

BBBBBBBBBBBB		AAAAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL


```
BBBBBBBBB      AAAAAA      SSSSSSSS      FFFFFFFF      IIIIII      NN      NN      DDDDDDDD
BBBBBBBBB      AAAAAA      SSSSSSSS      FFFFFFFF      IIIIII      NN      NN      DDDDDDDD
BB      BB      AA      AA      SS      FF      II      NN      NN      DD      DD
BB      BB      AA      AA      SS      FF      II      NN      NN      DD      DD
BB      BB      AA      AA      SS      FF      II      NNNN      NN      DD      DD
BBBBBBBBB      AA      AA      SSSSSS      FFFFFFFF      II      NN      NN      DD      DD
BBBBBBBBB      AA      AA      SSSSSS      FFFFFFFF      II      NN      NN      DD      DD
BB      BB      AAAAAAAAAA      SS      FF      II      NN      NN      DD      DD
BB      BB      AAAAAAAAAA      SS      FF      II      NN      NN      DD      DD
BB      BB      AA      AA      SS      FF      II      NN      NN      DD      DD
BB      BB      AA      AA      SS      FF      II      NN      NN      DD      DD
BBBBBBBBB      AA      AA      SSSSSSSS      FF      IIIIII      NN      NN      DDDDDDDD
BBBBBBBBB      AA      AA      SSSSSSSS      FF      IIIIII      NN      NN      DDDDDDDD

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 MODULE BASS$IND (
2 0002 0 IDENT = '1-009'
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1 ++
31 0031 1 FACILITY:
32 0032 1 Basic support library - user callable
33 0033 1
34 0034 1 ABSTRACT:
35 0035 1
36 0036 1 This module is the UPI level of the Basic FIND construct. Initially,
37 0037 1 it contains only the code for sequential I/O. This module will set
38 0038 1 up the I/O data base for the LUN and go directly to the REC level.
39 0039 1
40 0040 1
41 0041 1 ENVIRONMENT:
42 0042 1 User access mode - AST reentrant.
43 0043 1
44 0044 1 AUTHOR: Donald G. Petersen, CREATION DATE: 27-Feb-79
45 0045 1
46 0046 1 MODIFIED BY:
47 0047 1
48 0048 1 DGP, 27-Feb-79 : VERSION 01
49 0049 1 1-001 - original. DGP 27-Feb-79
50 0050 1 1-002 - Add BASS$IND_RECORD. DGP 02-Mar-79
51 0051 1 1-003 - More work on relative I/O. DGP 05-Mar-79
52 0052 1 1-004 - Add BASS$IND_KEY. DGP 06-Apr-79
53 0053 1 1-005 - Set up ISB$A_USER_FP. JBS 25-JUL-1979
54 0054 1 1-006 - Check for virtual use of this file; set block use. DGP 16-Oct-79
55 0055 1 1-007 - Add support for RFA access and manual record locking. PLL 1-Jun-82
56 0056 1 1-008 - RFAs are passed by ref. PLL 4-Jun-1982
57 0057 1 1-009 - allow REGARDLESS without manual record locking. MDL 14-Feb-1984
```

! Basic FIND construct
! File: BAS\$IND.B32 Edit: MDL1009

BAS\$FIND
1-009

M 3
16-Sep-1984 00:28:21
14-Sep-1984 11:54:58

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BAS\$FIND.B32;1

Page 2
(1)

: 58
: 59
: 60
0058 1 !--
0059 1
0060 1 !<BLF/PAGE>


```
62 0061 1 |
63 0062 1 | SWITCHES:
64 0063 1 |
65 0064 1 |
66 0065 1 | SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
67 0066 1 |
68 0067 1 |
69 0068 1 | LINKAGES
70 0069 1 |
71 0070 1 |
72 0071 1 | REQUIRE 'RTLIN:OTSLNK';           ! Define all linkages
73 0500 1 |
74 0501 1 |
75 0502 1 | TABLE OF CONTENTS:
76 0503 1 |
77 0504 1 |
78 0505 1 | FORWARD ROUTINE
79 0506 1 |     BASSFIND_KEY : NOVALUE,       ! UPI level Indexed FIND
80 0507 1 |     BASSFIND_RECORD : NOVALUE,    ! UPI level Relative FIND
81 0508 1 |     BASSFIND : NOVALUE,           ! UPI level Sequential FIND
82 0509 1 |     BASSFIND_RFA : NOVALUE;       ! UPI level RFA FIND
83 0510 1 |
84 0511 1 |
85 0512 1 | INCLUDE FILES:
86 0513 1 |
87 0514 1 |
88 0515 1 | REQUIRE 'RTLML:OTSISB';           ! ISB definitions
89 0683 1 |
90 0684 1 | REQUIRE 'RTLML:OTSLUB';           ! LUB definitions
91 0824 1 |
92 0825 1 | REQUIRE 'RTLIN:RTLPSECT';         ! Define DECLARE_PSECTS macro
93 0920 1 |
94 0921 1 | LIBRARY 'RTLSTARLE';             ! Starlet system macros
95 0922 1 |
96 0923 1 |
97 0924 1 | MACROS:
98 0925 1 |
99 0926 1 |     NONE
100 0927 1 |
101 0928 1 |
102 0929 1 | EQUATED SYMBOLS:
103 0930 1 |     NONE
104 0931 1 |
105 0932 1 |
106 0933 1 | PSECT DECLARATIONS:
107 0934 1 |
108 0935 1 | DECLARE_PSECTS (BAS);
109 0936 1 |
110 0937 1 | OWN STORAGE:
111 0938 1 |
112 0939 1 |     NONE
113 0940 1 |
114 0941 1 |
115 0942 1 | EXTERNAL REFERENCES:
116 0943 1 |
117 0944 1 |
118 0945 1 | EXTERNAL ROUTINE
```



```
: 119      0946 1  BAS$$STOP_IO : NOVALUE,          ! Signal fatal BASIC I/O error
: 120      0947 1  BAS$$REC_FIN : JSB_REC_IND1 NOVALUE, ! REC level - FIND indexed
: 121      0948 1  BAS$$REC_FRE : JSB_REC2 NOVALUE,    ! REC level - RMS interface FIND relative
: 122      0949 1  BAS$$REC_FSE : JSB_REC2 NOVALUE,    ! REC level processing - RMS interface
: 123      0950 1                                      ! FIND sequential
: 124      0951 1  BAS$$REC_FRFA : JSB_REC2 NOVALUE,    ! REC level - FIND by RFA
: 125      0952 1  BAS$$CB_PUSH : JSB_CB_PUSH NOVALUE, ! Load register CCB
: 126      0953 1  BAS$$CB_POP : JSB_CB_POP NOVALUE;    ! Done with register CCB
: 127      0954 1
: 128      0955 1  !+
: 129      0956 1  ! The following are the error codes used in this module.
: 130      0957 1  !-
: 131      0958 1
: 132      0959 1  EXTERNAL LITERAL
: 133      0960 1  BAS$K_ILLILLACC : UNSIGNED (8),      ! Illegal or illogical access
: 134      0961 1  BAS$K_IO_CHANOT : UNSIGNED (8);      ! I/O channel not open
: 135      0962 1  BAS$K_ILCRECLOC : UNSIGNED (8);      ! Illegal record locking clause
: 136      0963 1
```



```
138 0964 1 GLOBAL ROUTINE BAS$FIND (      ! FIND sequential
139 0965 1     UNIT,                      ! logical unit number
140 0966 1     LOCK_FLAGS                 ! manual locking bits
141 0967 1 ) : NOVALUE =
142 0968 1
143 0969 1 ++
144 0970 1 FUNCTIONAL DESCRIPTION:
145 0971 1
146 0972 1     This routine will set up the I/O data base for this LUN if necessary
147 0973 1     and then go directly to the REC level.  When control is returned to
148 0974 1     this routine, it pops the CCB off of the I/O system.  The actual inter-
149 0975 1     face to RMS is done at the REC level.  The next record is located.
150 0976 1
151 0977 1 FORMAL PARAMETERS:
152 0978 1
153 0979 1     UNIT.rlu.v      logical unit number
154 0980 1     [LOCK_FLAGS.rlu.v] if present, bits to pass on to the record level
155 0981 1                     to control manual record locking
156 0982 1
157 0983 1 IMPLICIT INPUTS:
158 0984 1
159 0985 1     LUB$V_VA_USE      virtual use of this file
160 0986 1
161 0987 1 IMPLICIT OUTPUTS:
162 0988 1
163 0989 1     ISB$B_STTM_TYPE  the statement type
164 0990 1     LUB$V_BLK_USE   non-virtual array use of this file
165 0991 1
166 0992 1 COMPLETION CODES:
167 0993 1
168 0994 1     NONE
169 0995 1
170 0996 1 SIDE EFFECTS:
171 0997 1
172 0998 1     Signals:
173 0999 1     BAS$K_IO_CHANOT (I/O channel not open)
174 1000 1     BAS$K_IL[ILLACC (illegal or illogical access)
175 1001 1
176 1002 1 --
177 1003 1
178 1004 2 BEGIN
179 1005 2
180 1006 2 BUILTIN
181 1007 2     FP,
182 1008 2     ACTUALCOUNT;
183 1009 2
184 1010 2 GLOBAL REGISTER
185 1011 2     CCB = K_CCB_REG : REF BLOCK [, BYTE];
186 1012 2
187 1013 2 LOCAL
188 1014 2     FMP : REF BLOCK [, BYTE],
189 1015 2     FLAGS;
190 1016 2
191 1017 2 LITERAL
192 1018 2     K_LOCK_ARG = 2;
193 1019 2
194 1020 2     FMP = .FP;
```



```
195 1021 2 +
196 1022 2 Allocate the LUB/ISB/RAB for this unit if necessary. Store new CB (con-
197 1023 2 trol block) in OTS$$A_CUR_LUB. Store signed unit number in LUB$W_LUN.
198 1024 2 -
199 1025 2 BASS$CB_PUSH (.UNIT, LUB$K_ILUN_MIN);
200 1026 2 CCB [ISB$A_USER_FP] = .FMP-[SF$[_SAVE_FP]];
201 1027 2 +
202 1028 2 If the channel is not open, give an error.
203 1029 2 FIND is not permitted on channel 0.
204 1030 2 -
205 1031 2
206 1032 2 IF ( NOT .CCB [LUB$V_OPENED]) THEN BASS$$STOP_IO (BAS$K_IO_CHANOT);
207 1033 2
208 1034 2 +
209 1035 2 Now that the data base is in place, store the statement type and go
210 1036 2 directly to the REC level.
211 1037 2 -
212 1038 2 CCB [ISB$B_STTM_TYPE] = ISB$K_ST_TY_FSE;
213 1039 2 +
214 1040 2 Check for virtual array usage and set block usage
215 1041 2 -
216 1042 2 IF .CCB [LUB$V_VA_USE] THEN BASS$$STOP_IO (BAS$K_ILLILLACC);
217 1043 2 CCB [LUB$V_BLK_USE] = 1;
218 1044 2
219 1045 2 IF ACTUALCOUNT () LSS K_LOCK_ARG
220 1046 2 THEN
221 1047 2     FLAGS = 0
222 1048 2 ELSE
223 1049 2     BEGIN
224 1050 2         +
225 1051 2         The ULK bit must set unless this is a REGARDLESS clause.
226 1052 2         -
227 1053 2         CASE .CCB [RAB$V_ULK] FROM 0 TO 1 OF
228 1054 2         SET
229 1055 2             [0]:
230 1056 2             IF (.LOCK_FLAGS AND RAB$M_RRL) NEQ 0
231 1057 2             THEN
232 1058 2                 FLAGS = .LOCK_FLAGS
233 1059 2             ELSE
234 1060 2                 BASS$$STOP_IO (BAS$K_ILLRECLOC);
235 1061 2
236 1062 2             [1]:
237 1063 2             FLAGS = .LOCK_FLAGS;
238 1064 2         TES;
239 1065 2         END;
240 1066 2
241 1067 2 BASS$REC_FSE (.FLAGS);
242 1068 2 +
243 1069 2 Now that the FIND has been done, pop the CCB off the I/O system.
244 1070 2 -
245 1071 2 BASS$CB_POP ();
246 1072 1 END;
```

!End of BASS\$FIND

.TITLE BASS\$FIND
.IDENT \1-009\

: 247 1073 1


```
249 1074 1 GLOBAL ROUTINE BAS$FIND_KEY (
250 1075 1     UNIT,
251 1076 1     KEY_NO,
252 1077 1     REL_OP,
253 1078 1     KEY,
254 1079 1     LOCK_FLAGS
255 1080 1 ) : NOVALUE =
256 1081 1
257 1082 1 ++
258 1083 1 FUNCTIONAL DESCRIPTION:
259 1084 1
260 1085 1     This routine will set up the I/O data base for this LUN if necessary
261 1086 1     and then go directly to the REC level. When control is returned to
262 1087 1     this routine, it pops the CCB off of the I/O system. The actual inter-
263 1088 1     face to RMS is done at the REC level. The next record is located
264 1089 1     based on the key of reference specified.
265 1090 1
266 1091 1 FORMAL PARAMETERS:
267 1092 1
268 1093 1     UNIT.rlu.v      logical unit number
269 1094 1     KEY_NO.rl.v     key of reference number
270 1095 1     REL_OP.rl.v     relational operator
271 1096 1     KEY.rt.dx       the key desired
272 1097 1     [LOCK_FLAGS.rlu.v] if present, specifies bits to pass on to record level
273 1098 1                   to control manual record locking
274 1099 1
275 1100 1 IMPLICIT INPUTS:
276 1101 1
277 1102 1     LUB$V_VA_USE      virtual array use of this file
278 1103 1
279 1104 1 IMPLICIT OUTPUTS:
280 1105 1
281 1106 1     ISB$B_STTM_TYPE   the statement type
282 1107 1     LUB$V_BLK_USE     non-virtual use of this file
283 1108 1
284 1109 1 COMPLETION CODES:
285 1110 1
286 1111 1     NONE
287 1112 1
288 1113 1 SIDE EFFECTS:
289 1114 1
290 1115 1     Signals:
291 1116 1     BAS$K_IO_CHANOT (I/O channel not open)
292 1117 1     BAS$K_ILCILLACC (Illegal or illogical access)
293 1118 1
294 1119 1 --
295 1120 1
296 1121 2 BEGIN
297 1122 2
298 1123 2 BUILTIN
299 1124 2     FP,
300 1125 2     ACTUALCOUNT;
301 1126 2
302 1127 2 GLOBAL REGISTER
303 1128 2     CCB = K_CCB_REG : REF BLOCK [, BYTE];
304 1129 2
305 1130 2 LOCAL
```



```
306      1131      2      FMP : REF BLOCK [, BYTE],
307      1132      2      FLAGS;
308      1133      2
309      1134      2      LITERAL
310      1135      2      K_LOCK_ARG = 5;
311      1136      2
312      1137      2      FMP = .FP;
313      1138      2
314      1139      2      + Allocate the LUB/ISB/RAB for this unit if necessary. Store new CB (con-
315      1140      2      trol block) in OTSS$A_CUR_LUB. Store signed unit number in LUB$W_LUN.
316      1141      2      -
317      1142      2      BAS$$CB PUSH (.UNIT, LUB$K_ILUN MIN);
318      1143      2      CCB [ISB$A_USER_FP] = .FMP-[SF$[_SAVE_FP];
319      1144      2      +
320      1145      2      - If the channel is not open, give an error.
321      1146      2
322      1147      2
323      1148      2      IF ( NOT .CCB [LUB$V_OPENED]) THEN BAS$$STOP_IO (BAS$K_IO_CHANOT);
324      1149      2
325      1150      2      +
326      1151      2      - Now that the data base is in place, store the statement type and go
327      1152      2      directly to the REC level.
328      1153      2
329      1154      2      CCB [ISB$B_STTM_TYPE] = ISB$K_ST_TY_FIN;
330      1155      2      +
331      1156      2      - Check for virtual array usage and set block usage
332      1157      2
333      1158      2      IF .CCB [LUB$V_VA_USE] THEN BAS$$STOP_IO(BAS$K_ILLILLACC);
334      1159      2      CCB [LUB$V_BLK_USE] = 1;
335      1160      2
336      1161      2      IF ACTUALCOUNT () LSS K_LOCK_ARG
337      1162      2      THEN
338      1163      2          FLAGS = 0
339      1164      2      ELSE
340      1165      2          BEGIN
341      1166      2              +
342      1167      2              - The ULK bit must set unless this is a REGARDLESS clause.
343      1168      2              -
344      1169      2              CASE .CCB [RAB$V_ULK] FROM 0 TO 1 OF
345      1170      2              SET
346      1171      2                  [0]:
347      1172      2                      IF (.LOCK_FLAGS AND RAB$M_RRL) NEQ 0
348      1173      2                      THEN
349      1174      2                          FLAGS = .LOCK_FLAGS
350      1175      2                      ELSE
351      1176      2                          BAS$$STOP_IO (BAS$K_ILLRECLOC);
352      1177      2
353      1178      2                  [1]:
354      1179      2                      FLAGS = .LOCK_FLAGS;
355      1180      2          TES;
356      1181      2          END;
357      1182      2
358      1183      2      BAS$$REC_FIN (.KEY_NO, .REL_OP, .KEY, .FLAGS);
359      1184      2      +
360      1185      2      - Now that the FIND has been done, pop the CCB off the I/O system.
361      1186      2
362      1187      2      BAS$$CB_POP ();
```


: 363 1188 1 END;

!End of BASSFIND_KEY

				083C 00000	.ENTRY	BASSFIND KEY, Save R2,R3,R4,R5,R11	: 1074
	54	00000000G	00	9E 00002	MOVAB	BASS\$STOP_IO, R4	:
	53		5D	D0 00009	MOVL	FP, FMP	: 1137
	50		08	CE 0000C	MNEGL	#8, R0	: 1142
	52	04	AC	D0 0000F	MOVL	UNIT, R2	:
		00000000G	00	16 00013	JSB	BASS\$CB_PUSH	:
FF4C	CB	0C	A3	D0 00019	MOVL	12(FMP), -180(CCB)	: 1143
	07	FC	AB	E8 0001F	BLBS	-4(CCB), 1\$	: 1148
	7E	00G	8F	9A 00023	MOVZBL	#BASSK_IO CHANOT, -(SP)	:
	64		01	FB 00027	CALLS	#1, BASS\$STOP_IO	:
FF71	CB		2F	90 0002A 1\$:	MOVB	#47, -143(CCB)	: 1154
	07	FF	AB	E9 0002F	BLBC	-1(CCB), 2\$	: 1158
	7E	00G	8F	9A 00033	MOVZBL	#BASSK_ILLLACC, -(SP)	:
	64		01	FB 00037	CALLS	#1, BASS\$STOP_IO	:
FF	AB		02	88 0003A 2\$:	BISB2	#2, -1(CCB)	: 1159
	05		6C	91 0003E	CMPB	(AP), #5	: 1161
			04	1E 00041	BGEQU	3\$	:
			53	D4 00043	CLRL	FLAGS	: 1163
			20	11 00045	BRB	7\$	:
52	06	AB	01	02 EF 00047 3\$:	EXTZV	#2, #1, 6(CCB), R2	: 1169
		01	00	52 CF 0004D	CASEL	R2, #0, #1	:
		0012	0004	00051 4\$:	.WORD	5\$-4\$,-	:
						6\$-4\$	:
			03	E0 00055 5\$:	BBS	#3, LOCK FLAGS, 6\$	: 1172
	09	14	AC	8F 9A 0005A	MOVZBL	#BASSK_ILRELOC, -(SP)	: 1176
			7E	01 FB 0005E	CALLS	#1, BASS\$STOP_IO	:
			64	04 11 00061	BRB	7\$	: 1172
			53	14 AC D0 00063 6\$:	MOVL	LOCK FLAGS, FLAGS	: 1179
			51	0C AC 7D 00067 7\$:	MOVQ	REL_OP, R1	: 1183
			50	08 AC D0 0006B	MOVL	KEY_NO, R0	:
		00000000G	00	16 0006F	JSB	BASS\$REC_FIN	:
		00000000G	00	16 00075	JSB	BASS\$CB_POP	: 1187
			04	0007B	RET		: 1188

; Routine Size: 124 bytes, Routine Base: _BAS\$CODE + 0077

: 364 1189 1


```
366 1190 1 GLOBAL ROUTINE BASFIND_RECORD (
367 1191 1     UNIT,
368 1192 1     RECORD_NUM,
369 1193 1     LOCK_FLAGS
370 1194 1 ) : NOVA[UE =
371 1195 1
372 1196 1
373 1197 1 ++
374 1198 1 FUNCTIONAL DESCRIPTION:
375 1199 1     This routine will set up the I/O data base for this LUN if necessary
376 1200 1     and then go directly to the REC level. When control is returned to
377 1201 1     this routine, it pops the CCB off of the I/O system. The actual inter-
378 1202 1     face to RMS is done at the REC level. The next record is located.
379 1203 1
380 1204 1 FORMAL PARAMETERS:
381 1205 1
382 1206 1     UNIT.rlu.v      logical unit number
383 1207 1     RECORD_NUM.rl.v  relative record number
384 1208 1     [LOCK_FLAGS.rlu.v] if present, specifies bits to pass on to record
385 1209 1                     level to control manual record locking
386 1210 1
387 1211 1 IMPLICIT INPUTS:
388 1212 1
389 1213 1     LUB$V_VA_USE      virtual array use of this file
390 1214 1
391 1215 1 IMPLICIT OUTPUTS:
392 1216 1
393 1217 1     ISB$B_STTM_TYPE   the statement type
394 1218 1     LUB$V_BLK_USE     non-virtual array use of this file
395 1219 1
396 1220 1 COMPLETION CODES:
397 1221 1
398 1222 1     NONE
399 1223 1
400 1224 1 SIDE EFFECTS:
401 1225 1
402 1226 1     Signals:
403 1227 1     BAS$K_IO_CHANOT (I/O channel not open)
404 1228 1     BAS$K_ILCILLACC (Illegal or illogical access)
405 1229 1
406 1230 1 --
407 1231 1
408 1232 2 BEGIN
409 1233 2
410 1234 2 BUILTIN
411 1235 2     FP,
412 1236 2     ACTUALCOUNT;
413 1237 2
414 1238 2 GLOBAL REGISTER
415 1239 2     CCB = K_CCB_REG : REF BLOCK [, BYTE];
416 1240 2
417 1241 2 LOCAL
418 1242 2     FMP : REF BLOCK [, BYTE],
419 1243 2     FLAGS;
420 1244 2
421 1245 2 LITERAL
422 1246 2     K_LOCK_ARG = 3;
```



```

423 1247 2
424 1248 2 FMP = .FP;
425 1249 2
426 1250 2 + Allocate the LUB/ISB/RAB for this unit if necessary. Store new CB (con-
427 1251 2 - trol block) in OTSS$A_CUR_LUB. Store signed unit number in LUB$W_LUN.
428 1252 2
429 1253 2 BAS$$CB_PUSH (.UNIT, LUB$K_ILUN_MIN);
430 1254 2 CCB [ISB$A_USER_FP] = .FMP-[SF$C_SAVE_FP];
431 1255 2
432 1256 2 + Give an error if the channel is not open.
433 1257 2
434 1258 2
435 1259 2 IF ( NOT .CCB [LUB$V_OPENED]) THEN BAS$$STOP_IO (BAS$K_IO_CHANOT);
436 1260 2
437 1261 2 +
438 1262 2 - Now that the data base is in place, store the statement type, store the index, and go
439 1263 2 directly to the REC level.
440 1264 2
441 1265 2 CCB [LUB$L_LOG_RECNO] = .RECORD_NUM;
442 1266 2 CCB [ISB$B_STTM_TYPE] = ISB$K_ST_TY_FRE;
443 1267 2
444 1268 2 + Check for virtual array usage and set block usage
445 1269 2
446 1270 2 IF .CCB [LUB$V_VA_USE] THEN BAS$$STOP_IO (BAS$K_ILLILLACC);
447 1271 2 CCB [LUB$V_BLK_USE] = 1;
448 1272 2
449 1273 2 IF ACTUALCOUNT () LSS K_LOCK_ARG
450 1274 2 THEN
451 1275 2     FLAGS = 0
452 1276 2 ELSE
453 1277 2     BEGIN
454 1278 2         IF (.CCB [RAB$L_ROP] AND RAB$M_ULK) EQL 0
455 1279 2         THEN
456 1280 2             BAS$$STOP_IO (BAS$K_ILLRECLOC)
457 1281 2         ELSE
458 1282 2             FLAGS = .LOCK_FLAGS;
459 1283 2         END;
460 1284 2 BAS$$REC_FRE (.FLAGS);
461 1285 2 +
462 1286 2 - Now that the FIND has been done, pop the CCB off the I/O system.
463 1287 2
464 1288 2 BAS$$CB_POP ();
465 1289 1 END;

```

!End of BAS\$FIND_RECORD

			083C 00000	.ENTRY	BAS\$FIND_RECORD, Save R2,R3,R4,R5,R11	: 1190
54	00000000G	00	9E 00002	MOVAB	BAS\$\$STOP_IO, R4	: 1248
53		5D	D0 00009	MOVL	FP, FMP	: 1253
50		08	CE 0000C	MNEGL	#8, R0	
52	04	AC	D0 0000F	MOVL	UNIT, R2	
	00000000G	00	16 00013	JSB	BAS\$\$CB_PUSH	
FF4C	CB	0C	A3 D0 00019	MOVL	12(FMP), -180(CCB)	: 1254
	07	FC	AB E8 0001F	BLBS	-4(CCB), 1\$	: 1259
	7E	00G	8F 9A 00023	MOVZBL	#BAS\$K_IO_CHANOT, -(SP)	

	64	01	FB	00027	CALLS	#1, BASS\$STOP_IO	:	1265
EO	AB	08	AC	D0 0002A	1\$:	MOVL	RECORD_NUM, -32(CCB)	:
FF71	CB	29	90	0002F		MOVB	#41, -T43(CCB)	:
	07	FF	AB	E9 00034		BLBC	-1(CCB), 2\$	:
	7E	00G	8F	9A 00038		MOVZBL	#BASSK_ILLILLACC, -(SP)	:
	64		01	FB 0003C		CALLS	#1, BASS\$STOP_IO	:
FF	AB		02	88 0003F	2\$:	BISB2	#2, -1(CCB)	:
	03		6C	91 00043		CMPB	(AP), #3	:
			04	1E 00046		BGEQU	3\$	:
			52	D4 00048		CLRL	FLAGS	:
			11	11 0004A		BRB	5\$	:
09	68		32	E0 0004C	3\$:	BBS	#50, (CCB), 4\$	:
	7E	00G	8F	9A 00050		MOVZBL	#BASSK_ILLRECLOC, -(SP)	:
	64		01	FB 00054		CALLS	#1, BASS\$STOP_IO	:
			04	11 00057		BRB	5\$	:
	52	0C	AC	D0 00059	4\$:	MOVL	LOCK_FLAGS, FLAGS	:
	50		52	D0 0005D	5\$:	MOVL	FLAGS, R0	:
			00	16 00060		JSB	BASS\$REC_FRE	:
			00	16 00066		JSB	BASS\$CB_POP	:
			04	0006C		RET		:

; Routine Size: 109 bytes, Routine Base: _BASS\$CODE + 00F3


```

467 1290 1 GLOBAL ROUTINE BAS$FIND_RFA (
468 1291 1     UNIT,
469 1292 1     RFA,
470 1293 1     LOCK_FLAGS
471 1294 1 ) : NOVA$UE =
472 1295 1
473 1296 1
474 1297 1 ++
475 1298 1 FUNCTIONAL DESCRIPTION:
476 1299 1     This routine will set up the I/O data base for this LUN if necessary
477 1300 1     and then go directly to the REC level. When control is returned to
478 1301 1     this routine, it pops the CCB off of the I/O system. The actual inter-
479 1302 1     face to RMS is done at the REC level. The record specified by the
480 1303 1     RFA is located.
481 1304 1
482 1305 1 FORMAL PARAMETERS:
483 1306 1
484 1307 1     UNIT.rlu.v      logical unit number
485 1308 1     RFA.rx.r        RFA address
486 1309 1     [LOCK_FLAGS.rlu.v] if present, specifies bits to pass on to record
487 1310 1                     level to control manual record locking
488 1311 1
489 1312 1 IMPLICIT INPUTS:
490 1313 1
491 1314 1     LUB$V_VA_USE      virtual array use of this file
492 1315 1
493 1316 1 IMPLICIT OUTPUTS:
494 1317 1
495 1318 1     ISB$B_STTM_TYPE   the statement type
496 1319 1     LUB$V_BLK_USE     non-virtual array use of this file
497 1320 1
498 1321 1 COMPLETION CODES:
499 1322 1
500 1323 1     NONE
501 1324 1
502 1325 1 SIDE EFFECTS:
503 1326 1
504 1327 1     Signals:
505 1328 1     BAS$K_IO_CHANOT (I/O channel not open)
506 1329 1     BAS$K_IL$ILLACC (Illegal or illogical access)
507 1330 1
508 1331 1 --
509 1332 1
510 1333 2 BEGIN
511 1334 2
512 1335 2 BUILTIN
513 1336 2     FP,
514 1337 2     ACTUALCOUNT;
515 1338 2
516 1339 2 GLOBAL REGISTER
517 1340 2     CCB = K_CCB_REG : REF BLOCK [, BYTE];
518 1341 2
519 1342 2 LOCAL
520 1343 2     FMP : REF BLOCK [, BYTE],
521 1344 2     FLAGS;
522 1345 2
523 1346 2 LITERAL
```



```

524      K_LOCK_ARG = 3;
525
526      FMP = .FMP;
527
528      + Allocate the LUB/ISB/RAB for this unit if necessary. Store new CB (con-
529      + trol block) in OTSS$A_CUR_LUB. Store signed unit number in LUB$W_LUN.
530
531      BAS$$CB_PUSH (.UNIT, LUB$K_ILUN MIN);
532      CCB [ISB$A_USER_FP] = .FMP-[SF$C_SAVE_FP];
533
534      + Give an error if the channel is not open.
535
536      IF ( NOT .CCB [LUB$V_OPENED]) THEN BAS$$STOP_IO (BAS$K_IO_CHANOT);
537
538      + Now that the data base is in place, store the statement type, store the RFA, and go
539      + directly to the REC level.
540
541      CH$MOVE (6, .RFA, CCB [RAB$W_RFA]);
542      CCB [ISB$B_STTM_TYPE] = ISB$R_ST_TY_FRFA;
543
544      + Check for virtual array usage and set block usage
545
546      IF .CCB [LUB$V_VA_USE] THEN BAS$$STOP_IO (BAS$K_ILLILLACC);
547      CCB [LUB$V_BLK_USE] = 1;
548
549      IF ACTUALCOUNT () LSS K_LOCK_ARG
550      THEN
551          FLAGS = 0
552      ELSE
553          BEGIN
554              IF (.CCB [RAB$L_ROP] AND RAB$M_ULK) EQL 0
555              THEN
556                  BAS$$STOP_IO (BAS$K_ILLRECLOC)
557              ELSE
558                  FLAGS = .LOCK_FLAGS;
559          END;
560      BAS$REC_FRFA (.FLAGS);
561
562      + Now that the FIND has been done, pop the CCB off the I/O system.
563
564      BAS$$CB_POP ();
565      END;
566
567      !End of BAS$FIND_RFA

```

		087C 00000	.ENTRY	BAS\$FIND_RFA, Save R2,R3,R4,R5,R6,R11	: 1290
56	00000000G	00 9E 00002	MOVAB	BAS\$\$STOP_IO, R6	: 1349
53		5D D0 00009	MOVL	FP, FMP	: 1354
50		08 CE 0000C	MNEGL	#8, R0	
52	04	AC D0 0000F	MOVL	UNIT, R2	
	00000000G	00 16 00013	JSB	BAS\$\$CB_PUSH	
FF4C	CB	0C A3 D0 00019	MOVL	12(FMP), -180(CCB)	: 1355
07	FC	AB E8 0001F	BLBS	-4(CCB), 1\$	: 1360

10	AB	08	7E	00G	8F	9A	00023	MOVZBL	#BASSK IO CHANOT, -(SP)	
		FF71	66		01	FB	00027	CALLS	#1, BASS\$STOP_IO	1366
			BC		06	28	0002A	MOV C3	#6, @RFA, 16(CCB)	1367
			CB		38	90	00030	MOVB	#56, -143(CCB)	1371
			07	FF	AB	E9	00035	BLBC	-1(CCB), 2\$	
			7E	00G	8F	9A	00039	MOVZBL	#BASSK ILLILLACC, -(SP)	
		FF	66		01	FB	0003D	CALLS	#1, BASS\$STOP_IO	
			AB		02	88	00040	BISB2	#2, -1(CCB)	1372
			03		6C	91	00044	CMPB	(AP), #3	1374
					04	1E	00047	BGEQU	3\$	
					52	D4	00049	CLRL	FLAGS	1376
					11	11	0004B	BRB	5\$	
09			6B		32	E0	0004D	BBS	#50, (CCB), 4\$	1379
			7E	00G	8F	9A	00051	MOVZBL	#BASSK ILLRECLOC, -(SP)	1381
			66		01	FB	00055	CALLS	#1, BASS\$STOP_IO	
					04	11	00058	BRB	5\$	
			52	0C	AC	D0	0005A	MOVL	LOCK_FLAGS, FLAGS	1383
			50		52	D0	0005E	MOVL	FLAGS, R0	1385
				00000000G	00	16	00061	JSB	BASS\$REC_FRFA	
				00000000G	00	16	00067	JSB	BASS\$CB_POP	1389
					04	0006D		RET		1390

; Routine Size: 110 bytes, Routine Base: _BASS\$CODE + 0160

:	568	1391	1
:	569	1392	1
:	570	1393	1 END
:	571	1394	1
:	572	1395	0 ELUDOM

!End of module - BASS\$FIND

PSECT SUMMARY

Name	Bytes	Attributes
_BASS\$CODE	462	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	6	0	581	00:01.1

BASSFIND
1-009

B 5
16-Sep-1984 00:28:21
14-Sep-1984 11:54:58

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASFIND.B32;1

Page 17
(6)

COMMAND QUALIFIERS

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

: Size: 462 code + 0 data bytes
: Run Time: 00:14.7
: Elapsed Time: 00:32.0
: Lines/CPU Min: 5709
: Lexemes/CPU-Min: 32476
: Memory Used: 129 pages
: Compilation Complete

0023 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

