

[illegible]

3

Sy

MT

MT
MT

MT

MT

MT
MTMT
MTMT
MTMT
MT

MT
MT

MT
MT

MT

MT

MT
MT

MI
MT

MT
MT

MT
MT

MT
MT

MT
MT

MT

111

227

MT

MT
MT

MT

MT

M1
M1MT
MT

M1

M1

10

1

1

100

10

1

```

  LL      IIIII
  LL      IIIII
  LL      II
  LL      II
  LL      II
  LL      II
  LL      II
  LL      II
  LL      II
  LL      II
  LL      II
  LL      II
  LL      II
  LL      II
  LLLLLLLLLL  IIIII
  LLLLLLLLLL  IIIII
  SSSSSSSSS
  SSSSSSSSS
  SS
  SS
  SS
  SS
  SSSSSS
  SSSSSS
  SS
  SS
  SS
  SS
  SSSSSSSS
  SSSSSSSS
  
```

```

(2)      59      HISTORY ; Detailed Current Edit History
(3)      99      DECLARATIONS ; Declarative Part of Module
(4)     139      MTH$TAN - Standard Single Precision Floating TAN
(5)     182      MTH$TAN_R5 - JSB entry point
(6)     233      MTH$TAN_R4 - JSB entry point
(7)     289      MTH$TAN - Standard Single Precision Floating TAN
(8)     332      MTH$TAN_R5 - JSB entry point
(9)     397      MTH$TAN_R4 - JSB entry point

