

```

PPPPPPPPPPPP      AAAAAAAAAA      SSSSSSSSSSSS      RRRRRRRRRRRR      TTTTTTTTTTTTTT      LLL
PPPPPPPPPPPPPP    AAAAAAAAAA      SSSSSSSSSSSS      RRRRRRRRRRRR      TTTTTTTTTTTTTT      LLL
PPPPPPPPPPPPPP    AAAAAAAAAA      SSSSSSSSSSSS      RRRRRRRRRRRR      TTTTTTTTTTTTTT      LLL
PPP              PPP  AAA          AAA  SSS              RRR              RRR              TTT              LLL
PPP              PPP  AAA          AAA  SSS              RRR              RRR              TTT              LLL
PPP              PPP  AAA          AAA  SSS              RRR              RRR              TTT              LLL
PPP              PPP  AAA          AAA  SSS              RRR              RRR              TTT              LLL
PPP              PPP  AAA          AAA  SSS              RRR              RRR              TTT              LLL
PPP              PPP  AAA          AAA  SSS              RRR              RRR              TTT              LLL
PPPPPPPPPPPPPP    AAA          AAA          SSSSSSSSSS      RRRRRRRRRRRR      TTT              LLL
PPPPPPPPPPPPPP    AAA          AAA          SSSSSSSSSS      RRRRRRRRRRRR      TTT              LLL
PPPPPPPPPPPPPP    AAA          AAA          SSSSSSSSSS      RRRRRRRRRRRR      TTT              LLL
PPP              AAAAAAAAAAAAAAAAAA          SSS          RRR          RRR          TTT              LLL
PPP              AAAAAAAAAAAAAAAAAA          SSS          RRR          RRR          TTT              LLL
PPP              AAAAAAAAAAAAAAAAAA          SSS          RRR          RRR          TTT              LLL
PPP              AAA          AAA          SSS          RRR              RRR          TTT              LLL
PPP              AAA          AAA          SSS          RRR              RRR          TTT              LLL
PPP              AAA          AAA          SSS          RRR              RRR          TTT              LLL
PPP              AAA          AAA          SSS          RRR              RRR          TTT              LLL
PPP              AAA          AAA          SSSSSSSSSSSS      RRR              RRR          TTT              LLLLLLLLLLLLLLLLLL
PPP              AAA          AAA          SSSSSSSSSSSS      RRR              RRR          TTT              LLLLLLLLLLLLLLLLLL
PPP              AAA          AAA          SSSSSSSSSSSS      RRR              RRR          TTT              LLLLLLLLLLLLLLLLLL

```

PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS

PAS

PAS

PAS
PAS
PAS
PAS
PAS

PAS
PAS
PAS
PAS
PAS
PAS
PAS

```
PPPPPPPP      AAAAAA      SSSSSSSS      WW      WW      RRRRRRRR      IIIIII      RRRRRRRR      EEEEEEEEE      GGGGGGGG
PPPPPPPP      AAAAAA      SSSSSSSS      WW      WW      RRRRRRRR      IIIIII      RRRRRRRR      EEEEEEEEE      GGGGGGGG
PP      PP      AA      AA      SS      WW      WW      RR      RR      RR      RR      RR      RR      GG
PP      PP      AA      AA      SS      WW      WW      RR      RR      RR      RR      RR      RR      GG
PP      PP      AA      AA      SS      WW      WW      RR      RR      RR      RR      RR      RR      GG
PPPPPPPP      AA      AA      SSSSSS      WW      WW      RRRRRRRR      IIIIII      RRRRRRRR      EEEEEEEEE      GG
PPPPPPPP      AA      AA      SSSSSS      WW      WW      RRRRRRRR      IIIIII      RRRRRRRR      EEEEEEEEE      GG
PP      AAAAAAAAAA      SS      WW      WW      RR      RR      RR      RR      RR      RR      GG      GGGGGG
PP      AAAAAAAAAA      SS      WW      WW      RR      RR      RR      RR      RR      RR      GG      GGGGGG
PP      AA      AA      SS      WWW      WWW      RR      RR      RR      RR      RR      RR      GG      GG
PP      AA      AA      SS      WWW      WWW      RR      RR      RR      RR      RR      RR      GG      GG
PP      AA      AA      SSSSSSSS      WW      WW      RR      RR      IIIIII      IIIIII      RR      RR      EEEEEEEEE      GGGGGG
PP      AA      AA      SSSSSSSS      WW      WW      RR      RR      IIIIII      IIIIII      RR      RR      EEEEEEEEE      GGGGGG

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 PASSWRITE_REALE_G ( %TITLE 'Write a G_floating in E format'
2 0002 0 IDENT = '1-002' ! File: PASWRIREG.B32 Edit: SBL1002
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: Pascal Language Support
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1 This module contains procedures which write a G_floating in
36 0036 1 exponential notation to a textfile or string.
37 0037 1
38 0038 1 ENVIRONMENT: User mode - AST reentrant
39 0039 1
40 0040 1 AUTHOR: Steven B. Lionel, CREATION DATE: 1-April-1981
41 0041 1
42 0042 1 MODIFIED BY:
43 0043 1
44 0044 1 1-001 - Original. SBL 1-April-1981
45 0045 1 1-002 - Make total-width a longword. SBL 30-June-1982
46 0046 1 --
47 0047 1
```


PASSWRITE_REAL Write a G_floating in E format
1-002 Declarations

F 5
16-Sep-1984 02:22:49
14-Sep-1984 12:52:06

VAX-11 Bliss-32 V4.0-742
[PASRTL.SRC]PASWRIREG.B32;1

Page 2
(2)

```
: 49      0048 1 %SBTTL 'Declarations'
: 50      0049 1
: 51      0050 1 PROLOGUE DEFINITIONS:
: 52      0051 1
: 53      0052 1
: 54      0053 1 REQUIRE 'RTLIN:PASPROLOG';
: 55      0117 1 ! Externals, linkages, PSECTs, structures
: 56      0118 1
: 57      0119 1 TABLE OF CONTENTS:
: 58      0120 1
: 59      0121 1
: 60      0122 1 FORWARD ROUTINE
: 61      0123 1 PASSWRITE_REAL_G: NOVALUE,
: 62      0124 1 PASSWRITEV_REAL_G: NOVALUE;
: 63      0125 1 ! Write to textfile
: 64      0126 1 ! Write to string
: 65      0127 1
: 66      0128 1 MACROS:
: 67      0129 1 NONE
: 68      0130 1
: 69      0131 1 EQUATED SYMBOLS:
: 70      0132 1
: 71      0133 1 NONE
: 72      0134 1
: 73      0135 1 FIELDS:
: 74      0136 1
: 75      0137 1 NONE
: 76      0138 1
: 77      0139 1 OWN STORAGE:
: 78      0140 1
: 79      0141 1 NONE
: 80      0142 1
```

```

82 0143 1 %SBTTL 'PASSWRITE REALE G - Write G_floating in E format to textfile'
83 0144 1 GLOBAL ROUTINE PASSWRITE_REALE_G (
84 0145 1 PFV: REF $PASSPFV_FILE_VARIABLE,      ! File variable
85 0146 1 VALUE_0, VALUE_1,                      ! Value to write
86 0147 1 TOTAL_WIDTH: SIGNED,                  ! Total field width
87 0148 1 ERROR                                ! Error unwind address
88 0149 1 ): NOVALUE =
89 0150 1
90 0151 1 ++
91 0152 1 FUNCTIONAL DESCRIPTION:
92 0153 1
93 0154 1 This procedure writes a G_floating value in exponential notation
94 0155 1 to the specified textfile.
95 0156 1
96 0157 1 CALLING SEQUENCE:
97 0158 1
98 0159 1 CALL PASSWRITE_REALE_G (PFV.mr.r, VALUE.rg.v, TOTAL_WIDTH.r.l.v
99 0160 1 [ERROR.j.r])
100 0161 1
101 0162 1 FORMAL PARAMETERS:
102 0163 1
103 0164 1 PFV - The Pascal File Variable (PFV) passed by reference.
104 0165 1 The structure of the PFV is defined in PASPFV.REQ.
105 0166 1
106 0167 1 VALUE - The G_floating value to write by immediate value.
107 0168 1 Note that this requires two argument list positions.
108 0169 1
109 0170 1 TOTAL_WIDTH - Total field width.
110 0171 1
111 0172 1 ERROR - Optional. Address to unwind to if an error occurs.
112 0173 1
113 0174 1 IMPLICIT INPUTS:
114 0175 1
115 0176 1 NONE
116 0177 1
117 0178 1 IMPLICIT OUTPUTS:
118 0179 1
119 0180 1 NONE
120 0181 1
121 0182 1 ROUTINE VALUE:
122 0183 1
123 0184 1 NONE
124 0185 1
125 0186 1 SIDE EFFECTS:
126 0187 1
127 0188 1 If the file is the standard file OUTPUT, it is implicitly opened.
128 0189 1
129 0190 1 SIGNALLED ERRORS:
130 0191 1
131 0192 1 LINTOOLON - line too long
132 0193 1 NEGWIDDIG - negative Width or Digits specification is not allowed
133 0194 1
134 0195 1 --
135 0196 1
136 0197 2 BEGIN
137 0198 2
138 0199 2 LOCAL

```



```

139 0200 2      FCB: REF $PASSFCB CONTROL_BLOCK,      ! File control block
140 0201 2      FIELD_WIDTH: SIGNED,                  ! Minimum/actual width
141 0202 2      REMAINING_WIDTH,                      ! Maximum width
142 0203 2      PFV_ADDR: VOLATILE,                   ! Enable argument
143 0204 2      UNWIND_ACT: VOLATILE,                  ! Enable argument
144 0205 2      ERROR_ADDR: VOLATILE;                 ! Enable argument
145 0206 2
146 0207 2      BUILTIN
147 0208 2      ACTUALCOUNT;
148 0209 2
149 0210 2      ENABLE
150 0211 2      PASS$IO_HANDLER (PFV_ADDR, UNWIND_ACT, ERROR_ADDR);      ! Enable error handler
151 0212 2
152 0213 2      !+
153 0214 2      ! Get ERROR parameter, if present.
154 0215 2      !-
155 0216 2
156 0217 2      IF ACTUALCOUNT () GEQU 5
157 0218 2      THEN
158 0219 2      ERROR_ADDR = .ERROR;                    ! Set unwind address
159 0220 2
160 0221 2      PFV_ADDR = PFV [PFV$R_PFV];             ! Set PFV address
161 0222 2
162 0223 2      !+
163 0224 2      ! Validate PFV and get PFV.
164 0225 2      !-
165 0226 2
166 0227 2      PASS$VALIDATE_PFV (PFV [PFV$R_PFV]; FCB);
167 0228 2
168 0229 2      !+
169 0230 2      ! Set unwind action to unlock file.
170 0231 2      !-
171 0232 2
172 0233 2      UNWIND_ACT = PASS$K_UNWIND_UNLOCK;
173 0234 2
174 0235 2      !+
175 0236 2      ! Do common initialization.
176 0237 2      !-
177 0238 2
178 0239 2      PASS$INIT_WRITE (PFV [PFV$R_PFV], FCB [FCB$R_FCB]; FCB);
179 0240 2
180 0241 2      !+
181 0242 2      ! Get field width and maximum width. Ensure that field width is not
182 0243 2      ! negative.
183 0244 2      !-
184 0245 2
185 0246 2      FIELD_WIDTH = .TOTAL_WIDTH;
186 0247 2      IF .FIELD_WIDTH LSS 0
187 0248 2      THEN
188 0249 2      $PASS$IO_ERROR (PASS$NEGWIDDIG, 0);
189 0250 2      REMAINING_WIDTH = .FCB [FCB$A_RECORD_END] - .FCB [FCB$A_RECORD_CUR];
190 0251 2
191 0252 2      !+
192 0253 2      ! Do the convert. If it fails, give an error.
193 0254 2      !-
194 0255 2
195 0256 2      IF NOT PASS$CVT_G_T (VALUE_0,           ! Value to convert

```

```

196      0257 2      .FCB [FCB$A_RECORD_CUR], ! Destination
197      0258 2      FIELD_WIDTH, ! Minimum/actual width
198      0259 2      .REMAINING_WIDTH) ! Maximum width
199      0260 2
200      0261 2      THEN
201      0262 2      $PASSIO_ERROR (PASS_LINTOOLON,1,(.FIELD_WIDTH-.REMAINING_WIDTH));
202      0263 2
203      0264 2      !+
204      0265 2      ! Advance the record pointer.
205      0266 2      !-
206      0267 2      FCB [FCB$A_RECORD_CUR] = .FCB [FCB$A_RECORD_CUR] + .FIELD_WIDTH;
207      0268 2
208      0269 2      !+
209      0270 2      ! Call WRITE epilogue routine to move the last character written to the
210      0271 2      ! user's buffer and to unlock the file variable.
211      0272 2      !-
212      0273 2
213      0274 2      PASS$END_WRITE (PFV [PFV$R_PFV], FCB [FCB$R_FCB]);
214      0275 2
215      0276 2      RETURN;
216      0277 2
217      0278 1      END;

```

! End of routine PASSWRITE_REALE_G

.TITLE PASSWRITE_REALE_G Write a G_floating in E forma
 .IDENT \1-002\

.EXTRN PASSWRITE_REALE_G
 .EXTRN PASSWRITEV_REALE_G
 .EXTRN PASS\$IO_HANDLER
 .EXTRN PASS\$VACIDATE_PFV
 .EXTRN PASS\$INIT_WRITE
 .EXTRN PASS\$SIGNAL, PASS\$NEGWIDDIG
 .EXTRN PASS\$CVT_G_T, PASS\$LINTOOLON
 .EXTRN PASS\$END_WRITE

.PSECT _PASSCODE, NOWRT, SHR, PIC, 2

.ENTRY PASSWRITE_REALE_G, Save R2,R3,R4,R5,R6,R7,- R8 : 0144
 MOVAB PASS\$SIGNAL, R8
 SUBL2 #16, SP
 CLRQ ERROR_ADDR : 0197
 CLRL PFV_ADDR
 MOVAL 4\$, -(FP)
 CMPB (AP), #5 : 0217
 BLSSU 1\$
 MOVL ERROR, ERROR_ADDR : 0219
 MOVL PFV, R6 : 0221
 MOVL R6, PFV_ADDR
 JSB PASS\$VACIDATE_PFV : 0227
 MOVL #1, UNWIND_ACT : 0233
 JSB PASS\$INIT_WRITE : 0239
 MOVL TOTAL_WIDTH, FIELD_WIDTH : 0246
 BGEQ 2\$: 0247
 CLRL -(SP) : 0249

01FC 00000
 58 00000000G 00 9E 00002
 5E 04 AE 7C 0000C
 0C 0C AE 04 0000F
 6D 0067 CF DE 00012
 05 6C 91 00017
 04 AE 14 AC D0 0001C
 56 04 AC D0 00021 1\$:
 0C AE 56 D0 00025
 08 AE 00000000G 00 16 00029
 AE 01 D0 0002F
 00000000G 00 16 00033
 6E 10 AC D0 00039
 0A 18 0003D
 7E D4 0003F

PASSWRITE_REALE		Write a G_floating in E format		J 5		16-Sep-1984 02:22:49		VAX-11 Bliss-32 V4.C=742		Page 6	
1-002		PASSWRITE_REALE_G - Write G_floating in E format		14-Sep-1984 12:52:06		[PASRTL.SRC]PASWIREG.B32;1				(3)	

52	F0	A7	EC	A7	C3	00049	2\$:	MOVZBL	#PASSK NEGWIDDIG, -(SP)		
				52	DD	0004F		CALLS	#2, PASS\$SIGNAL		
			04	AE	9F	00051		RET			
			EC	A7	DD	00054		SUBL3	-20(FCB), -16(FCB), REMAINING_WIDTH		0250
			08	AC	9F	00057		PUSHL	REMAINING_WIDTH		0259
00000000G	00			04	FB	0005A		PUSHAB	FIELD_WIDTH		0256
	0E			50	E8	00061		PUSHL	-20(FCB)		0257
7E	6E			52	C3	00064		PUSHAB	VALUE 0		0256
				01	DD	00068		CALLS	#4, PASS\$CVT_G_T		
	7E		00G	8F	9A	0006A		BLBS	R0, 3\$		
	68			03	FB	0006E		SUBL3	REMAINING_WIDTH, FIELD_WIDTH, -(SP)		0261
				04	00071			PUSHL	#1		
		EC	A7	6E	C0	00072	3\$:	MOVZBL	#PASSK LINTOOLON, -(SP)		
				00	16	00076		CALLS	#3, PASS\$SIGNAL		
				04	0007C			RET			
				0000	0007D	4\$:		ADDL2	FIELD_WIDTH, -20(FCB)		0267
				50	08	AC	D0	JSB	PASS\$END_WRITE		0274
				04	A0	D0	00083	RET			0278
				F4	A0	9F	00087	.WORD	Save nothing		0197
				F8	A0	9F	0008A	MOVL	8(AP), R0		
				FC	A0	9F	0008D	MOVL	4(R0), R0		
				03	DD	00090		PUSHAB	ERROR_ADDR		
				5E	DD	00092		PUSHAB	UNWIND_ACT		
				AC	7D	00094		PUSHAB	PFV_ADDR		
00000000G	7E	04		03	FB	00098		PUSHL	#3		
	00			04	0009F			PUSHL	SP		
								MOVQ	4(AP), -(SP)		
								CALLS	#3, PASS\$IO_HANDLER		
								RET			


```

; Routine Size: 160 bytes,   Routine Base: _PASS$CODE + 0000

; 218      0279 1
; 219      0280 1 !<BLF/PAGE>

```



```

221 0281 1 %SBTTL 'PASSWRITEV_REALE_G - Write G_floating in E format to string'
222 0282 1 GLOBAL ROUTINE PASSWRITEV_REALE_G (
223 0283 1     MAX_LENGTH: WORD,                                ! Maximum length of string
224 0284 1     STRING_LINE: REF VECTOR [, WORD],              ! String to write to
225 0285 1     VALUE0,VALUE1,                                  ! Value to write
226 0286 1     TOTAL_WIDTH: SIGNED,                            ! Total field width
227 0287 1     ERROR                                           ! Error unwind address
228 0288 1 ) : NOVALUE =
229 0289 1
230 0290 1 ++
231 0291 1 FUNCTIONAL DESCRIPTION:
232 0292 1
233 0293 1     This procedure writes a G_floating in exponential format
234 0294 1     to the specified string.
235 0295 1
236 0296 1 CALLING SEQUENCE:
237 0297 1
238 0298 1     CALL PASSWRITEV_REALE_G (MAX_LENGTH.rw.v, STRING_LINE.wvt.r,
239 0299 1     VALUE.rg.v, TOTAL_WIDTH.r[v [, ERROR.j.r])
240 0300 1
241 0301 1 FORMAL PARAMETERS:
242 0302 1
243 0303 1     MAX_LENGTH      - The maximum length of STRING_LINE.
244 0304 1
245 0305 1     STRING_LINE    - A varying string to which the output will be appended.
246 0306 1
247 0307 1     VALUE          - The value to write. Note that the G_floating value
248 0308 1                     is passed by immediate value in two argument list
249 0309 1                     entries.
250 0310 1
251 0311 1     TOTAL_WIDTH     - The width of the field to write.
252 0312 1
253 0313 1     ERROR          - Optional. If specified, the address to unwind to
254 0314 1                     in case of an error.
255 0315 1
256 0316 1 IMPLICIT INPUTS:
257 0317 1
258 0318 1     NONE
259 0319 1
260 0320 1 IMPLICIT OUTPUTS:
261 0321 1
262 0322 1     NONE
263 0323 1
264 0324 1 ROUTINE VALUE:
265 0325 1
266 0326 1     NONE
267 0327 1
268 0328 1 SIDE EFFECTS:
269 0329 1
270 0330 1     NONE
271 0331 1
272 0332 1 SIGNALLED ERRORS:
273 0333 1
274 0334 1     See PASSWRITE_REALE_G
275 0335 1
276 0336 1 --
277 0337 1

```

```

278      0338      2      BEGIN
279      0339      2
280      0340      2      LOCAL
281      0341      2          PFV: $PASSPFV FILE VARIABLE,      ! Pascal File Variable
282      0342      2          ARG_LIST: VECTOR [5, LONG],      ! Argument list
283      0343      2          PFV_ADDR: VOLATILE,              ! Enable argument
284      0344      2          UNWIND_ACT: VOLATILE,            ! Enable argument
285      0345      2          ERROR_ADDR: VOLATILE;            ! Enable argument
286      0346      2
287      0347      2      BUILTIN
288      0348      2          ACTUALCOUNT;                      ! Count of arguments
289      0349      2
290      0350      2      ENABLE
291      0351      2          PASS$IO_HANDLER (PFV_ADDR, UNWIND_ACT, ERROR_ADDR);      ! Enable error handler
292      0352      2
293      0353      2      !+
294      0354      2      ! Get ERROR parameter, if present.
295      0355      2      !-
296      0356      2
297      0357      2      IF ACTUALCOUNT () GEQU 6
298      0358      2      THEN
299      0359      2          ERROR_ADDR = .ERROR;                ! Set unwind address
300      0360      2
301      0361      2      PFV_ADDR = PFV [PFV$R_PFV];            ! Set PFV address
302      0362      2
303      0363      2      !+
304      0364      2      ! Set up ARG_LIST.
305      0365      2      !-
306      0366      2
307      0367      2      ARG_LIST [0] = 4;                      ! Four arguments
308      0368      2      ARG_LIST [1] = PFV [PFV$R_PFV];        ! PFV address
309      0369      2      ARG_LIST [2] = .VALUE0;                ! Value to write
310      0370      2      ARG_LIST [3] = .VALUE1;
311      0371      2      ARG_LIST [4] = .TOTAL_WIDTH;          ! Field width
312      0372      2
313      0373      2      !+
314      0374      2      ! Call PASS$DO_WRITEV to do the work, giving it the address of
315      0375      2      ! PASSWRITE_REALE_G to call.
316      0376      2      !-
317      0377      2
318      0378      2      PASS$DO_WRITEV (PFV [PFV$R_PFV], .MAX_LENGTH, STRING_LINE [0], ARG_LIST,
319      0379      2          PASSWRITE_REALE_G);
320      0380      2
321      0381      2      RETURN;
322      0382      2
323      0383      1      END;

```

```

! End of routine PASSWRITEV_REALE_G

```

```

.EXTRN  PASS$DO_WRITEV

.ENTRY  PASSWRITEV_REALE_G, Save R2,R3,R4,R5,R6      : 0282
SUBL2   #44, SP
CLRL    ERROR_ADDR                                  : 0338
CLRL    UNWIND_ACT
MOVAL   2$, (FP)
CMPB    (AP), #6                                     : 0357

```

```

007C 00000
5E      2C  C2 00002
        7E  D4 00005
        AE  7C 00007
        6D      04
        06      003E
        CF  DE 0000A
        6C  91 0000F

```


PASSWRITE_REALE Write a G floating in E format
1-002

PASSWRITEV_REALE_G - Write G floating in E form

M 5
16-Sep-1984 02:22:49
14-Sep-1984 12:52:06

VAX-11 Bliss-32 V4.0-742
[PASRTL.SRC]PASWRIREG.532;1

Page 9
(4)

08	6E	18	04	1F	00012	BLSSU	1\$		
0C	AE	20	AC	DO	00014	MOVL	ERROR, ERROR_ADDR	:	0359
10	AE		AE	9E	00018	MOVAB	PFV, PFV_ADDR	:	0361
14	AE	20	04	DO	0001D	MOVL	#4, ARG_LIST	:	0367
1C	AE	0C	AE	9E	00021	MOVAB	PFV, ARG_LIST+4	:	0368
	AE	14	AC	7D	00026	MOVQ	VALUE0, ARG_LIST+8	:	0369
	AE	14	AC	DO	0002B	MOVL	TOTAL_WIDTH, ARG_LIST+16	:	0371
	55	FF2C	CF	9E	00030	MOVAB	PASSWRITE_REALE_G, R5	:	0378
	54	0C	AE	9E	00035	MOVAB	ARG_LIST, R4	:	
	56	20	AE	9E	00039	MOVAB	PFV, R6	:	
	53	08	AC	DO	0003D	MOVL	STRING_LINE, R3	:	
	52	04	AC	3C	00041	MOVZWL	MAX_LENGTH, R2	:	
		00000000G	00	16	00045	JSB	PASS\$DO_WRITEV	:	
				04	0004B	RET		:	0383
				0000	0004C	.WORD	Save nothing	:	0338
	50	08	AC	DO	0004E	MOVL	8(AP), R0	:	
	50	04	A0	DO	00052	MOVL	4(R0), R0	:	
		D0	A0	9F	00056	PUSHAB	ERROR_ADDR	:	
		D4	A0	9F	00059	PUSHAB	UNWIND_ACT	:	
		D8	A0	9F	0005C	PUSHAB	PFV_ADDR	:	
			03	DD	0005F	PUSHL	#3	:	
			5E	DD	00061	PUSHL	SP	:	
	7E	04	AC	7D	00063	MOVQ	4(AP), -(SP)	:	
00000000G	00		03	FB	00067	CALLS	#3, PASS\$IO_HANDLER	:	
			04	0006E	RET			:	

; Routine Size: 111 bytes, Routine Base: _PASS\$CODE + 00A0

; 324 0384 1
; 325 0385 1 !<BLF/PAGE>

```

: 327      0386 1 END
: 328      0387 1
: 329      0388 0 ELUDOM
! End of module PASSWRITE_REAL_G

```

PSECT SUMMARY

Name	Bytes	Attributes
_PASSCODE	271	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	00:01.0
\$255\$DUA28:[PASRTL.OBJ]PASLIB.L32;1	427	97	22	33	00:00.4

COMMAND QUALIFIERS

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

```

: 330      0389 0
: Size:      271 code + 0 data bytes
: Run Time:   00:07.2
: Elapsed Time: 00:19.4
: Lines/CPU Min: 3232
: Lexemes/CPU-Min: 12739
: Memory Used: 80 pages
: Compilation Complete

```


0298 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PASWIREH
LIS

PASWIREL
LIS

PASWIREG
LIS

PASWISTR
LIS

PATCH

PATDEF
MDL

PASWIREF
LIS

PATCH
MAP

SRMDEF
MDL

PASWIRFH
LIS

PASWIRAR
LIS

PASWIREL
LIS

PASWIRLS
LIS

PASWIRINT
LIS

PASWIRIOCT
LIS

PASWIRFG
LIS

BSTRUC
REQ

PASWIRFF
LIS

CHRKEY
REQ

PASWIRFD
LIS