



```

LL          IIIIII  BBBB BBBB  IIIIII  NN      NN  TTTTTTTTTT  RRRRRRRR  EEEEEEEEEE
LL          IIIIII  BBBB BBBB  IIIIII  NN      NN  TTTTTTTTTT  RRRRRRRR  EEEEEEEEEE
LL          II      BB      BB      BB      NN      NN      TT      RR      RR  EE
LL          II      BB      BB      BB      NN      NN      TT      RR      RR  EE
LL          II      BB      BB      BB      NN      NN      TT      RR      RR  EE
LL          II      BB      BB      BB      NN      NN      TT      RR      RR  EE
LL          II      BBBB BBBB  IIIIII  NN      NN      TT      RRRRRRRR  EEEEEEEEEE
LL          II      BBBB BBBB  IIIIII  NN      NN      TT      RRRRRRRR  EEEEEEEEEE
LL          II      BB      BB      BB      NN      NN      TT      RR      RR  EE
LL          II      BB      BB      BB      NN      NN      TT      RR      RR  EE
LL          II      BB      BB      BB      NN      NN      TT      RR      RR  EE
LL          II      BB      BB      BB      NN      NN      TT      RR      RR  EE
LL          II      BB      BB      BB      NN      NN      TT      RR      RR  EE
LL          II      BB      BB      BB      NN      NN      TT      RR      RR  EE
LLLLLLLLLLL IIIIII  BBBB BBBB  IIIIII  NN      NN      TT      RR      RR  EEEEEEEEEE
LLLLLLLLLLL IIIIII  BBBB BBBB  IIIIII  NN      NN      TT      RR      RR  EEEEEEEEEE

```

```

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
LLLLLLLLLLL IIIIII  SSSSSSSS
LLLLLLLLLLL IIIIII  SSSSSSSS

```



```
1 0001 0 %TITLE 'LIBSBINARY TREE - Subroutines to manipulate balanced binary trees'
2 0002 0 MODULE LIBSBINARY_TREE (
3 0003 0 IDENT = '1-001' ! File: LIBBINTRE.B32 Edit: BLS1001
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 ++
32 0032 1 FACILITY: General Utility Library
33 0033 1
34 0034 1 ABSTRACT:
35 0035 1
36 0036 1 These procedures allow easy manipulation of balanced binary trees
37 0037 1
38 0038 1 ENVIRONMENT: Runs at any access mode - AST reentrant
39 0039 1
40 0040 1 AUTHOR: Benn L. Schreiber, CREATION DATE: 08-May-1981
41 0041 1
42 0042 1 MODIFIED BY:
43 0043 1
44 0044 1 1-001 - Original. BLS 08-May-1981
45 0045 1 --
46 0046 1
```

```

48 0047 1 %SBTTL 'Declarations'
49 0048 1
50 0049 1  : SWITCHES:
51 0050 1
52 0051 1
53 0052 1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
54 0053 1
55 0054 1
56 0055 1  : LINKAGES:
57 0056 1
58 0057 1
59 0058 1 LINKAGE
60 0059 1   jsb_insert = JSB :                               !Linkage for INSERT_NODE
61 0060 1     GLOBAL (currentnode = 11,
62 0061 1       keyname = 10,
63 0062 1       compare_routine = 9,
64 0063 1       alloc_routine = 8,
65 0064 1       blockretadr = 7,
66 0065 1       foundintree = 6,
67 0066 1       controlflags = 5,
68 0067 1       user_context = 4)
69 0068 1     PRESERVE (2,3),
70 0069 1
71 0070 1   jsb_traverse = JSB (REGISTER = 0) :             !Linkage for traverse
72 0071 1     GLOBAL (action_routine = 11,
73 0072 1       user_context = 10)
74 0073 1     NOTUSED (2,3,4,5,6,7,8,9);
75 0074 1
76 0075 1
77 0076 1  : TABLE OF CONTENTS:
78 0077 1
79 0078 1
80 0079 1 FORWARD ROUTINE
81 0080 1   insert_node : jsb_insert,                       !Insert node into tree
82 0081 1   lib$insert_tree,                               !Entry point to insert node
83 0082 1   lib$lookup_tree,                             !Entry point to lookup node
84 0083 1   traverse_tree : jsb_traverse,                !Internal routine to insert into tree
85 0084 1   lib$traverse_tree;                            !Entry point to traverse tree
86 0085 1
87 0086 1
88 0087 1  : INCLUDE FILES:
89 0088 1
90 0089 1
91 0090 1 LIBRARY 'RTLSTARLE';                             ! System symbols, typically from SYSS$LIBRARY:STARLET.L32
92 0091 1
93 0092 1 REQUIRE 'RTLIN:RTLPSECT';                       ! Define PSECT declarations macros
94 0187 1
95 0188 1
96 0189 1  : MACROS:
97 0190 1
98 0191 1
99 0192 1 MACRO
100 0193 1
101 0194 1   : Define offsets in basic node
102 0195 1
103 0196 1   NODE$LEFT = 0,0,32,0%,                          !Left subtree pointer
104 0197 1   NODE$RIGHT = 4,0,32,0%,                          !Right subtree pointer

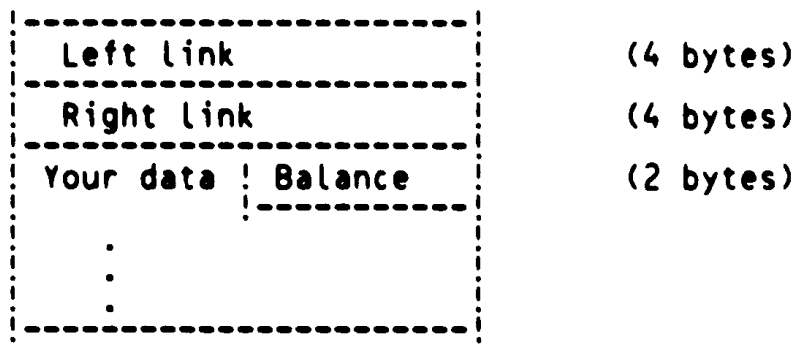
```

```

: 105      0198      1      NODE$W_BAL = 8,0,16,1%      !Current node balance
: 106      0199      1      NODE$L_PTR = 10,0,32,0%      !Pointer to user data
: 107      0200      1
: 108      0201      1      ! Define flags in control_flags argument to lib$insert_tree
: 109      0202      1
: 110      0203      1      TREE$V_INS_DUP = 0,0,1,0%      !Insert duplicates
: 111      0204      1
: 112      0205      1
: 113      0206      1      ! EQUATED SYMBOLS:
: 114      0207      1
: 115      0208      1
: 116      0209      1      LITERAL
: 117      0210      1      true = 1      !Success
: 118      0211      1      false = 0;      !Failure
: 119      0212      1
: 120      0213      1
: 121      0214      1      ! FIELDS:
: 122      0215      1
: 123      0216      1      NONE
: 124      0217      1
: 125      0218      1      ! PSECTS:
: 126      0219      1
: 127      0220      1
: 128      0221      1      DECLARE_PSECTS (LIB);      ! Declare PSECTS for LIB$ facility
: 129      0222      1
: 130      0223      1
: 131      0224      1      ! OWN STORAGE:
: 132      0225      1
: 133      0226      1      NONE
: 134      0227      1
: 135      0228      1      ! EXTERNAL REFERENCES:
: 136      0229      1
: 137      0230      1
: 138      0231      1      EXTERNAL LITERAL
: 139      0232      1      lib$normal,      !Success
: 140      0233      1      lib$keyalrins,      !Key already inserted--not reinserted
: 141      0234      1      lib$keynotfou,      !Key not found
: 142      0235      1      lib$insvirmem;      !Insufficient virtual memory

```

```
144 0236 1 %SBTTL 'Data Structure Description'  
145 0237 1 ++  
146 0238 1 A node is structured as follows:  
147 0239 1  
148 0240 1  
149 0241 1  
150 0242 1  
151 0243 1  
152 0244 1  
153 0245 1  
154 0246 1  
155 0247 1  
156 0248 1  
157 0249 1  
158 0250 1  
159 0251 1  
160 0252 1  
161 0253 1  
162 0254 1  
163 0255 1  
164 0256 1  
165 0257 1  
166 0258 1  
167 0259 1  
168 0260 1  
169 0261 1  
170 0262 1  
171 0263 1  
172 0264 1  
173 0265 1  
174 0266 1  
175 0267 1  
176 0268 1  
177 0269 1  
178 0270 1  
179 0271 1
```



Therefore, any data structures you declare that will be manipulated by these routines must contain 10 bytes of reserved data at the beginning for the binary tree routines.

These routines can be used for data allocated in one of two ways:

- 1) The node (described above) is followed immediately by your data. In this case your allocation routine must allocate a block equal in size to the sum of your data + 10 node bytes.
- 2) The node which is allocated contains the data described above and a longword pointer to your data. This is useful if some data must be entered into two or more tables simultaneously. The user\_data argument passed to lib\$insert\_tree should be the address of the data, which will be passed to your compare and allocated routines. The allocate routine simply stores this user\_data in the field of the allocated node.

--

```

181 0272 1 %SBTTL 'INSERT_NODE - insert node into balanced tree'
182 0273 1 ROUTINE insert_node : jsb_insert =
183 0274 2 BEGIN
184 0275 2
185 0276 2 Recursive routine to insert the new node into the tree
186 0277 2
187 0278 2 IMPLICIT INPUTS:
188 0279 2
189 0280 2     currentnode      Current node in tree walk
190 0281 2     keyname          Address of key
191 0282 2     compare_routine Address of routine to compare key with current node
192 0283 2     alloc_routine   Address of routine to allocate new node
193 0284 2     blockretadr     Address to return allocated node
194 0285 2
195 0286 2 ROUTINE VALUE:
196 0287 2
197 0288 2     true [=1]        Tree is balanced
198 0289 2     false [=0]      Tree not balanced
199 0290 2
200 0291 2 LOCAL
201 0292 2     down_left : REF BLOCK[.BYTE],           !Pointer to left son
202 0293 2     down_right : REF BLOCK[.BYTE],         !Pointer to right son
203 0294 2     save_current : REF BLOCK[.BYTE],       !Temp to save currentnode
204 0295 2     status,                                     !Return status from calls
205 0296 2     in_balance;                                !True if subtree in balance
206 0297 2
207 0298 2 EXTERNAL REGISTER
208 0299 2     currentnode = 11 : REF BLOCK[.BYTE],     !Pointer to current node
209 0300 2     keyname = 10 : REF VECTOR[.BYTE],        !Pointer to new key
210 0301 2     compare_routine = 9,                    !Address of caller's compare routine
211 0302 2     alloc_routine = 8,                      !Address of caller's allocate node routine
212 0303 2     blockretadr = 7 : REF VECTOR[.LONG],     !Address to return result to caller
213 0304 2     foundintree = 6,                        !True if entry found in tree, false if inser
214 0305 2     controlflags = 5 : BLOCK[4,.BYTE],      !User control flags
215 0306 2     user_context = 4;                       !User context to pass to allocate node and c
216 0307 2
217 0308 2
218 0309 2 Check for bottom of tree
219 0310 2
220 0311 2 IF .currentnode EQL 0
221 0312 2 THEN BEGIN
222 0313 2     IF NOT (.alloc_routine) (.keyname,save_current,.user_context) !Allocate a node
223 0314 2     THEN RETURN (foundintree = lib$_insvirmem); ! returning an error if no memory is
224 0315 2     currentnode = .save_current; !Set it as the current node
225 0316 2     currentnode[node$_left] = 0; !Initialize the node pointers
226 0317 2     currentnode[node$_right] = 0;
227 0318 2     currentnode[node$_w_bal] = 0; ! and the node balance
228 0319 2     blockretadr[0] = .currentnode; !Return address to caller
229 0320 2     RETURN false !Return tree not balanced
230 0321 2     END;
231 0322 2
232 0323 2 Check if key is to the left or right subtree
233 0324 2
234 0325 2 save_current = .currentnode; !Save address of current node
235 0326 2 IF (in_balance = (.compare_routine)(.keyname,.currentnode, !Call caller's compare routine
236 0327 2     .user_context)) LEQ 0
237 0328 2 THEN BEGIN

```

```
238 0329 3 | | |
239 0330 3 | | | Insert the node into the left subtree
240 0331 3 | | |
241 0332 3 | | | IF .in_balance EQL 0 | | | !First check if this is really it
242 0333 3 | | | AND NOT .controlflags[tree$ins_dup] | | | ! and that duplicates are not to be
243 0334 4 | | | THEN BEGIN | | |
244 0335 4 | | | | | |
245 0336 4 | | | | | | We found a match in the table, and caller requested that
246 0337 4 | | | | | | duplicates are not to be inserted
247 0338 4 | | | | | |
248 0339 4 | | | | | | blockretadr[0] = .currentnode; | | | ! and if it is return to caller
249 0340 5 | | | | | | RETURN (foundintree = lib$keyalrins)
250 0341 5 | | | | | | END;
251 0342 3 | | | | | | currentnode = .currentnode[node$l_left]; | | | !Insert into left subtree
252 0343 3 | | | | | | in_balance = insert_node(); | | | !Recurse to insert
253 0344 4 | | | | | | IF (.foundintree EQL lib$keyalrins) | | | !If we found it
254 0345 4 | | | | | | OR (.foundintree EQL lib$insvirmem)
255 0346 3 | | | | | | THEN RETURN true; | | | ! then we are done
256 0347 3 | | | | | | down_left = .currentnode; | | | !Save address of new subtree
257 0348 3 | | | | | | currentnode = .save_current; | | | !Return to father node
258 0349 3 | | | | | | currentnode[node$l_left] = .down_left; | | | !Father points to new left subtree
259 0350 3 | | | | | | IF .in_balance | | | !If tree is in balance
260 0351 3 | | | | | | THEN RETURN true | | | ! then all done, return true
261 0352 4 | | | | | | ELSE BEGIN | | |
262 0353 4 | | | | | | | | |
263 0354 4 | | | | | | | | | The left subtree has grown higher, restore balance
264 0355 4 | | | | | | | | |
265 0356 4 | | | | | | | | | currentnode[node$w_bal] = .currentnode[node$w_bal] - 1; | | | !Perform left shift of tree
266 0357 4 | | | | | | | | | IF .currentnode[node$w_bal] EQL 0 | | | ! If in perfect balance
267 0358 4 | | | | | | | | | THEN RETURN true | | | ! then return in balance
268 0359 5 | | | | | | | | | ELSE BEGIN | | |
269 0360 5 | | | | | | | | | IF .currentnode[node$w_bal] | | | !Exit if AVL balanced
270 0361 5 | | | | | | | | | THEN RETURN false; | | | ! Return tree not in balance
271 0362 5 | | | | | | | | | down_left = .currentnode[node$l_left]; | | | !Get left subtree
272 0363 5 | | | | | | | | | IF .down_left[node$w_bal] LSS 0 | | | !Test balance of left subtree
273 0364 6 | | | | | | | | | THEN BEGIN | | |
274 0365 6 | | | | | | | | | | | |
275 0366 6 | | | | | | | | | | | | Perform single LL rotation
276 0367 6 | | | | | | | | | | | |
277 0368 6 | | | | | | | | | | | | currentnode[node$l_left] = .down_left[node$l_right];
278 0369 6 | | | | | | | | | | | | down_left[node$l_right] = .currentnode;
279 0370 6 | | | | | | | | | | | | currentnode[node$w_bal] = 0; | | | !Mark in balance
280 0371 6 | | | | | | | | | | | | currentnode = .down_left; | | | !Set new root of subtree
281 0372 6 | | | | | | | | | | | | currentnode[node$w_bal] = 0; | | | !Mark in balance
282 0373 6 | | | | | | | | | | | | RETURN true | | | !Return tree in balance
283 0374 6 | | | | | | | | | | | | END
284 0375 6 | | | | | | | | | | | | ELSE BEGIN | | |
285 0376 6 | | | | | | | | | | | | | | |
286 0377 6 | | | | | | | | | | | | | | | Perform double LR rotation
287 0378 6 | | | | | | | | | | | | | | |
288 0379 6 | | | | | | | | | | | | | | | down_right = .down_left[node$l_right]; | | | !Get right son of left subtree
289 0380 6 | | | | | | | | | | | | | | | down_left[node$l_right] = .down_right[node$l_left];
290 0381 6 | | | | | | | | | | | | | | | down_right[node$l_left] = .down_left;
291 0382 6 | | | | | | | | | | | | | | | currentnode[node$l_left] = .down_right[node$l_right];
292 0383 6 | | | | | | | | | | | | | | | down_right[node$l_right] = .currentnode;
293 0384 6 | | | | | | | | | | | | | | | currentnode[node$w_bal] = 0; | | | !Initialize to balanced
294 0385 6 | | | | | | | | | | | | | | | down_left[node$w_bal] = 0;
```



```

295      0386 6          IF .down_right[node$w_bal] GTR 0          !Check balance of new subtree
296      0387 6          THEN down_left[node$w_bal] = -1        ! Mark left side heavy
297      0388 6          ELSE IF .down_right[node$w_bal] LSS 0
298      0389 6          THEN currentnode[node$w_bal] = 1;      ! Mark left side heavy
299      0390 6          currentnode = .down_right;             !Set new root of subtree
300      0391 6          currentnode[node$w_bal] = 0;           !Mark in balance
301      0392 6          RETURN true                             !Return tree in balance
302      0393 6          END
303      0394 5          END
304      0395 4          END
305      0396 3          END
306      0397 3          : Insert node into the right subtree
307      0398 3          :
308      0399 3          ELSE BEGIN
309      0400 3          currentnode = .currentnode[node$l_right];
310      0401 3          in_balance = insert_node();
311      0402 3          IF (.foundintree EQ[ lib$ keyalrins)
312      0403 4          OR (.foundintree EQL [lib$ insvirmem)
313      0404 4          THEN RETURN true;
314      0405 3          down_right = .currentnode;
315      0406 3          currentnode = .save_current;
316      0407 3          currentnode[node$l_right] = .down_right;
317      0408 3          IF .in_balance
318      0409 3          THEN RETURN true
319      0410 3          ELSE BEGIN
320      0411 4          : The right subtree has grown higher -- restore balance
321      0412 4          :
322      0413 4          currentnode[node$w_bal] = .currentnode[node$w_bal] + 1;
323      0414 4          IF .currentnode[node$w_bal] EQL 0
324      0415 4          THEN RETURN true;
325      0416 4          IF .currentnode[node$w_bal]
326      0417 4          THEN RETURN false;
327      0418 4          down_right = .currentnode[node$l_right];
328      0419 4          IF .down_right[node$w_bal] GTR 0
329      0420 4          THEN BEGIN
330      0421 4          : Perform single RR rotation
331      0422 5          :
332      0423 5          currentnode[node$l_right] = .down_right[node$l_left];
333      0424 5          down_right[node$l_left] = .currentnode;
334      0425 5          currentnode[node$w_bal] = 0;
335      0426 5          currentnode = .down_right;
336      0427 5          currentnode[node$w_bal] = 0;
337      0428 5          RETURN true
338      0429 5          END
339      0430 5          ELSE BEGIN
340      0431 5          : Perform double RL rotation
341      0432 5          :
342      0433 5          down_left = .down_right[node$l_left];
343      0434 5          down_right[node$l_left] = .down_left[node$l_right];
344      0435 5          down_left[node$l_right] = .down_right;
345      0436 5          currentnode[node$l_right] = .down_left[node$l_left];
346      0437 5          down_left[node$l_left] = .currentnode;
347      0438 5          currentnode[node$w_bal] = 0;
348      0439 5          : Initialize balance indicators
349      0440 5
350      0441 5
351      0442 5
    
```

```

: 352      0443 5      down_right[node$w_bal] = 0;
: 353      0444 5      IF .down_left[node$w_bal] LSS 0
: 354      0445 5      THEN down_right[node$w_bal] = 1
: 355      0446 5      ELSE IF .down_left[node$w_bal] GTR 0
: 356      0447 5      THEN currentnode[node$w_bal] = -1;
: 357      0448 5      currentnode = .down_left;
: 358      0449 5      currentnode[node$w_bal] = 0;
: 359      0450 5      RETURN true
: 360      0451 5      END
: 361      0452 4      END
: 362      0453 2      END;
: 363      0454 2
: 364      0455 1      END;