```

MTH\$TAN
1-020

J 1
; Floating Point Tangent routine

16-SEP-1984 01:51:57 VAX/VMS Macro V04-00
6-SEP-1984 11:27:19 [MTHRTL.SRC]MTH\$TAN.MAR;1

Page 1
(1)

```
0000 1 .TITLE MTH$TAN ; Floating Point Tangent routine
0000 2 ; (TAN, TAND)
0000 3 .IDENT /1-020/ ; File: MTH$TAN.MAR EDIT:RNH1020
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 FACILITY: MATH LIBRARY
0000 30 ++
0000 31 ABSTRACT:
0000 32
0000 33 MTH$TAN is a function which returns the floating point tangent
0000 34 of its single precision floating point radian argument. The call is
0000 35 standard call-by-reference. It does a JSB to MTH$TAN_R5.
0000 36
0000 37 MTH$TAND is a function which returns the floating point tangent
0000 38 of its single precision floating point degree argument. The call is
0000 39 standard call-by-reference. It does a JSB to MTH$TAND_R5.
0000 40
0000 41 MTH$TAN_R5, and MTH$TAND_R5 are JSB entry points that JSB to MTH$SINCOS_R5
0000 42 and MTH$SINCOSD_R5 respectively. MTH$TAN_R4, and MTH$TAND_R4 cannot use
0000 43 the above two routines because they are _R4 routines, so they JSB to
0000 44 MTH$SIN_R4, MTH$COS_R4, and MTH$SIND_R4, MTH$COSD_R4 routines.
0000 45
0000 46 --
0000 47
0000 48 VERSION: 1
0000 49
0000 50 HISTORY:
0000 51 AUTHOR:
0000 52 Peter Yuo, 29-Jun-77: Version 01
0000 53
0000 54 MODIFIED BY:
0000 55
0000 56
0000 57
```

```
0000 59 .SBTTL HISTORY ; Detailed Current Edit History
0000 60
0000 61
0000 62 ; ALGORITHMIC DIFFERENCES FROM FP-11/C ROUTINE: none
0000 63
0000 64 ; The result is reserved operand when COS(X) = 0, instead
0000 65 ; of largest or smallest representable floating number.
0000 66
0000 67 ; Edit History
0000 68
0000 69 ; 01-2 Error handling mechanism changed. Instead of having
0000 70 ; MTH$FLG_JACKET at the entrance, MTH$$ERROR_CON is stored on the
0000 71 ; top of the stack frame so when error happened in MTH$$SIN or MTH$COS
0000 72 ; the message will be hid away, but will get signalled in MTH$TAN.
0000 73 ; 01-3 The call to MTH$ERROR_CON is changed to MTH$$SIGNAL_CON
0000 74 ; 0-4 MTH$$ERROR changed to MTH$$SIGNAL.
0000 75 ; MTH$... changed to MTH_....
0000 76 ; Changed error handling mechanism. Put error result in R0 before
0000 77 ; calling MTH$$SIGNAL in order to allow user modify error result.
0000 78 ; 1-005 - Put version number in standard format (three digits of edit
0000 79 ; number) and update copyright notice. JBS 16-NOV-78
0000 80 ; 1-006 - Change MTH_FLOOVEMAT to MTH$K_FLOOVEMAT. JBS 07-DEC-78
0000 81 ; 1-007 - Remove $SRMDEF macro - not needed. JBS 16-DEC-78
0000 82 ; 1-008 - Add "" to the PSECT directive. JBS 22-DEC-78
0000 83 ; 1-009 - Fix error handling and detection. SBL 02-Feb-79
0000 84 ; 1-010 - Declare externals. SBL 17-May-1979
0000 85 ; 1-011 - Add JSB entry point. JBS 16-AUG-1979
0000 86 ; 1-012 - Make external references longword, and remove MTH$$SIGNAL_CON
0000 87 ; when doing our own signal. JBS 16-AUG-1979
0000 88 ; 1-013 - Correct a typo in edit 011. JBS 17-AUG-1979
0000 89 ; 1-014 - Have CALL entry JSB to JSB entry. Use correct signalling
