

```

PPPPPPPPPPPP      AAAA AAAA      SSSSSSSSSSSS  RRRRRRRRRRRR  TTTTTTTTTTTTTT  LLL
PPPPPPPPPPPP      AAAAAA AAAA      SSSSSSSSSSSS  RRRRRRRRRRRR  TTTTTTTTTTTTTT  LLL
PPPPPPPPPPPP      AAAAAA AAAA      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
PPPPPPPPPPPP      AAA          AAA          SSSSSSSSSS  RRRRRRRRRRRR  TTT          LLL
PPPPPPPPPPPP      AAA          AAA          SSSSSSSSSS  RRRRRRRRRRRR  TTT          LLL
PPPPPPPPPPPP      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

```

[illegible]


```
1 0001 0 MODULE PASSWRITE_REALE_F.( %TITLE 'Write an F_floating in E format'
2 0002 0 IDENT = '1-002' ! File: PASWRIREF.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 a procedure which writes an F_floating in
36 0036 1 exponential format to a textfile.
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_REALE Write an F floating in E format
1-002 Declarations

H 4
16-Sep-1984 02:22:17
14-Sep-1984 12:52:06

VAX-11 Bliss-32 V4.0-742
[PASRTL.SRC]PASWRITE.REF.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_REALE F: NOVALUE,
: 62      0124 1 PASSWRITEV_REALE_F: 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 XSBTTL 'PASSWRITE REALE F - Write F_floating in E format to textfile'
83 0144 1 GLOBAL ROUTINE PASSWRITE_REALE_F (
84 0145 1     PFV: REF $PASSPFV_FILE_VARIABLE,      ! File variable
85 0146 1     VALUE,                               ! 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 an F_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_F (PFV.mr.r, VALUE.rf.v, TOTAL_WIDTH.rl.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 F_floating value to write.
107 0168 1
108 0169 1     TOTAL_WIDTH  - Total field width.
109 0170 1
110 0171 1     ERROR       - Optional. Address to unwind to if an error occurs.
111 0172 1
112 0173 1 IMPLICIT INPUTS:
113 0174 1
114 0175 1     NONE
115 0176 1
116 0177 1 IMPLICIT OUTPUTS:
117 0178 1
118 0179 1     NONE
119 0180 1
120 0181 1 ROUTINE VALUE:
121 0182 1
122 0183 1     NONE
123 0184 1
124 0185 1 SIDE EFFECTS:
125 0186 1
126 0187 1     If the file is the standard file OUTPUT, it is implicitly opened.
127 0188 1
128 0189 1 SIGNALLED ERRORS:
129 0190 1
130 0191 1     LINTOOLON - line too long
131 0192 1     NEGWIDDIG - negative Width or Digits specification is not allowed
132 0193 1
133 0194 1 --
134 0195 1
135 0196 2 BEGIN
136 0197 2
137 0198 2 LOCAL
138 0199 2     FCB: REF $PASSFCB_CONTROL_BLOCK,      ! File control block

```

```

139 0200 2 FIELD WIDTH: SIGNED, ! Minimum width
140 0201 2 REMAINING_WIDTH, ! Width remaining on line
141 0202 2 PFV_ADDR: VOLATILE, ! Enable argument
142 0203 2 UNWIND_ACT: VOLATILE, ! Enable argument
143 0204 2 ERROR_ADDR: VOLATILE; ! Enable argument
144 0205 2
145 0206 2 BUILTIN
146 0207 2 ACTUALCOUNT;
147 0208 2
148 0209 2 ENABLE
149 0210 2 PASS$IO_HANDLER (PFV_ADDR, UNWIND_ACT, ERROR_ADDR); ! Enable error handler
150 0211 2
151 0212 2 !+
152 0213 2 ! Get ERROR parameter, if present.
153 0214 2 !-
154 0215 2
155 0216 2 IF ACTUALCOUNT () GEQU 4
156 0217 2 THEN
157 0218 2 ERROR_ADDR = .ERROR; ! Set unwind address
158 0219 2
159 0220 2 PFV_ADDR = PFV [PFV$R_PFV]; ! Set PFV address
160 0221 2
161 0222 2 !+
162 0223 2 ! Validate PFV and get PFV.
163 0224 2 !-
164 0225 2
165 0226 2 PASS$VALIDATE_PFV (PFV [PFV$R_PFV]; FCB);
166 0227 2
167 0228 2 !+
168 0229 2 ! Set unwind action to unlock file.
169 0230 2 !-
170 0231 2
171 0232 2 UNWIND_ACT = PASS$K_UNWIND_UNLOCK;
172 0233 2
173 0234 2 !+
174 0235 2 ! Do common initialization.
175 0236 2 !-
176 0237 2
177 0238 2 PASS$INIT_WRITE (PFV [PFV$R_PFV], FCB [FCB$R_FCB]; FCB);
178 0239 2
179 0240 2 !+
180 0241 2 ! Get field width and remaining width. Check for valid field width.
181 0242 2 !-
182 0243 2
183 0244 2 FIELD_WIDTH = .TOTAL_WIDTH;
184 0245 2 IF .FIELD_WIDTH LSS 0
185 0246 2 THEN
186 0247 2 $PASS$IO_ERROR (PASS$NEGWIDDIG, 0);
187 0248 2 REMAINING_WIDTH = .FCB [FCB$A_RECORD_END] - .FCB [FCB$A_RECORD_CUR];
188 0249 2
189 0250 2 !+
190 0251 2 ! Convert to text and put in record buffer. If failure, signal an error.
191 0252 2 ! FIELD_WIDTH will be modified to have the actual width used.
192 0253 2 !-
193 0254 2
194 0255 2 IF NOT PASS$CVT_F_T (VALUE, ! Value to convert
195 0256 2 .FCB [FCB$A_RECORD_CUR], ! Destination

```



```

196      0257 2
197      0258 2
198      0259 2
199      0260 2
200      0261 2
201      0262 2
202      0263 2
203      0264 2
204      0265 2
205      0266 2
206      0267 2
207      0268 2
208      0269 2
209      0270 2
210      0271 2
211      0272 2
212      0273 2
213      0274 2
214      0275 2
215      0276 2
216      0277 1

      FIELD_WIDTH,      ! Minimum/actual width
      .REMAINING_WIDTH) ! Width remaining on line
THEN
  SPASSIO_ERROR (PASS_LINTOOLON,1,(.FIELD_WIDTH-.REMAINING_WIDTH));

  !+
  ! Advance the record pointer.
  !-

  FCB [FCB$A_RECORD_CUR] = .FCB [FCB$A_RECORD_CUR] + .FIELD_WIDTH;

  !+
  ! Call WRITE epilogue routine to move the last character written to the
  ! user's buffer and to unlock the file variable.
  !-

  PASS$END_WRITE (PFV [PFV$R_PFV], FCB [FCB$R_FCB]);

  RETURN;

END;

```

! End of routine PASSWRITE_REALE_F

.TITLE PASSWRITE_REALE_F Write an F floating in E form
 at

.IDENT \1-002\

.EXTRN PASSWRITE_REALE_F
 .EXTRN PASSWRITE_REALE_F
 .EXTRN PASS\$IO_HANDLER
 .EXTRN PASS\$VACIDATE_PFV
 .EXTRN PASS\$INIT_WRITE
 .EXTRN PASS\$SIGNAL, PASS\$K_NEGWIDDIG
 .EXTRN PASS\$CVT_F_T, PASS\$K_LINTOOLON
 .EXTRN PASS\$END_WRITE

.PSECT _PASSCODE,NOWRT, SHR, PIC,2

.ENTRY PASSWRITE_REALE_F, Save R2,R3,R4,R5,R6,R7,- ; 0144
 R8
 MOVAB PASS\$SIGNAL, R8
 SUBL2 #16, SP
 CLRQ ERROR_ADDR ; 0196
 CLRL PFV_ADDR
 MOVAL 4\$,-(FP)
 CMPB (AP), #4 ; 0216
 BLSSU 1\$
 MOVL ERROR, ERROR_ADDR ; 0218
 MOVL PFV, R6 ; 0220
 MOVL R6, PFV_ADDR
 JSB PASS\$VACIDATE_PFV ; 0226
 MOVL #1, UNWIND_ACT ; 0232
 JSB PASS\$INIT_WRITE ; 0238
 MOVL TOTAL_WIDTH, FIELD_WIDTH ; 0244
 BGEQ 2\$; 0245
 CLRL -(SP) ; 0247
 MOVZBL #PASS\$K_NEGWIDDIG, -(SP)

01FC 00000
 58 00000000G 00 9E 00002
 5E 10 C2 00009
 04 AE 7C 0000C
 0C AE D4 0000F
 6D 0067 CF DE 00012
 04 6C 91 00017
 05 1F 0001A
 04 AE 10 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 0C AC D0 00039
 0A 18 0003D
 7E D4 0003F
 00G 8F 9A 00041

		68		02	FB	00045	CALLS	#2, PASS\$SIGNAL	
					04	00048	RET		
52	F0	A7	EC	A7	C3	00049	2\$:	SUBL3	-20(FCB), -16(FCB), REMAINING_WIDTH
				52	DD	0004F		PUSHL	REMAINING_WIDTH
			04	AE	9F	00051		PUSHAB	FIELD_WIDTH
			EC	A7	DD	00054		PUSHL	-20(FCB)
			08	AC	9F	00057		PUSHAB	VALUE
	00000000G	00		04	FB	0005A		CALLS	#4, PASS\$CVT_F_T
		0E		50	E8	00061		BLBS	R0, 3\$
7E		6E		52	C3	00064		SUBL3	REMAINING_WIDTH, FIELD_WIDTH, -(SP)
				01	DD	00068		PUSHL	#1
		7E	00G	8F	9A	0006A		MOVZBL	#PASS\$K_LINTOOLON, -(SP)
		68		03	FB	0006E		CALLS	#3, PASS\$SIGNAL
					04	00071		RET	
			EC	A7	6E	C0	00072	3\$:	ADDL2
					00	16	00076		JSB
					04	0007C		RET	PASS\$END_WRITE
					0000	0007D	4\$:	.WORD	Save nothing
		50	08	AC	D0	0007F		MOVL	8(AP), R0
		50	04	A0	D0	00083		MOVL	4(R0), R0
			F4	A0	9F	00087		PUSHAB	ERROR_ADDR
			F8	A0	9F	0008A		PUSHAB	UNWIND_ACT
			FC	A0	9F	0008D		PUSHAB	PFV_ADDR
				03	DD	00090		PUSHL	#3
				5E	DD	00092		PUSHL	SP
		7E	04	AC	7D	00094		MOVQ	4(AP), -(SP)
	00000000G	00		03	FB	00098		CALLS	#3, PASS\$IO_HANDLER
					04	0009F		RET	

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

; 217 0278 1
; 218 0279 1 !<BLF/PAGE>


```

: 220 0280 1 %SBTTL 'PASSWRITEV REALE F - Write F_floating in E format to string'
: 221 0281 1 GLOBAL ROUTINE PASSWRITEV_REALE_F (
: 222 0282 1     MAX_LENGTH: WORD,                                ! Maximum length of string
: 223 0283 1     STRING_LINE: REF VECTOR [, WORD],              ! String to write to
: 224 0284 1     VALUE,                                           ! Value to write
: 225 0285 1     TOTAL_WIDTH: SIGNED,                             ! Total field width
: 226 0286 1     ERROR                                           ! Error unwind address
: 227 0287 1 ) : NOVALUE =
: 228 0288 1
: 229 0289 1 ++
: 230 0290 1 FUNCTIONAL DESCRIPTION:
: 231 0291 1     This procedure writes an F_floating in E format
: 232 0292 1     to the specified string.
: 233 0293 1
: 234 0294 1 CALLING SEQUENCE:
: 235 0295 1     CALL PASSWRITEV_REALE_F (MAX_LENGTH.rw.v, STRING_LINE.wvt.r,
: 236 0296 1     VALUE.rf.v, TOTAL_WIDTH.r[v [, ERROR.j.r])
: 237 0297 1
: 238 0298 1 FORMAL PARAMETERS:
: 239 0299 1
: 240 0300 1     MAX_LENGTH      - The maximum length of STRING_LINE.
: 241 0301 1
: 242 0302 1     STRING_LINE    - A varying string to which the output will be appended.
: 243 0303 1
: 244 0304 1     VALUE          - The value to write.
: 245 0305 1
: 246 0306 1     TOTAL_WIDTH    - The width of the field to write.
: 247 0307 1
: 248 0308 1     ERROR          - Optional. If specified, the address to unwind to
: 249 0309 1     in case of an error.
: 250 0310 1
: 251 0311 1 IMPLICIT INPUTS:
: 252 0312 1
: 253 0313 1     NONE
: 254 0314 1
: 255 0315 1 IMPLICIT OUTPUTS:
: 256 0316 1
: 257 0317 1     NONE
: 258 0318 1
: 259 0319 1 ROUTINE VALUE:
: 260 0320 1
: 261 0321 1     NONE
: 262 0322 1
: 263 0323 1 SIDE EFFECTS:
: 264 0324 1
: 265 0325 1     NONE
: 266 0326 1
: 267 0327 1 SIGNALLED ERRORS:
: 268 0328 1
: 269 0329 1     See PASSWRITE_REALE_F
: 270 0330 1
: 271 0331 1 --
: 272 0332 1
: 273 0333 1
: 274 0334 1
: 275 0335 2 BEGIN
: 276 0336 2

```

```

277 0337 2 LOCAL
278 0338 2 PFV: $PASSPFV FILE VARIABLE, ! Pascal File Variable
279 0339 2 ARG_LIST: VECTOR [4, LONG], ! Argument list
280 0340 2 PFV_ADDR: VOLATILE, ! Enable argument
281 0341 2 UNWIND_ACT: VOLATILE, ! Enable argument
282 0342 2 ERROR_ADDR: VOLATILE; ! Enable argument
283 0343 2
284 0344 2 BUILTIN
285 0345 2 ACTUALCOUNT; ! Count of arguments
286 0346 2
287 0347 2 ENABLE
288 0348 2 PASS$IO_HANDLER (PFV_ADDR, UNWIND_ACT, ERROR_ADDR); ! Enable error handler
289 0349 2
290 0350 2 !+
291 0351 2 ! Get ERROR parameter, if present.
292 0352 2 !-
293 0353 2
294 0354 2 IF ACTUALCOUNT () GEQU 5
295 0355 2 THEN
296 0356 2 ERROR_ADDR = .ERROR; ! Set unwind address
297 0357 2
298 0358 2 PFV_ADDR = PFV [PFV$R_PFV]; ! Set PFV address
299 0359 2
300 0360 2 !+
301 0361 2 ! Set up ARG_LIST.
302 0362 2 !-
303 0363 2
304 0364 2 ARG_LIST [0] = 3; ! Three arguments
305 0365 2 ARG_LIST [1] = PFV [PFV$R_PFV]; ! PFV address
306 0366 2 ARG_LIST [2] = .VALUE; ! Value to write
307 0367 2 ARG_LIST [3] = .TOTAL_WIDTH; ! Field width
308 0368 2
309 0369 2 !+
310 0370 2 ! Call PASS$DO_WRITEV to do the work, giving it the address of
311 0371 2 ! PASSWRITE_REALE_F to call.
312 0372 2 !-
313 0373 2
314 0374 2 PASS$DO_WRITEV (PFV [PFV$R_PFV], .MAX_LENGTH, STRING_LINE [0], ARG_LIST,
315 0375 2 PASSWRITE_REALE_F);
316 0376 2
317 0377 2 RETURN;
318 0378 2
319 0379 1 END; ! End of routine PASSWRITEV_REALE_F

```

				.EXTRN	PASS\$DO_WRITEV	
5E		28	007C 00000	.ENTRY	PASSWRITEV_REALE_F, Save R2,R3,R4,R5,R6	: 0281
		7E	C2 00002	SUBL2	#40, SP	: 0335
	04	AE	D4 00005	CLRL	ERROR_ADDR	
6D	0039	CF	7C 00007	CLRQ	UNWIND_ACT	
05		6C	DE 0000A	MOVAL	2\$, (FP)	
		04	91 0000F	CMPB	(AP), #5	: 0354
		04	1F 00012	BLSSU	1\$	
08	6E	AC	DO 00014	MOVL	ERROR, ERROR_ADDR	: 0356
	AE	1C	AE 9E 00018 1\$:	MOVAB	PFV, PFV_ADDR	: 0358

0C	AE		03	D0	0001D
10	AE	1C	AE	9E	00021
14	AE	0C	AC	7D	00026
	55	FF31	CF	9E	0002B
	54	0C	AE	9E	00030
	56	1C	AE	9E	00034
	53	08	AC	D0	00038
	52	04	AC	3C	0003C
		00000000G	00	16	00040
				04	00046
				0000	00047
	50	08	AC	D0	00049
	50	04	A0	D0	0004D
		D4	A0	9F	00051
		D8	A0	9F	00054
		DC	A0	9F	00057
			03	DD	0005A
			5E	DD	0005C
	7E	04	AC	7D	0005E
00000000G	00		03	FB	00062
				04	00069

```

MOV#3, ARG_LIST
MOVABPFV, ARG_LIST+4
MOVQVALUE, ARG_LIST+8
MOVABPASS$WRITE_REAL_F, R5
MOVABARG_LIST, R4
MOVABPFV, R6
MOVLSTRING LINE, R3
MOVZWLMAX_LENGTH, R2
JSBPASS$$DO_WRITEV
RET

.WORDSave nothing
MOVL8(AP), R0
MOVL4(R0), R0
PUSHABERROR_ADDR
PUSHABUNWIND_ACT
PUSHABPFV_ADDR
PUSHL#3
PUSHLSP
MOVQ4(AP), -(SP)
CALLSPASS$$IO_HANDLER
RET

```

0364
0365
0366
0374

0379
0335

```
; Routine Size: 106 bytes,    Routine Base: _PAS$CODE + 00A0
```

```

; 320          0380 1
; 321          0381 1 !<BLF/PAGE>

```

PASSWRITE_REALE Write an F_floating in E format
1-002

PASSWRITEV_REALE_F - Write F_floating in E form

C 5
16-Sep-1984 02:22:17
14-Sep-1984 12:52:06

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

Page 10
(5)

: 323 0382 1 END
: 324 0383 1
: 325 0384 0 ELUDOM

! End of module PASSWRITE_REALE_F

PSECT SUMMARY

Name	Bytes	Attributes
_PASSCODE	266	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\$:PASWRIREF/OBJ=OBJ\$:PASWRIREF MSRC\$:PASWRIREF/UPDATE=(ENH\$:PASWRIREF)

: 326 0385 0
: Size: 266 code + 0 data bytes
: Run Time: 00:07.0
: Elapsed Time: 00:18.1
: Lines/CPU Min: 3285
: Lexemes/CPU-Min: 12751
: Memory Used: 80 pages
: Compilation Complete

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PASWIREH
LIS

PASWITEL
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