

[illegible]

3

Sy

LI

LI
LI

LI

LI
LI

LI

LI
LILI
LILI
LI

LI

LI
LI

LI

LI
LI

LI

LI
LI

LI

11

LI

LI
LI

LI

LI
LI

21

LI

LI
LI

22

LI
LI

LI

LI
LI

LI

1

```

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
LL          II             SSSSSS
LLLLLLLLLLLL IIIIII          SSSSSSSS
LLLLLLLLLLLL IIIIII          SSSSSSSS

```


(2)	47	HISTORY	; Detailed current edit history
(3)	65	DECLARATIONS	
(5)	130	OTSSCVT_T_F	- convert text to F_floating
(7)	242	Initialization	
(8)	288	Main loop	
(11)	393	End-of-string processing	
(14)	560	Table of offsets to action routines	

```
0000 1      .TITLE OTSSCVTTF      ; Convert text to real (F only)
0000 2      .IDENT /1-005/        ; File: OTSCVTTF.MAR Edit: JCW1005
0000 3
0000 4 :
0000 5 :*****
0000 6 :
0000 7 :*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 :*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 :*  ALL RIGHTS RESERVED.
0000 10 :
0000 11 :*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 :*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 :*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 :*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 :*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 :*  TRANSFERRED.
0000 17 :
0000 18 :*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 :*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 :*  CORPORATION.
0000 21 :
0000 22 :*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 :*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 :
0000 25 :
0000 26 :*****
0000 27 :
0000 28 :
0000 29 :++
0000 30 : FACILITY: Language-independent support library
0000 31 :
0000 32 : ABSTRACT:
0000 33 :
0000 34 :     Performs conversion of character strings containing numbers to
0000 35 :     the F-floating data type. This routine supports Fortran F, E, D
0000 36 :     and G-format conversion, as well as similar types in other
0000 37 :     languages.
0000 38 :
0000 39 : ENVIRONMENT: Runs at any access mode, AST Reentrant
0000 40 :
0000 41 : AUTHOR: John A. Wheeler, 17-Apr-1981: Version 1
0000 42 :
0000 43 : MODIFIED BY:
0000 44 :
0000 45 :--
```


OTSS\$CVTTF
1-005

; Convert text to real (F only) F 8 16-SEP-1984 00:29:11 VAX/VMS Macro V04-00 Page 2
HISTORY ; Detailed current edit histor 6-SEP-1984 11:13:40 [LIBRTL.SRC]OTSCVTTF.MAR;1 (2)

```
0000 47      .SBTTL HISTORY                      ; Detailed current edit history
0000 48
0000 49 : EDIT HISTORY:
0000 50 :
0000 51 : 1-001 - Original. JAW 17-Apr-1981
0000 52 : 1-002 - Don't define HANDLER as global symbol. SBL 13-May-1981
0000 53 : 1-003 - Add call to OTSS$CVT_T_D to assure a correctly-rounded result
0000 54 :           in all cases. Also improve exception handler. JAW 12-Jul-1981
0000 55 : 1-004 - Add logic to handle tabs amid leading spaces. Add check for
0000 56 :           depth in handler. Add check of result of OTSS$CVT_T_D. Remove
0000 57 :           instruction which canceled handler. Note ASCII dependencies.
0000 58 :           JAW 23-Aug-1981
0000 59 : 1-005 - If ext_bits was not omitted from the parameter list, but was
0000 60 :           zero, a protection violation was caused by MOV B R5, aext_bits(AP).
0000 61 :           I found this bug while testing the routine to see what it would do.
0000 62 :           JCW 1-NOV-1983
0000 63 :
```

OT
1-

04

04

4E

04

```
0000 65      .SBTTL  DECLARATIONS
0000 66      :
0000 67      : LIBRARY MACRO CALLS:
0000 68      :
0000 69      $PSLDEF      : PSL$ symbols
0000 70      $CHFDEF     : CHF$ symbols
0000 71      $SSDEF      : SS$ symbols
0000 72      :
0000 73      : EXTERNAL SYMBOLS:
0000 74      :
0000 75      .DSABL  GBL      : Require all external symbols to be declare
0000 76      .EXTRN  OTSS$ INPCONERR
0000 77      .EXTRN  OTSS$CVT_T_D      : D_floating input conversion
0000 78      :
0000 79      :
0000 80      : MACROS:
0000 81      :
0000 82      : NONE
0000 83      :
0000 84      : PSECT DECLARATIONS:
0000 85      :
00000000 86      .PSECT  _OTSS$CODE      PIC, SHR, QUAD, EXE, NOWRT
0000 87      :
0000 88      : EQUATED SYMBOLS:
0000 89      :
0000 90      : Flags passed by the caller (relocated from bits 0:6 to bits 8:14, to
0000 91      : allow this routine's local flags to be referenced with BICB and BISB).
0000 92      :
0000 93      :
00000008 0000 94  V_SKIPBLANKS      = 8      : If set, blanks are ignored.
00000009 0000 95      : (Bit 0 for caller)
00000009 0000 96  V_ONLY_E          = 9      : If set, only E or e exponents
00000009 0000 97      : are allowed (BASIC+2, PL/I).
00000009 0000 98      : (Bit 1 for caller)
0000000A 0000 99  V_ERR_UFLO         = 10     : If set, underflow is an error.
0000000A 0000 100      : (Bit 2 for caller)
0000000B 0000 101  V_DONTROUND       = 11     : If set, don't round value.
0000000B 0000 102      : (Bit 3 for caller)
0000000C 0000 103  V_SKIPTABS        = 12     : If set, tabs are ignored.
0000000C 0000 104      : (Bit 4 for caller)
0000000D 0000 105  V_EXP_LETTER     = 13     : If set, an exponent must begin
0000000D 0000 106      : with a valid exponent letter.
0000000D 0000 107      : If clear, the exponent letter
0000000D 0000 108      : may be omitted.
0000000E 0000 109      : (Bit 5 for caller)
0000000E 0000 110  V_FORCESCALE     = 14     : If set, the scale factor is
0000000E 0000 111      : always applied. If clear, it
0000000E 0000 112      : is only applied if there is no
0000000E 0000 113      : exponent present in the string
0000000E 0000 114      : (Bit 6 for caller)
0000000E 0000 115
```


	0000	117	:	Local flags. Do not use bits 0, 1, 3 or 5, which get set when the	
	0000	118	:	sign character (^X2B or ^X2D) is OR'ed into R2. (ASCII dependency.)	
	0000	119	:		
00000002	0000	120	V_MINUS	= 2	: Minus sign seen--MUST BE BIT 2
00000004	0000	121	V_SIGN	= 4	: Sign (plus or minus) seen
00000006	0000	122	V_POINT	= 6	: Decimal point seen
00000007	0000	123	V_EXPO	= 7	: Exponent seen
	0000	124	:		
00000010	0000	125	M_SIGN	= 12V_SIGN	: Mask for V_SIGN
00000004	0000	126	M_MINUS	= 12V_MINUS	: Mask for V_MINUS
00000040	0000	127	M_POINT	= 12V_POINT	: Mask for V_POINT
	0000	128	:		

```
0000 130 .SBTTL OTSSCVT_T_F - convert text to F_floating
0000 131
0000 132 :++
0000 133 : FUNCTIONAL DESCRIPTION:
0000 134 :
0000 135 : OTSSCVT_T_F converts a text string containing a representation
0000 136 : of a numeric value to an F_floating representation of that
0000 137 : value. The routine supports FORTRAN F, E, D and G input type
0000 138 : conversion as well as similar types for other languages.
0000 139 :
0000 140 : The description of the text representation converted by
0000 141 : OTSSCVT_T_F is as follows:
0000 142 :
0000 143 : <0 or more blanks>
0000 144 : <"+" or nothing>
0000 145 : <0 or more decimal digits>
0000 146 : <"." or nothing>
0000 147 : <0 or more decimal digits>
0000 148 : <exponent or nothing, where exponent is:
0000 149 : <
0000 150 : <"E", "e", "D", "d", "Q", "q">
0000 151 : <0 or more blanks>
0000 152 : <"+" or "-" or nothing>>
0000 153 : or
0000 154 : <"+" or "-">>
0000 155 : <0 or more decimal digits>>
0000 156 : <end of string>
0000 157 :
0000 158 : Notes: 1. Unless the caller flag bit V_SKIPBLANKS
0000 159 : is set, blanks are equivalent to
0000 160 : decimal "0". If V_SKIPBLANKS is set,
0000 161 : blanks are always ignored.
0000 162 : 2. There is no difference in semantics
0000 163 : between any of the 6 valid exponent
0000 164 : letters.
0000 165 : 3. If the caller flag bit V_ONLY_E is set,
0000 166 : the only valid exponent letters are
0000 167 : "E" and "e"; any others will be treated
0000 168 : as an invalid character.
0000 169 : 4. If the caller flag bit V_SKIPTABS is set,
0000 170 : tab characters are ignored else they are
0000 171 : an error.
0000 172 : 5. If the caller flag bit V_EXP_LETTER is set,
0000 173 : the exponent, if present, must start with
0000 174 : a valid exponent letter, e.g. 1.2E32.
0000 175 : If clear, the exponent letter may be omitted,
0000 176 : e.g. 1.2+32.
0000 177 :
0000 178 : CALLING SEQUENCE:
0000 179 :
0000 180 : status.wlc.v = OTSSCVT_T_F (in_str.rt.dx, value.wf.r
0000 181 : [, digits_in_fract.rlu.v
0000 182 : [, scale_factor.rl.v
0000 183 : [, caller_flags.rlu.v,
0000 184 : [, ext_bits.wb.r]]])
```



```
0000 186 :  
0000 187 : INPUT PARAMETERS:  
0000 188 :  
00000004 0000 189      inp_str      = 4      ; Input string descriptor  
0000000C 0000 190      digits_in_fract = 12    ; If no decimal point is  
0000 191      ; present in input, specifies  
0000 192      ; how many digits are to be  
0000 193      ; treated as being to the right  
0000 194      ; of the decimal point.  
0000 195      ; If omitted, 0 is the default.  
0000 196  
00000010 0000 197      scale_factor = 16    ; Signed scale factor. If  
0000 198      ; present, and exponent absent,  
0000 199      ; the result value is divided by  
0000 200      ; 10**factor. If V_FORCESCALE  
0000 201      ; is set, the scale factor is  
0000 202      ; always applied.  
0000 203  
00000014 0000 204      flags      = 20    ; Flags supplied by caller  
0000 205      ;  
0000 206 : IMPLICIT INPUTS:  
0000 207 :  
0000 208 : NONE  
0000 209 :  
0000 210 : OUTPUT PARAMETERS:  
0000 211 :  
00000008 0000 212      value      = 8      ; Floating result by reference  
0000 213      ;  
00000018 0000 214      ext_bits   = 24    ; If present, the value will  
0000 215      ; NOT be rounded and the first  
0000 216      ; 8 bits after truncation will  
0000 217      ; be returned in this argument  
0000 218      ; as a byte. This value is  
0000 219      ; suitable for use as the  
0000 220      ; extension operand in an EMOD  
0000 221      ; instruction.  
0000 222 :  
0000 223 : IMPLICIT OUTPUTS:  
0000 224 :  
0000 225 : NONE  
0000 226 :  
0000 227 : COMPLETION CODES:  
0000 228 :  
0000 229 :  
0000 230      OTSS_INPCONERR - Error if illegal character in input, floating  
0000 231      ; overflow, or floating underflow (if enabled).  
0000 232 :  
0000 233      SSS_NORMAL    - Success  
0000 234 :  
0000 235 : SIDE EFFECTS:  
0000 236 :  
0000 237 : NONE  
0000 238 :  
0000 239 :  
0000 240 :--
```

```
0000 242 .SBTTL Initialization
0000 243
0000 244 ;+
0000 245 ; Register usage:
0000 246
0000 247 R0:R1 - Descriptor for input string
0000 248 R2 - flags (caller's and ours)
0000 249 R3 - digits in fraction
0000 250 R4:R5 - significand, as D-floating number
0000 251 R6 - next character
0000 252 R7 - utility register
0000 253 R8 - count of digits held in R9 (up to 9)
0000 254 R9 - digit accumulator
0000 255 R10 - count of significant digits seen
0000 256 ;+
0000 257 ;
0000 258 ;+
0000 259 ; NOTE: This routine contains specific dependencies on the ASCII
0000 260 ; character set, as noted in comments throughout.
0000 261 ;+
0000 262 ;
0000 263 .ENTRY OTSSCVT_T_F, ^M<R2, R3, R4, R5, R6, R7, R8, R9, R10>
0002 264 ;+
0002 265 ; Initialization
0002 266 ;+
0002 267 ;
0006 268 MOVQ @inp_str(AP), R0 ; R0:R1 = string descriptor.
0008 269 CLRQ R2 ; Flags and digits in fract = 0.
0008 270 CMPB (AP), #<flags/4> ; Is fifth argument present?
000B 271 BLSSU 10$, ; Branch if not, leaving R2 = 0.
000D 272 ASHL #8, flags(AP), R2 ; Get caller flags & clear ours.
0012 273 CLRD R4 ; R4:R5 (significand) = 0
0014 274 ;+
0014 275 ; Begin digit accumulation for fraction or exponent
0014 276 ;+
0014 277 ;
0014 278 CLRQ R8 ; Count and accumulator = 0
0016 279 CLRL R10 ; Significant digit count = 0
0018 280
0018 281 RESTRT: SKPC #^A/ /, R0, (R1) ; Skip blanks (clears LH(R0)).
001C 282 BBC #V_SKIPTABS, R2, NEXT ; Branch if tabs are illegal.
0020 283 MOVL R0, R7 ; Save character pointer in R7.
0023 284 SKPC #^A/ /, R0, (R1) ; Skip tabs if present.
0027 285 CMPL R0, R7 ; Were any tabs skipped?
002A 286 BNEQ RESTRT ; If so, try for more blanks.
```

50 04 BC 7D 0002 267
05 52 7C 0006 268
05 6C 91 0008 269
52 14 AC 05 1F 000B 270
08 78 000D 271
54 7C 0012 272 10\$:
0014 273
0014 274
0014 275
0014 276
58 7C 0014 278
5A D4 0016 279
0018 280
61 50 20 3B 0018 281
0C 52 0C E1 001C 282
57 50 D0 0020 283
61 50 09 3B 0023 284
57 50 D1 0027 285
EC 12 002A 286


```
002C 288 .SBTTL Main loop
002C 289
03 50 F4 002C 290 NEXT: SOBGEQ R0, 10$ ; Check for end of string.
00A1 31 002F 291 BRW END ; Branch at end.
56 81 98 0032 292 10$: CVTBL (R1)+, R6 ; Fetch next character.
12 19 0035 293 BLSS ERROR ; Error if between 128 and 255.
; (ASCII dependency.)
6B 56 06 E4 0037 294 BBSC #6, R6, LETTER ; Branch if between 64 and 127
; and map character to 0:63.
; (ASCII dependency.)
57 0271'CF46 9A 003B 295 MOVZBL W^OFFSET[R6], R7 ; Get branch displacement.
45'AF47 17 0041 296 JMP B^ACTION[R7] ; Dispatch to action routine.
0045 300
0045 301 ACTION:
0045 302
0045 303 ;+
0045 304 ; Tab
0045 305 ; -
0045 306
50 E3 52 0C E0 0045 307 TAB: BBS #V_SKIPTABS, R2, NEXT ; If tabs are legal, ignore.
00000000'8F D0 0049 308 ERROR: MOVL #OTSS_INPCONERR, R0 ; Else error.
08 BC D4 0050 309 CLRF avalue(AP) ; Store zero as result on error.
04 0053 310 RET ; Return to caller.
0054 311
0054 312 ;+
0054 313 ; Decimal point
0054 314 ; -
0054 315
F1 52 06 E2 0054 316 POINT: BBSS #V_POINT, R2, ERROR ; If 2nd decimal point, error.
53 D4 0058 317 CLRL R3 ; Reset digits_in_fraction.
32 11 005A 318 BRB DIGDON ; Go indicate sign now known.
005C 319
005C 320 ;+
005C 321 ; Blank
005C 322 ; -
005C 323
CC 52 08 E0 005C 324 BLANK: BBS #V_SKIPBLANKS, R2, NEXT ; If BLANK=NULL, ignore blank.
56 30 90 0060 325 MOVB #^A/0/, R6 ; Otherwise treat as a zero.
0063 326
0063 327 ;+
0063 328 ; Zero
0063 329 ; -
0063 330
5A B5 0063 331 ZERO: TSTW R10 ; Has significance started yet?
25 13 0065 332 BEQL NONSIG ; Branch if nonsignificant zero.
0067 333
0067 334 ;+
0067 335 ; Digit (1-9, or significant zero)
0067 336 ; -
0067 337
02 5A 26 F3 0067 338 DIGIT: AOBLEQ #38, R10, USE ; Are we beyond the 38th digit?
1F 11 006B 339 BRB NONSIG ; Yes: treat as nonsignificant.
12 58 09 F3 006D 340 USE: AOBLEQ #9, R8, ROOM ; Is there room for digit in R9?
D4 52 07 E0 0071 341 BBS #V_EXP0, R2, ERROR ; No. If exponent digit, error.
58 59 6E 0075 342 CVTLD R9, R8 ; Float R9 into R8:R9.
54 0220'CF 64 0078 343 MULD W^TEN9, R4 ; Accumulate a super-digit.
54 58 60 007D 344 ADDD R8, R4 ; ...
```

```

58 01 7D 0080 345      MOVQ    #1, R8      ; R8 (digit count) = 1
      0083 346      ; R9 (digit accumulator) = 0
      0083 347
59 59 6949 DE 0083 348 ROOM: MOVAL    (R9)[R9], R9 ; Multiply R9 by 10 and add the
59 D0 A649 3E 0087 349      MOVAV    -^X30(R6)[R9], R9 ; digit in R6 (strip zone bits).
      008C 350      ; (ASCII dependency.)
      53 D6 008C 351 NONSIG: INCL    R3      ; Count digit in fraction.
52 10 88 008E 352 DIGDON: BISB     #M SIGN, R2 ; Indicate sign now known.
      99 11 0091 353      BRB      NEXT      ; Go get next character.
      0093 354

```



```
0093 356
0093 357 :+
0093 358 : Sign (+ or -) (^X2B or ^X2D, respectively)
0093 359 :-
0093 360
05 52 04 E2 0093 361 SIGN: BBSS #V_SIGN, R2, SIGN2 : Branch if second sign.
52 56 88 0097 362 SIGN1: BISB R6, R2 : OR the sign character into R2.
009A 363 : (ASCII dependency.)
009A 364 : Go get next character.
A9 52 90 11 009A 364 SIGN2: BRB NEXT
0D E0 009C 365 SIGN2: BBS #V_EXP_LETTER, R2, ERROR : If letter is required, error.
50 D6 00A0 366 INCL R0 : Throw back the second sign,
51 D7 00A2 367 DECL R1 :
0E 11 00A4 368 BRB E : and pretend we saw an E.
00A6 369
00A6 370 :+
00A6 371 : Any character whose value is ^X40 or greater (normally a letter)
00A6 372 :-
00A6 373
09 C3'AF 56 E0 00A6 374 LETTER: BBS R6, B^LTRE, E : Branch if letter is E or e.
99 CB'AF 56 E1 00AB 375 BBS R6, B^LTRDQ, ERROR : Branch if letter is not DdQq.
95 52 09 E0 00B0 376 DQ: BBS #V_ONLY_E, R2, ERROR : If D or Q not allowed, error.
52 10 8A 00B4 377 E: BICB #M_SIGN, R2 : Allow sign again after letter.
8E 52 07 E2 00B7 378 EXPO: BBSS #V_EXPO, R2, ERROR : If second exponent, error.
23 11 00BB 379 BRB TERM : Go terminate the significand.
00BD 380 : Equivalent to BSB TERM.
52 04 8A 00BD 381 EXPO1: BICB #M_MINUS, R2 : Restore default sign (plus).
FF55 31 00C0 382 BRW RESTR : Go restart digit accumulation.
00C3 383
00C3 384 :+
00C3 385 : Bit vectors for use in identifying valid exponent letters.
00C3 386 :-
00C3 387
00000020 00000020 00C3 388 LTRE: .QUAD ^X0000002000000020 : Bits to identify E and e.
00CB 389 : (ASCII dependency.)
00020010 00020010 00CB 390 LTRDQ: .QUAD ^X0002001000020010 : Bits to identify D, Q, d, q.
00D3 391 : (ASCII dependency.)
```

```
00D3 393 .SBTTL End-of-string processing
00D3 394
09 52 07 E1 00D3 395 END: BBC #V_EXPO, R2, TERM ; Branch if no exponent seen.
42 52 02 E1 00D7 396 BBC #V_MINUS, R2, FINISH ; Is the exponent negative?
59 59 CE 00DB 397 MNEGL R9, R9 ; Yes: negate it.
3D 11 00DE 398 BRB FINISH ; Go finish up.
00E0 399
00E0 400 ;+
00E0 401 ; Terminate the significand. (The code between TERM and TERMX is also
00E0 402 ; "called" at EXPO.)
00E0 403 ;--
00E0 404
54 54 73 00E0 405 TERM: TSTD R4 ; Were there more than 9 digits?
OE 13 00E2 406 BEQL 10$ ; Branch if not.
56 59 6E 00E4 407 CVTLD R9, R6 ; Yes: float final super-digit.
54 54 64 00E7 408 MULD W*ENTAB[R8], R4 ; Make room in R4.
54 56 60 00ED 409 ADDD R6, R4 ; Accumulate final super-digit.
03 54 59 6E 00F0 410 BRB 20$ ; Continue.
52 52 02 E1 00F2 411 10$: CVTLD R9, R4 ; Float the significand.
54 54 72 00F5 412 20$: BBC #V_MINUS, R2, 30$ ; Was there a minus sign?
0B 52 06 E2 00F9 413 30$: MNEGL R4, R4 ; Yes: negate the result.
0100 414 30$: BBSS #V_POINT, R2, 50$ ; Was there a decimal point?
0100 415 ; Disallow second decimal point.
03 53 D4 0100 416 40$: CLRL R3 ; Set digits_in_fract to 0.
6C 91 0102 417 (AP), #<digits_in_fract/4> ; Is third argument present?
04 1F 0105 418 CMPB 50$ ; Branch if not.
53 0C AC D0 0107 419 MOVL digits_in_fract(AP), R3 ; Yes: use the argument.
5A 26 C2 010B 420 50$: SUBL #38, R10 ; Did we ignore any digits?
03 15 010E 421 BLEQ 60$ ; Branch if not.
53 5A C2 0110 422 60$: SUBL R10, R3 ; If so, reduce R3 accordingly.
53 DD 0113 423 PUSHL R3 ; Save digits_in_fract on stack.
58 7C 0115 424 60$: CLRQ R8 ; Count and accumulator = 0
5A D4 0117 425 CLRL R10 ; Significand digit count = 0
A0 52 07 E0 0119 426 TERMX: BBS #V_EXPO, R2, EXP01 ; Continue at EXP01 if V_EXPO=1.
011D 427 ; Otherwise continue at FINISH.
011D 428
011D 429
011D 430 ;+
011D 431 ; Here to finish up. The significand is in R4:R5 in D_floating, and the
011D 432 ; exponent is in R9 as an integer. Digits_in_fract is on the stack.
011D 433 ;--
011D 434
011D 435
54 73 011D 436 FINISH: TSTD R4 ; Is the significand zero?
6D 13 011F 437 BEQL STORE1 ; If so, no scaling is needed.
04 6C 91 0121 438 CMPB (AP), #<scale_factor/4> ; Is fourth argument present?
0C 1F 0124 439 BLSSU NOSF ; Branch if not.
04 52 0E E0 0126 440 BBS #V_FORCESCALE, R2, APSF ; Branch if scaling is required.
04 52 07 E0 012A 441 BBS #V_EXPO, R2, NOSF ; Branch if exponent is present.
59 10 AC C2 012E 442 APSF: SUBL scale_factor(AP), R9 ; Apply explicit scale factor.
0132 443 NOSF:
6D 0230'CF 9E 0132 444 MOVAB W*HANDLER, (FP) ; Establish exception handler.
04 52 0A E1 0137 445 BBC #V_ERR_UFLO, R2, 10$ ; Is underflow an error?
0040 8F BB 013B 446 BISPSW #PSLSM-FU ; Yes: enable it.
50 0A CE 013F 447 10$: MNEGL #10, R0 ; R0 = -10 (for use in ACBL)
59 8E C2 0142 448 SUBL (SP)+, R9 ; Apply digits in fraction.
0145 449
```



```
0145 451 :+
0145 452 : We now have the final scale factor (FSF) in R9.
0145 453 :-
0145 454
2D 13 0145 455 BEQL STORE ; If zero, no scaling is needed.
1F 14 0147 456 BGTR POSFSF ; Branch if FSF is positive.
59 59 CE 0149 457 MNEGL R9, R9 ; R9 = !FSF!
54 0228'CF 07 11 014C 458 BRB NEGFSF ; Continue with negative FSF.
FFF3 59 50 01 13 0153 459 NEGX: DIVD W^TEN10, R4 ; Divide R4 by 10**10.
54 0228'CF49 1F 13 0155 460 BEQL STORE ; If R4 goes to zero, quit.
54 0228'CF49 01 F1 0155 461 NEGFSF: ACBL #1, R0, R9, NEGX ; Deduct 10 from FSF and test.
54 0228'CF49 11 66 015B 462 DIVD W^TENTAB+^D80[R9], R4 ; Apply remainder of FSF.
11 11 0161 463 BRB STORE ; Continue.
54 0228'CF 64 0163 464
FFF5 59 50 01 F1 0168 465 POSX: MULD W^TEN10, R4 ; Multiply R4 by 10**10.
54 0228'CF49 64 016E 466 POSFSF: ACBL #1, R0, R9, POSX ; Deduct 10 from FSF and test.
0174 467 MULD W^TENTAB+^D80[R9], R4 ; Apply remainder of FSF.
0174 468 :+
0174 469 : Test the result to see whether rounding error could have affected the
0174 470 : 25th fraction bit, which is the rounding bit for the conversion to F.
0174 471 : A carry into this bit may have occurred if it is a one followed by a
0174 472 : string of zeroes, or is a zero followed by a string of ones. The
0174 473 : error cannot exceed 8 LSB, since a maximum of sixteen floating-point
0174 474 : instructions capable of introducing error have been executed.
0174 475 :-
57 55 000F0000 8F CB 0174 476 STORE: BICL3 #^X000F0000, R5, R7 ; Mask off low-order 4 bits.
FFF07FFF 8F 57 D1 017C 477 CMPL R7, #^XFFF07FFF ; Is rounding bit 0, others 1?
00008000 8F 2E 13 0183 478 BEQL CALLTD ; If so, call OTSS$CVT_T_D.
06 25 13 0185 479 CMPL R7, #^X00008000 ; Is rounding bit 1, others 0?
14 52 0A 1E 0191 480 BEQL CALLTD ; If so, call OTSS$CVT_T_D.
08 BC 54 76 0197 481 STORE1: CMPB (AP), #<ext_bits/4> ; Is sixth argument present?
12 11 019B 482 BGEQU EXTBIT ; Branch if so.
18 AC D5 019D 483 BBS #V_DONTROUND, R2, NORND ; If bit 3 is set, don't round.
55 55 F8 8F 78 01A2 484 CVTDF R4, @value(AP) ; Store rounded value.
18 BC 55 90 01A7 485 BRB RET1 ; And return.
08 BC 54 50 01AB 486 EXTBIT: TSTL ext_bits(AP) ; ext_bits might be zero.
50 01 04 01B2 487 BEQL NORND ; Skip this code if ext_bits(AP)=0.
01B3 493 ASHL #-8, R5, R5 ; Align next 8 bits in R5<0:7>.
MOV B 54 50 01AF 488 MOV B R5, @ext_bits(AP) ; Store extra precision bits.
01B3 493 MOV B 54 50 01AF 489 NORND: MOVF R4, @value(AP) ; Store unrounded value.
01B3 493 RET1: MOVL #$$$_NORMAL, R0 ; Indicate success.
01B3 493 RET ; Return to caller.
```

```
01B3 495 :+
01B3 496 : Here to call OTSSCVT_T_D. Build a list of five arguments, including
01B3 497 : any passed by the caller, on the stack.
01B3 498 :-
5E 08 C2 01B3 499 CALLTD: SUBL #8, SP ; Make room for double result.
01B6 500 ; (This specific location may or
01B6 501 ; may not be used.)
7E D4 01B6 502 CLRL -(SP) ; Provide default fifth arg.
7E 7C 01B8 503 CLRL -(SP) ; Provide default 3rd, 4th arg.
57 6C 9A 01BA 504 MOVZBL (AP), R7 ; Extend argument count.
6C47 DD 01BD 505 10$: PUSHL (AP)[R7] ; Copy each argument passed.
FA 57 F5 01C0 506 SOBGTR R7, 10$ ; Step to next argument.
04 AE 14 AE DE 01C3 507 MOVAL 20(SP), 4(SP) ; Substitute our result field.
00000000'GF 05 FB 01C8 508 CALLS #5, G^OTSSCVT_T_D ; Call double input conversion,
01CF 509 ; always passing five arguments.
03 50 E8 01CF 510 BLBS R0, 20$ ; If successful, proceed.
FE74 31 01D2 511 BRW ERROR ; Else return OTSS INPCONERR.
54 8E 7D 01D5 512 20$: MOVQ (SP)+, R4 ; Transfer result to R4:R5.
B4 11 01D8 513 BRB STORE1 ; Continue.
01DA 514
01DA 515 :+
01DA 516 : Table of powers of ten. This is a zero-origin table whose 0th entry
01DA 517 : is never referenced (since we never want to multiply/divide by 10**0).
01DA 518 :-
01DA 519
01DA 520 .ALIGN QUAD
00000000 000001D8 01E0 521 TENTAB=-8
00000000 00004220 01E0 522 .QUAD ^X000000000000004220 ; .DOUBLE 1.0E1
00000000 000043C8 01E8 523 .QUAD ^X0000000000000043C8 ; .DOUBLE 1.0E2
00000000 0000457A 01F0 524 .QUAD ^X00000000000000457A ; .DOUBLE 1.0E3
00000000 4000471C 01F8 525 .QUAD ^X000000004000471C ; .DOUBLE 1.0E4
00C00000 500048C3 0200 526 .QUAD ^X00000000500048C3 ; .DOUBLE 1.0E5
00000000 24004A74 0208 527 .QUAD ^X0000000024004A74 ; .DOUBLE 1.0E6
00000000 96804C18 0210 528 .QUAD ^X0000000096804C18 ; .DOUBLE 1.0E7
00000000 BC204DBE 0218 529 .QUAD ^X00000000BC204DBE ; .DOUBLE 1.0E8
00000000 6B284F6E 0220 530 TEN9: .QUAD ^X000000006B284F6E ; .DOUBLE 1.0E9
00000000 02F95115 0228 531 TEN10: .QUAD ^X0000000002F95115 ; .DOUBLE 1.0E10
0230 532
0230 533
0230 534 HANDLER:
0000 0230 .WORD ^M<> ; Save nothing
0232 535
0232 536 :+
0232 537 : The only anticipated exceptions are floating overflow and floating
0232 538 : underflow. Continue at ERROR in either case. Otherwise resignal.
0232 539 :-
50 04 AC D0 0232 540 MOVL CHF$SIGARGLIST(AP), R0 ; R0 = address of signal array
048C 8F 04 A0 B1 0236 541 CMPW CHF$SIG_NAME(R0), #SS$FLTUVF ; Overflow trap?
18 13 023C 542 BEQL ERR
049C 8F 04 A0 B1 023E 543 CMPW CHF$SIG_NAME(R0), #SS$FLTUND ; Underflow trap?
10 13 0244 544 BEQL ERR
04B4 8F 04 A0 B1 0246 545 CMPW CHF$SIG_NAME(R0), #SS$FLTUVF_F ; Overflow fault?
08 13 024C 546 BEQL ERR
04C4 8F 04 A0 B1 024E 547 CMPW CHF$SIG_NAME(R0), #SS$FLTUND_F ; Underflow fault?
13 12 0254 548 BNEQ RESIG
51 08 AC D0 0256 549 ERR: MOVL CHF$MCHARGLIST(AP), R1 ; R1 = addr of mechanism array
08 A1 D5 025A 550 TSTL CHF$MCH_DEPTH(R1) ; Is the depth zero?
0A 12 025D 551 BNEQ RESIG ; No: resignal.
```


OTSSCVTTF
1-005

; Convert text to real (F only)
End-of-string processing

E 9

16-SEP-1984 00:29:11 VAX/VMS Macro V04-00
6-SEP-1984 11:13:40 [LIBRTL.SRC]OTSCVTTF.MAR;1

Page 14
(13)

08 A0	FDE6 CF	9E 025F	552	MOVAB	ERROR, CHF\$SIG_NAME+4(R0) ; Change return PC to ERROR.
	50 01	D0 0265	553	MOVL	#SS\$_CONTINUE, R0 ; Continue.
		04 0268	554	RET	; Return.
			0269		
50	00000918 8F	D0 0269	556	RESIG: MOVL	#SS\$_RESIGNAL, R0 ; None of the above; resignal.
		04 0270	557	RET	; Return.
			0271		
			558		

```

                                0271 560      .SBTTL Table of offsets to action routines
                                0271 561
00000000 0271 562 HT=TAB-ACTION      ; Tab
00000004 0271 563 er=ERROR-ACTION    ; Error
0000000F 0271 564 DP=POINT-ACTION    ; Decimal point
00000017 0271 565 BL=BLANK-ACTION    ; Blank
0000001E 0271 566 ZR=ZERO-ACTION     ; Zero
00000022 0271 567 DG=DIGIT-ACTION    ; Digit (1-9)
0000004E 0271 568 SI=SIGN-ACTION     ; Sign (+ or -)
                                0271 569
                                0271 570 ; (ASCII dependency.)
                                0271 571
04 04 00 04 04 04 04 04 04 04 04 04 04 04 04 04 0271 572 OFFSET: .BYTE er,er,er,er,er,er,er,er,er,HT,er,er,er,er,er,er ; 00-0F
04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 027D 573      .BYTE er,er,er,er,er,er,er,er,er,er,er,er,er,er,er,er ; 10-1F
4E 04 04 04 04 04 04 04 04 04 04 04 04 04 04 04 0281 574      .BYTE BL,er,er,er,er,er,er,er,er,er,SI,er,SI,DP,er ; 20-2F
04 04 22 22 22 22 22 22 22 22 22 22 22 22 22 22 0291 575      .BYTE ZR,DG,DG,DG,DG,DG,DG,DG,DG,DG,er,er,er,er,er,er ; 30-3F
                                029D
                                02A1
                                02AD
                                02B1 576
                                02B1 577      .END

```


OTSSCVTTF
Symbol table

; Convert text to real (F only)

G 9

16-SEP-1984 00:29:11 VAX/VMS Macro V04-00
6-SEP-1984 11:13:40 [LIBRTL.SRC]OTSCVTTF.MAR;1

Page 16
(14)

ACTION	00000045	R	02
APSF	0000012E	R	02
BL	= 00000017		
BLANK	0000005C	R	02
CALLTD	000001B3	R	02
CHFSL_MCHARGLST	= 00000008		
CHFSL_MCH_DEPTH	= 00000008		
CHFSL_SIGARGLST	= 00000004		
CHFSL_SIG_NAME	= 00000004		
DG	= 00000022		
DIGDON	0000008E	R	02
DIGIT	00000067	R	02
DIGITS_IN_FRACT	= 0000000C		
DP	= 0000000F		
DQ	000000B0	R	02
E	000000B4	R	02
END	000000D3	R	02
ER	= 00000004		
ERR	00000256	R	02
ERROR	00000049	R	02
EXPO	000000B7	R	02
EXPO1	000000BD	R	02
EXTBIT	0000019D	R	02
EXT_BITS	= 00000018		
FINISH	0000011D	R	02
FLAGS	= 00000014		
HANDLER	00000230	R	02
HT	= 00000000		
INP_STR	= 00000004		
LETTER	000000A6	R	02
LTRDQ	000000CB	R	02
LTRE	000000C3	R	02
M_MINUS	= 00000004		
M_SIGN	= 00000010		
NEGFSF	00000155	R	02
NEGX	0000014E	R	02
NEXT	0000002C	R	02
NONSIG	0000008C	R	02
NORND	000001AB	R	02
NOSF	00000132	R	02
OFFSET	00000271	R	02
OTSSCVT-T-D	*****	X	00
OTSSCVT-T-F	00000000	RG	02
OTSS_INPCONERR	*****	X	00
POINT	00000054	R	02
POSFSF	00000168	R	02
POSX	00000163	R	02
PSLSM_FU	= 00000040		
RESIG	00000269	R	02
RESTR	00000018	R	02
RET1	000001AF	R	02
ROOM	00000083	R	02
SCALE_FACTOR	= 00000010		
SI	= 0000004E		
SIGN	00000093	R	02
SIGN1	00000097	R	02
SIGN2	0000009C	R	02

SSS_CONTINUE	= 00000001		
SSS_FLTOVF	= 00000048C		
SSS_FLTOVF_F	= 0000004B4		
SSS_FLTUND	= 00000049C		
SSS_FLTUND_F	= 0000004C4		
SSS_NORMAL	= 00000001		
SSS_RESIGNAL	= 000000918		
STORE	00000174	R	02
STORE1	0000018E	R	02
TAB	0C000045	R	02
TEN10	00000228	R	02
TEN9	00000220	R	02
TERM	000000E0	R	02
TERMX	00000119	R	02
USE	0000006D	R	02
VALUE	= 00000008		
V_DONTROUND	= 0000000B		
V_ERR_UFLO	= 0000000A		
V_EXP0	= 00000007		
V_EXP_LETTER	= 0000000D		
V_FORCESCALE	= 0000000E		
V_MINUS	= 00000002		
V_ONLY_E	= 00000009		
V_POINT	= 00000006		
V_SIGN	= 00000004		
V_SKIPBLANKS	= 00000008		
V_SKIPTABS	= 0000000C		
ZERO	00000063	R	02
ZR	= 0000001E		

OT
Sy
BA
CA
CH
CO
DE
DI
DO
ER
EX
FO
IN
OT
OT
RE
VA
VA
V-
V-
V-
V-
PS
--
-0
Ph
--
In
Co
Pa
Sy
Pa
Sy
Ps
Cr
As
Th
52
Th
38
0

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_OTSSCODE	000002B1 (689.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC QUAD

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.05	00:00:01.41
Command processing	106	00:00:00.30	00:00:03.19
Pass 1	217	00:00:03.63	00:00:17.03
Symbol table sort	0	00:00:00.53	00:00:03.82
Pass 2	111	00:00:01.07	00:00:06.35
Symbol table output	10	00:00:00.06	00:00:00.06
Psect synopsis output	3	00:00:00.01	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	480	00:00:05.65	00:00:31.88

The working set limit was 1200 pages.

31217 bytes (61 pages) of virtual memory were used to buffer the intermediate code.

There were 30 pages of symbol table space allocated to hold 535 non-local and 11 local symbols.

577 source lines were read in Pass 1, producing 14 object records in Pass 2.

10 pages of virtual memory were used to define 9 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	6

529 GETS were required to define 6 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS\$:OTSCVTTF/OBJ=OBJ\$:OTSCVTTF MSRC\$:OTSCVTTF/UPDATE=(ENH\$:OTSCVTTF)

0212 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