0000 90 ; technique for JSB entry. SBL 31-Oct-1979
0000 91 ; 1-015 - Reduce argument limit to 2**30 to match SIN/COS. SBL 2-Nov-1979
0000 92 ; 1-016 - Added degree entry points. RNH 8-MAR-1981
0000 93 ; 1-017 - Undo edit 1-015. SIN/COS can now accept this argument limit. RNH 26-AUG-8
0000 94 ; 1-018 - Add MTH$TAN_R5, and MTH$TAND_R5. Rearrange the routine for simplicity.
0000 95 ; RNH 27-AUG-81.
0000 96 ; 1-019 - Change external references from W^ to G^. RNH 06-Oct-81
0000 97 ; 1-020 - Missed a W^. RNH 08-Oct-81
```

```
0000 99      .SBTTL  DECLARATIONS      ; Declarative Part of Module
0000 100
0000 101 :
0000 102 : INCLUDE FILES:
0000 103 :
0000 104 :
0000 105 :
0000 106 : EXTERNAL SYMBOLS:
0000 107 :
0000 108      .DSABL  GBL
0000 109      .EXTRN  MTH$SINCOS_R5
0000 110      .EXTRN  MTH$SINCOSD_R5
0000 111      .EXTRN  MTH$SIN_R4
0000 112      .EXTRN  MTH$COS_R4
0000 113      .EXTRN  MTH$K_FCOOVMAT
0000 114      .EXTRN  MTH$SSIGNAL
0000 115      .EXTRN  MTH$K_FLOUNDMAT
0000 116      .EXTRN  MTH$JACKET_TST
0000 117      .EXTRN  MTH$SIND_R4
0000 118      .EXTRN  MTH$COSD_R4
0000 119
0000 120 :
0000 121 : EQUATED SYMBOLS:
2EE10365 0000 122      F_SMALLEST_DEG = ^X2EE10365
0000 123 :
0000 124 : MACROS:
0000 125      $SFDEF                      ; Define SF (Stack Frame) symbols
0000 126 :
0000 127 :
0000 128 : PSECT DECLARATIONS:
0000 129
00000000 130      .PSECT  _MTH$CODE      PIC,SHR, LONG, EXE, NOWRT
0000 131                      ; program section for math routines
0000 132 :
0000 133 : OWN STORAGE: none
0000 134 :
0000 135 : CONSTANTS:
0000 136
00000004 0000 137      X = 4                      ;Position of argument from AP.
```

```
0000 139      .SBTTL MTH$TAN - Standard Single Precision Floating TAN
0000 140
0000 141
0000 142 :++
0000 143 : FUNCTIONAL DESCRIPTION:
0000 144 :
0000 145 : TAN - single precision floating point function
0000 146 :
0000 147 : For algorithm, see MTH$TAN_R5.
0000 148 :
0000 149 : CALLING SEQUENCE:
0000 150 :
0000 151 : TAN.wf.v = MTH$TAN(X.rf.r)
0000 152 :
0000 153 : INPUT PARAMETERS:
0000 154 :
0000 155 : X.rf.r      Address of value of angle in radians.
0000 156 :
0000 157 : IMPLICIT INPUTS: none
0000 158 :
0000 159 : OUTPUT PARAMETERS:
0000 160 :
0000 161 : VALUE: floating tangent of the argument
0000 162 :
0000 163 : IMPLICIT OUTPUTS: none
0000 164 :
0000 165 : COMPLETION CODES: none
0000 166 :
0000 167 : SIDE EFFECTS:
0000 168 :
0000 169 : NONE
0000 170 :
0000 171 : ---
0000 172 :
0000 173
403C 0000 174      .ENTRY MTH$TAN, ^M<IV, R2, R3, R4, R5>
0002 175      ; standard call-by-reference entry
0002 176      ; disable DV (and FU), enable IV
0002 177      MTH$FLAG_JACKET
0002
6D 00000000'GF 9E 0002      MOVAB G^MTH$$JACKET_HND, (FP)
0009      ; set handler address to jacket
0009      ; handler
0009
50 04 BC 50 0009 178      MOVF @X(AP), R0      ; R0 = argument
01 10 000D 179      BSBB MTH$TAN_R5      ; Get the tangent
04 000F 180      RET      ; Return with result in R0
```

```
0010 182 .SBTTL MTH$TAN_R5 - JSB entry point
0010 183
0010 184 :++
0010 185 : FUNCTIONAL DESCRIPTION
0010 186 :
0010 187 : TAN - single precision floating point function
0010 188 :
0010 189 : Algorithmic steps:
0010 190 :
0010 191 : 1. Compute SIN and COS in one JSB. Neither computation should fail.
0010 192 : 2. If COS is zero, error MTH$_FLOOVEMAT and return with reserved operand.
0010 193 : 3. Return SIN / COS.
0010 194 :
0010 195 : CALLING SEQUENCE:
0010 196 :
0010 197 :     MOVF    argument, R0
0010 198 :     JSB     MTH$TAN_R5
0010 199 :
0010 200 : INPUT PARAMETERS:
0010 201 :
0010 202 :     R0 contains x
0010 203 :
0010 204 : OUTPUT PARAMETERS:
0010 205 :
0010 206 :     NONE
0010 207 :
0010 208 : IMPLICIT OUTPUTS:
0010 209 :
0010 210 :     NONE
0010 211 :
0010 212 : RESULT VALUE:
0010 213 :
0010 214 :     The tangent of x
0010 215 :
0010 216 : SIDE EFFECTS:
0010 217 :
0010 218 :     NONE
0010 219 :
0010 220 : --
0010 221 MTH$TAN_R5::
0010 222 JSB     MTH$_SINCOS_R5
0010 223 TSTF    R1
0010 224 BEQL    20$
0010 225 DIVF2   R1, R0
0010 226 RSB
0010 227 :
0010 228 : Branch to common error code
0010 229 :
0010 230 20$:
0010 231 BRW     COSZER

00000000'EF 16 0010 222 JSB     MTH$_SINCOS_R5      ; Compute SIN(X), and COS(x)
51 53 0016 223 TSTF    R1                ; Is COS(X) EQL 0 ?
04 13 0018 224 BEQL    20$              ; If so, error
50 51 46 001A 225 DIVF2   R1, R0         ; Compute SIN(x) / COS(x)
05 001D 226 RSB                    ; Return to caller
001E 227 :
001E 228 : Branch to common error code
001E 229 :
001E 230 20$:
00C4 31 001E 231 BRW     COSZER          ;
```

```
0021 233 .SBTTL MTH$TAN_R4 - JSB entry point
0021 234
0021 235 :++
0021 236 : FUNCTIONAL DESCRIPTION
0021 237 :
0021 238 : TAN - single precision floating point function
0021 239 :
0021 240 : Algorithmic steps:
0021 241 :
0021 242 : 1. Compute SIN, and then COS. Neither computation should fail.
0021 243 : 2. If COS is zero, error MTH$_FLOOVEMAT and return with reserved operand.
0021 244 : 3. Return SIN / COS.
0021 245 :
0021 246 : CALLING SEQUENCE:
0021 247 :
0021 248 :     MOVF    argument, R0
0021 249 :     JSB     MTH$TAN_R4
0021 250 :
0021 251 : INPUT PARAMETERS:
0021 252 :
0021 253 :     R0 contains x
0021 254 :
0021 255 : OUTPUT PARAMETERS:
0021 256 :
0021 257 :     NONE
0021 258 :
0021 259 : IMPLICIT OUTPUTS:
0021 260 :
0021 261 :     NONE
0021 262 :
0021 263 : RESULT VALUE:
0021 264 :
0021 265 :     The tangent of x
0021 266 :
0021 267 : SIDE EFFECTS:
0021 268 :
0021 269 :     NONE
0021 270 :
0021 271 :--
0021 272 MTH$TAN_R4::
0021 273     PUSHL   R0
0021 274     JSB     MTH$COS_R4
0029 275     MOVF   R0, -(SP)
002C 276     BEQL   20$,
002E 277     MOVL   4(SP), R0
0032 278     JSB     MTH$SIN_R4
0038 279     DIVF2   (SP)+, R0
003B 280     ADDL2   #4, SP
003E 281     RSB
003F 282 :+
003F 283 : Restore stack, and go to common error code.
003F 284 :-
003F 285 20$:
003F 286     ADDL2   #8, SP
0042 287     BRW    COSZER

00000000 50 DD 0021 273
7E 50 50 0023 274
11 13 0029 275
50 04 AE D0 002C 276
00000000 EF 16 002E 277
50 8E 46 0032 278
5E 04 C0 0038 279
05 003B 280
003E 281
003F 282
003F 283
003F 284
003F 285
5E 08 C0 003F 286
00A0 31 0042 287
```

; entry point
; Save argument
; Compute COS(x)
; Put on stack and test for zero
; If so, error
; Get argument back
; Compute SIN(x)
; Compute SIN(x) / COS(x)
; Remove argument from stack
; Return to caller

; Discard COS and argument
; Go to common error code

```
0045 289 .SBTTL MTH$TAND - Standard Single Precision Floating TAN
0045 290
0045 291
0045 292 :++
0045 293 : FUNCTIONAL DESCRIPTION:
0045 294 :
0045 295 : TAND - Single precision floating point function
0045 296 :
0045 297 : For algorithm, see MTH$TAND_R5.
0045 298
0045 299 : CALLING SEQUENCE:
0045 300 :
0045 301 : TAND.wf.v = MTH$TAND(X.rf.r)
0045 302 :
0045 303 : INPUT PARAMETERS:
0045 304 :
0045 305 : X.rf.r ;Address of value of angle in degrees.
0045 306 :
0045 307 : IMPLICIT INPUTS: none
0045 308 :
0045 309 : OUTPUT PARAMETERS:
0045 310 :
0045 311 : VALUE: floating tangent of the argument
0045 312 :
0045 313 : IMPLICIT OUTPUTS: none
0045 314 :
0045 315 : COMPLETION CODES: none
0045 316 :
0045 317 : SIDE EFFECTS:
0045 318 :
0045 319 : NONE
0045 320 :
0045 321 :---
0045 322 :
0045 323 :
403C 0045 324 .ENTRY MTH$TAND, ^M<IV, R2, R3, R4, R5>
0047 325 ; standard call-by-reference entry
0047 326 ; disable DV (and FU), enable IV
0047 327
0047 MTH$FLAG_JACKET
0047
6D 00000000'GF 9E 0047 MOVAB G^MTH$$JACKET_HND, (FP)
004E ; set handler address to jacket
004E ; handler
004E
50 04 BC 50 004E 328 MOVF @X(AP), R0 ; R0 = argument
01 10 0052 329 BSBB MTH$TAND_R5 ; Get the tangent
04 0054 330 RET ; Return with result in R0
```

```
0055 332 .SBTTL MTH$TAN_R5 - JSB entry point
0055 333
0055 334 :++
0055 335 : FUNCTIONAL DESCRIPTION
0055 336 :
0055 337 : TAND - Single precision floating point function
0055 338 :
0055 339 : Algorithmic steps:
0055 340 :
0055 341 : 1. Check for argument too small for SIND. If so, return zero, and
0055 342 : signal floating point underflow if enabled.
0055 343 : 2. Compute SIND and COSD in one JSB. Neither computation should fail.
0055 344 : 3. If COSD is zero, error MTH$_FLOOVEMAT and return with reserved operand.
0055 345 : 4. Return SIND / COSD.
0055 346 :
0055 347 : CALLING SEQUENCE:
0055 348 :
0055 349 : MOVF argument, R0
0055 350 : JSB MTH$TAN_R5
0055 351 :
0055 352 : INPUT PARAMETERS:
0055 353 :
0055 354 : R0 contains x
0055 355 :
0055 356 : OUTPUT PARAMETERS:
0055 357 :
0055 358 : NONE
0055 359 :
0055 360 : IMPLICIT OUTPUTS:
0055 361 :
0055 362 : NONE
0055 363 :
0055 364 : RESULT VALUE:
0055 365 :
0055 366 : The tangent of x
0055 367 :
0055 368 : SIDE EFFECTS:
0055 369 :
0055 370 :
0055 371 :
0055 372 : Signal MTH$_FLOUNDMAT if !x! < 180/pi*2**-128
0055 373 :--
0055 374 :
0055 375 :
0055 376 MTH$TAN_R5::
0055 377 BICL3 #^X8000, R0, R1 ; R1 = !X!
0055 378 CMPW #^X380, R1 ; Compare with 2**-121
0055 379 BGEQ 30$ ; No underflow possible
0055 380 CMPF #F SMALLEST_DEG, R1 ; Better test. Compare
0055 381 BLSS 30$ ; !X! with 180/pi*2**-128
0055 382 TSTL R0 ; Check for zero
0055 383 BNEQ UNFL ; ARG too small and not 0
0055 384 RSB ; Return R0 = 0
0055 385 :+
0072 386 : We now know that MTH$SINCOSD_R5 routine will not fail, and that the
0072 387 : divide following them will not fail.
0072 388 :-
```

```
51 50 00008000 8F CB
51 0380 8F B1
0E 18 0062
51 2EE10365 8F 51 0064
05 19 006B
50 D5 006D
50 12 006F
05 0071
```

MTH\$TAN
1-020

: Floating Point Tangent routine
MTH\$TAN_R5 - JSB entry point

E 2

16-SEP-1984 01:51:57
6-SEP-1984 11:27:19

VAX/VMS Macro V04-00
[MTHRTL.SRC]MTH\$TAN.MAR;1

Page 9
(8)

00000000	'Ef	16	0072	389	30\$:	JSB	MTH\$SINCOSD_R5	:	Compute SIND(X) and COSD(X)
	51	53	0072	390		TSTF	R1	:	Is COSD(X) EQL 0 ?
	69	13	0078	391		BEQL	COSZER	:	If so, error
50	51	46	007A	392		DIVF2	R1, R0	:	Compute SIND(x) / COSD(x)
		05	007C	393		RSB		:	Return to caller
			007F	394					
			0080	395					

```
0080 397      .SBTTL MTH$TAN_R4 - JSB entry point
0080 398
0080 399 :++
0080 400 : FUNCTIONAL DESCRIPTION
0080 401 :
0080 402 : TAND - Single precision floating point function
0080 403 :
0080 404 : Algorithmic steps:
0080 405 :
0080 406 : 1. Check for argument too small for SIND. If so, return zero, and
0080 407 : signal floating point underflow if enabled.
0080 408 : 2. Compute SIND, and then COSD. Neither computation should fail.
0080 409 : 3. If COSD is zero, error MTH$_FLOOVEMAT and return with reserved operand.
0080 410 : 4. Return SIND / COSD.
0080 411 :
0080 412 : CALLING SEQUENCE:
0080 413 :
0080 414 :     MOVF    argument, R0
0080 415 :     JSB     MTH$TAN_R4
0080 416 :
0080 417 : INPUT PARAMETERS:
0080 418 :
0080 419 :     R0 contains x
0080 420 :
0080 421 : OUTPUT PARAMETERS:
0080 422 :
0080 423 :     NONE
0080 424 :
0080 425 : IMPLICIT OUTPUTS:
0080 426 :
0080 427 :     NONE
0080 428 :
0080 429 : RESULT VALUE:
0080 430 :
0080 431 :     The tangent of x
0080 432 :
0080 433 : SIDE EFFECTS:
0080 434 :
0080 435 :
0080 436 :
0080 437 :     Signal MTH$_FLOUNDMAT if |x| < 180/pi*2**-128
0080 438 :--
0080 439
0080 440
0080 441 MTH$TAN R4::      ; entry point
0080 442      BICL3    #^X8000, R0, R1      ; R1 = |x|
0080 443      CMPW     #^X380, R1           ; Compare with 2**-121
0080 444      BGEQ    30$                 ; No underflow possible
0080 445      CMPF     #F_SMALLEST_DEG, R1 ; Better test. Compare
0080 446      BLSS    30$                 ; |x| with 180/pi*2**-128
0080 447      TSTL    R0                  ; Check for zero
0080 448      BNEQ    UNFL                 ; ARG too small and not 0
0080 449      RSB
0080 450 :+
0080 451 : We now know that the SIND and COSD routines will not fail, and that the
0080 452 : divide following them will not fail.
0080 453 :-

51  50  00008000 8F  CB 0080 441 MTH$TAN R4::      ; entry point
51  51  0380 8F  B1 0080 442      BICL3    #^X8000, R0, R1      ; R1 = |x|
51  51  2EE10365 8F  18 0080 443      CMPW     #^X380, R1           ; Compare with 2**-121
51  51  2EE10365 8F  51 0080 444      BGEQ    30$                 ; No underflow possible
51  51  2EE10365 8F  19 0080 445      CMPF     #F_SMALLEST_DEG, R1 ; Better test. Compare
51  51  2EE10365 8F  05 0080 446      BLSS    30$                 ; |x| with 180/pi*2**-128
51  51  2EE10365 8F  D5 0080 447      TSTL    R0                  ; Check for zero
51  51  2EE10365 8F  12 0080 448      BNEQ    UNFL                 ; ARG too small and not 0
51  51  2EE10365 8F  05 0080 449      RSB
51  51  2EE10365 8F  05 0080 450 :+
51  51  2EE10365 8F  05 0080 451 : We now know that the SIND and COSD routines will not fail, and that the
51  51  2EE10365 8F  05 0080 452 : divide following them will not fail.
51  51  2EE10365 8F  05 0080 453 :-
```

```
00000000 50 DD 009D 454 30$: PUSHL R0 ; Save argument
7E 50 EF 16 009F 455 JSB MTH$COSD R4 ; Compute COSD(x)
11 50 00A5 456 MOVF R0, -(SP) ; Put on stack and test for zero
50 04 AE D0 0CAA 457 BEQL 20$ ; If so, error
00000000 EF 16 00AE 458 MOVL 4(SP), R0 ; Get argument back
50 8E 46 00B4 459 JSB MTH$SIND R4 ; Compute SIND(x)
5E 04 C0 00B7 460 DIVF2 (SP)+, R0 ; Compute SIND(x) / COSD(x)
05 00BA 461 ADDL2 #4, SP ; Remove argument from stack
00BB 462 RSB ; Return to caller
00BB 463 ;+
00BB 464 ; Restore stack, and go to common error code.
00BB 465 ;-
00BB 466 20$:
5E 08 C0 00BB 467 ADDL2 #8, SP ; Discard COSD and argument
0024 31 00BE 468 BRW COSZER ; Go to common error code
```

```
00C1 470 :  
00C1 471 : COMMON ERROR PATHS  
00C1 472 :  
00C1 473 :  
00C1 474 :  
00C1 475 :  
00C1 476 : Come here if underflow; signal error if FU is set. Always return 0.0  
00C1 477 :  
00C1 478 UNFL:  
00000000'GF 52 DC 00C1 479 MOVPSL R2 ; R2 = user's or jacket routine's PSL  
00 00 FB 00C3 480 CALLS #0, G^MTH$$JACKET_TST ; R0 = TRUE if JSB from jacket routine  
52 04 50 E9 00CA 481 BLBC R0, 10$ ; branch if user did JSB  
04 AD 3C 00CD 482 MOVZWL SF$W_SAVE_PSW(FP), R2 ; get user PSL saved by CALL  
50 D4 00D1 483 10$: CLRL R0 ; R0 = result. LIB$SIGNAL will save in  
00D3 484 ; CHF$L_MCH_R0/R1 so any handler can fixup  
0D 52 06 E1 00D3 485 BBC #6, R2, 20$ ; has user enabled floating underflow?  
6E DD 00D7 486 PUSHL (SP) ; yes, return PC from special routine  
7E 00'8F 9A 00D9 487 MOVZBL #MTH$K_FLOUNDMAT, -(SP) ; trap code for hardware floating underflow  
00DD 488 ; convert to MTH$FLOUNDMAT (32-bit VAX-11  
00DD 489 ; exception code)  
00000000'GF 02 FB 00DD 490 CALLS #2, G^MTH$$SIGNAL ; signal (condition, PC)  
05 00E4 491 20$: RSB ; return  
00E5 492  
00E5 493 ;+  
00E5 494 : Come here if COS(X) or COSD(X) is zero. This means that TAN(X) is infinite.  
00E5 495 :-  
00E5 496 COSZER:  
7E 00'8F 6E DD 00E5 497 PUSHL (SP) ; User "call" PC  
01 0F 9A 00E7 498 MOVZBL #MTH$K_FLOOVEMAT, -(SP) ; Condition value  
50 01 78 00EB 499 ASHL #15, #T, R0 ; R0 = reserved operand  
00000000'GF 02 FB 00EF 500 CALLS #2, G^MTH$$SIGNAL ; Signal the error  
05 00F6 501 RSB ; Return to caller.  
00F7 502  
00F7 503 .END
```

MTH\$TAN
Symbol table

; Floating Point Tangent routine 1 2

16-SEP-1984 01:51:57 VAX/VMS Macro V04-00
6-SEP-1984 11:27:19 [MTHRTL.SRC]MTH\$TAN.MAR;1

Page 13
(10)

```
COSZER      000000E5 R    02
F_SMALLEST DEG 2EE10365
MTH$$JACKET_HND ***** X    02
MTH$$JACKET_TST ***** X    00
MTH$$SIGNAL    ***** X    00
MTH$COSD_R4    ***** X    00
MTH$COS_R4     ***** X    00
MTH$K_FCOOVEMAT ***** X    00
MTH$K_FLOUNDMAT ***** X    00
MTH$$INCOSD_R5 ***** X    00
MTH$$INCOS_R5  ***** X    00
MTH$$IND_R4    ***** X    00
MTH$$IN_R4     ***** X    00
MTH$TAN       00000000 RG   02
MTH$TAND      00000045 RG   02
MTH$TAND_R4   00000080 RG   02
MTH$TAND_R5   00000055 RG   02
MTH$TAN_R4    00000021 RG   02
MTH$TAN_R5    00000010 RG   02
SF$W_SAVE_PSW = 00000004
UNFL          000000C1 R    02
X             = 00000004
```

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

PSECT name	Allocation	PSECT No.	Attributes
. ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$AB\$\$	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_MTH\$CODE	000000F7 (247.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.06	00:00:00.72
Command processing	112	00:00:00.71	00:00:03.39
Pass 1	126	00:00:01.60	00:00:05.85
Symbol table sort	0	00:00:00.04	00:00:00.06
Pass 2	94	00:00:01.15	00:00:05.70
Symbol table output	3	00:00:00.03	00:00:00.03
Psect synopsis output	3	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	369	00:00:03.63	00:00:15.79

The working set limit was 900 pages.
8464 bytes (17 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 50 non-local and 7 local symbols.
563 source lines were read in Pass 1, producing 16 object records in Pass 2.
9 pages of virtual memory were used to define 8 macros.

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

Macro library name

Macros defined

_S255\$DUA28:[SYSLIB]STARLET.MLB;2

4

88 GETS were required to define 4 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS\$:MTH\$TAN/OBJ=OBJ\$:MTH\$TAN MSRC\$:MTHJACKET/UPDATE=(ENH\$:MTHJACKET)+MSRC\$:MT

0264 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY