
	0000G	DF	00	BE 0E 00016	INSQUE	@NAMBLK, @LIB\$GL_DELMODL+4	: 1516
		51		6E D0 0001C	MOVL	NAMBLK, R1	: 1517
		50	04	AC D0 0001F	MOVL	MODESC, R0	
		09		60 90 00023	MOVB	(R0), 9(R1)	
0A	A1	04		60 28 00027	MOVC3	(R0), @4(R0), 10(R1)	: 1518
		50		01 D0 0002D	MOVL	#1, R0	: 1519
				04 00030 1\$:	RET		: 1520

; Routine Size: 49 bytes, Routine Base: \$CODE\$ + 0480

```
467 1521 1 ROUTINE lib_remove_syms =
468 1522 2 BEGIN
469 1523 2
470 1524 2 This routine removes global symbols from object libraries
471 1525 2
472 1526 2 LOCAL
473 1527 2     status,
474 1528 2     symnb : REF BBLOCK,
475 1529 2     symdesc : BBLOCK [dsc$sc_s_bln];
476 1530 2
477 1531 2 BUILTIN
478 1532 2     REMQUE;
479 1533 2
480 1534 2 IF .lib$gl_type NEQ lbr$sc_typ_obj
481 1535 2     AND .lib$gl_type NEQ [lbr$sc_typ_shstb]
482 1536 2 THEN BEGIN
483 1537 2     SIGNAL (lib$_notobjlib, 1, lib$gl_libfdb [fdb$l_namdesc]);
484 1538 2     RETURN lib$_notobjlib;
485 1539 2 END;
486 1540 2
P 487 1541 2 perform (lbr$set_index (lib$gl_libctl, lib$gl_objgsdix), !set index to delete global syms
488 1542 2     lib$_indexerr, 1, lib$gl_libfdb [fdb$l_namdesc]);
489 1543 2 WHILE NOT REMQUE (.lib$gl_objsyml, symnb) !get next module to delete
490 1544 2 DO BEGIN
491 1545 3     symdesc [dsc$w_length] = .symnb [lnb$b_namlng]; !Set up name descriptor
492 1546 3     symdesc [dsc$a_pointer] = symnb [lnb$t_name];
493 1547 3     IF NOT CH$FAIL (CH$FIND_CH (.symnb [lnb$b_namlng], symnb [lnb$t_name],
494 1548 3         %ASCII '%'))
495 1549 3         OR NOT CH$FAIL (CH$FIND_CH (.symnb [lnb$b_namlng], symnb [lnb$t_name],
496 1550 3             %ASCII '+'))
497 1551 4     THEN BEGIN
498 1552 5         IF (status = lbr$get_index (lib$gl_libctl, lib$gl_objgsdix,
499 1553 4             add_rem_sym, symdesc)) EQL [lbr$_nomtchfou]
500 1554 4             THEN SIGNAL (lib$_nomtchfou, 1, symdesc)
501 1555 4             ELSE IF NOT .status
502 1556 4                 THEN SIGNAL (lib$_indexerr, 1, lib$gl_libfdb [fdb$l_namdesc],
503 1557 4                     .status, lbr$_ret_rmsstv());
504 1558 4     END
505 1559 4     ELSE BEGIN
506 1560 5         IF (status = lbr$delete_key (lib$gl_libctl, symdesc)) !Delete the symbol
507 1561 4             THEN lib_log_op (lib$_removed, symdesc, .lib$gl_libfdb) !log if logging
508 1562 4             ELSE SIGNAL (lib$_delkeyerr, 2, symdesc, lib$gl_libfdb [fdb$l_namdesc], .status);
509 1563 3     END;
510 1564 3     lib_free_mem (lnb$sc_fixedsize+.symnb [lnb$b_namlng], .symnb); !Deallocate the block
511 1565 2     END;
512 1566 2 RETURN true
513 1567 1 END;
```

```
01FC 00000 LIB_REMOVE_SYMS:
58      0000G CF 9E 00002      .WORD Save R2,R3,R4,R5,R6,R7,R8
57 00000000G 8F D0 00007      MOVAB LIB$GL_LIBCTL, R8
56 00000000G 8F D0 0000E      MOVL #LIB$_INDEXERR, R7
                          MOVL #LIB$_NOTOBJLIB, R6
```

```
; 1521
:
:
```

	55	0000G	CF	9E	00015	MOVAB	LIB\$GL_LIBFDB, R5		
	54	00000000G	00	9E	0001A	MOVAB	LIB\$SIGNAL, R4		
	5E		08	C2	00021	SUBL2	#8, SP		
	01	0000G	CF	D1	00024	CMPL	LIB\$GL_TYPE, #1	1534	
			16	13	00029	BEQL	1\$		
	05	0000G	CF	D1	0002B	CMPL	LIB\$GL_TYPE, #5	1535	
			0F	13	00030	BEQL	1\$		
7E	65		10	C1	00032	ADDL3	#16, LIB\$GL_LIBFDB, -(SP)	1537	
			01	DD	00036	PUSHL	#1		
			56	DD	00038	PUSHL	R6		
	64		03	FB	0003A	CALLS	#3, LIB\$SIGNAL		
	50		56	D0	0003D	MOVL	R6, R0	1538	
				04	00040	RET			
		0000G	CF	9F	00041	PUSHAB	LIB\$GL_OBJGSDIX	1542	
			58	DD	00045	PUSHL	R8		
	00000000G	00	02	FB	00047	CALLS	#2, LBR\$SET_INDEX		
		0D	50	E8	0004E	BLBS	STATUS, 2\$		
			50	DD	00051	PUSHL	STATUS		
7E	65		10	C1	00053	ADDL3	#16, LIB\$GL_LIBFDB, -(SP)		
			01	DD	00057	PUSHL	#1		
			57	DD	00059	PUSHL	R7		
	64		04	FB	0005B	CALLS	#4, LIB\$SIGNAL		
	52	0000G	DF	0F	0005E	REMQUE	@LIB\$GL_OBJSYRML, SYMNB	1543	
			03	1C	00063	BVC	3\$		
		00B9	31	00065	BRW	12\$			
	6E	09	A2	9B	00068	MOVZBW	9(SYMNB), SYMDESC	1545	
	AE	0A	A2	9E	0006C	MOVAB	10(SYMNB), SYMDESC+4	1546	
	50	09	A2	9A	00071	MOVZBL	9(SYMNB), R0	1547	
0A	A2		25	3A	00075	LOCC	#37, R0, 10(SYMNB)		
			02	12	0007A	BNEQ	4\$		
			51	D4	0007C	CLRL	R1		
			51	D5	0007E	TSTL	R1	1548	
			11	12	00080	BNEQ	6\$		
	50	09	A2	9A	00082	MOVZBL	9(SYMNB), R0	1549	
0A	A2		2A	3A	00086	LOCC	#42, R0, 10(SYMNB)		
	50		02	12	0008B	BNEQ	5\$		
			51	D4	0008D	CLRL	R1		
			51	D5	0008F	TSTL	R1	1550	
			46	13	00091	BEQL	8\$		
			5E	DD	00093	PUSHL	SP	1552	
		0000V	CF	9F	00095	PUSHAB	ADD REM_SYM		
		0000G	CF	9F	00099	PUSHAB	LIB\$GL_OBJGSDIX		
			58	DD	0009D	PUSHL	R8		
	00000000G	00	04	FB	0009F	CALLS	#4, LBR\$GET_INDEX		
		53	50	D0	000A6	MOVL	R0, STATUS		
	00000000G	8F	53	D1	000A9	CMPL	STATUS, #LBR\$_NOMTCHFOU	1553	
			0F	12	000B0	BNEQ	7\$		
			5E	DD	000B2	PUSHL	SP	1554	
			01	DD	000B4	PUSHL	#1		
		00000000G	8F	DD	000B6	PUSHL	#LIB\$_NOMTCHFOU		
	64		03	FB	000BC	CALLS	#3, LIB\$SIGNAL		
			4F	11	000BF	BRB	11\$		
	4C		53	E8	000C1	BLBS	STATUS, 11\$	1555	
	00000000G	00	00	FB	000C4	CALLS	#0, LBR\$RET_RMSSTV	1557	
			50	DD	000CB	PUSHL	R0		
			53	DD	000CD	PUSHL	STATUS		
7E	65		10	C1	000CF	ADDL3	#16, LIB\$GL_LIBFDB, -(SP)	1556	

			01	DD	000D3	PUSHL	#1		
			57	DD	000D5	PUSHL	R7		
			34	11	000D7	BRB	10\$		
		4100	8F	BB	000D9	8\$:	PUSHR	#^M<R8, SP>	1560
00000000G	00		02	FB	000DD		CALLS	#2, LBR\$DELETE_KEY	
	53		50	D0	000E4		MOVL	R0, STATUS	
	12		53	E9	000E7		BLBC	STATUS, 9\$	
			65	DD	000EA		PUSHL	LIB\$GL_LIBFDB	1561
		04	AE	9F	000EC		PUSHAB	SYMDESC	
		00000000G	8F	DD	000EF		PUSHL	#LIB\$ REMOVED	
0000V	CF		03	FB	000F5		CALLS	#3, LIB_LOG_OP	
			14	11	000FA		BRB	11\$	
			53	DD	000FC	9\$:	PUSHL	STATUS	1562
7E	65		10	C1	000FE		ADDL3	#16, LIB\$GL_LIBFDB, -(SP)	
		08	AE	9F	00102		PUSHAB	SYMDESC	
			02	DD	00105		PUSHL	#2	
		00000000G	8F	DD	00107		PUSHL	#LIB\$ DELKEYERR	
	64		05	FB	0010D	10\$:	CALLS	#5, LIB\$SIGNAL	
			52	DD	00110	11\$:	PUSHL	SYMBOL	1564
	7E	09	A2	9A	00112		MOVZBL	9(SYMBOL), -(SP)	
	6E		0A	C0	00116		ADDL2	#10, (SP)	
0000G	CF		02	FB	00119		CALLS	#2, LIB_FREE_MEM	
			FF3D	31	0011E		BRW	2\$	1543
	50		01	D0	00121	12\$:	MOVL	#1, R0	1566
			04	00124			RET		1567

; Routine Size: 293 bytes, Routine Base: \$CODE\$ + 04B1

```
514 1568 1 !
515 1569 1 ROUTINE add_rem_sym (modesc) =
516 1570 2 BEGIN
517 1571 2
518 1572 2 ! This routine is called by lbr$get_index when looking for symbols to delete.
519 1573 2 ! The name is inserted into the list of symbols to delete
520 1574 2
521 1575 2 MAP
522 1576 2 modesc : REF BBLOCK;
523 1577 2
524 1578 2 LOCAL
525 1579 2 namblk : REF BBLOCK;
526 1580 2
527 1581 2 BUILTIN
528 1582 2 INSQUE;
529 1583 2
530 1584 2 perform (lib_get_mem (lnb$fixedsize + .modesc [dsc$w_length], namblk));
531 1585 2 INSQUE (.namblk, .lib$gl_objsyml [1]);
532 1586 2 namblk [lnb$b_namlng] = .modesc [dsc$w_length];
533 1587 2 CH$MOVE (.modesc [dsc$w_length], .modesc [dsc$a_pointer], namblk [lnb$t_name]);
534 1588 2 RETURN true
535 1589 1 END;
```

!Of add_rem_sym

003C 00000 ADD_REM_SYM:

LIB_PROCMD
V04=000

J 6
16-Sep-1984 02:03:14
14-Sep-1984 12:38:09

VAX-11 Bliss-32 V4.0-742
[LIBRAR.SRC]PROCMD.B32;1

Page 23
(7)

		5E		04	C2	00002	.WORD	Save R2,R3,R4,R5	:	1569
				5E	DD	00005	SUBL2	#4, SP	:	
		7E		04	BC	00007	PUSHL	SP	:	1584
		6E			0A	0000B	MOVZWL	@MODESC, -(SP)	:	
	0000G	CF			02	FB	ADDL2	#10, (SP)	:	
		1A			50	E9	CALLS	#2, LIB_GET_MEM	:	
	0000G	DF		00	BE	0E	BLBC	STATUS, -1\$	:	
		51			6E	D0	INSQUE	@NAMBLK, @LIB\$GL_OBJSYRML+4	:	1585
		50		04	AC	D0	MOVL	NAMBLK, R1	:	1586
		A1			60	90	MOVL	MODESC, R0	:	
0A	A1	09			60	28	MOVB	(R0), 9(R1)	:	
		04			01	D0	MOVC3	(R0), @4(R0), 10(R1)	:	1587
		50			04	00030	MOVL	#1, R0	:	1588
							RET		:	1589

; Routine Size: 49 bytes, Routine Base: \$CODE\$ + 05D6

```
: 537      1590 1 ROUTINE lib_insert_mods =  
: 538      1591 2 BEGIN  
: 539      1592 2 |  
: 540      1593 2 | This routine inserts modules into the library  
: 541      1594 2 |  
: 542      1595 2 WHILE next_input_file ()  
: 543      1596 2 DO (.lib$at_inpdspat [.lib$gl_type]) ();      !Insert each into the library  
: 544      1597 2 RETURN true  
: 545      1598 1 END;
```

```
                                0000 00000 LIB_INSERT_MODS:  
                                .WORD Save nothing  
0000G CF      00 FB 00002 1$: CALLS #0, NEXT_INPUT_FILE      : 1590  
                                50 E9 00007 BLBC R0, 2$      : 1595  
                                0000G CF D0 0000A MOVL LIB$GL_TYPE, R0      : 1596  
                                0000GCF 40 D0 0000F MOVL LIB$AL_INPDSPAT[R0], R0  
                                50      00 FB 00015 CALLS #0, (R0)  
                                60      E8 11 00018 BRB 1$  
                                50      01 D0 0001A 2$: MOVL #1, R0      : 1597  
                                04 0001D RET      : 1598
```

; Routine Size: 30 bytes, Routine Base: \$CODE\$ + 0607

```
: 547      1599 1 GLOBAL ROUTINE lib_log_op (opcode, modesc, libfdb) =
: 548      1600 2 BEGIN
: 549      1601 2
: 550      1602 2 | This routine logs the operation on the console if /LOG was specified.
: 551      1603 2 |
: 552      1604 2 MAP
: 553      1605 2     libfdb : REF BBLOCK,
: 554      1606 2     modesc : REF BBLOCK;
: 555      1607 2
: 556      1608 2
: 557      1609 2 IF .lib$gl_ctlmsk [lib$v_log]                                !Logging?
: 558      1610 2 THEN SIGNAL (.opcode, 2, .modesc, libfdb [fdb$l_namdesc]);
: 559      1611 2 RETURN true
: 560      1612 1 END;
```

14	0000G	CF	0000	00000	.ENTRY	LIB_LOG_OP, Save nothing	: 1599
7E	0C	AC	06	E1 00002	BBC	#6, LIB\$GL_CTLMSK+2, 1\$	: 1609
			10	C1 00008	ADDL3	#16, LIBFDB, -(SP)	: 1610
			08	AC DD 0000D	PUSHL	MODESC	:
			02	DD 00010	PUSHL	#2	:
			04	AC DD 00012	PUSHL	OPCODE	:
00000000G	00		04	FB 00015	CALLS	#4, LIB\$SIGNAL	:
	50		01	D0 0001C	MOVL	#1, R0	: 1611
			04	0001F	RET		: 1612

; Routine Size: 32 bytes, Routine Base: \$CODE\$ + 0625

```

: 562      1613 1 GLOBAL ROUTINE lib_log_upd ( operation, module_desc ) =
: 563      1614 2 BEGIN
: 564      1615 2 |+++
: 565      1616 2 |
: 566      1617 2 |   If a library update history is maintained for the library,
: 567      1618 2 |   build a list of the module names of modules that have been
: 568      1619 2 |   updated in library.
: 569      1620 2 |---
: 570      1621 2 ---
: 571      1622 2 MAP
: 572      1623 2   module_desc : REF BBLOCK [dsc$c_s_bln];
: 573      1624 2 BUILTIN
: 574      1625 2   INSQUE;
: 575      1626 2
: 576      1627 2 IF .lib$gl_ctlmsk [lib$v_recordluh]
: 577      1628 2 THEN
: 578      1629 2   BEGIN
: 579      1630 2     LOCAL
: 580      1631 2       namblk : REF BBLOCK;
: 581      1632 2
: 582      1633 2     perform (lib_get_mem (lnb$c_fixedsize + .module_desc [dsc$w_length], namblk) );
: 583      1634 2     INSQUE (.namblk, .lib$gl_modupdl [1] );
: 584      1635 2     namblk [lnb$b_namng] = .module_desc [dsc$w_length];
: 585      1636 2     CH$MOVE(.module_desc [dsc$w_length], .module_desc [dsc$a_pointer],
: 586      1637 2       namblk [lnb$t_name] );
: 587      1638 2     namblk [lnb$b_flags] = .operation;
: 588      1639 2   END;
: 589      1640 2 RETURN true;
: 590      1641 1 END;      ! global routine lib_log_upd
```

				007C	00000	.ENTRY	LIB_LOG_UPD, Save R2,R3,R4,R5,R6	: 1613
		5E		04	C2	SUBL2	#4, SP	
2D	0000G	CF		06	E1	BBC	#6, LIB\$GL_CTLMSK+3, 1\$	: 1627
		7E	08	5E	DD	PUSHL	SP	: 1633
		6E		BC	3C	MOVZWL	@MODULE_DESC, -(SP)	
	0000G	CF		0A	C0	ADDL2	#10, (SP)	
		1F		02	FB	CALLS	#2, LIB_GET_MEM	
	0000G	DF	00	50	E9	BLBC	STATUS, -2\$	
		56		BE	0E	INSQUE	@NAMBLK, @LIB\$GL_MODUPDL+4	: 1634
		50	08	6E	D0	MOVL	NAMBLK, R6	: 1635
	09	A6		AC	D0	MOVL	MODULE_DESC, R0	
0A	A6	04		60	90	MOVB	(R0), 9(R6)	
		08	04	60	28	MOV3	(R0), @4(R0), 10(R6)	: 1637
		50		AC	90	MOVB	OPERATION, 8(R6)	: 1638
				01	D0	MOVL	#1, R0	: 1640
				04	0003B	RET		: 1641

; Routine Size: 60 bytes, Routine Base: \$CODE\$ + 0645

```
592 1642 1 ROUTINE lib_put_luh ( opcode ) =
593 1643 2 BEGIN
594 1644 3 +++
595 1645 4
596 1646 5 Build the library update history records and put them into
597 1647 6 the history.
598 1648 7
599 1649 8 ---
600 1650 9 FIELD ! define a descriptor for $GETJPI
601 1651 10 itmlst flds =
602 1652 11 SET
603 1653 12 jpi_usrnam_buflen = [0,0,16,0],
604 1654 13 jpi_usrnam_cod = [0,16,16,0],
605 1655 14 jpi_usrnam_bufadr = [1,0,32,0],
606 1656 15 jpi_usrnam_len = [2,0,32,0],
607 1657 16 jpi_end = [3,0,32,0]
608 1658 17 TES;
609 1659 18 LOCAL
610 1660 19 status,
611 1661 20 modnb : REF BBLOCK, ! entry in linked list of module names
612 1662 21 mod_desc : BBLOCK [dsc$c_s_bln],
613 1663 22 luhrec : BBLOCK [dsc$c_s_bln], ! library update history record being built
614 1664 23 itmlst : BLOCK [4, LONG] FIELD (itmlst_flds), ! descriptor for $GETJPI
615 1665 24 iosb : VECTOR [2, LONG], ! status block for $GETJPI
616 1666 25 usrnamlen : WORD,
617 1667 26 usrnambuf : BLOCK [12, BYTE];
618 1668 27 BIND
619 1669 28 luhlen = luhrec [dsc$w_length];
620 1670 29 BUILTIN
621 1671 30 INSQUE,
622 1672 31 REMQUE;
623 1673 32
624 1674 33 IF NOT .lib$gl_ctlmsk [lib$v_recordluh] ! quit if history not maintained
625 1675 34 THEN RETURN true;
626 1676 35
627 1677 36 !
628 1678 37 initialize descriptor for $GETJPI
629 1679 38 which will return USERNAME of the process modifying library
630 1680 39
631 1681 40 itmlst [jpi_usrnam_cod] = jpi$username;
632 1682 41 itmlst [jpi_usrnam_buflen] = 12;
633 1683 42 itmlst [jpi_usrnam_bufadr] = usrnambuf;
634 1684 43 itmlst [jpi_usrnam_len] = usrnamlen;
635 1685 44 itmlst [jpi_end] = 0;
636 1686 45 status = $GETJPI (efn=1, iosb=iosb, itmlst= itmlst); ! Call GET Job Process Information
637 1687 46 $WAITFR (EFN=1); ! set event flag wait
638 1688 47 IF NOT .iosb[0] THEN RETURN .iosb[0]; ! check if an error
639 1689 48 perform (lib_get_mem (lbr$c_maxrecsiz, luhrec [dsc$a_pointer] )); ! get the space for the LUH record
640 1690 49 BEGIN
641 1691 50 LOCAL
642 1692 51 modupdl : VECTOR [2, LONG], ! save module names from different operations
643 1693 52 restore; ! remember that module names have been saved
644 1694 53
645 1695 54 BIND
646 1696 55 luhrecbuf = .luhrec [dsc$a_pointer] : BBLOCK;
647 1697 56
648 1698 57 restore = false;
```

```

: 649      1699  3      modupdl [0] = modupdl;
: 650      1700  3      modupdl [1] = modupdl;
: 651      1701  3      CH$FILL (0, lbr$c_maxrecsiz, luhrecbuf);           ! fill it with nulls
: 652      1702  3      luhrecbuf [lhe$b_usrnamlen] = .usrnamlen;
: 653      1703  3      CH$MOVE ( .usrnamlen, usrnambuf, luhrecbuf [lhe$t_usrnam]); ! copy username into LUH record
: 654      1704  3      luhrecbuf [lhe$b_modcode] = .opcode;           ! record what kind of operation was performed with these mod
: 655      1705  3
: 656      1706  3      !
: 657      1707  3      ! get the date and time
: 658      1708  3      $GETTIM (TIMADR = luhrecbuf [lhe$l_time]);
: 659      1709  3      luhlen = lhe$c_fixedsiz;
: 660      1710  3      luhrecbuf [lhe$w_modcnt] = 0;           ! Tally of module names in record
: 661      1711  3
: 662      1712  3      !
: 663      1713  3      ! copy over module names
: 664      1714  3      WHILE NOT REMQUE (.lib$gl_modupdl, modnb) DO
: 665      1715  4      BEGIN
: 666      1716  4      IF .modnb [lnb$b_flags] EQL .opcode
: 667      1717  4      THEN
: 668      1718  5      BEGIN
: 669      1719  5      IF .luhlen + .modnb [lnb$b_namlng] + 1 GTR lbr$c_maxrecsiz ! Would next modulename make record
: 670      1720  5      THEN ! then put what we've got and then b
: 671      1721  6      BEGIN
: 672      1722  6      status = lbr$put_history (lib$gl_libctl, luhrec);           ! copy the record into the list of h
: 673      1723  6      IF NOT .status
: 674      1724  6      THEN
: 675      1725  7      BEGIN ! If there's been an error, deallocate memory and quit
: 676      1726  7      lib free_mem ( lbr$c_maxrecsiz, .luhrec [dsc$a_pointer] );
: 677      1727  7      RETURN .status;
: 678      1728  6      END;
: 679      1729  6      luhlen = lhe$c_fixedsiz;           ! header for continuation record is the same
: 680      1730  6      luhrecbuf [lhe$w_modcnt] = 0;           ! except for module count which is reset to 0
: 681      1731  5      END;
: 682      1732  6      BEGIN
: 683      1733  6      BIND
: 684      1734  6      modnamlen = luhrecbuf + .luhlen : BYTE;
: 685      1735  6      modnamlen = .modnb [lnb$b_namlng];
: 686      1736  5      END;
: 687      1737  5      luhlen = .luhlen + 1;           ! Allow 1 byte to store module name
: 688      1738  5      CH$MOVE (.modnb [lnb$b_namlng], modnb [lnb$t_name],           ! copy in module name
: 689      1739  5      luhrecbuf + .luhlen);
: 690      1740  5      luhrecbuf [lhe$w_modcnt] = .luhrecbuf [lhe$w_modcnt] + 1; ! Tally module names in record
: 691      1741  5      luhlen = .luhlen + .modnb [lnb$b_namlng];
: 692      1742  5      END ! if the module name matches the operation being logged
: 693      1743  4      ELSE
: 694      1744  5      BEGIN
: 695      1745  5      INSQUE ( .modnb, .modupdl [1] );           ! save module name
: 696      1746  5      restore = true;
: 697      1747  4      END;
: 698      1748  3      END;           ! while
: 699      1749  3
: 700      1750  3      IF .restore
: 701      1751  3      THEN
: 702      1752  3      WHILE NOT REMQUE (.modupdl, modnb) DO
: 703      1753  3      INSQUE (.modnb, .lib$gl_modupdl [1] );
: 704      1754  3
: 705      1755  3      IF .luhrecbuf [lhe$w_modcnt] GTR 0
```

```
: 706      1756  3      THEN
: 707      1757  3      status = lbr$put_history ( lib$gl_libctl, luhrec);      ! out of module names so put what we've got
: 708      1758  2      END;
: 709      1759  2
: 710      1760  2      lib_free_mem ( lbr$sc_maxrecsiz, .luhrec [dsc$a_pointer] );      ! return memory
: 711      1761  2      RETURN .status;
: 712      1762  1      END;      ! routine lib_put_luh
: INFO#250      L1:1702
: Referenced LOCAL symbol USRNAMLEN is probably not initialized
```

.EXTRN SYSS\$GETJPI, SYSS\$WAITFR
.EXTRN SYSS\$GETTIM

07FC 00000 LIB_PUT_LUH:

```
.WORD      Save R2,R3,R4,R5,R6,R7,R8,R9,R10      : 1642
MOVAB      LBR$PUT_HISTORY, R10
MOVAB      -64(SP), SP
BBS        #6, LIB$GL_CTLMSK+3, 1$      : 1674
MOVL       #1, R0      : 1675
RET
MOVL       #33685516, ITMLST      : 1682
MOVAB      USRNAMBUF, ITMLST+4      : 1683
MOVAB      USRNAMLEN, ITMLST+8      : 1684
CLRL       ITMLST+12      : 1685
CLRQ       -(SP)      : 1686
PUSHAB     IOSB
PUSHAB     ITMLST
CLRQ       -(SP)
PUSHL      #1
CALLS      #7, SYSS$GETJPI
MOVL       R0, STATUS
PUSHL      #1      : 1687
CALLS      #1, SYSS$WAITFR      : 1688
BLBS       IOSB, 2$      : 1688
MOVL       IOSB, R0
RET
PUSHAB     LUHREC+4      : 1689
MOVZWL     #2048, -(SP)
CALLS      #2, LIB_GET_MEM
BLBS       STATUS, -3$
RET
MOVL       LUHREC+4, R6      : 1696
CLRL       RESTORE      : 1698
MOVAB      MODUPDL, MODUPDL      : 1699
MOVAB      MODUPDL, MODUPDL+4      : 1700
MOVC5      #0, (SP), #0, #2048, (R6)      : 1701
MOVB       USRNAMLEN, 11(R6)      : 1702
MOVC3      USRNAMLEN, USRNAMBUF, 12(R6)      : 1703
MOVB       OPERCODE, 10(R6)      : 1704
PUSHL      R6      : 1708
CALLS      #1, SYSS$GETTIM
MOVL       #24, LUHLEN      : 1709
CLRQ       8(R6)      : 1710
REMQUE     @LIB$GL_MODUPDL, MODN8      : 1714
```

```
04      0000G      5A 00000000G 00 9E 00002
5E      C0      AE 9E 00009
CF      06      E0 0000D
50      01      D0 00013
04      04      00016
20      AE 0202000C 8F D0 00017 1$:
24      AE      0C      AE 9E 0001F
28      AE      6E 9E 00024
2C      AE D4 00028
7E 7C 0002B
20      AE 9F 0002D
2C      AE 9F 00030
7E 7C 00033
01      DD 00035
00000000G 00      07 FB 00037
59      50 D0 0003E
01      DD 00041
00000000G 00      01 FB 00043
05      18      AE E8 0004A
50      18      AE D0 0004E
04      00052
34      AE 9F 00053 2$:
0000G 7E 0800 8F 3C 00056
CF      02 FB 0005B
01      50 E8 00060
04      00063
56      34      AE D0 00064 3$:
04      AE 58 D4 00068
08      AE 04      AE 9E 0006A
AE      04      AE 9E 0006F
6E      00      2C 00074
0800 8F      00      66      0007B
00      6E 90 0007C
OC      A6      0C      AE 6E 28 00080
OA      A6      04      AC 90 00086
56      DD 0008B
00000000G 00      01 FB 0008D
30      AE      18 D0 00094
08      A6 B4 00098
57      0000G DF 0F 0009B 4$:
0000G
```

04	AC	08	A7	08	60	1D	000A0	BVS	7\$	:	1716
					00	ED	000A2	CMPZV	#0, #8, 8(MODNB), OPERCODE	:	
					4E	12	000A9	BNEQ	6\$	:	1719
				50	09	A7	9A	MOVZBL	9(MODNB), R0	:	
				50	30	AE	C0	ADDL2	LUHLEN, R0	:	
						50	D6	INCL	R0	:	
		00000800		8F		50	D1	CMPL	R0, #2048	:	
						17	15	BLEQ	5\$	:	
					30	AE	9F	PUSHAB	LUHREC	:	1722
					0000G	CF	9F	PUSHAB	LIB\$GL_LIBCTL	:	
				6A		02	FB	CALLS	#2, LBR\$PUT_HISTORY	:	
				59		50	D0	MOVL	R0, STATUS	:	1723
				56		59	E9	BLBC	STATUS, 10\$	:	1729
		30		AE		18	D0	MOVL	#24, LUHLEN	:	1730
					08	A6	B4	CLRW	8(R6)	:	1734
		50		56	30	AE	C1	ADDL3	LUHLEN, R6, R0	:	1735
				60	09	A7	90	MOVB	9(MODNB), (R0)	:	1737
					30	AE	D6	INCL	LUHLEN	:	1738
				50	09	A7	9A	MOVZBL	9(MODNB), R0	:	1739
		30	BE46	0A		50	28	MOVC3	R0, 10(MODNB), @LUHLEN[R6]	:	1740
				A7	08	A6	B6	INCL	8(R6)	:	1741
				50	09	A7	9A	MOVZBL	9(MODNB), R0	:	
				AE		50	C0	ADDL2	R0, LUHLEN	:	1716
						A2	11	BRB	4\$	:	1745
				08		67	0E	INSQUE	(MODNB), @MODUPDL+4	:	1746
				BE		01	D0	MOVL	#1, RESTORE	:	1714
				58		99	11	BRB	4\$	:	1750
				0D		58	E9	BLBC	RESTORE, 9\$	:	1752
				57	04	BE	0F	REMQUE	@MODUPDL, MODNB	:	
						07	1D	BVS	9\$	:	1753
		0000G		DF		67	0E	INSQUE	(MODNB), @LIB\$GL_MODUPDL+4	:	1755
						F3	11	BRB	8\$	:	1757
					08	A6	B5	TSTW	8(R6)	:	
						0D	13	BEQL	10\$	:	1760
					30	AE	9F	PUSHAB	LUHREC	:	
					0000G	CF	9F	PUSHAB	LIB\$GL_LIBCTL	:	
				6A		02	FB	CALLS	#2, LBR\$PUT_HISTORY	:	
				59		50	D0	MOVL	R0, STATUS	:	
					34	AE	DD	PUSHL	LUHREC+4	:	
				7E	0800	8F	3C	MOVZWL	#2048, -(SP)	:	1761
		0000G		CF		02	FB	CALLS	#2, LIB_FREE_MEM	:	
				50		59	D0	MOVL	STATUS, R0	:	1762
						04	00134	RET		:	

; Routine Size: 309 bytes, Routine Base: \$CODE\$ + 0681

: 713 1763 1 END
: 714 1764 0 ELUDOM

! Of module

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

LIB_PROCMD
V04=000

E 7
16-Sep-1984 02:03:14
14-Sep-1984 12:38:09

VAX-11 Bliss-32 V4.0-742
[LIBRAR.SRC]PROCMD.B32;1

Page 31
(11)

Name	Bytes	Attributes
\$OWNS	6	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODE\$	1974	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	54	0	581	00:01.0

: Information: 1
: Warnings: 0
: Errors: 0

COMMAND QUALIFIERS

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

: Size: 1974 code + 6 data bytes
: Run Time: 00:40.5
: Elapsed Time: 01:22.4
: Lines/CPU Min: 2612
: Lexemes/CPU-Min: 32819
: Memory Used: 291 pages
: Compilation Complete

0202

AH-BT13A-SE
 VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY