

```

BBBBBBBBBBBBBB      AAAAAAAAAA      SSSSSSSSSSSS      RRRRRRRRRRRR      TTTTTTTTTTTTTT      LLL
BBBBBBBBBBBBBB      AAAAAAAAAA      SSSSSSSSSSSS      RRRRRRRRRRRR      TTTTTTTTTTTTTT      LLL
BBBBBBBBBBBBBB      AAAAAAAAAA      SSSSSSSSSSSS      RRRRRRRRRRRR      TTTTTTTTTTTTTT      LLL
BBB      BBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      LLL
BBB      BBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      LLL
BBB      BBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      LLL
BBB      BBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      LLL
BBB      BBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      LLL
BBB      BBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      LLL
BBB      BBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      LLL
BBBBBBBBBBBBBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      LLL
BBBBBBBBBBBBBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      LLL
BBBBBBBBBBBBBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      LLL
BBB      BBB      AAAAAAAAAAAAAA      SSS      RRR      RRR      TTT      LLL
BBB      BBB      AAAAAAAAAAAAAA      SSS      RRR      RRR      TTT      LLL
BBB      BBB      AAAAAAAAAAAAAA      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
BBBBBBBBBBBBBB      AAA      AAA      SSSSSSSSSSSS      RRR      RRR      TTT      LLLLLLLLLLLLLLLL
BBBBBBBBBBBBBB      AAA      AAA      SSSSSSSSSSSS      RRR      RRR      TTT      LLLLLLLLLLLLLLLL
BBBBBBB6BBBBBB      AAA      AAA      SSSSSSSSSSSS      RRR      RRR      TTT      LLLLLLLLLLLLLLLL

```

```

BBBBBBBB      AAAAAA      SSSSSSSS      CCCCCCCC      MM      MM      PPPPPPPP      AAAAAA      PPPPPPPP      PPPPPPPP
BBBBBBBB      AAAAAA      SSSSSSSS      CCCCCCCC      MM      MM      PPPPPPPP      AAAAAA      PPPPPPPP      PPPPPPPP
BB      BB      AA      AA      SS      CC      MMMM      MMMM      PP      PP      AA      AA      PP      PP      PP      PP
BB      BB      AA      AA      SS      CC      MMMM      MMMM      PP      PP      AA      AA      PP      PP      PP      PP
BB      BB      AA      AA      SS      CC      MM      MM      PP      PP      AA      AA      PP      PP      PP      PP
BBBBBBBB      AA      AA      SSSSSS      CC      MM      MM      PPPPPPPP      AA      AA      PPPPPPPP      PPPPPPPP
BBBBBBBB      AA      AA      SSSSSS      CC      MM      MM      PPPPPPPP      AA      AA      PPPPPPPP      PPPPPPPP
BB      BB      AAAAAAAAAA      SS      CC      MM      MM      PP      AAAAAAAAAA      PP      PP
BB      BB      AAAAAAAAAA      SS      CC      MM      MM      PP      AAAAAAAAAA      PP      PP
BB      BB      AA      AA      SS      CC      MM      MM      PP      AA      AA      PP      PP
BB      BB      AA      AA      SS      CC      MM      MM      PP      AA      AA      PP      PP
BBBBBBBB      AA      AA      SSSSSSSS      CCCCCCCC      MM      MM      PP      AA      AA      PP      PP
BBBBBBBB      AA      AA      SSSSSSSS      CCCCCCCC      MM      MM      PP      AA      AA      PP      PP

```

```

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 BAS$CMP_APPROX (
2 0002 0 IDENT = '1-006'
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: BASIC-PLUS-2 Miscellaneous
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1 This module implements the BASIC approximate compare function,
36 0036 1 which indicates whether or not its two arguments will print the same.
37 0037 1
38 0038 1 ENVIRONMENT: VAX-11 User Mode
39 0039 1
40 0040 1 AUTHOR: John Sauter, CREATION DATE: 01-MAR-1979
41 0041 1
42 0042 1 MODIFIED BY:
43 0043 1
44 0044 1 1-001 - Original. JBS 01-MAR-1979
45 0045 1 1-002 - Change from JSB to CALL entry points. JBS 21-MAR-1979
46 0046 1 1-003 - Reverse the result definition, so TSI BEQL will work. JBS 25-MAR-1979
47 0047 1 1-004 - Use the new conversion routines. JBS 11-JUL-1979
48 0048 1 1-005 - Add entry points for g and h floating. PLL 26-Oct-81
49 0049 1 1-006 - Fix calls to conversion routines in g and h routines.
50 0050 1 --
51 0051 1
52 0052 1 !<BLF/PAGE>
    
```

! File: BAS\$CMPAPP.B32 Edit: PLL1607

```

54      0053 1 |
55      0054 1 | SWITCHES:
56      0055 1 |
57      0056 1 |
58      0057 1 | SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
59      0058 1 |
60      0059 1 |
61      0060 1 | LINKAGES:
62      0061 1 |
63      0062 1 |     NONE
64      0063 1 |
65      0064 1 | TABLE OF CONTENTS:
66      0065 1 |
67      0066 1 |
68      0067 1 | FORWARD ROUTINE
69      0068 1 |     BAS$CMPF_APP,      ! Compare floating values
70      0069 1 |     BAS$CMPD_APP,      ! Compare double values
71      0070 1 |     BAS$CMPI_APP,      ! Compare g floating values
72      0071 1 |     BAS$CMPI_APP,      ! Compare h floating values
73      0072 1 |
74      0073 1 |
75      0074 1 | INCLUDE FILES:
76      0075 1 |
77      0076 1 |
78      0077 1 | REQUIRE 'RTLIN:RTLPSECT';      ! Macros for defining psects
79      0172 1 |
80      0173 1 | LIBRARY 'RTLSTARLE';      ! System definitions
81      0174 1 |
82      0175 1 |
83      0176 1 | MACROS:
84      0177 1 |
85      0178 1 |     NONE
86      0179 1 |
87      0180 1 | EQUATED SYMBOLS:
88      0181 1 |
89      0182 1 |     NONE
90      0183 1 |
91      0184 1 | PSECTS:
92      0185 1 |
93      0186 1 | DECLARE_PSECTS (BAS);      ! Declare psects for BAS$ facility
94      0187 1 |
95      0188 1 | OWN STORAGE:
96      0189 1 |
97      0190 1 |     NONE
98      0191 1 |
99      0192 1 | EXTERNAL REFERENCES:
100     0193 1 |
101     0194 1 |
102     0195 1 | EXTERNAL ROUTINE
103     0196 1 |     BAS$CVT_OUT_D_G,    ! Convert double to text
104     0197 1 |     BAS$CVT_OUT_G_G,    ! Convert gfloat to text
105     0198 1 |     BAS$CVT_OUT_H_G,    ! Convert hfloat to text
106     0199 1 |     BAS$$STOP :~NOVALUE; ! signals fatal error
107     0200 1 |
108     0201 1 |
109     0202 1 | The following are the error codes used in this module.
110     0203 1 |

```

BAS\$CMP_APPROX
1-006

N 7
16-Sep-1984 00:08:28 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 11:54:47 [BASRTL.SRC]BAS\$CMPAPP.B32;1

```
: 111      0204 1  
: 112      0205 1 EXTERNAL LITERAL  
: 113      0206 1   BAS$K_PROLOSSOR;  
: 114      0207 1
```

. Program lost, sorry

```

116 0208 1 GLOBAL ROUTINE BASSCMPF_APP (           ! Compare floating values
117 0209 1     VAL1,                               ! First value to compare
118 0210 1     VAL2                               ! Second value to compare
119 0211 1     ) =
120 0212 1
121 0213 1
122 0214 1  **
123 0215 1  FUNCTIONAL DESCRIPTION:
124 0216 1  Compares two floating values.  If the values will PRINT the same,
125 0217 1  they are considered "approximately equal".  This function is useful
126 0218 1  in Computer Assisted Education applications, when one value is
127 0219 1  computed and another is read from the terminal.  If the two values
128 0220 1  print the same we do not want to say: "Wrong! You typed x, correct
129 0221 1  answer is x."
130 0222 1
131 0223 1  CALLING SEQUENCE:
132 0224 1
133 0225 1     PUSHL   VAL2
134 0226 1     PUSHL   VAL1
135 0227 1     CALLS  #2, BASSCMPF_APP
136 0228 1     TSTL    R0
137 0229 1     BEQL   1$                               ; Branch if VAL1 == VAL2
138 0230 1
139 0231 1  FORMAL PARAMETERS:
140 0232 1
141 0233 1     VAL1.rf.v      The first value to compare
142 0234 1     VAL2.rf.v      The second value to compare
143 0235 1
144 0236 1  IMPLICIT INPUTS:
145 0237 1
146 0238 1     NONE
147 0239 1
148 0240 1  IMPLICIT OUTPUTS:
149 0241 1
150 0242 1     NONE
151 0243 1
152 0244 1  ROUTINE VALUE:
153 0245 1  COMPLETION CODES:
154 0246 1
155 0247 1     1 = not approximately equal,
156 0248 1     0 = approximately equal.
157 0249 1
158 0250 1  SIDE EFFECTS:
159 0251 1
160 0252 1     NONE
161 0253 1
162 0254 1  --
163 0255 1
164 0256 2  BEGIN
165 0257 2
166 0258 2  BUILTIN
167 0259 2  CVTFD:
168 0260 2
169 0261 2  LOCAL
170 0262 2  D VALUE : VECTOR [2],
171 0263 2  DSC1 : BLOCK [8, BYTÉ],
172 0264 2  BUF1 : VECTOR [14, BYTÉ],

```

```

173 0265 2          DSC2 : BLOCK [8, BYTE],
174 0266 2          BUF2 : VECTOR [14, BYTE],
175 0267 2          RET_LENGTH;
176 0268 2
177 0269 2
178 0270 2  !+ Call the formatter for each value to get two character strings.
179 0271 2  !-
180 0272 2          DSC1 [DSC$W_LENGTH] = 14;
181 0273 2          DSC1 [DSC$B_DTYPE] = DSC$K_DTYPE_T;
182 0274 2          DSC1 [DSC$B_CLASS] = DSC$K_CLASS_S;
183 0275 2          DSC1 [DSC$A_POINTER] = BUF1;
184 0276 2          CVTFD (VAL1, D VALUE [0]);
185 0277 2          BASSCVT_OUT_D_G (D VALUE, 0, RET_LENGTH, DSC1, 0);
186 0278 2          DSC1 [DSC$W_LENGTH] = .RET_LENGTH;
187 0279 2
188 0280 2          DSC2 [DSC$W_LENGTH] = 14;
189 0281 2          DSC2 [DSC$B_DTYPE] = DSC$K_DTYPE_T;
190 0282 2          DSC2 [DSC$B_CLASS] = DSC$K_CLASS_S;
191 0283 2          DSC2 [DSC$A_POINTER] = BUF2;
192 0284 2          CVTFD (VAL2, D VALUE [0]);
193 0285 2          BASSCVT_OUT_D_G (D VALUE, 0, RET_LENGTH, DSC2, 0);
194 0286 2          DSC2 [DSC$W_LENGTH] = .RET_LENGTH;
195 0287 2
196 0288 2  !+ If the two strings are equal, the values are approximately equal.
197 0289 2  !-
198 0290 2
199 0291 2          IF (.DSC1 [DSC$W_LENGTH] NEQU .DSC2 [DSC$W_LENGTH]) THEN RETURN (1);
200 0292 2
201 0293 2          IF (CH$EQL (.RET_LENGTH, .DSC1 [DSC$A_POINTER], .RET_LENGTH, .DSC2 [DSC$A_POINTER])) THEN RETURN (0);
202 0294 2
203 0295 2          RETURN (1);
204 0296 1          END;

```

! end of BASSCMPF_APP

```

.TITLE BASSCMP_APPROX
.IDENT \1-006\

.EXTRN BASSCVT_OUT_D_G
.EXTRN BASSCVT_OUT_G_G
.EXTRN BASSCVT_OUT_H_G
.EXTRN BASS$STOP, BASS$K_PROLOSSOR

.PSECT _BASSCODE, NOWRT, SHR, PIC, 2

```

			001C 00000	.ENTRY BASSCMPF_APP, Save R2,R3,R4	: 0208
	54	00000000G	00 9E 00002	MOVAB BASSCVT_OUT_D_G, R4	:
	5E		3C C2 00009	SUBL2 #60, SP	:
2C	AE	010E000E	8F D0 0000C	MOVL #17694734, DSC1	: 0272
30	AE	1C	AE 9E 00014	MOVAB BUF1, DSC1+4	: 0275
34	AE	04	AC 56 00019	CVTFD VAL1, D_VALUE	: 0276
			7E D4 0001E	CLRL -(SP)	: 0277
		30	AE 9F 00020	PUSHAB DSC1	:
		08	AE 9F 00023	PUSHAB RET_LENGTH	:
			7E D4 00026	CLRL -(SP)	:
		44	AE 9F 00028	PUSHAB D_VALUE	:
	64		05 FB 0002B	CALLS #5, BASSCVT_OUT_D_G	:
2C	AE		6E B0 0002E	MOVW RET_LENGTH, DSC2	: 0278

	14	AE	010E000E	8F	D0	00032	MOVL	#17694734, DSC2	:	0280	
	18	AE	04	AE	9E	0003A	MOVAB	BUF2, DSC2+4	:	0283	
	34	AE	08	AC	56	0003F	CVTFD	VAL2, D_VALUE	:	0284	
				7E	D4	00044	CLRL	-(SP)	:	0285	
				18	AE	9F	00046	PUSHAB	DSC2	:	
				08	AE	9F	00049	PUSHAB	RET_LENGTH	:	
				7E	D4	0004C	CLRL	-(SP)	:		
				44	AE	9F	0004E	PUSHAB	D_VALUE	:	
		64		05	FB	00051	CALLS	#5, BASSCVT_OUT_D_G	:		
	14	AE		6E	B0	00054	MOVW	RET_LENGTH, DSC2	:	0286	
	14	AE	2C	AE	81	00058	CMPW	DSC1, DSC2	:	0291	
				08	12	0005D	BNEQ	1\$:		
18	BE			6E	29	0005F	CMPC3	RET_LENGTH, @DSC1+4, @DSC2+4	:	0293	
				04	13	00065	BEQL	2\$:		
		50		01	D0	00067	MOVL	#1, R0	:	0295	
				04	0006A		RET		:		
				50	D4	0006B	CLRL	R0	:	0296	
				04	0006D		RET		:		

; Routine Size: 110 bytes, Routine Base: _BASSCODE + 0000

; 205 0297 1


```

207 0298 1 GLOBAL ROUTINE BASSCMPD_APP (
208 0299 1     VAL1_LO,
209 0300 1     VAL1_HI, VAL2_LO,
210 0301 1     VAL2_HI) =
211 0302 1
212 0303 1
213 0304 1
214 0305 1
215 0306 1
216 0307 1
217 0308 1
218 0309 1
219 0310 1
220 0311 1
221 0312 1
222 0313 1
223 0314 1
224 0315 1
225 0316 1
226 0317 1
227 0318 1
228 0319 1
229 0320 1
230 0321 1
231 0322 1
232 0323 1
233 0324 1
234 0325 1
235 0326 1
236 0327 1
237 0328 1
238 0329 1
239 0330 1
240 0331 1
241 0332 1
242 0333 1
243 0334 1
244 0335 1
245 0336 1
246 0337 1
247 0338 1
248 0339 1
249 0340 1
250 0341 1
251 0342 1
252 0343 1
253 0344 1
254 0345 1
255 0346 2
256 0347 2
257 0348 2
258 0349 2
259 0350 2
260 0351 2
261 0352 2
262 0353 2
263 0354 2

```

GLOBAL ROUTINE BASSCMPD_APP (

 VAL1_LO, : Compare double values
 VAL1_HI, VAL2_LO, : First value to compare
 VAL2_HI) = : Second value to compare

 **

 FUNCTIONAL DESCRIPTION:

 Compares two double values. If the values will PRINT the same, they are considered "approximately equal". This function is useful in Computer Assisted Education applications, when one value is computed and another is read from the terminal. If the two values print the same we do not want to say: "Wrong! You typed x, correct answer is x."

 CALLING SEQUENCE:

```

        MOVD    VAL2, -(SP)
        MOVD    VAL1, -(SP)
        CALLS   #4, BASSCMPD_APP
        TSTL    R0
        BEQL    R0, R1          ; Branch if VAL1 == VAL2

```

 FORMAL PARAMETERS:

```

        VAL1.rd.v    The first value to compare
        VAL2.rd.v    The second value to compare

```

 IMPLICIT INPUTS:

 NONE

 IMPLICIT OUTPUTS:

 NONE

 ROUTINE VALUE:

 COMPLETION CODES:

```

        1 = not approximately equal,
        0 = approximately equal.

```

 SIDE EFFECTS:

 NONE

 --

 BEGIN

 LOCAL

```

        D VALUE : VECTOR [2],
        DSC1 : BLOCK [8, BYTE],
        BUF1 : VECTOR [14, BYTE],
        DSC2 : BLOCK [8, BYTE],
        BUF2 : VECTOR [14, BYTE],
        RET_LENGTH;

```

```

264 0355 2
265 0356 2
266 0357 2
267 0358 2
268 0359 2
269 0360 2
270 0361 2
271 0362 2
272 0363 2
273 0364 2
274 0365 2
275 0366 2
276 0367 2
277 0368 2
278 0369 2
279 0370 2
280 0371 2
281 0372 2
282 0373 2
283 0374 2
284 0375 2
285 0376 2
286 0377 2
287 0378 2
288 0379 2
289 0380 2
290 0381 2
291 0382 2
292 0383 2
293 0384 2
294 0385 1

```

```

!+
Call the formatter for each value to get two character strings.
-
DSC1 [DSC$W_LENGTH] = 14;
DSC1 [DSC$B_DTYPE] = DSC$K_DTYPE_T;
DSC1 [DSC$B_CLASS] = DSC$K_CLASS_S;
DSC1 [DSC$A_POINTER] = BUF1;
D_VALUE [0] = .VAL1_LO;
D_VALUE [1] = .VAL1_HI;
BAS$CVT OUT_D_G (D_VALUE, 0, RET_LENGTH, DSC1, 0);
DSC1 [DSC$W_LENGTH] = .RET_LENGTH;

DSC2 [DSC$W_LENGTH] = 14;
DSC2 [DSC$B_DTYPE] = DSC$K_DTYPE_T;
DSC2 [DSC$B_CLASS] = DSC$K_CLASS_S;
DSC2 [DSC$A_POINTER] = BUF2;
D_VALUE [0] = .VAL2_LO;
D_VALUE [1] = .VAL2_HI;
BAS$CVT OUT_D_G (D_VALUE, 0, RET_LENGTH, DSC2, 0);
DSC2 [DSC$W_LENGTH] = .RET_LENGTH;

!+
If the two strings are equal, the values are approximately equal.
-
IF (.DSC1 [DSC$W_LENGTH] NEQU .DSC2 [DSC$W_LENGTH]) THEN RETURN (1);
IF (CH$EQL (.RET_LENGTH, .DSC1 [DSC$A_POINTER], .RET_LENGTH, .DSC2 [DSC$A_POINTER])) THEN RETURN (0);
RETURN (1);
END;
! end of BASSCMPF_APP

```

			001C 00000	.ENTRY	BASSCMPD APP, Save R2,R3,R4	: 0298
	54	00000000G	00 9E 00002	MOVAB	BASSCVT_OUT_D_G, R4	
	5E		3C C2 00009	SUBL2	#60, SP	
2C	AE	010E000E	8F D0 0000C	MOVL	#17694734, DSC1	: 0359
30	AE	1C	AE 9E 00014	MOVAB	BUF1, DSC1+4	: 0362
34	AE	04	AC 7D 00019	MOVQ	VAL1_LO, D_VALUE	: 0363
			7E D4 0001E	CLRL	-(SP)	: 0365
		30	AE 9F 00020	PUSHAB	DSC1	
		08	AE 9F 00023	PUSHAB	RET_LENGTH	
			7E D4 00026	CLRL	-(SP)	
		44	AE 9F 00028	PUSHAB	D_VALUE	
	64		05 FB 0002B	CALLS	#5, BASSCVT_OUT_D_G	
2C	AE		6E B0 0002E	MOVW	RET_LENGTH, DSC1	: 0366
14	AE	010E000E	8F D0 00032	MOVL	#17694734, DSC2	: 0368
18	AE	04	AE 9E 0003A	MOVAB	BUF2, DSC2+4	: 0371
34	AE	0C	AC 7D 0003F	MOVQ	VAL2_LO, D_VALUE	: 0372
			7E D4 00044	CLRL	-(SP)	: 0374
		18	AE 9F 00046	PUSHAB	DSC2	
		08	AE 9F 00049	PUSHAB	RET_LENGTH	
			7E D4 0004C	CLRL	-(SP)	
		44	AE 9F 0004E	PUSHAB	D_VALUE	

BASSCMP_APPROX
1-006

G 8
16-Sep-1984 00:08:28
14-Sep-1984 11:54:47

VAX-11 Bliss-32 v4.0-742
[BASRTL.SRC]BASCMPAPP.B32;1

Page 9
(4)

		64		05	FB	00051		CALLS	#5, BASSCVT_OUT_D_G	:	
		14	AE	6E	B0	00054		MOVW	RET_LENGTH, DSC2	:	0375
		14	AE	2C	AE	B1	00058	CMPW	DSC1, DSC2	:	0380
				08	12	0005D		BNEQ	1\$:	
18	BE	30	BE	6E	29	0005F		CMPC3	RET_LENGTH, @DSC1+4, @DSC2+4	:	0382
			50	04	13	00065		BEQL	2\$:	
				01	D0	00067	1\$:	MOVL	#1, R0	:	0384
					04	0006A		RET		:	
				50	D4	0006B	2\$:	CLRL	R0	:	0385
					04	0006D		RET		:	

; Routine Size: 110 bytes, Routine Base: _BASSCODE + 006E

```

: 296 0386 1 GLOBAL ROUTINE BASSCMPG_APP (           ! Compare g floating values
: 297 0387 1   VAL1_LO,                          ! First value to compare
: 298 0388 1   VAL1_HI, VAL2_LO,                  ! Second value to compare
: 299 0389 1   VAL2_HI) =
: 300 0390 1
: 301 0391 1  +-+
: 302 0392 1  FUNCTIONAL DESCRIPTION:
: 303 0393 1
: 304 0394 1      Compares two g floating values.  If the values will PRINT the same,
: 305 0395 1      they are considered "approximately equal".  This function is useful
: 306 0396 1      in Computer Assisted Education applications, when one value is
: 307 0397 1      computed and another is read from the terminal.  If the two values
: 308 0398 1      print the same we do not want to say: "Wrong! You typed x, correct
: 309 0399 1      answer is x."
: 310 0400 1
: 311 0401 1  CALLING SEQUENCE:
: 312 0402 1
: 313 0403 1      MOVG     VAL2, -(SP)
: 314 0404 1      MOVG     VAL1, -(SP)
: 315 0405 1      CALLS   #4, BASSCMPG_APP
: 316 0406 1      TSTL    R0
: 317 0407 1      BEQL    1$                      ; Branch if VAL1 == VAL2
: 318 0408 1
: 319 0409 1  FORMAL PARAMETERS:
: 320 0410 1
: 321 0411 1      VAL1.rg.v      The first value to compare
: 322 0412 1      VAL2.rg.v      The second value to compare
: 323 0413 1
: 324 0414 1  IMPLICIT INPUTS:
: 325 0415 1
: 326 0416 1      NONE
: 327 0417 1
: 328 0418 1  IMPLICIT OUTPUTS:
: 329 0419 1
: 330 0420 1      NONE
: 331 0421 1
: 332 0422 1  ROUTINE VALUE:
: 333 0423 1  COMPLETION CODES:
: 334 0424 1
: 335 0425 1      1 = not approximately equal,
: 336 0426 1      0 = approximately equal.
: 337 0427 1
: 338 0428 1  SIDE EFFECTS:
: 339 0429 1
: 340 0430 1      NONE
: 341 0431 1
: 342 0432 1  --
: 343 0433 1
: 344 0434 2  BEGIN
: 345 0435 2
: 346 0436 2  LOCAL
: 347 0437 2      G_VALUE : VECTOR [2],
: 348 0438 2      DSC1  : BLOCK [8, BYTE],
: 349 0439 2      BUF1  : VECTOR [14, BYTE],
: 350 0440 2      DSC2  : BLOCK [8, BYTE],
: 351 0441 2      BUF2  : VECTOR [14, BYTE],
: 352 0442 2      RET_LENGTH;

```

```

353 0443 2
354 0444 2
355 0445 2 Call the formatter for each value to get two character strings.
356 0446 2
357 0447 2 DSC1 [DSC$W_LENGTH] = 14;
358 0448 2 DSC1 [DSC$B_DTYPE] = DSC$K_DTYPE_T;
359 0449 2 DSC1 [DSC$B_CLASS] = DSC$K_CLASS_S;
360 0450 2 DSC1 [DSC$A_POINTER] = BUF1;
361 0451 2 G_VALUE [0] = .VAL1_LO;
362 0452 2 G_VALUE [1] = .VAL1_HI;
363 0453 2 BAS$CVT OUT_G G (G VALUE, 0, RET_LENGTH, DSC1);
364 0454 2 DSC1 [DSC$W_LENGTH] = .RET_LENGTH;
365 0455 2
366 0456 2 DSC2 [DSC$W_LENGTH] = 14;
367 0457 2 DSC2 [DSC$B_DTYPE] = DSC$K_DTYPE_T;
368 0458 2 DSC2 [DSC$B_CLASS] = DSC$K_CLASS_S;
369 0459 2 DSC2 [DSC$A_POINTER] = BUF2;
370 0460 2 G_VALUE [0] = .VAL2_LO;
371 0461 2 G_VALUE [1] = .VAL2_HI;
372 0462 2 BAS$CVT OUT_G G (G VALUE, 0, RET_LENGTH, DSC2);
373 0463 2 DSC2 [DSC$W_LENGTH] = .RET_LENGTH;
374 0464 2
375 0465 2 If the two strings are equal, the values are approximately equal.
376 0466 2
377 0467 2
378 0468 2 IF (.DSC1 [DSC$W_LENGTH] NEQU .DSC2 [DSC$W_LENGTH]) THEN RETURN (1);
379 0469 2
380 0470 2 IF (CH$EQL (.RET_LENGTH, .DSC1 [DSC$A_POINTER], .RET_LENGTH, .DSC2 [DSC$A_POINTER])) THEN RETURN (0);
381 0471 2
382 0472 2 RETURN (1);
383 0473 1 END;

```

! end of BAS\$CMPG_APP

			001C 00000	.ENTRY	BAS\$CMPG_APP, Save R2,R3,R4	: 0386
	54	00000000G	00 9E 00002	MOVAB	BAS\$CVT_OUT_G_G, R4	:
	5E		3C C2 00009	SUBL2	#60, SP	:
2C	AE	010E000E	8F D0 0000C	MOVL	#17694734, DSC1	: 0447
30	AE	1C	AE 9E 00014	MOVAB	BUF1, DSC1+4	: 0450
34	AE	04	AC 7D 00019	MOVQ	VAL1_LO, G_VALUE	: 0451
		2C	AE 9F 0001E	PUSHAB	DSC1	: 0453
		04	AE 9F 00021	PUSHAB	RET_LENGTH	:
		7E	D4 00024	CLRL	-(SP)	:
		40	AE 9F 00026	PUSHAB	G_VALUE	:
	64		04 FB 00029	CALLS	#4, BAS\$CVT_OUT_G_G	:
2C	AE		6E B0 0002C	MOVW	RET_LENGTH, DSC1	: 0454
14	AE	010E000E	8F D0 00030	MOVL	#17694734, DSC2	: 0456
18	AE	04	AE 9E 00038	MOVAB	BUF2, DSC2+4	: 0459
34	AE	0C	AC 7D 0003D	MOVQ	VAL2_LO, G_VALUE	: 0460
		14	AE 9F 00042	PUSHAB	DSC2	: 0462
		04	AE 9F 00045	PUSHAB	RET_LENGTH	:
		7E	D4 00048	CLRL	-(SP)	:
		40	AE 9F 0004A	PUSHAB	G_VALUE	:
	64		04 FB 0004D	CALLS	#4, BAS\$CVT_OUT_G_G	:
14	AE		6E B0 00050	MOVW	RET_LENGTH, DSC2	: 0463

BAS\$CMP_APPROX
1-006

J 8
16-Sep-1984 00:08:28
14-Sep-1984 11:54:47

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

Page 12
(5)

	14	AE	2C	AE	B1	00054		CMPW	DSC1, DSC2	:	0468	
				08	12	00059		BNEQ	1\$:		
18	BE	30	BE	6E	29	0005B		CMPC3	RET_LENGTH, @DSC1+4, @DSC2+4	:	0470	
				04	13	00061		BEQL	2\$:		
				50	01	D0	00063	1\$:	MOVL	#1, R0	:	0472
						04	00066		RET		:	
					50	D4	00067	2\$:	CLRL	R0	:	0473
						04	00069		RET		:	

; Routine Size: 106 bytes, Routine Base: _BAS\$CODE + 00DC

```

385 0474 1 GLOBAL ROUTINE BAS$CMPH_APP (           ! Compare h floating values
386 0475 1     VAL1_0,                             ! First value to compare
387 0476 1     VAL1_1,
388 0477 1     VAL1_2,
389 0478 1     VAL1_3,
390 0479 1     VAL2_0,                             ! Second value to compare
391 0480 1     VAL2_1,
392 0481 1     VAL2_2,
393 0482 1     VAL2_3) =
394 0483 1
395 0484 1 ++
396 0485 1 FUNCTIONAL DESCRIPTION:
397 0486 1
398 0487 1     Compares two h floating values.  If the values will PRINT the same,
399 0488 1     they are considered "approximately equal".  This function is useful
400 0489 1     in Computer Assisted Education applications, when one value is
401 0490 1     computed and another is read from the terminal.  If the two values
402 0491 1     print the same we do not want to say: "Wrong! You typed x, correct
403 0492 1     answer is x."
404 0493 1
405 0494 1 CALLING SEQUENCE:
406 0495 1
407 0496 1     MOVH    VAL2, -(SP)
408 0497 1     MOVH    VAL1, -(SP)
409 0498 1     CALLS  #4, BAS$CMPH_APP
410 0499 1     TSTL    R0
411 0500 1     BEQL    1$                               ; Branch if VAL1 == VAL2
412 0501 1
413 0502 1 FORMAL PARAMETERS:
414 0503 1
415 0504 1     VAL1.rh.v      The first value to compare
416 0505 1     VAL2.rh.v      The second value to compare
417 0506 1
418 0507 1 IMPLICIT INPUTS:
419 0508 1
420 0509 1     NONE
421 0510 1
422 0511 1 IMPLICIT OUTPUTS:
423 0512 1
424 0513 1     NONE
425 0514 1
426 0515 1 ROUTINE VALUE:
427 0516 1 COMPLETION CODES:
428 0517 1
429 0518 1     1 = not approximately equal,
430 0519 1     0 = approximately equal.
431 0520 1
432 0521 1 SIDE EFFECTS:
433 0522 1
434 0523 1     NONE
435 0524 1
436 0525 1 --
437 0526 1
438 0527 2 BEGIN
439 0528 2
440 0529 2 LOCAL
441 0530 2     H_VALUE : VECTOR [4],

```

```

: 442      0531      2          DSC1 : BLOCK [8, BYTE],
: 443      0532      2          BUF1 : VECTOR [14, BYTE],
: 444      0533      2          DSC2 : BLOCK [8, BYTE],
: 445      0534      2          BUF2 : VECTOR [14, BYTE],
: 446      0535      2          RET_LENGTH;
: 447      0536      2
: 448      0537      2
: 449      0538      2          !+ Call the formatter for each value to get two character strings.
: 450      0539      2          !-
: 451      0540      2          DSC1 [DSC$W_LENGTH] = 14;
: 452      0541      2          DSC1 [DSC$B_DTYPE] = DSC$K_DTYPE_T;
: 453      0542      2          DSC1 [DSC$B_CLASS] = DSC$K_CLASS_S;
: 454      0543      2          DSC1 [DSC$A_POINTER] = BUF1;
: 455      0544      2          H_VALUE [0] = .VAL1_0;
: 456      0545      2          H_VALUE [1] = .VAL1_1;
: 457      0546      2          H_VALUE [2] = .VAL1_2;
: 458      0547      2          H_VALUE [3] = .VAL1_3;
: 459      0548      2          BASSCVT OUT H G (H VALUE, 0, RET_LENGTH, DSC1);
: 460      0549      2          DSC1 [DSC$W_LENGTH] = .RET_LENGTH;
: 461      0550      2
: 462      0551      2          DSC2 [DSC$W_LENGTH] = 14;
: 463      0552      2          DSC2 [DSC$B_DTYPE] = DSC$K_DTYPE_T;
: 464      0553      2          DSC2 [DSC$B_CLASS] = DSC$K_CLASS_S;
: 465      0554      2          DSC2 [DSC$A_POINTER] = BUF2;
: 466      0555      2          H_VALUE [0] = .VAL2_0;
: 467      0556      2          H_VALUE [1] = .VAL2_1;
: 468      0557      2          H_VALUE [2] = .VAL2_2;
: 469      0558      2          H_VALUE [3] = .VAL2_3;
: 470      0559      2          BASSCVT OUT H G (H VALUE, 0, RET_LENGTH, DSC2);
: 471      0560      2          DSC2 [DSC$W_LENGTH] = .RET_LENGTH;
: 472      0561      2          !+
: 473      0562      2          !- If the two strings are equal, the values are approximately equal.
: 474      0563      2
: 475      0564      2
: 476      0565      2          IF (.DSC1 [DSC$W_LENGTH] NEQU .DSC2 [DSC$W_LENGTH]) THEN RETURN (1);
: 477      0566      2
: 478      0567      2          IF (CH$EQL (.RET_LENGTH, .DSC1 [DSC$A_POINTER], .RET_LENGTH, .DSC2 [DSC$A_POINTER])) THEN RETURN (0);
: 479      0568      2
: 480      0569      2          RETURN (1);
: 481      0570      1          END;

```

! end of BASSCMPH_APP

			001C 0000	.ENTRY	BASSCMPH APP, Save R2,R3,R4	: 0474
	54	0000000G	00 9E 00002	MOVAB	BASSCVT_OUT_H_G, R4	:
	5E	BC	AE 9E 00009	MOVAB	-68(SP), SP	:
2C	AE	010E000E	8F D0 0000D	MOVL	#17694734, DSC1	: 0540
30	AE	1C	AE 9E 00015	MOVAB	BUF1, DSC1+4	: 0543
34	AE	04	AC 7D 0001A	MOVQ	VAL1_0, H_VALUE	: 0544
3C	AE	0C	AC 7D 0001F	MOVQ	VAL1_2, H_VALUE+8	: 0546
		2C	AE 9F 00024	PUSHAB	DSC1	: 0548
		04	AE 9F 00027	PUSHAB	RET_LENGTH	:
		7E	D4 0002A	CLRL	-(SP)	:
		40	AE 9F 0002C	PUSHAB	H_VALUE	:
	64	04	FB C002F	CALLS	#4, BASSCVT_OUT_H_G	:

2C	AE		6E	B0	00032	MOVW	RET_LENGTH, DSC1	:	0549		
14	AE	010E000E	8F	D0	00036	MOVL	#17894734, DSC2	:	0551		
18	AE	04	AE	9E	0003E	MOVAB	BUF2, DSC2+4	:	0554		
34	AE	14	AC	7D	00043	MOVQ	VAL2_0, H_VALUE	:	0555		
3C	AE	1C	AC	7D	00048	MOVQ	VAL2_2, H_VALUE+8	:	0557		
		14	AE	9F	0004D	PUSHAB	DSC2	:	0559		
		04	AE	9F	00050	PUSHAB	RET_LENGTH	:			
		40	7E	D4	00053	CLRL	-(SP)	:			
		64	AE	9F	00055	PUSHAB	H_VALUE	:			
		14	AE	04	FB	00058	CALLS	#Z, BASSCVT_OUT_H_G	:		
14	AE		6E	B0	0005B	MOVW	RET_LENGTH, DSC2	:	0560		
14	AE	2C	AE	B1	0005F	CMPW	DSC1, DSC2	:	0565		
		18	BE	08	12	00064	BNEQ	1\$:		
		30	BE	6E	29	00066	CMPC3	RET_LENGTH, @DSC1+4, @DSC2+4	:	0567	
				04	13	0006C	BEQL	2\$:		
				01	D0	0006E	1\$:	MOVL	#1, R0	:	0569
				04	00071		2\$:	RET		:	
				50	D4	00072		CLRL	R0	:	0570
				04	00074			RET		:	

; Routine Size: 117 bytes, Routine Base: _BAS\$CODE + 0146

```

: 482      0571 1
: 483      0572 1 END
: 484      0573 1
: 485      0574 0 ELUDOM

```

! end of module BASSCMP_APPROX

PSECT SUMMARY

Name	Bytes	Attributes
_BAS\$CODE	443	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

COMMAND QUALIFIERS

BAS\$CMP_APPROX
1-006

N 8
16-Sep-1984 00:08:28
14-Sep-1984 11:54:47

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

Page 16
(6)

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

: Size: 443 code + 0 data bytes
: Run Time: 00:11.4
: Elapsed Time: 00:22.7
: Lines/CPU Min: 3010
: Lexemes/CPU-Min: 22290
: Memory Used: 66 pages
: Compilation Complete