```

!Of insert\_node

```

.TITLE LIB$BINARY_TREE LIB$BINARY TREE - Subroutines to
       o manipulate bal
.IDENT \1-001\
.EXTRN LIB$NORMAL, LIB$KEYALRINS
.EXTRN LIB$KEYNOTFOU, LIB$INSVIRMEM
.PSECT _LIB$CODE, NOWRT, SHR, PIC, 2

```

	52	DD	00000	INSERT_NODE:		
				PUSHL	R2	: 0273
5E	04	C2	00002	SUBL2	#4, SP	
	5B	D5	00005	TSTL	CURRENTNODE	: 0311
	24	12	00007	BNEQ	3\$	
	54	DD	00009	PUSHL	USER_CONTEXT	: 0313
	04	AE	9F 0000B	PUSHAB	SAVE_CURRENT	
	5A	DD	0000E	PUSHL	KEYNAME	
68	03	FB	00010	CALLS	#3, (ALLOC_ROUTINE)	
09	50	E8	00013	BLBS	RO, 1\$	
56	00000000G	8F	D0 00016	MOVL	#LIB\$INSVIRMEM, FOUNDINTREE	: 0314
	2F	11	0001D	BRB	5\$	
5B	6E	D0	0001F 1\$:	MOVL	SAVE_CURRENT, CURRENTNODE	: 0315
	6B	7C	00022	CLRQ	(CURRENTNODE)	: 0316
	08	AB	B4 00024	CLRW	8(CURRENTNODE)	: 0318
67	5B	D0	00027	MOVL	CURRENTNODE, (BLOCKRETADR)	: 0319
	010E	31	0002A 2\$:	BRW	19\$	: 0320
6E	5B	D0	0002D 3\$:	MOVL	CURRENTNODE, SAVE_CURRENT	: 0325
	54	DD	00030	PUSHL	USER_CONTEXT	: 0327
7E	5A	7D	00032	MOVQ	KEYNAME, -(SP)	: 0326
69	03	FB	00035	CALLS	#3, (COMPARE_ROUTINE)	
	50	D5	00038	TSTL	IN_BALANCE	: 0327
	03	15	0003A	BLEQ	4\$	
	0085	31	0003C	BRW	10\$	
	13	12	0003F 4\$:	BNEQ	6\$	: 0332
10	55	E8	00041	BLBS	CONTROLFLAGS, 6\$	: 0333
67	5B	D0	00044	MOVL	CURRENTNODE, (BLOCKRETADR)	: 0339
56	00000000G	8F	D0 00047	MOVL	#LIB\$KEYALRINS, FOUNDINTREE	: 0340
50	56	D0	0004E 5\$:	MOVL	FOUNDINTREE, RO	
	00E9	31	00051	BRW	20\$	
5B	68	D0	00054 6\$:	MOVL	(CURRENTNODE), CURRENTNODE	: 0342
	A7	10	00057	BSBB	INSERT_NODE	: 0343
00000000G	8F	56	D1 00059	CMPL	FOUNDINTREE, #LIB\$KEYALRINS	: 0344

00000000G	8F	79	13	00060	BEQL	11\$	0345		
		56	D1	00062	CML	FOUNDTREE, #LIBS_INSVIRMEM			
		70	13	00069	BEQL	11\$			
	51	5B	D0	0006B	MOVL	CURRENTNODE, DOWN_LEFT	0347		
	5B	6E	D0	0006E	MOVL	SAVE_CURRENT, CURRENTNODE	0348		
	6B	51	D0	00071	MOVL	DOWN_LEFT, (CURRENTNODE)	0349		
	70	50	E8	00074	BLBS	IN BALANCE, 12\$	0350		
	50	08	AB	9E	00077	MOVAB	8(CURRENTNODE), R0	0356	
		60	B7	0007B	DECW	(R0)			
		71	13	0007D	BEQL	13\$	0357		
	AB	60	E8	0007F	BLBS	(R0), 2\$	0360		
	51	6B	D0	00082	MOVL	(CURRENTNODE), DOWN_LEFT	0362		
		08	A1	B5	00085	TSTW	8(DOWN_LEFT)	0363	
		0D	18	00088	BGEQ	7\$			
	6B	04	A1	D0	0008A	MOVL	4(DOWN_LEFT), (CURRENTNODE)	0368	
04	A1		5B	D0	0008E	MOVL	CURRENTNODE, 4(DOWN_LEFT)	0369	
			60	B4	00092	CLRW	(R0)	0370	
			0099	31	00094	BRW	16\$	0371	
	52	04	A1	D0	00097	7\$:	MOVL	4(DOWN_LEFT), DOWN_RIGHT	0379
04	A1		62	D0	0009B	MOVL	(DOWN_RIGHT), 4(DOWN_LEFT)	0380	
	62		51	D0	0009F	MOVL	DOWN_LEFT, (DOWN_RIGHT)	0381	
	6B	04	A2	D0	000A2	MOVL	4(DOWN_RIGHT), (CURRENTNODE)	0382	
04	A2		5B	D0	000A6	MOVL	CURRENTNODE, 4(DOWN_RIGHT)	0383	
			60	B4	000AA	CLRW	(R0)	0384	
		08	A1	B4	000AC	CLRW	8(DOWN_LEFT)	0385	
		08	A2	B5	000AF	TSTW	8(DOWN_RIGHT)	0386	
			06	15	000B2	BLEQ	8\$		
08	A1		01	AE	000B4	MNEGW	#1, 8(DOWN_LEFT)	0387	
			05	11	000B8	BRB	9\$		
			03	18	000BA	8\$:	BGEQ	9\$	0388
	60		01	B0	000BC	MOVW	#1, (R0)	0389	
	5B		52	D0	000BF	9\$:	MOVL	DOWN_RIGHT, CURRENTNODE	0390
			6F	11	000C2	BRB	17\$	0391	
	5B	04	AB	D0	000C4	10\$:	MOVL	4(CURRENTNODE), CURRENTNODE	0401
			FF35	30	000C8	BSBW	INSERT_NODE	0402	
00000000G	8F		56	D1	000CB	CML	FOUNDTREE, #LIBS_KEYALRINS	0403	
00000000G	8F		62	13	000D2	BEQL	18\$		
			56	D1	000D4	CML	FOUNDTREE, #LIBS_INSVIRMEM	0404	
			59	13	000DB	11\$:	BEQL	18\$	
	52		5B	D0	000DD	MOVL	CURRENTNODE, DOWN_RIGHT	0406	
	5B		6E	D0	000E0	MOVL	SAVE_CURRENT, CURRENTNODE	0407	
04	AB		52	D0	000E3	MOVL	DOWN_RIGHT, 4(CURRENTNODE)	0408	
	4C		50	E8	000E7	12\$:	BLBS	IN BALANCE, 18\$	0409
	50	08	AB	9E	000EA	MOVAB	8(CURRENTNODE), R0	0415	
			60	B6	000EE	INCW	(R0)		
			44	13	000F0	13\$:	BEQL	18\$	0416
	46		60	E8	000F2	BLBS	(R0), 19\$	0418	
	52	04	AB	D0	000F5	MOVL	4(CURRENTNODE), DOWN_RIGHT	0420	
		08	A2	B5	000F9	TSTW	8(DOWN_RIGHT)	0421	
			0B	15	000FC	BLEQ	14\$		
04	AB		62	D0	000FE	MOVL	(DOWN_RIGHT), 4(CURRENTNODE)	0426	
	62		5B	D0	00102	MOVL	CURRENTNODE, (DOWN_RIGHT)	0427	
			60	B4	00105	CLRW	(R0)	0428	
			B6	11	00107	BRB	9\$	0429	
	51		62	D0	00109	14\$:	MOVL	(DOWN_RIGHT), DOWN_LEFT	0437
	62	04	A1	D0	0010C	MOVL	4(DOWN_LEFT), (DOWN_RIGHT)	0438	
04	A1		52	D0	00110	MOVL	DOWN_RIGHT, 4(DOWN_LEFT)	0439	

```

04 AB          61 D0 00114      MOVL (DOWN_LEFT), 4(CURRENTNODE)      : 0440
61             5B D0 00118      MOVL CURRENTNODE, (DOWN_LEFT)       : 0441
                60 B4 0011B      CLRW (R0)                            : 0442
                08 A2 B4 0011D      CLRW 8(DOWN_RIGHT)                   : 0443
                08 A1 B5 00120      TSTW 8(DOWN_LEFT)                    : 0444
                06 18 00123      BGEQ 15$                             :
08 A2          01 B0 00125      MOVW #1, 8(DOWN_RIGHT)                : 0445
                05 11 00129      BRB 16$                              :
                03 15 0012B 15$: BLEQ 16$                               : 0446
                60          01 AE 0012D MNEGW #1, (R0)                       : 0447
                5B          51 D0 00130 16$: MOVL DOWN_LEFT, CURRENTNODE       : 0448
                08 AB B4 00133 17$: CLRW 8(CURRENTNODE)                       : 0449
                50          01 D0 00136 18$: MOVL #1, R0                          : 0450
                02 11 00139      BRB 20$                              : 0400
                50 D4 0013B 19$: CLRL R0                                       : 0455
                04 C0 0013D 20$: ADDL2 #4, SP                                  :
                04 BA 00140      POPR #^M<R2>                               :
                05 00142      RSB                                           :

```

; Routine Size: 323 bytes, Routine Base: \_LIB\$CODE + 0000

```

366 0456 1 %SBTTL 'LIB$INSERT_TREE - User entry to insert in tree'
367 0457 1 GLOBAL ROUTINE lib$insert_tree (treehead, symbolstring, control_flags,
368 0458 1 compare_rtn, alloc_rtn, blockaddr, user_data) =
369 0459 2 BEGIN
370 0460 2
371 0461 2 This routine adds an entry to a balanced binary tree
372 0462 2
373 0463 2 INPUTS:
374 0464 2
375 0465 2     treehead      Address of listhead of tree
376 0466 2     symbolstring  Address of name of thing to insert (this is simply passed along to the compare
377 0467 2                  and allocation routines--the binary tree routines do not inspect this argume
378 0468 2     control_flags Address of control flags longword
379 0469 2                  Bit 0 (tree$v_ins_dup) set ==> insert duplicate entries
380 0470 2                  clear ==> return address of existing entry if duplicate
381 0471 2     compare_rtn  Address of routine to do compare. Called via CALLS with
382 0472 2                  three arguments. 4(AP)=Address of key, 8(AP)=Address of
383 0473 2                  current node to compare with, 12(AP) =user data argument
384 0474 2     alloc_rtn    Address of routine to allocate new node. Called via CALLS
385 0475 2                  with three arguments. 4(AP)=Address of key, 8(AP)=Address
386 0476 2                  of location to return allocated block address, 12(AP) = user_data argument
387 0477 2                  or 0 if none supplied
388 0478 2     user_data    OPTIONAL - Longword of context to pass to allocation
389 0479 2                  routine.
390 0480 2
391 0481 2 OUTPUTS:
392 0482 2
393 0483 2     blockaddr    Returns address of new (or found) entry
394 0484 2
395 0485 2 ROUTINE VALUE:
396 0486 2
397 0487 2     lib$normal   key inserted (success status)
398 0488 2     lib$keyalrns key found in tree, not inserted (alternate success status)
399 0489 2     lib$insvirmem allocation routine returned an error (error status)
400 0490 2
401 0491 2 MAP
402 0492 2     treehead : REF VECTOR[.LONG],
403 0493 2     control_flags : REF VECTOR[.LONG];
404 0494 2
405 0495 2 GLOBAL REGISTER
406 0496 2     currentnode = 11 : REF BLOCK[.BYTE],           !Pointer to current node
407 0497 2     keyname = 10 : REF VECTOR[.BYTE],             !Address of key to insert
408 0498 2     compare_routine = 9,                          !Address of caller's compare routine
409 0499 2     alloc_routine = 8,                            !Address of user's allocate node routine
410 0500 2     blockretadr = 7 : REF VECTOR[.LONG],         !Address to return result address
411 0501 2     foundintree = 6,                              !True if found in tree, false if inserted
412 0502 2     controlflags = 5 : BLOCK[4,.BYTE],          !Caller control flags
413 0503 2     user_context = 4;                             !Caller context
414 0504 2
415 0505 2 BUILTIN
416 0506 2     NULLPARAMETER;
417 0507 2
418 0508 2
419 0509 2     Initialize registers
420 0510 2
421 0511 2     currentnode = .treehead[0];                 !Get head of tree
422 0512 2     keyname = .symbolstring;                    !Point to key
    
```

```

: 423      0513 2 compare_routine = .compare_rtn;           !Set address of compare routine
: 424      0514 2 alloc_routine = .alloc_rtn;           !Set address of allocation routine
: 425      0515 2 blockretadr = .blockaddr;           !Set address to return result in
: 426      0516 2 foundintree = lib$normal;           !Assume will not find in tree
: 427      0517 2 controlflags = .control_flags[0];   !Pick up the caller's control flags
: 428      0518 2 IF NOT NULL[PARAMETER(7)]           !If user_data argument supplied
: 429      0519 2     THEN user_context = .user_data   !then use it
: 430      0520 2     ELSE user_context = 0;           ! otherwise default to 0
: 431      0521 2 ;
: 432      0522 2 ; Call recursive routine to insert node into tree
: 433      0523 2 ;
: 434      0524 2 insert_node();                       !Insert new node into tree
: 435      0525 2 ;
: 436      0526 2 IF .foundintree EQL lib$normal       !If we inserted the node
: 437      0527 2     THEN treehead[0] = .currentnode; ! then set new root of tree
: 438      0528 2 ;
: 439      0529 2 RETURN .foundintree                 !Return state of entry insert
: 440      0530 1 END;                                 ! End of routine LIB$INSERT_TREE
    
```

			OFFC 00000		.ENTRY LIB\$INSERT_TREE, Save R2,R3,R4,R5,R6,R7,R8,-;	0457
					R9,R10,R11	
52	00000000G	BF	D0	00002	MOVL #LIB\$NORMAL, R2	
5B	04	BC	D0	00009	MOVL @TREEHEAD, CURRENTNODE	0511
5A	08	AC	D0	0000D	MOVL SYMBOLSTRING, KEYNAME	0512
59	10	AC	D0	00011	MOVL COMPARE RTN, COMPARE ROUTINE	0513
58	14	AC	D0	00015	MOVL ALLOC RTN, ALLOC ROUTINE	0514
57	18	AC	D0	00019	MOVL BLOCKADDR, BLOCKRETADR	0515
56		52	D0	0001D	MOVL R2, FOUNDINTREE	0516
55	0C	BC	D0	00020	MOVL @CONTROL_FLAGS, CONTROLFLAGS	0517
07		6C	91	00024	CMPB (AP), #7	0518
		0B	1F	00027	BLSSU 1\$	
		1C	AC	D5	TSTL 28(AP)	
		06	13	0002C	BEQL 1\$	
54	1C	AC	D0	0002E	MOVL USER_DATA, USER_CONTEXT	0519
		02	11	00032	BRB 2\$	
		54	D4	00034	CLRL USER_CONTEXT	0520
		FE84	30	00036	BSBW INSERT NODE	0524
52		56	D1	00039	CMPB FOUNDINTREE, R2	0526
		04	12	0003C	BNEQ 3\$	
04	BC	5B	D0	0003E	MOVL CURRENTNODE, @TREEHEAD	0527
50		56	D0	00042	MOVL FOUNDINTREE, R0	0529
		04	00	00045	RET	0530

; Routine Size: 70 bytes, Routine Base: \_LIB\$CODE + 0143

```

442 0531 1 %SBTTL 'LIB$LOOKUP TREE - User entry to lookup in tree'
443 0532 1 GLOBAL ROUTINE lib$lookup_tree (treehead, symbolstring,
444 0533 1 compare_rtn, blockaddr) =
445 0534 2 BEGIN
446 0535 2
447 0536 2 This routine adds an entry to a balanced binary tree
448 0537 2
449 0538 2 INPUTS:
450 0539 2
451 0540 2     treehead      Address of listhead of tree
452 0541 2     symbolstring  Address of ASCII symbol name
453 0542 2     compare_rtn  Address of routine to do compare. Called via JSB with
454 0543 2                  two arguments. R0=Address of key, R1=Address of
455 0544 2                  current node to compare with.
456 0545 2
457 0546 2 OUTPUTS:
458 0547 2
459 0548 2     blockaddr     Returns address of new (or found) entry
460 0549 2
461 0550 2 ROUTINE VALUE:
462 0551 2
463 0552 2     lib$_normal   key found (success status)
464 0553 2     lib$_keynotfou key not found (error status)
465 0554 2
466 0555 2 MAP
467 0556 2     treehead : REF VECTOR[ LONG],
468 0557 2     blockaddr : REF VECTOR[ LONG];
469 0558 2
470 0559 2 LOCAL
471 0560 2     ch_result;
472 0561 2
473 0562 2 GLOBAL REGISTER
474 0563 2     currentnode = 11 : REF BLOCK[ BYTE],           !Address of current node
475 0564 2     keyname = 10 : REF VECTOR[ BYTE],             !Address of key
476 0565 2     compare_routine = 9;                          !Caller's compare routine
477 0566 2
478 0567 2
479 0568 2 Initialize registers
480 0569 2
481 0570 2 currentnode = .treehead[0];                       !Get head of tree
482 0571 2 keyname = .symbolstring;                          !Point at key to lookup
483 0572 2 compare_routine = .compare_rtn;                  !Get caller compare routine address
484 0573 2
485 0574 2 Search symbol table to see if already in table
486 0575 2
487 0576 2 WHILE .currentnode NEQ 0                          !While there is a node to examine
488 0577 2 DO IF (ch_result =                               !Compare key with current node
489 0578 2     (.compare_routine)(.keyname,.currentnode)) EQL 0
490 0579 2 THEN BEGIN
491 0580 2     Found -- return address to caller
492 0581 2
493 0582 2     blockaddr[0] = .currentnode;
494 0583 2     RETURN lib$_normal
495 0584 2 END
496 0585 2 ELSE IF .ch_result LSS 0                          !Test if left or right branch
497 0586 2 THEN currentnode = .currentnode[node$_left]
498 0587 2

```

```

: 499      0588 2      ELSE currentnode = .currentnode[node$_right];
: 500      0589 2
: 501      0590 2 RETURN lib$_keynotfou      !Return entry not found
: 502      0591 1 END;                      ! End of routine LIB$LOOKUP_TREE
  
```

			0E00 00000	.ENTRY	LIB\$LOOKUP_TREE, Save R9,R10,R11	: 0532
5B	04	BC	D0 00002	MOVL	@TREEHEAD, CURRENTNODE	: 0570
5A	08	AC	D0 00006	MOVL	SYMBOLSTRING, KEYNAME	: 0571
59	0C	AC	D0 0000A	MOVL	COMPARE_RTN, COMPARE_ROUTINE	: 0572
		5B	D5 0000E 1\$:	TSTL	CURRENTNODE	: 0576
		23	13 00010	BEQL	4\$	
7E		5A	7D 00012	MOVQ	KEYNAME, -(SP)	: 0578
69		02	FB 00015	CALLS	#2, (COMPARE_ROUTINE)	
		50	D5 00018	TSTL	CH_RESULT	
		0C	12 0001A	BNEQ	2\$	
10	BC	5B	D0 0001C	MOVL	CURRENTNODE, @BLOCKADDR	: 0583
	50	8F	D0 00020	MOVL	#LIB\$_NORMAL, R0	: 0584
		04	00027	RET		
		05	18 00028 2\$:	BGEQ	3\$	: 0586
5B		6B	D0 0002A	MOVL	(CURRENTNODE), CURRENTNODE	: 0587
		DF	11 0002D	BRB	1\$	
5B	04	AB	D0 0002F 3\$:	MOVL	4(CURRENTNODE), CURRENTNODE	: 0588
		D9	11 00033	BRB	1\$	: 0577
50	00000000G	8F	D0 00035 4\$:	MOVL	#LIB\$_KEYNOTFOU, R0	: 0590
		04	0003C	RET		: 0591

: Routine Size: 61 bytes, Routine Base: \_LIB\$CODE + 0189



```

504 0592 1 %SBTTL 'traverse_tree - Internal routine to traverse the tree'
505 ^593 1 ROUTINE traverse_tree (currentnode) : jsb_traverse =
506 0594 2 BEGIN
507 0595 2
508 0596 2 Internal routine to traverse the tree recursively
509 0597 2
510 0598 2 INPUTS:
511 0599 2
512 0600 2 currentnode Address of current node
513 0601 2
514 0602 2 MAP
515 0603 2 currentnode : REF BLOCK[,BYTE];
516 0604 2
517 0605 2 EXTERNAL REGISTER
518 0606 2 action_routine = 11, !Address of user routine to call
519 0607 2 user_context = 10; !User_data argument
520 0608 2
521 0609 2 LOCAL
522 0610 2 status,
523 0611 2 right_subtree : REF BLOCK[,BYTE];
524 0612 2
525 0613 2 IF .currentnode EQL 0 !If current node null
526 0614 2 THEN RETURN lib$_normal; ! then simply return
527 0615 2
528 0616 2 IF .currentnode[node$l_left] NEQ 0 !Process the left subtree if it exists
529 0617 2 THEN IF NOT (status = traverse_tree(.currentnode[node$l_left]))
530 0618 2 THEN RETURN .status;
531 0619 2
532 0620 2 right_subtree = .currentnode[node$l_right]; !Get pointer to right subtree
533 0621 2
534 0622 2 IF NOT (status = (.action_routine)(.currentnode, .user_context))!Call user routine for current node
535 0623 2 THEN RETURN .status;
536 0624 2
537 0625 2 IF .right_subtree NEQ 0 !Process the right subtree if it exists
538 0626 2 THEN IF NOT (status = traverse_tree(.right_subtree))
539 0627 2 THEN RETURN .status;
540 0628 2
541 0629 2 RETURN lib$_normal
542 0630 2
543 0631 1 END; !Of traverse_tree

```

	5E		04	C2 0000	TRAVERSE TREE:		
					SUBL2	#4, SP	: 0593
					PUSHL	R0	
					BEQL	3\$	: 0613
		00	BE	D5 0007	TSTL	@CURRENTNODE	: 0616
			0C	13 000A	BEQL	1\$	
	50	00	BE	D0 000C	MOVL	@CURRENTNODE, R0	: 0617
			EE	10 0010	BSBB	TRAVERSE TREE	
	51		50	D0 0012	MOVL	R0, STATUS	
	27		51	E9 0015	BLBC	STATUS, 2\$	
	50		6E	04 C1 0018	ADDL3	#4, CURRENTNODE, R0	: 0620
	04	AE	60	D0 001C	MOVL	(R0), RIGHT_SUBTREE	

LIBSBINARY\_TREE LIBSBINARY\_TREE - Subroutines to manipulate bal 16-Sep-1984 00:38:05  
 1-001 traverse\_tree - Internal routine to traverse th 14-Sep-1984 12:38:26

G 14

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

Page 16  
 (7)

		5A	DD	00020	PUSHL	USER_CONTEXT	:	0622
		AE	DD	00022	PUSHL	CURRENTNODE	:	
6B		02	FB	00025	CALLS	#2, (ACTION_ROUTINE)	:	
51		50	DO	00028	MOVL	RO, STATUS	:	
11		51	E9	0002B	BLBC	STATUS, 2\$	:	
	04	AE	D5	0002E	TSTL	RIGHT_SUBTREE	:	0625
		11	13	00031	BEQL	3\$	:	
50	04	AE	DO	00033	MOVL	RIGHT_SUBTREE, RO	:	0626
		C7	10	00037	BSBB	TRAVERSE_TREE	:	
51		50	DO	00039	MOVL	RO, STATUS	:	
05		51	E8	0003C	BLBS	STATUS, 3\$	:	
50		51	DO	0003F	MOVL	STATUS, RO	:	0627
		07	11	00042	BRB	4\$	:	
50	00000000G	8F	DO	00044	MOVL	#LIB\$ NORMAL, RO	:	0629
5E		08	CO	0004B	ADDL2	#8, SP	:	0631
		05	0004E	RSB			:	

: Routine Size: 79 bytes, Routine Base: \_LIB\$CODE + 01C6

```

: 545 0632 1 %SBTTL 'LIB$TRAVERSE_TREE - traverse binary tree'
: 546 0633 1 GLOBAL ROUTINE lib$traverse_tree (treehead, action_rtn, user_data) =
: 547 0634 2 BEGIN
: 548 0635 2
: 549 0636 2 This routine calls a routine for each node in the tree (in order)
: 550 0637 2
: 551 0638 2 INPUTS:
: 552 0639 2
: 553 0640 2 treehead Address of listhead of tree
: 554 0641 2 action_rtn Address of action routine to call
: 555 0642 2 Called with 2 args. 4(AP) = node address,
: 556 0643 2 8(AP) = user_data argument
: 557 0644 2 user_data OPTIONAL - longword of context passed to action routine
: 558 0645 2
: 559 0646 2 ROUTINE VALUE:
: 560 0647 2
: 561 0648 2 lib$normal Traversal completed successfully
: 562 0649 2 any other status Code returned by action routine
: 563 0650 2 --
: 564 0651 2
: 565 0652 2 MAP
: 566 0653 2 treehead : REF VECTOR[,LONG];
: 567 0654 2
: 568 0655 2 GLOBAL REGISTER
: 569 0656 2 action_routine = 11, !Keep address of routine in a register
: 570 0657 2 user_context = 10;
: 571 0658 2
: 572 0659 2 BUILTIN
: 573 0660 2 NULLPARAMETER;
: 574 0661 2
: 575 0662 2 action_routine = .action_rtn; !Set action routine address
: 576 0663 2 IF NULLPARAMETER(3)
: 577 0664 2 THEN user_context = 0 !Send a 0 if not specified
: 578 0665 2 ELSE user_context = .user_data; !Pick up user_data
: 579 0666 2
: 580 0667 2 RETURN traverse_tree (.treehead[0]) !Call routine to traverse the tree recursively
: 581 0668 2
: 582 0669 1 END; ! End of routine LIB$TRAVERSE_TREE

```

			0C00	0000	.ENTRY	LIB\$TRAVERSE_TREE, Save R10,R11	: 0633
5B	08	AC	D0	00002	MOVL	ACTION_RTN, ACTION_ROUTINE	: 0662
03		6C	91	00006	CMPB	(AP), #3	: 0663
		05	1F	00009	BLSSU	1\$	
		0C	AC	D5 0000B	TSTL	12(AP)	
			04	12 0000E	BNEQ	2\$	
			5A	D4 00010	CLRL	USER_CONTEXT	: 0664
			04	11 00012	BRB	3\$	
5A	0C	AC	D0	00014	MOVL	USER_DATA, USER_CONTEXT	: 0665
50	04	BC	D0	00018	MOVL	@TREEHEAD, R0	: 0667
		93	10	0001C	BSBB	TRAVERSE_TREE	
			04	0001E	RET		: 0669

; Routine Size: 31 bytes, Routine Base: \_LIB\$CODE + 0215

LIB\$BINARY\_TREE LIB\$BINARY\_TREE - Subroutines to manipulate bal  
1-001 LIB\$TRAVERSE\_TREE - traverse binary tree

1 14  
16-Sep-1984 00:38:05  
14-Sep-1984 12:38:26

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

```

: 584          0670 1 END                                ! End of module LIB$BINARY_TREE
: 585          0671 1
: 586          0672 0 ELUDOM
  
```

PSECT SUMMARY

Name	Bytes	Attributes
_LIB\$CODE	564	NOVEC,NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	0	0	581	00:00.8

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:LIBBINTRE/OBJ=OBJ\$:LIBBINTRE MSRCS\$:LIBBINTRE/UPDATE=(ENHS\$:LIBBINTRE)

```

: Size:          564 code + 0 data bytes
: Run Time:      00:10.8
: Elapsed Time: 00:48.8
: Lines/CPU Min: 3733
: Lexemes/CPU-Min: 26227
: Memory Used: 162 pages
: Compilation Complete
  
```

RTLMACB32 REQ	STRMACROS REQ	RTLOOBS REQ	RTLPSECT REQ	STRLNK REQ	RTLMAC MAR	LIBASCEBC LIS	LIBASTINP LIS	LIBBINTRE LIS	LIBCHAR LIS
					LIBDEF FOR	LIBANASTR LIS	LIBASNMBX LIS	LIBBBCCI LIS	
					LIBABUPCA LIS	LIBADDP LIS			
					SIGDEF FOR		LIBASCTIM LIS		
							LIBATTACH LIS		
					LIBTABMAC MAR			LIBBBSST LIS	LIBCALLG LIS
					LIBA2EREV LIS				LIBCLICAL LIS
						LIBADDP LIS			