


```

MM      MM      TTTTTTTTTT  HH      HH      SSSSSSSS  IIIIII  NN      NN      CCCCCCCC  000000  SSSSSSSS
MM      MM      TTTTTTTTTT  HH      HH      SSSSSSSS  IIIIII  NN      NN      CCCCCCCC  000000  SSSSSSSS
MMMM    MMMM    TT          HH      HH      SS          II      NN      NN      CC          00      00      SS
MMMM    MMMM    TT          HH      HH      SS          II      NN      NN      CC          00      00      SS
MM      MM      TT          HH      HH      SS          II      NNNN   NN      CC          00      00      SS
MM      MM      TT          HH      HH      SS          II      NNNN   NN      CC          00      00      SS
MM      MM      TT          HHHHHHHHHH  SSSSSS   II      NN      NN      CC          00      00      SSSSSS
MM      MM      TT          HHHHHHHHHH  SSSSSS   II      NN      NN      CC          00      00      SSSSSS
MM      MM      TT          HH      HH      SS          II      NN      NNNN   CC          00      00      SS
MM      MM      TT          HH      HH      SS          II      NN      NNNN   CC          00      00      SS
MM      MM      TT          HH      HH      SS          II      NN      NN      CC          00      00      SS
MM      MM      TT          HH      HH      SSSSSSSS  IIIIII  NN      NN      CCCCCCCC  000000  SSSSSSSS
MM      MM      TT          HH      HH      SSSSSSSS  IIIIII  NN      NN      CCCCCCCC  000000  SSSSSSSS

```

```

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

```

(2)	82	HISTORY	: Detailed Current Edit History
(3)	158	DECLARATIONS	- Declarative Part of Module
(5)	216	COEFFICIENT TABLES	- Series Coefficients
(6)	310	MTH\$SINCOS	- Radian arguments
(8)	375	MTH\$SIN	
(8)	392	MTH\$COS	
(9)	408	MTH\$SINCOSD	- Degrees
(11)	485	MTH\$SINCOS_R5	
(12)	583	MTH\$SIN_R4	
(14)	688	MTH\$COS_R4	
(15)	768	MTH\$SINCOSD_R5	
(16)	812	MTH\$SIND_R4	
(17)	865	MTH\$COSD_R4	
(19)	901	REDUCE_MEDIUM	
(20)	957	REDUCE_LARGE	
(21)	1287	REDUCE_DEGRFES	
(23)	1376	RADIAN_POLYNOMIALS	: Polynomials for arguments in radians
(24)	1452	CYCLE_POLYNOMIALS	: Polynomials for arguments in cycles
(25)	1523	DEGREE_POLYNOMIALS	
(27)	1589	DEGENERATE_SOLUTIONS	

```
0000 1 .TITLE MTHSSINCOS ; Floating Point Sine, Cosine and Sincos
0000 2 ; Functions
0000 3 .IDENT /2-004/ ; File: MTHSINCOS.MAR EDIT: JCW2004
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 : MTHSSIN and MTHSCOS are functions which return the floating point
0000 34 : sine or cosine value of their single precision floating point argu-
0000 35 : ment (radians). The call is standard call-by-reference.
0000 36 : MTHSSIN_R4 and MTHSCOS_R4 are special routines which are the same
0000 37 : as MTHSSIN and MTHSCOS except a faster non-standard JSB call is
0000 38 : used with the argument in R0 and no registers are saved.
0000 39 :
0000 40 : MTHSSINCOS is a routine which returns the floating point sine and
0000 41 : cosine value of its single precision floating point radian argument.
0000 42 : The call is standard call-by-reference. MTHSSINCOS_R5 is a special
0000 43 : routine which is the same as MTHSSINCOS, except a faster non-
0000 44 : standard JSB call is used with the argument in R0 and no registers
0000 45 : are saved.
0000 46 :
0000 47 : MTHSSIND and MTHSCOSD are functions which return the floating point
0000 48 : sine or cosine value of their single precision floating point argu-
0000 49 : ment (degrees). The call is standard call-by-reference.
0000 50 : MTHSSIND_R4 and MTHSCOSD_R4 are special routines which are the same
0000 51 : as MTHSSIND and MTHSCOSD except a faster non-standard JSB call is
0000 52 : used with the argument in R0 and no registers are saved.
0000 53 :
0000 54 : MTHSSINCOSD is a routine which returns the floating point sine and
0000 55 : cosine value of its single precision floating point degree argument.
0000 56 : The call is standard call-by-reference. MTHSSINCOSD_R5 is a special
0000 57 : routine which is the same as MTHSSINCOSD, except a faster non-
```

```
0000 58 : standard JSB call is used with the argument in R0 and no registers
0000 59 : are saved.
0000 60 :
0000 61 : --
0000 62 :
0000 63 : VERSION:      1
0000 64 :
0000 65 : HISTORY:
0000 66 : AUTHOR:
0000 67 :      MARY PAYNE & JUD LEONARD, 25-MAY-78:   Version 0
0000 68 :
0000 69 : MODIFIED BY:
0000 70 :
0000 71 : 1-1  Tryggve Fossum, 28-May-78
0000 72 :
0000 73 :
0000 74 : VERSION:      2
0000 75 :
0000 76 : HISTORY:
0000 77 : AUTHOR:
0000 78 :      BOB HANEK, 25-MAY-78:   Version 2
0000 79 :
0000 80 :
```

```

0000 82      .SBTTL HISTORY ; Detailed Current Edit History
0000 83
0000 84
0000 85 : ALGORITHMIC DIFFERENCES FROM FP-11C ROUTINE:
0000 86 : 1. Uses POLYF, so more accuracy.
0000 87 : 2. Checks for 1-2**14 before polynomial instead of checking for
0000 88 :      greater than 1.0 afterwards.
0000 89 : 3. Signals complete loss of significance.
0000 90
0000 91 : Edit History for Version 1 of MTH$SINCOS
0000 92
0000 93
0000 94 : 1-2 Add JSB entry points, MTH$COS_R4 and MTH$SIN_R4. Tryggve, JUN-12-78
0000 95 : 1-3 Change routines so as not to use R5 and R6. Tryggve, Jun-15-78
0000 96 : 1-4 Use EMODD if |x| > 8. Tryggve, June-26-78
0000 97 : 1-5 Do CVTDF after EMODD. TNH June-26-78
0000 98 : 1-6 Check argument range after EMODF. Tryggve June-27-78
0000 99 : 1-7 Do CVTFD before EMODD at CBIG. TNH 25-July-78
0000 100 : 1-8 Fix overflow SIN(2**25*PI), too large answers SIN(2**I*PI)
0000 101 :      I=21:24. TNH 26-July-78
0000 102 : 1-9 Same fix for negative arg as in 1-8. TNH 26-July-78
0000 103 : 1-10 Pickup arg with MOVF so reserved op check. TNH 16-Aug-78
0000 104 : 1-011 - Put version number in standard format: three digits in
0000 105 :      edit number field. JBS 16-NOV-78
0000 106 : 1-012 - Change MTH_SINSIGLOS to MTH$K_SINCOSSIG. JBS 07-DEC-78
0000 107 : 1-013 - Add " " to the PSECT directive. JBS 22-DEC-78
0000 108 : 1-014 - Add RSB after call to MTH$SIGNAL so that TAN doesn't
0000 109 :      go into infinite loop. SBL 02-Feb-79
0000 110 : 1-015 - Declare externals. SBL 17-May-1979
0000 111 : 1-016 - Change MTH$K_SINCOSSIG to MTH$K_SIGLOSMAT. JBS 19-SEP-1979
0000 112 : 1-017 - Change argument limit to 2**31 so as to be compatible with
0000 113 :      higher precision routines. SBL 31-Oct-1979
0000 114 : 1-018 - Reduce limit to 2**30 because this routine can't handle 2**31.
0000 115 :      SBL 2-Nov-1979
0000 116 : 1-019 - Add MTH$FLAG JACKET, somehow left out! SBL 2-Nov-1979
0000 117 : 1-019A - Changed BLSS to BLSSU after unbiasing the exponent to exercise
0000 118 :      small argument logic. This logic was never reached prior to this
0000 119 :      change.
0000 120 : - Modified logic for processing reduced arguments close to pi (sin)
0000 121 :      and pi/2 (cos) to perform the operation 1 - RARG in double
0000 122 :      precision. Performing the operation in single precision results
0000 123 :      in losing as much as the last 6 bits. This modification was made
0000 124 :      only for input argument with magnitude less than 2**22.
0000 125 : - Modified the first word of the LOW OF CO entry of the CSTB2 table
0000 126 :      from octa 022174 to octal 32174. This eliminated a negative 1 lsb
0000 127 :      biasing of the values returned by COS for small arguments.
0000 128 : - RNH 12-FEB-1981
0000 129 : 1-20 - Added degree entry points. RNH 1-MAR-1981
0000 130
0000 131
0000 132 : Edit History for Version 2 of MTH$SINCOS
0000 133
0000 134 : Algorithmic differences form Version 1
0000 135 : 1) Introduction of SINCOS entry point
0000 136 : 2) Elimination of the size restriction on the argument
0000 137 : 3) Elinination of the MTH$K_SIGLOSMAT error
0000 138 : 4) Introduction of a possible underflow error for radian arguments

```

```
0000 139 :  
0000 140 : 2-001 - Original  
0000 141 : 2-002 - Modified REDUCE_LARGE to eliminate potential bug (similar to the  
0000 142 : bugs mentioned in QAR 896.) RNH 14-Jan-82  
0000 143 : 2-003 - Modified MTHSSIN so that it returns the same values as MTHSSINcos  
0000 144 : in the interval (0.5,PI/4). The fix was to modify the special code  
0000 145 : used in MTHSSINCOS for !X! in that interval. The modification  
0000 146 : of this code was done because there was no need to JSB to  
0000 147 : NEEDS_DOUBLE_SINCOS to compute COS. This problem was discovered  
0000 148 : by J. Clinkenbeard in the Fortran group. JCW 25-Oct-82.  
0000 149 : 2-004 - To reduce the LSB error to less than 1, the COSTBC1 table was changed  
0000 150 : from a polynomial of degree 3 (Hart 3820) to a polynomial of degree 4  
0000 151 : (Hart 3821). The C0 coefficient is Hart's C0; the earlier version used  
0000 152 : C0-1 to improve accuracy. Now, it is more accurate to use C0. The  
0000 153 : appropriate code modifications were also made. Various comments were  
0000 154 : also corrected and one unnecessary line of code was removed.  
0000 155 : JCW 21-Jun-84  
0000 156 : --
```

```

0000 158          .SBTTL  DECLARATIONS          -          Declarative Part of Module
0000 159
0000 160 :
0000 161 : INCLUDE FILES:          MTH$JACKET.MAR
0000 162 :
0000 163 : EXTERNAL SYMBOLS:
0000 164 :
0000 165          .DSABL  GBL
0000 166          .EXTRN  MTH$AL_4_OV_PI
0000 167          .EXTRN  MTH$$SIGNAL
0000 168          .EXTRN  MTH$K_FLOUNDMAT
0000 169          .EXTRN  MTH$$JACKET_TST
0000 170 :
0000 171 : EQUATED SYMBOLS:
0000 172
0FDB4049 0000 173          LF_PI_OV_4          = ^X0FDB4049          : 0.78539819
31D641E2 0000 174          LF_9_PI_OV_4          = ^X31D641E2          : 7.06858349
F9834022 0000 175          LF_2_OV_PI          = ^XF9834022          : .636619772
CBE44116 0000 176          LF_3_PI_OV_4          = ^XCBE44116          : 2.35619450
53D1417B 0000 177          LF_5_PI_OV_4          = ^X53D1417B          : 3.92699075
EDDF41AF 0000 178          LF_7_PI_OV_4          = ^XEDDF41AF          : 5.49778700
0000 179
00004334 0000 180          W_45                      = ^X4334              : First word of 45 in F-format
00004334 0000 181          LF_45                      = ^X00004334
0000C334 0000 182          LF_M45                    = ^X0000C334
2EE13D65 0000 183          LF_SMALLD                  = ^X2EE13D65          : 180/PI*2^-12
000000B6 0000 184          X_T_OV_45                   = ^XB6
0B603DB6 0000 185          LF_T_OV_45                   = ^X0B603DB6
A3513BEF 0000 186          LF_CONVERT                  = ^XA3513BEF          : pi/180 - 2^-6
2EE142E5 0000 187          LF_90_OV_PI                  = ^X2EE142E5          : 28.64789009
2EE10365 0000 188          LF_SMALLEST_DEG          = ^X2EE10365          : 2^-128*180/PI
0000 189 :
0000 190 : MACROS:
0000 191
0000 192          $$FDEF                    : Define SFS (stack frame) symbols
0000 193          $PSLDEF                   : Define PSL (Processor Status Longword)
0000 194                                     : symbols
0000 195
0000 196 : PSECT DECLARATIONS:
0000 197
00000000 198          .PSECT  _MTH$CODE          PIC,SHR,LONG,EXE,NOWRT
0000 199                                     : program section for math routines
0000 200 :
0000 201 : OWN STORAGE: none
0000 202 :
0000 203 : CONSTANTS:
0000 204
0000 205 D_PI_OV_2:
68C2A221 0FDA40C9 0000 206          .QUAD  ^X68C2A2210FDA40C9          : 1.5707963267948966
0008 207 D_PI:
68C2A221 0FDA4149 0008 208          .QUAD  ^X68C2A2210FDA4149          : 3.1415926535897932
0010 209 D_3_PI_OV_2:
0E92F999 CBE34196 0010 210          .QUAD  ^X0E92F999CBE34196          : 4.7123889803846899
0018 211 D_2_PI:
68C2A221 0FDA41C9 0018 212          .QUAD  ^X68C2A2210FDA41C9          : 6.2831853071795865
0020 213

```



```

0020 215
0020 216          .SBTTL COEFFICIENT TABLES          -          Series Coefficients
0020 217
0020 218
0020 219
0020 220
0020 221 :
0020 222 : Polynomial Coefficient tables for arguments in radians
0020 223 :
0020 224
0020 225 COSTBR1:          : COS coefficients for arguments less than 1/2
6CE6BBB4 0020 226          .LONG          ^X6CE6BBB4          : C3 = -.13765364E-02
ABA53E2A 0024 227          .LONG          ^XABA53E2A          : C2 = .41664738E-01
FFFDBFFF 0028 228          .LONG          ^XFFFDBFFF          : C1 = -.49999991E+00
00004080 002C 229          .LONG          ^X00004080         : C0 = .10000000E+01
00000004 0030 230 COSLENR1 = .-COSTBR1/4
0030 231
0030 232 COSTBR2:          : COS coefficients for arguments greater than 1/2
836538CC 0030 233          .LONG          ^X836538CC          : C4 = .24379880E-04
03C3BBB6 0034 234          .LONG          ^X03C3BBB6          : C3 = -.13886619E-02
AA9D3E2A 0038 235          .LONG          ^XAA9D3E2A          : C2 = .41666616E-01
3C263284 003C 236          .LONG          ^X3C263284          : C1 = .38485437E-08
00000000 0040 237          .LONG          ^X00000000         : C0 = .00000000E+00
00000005 0044 238 COSLENR2 = .-COSTBR2/4
0044 239
0044 240 SINTBR:          : SIN coefficients
8AE4BA4C 0044 241          .LONG          ^X8AE4BA4C          : C3 = -.195066968E-03
83023D08 0048 242          .LONG          ^X83023D08          : C2 = .833201595E-02
AAA0BF2A 004C 243          .LONG          ^XAAA0BF2A          : C1 = -.166666508E+00
8ADEB25E 0050 244          .LONG          ^X8ADEB25E          : C0 = -.323841887E-08
00000004 0054 245 SINLENR = .-SINTBR/4
0054 246
0054 247
0054 248
0054 249
0054 250
0054 251 :
0054 252 : Polynomial coefficients for arguments in cycles
0054 253 :
0054 254
0054 255 COSTBC1:          : COS coefficients for arguments less than 2/pi
0054 256
CFAA376C 0054 257          .LONG          ^XCFAA376C          : C4 = 0.35287617E-05
E275BAAA 0058 258          .LONG          ^XE275BAAA          : C3 = -.32593650E-03
E0ED3D81 005C 259          .LONG          ^XE0ED3D81          : C2 = 0.15854323E-01
E9E6BF9D 0060 260          .LONG          ^XE9E6BF9D          : C1 = -.30842513E+00
00004080 0064 261          .LONG          ^X00004080         : C0 = 0.10000000E+01
00000005 0068 262 COSLENC1 = .-COSTBC1/4
0068 263
0068 264 COSTBC2:          : COS coefficients for arguments greater than 2/pi
CFAA376C 0068 265          .LONG          ^XCFAA376C          : C4 = 0.35287617E-05
E275BAAA 006C 266          .LONG          ^XE275BAAA          : C3 = -.32593650E-03
E0ED3D81 0070 267          .LONG          ^XE0ED3D81          : C2 = 0.15854323E-01
4F32BE6F 0074 268          .LONG          ^X4F32BE6F          : C1 = -.58425136E-01
00000000 0078 269          .LONG          ^X00000000         : C0 = 0.00000000E+00
00000005 007C 270 COSLENC2 = .-COSTBC2/4
007C 271

```

```

007C 272 SINTBC: ; SIN coef for arg in cycles
C97DB916 007C 273 .LONG ^XC97DB916 ; C3 = -.35950437E-04
2F493C23 0080 274 .LONG ^X2F493C23 ; C2 = 0.24900010E-02
5DDCBEA5 0084 275 .LONG ^X5DDCBEA5 ; C1 = -.80745429E-01
FDA93E10 0088 276 .LONG ^XFDA93E10 ; C0 = 0.35398159E-01
00000004 008C 277 SINLENC = .-SINTBC/4
008C 278
008C 279
008C 280
008C 281
008C 282
008C 283 ;
008C 284 ; Polynomial coefficients for arguments in degrees
008C 285 ;
008C 286
008C 287 COSDTB1: ; COS coefficients for arguments less than 90/pi
E231AA2D 008C 288 .LONG ^XE231AA2D ; C3 = -0.386099064E-13
D3133284 0090 289 .LONG ^XD3133284 ; C2 = 0.386570198E-08
B502BA1F 0094 290 .LONG ^XB502BA1F ; C1 = -0.152308523E-03
00004080 0098 291 .LONG ^X00004080 ; C0 = 0.100000000E+01
00000003 009C 292 COSDLN1 = .-COSDTB1/4 - 1
009C 293
009C 294 COSDTB2: ; COS coefficients for arguments greater than 90/pi
C1342177 009C 295 .LONG ^XC1342177 ; C4 = 0.209856393E-18
C612AA30 00A0 296 .LONG ^XC612AA30 ; C3 = -0.392516491E-13
D8803284 00A4 297 .LONG ^XD8803284 ; C2 = 0.386631882E-08
A876B8FD 00A8 298 .LONG ^XA876B8FD ; C1 = -0.302383960E-04
9935AF76 00AC 299 .LONG ^X9935AF76 ; C0 = -0.560699993E-10
00000004 00B0 300 COSDLN2 = .-COSDTB2/4 - 1
00B0 301
00B0 302 SINDTB: ; SIN coefficients
D7E3A5DD 00B0 303 .LONG ^XD7E3A5DD ; C3 = -0.962091984E-16
630C2E6D 00B4 304 .LONG ^X630C2E6D ; C2 = 0.134938831E-10
DC01B66D 00B8 305 .LONG ^XDC01B66D ; C1 = -0.886095279E-06
A3513BEF 00BC 306 .LONG ^XA3513BEF ; C0 = 0.182829250E-02
00000003 00C0 307 SINDLN = .-SINDTB/4 - 1
00C0 308

```

00C0 310 .SBTTL MTHSSINCOS - Radian arguments

00C0 311
00C0 312
00C0 313
00C0 314 : FUNCTIONAL DESCRIPTION:

00C0 315 :
00C0 316 : The SIN, COS and SINCOS routines are based on octant reduction. Given an
00C0 317 : argument, x, it is written in the form

00C0 318 :
00C0 319 : $x = I1*(2*pi) + I*(pi/4) + Y1,$
00C0 320 :

00C0 321 : where I1 and I are integers, $0 \leq I < 8$ and $0 \leq Y1 < pi/4$. Since SIN and
00C0 322 : COS have a period of $2*pi$ it follows that

00C0 323 :
00C0 324 : $SIN(x) = SIN(I*(pi/4) + Y1)$ and
00C0 325 : $COS(x) = COS(I*(pi/4) + Y1).$
00C0 326 :

00C0 327 : Using the trigonometric identities for the sum and difference of two angles,
00C0 328 : the following table can be generated:

If I =	then SIN(x) =	and COS(x) =
-----	-----	-----
0	SIN(Y1)	COS(Y1)
1	COS(pi/4-Y1)	SIN(pi/4-Y1)
2	COS(Y1)	-SIN(Y1)
3	SIN(pi/4-Y1)	-COS(pi/4-Y1)
4	-SIN(Y1)	-COS(Y1)
5	-COS(pi/4-Y1)	-SIN(pi/4-Y1)
6	-COS(Y1)	SIN(Y1)
7	-SIN(pi/4-Y1)	COS(pi/4-Y1)

00C0 340 :
00C0 341 : Let Y be defined as $Y = Y1$ if I is even and $Y = pi/4 - Y1$, if I is odd, then
00C0 342 : each entry of the above table is of the form $\pm SIN(Y)$ or $\pm COS(Y)$. Based
00C0 343 : on the above remarks, the SIN, COS and SINCOS routines process the input
00C0 344 : argument x, to obtain I and Y, and based on I selects a suitable polynomial
00C0 345 : approximation, p(Y), to evaluate the desired function.
00C0 346 :
00C0 347 :

00C0 348 : INPUT PARAMETERS:
00C0 349 :

00000004	00C0	350	LONG = 4	
00000004	00C0	351	x = 1*LONG	: x is input angle in radians
00000008	00C0	352	sine = 2*LONG	: sine is SIN(x)
0000000C	00C0	353	cosine = 3*LONG	: cosine is COS(x)
	00C0	354		

```

00C0 356
00C0 357
00C0 358
00C0 359 : Return sine and cosine of argument
00C0 360 :
00C0 361 :
003C 00C0 362
00C0 363 .ENTRY MTH$SINCOS, ^M<R2, R3, R4, R5>
00C2 364
00C2 365 MTH$FLAG_JACKET
6D 00000000'GF 9E 00C2 MOVAB G^MTH$$JACKET_HND, (FP)
00C9 ; set handler address to jacket
00C9 ; handler
00C9
00C9 366
50 04 BC 50 00C9 367 MOVF @x(AP), R0
00000148'EF 16 00CD 368 JSB MTH$SINCOS_R5
08 BC 50 D0 00D3 369 MOVL R0, @sine(AP)
0C BC 51 D0 00D7 370 MOVL R1, @cosine(AP)
04 00DB 371 RET
00DC 372
00DC 373
00DC 374
00DC 375 .SBTTL MTH$SIN
00DC 376
00DC 377 :
00DC 378 : Return sine of argument
00DC 379 :
00DC 380 :
00DC 381 :
001C 00DC 382 .ENTRY MTH$SIN, ^M<R2, R3, R4>
00DE 383
00DE 384 MTH$FLAG_JACKET
6D 00000000'GF 9E 00DE MOVAB G^MTH$$JACKET_HND, (FP)
00E5 ; set handler address to jacket
00E5 ; handler
00E5
00E5 385
50 04 BC 50 00E5 386 MOVF @x(AP), R0
0000020F'EF 16 00E9 387 JSB MTH$SIN_R4
04 00EF 388 RET
00F0 389
00F0 390
00F0 391
00F0 392 .SBTTL MTH$COS
00F0 393
00F0 394 :
00F0 395 : Return cosine of argument
00F0 396 :
00F0 397 :
001C 00F0 398 .ENTRY MTH$COS, ^M<R2, R3, R4>
00F2 400
00F2 401 MTH$FLAG_JACKET
00F2

```

```
6D 00000000'GF 9E 00F2          MOVAB  G^MTH$$JACKET_HND, (FP)
                    00F9
                    00F9
                    00F9
                    00F9 402
                    50 04 BC 50 00F9 403
000002B6'EF    16 00FD 404
                    04 0103 405
                    0104 406

                    MOVF  @x(AP), R0
                    JSB   MTH$COS_R4
                    RET
```

; set handler address to jacket
; handler

0104 408 .SBTTL MTH\$SINCOSD - Degrees

0104 409

0104 410

0104 411 :

0104 412 : FUNCTIONAL DESCRIPTION:

0104 413 :

0104 414 : The SIND, COSD and SINCOSD routines are based on octant reduction. Given an
0104 415 : argument, x, it is written in the form

0104 416 :

0104 417 :

$$x = I1*360 + I*45 + Y1,$$

0104 418 :

0104 419 : where I1 and I are integers, $0 \leq I < 8$ and $0 \leq Y1 < 45$. Since SIND and
0104 420 : COSD have a period of 360 it follows that

0104 421 :

0104 422 :

$$\begin{aligned} \text{SIND}(x) &= \text{SIND}(I*45 + Y1) \text{ and} \\ \text{COSD}(x) &= \text{COSD}(I*45 + Y1). \end{aligned}$$

0104 423 :

0104 424 :

0104 425 : Using the trigonometric identities for the sum and difference of two angles,
0104 426 : the following table can be generated:

0104 427 :

0104 428 :

0104 429 :

0104 430 :

0104 431 :

0104 432 :

0104 433 :

0104 434 :

0104 435 :

0104 436 :

0104 437 :

0104 438 :

0104 439 :

0104 440 :

0104 441 :

0104 442 :

0104 443 :

0104 444 :

0104 445 :

00000004 0104 446

00000008 0104 447

0000000C 0104 448

0104 449

LONG = 4
sind = 2*LONG
cosd = 3*LONG

If I =	then SIND(x) =	and COSD(x) =
-----	-----	-----
0	SIND(Y1)	COSD(Y1)
1	COSD(45-Y1)	SIND(45-Y1)
2	COSD(Y1)	-SIND(Y1)
3	SIND(45-Y1)	-COSD(45-Y1)
4	-SIND(Y1)	-COSD(Y1)
5	-COSD(45-Y1)	-SIND(45-Y1)
6	-COSD(Y1)	SIND(Y1)
7	-SIND(45-Y1)	COS(45-Y1)

Let Y be defined as $Y = Y1$ if I is even and $Y = 45 - Y1$, if I is odd, then each entry of the above table is of the form $\pm \text{SIN}(Y)$ or $\pm \text{COS}(Y)$. Based on the above remarks, the SIND, COSD and SINCOSD routines process the input argument x, to obtain I and Y, and based on I selects a suitable polynomial approximation, p(Y), to evaluate the desired function.

		003C	0104	451	.ENTRY MTHSSINCOSD	^M<R2, R3, R4, R5>
			0106	452		
			0106	453	MTH\$FLAG_JACKET	
6D	00000000'GF	9E	0106		MOVAB	G^MTH\$\$JACKET_HND, (FP)
			010D			; set handler address to jacket
			010D			; handler
			010D	45		
	50 04 BC	50	010D	455	MOVF	@X(AP), R0
	0000033D'EF	16	0111	456	JSB	MTHSSINCOSD_R5
	08 BC 50	D0	0117	457	MOVL	R0, @sind(AP)
	0C BC 51	D0	011B	458	MOVL	R1, @cosd(AP)
			011F	459		
		04	011F	460	RET	
			0120	461		
			0120	462		
			0120	463		
		001C	0120	464	.ENTRY MTHSSIND	^M<R2, R3, R4>
			0122	465		
			0122	466	MTH\$FLAG_JACKET	
6D	00000000'GF	9E	0122		MOVAB	G^MTH\$\$JACKET_HND, (FP)
			0129			; set handler address to jacket
			0129			; handler
			0129	467		
	50 04 BC	50	0129	468	MOVF	@X(AP), R0
	0000038D'EF	16	012D	469	JSB	MTHSSIND_R4
			0133	470		
		04	0133	471	RET	
			0134	472		
			0134	473		
			0134	474		
		001C	0134	475	.ENTRY MTH\$COSD	^M<R2, R3, R4>
			0136	476		
			0136	477	MTH\$FLAG_JACKET	
6D	00000000'GF	9E	0136		MOVAB	G^MTH\$\$JACKET_HND, (FP)
			013D			; set handler address to jacket
			013D			; handler
			013D	478		
	50 04 BC	50	013D	479	MOVF	@X(AP), R0
	000003F1'EF	16	0141	480	JSB	MTH\$COSD_R4
			0147	481		
		04	0147	482	RET	
			0148	483		

```

0148 485      .SBTTL MTH$SINCOS_R5
0148 486
0148 487      ; This routine computes the SIN and COS of the F-format value of R0. The
0148 488      ; computation is performed one of three ways depending on the size of the
0148 489      ; input argument, X:
0148 490
0148 491      ; 1) If |X| < pi/4, then X is used directly in polynomial approximation
0148 492      ; of SIN and COS.
0148 493      ; 2) If pi/4 <= |X| < 9*pi/4, then the subroutine REDUCE_MEDIUM is called
0148 494      ; to reduce the argument to an equivalent argument in radians, Y, and
0148 495      ; the octant, I, containing the argument. Y is then evaluated in two
0148 496      ; polynomials chosen as a function of I, to compute SIN(X) and COS(X).
0148 497      ; 3) If 9*pi/4 <= |X|, then the subroutine REDUCE_LARGE is called to
0148 498      ; reduce the argument to an equivalent argument in cycles, Y, and the
0148 499      ; octant, I, containing the argument. Y is then evaluated in two
0148 500      ; polynomials chosen as a function of I, to compute SIN(X) and COS(X).
0148 501
0148 502 MTH$SINCOS_R5::
0148 503     MOVF    R0, R4          ; R4 = X
0148 504     BGEQ   POS_SINCOS
0148 505     JSB    SINCOS          ; R0 = SIN(|X|), R1 = COS(X)
0148 506     MNEGF  R0, R0          ; R0 = SIN(X)
0148 507     RSB
0148 508
0148 509 SINCOS:
0148 510     BICW   #^X8000, R0    ; R0 = |X|
0148 511     POS_SINCOS:
0148 512     CMPF   #LF_PI_OV_4, R0 ; Compare pi/4 with |X|
0148 513     BGTR   SMALL_SINCOS   ; No argument reduction is necessary
0148 514     CMPF   #LF_9_PI_OV_4, R0 ; Compare 9*pi/4 with |X|
0148 515     BGEQ   1$
0148 516     BRW   LARGE_SINCOS   ; Use special logic for |X| > 9*pi/4
0148 517
0148 518     ; pi/4 <= |X| < 9*pi/4
0148 519     ;
0148 520     ;
0148 521     1$: JSB    REDUCE_MEDIUM ; Medium argument reduction routine
0148 522     ; R3/R4 = Y = reduced argument
0148 523     ; R2 = octant
0148 524     MOVQ   R3, -(SP)      ; Save reduced argument on stack
0148 525     PUSHL  R2            ; Save octant bits on stack
0148 526     JSB    M_COS         ; R0 = COS(X)
0148 527     MOVL  (SP)+, R2      ; R2 = Octant bits
0148 528     MOVQ  (SP)+, R3      ; R3/R4 = reduced argument
0148 529     PUSHL  R0            ; Save COS(X) on stack
0148 530     JSB    M_SIN         ; R0 = SIN(X)
0148 531     MOVL  (SP)+, R1      ; R1 = COS(X)
0148 532     RSB
0148 533     ;
0148 534     ; Logic for small arguments. |X| < pi/4.
0148 535     ;
0148 536     ;
0148 537     SMALL_SINCOS:
0148 538     CMPW   #^X4000, R0   ; Compare 1/2 with |X|
0148 539     BLSS   2$
0148 540     CMPW   #^X3A80, R0  ; Sufficient overhang not available
0148 541     BGEQ   1$           ; Compare with 2^12-12
0148 541     ; No polynomial evaluation is needed

```



```

54 50 D0 01A2 542      MOVL  R0, R4          ; R4 = !X!
FE71 55 54 45 01A5 543      MULF3 R4, R4, R5       ; R5 = X*X
CF 03 55 55 01A9 544      POLYF R5, #COSLENR1-1, COSTBR1 ; R0 = COS(X)
FE8D 50 50 DD 01AF 545      PUSHL R0              ; Save COS(X) on stack
CF 03 55 55 01B1 546      POLYF R5, #SINLENR-1, SINTBR   ; R0 = q(X^2)
50 54 44 01B7 547      MULF  R4, R0          ; R0 = X*q(X^2)
50 54 40 01BA 548      ADDF  R4, R0          ; R0 = SIN(X)
51 8E D0 01BD 549      MOVL  (SP)+, R1       ; R1 = COS(X)
05 05 01C0 550      RSB
51 08 50 01C1 552 1$:      MOVF  #1.0, R1        ; R0 = X, R1 = 1.0 = COS(X)
05 05 01C4 553      RSB
01C5 554
01C5 555
53 50 DD 01C5 556 2$:      PUSHL R0              ; Save !x! on stack
53 50 56 01C7 557      CVTFD R0, R3         ; R3/R4 = !X!
53 53 64 01CA 558      MULD  R3, R3         ; R3/R4 = X^2
0000072D'EF 53 DD 01CD 559      PUSHL R3              ; Save x^2 on stack
55 50 16 01CF 560      JSB   NEEDS_DOUBLE_SINCOS ; Use special logic to obtain overhang
FE63 50 8E D0 01D5 561      MOVL  R0, R5         ; Save COS(X) in R5
CF 03 50 55 01DB 562      MOVL  (SP)+, R0      ; R0 = X^2
50 6E 44 01E1 563      POLYF R0, #SINLENR-1, SINTBR ; R0 = q(X^2)
50 8E 40 01E4 564      MULF  (SP)+, R0     ; R0 = X*q(X^2)
51 55 D0 01E7 565      ADDF  (SP)+, R0     ; R0 = SIN(X)
05 05 01EA 566      MOVL  R5, R1        ; R1 = COS(X)
01EB 567      RSB
01EB 568
01EB 569
00000490'EF 16 01EB 570 LARGE_SINCOS:
01F1 571      JSB   REDUCE_LARGE   ; R3/R4 = reduced argument (in cycles)
01F1 572      PUSHL R2            ; R2 = octant bits
7E 52 DD 01F1 573      PUSHL R2            ; Save octant bits on stack
00000311'EF 7D 01F3 574      MOVQ  R3, -(SP)     ; Save reduced argument on stack
53 8E 16 01F6 575      JSB   L COS         ; R0 = COS(X)
52 8E 7D 01FC 576      MOVQ  (SP)+, R3     ; Reduced argument in R3/R4
6E 6E D0 01FF 577      MOVL  (SP), R2      ; R2 = octant bits
0000028A'EF 6E 50 D0 0202 578      MOVL  R0, (SP)     ; R1 = COS(X)
51 8E 16 0205 579      JSB   L SIN         ; R0 = SIN(X)
05 05 020B 580      MOVL  (SP)+, R1    ; R1 = COS(X)
05 05 020E 581      RSB

```

```

020F 583      .SBTTL MTH$SIN_R4
020F 584
020F 585      ; This routine computes the SIN of the F-format value of R0. The computation
020F 586      ; is performed one of three ways depending on the size of the input argument,
020F 587      ; X:
020F 588
020F 589      1) If |X| < pi/4, then X is used directly in a polynomial approximation
020F 590      of SIN.
020F 591      2) If pi/4 =< |X| < 9*pi/4, then the subroutine REDUCE_MEDIUM is called
020F 592      to reduce the argument to an equivalent argument in radians, Y, and
020F 593      the octant, I, containing the argument. Y is then evaluated in a
020F 594      polynomial chosen as a function of I to compute SIN(X).
020F 595      3) If 9*pi/4 =< |X|, then the subroutine REDUCE_LARGE is called to
020F 596      reduce the argument to an equivalent argument in cycles, Y, and the
020F 597      octant, I, containing the argument. Y is then evaluated in a
020F 598      polynomial chosen as a function of I to compute SIN(X).
020F 599
020F 600 MTH$SIN_R4::
020F 601      TSTF      R0          ; Check the sign of R0
020F 602      BGEQ     POS_SIN
020F 603      JSB      SIN        ; R0 = SIN(|X|)
020F 604      MNEGF   R0, R0     ; R0 = SIN(X)
020F 605      RSB
020F 606
020F 607 SIN:
020F 608      BICW     #*X8000, R0 ; R0 = |X|
020F 609 POS_SIN:
020F 610      CMPF     #LF_PI_OV_4, R0 ; Compare pi/4 with |X|
020F 611      BGTR     SMALL_SIN    ; No argument reduction is necessary
020F 612      CMPF     #LF_9_PI_OV_4, R0 ; Compare 9*pi/4 with |X|
020F 613      BLSS     LARGE_SIN  ; Use special logic for |X| > 9*pi/4
020F 614
020F 615      ;
020F 616      ; pi/4 =< |X| < 9*pi/4
020F 617      ;
020F 618      JSB      REDUCE_MEDIUM ; Medium argument reduction routine
020F 619      ; R3/R4 = Y = reduced argument
020F 620      ; R2 = octant
020F 621 M_SIN: CASEB R2, #0, #7 ; Branch to one of four polynomial
020F 622      ; evaluations depending on the
020F 623      ; octant bits.
020F 624      .WORD   P_SIN_R-1$
020F 625      .WORD   P_COS_R-1$
020F 626      .WORD   P_COS_R-1$
020F 627      .WORD   N_SIN_R-1$
020F 628      .WORD   N_SIN_R-1$
020F 629      .WORD   N_COS_R-1$
020F 630      .WORD   N_COS_R-1$
020F 631      .WORD   P_SIN_R-1$
020F 632      ;
020F 633      ; Logic for small arguments. |X| < pi/4.
020F 634      ;
020F 635
020F 636 SMALL_SIN:
020F 637      CMPW     #*X4000, R0 ; Compare 1/2 with |X|
020F 638      BLSS     2$          ; Sufficient overhang not available
020F 639      CMPW     #*X3A80, R0 ; Compare with 2^-12

```

```

FDDC CF      54 12 18 025A 640      BGEQ      1$      : No polynomial evaluation is needed
              50 50 00 025C 641      MOVL      R0, R4      : R4 = X
              50 50 44 025F 642      MULF      R0, R0      : R0 = X*X
              03 50 55 0262 643      POLYF     R0, #SINLENR-1, SINTBR : R0 = q(x^2)
              50 54 44 0268 644      MULF      R4, R0      :
              50 54 40 026B 645      ADDF      R4, R0      : R0 = SIN(X)
              05 026E 646 1$:      RSB
              026F 647
              50 50 DD 026F 648 2$:      PUSHL     R0      : Save |x| on stack
              50 50 56 0271 649      CVTFD    R0, R0      : R0/R1 = |X|
              50 50 64 0274 650      MULD2    R0, R0      : R0/R1 = X^2
FDC7 CF      03 50 55 0277 651      POLYF     R0, #SINLENR-1, SINTBR : R0 = q(X^2)
              50 6E 44 027D 652      MULF2    (SP), R0      : R0 = X*q(X^2)
              50 8E 40 0280 653      ADDF2    (SP)+, R0     : R0 = SIN(X)
              05 0283 654      RSB
              0284 655
              000J0490'EF 16 0284 656 LARGE_SIN:
              028A 658      JSB      REDUCE_LARGE      : R3/R4 = reduced argument (in cycles)
              53 05 028A 659 L_SIN:      TSTL      R3      : R2 = octant bits
              14 13 028C 660      BEQL     DEGENERATE_CASE_SIN : Check for degenerate case
              07 00 52 8F 028E 661
              028E 662      CASEB    R2, #0, #7
              0292 663
              058A' 0292 664 1$:      .WORD    P_SIN_C-1$
              051A' 0294 665      .WORD    P_COS_C-1$
              051A' 0296 666      .WORD    P_COS_C-1$
              058A' 0298 667      .WORD    P_SIN_C-1$
              0585' 029A 668      .WORD    N_SIN_C-1$
              054D' 029C 669      .WORD    N_COS_C-1$
              054D' 029E 670      .WORD    N_COS_C-1$
              0585' 02A0 671      .WORD    N_SIN_C-1$
              02A2 672
              02A2 673
              02A2 674 DEGENERATE_CASE_SIN:
              02A2 675
              52 52 52 01 8A 02A2 676      BICB     #1, R2      : Compute index as (R2 - 1)/2
              03 03 00 FF 8F 9C 02A5 677      ROTL     #-1, R2, R2
              02AA 678      CASEB    R2, #0, #3
              02AE 679
              0647' 02AE 680 1$:      .WORD    P ONE-1$
              0653' 02B0 681      .WORD    UNFL -1$
              064B' 02B2 682      .WORD    N ONE-1$
              0653' 02B4 683      .WORD    UNFL -1$
              02B6 684

```

```

02B6 686
02B6 687
02B6 688          .SBTTL MTH$COS_R4
02B6 689
02B6 690 ; This routine computes the COS of the F-format value of R0. The computation
02B6 691 ; is performed one of three ways depending on the size of the input argument,
02B6 692 ; X. The processing is the same as described for MTH$SIN_R4.
02B6 693 ;
02B6 694
02B6 695 MTH$COS_R4::
50 50 8000 8F 53 02B6 696      TSTF      R0          ; Check for reserved operand
50 50 UFDB4049 8F 51 02B8 697      BICW      #^X8000, R0      ; R0 = |X|
23 14 02C4 699      BGTR      SMALL_COS ; Compare pi/4 with |X|
50 31D641E2 8F 51 02C6 700      CMPF      #LF 9*PI OV_4, R0 ; No argument reduction is necessary
3C 19 02CD 701      BLSS      LARGE_COS  ; Compare 9*pi/4 with |X|
02CF 702          ; Use special logic for |X| > 9*pi/4
02CF 703          ;
02CF 704          ; pi/4 <= |X| < 9*pi/4
02CF 705          ;
0000042C'EF 16 02CF 706      JSB      REDUCE_MEDIUM ; Medium argument reduction routine
02D5 707          ; R3/R4 = Y = reduced argument
07 00 52 8F 02D5 708          ; R2 = octant
02D9 709 M_COS: CASEB R2, #0, #7 ; Branch to one of four polynomial
0445' 02D9 711 1$: .WORD P_COS_R-1$ ; evaluations depending on the
0486' 02DB 712      .WORD N_SIN_R-1$ ; octant bits.
0486' 02DD 713      .WORD N_SIN_R-1$
047B' 02DF 714      .WORD N_COS_R-1$
047B' 02E1 715      .WORD N_COS_R-1$
048B' 02E3 716      .WORD P_SIN_R-1$
048B' 02E5 717      .WORD P_SIN_R-1$
0445' 02E7 718      .WORD P_COS_R-1$
02E9 719
02E9 720          ;
02E9 721          ; Logic for small arguments. |X| < pi/4.
02E9 722          ;
02E9 723          ;
02E9 724 SMALL_COS:
50 4000 8F B1 02E9 725      CMPW      #^X4000, R0 ; Compare 1/2 with |X|
06 18 02EE 726      BGEQ      1$ ; Sufficient overhang is available
53 50 56 02F0 727      CVTFD     R0, R3 ; R3/R4 = |X|
0434 31 02F3 728      BRW      NEEDS_DOUBLE ; Use special logic to obtain overhang
50 3A80 8F B1 02F6 729 1$: CMPW      #^X3A80, R0 ; Compare with 2^12-1
0A 18 02FB 730      BGEQ      2$ ; No polynomial evaluation is needed
50 50 44 02FD 731      MULF     RO,RO ; R0 = X*X
FD1A CF 03 50 55 0300 732      POLYF     RO, #COSLENR1-1, COSTBR1 ; R0 = COS(X)
05 0306 733      RSB
50 08 50 0307 734      ;
05 030A 735 2$: MOVF      #1.0, R0 ; R0 = 1.0 = COS(X)
030B 736      RSB
030B 737
030B 738
00000490'EF 16 030B 739 LARGE_COS:
0311 740      JSB      REDUCE_LARGE ; R3/R4 = reduced argument (in cycles)
53 D5 0311 741      ; R2 = octant bits
0311 742 L_COS: TSTL      R0 ; Check for degenerate case

```

```

14 13 0313 743 BEQL DEGENERATE_CASE_COS
07 00 52 8F 0315 744
0493' 0315 745 1$: CASEB R2, #0, #7
0503' 0319 746 .WORD P_COS_C-1$
04FE' 031B 747 .WORD P_SIN_C-1$
04C6' 031D 748 .WORD N_SIN_C-1$
04C6' 031F 749 .WORD N_COS_C-1$
04FE' 0321 750 .WORD N_COS_C-1$
0503' 0323 751 .WORD N_SIN_C-1$
0493' 0325 752 .WORD P_SIN_C-1$
0327 753 .WORD P_COS_C-1$
0329 754
0329 755
0329 756 DEGENERATE_CASE_COS:
0329 757
52 52 52 FF 01 8A 0329 758 BICB #1, R2 ; Compute index as (R2 - 1)/2
03 00 52 8F 9C 032C 759 ROTL #-1, R2, R2
0331 760 CASEB R2, #0, #3
0335 761
05CC' 0335 762 1$: .WORD UNFL -1$
05C4' 0337 763 .WORD N_ONE-1$
05CC' 0339 764 .WORD UNFL -1$
05C0' 033B 765 .WORD P_ONE-1$
033D 766

```

```

033D 768      .SBTTL  MTH$SINCOSD_R5
033D 769
033D 770      ; This routine computes the SIND and COSD of the F-format value of R0.  The
033D 771      ; computation is performed one of two ways depending on the size of the input
033D 772      ; argument, X:
033D 773      ;
033D 774      ; 1) If |X| < 45, then X is used directly in polynomial approximation
033D 775      ; of SIND and COSD.
033D 776      ; 2) If 45 <= |X|, then the subroutine REDUCE_DEGREES is called to reduce
033D 777      ; the argument to an equivalent argument in degrees, Y, and the
033D 778      ; octant, I, containing the argument.  Y is then evaluated in two
033D 779      ; polynomials chosen as a function of I, to compute SIND(X) and
033D 780      ; COSD(X).
033D 781
033D 782      MTH$SINCOSD_R5::
033D 783          TSTF      R0
033D 784          BGEQ      SINCOSD
033D 785          BICW      #^X8000, R0
033D 786          JSB      SINCOSD
033D 787          MNEGF    R0, R0
033D 788          RSB
033D 789
033D 790      SINCOSD:
033D 791          CMPW      #W 45, R0
033D 792          BGTR      SMALL_SINCOSD
033D 793          JSB      REDUCE_DEGREES
033D 794          MOVQ     R3, -(SP)
033D 795          JSB      EVAL_COSD
033D 796          MOVQ     (SP)+, R3
033D 797          PUSHL   R0
033D 798          JSB      EVAL_SIND
033D 799          MOVL    (SP)+, R1
033D 800          RSB
033D 801
033D 802
033D 803      SMALL_SINCOSD:
033D 804          PUSHL   R0
033D 805          JSB      SMALL_COSD
033D 806          MOVL    R0, R5
033D 807          MOVL    (SP)+, R0
033D 808          JSB      SMALL_SIND
033D 809          MOVL    R5, R1
033D 810          RSB

50      8000 8F      AA 0341 785          ; R0 = |X|
00000350'EF 16 0346 786          ; R0/R1 = SIND(|X|)/COSD(|X|)
50      50      52 034C 787          ; R0 = -SIND(|X|)
50      50      05 034F 788

50      4334 8F      B1 0350 791          ; Compare 45 to |X|
0000069C'EF 14 0355 792          ; special processing for small arg
7E      53      16 0357 793          ; R3/R4 = octant/reduced argument
00000405'EF 7D 035D 794          ; Save octant bits and reduced arg
53      8E      16 0360 795          ; R0 = COSD(Y)
000003AD'EF DD 0366 796          ; R3/R4 = octant/reduced argument
51      8E      7D 0369 797          ; Save COSD(Y)
51      8E      16 036B 798          ; R0 = SIND(Y)
51      8E      DD 0371 799          ; R1 = COSD(Y)
51      8E      05 0374 800

00000419'EF DD 0375 801
55      50      0375 802
50      8E      0375 803
000003C1'EF DD 0375 804          ; Save argument
51      55      16 0377 805          ; R0 = COSD(|X|)
50      8E      DD 037D 806          ; R5 = COSD(|X|)
51      55      DD 0380 807          ; R0 = argument
51      55      16 0383 808          ; R0 = SIND(X)
51      55      DD 0389 809          ; R1 = COSD(|X|)
51      55      05 038C 810

```

```

038D 812      .SBTTL MTH$SIND_R4
038D 813
038D 814      ; This routine computes the SIND of the F-format value of R0. The computation
038D 815      ; is performed one of two ways depending on the size of the input argument, X:
038D 816      ;
038D 817      ; 1) If |X| < 45, then X is used directly in polynomial approximation
038D 818      ; of SIND.
038D 819      ; 2) If 45 =< |X|, then the subroutine REDUCE_DEGREES is called to reduce
038D 820      ; the argument to an equivalent argument in degrees, Y, and the
038D 821      ; octant, I, containing the argument. Y is then evaluated in two
038D 822      ; polynomials chosen as a function of I, to compute SIND(X).
038D 823
038D 824 MTH$SIND_R4::
038D 825      TSTF      R0          ; R0 = X
038D 826      BGEQ      POS_SIND
038D 827      JSB       NEG_SIND   ; R0 = SIND(|X|)
038D 828      MNEGF     R0, R0    ; R0 = -SIND(|X|)
038D 829      RSB
038D 830
038D 831 NEG_SIND:
038D 832      BICW      #^X8000, R0 ; R0 = |X|
038D 833 POS_SIND:
038D 834      CMPW      #W 45, R0   ; Compare 45 to |X|
038D 835      BGTR      SMALL_SIND  ; special processing for small arg
038D 836      JSB       REDUCE_DEGREES ; R3/R4 = octant/reduced argument
038D 837
038D 838 EVAL_SIND:
038D 839      CASEB     R3, #0, #7
038D 840 1$:      .WORD   P_SIN_D-1$
038D 841      .WORD   P_COS_D-1$
038D 842      .WORD   P_COS_D-1$
038D 843      .WORD   P_SIN_D-1$
038D 844      .WORD   N_SIN_D-1$
038D 845      .WORD   N_COS_D-1$
038D 846      .WORD   N_COS_D-1$
038D 847      .WORD   N_SIN_D-1$
038D 848
038D 849
038D 850 SMALL_SIND:
038D 851      CMPF      #LF_SMALLD, R0 ; Compare 180/pi*2^-12 with |x|
038D 852      BGTR      1$          ; No polynomial evaluation is
038D 853      MOVL      R0, R4      ; necessary
038D 854      BRW      P_SIN_D
038D 855 1$:      TSTF      R0
038D 856      BEQL      3$          ; Check for zero
038D 857      CMPF      #LF_SMALLEST_DEG, R0 ; Return if R0 = 0
038D 858      BLEQ      2$          ; Check for possible underflow on
038D 859      BRW      UNFL        ; conversion to radians
038D 860 2$:      MULF3     #LF_CONVERT, R0, R1 ; Underflow will occur on conversion
038D 861      SUBW      #^X300, R0 ; R1 = (pi/180 - 2^-6)*|x|
038D 862      ADDF      R1, R0    ; R0 = |X|*2^-6
038D 863 3$:      RSB          ; R0 = SIND(|X|) = (pi/180):|X|
038D 864
038D 865
038D 866
038D 867
038D 868
038D 869
038D 870
038D 871
038D 872
038D 873
038D 874
038D 875
038D 876
038D 877
038D 878
038D 879
038D 880
038D 881
038D 882
038D 883
038D 884
038D 885
038D 886
038D 887
038D 888
038D 889
038D 890
038D 891
038D 892
038D 893
038D 894
038D 895
038D 896
038D 897
038D 898
038D 899
038D 900
038D 901
038D 902
038D 903
038D 904
038D 905
038D 906
038D 907
038D 908
038D 909
038D 910
038D 911
038D 912
038D 913
038D 914
038D 915
038D 916
038D 917
038D 918
038D 919
038D 920
038D 921
038D 922
038D 923
038D 924
038D 925
038D 926
038D 927
038D 928
038D 929
038D 930
038D 931
038D 932
038D 933
038D 934
038D 935
038D 936
038D 937
038D 938
038D 939
038D 940
038D 941
038D 942
038D 943
038D 944
038D 945
038D 946
038D 947
038D 948
038D 949
038D 950
038D 951
038D 952
038D 953
038D 954
038D 955
038D 956
038D 957
038D 958
038D 959
038D 960
038D 961
038D 962
038D 963
038D 964
038D 965
038D 966
038D 967
038D 968
038D 969
038D 970
038D 971
038D 972
038D 973
038D 974
038D 975
038D 976
038D 977
038D 978
038D 979
038D 980
038D 981
038D 982
038D 983
038D 984
038D 985
038D 986
038D 987
038D 988
038D 989
038D 990
038D 991
038D 992
038D 993
038D 994
038D 995
038D 996
038D 997
038D 998
038D 999
038D 1000

```

```

      03F1 865      .SBTTL MTH$COSD_R4
      03F1 866
      03F1 867 ; This routine computes the COSD of the F-format value of R0. The computation
      03F1 868 ; is performed one of two ways depending on the size of the input argument, X:
      03F1 869 ; Details are given in the discussion on MTH$SIND_R4.
      03F1 870
      03F1 871 MTH$COSD_R4::
50      8000 8F 53 03F1 872      TSTF      R0      ; Check for reserved operand
50      4334 8F B1 03F3 873      BICW      #^X8000, R0 ; R0 = !X!
      0000069C'EF 14 03F8 874      CMPW      #W 45, R0 ; Compare 45 to !X!
      0000069C'EF 16 03FD 875      BGTR      SMALL_COSD ;
      0000069C'EF 16 03FF 876      JSB       REDUCE_DEGREES ; R3/R4 = octant/reduced argument
      0405 877
      0405 878 EVAL_COSD:
07      00 53 8F 0405 879      CASEB     R3, #0, #7
      0434' 0409 880 1$:      .WORD     P_COS_D-1$
      04D4' 040B 881      .WORD     P_SIN_D-1$
      04D1' 040D 882      .WORD     N_SIN_D-1$
      0480' 040F 883      .WORD     N_COS_D-1$
      0480' 0411 884      .WORD     N_COS_D-1$
      04D1' 0413 885      .WORD     N_SIN_D-1$
      04D4' 0415 886      .WORD     P_SIN_D-1$
      0434' 0417 887      .WORD     P_COS_D-1$
      0419 888
      0419 889
      0419 890 SMALL_COSD:
50      2EE13D65 8F 51 0419 891      CMPF      #LF_SMALLD, R0 ; Compare 180/pi*2^-12 with !X!
      00000000'00 14 0420 892      BGTR      1$ ; Check if polyinomial evaluation is
      00000000'00 14 0422 893      MOVL     R0, R4 ; necessary.
      00000000'00 31 0425 894      BRW      P_COS_D ; POLY needed
      00000000'00 50 0428 895 1$:      MOVF     #T, R0 ; R0 = 1. = COSD(!X!)
      00000000'00 05 042B 896      RSB
      00000000'00 05 042C 897

```



```

042C 899
042C 900
042C 901      .SBTTL  REDUCE_MEDIUM
042C 902
042C 903
042C 904      ; This routine assumes that the absolute value of the argument, X, is in R0
042C 905      ; and that pi/4 =< |X| < 9*pi/4. It returns a double precision reduced
042C 906      ; argument, Y, in R3/R4, and the octant bits in R2.
042C 907
042C 908      ; The reduced argument is obtained by locating the octant that X is in through
042C 909      ; a binary search and then subtracting off a suitable multiple of pi/2
042C 910
042C 911
042C 912  REDUCE_MEDIUM:
50 53D1417B 51 D4 042C 913      CLRL      R1      ; R0/R1 = |X|
      8F 51 042E 914      CMPF      #LF_5_PI_OV_4, R0
      29 15 0435 915      BLEQ      5$,
50 CBE44116 8F 51 0437 916      CMPF      #LF_3_PI_OV_4, R0
      10 15 043E 917      BLEQ      3$,
53 50 FBBC CF 63 0440 918      SUBD3     D PI_OV_2, R0, R3
      04 18 0446 919      BGEQ      2$,
      52 01 D0 0448 920      MOVL      #1, R2
      05 044B 921      RSB
      044C 922
      52 02 D0 044C 923 2$:      MOVL      #2, R2
      05 044F 924      RSB
      0450 925
53 50 FBB4 CF 63 0450 926 3$:      SUBD3     D PI, R0, R3
      04 18 0456 927      BGEQ      4$,
      52 03 D0 0458 928      MOVL      #3, R2
      05 045B 929      RSB
      045C 930
      52 04 D0 045C 931 4$:      MOVL      #4, R2
      05 045F 932      RSB
      0460 933
50 EDDF41AF 8F 51 0460 934 5$:      CMPF      #LF_7_PI_OV_4, R0
      10 15 0467 935      BLEQ      7$,
53 50 FBA3 CF 63 0469 936      SUBD3     D 3_PI_OV_2, R0, R3
      04 18 046F 937      BGEQ      6$,
      52 05 D0 0471 938      MOVL      #5, R2
      05 0474 939      RSB
      0475 940
      52 06 D0 0475 941 6$:      MOVL      #6, R2
      05 0478 942      RSB
      0479 943
53 50 FB9B CF 63 0479 944 7$:      SUBD3     D 2_PI, R0, R3
      04 18 047F 945      BGEQ      8$,
      52 07 D0 0481 946      MOVL      #7, R2
      05 0484 947      RSB
      0485 948
      52 D4 0485 949 8$:      CLRL      R2
      05 0487 950      RSB
      0488 951
      0488 952
      0488 953
      0488 954
      0488 955

```

```

0488 957      .SBTTL  REDUCE_LARGE
0488 958
0488 959      :
0488 960      : This routine is used to reduce large arguments (|X| >= 9*pi/4) modulo pi/4.
0488 961      : It returns the reduced argument, Y, in R3/R4 in units of cycles, and returns
0488 962      : the octant bits, I, in R2.
0488 963      :
0488 964      : The method of reduction is as follows:
0488 965
0488 966      x*(4/pi) = 2^n*f*(4/pi) where n is an integer and 1/2 <= f < 1
0488 967      = 2^(n-24)*(2^24*f)*(4/pi)
0488 968      = (2^24*f)*(2^(n-24)*4/pi)
0488 969      = K*C, where K = 2^24*f is an integer and C = 2*(n-24)*4/pi
0488 970      Let L = K*C modulo 8, where 0 <= L < 8, and let I = the integer(L) and
0488 971      h = fract(L), then if I is even Y = h, otherwise Y = 1-h
0488 972      :
0488 973      : CONSTANTS:
0488 974
00001E80 0488 975      L_INT WEIGHT = ^X1E80      ; weights exponent by 61
00001000 0488 976      W_TERM WEIGHT = ^X1000     ; weights exponent by 32
00004000 0488 977      W_MAX WEIGHT = ^X4000     ; maximum unbiased exponent
00000019 0488 978      W_ADJUST  = ^X19        ; Used to locate binary point in
0488 979      ; MTH$AL_4_OV_PI table
00000000 00006080 0488 980 D_2_TO_64:
0488 981      .QUAD  ^X00000000000006080 ; 2^64
0490 982
0490 983
0490 984
0490 985 REDUCE_LARGE:
0490 986 :
0490 987 : The first step is to obtain the location of the binary point in the represen-
0490 988 : tation of C = 2^(n-p)*(4/pi) in two parts - the number of longwords from
0490 989 : the start and the number of bits from the most significant bit of the next
0490 990 : longword. Also K = 2^24*f must be obtained.
0490 991 :
0490 992      ROTL  #-7, R0, R2      ; Get exp field in first byte of R2
0495 993      SUBW  #W_ADJUST, R2   ; Unbias exp and adjust for leading
0498 994      ; zeroes. R2 = location of binary
0498 995      ; point
0498 996      ROTL  #-3, R2, R3     ; Divide R2 by 32 and mull by 4 to get
049D 997      BICL  #^XFFFFFFE3, R3 ; R3 = # of longwords (in bytes) to
04A4 998      ; binary point.
04A4 999      MOVAL MTH$AL_4_OV_PI, R1 ; Get base address of MTH$AL_4_OV_PI
04AB 1000      ; table
04AB 1001      SUBL  R3, R1          ; R1 points to 1st quadword of interest
04AE 1002      BICB  #^XE0, R2     ; R2(7:0) = # of bits within longword
04B2 1003
04B2 1004      BICW  #^X7F80, R0     ; Clear exponent field
04B7 1005      BISW  #^X4C00, R0   ; R0 = 2^24*f
04BC 1006      CVTFL R0, R0      ; R0 = K
04BF 1007
04BF 1008 :
04BF 1009 : The next step is to generate an approximation to C, call it C' to be used
04BF 1010 : in computing x*4/pi. C' will consist of the first three integer bits of
04BF 1011 : C and the first 61 fraction bits of C. These bits will be obtained from a
04BF 1012 : constant stored in the integer array MTH$AL_4_OV_PI.
04BF 1013 :

```

```

04BF 1014 : NOTE: The ASHQ, ADDL, and MULL instructions in the follow sections may
04BF 1015 : result in an integer overflow trap. The overflow incurred is intentional,
04BF 1016 : so that the IV bit must be turned off. The IV bit is not restored until
04BF 1017 : after all of the necessary fraction bits have been generated.
04BF 1018 :
6E   FFFFFFFD 7E   DC   04BF 1019   MOVPSL  -(SP)           ; Put current PSL on stack
      8F   CA   04C1 1020   BICL   #^C<PSLSM_IV>, (SP) ; (SP) = current IV bit
      20   B9   04C8 1021   BICPSW #PSLSM_IV      ; Clear integer overflow bit
04CA 1022
04CA 1023 :
04CA 1024 : Get C'' = C(0):C(1) in R3/R4. C(0) and C(1) are unsigned integers generated
04CA 1025 : from the binary representation of C. The high three bits of C(0)
04CA 1026 : are the the first three bits to the left of the binary point of C. Bits 28:0
04CA 1027 : of C(0) and bits 31:0 are the first 61 bits to the right of the binary point
04CA 1028 : of C.
04CA 1029 :
04CA 1030 :
7E   61   52   79   04CA 1031   ASHQ   R2, (R1), -(SP) ; Shift the proper quadword so that
      51   04   C2   04CE 1032   : (SP)+4 has C(0) in it
      53   61   52   79   04D1 1033   SUBL   #4, R1          ; R1 points to next quadword in
04D1 1034   : MTHSAL 4_OV_PI table
04D1 1035   ASHQ   R2, (R1), R3  ; Shift quadword to R3 so that R4 has
04D5 1036   : C(1) in it
04D5 1037 :
04D5 1038 :
04D5 1039 : Generate the low 64 bits of the product K*C'' = L. This product is
04D5 1040 : equivalent to multiplying K times C'' modulo 8. The result of the
04D5 1041 : product is in R3:R4 with bits 63:61 the octant bits, 60:24 the valid fraction
04D5 1042 : bits of the product and 23:0 non-valid fraction bits that will be used later
04D5 1043 : if more fraction bits need to be generated.
04D5 1044 :
04D5 1045 :
53   00   54   50   7A   04D5 1046   EMUL   R0, R4, #0, R3  ; R3/R4 = K*C(1)
      03   18   04DA 1047   BGEQ   1$             ; Check if R4 was negative
      54   50   C0   04DC 1048   ADDL   R0, R4         ; Adjust for signed mult. error
6E   50   04   AE   C5   04DF 1049 1$: MULL3  4(SP), R0, (SP) ; (SP) = Low 32 bits of K*C(0)
      54   6E   C0   04E4 1050   ADDL   (SP), R4      ; R3/R4 = L = low order bits of K*C''
04E7 1051
04E7 1052 :
04E7 1053 : At this point there may or may not be enough valid bits in R3/R4 to generate
04E7 1054 : Y. If the first 7 fraction bits are all 1's or 0's, there a possibility of
04E7 1055 : loss of significance when computing Y. Consequently, we must check for loss
04E7 1056 : of significance before converting R3/R4 to Y and I.
04E7 1057 :
04E7 1058 :
6E   54   00200000 8F   C1   04E7 1059   ADDL3  #^X200000, R4, (SP) ; If the first 8 fraction bits are 1's
6E   6E   3FC00000 8F   D3   04EF 1060   BITL   #^X3FC00000, (SP) ; and the reduced arg = 1-f or the
      47   12   04F6 1061   BNEQ   CONVERT       ; first 8 bit are 0 and the reduced
04F8 1062   : arg = f, then (and only then) bits
04F8 1063   : 29:22 are 0 and significance will
04F8 1064   : be lost.
04F8 1065 :
04F8 1066 :
04F8 1067 : More bits need to be generated to cover the loss of significance. There are
04F8 1068 : not enough registers to hold all the potential extra bits, so that the bits
04F8 1069 : already generated must be put on the stack.
04F8 1070 :

```

```

    5E 04 C2 04F8 1071          SUBL #4, SP          ; Allocate an extra longwords on stack
    6E 53 7D 04FB 1072          ;                   ; for later use
    0000065A'EF 16 04FE 1073          ;                   ; Put L onto stack
    6E 54 C0 0504 1074          JSB GEN_MORE_BITS ; R3/R4 contain 56 additional bits
    05 1E 0507 1075          ADDL R4, (SP)      ; Add new high order bits to old low
    04 AE D6 0509 1076          BCC 2$           ; order bits
    2C 11 050C 1077          INCL 4(SP)       ; Check for possible carry and adjust
    14 04 AE 1D E0 050E 1078          BRB 4$          ; high order bits accordingly
    0513 1079          ;                   ; A carry indicates a minor loss of
    0513 1080          ;                   ; significance has occurred
    0513 1081          2$: BBS #29, 4(SP), 3$ ; Check for leading ones or zeros
    0513 1082          ; Lost significance due to leading zeros
    54 04 AE 14 00 EA 0513 1083          FFS #0, #20, 4(SP), R4 ; If at least one bit is set. This
    1F 12 0519 1084          BNEQ 4$          ; means lost significance was minor.
    6E 1FFFFFFF 8F D1 051B 1085          CMPL #^X1FFFFFFF, (SP) ; One of the three high bits is set.
    16 15 0522 1086          BLEQ 4$          ; lost significance was minor.
    00DE 31 0524 1087          BRW LEADING_ZEROS
    0527 1088          ; Lost significance due to leading ones
    0527 1089          3$: FFC #0, #20, 4(SP), R4 ; If at least one bit is clear. This
    0527 1090          BNEQ 4$          ; means lost significance was minor.
    54 04 AE 14 00 EB 0527 1091          CMPL #^XE0000000, (SP) ; One of the three high bits is clear.
    08 12 052D 1092          BGEQ 4$          ; lost significance was minor.
    6E E0000000 8F D1 052F 1093          BRB LEADING_ONES
    02 1E 0536 1094          4$: MOVQ (SP), R3 ; Move bits to registers
    4B 11 0538 1095          TSTL (SP)+ ; Remove extra longword from stack
    53 6E 7D 053A 1101          5E 08 C0 053F 1106 CONVERT:
    8E D5 053D 1102          ADDL #8, SP ; Clear stack
    053F 1103          ;
    053F 1104          ; Convert bit string to double precision reduced argument.
    053F 1105          ;
    0542 1106          ; Isolate octant bits and convert low order bits to double precision.
    0542 1107          ;
    0542 1108          ;
    0542 1109          ;
    0542 1110          ;
    0542 1111          ;
    0542 1112          ;
    0542 1113          ;
    52 54 03 1D EF 0542 1114          EXTZV #29, #3, R4, R2 ; R2 = octant bits
    50 53 6E 0547 1115          CVTLD R3, R0 ; R0/R1 = 2^61*FLO
    02 18 054A 1116          BGEQ 1$ ; Check for signed conversion error
    54 D6 054C 1117          INCL R4 ; and adjust R5 accordingly
    1A 52 E9 054E 1118          1$: BLBC R2, 2$ ; Check for odd or even octant bits
    0551 1119          ;
    0551 1120          ; Octant bits are odd. Reduced argument equals 1 - f.
    0551 1121          ;
    54 E0000000 8F C8 0551 1122          BISL #^XE0000000, R4 ; Set octant bits
    53 53 54 6E 0558 1123          CVTLD R4, R3 ; R3/R4 = -2^29*(1 - FHI)
    1000 8F A0 055B 1124          ADDW #W_TERM_WEIGHT, R3 ; R3/R4 = -2^61*(1 - FHI)
    53 53 50 60 0560 1125          ADDD R0, R3 ; R3/R4 = -2^61*(1 - FHI - FLO)
    9E80 8F A2 0563 1126          SUBW #^X9E80, R3 ; R3/R4 = 1 - F
    00E9 31 0568 1127          BRW RESTORE
    
```

```

056B 1128
056B 1129 : Octant bits are even. Reduced argument equals f
056B 1130
54 E000000 8F CA 056B 1131 2$: BICL #^XE000000, R4 ; Clear octant bits
    53 53 54 6E 0572 1132 CVTLD R4, R3 ; R3/R4 = 2^29*FHI
    1000 8F A0 0575 1133 ADDW #W_TERM_WEIGHT, R3 ; R3/R4 = 2^61*FHI
    53 53 50 60 057A 1134 ADDD R0, R3 ; R3/R4 = 2^61*(FHI + FLO)
53 1E80 8F A2 057D 1135 SUBW #^X1E80, R3 ; R3/R4 = F = FHI + FLO
    00CF 31 0582 1136 BRW RESTORE
0585 1137
0585 1138
0585 1139
0585 1140 :
0585 1141 : At this point it has been determined that there is a major loss of
0585 1142 : significance and the processing begins a looping phase. Each iteration of
0585 1143 : the loop will generate additional extra bits of K*C' until enough significant
0585 1144 : bits to compute Y are available. During this time the stack will contain
0585 1145 : four longwords allocated as follows:
0585 1146 :
0585 1147 : SP ----> L2 : L1 and L2 are the last 2 longwords worth of
0585 1148 : L1 : significant bits generated
0585 1149 : L0 : 1st three bits of L0 are the octant bits
0585 1150 : W : Weighting factor - keeps track of the
0585 1151 : number of iterations that have failed
0585 1152 : to produce enough significant bits.
0585 1153
0585 1154
0585 1155 LEADING_ONES:
0585 1156
0585 1157 : If processing continues here it is known that the loss of significance is due
0585 1158 : to a string of leading ones.
0585 1159
0C AE 7E 53 D0 0585 1160 MOVL R3, -(SP) ; Put L2 onto stack
    00001E80 8F D0 0588 1161 MOVL #L_INT_WEIGHT, 12(SP) ; W = exp bias for last longword
    0590 1162 ; of the product K*C'
0590 1163
04 AE FFE0000 8F D1 0590 1164 LOOP_1: CMPL #^XFFE0000, 4(SP) ; Check L2 for enough significant bits
    3F 1A 0598 1165 BGTRU CONVERT_1B ; Enough bits. Convert to floating.
    0000065A EF 16 059A 1166 JSB GEN_MORE_BITS ; R3/R4 contains an 56 additional 56
    54 6E C0 05A0 1167 ; bits of K*C'.
    05A0 1168 ADDL (SP), R4 ; Add low order bits of existing
    05A3 1169 ; product, L2, to high order bits of
    05A3 1170 ; new product, R4.
    05A3 1171 BCC 1$ ; Check for carry on previous ADDL
    04 AE 0B 1E 05A3 1171 BCC 1$ ; Add carry into L1
    06 1E 05A8 1172 INCL 4(SP) ; Check for carry on previous ADDL
    08 AE 0B 1E 05A8 1173 BCC 1$ ; Add carry into L0. If carry
    0087 31 05AA 1174 INCL 8(SP) ; propagates to L0 enough fraction
    05AD 1175 BRW CONVERT_OA ; bits are available to produce
    0580 1176 ; equivalent argument
    0580 1177
    0580 1178
04 AE FFFFFFF 8F D1 0580 1179 1$: CMPL #-1, 4(SP) ; Check for L1 all 1's
    1C 1A 0588 1180 BGTRU CONVERT_1A ; Not all 1's. Enough precision bits
    05BA 1181 ; to compute Y
    05BA 1182 MOVQ R3, (SP) ; Compress representation of L
FFCB OC AE 1000 8F 6E 53 7D 05BA 1182 ACBW #W_MAX_WEIGHT, #W_TERM_WEIGHT, 12(SP), LOOP_1 ; Increment weighting factor. If
    4000 8F 3D 058D 1183
    05CB 1184

```

```

05C8 1185 ; weighting factor is greater than
05C8 1186 ; 128 then no more bits need to be
05C8 1187 ; generated.
05C8 1188
05C8 1189
05C8 1190 : The weighting factor is greater than 128. This means that the reduced
05C8 1191 : argument is either not distinguishable from 1 or too small to be represented
05C8 1192 : in F-format (i.e. underflow.) Zero is returned in R3 for the reduced
05C8 1193 : argument to signal this occurrence. Note that under these conditions the
05C8 1194 : correct function value is one of the values 0, +/-1. The
05C8 1195 : correct choice is determined by the calling program based on the octant bits
05C8 1196 : returned in R1.
05C8 1197
52 08 AE 03 53 D4 05C8 1198 CLRL R3 ; Reduced argument is zero
SE 1D EF 05CA 1199 EXTZV #29, #3, 8(SP), R2 ; R2 = octant bits
007E 10 CO 05D0 1200 ADDL #16, SP ; Clear stack
31 05D3 1201 BRW RESTORE
05D6 1202
05D6 1203 CONVERT_1A:
6E 53 D0 05D6 1204 MOVL R3, (SP) ; Put L2 onto stack
05D9 1205 CONVERT_1B:
50 8E 6E 05D9 1206 CVTLD (SP)+, R0 ; R0/R1 = 2^W*L2
02 18 05DC 1207 BGEQ 1$ ; Check for signed conversion error
6E D6 05DE 1208 INCL (SP) ; and adjust L1 accordingly.
53 6E 6E 05E0 1209 1$: CVTLD (SP), R3 ; Convert L1 to double and check for
07 18 05E3 1210 BGEQ 2$ ; signed conversion error.
53 9000 8F A0 05E5 1211 ADDW #^X9000, R3 ; R3/R4 = 2^W*(2^64 - L1)
08 11 05EA 1212 BRB 3$
05EC 1213
53 53 03E8 8F A0 05EC 1214 2$: ADDW #1000, R3 ; R3/R4 = 2^W*L1
FE92 CF 53 63 05F1 1215 SUBD3 R3, D_2_TO_64, R3 ; R3/R4 = 2^W*(2^64 - L1)
05F7 1216
52 8E 53 50 62 05F7 1217 3$: SUBD R0, R3 ; R3/R4 = 2^W*(2^64 - L1 - L2)
03 1D EF 05FA 1218 EXTZV #29, #3, (SP)+, R2 ; R2 = octant bits
53 8E C2 05FF 1219 SUBL (SP)+, R3 ; R3/R4 = Y
004F 31 0602 1220 BRW RESTORE
0605 1221
0605 1222
0605 1223 LEADING_ZEROS:
0605 1224
0605 1225
0605 1226 : If processing continues here it is known that the loss of significance is due
0605 1227 : to a string of leading zeros. Note that it is known that the loop for
0605 1228 : leading zeros will terminate before an underflow condition occurs so that the
0605 1229 : loop does not include a test for underflow.
0605 1230
0C AE 7E 53 D0 0605 1231 MOVL R3, -(SP) ; Put L2 onto stack
00001E80 8F D0 0608 1232 MOVL #L_INT_WEIGHT, 12(SP) ; W = exp bias for last longword
0610 1233 ; of the product K*C'
0610 1234
04 AE 001FFFFFF 8F D1 0610 1235 LOOP_0: CMPL #^X001FFFFFF, 4(SP) ; Check L2 for enough fraction bits
0000065A'EF 20 19 0618 1236 BLSS CONVERT_0B ; Enough bits. Convert to floating
061A 1237 JSB GEN_MORE_BITS ; R3/R4 contain an additional 56 bits
0620 1238 ; of K*C'.
54 6E C0 0620 1239 ADDL (SP), R4 ; Add low order bits of existing
0623 1240 ; product, (SP), to high order bits
0623 1241 ; of new product, R4.

```

```

03 1E 0623 1242      BCC      1$      ; Check for carry on previous ADDL
04 AE D6 0625 1243      INCL      4(SP)    ; Add in carry. Note: High bits of R4
      0628 1244      ; are 0 so no carry possible on INCL
      54 D5 0628 1245 1$: TSTL      R4      ; Check for all 0's
      0B 12 062A 1246      BNEQ     CONVERT_OA ; Not all 0's. Enough precision bits.
OC AE 6E 53 7D 062C 1247      MOVQ     R3, (SP)   ; Compress representation of L
      1000 8F A0 062F 1248      ADDW     #W_TERM_WEIGHT, 12(SP) ; Increment weighting factor.
      D9 11 0635 1249      BRB      LOOP_0
      0637 1250
      0637 1251 CONVERT_OA:
6E 54 D0 0637 1252      -MOVL    R4, (SP)   ; Put L2 onto stack
      50 6E 6E 063A 1253 CONVERT_OB:
      02 18 063D 1255      -CVTLD   (SP), R0   ; R0/R1 = 2^W*L2
      6E D6 063F 1256      BGEQ     1$      ; If L2 is negative, adjust L1 to
      0641 1257      INCL     (SP)   ; to reflect signed rather than
      53 6E 6E 0641 1258 1$: CVTLD   (SP), R3   ; unsigned conversion
      1000 8F A0 0644 1259      ADDW     #W_TERM_WEIGHT, R3 ; Convert L1 to double
      53 50 60 0649 1260      ADD     R0, R3     ; R3/R4 = 2^W*L1
52 8E 03 1D EF 064C 1261      EXTZV   #29, #3, (SP)+, R2 ; R3/R4 = 2^W*(L1 + L2)
      53 8E C2 0651 1262      SUBL    (SP)+, R3  ; R2 = octant bits
      0654 1263      ; R3/R4 = Y
      0654 1264
      0654 1265 RESTORE:
      SE 6E B8 0654 1266      BISPSW  (SP)      ; Restore IV bit and exit
      04 C0 0656 1267      ADDL    #4, SP   ; Remove mask from stack
      05 0659 1268      RSB
      065A 1269
      065A 1270 GEN_MORE_BITS:
      065A 1271
      065A 1272 ;
      065A 1273 ; This subroutine generates 56 extra fraction bits and stores them in R3/R4
      065A 1274 ;
      065A 1275 ;
      51 04 C2 065A 1276      SUBL    #4, R1     ; Adjust pointer to next quadword
53 53 61 52 79 065D 1277      ASHQ   R2, (R1), R3 ; Put next coefficient in R4
      00 54 50 7A 0661 1278      EMUL   R0, R4, #0, R3 ; Generate the fraction bits in R3/R4
      03 18 0666 1279      BGEQ   1$      ; Check for signed arithmetic error
      54 50 C0 0668 1280      ADDL   R0, R4   ; Negative factor, adjust R4
      05 066B 1281 1$: RSB
      066C 1282
      066C 1283
      066C 1284
      066C 1285

```

```

066C 1287          .SBTTL REDUCE_DEGREES
066C 1288
066C 1289 ; This routine assumes that the absolute value of the argument is in R0.
066C 1290 ; The reduction process is performed in two stages. The first stage of
066C 1291 ; the reduction reduces the argument modulo 360 to a value less than 2^24,
066C 1292 ; and the second stage reduces the argument to less than 45.
066C 1293
066C 1294 ; Constants used in this reduction:
066C 1295 ;
066C 1296
0008 0004 0002 0001 066C 1297 POWER_MOD_360_0: ; Powers of 2 modulo 360 for t1 = 0
0080 0040 0020 0010 066C 1298      .WORD 1, 2, 4, 8
00F8 0130 0098 0100 0674 1299      .WORD 16, 32, 64, 128
067C 1300      .WORD 256, 152, 304, 248
0684 1301
0684 1302 POWER_MOD_360_1: ; Powers of 2 modulo 360 for t1 <> 0
0008 0088 0110 0088 0684 1303      .WORD 136, 272, 184, 8
0080 0040 0020 0010 068C 1304      .WORD 16, 32, 64, 128
00F8 0130 0098 0100 0694 1305      .WORD 256, 152, 304, 248
069C 1306
069C 1307 REDUCE_DEGREES:
50 4C80 8F B1 069C 1308      CMPW #^X4C80, R0 ; Compare |x| with 2^24
06A1 1309      BGTR LAST_STEP ; Branch to special logic for med arg
06A3 1310
06A3 1311 ;
06A3 1312 ; It is assumed here that the argument is greater than 2^24.
06A3 1313 ;
06A3 1314 ; The argument is reduced as follows:
06A3 1315 ; Let x = 2^t*f, where t > 24 and 1/2 <= f < 1. And let J = 2^24*f =
06A3 1316 ; 2^15*J1 + J2 and K = 2^(t-24). Since 2^15 = 8 modulo 360, we have that
06A3 1317 ; J = 8*J1 + J2 modulo 360. Now let t' = t - 24 = 12*t1 + t2. Note that
06A3 1318 ; (2^12)^2 = 2^24 = (2^9)*(2^15) = (2^9)*(2^3) = 2^12 modulo 360. Hence,
06A3 1319 ; if t1 is not zero, K = 2^t' = 2^(12*t1+t2) = [2^(12*t1)]*[2^t2] =
06A3 1320 ; [2^12]*[2^t2] = 136*2^t2 modulo 360. For t1 = 0, K = 2^t2. Consequently
06A3 1321 ; define K' congruent to 2^t2 if t1=0 and congruent to 136*2^t2 otherwise,
06A3 1322 ; where 0 <= K' < 360. Then x' = K'*(8*J1 + J2) is congruent to x modulo
06A3 1323 ; 360 and x' < 2^24
06A3 1324
06A3 1325
51 50 00007F80 8F CB 06A3 1326      BICL3 #^X7F80, R0, R1 ;
51 4C00 8F AB 06AB 1327      BISW #^X4C00, R1 ; R1 = J
50 51 C2 06B0 1328      SUBL R1, R0 ; R0 = t'*2^7
06B3 1329
52 51 7FFF0000 8F CB 06B3 1330      BICL3 #^X7FFF0000, R1, R2 ; R2 = J1*2^15
51 52 42 06BB 1331      SUBF R2, R1 ; R1 = J2
52 0600 8F A2 06BE 1332      SUBW #^X600, R2 ; R2 = J1
51 52 40 06C3 1333      ADDF R2, R1 ; R1 = 8*J1 + J2 = J modulo 360
06C6 1334
50 50 F9 8F 9C 06C6 1335      ROTL #-7, R0, R0 ; R0 = t'
52 50 0C 87 06CB 1336      DIVB3 #12, R0, R2 ; R2 = t1
52 0C 84 06CF 1337      MULB #12, R2 ; R2 = 12*t1
50 52 82 06D2 1338      SUBB R2, R0 ; R0 = t2
06D5 1339
06D5 1340      TSTB R2 ; Check for t1 = 0 and choose K'
07 12 06D7 1341      BNEQ 1$ ; accordingly
50 8F AF40 4D 06D9 1342      CVTWF POWER_MOD_360_0[R0], R0 ; R0 = K'
05 11 06DE 1343      BRB 2$

```



```

50  A0 AF40 4D 06E0 1344 1$: CVTWF POWER_MOD_360_1[R0], R0 ; R0 = K'
    50  51 44 06E5 1345 2$: MULF R1, R0 ; R0 = X' (mod 45) 0 =< R0 < 2^24
    06E8 1346
    06E8 1347
    06E8 1348 LAST_STEP:
    06E8 1349 : Argument reduction scheme for arguments with absolute value less than 2^24
    06E8 1350 : The reduced argument Y is computed as follows:
    06E8 1351 :   Let I = int(X/45)
    06E8 1352 :   if I is even
    06E8 1353 :     then Y = X - 45*I
    06E8 1354 :   else Y = (I+1)*45 - x
    06E8 1355
    06E8 1356
    06E8 1357
53  50  B6 8F 0B603DB6 8F 54 06E8 1358 EMCDF #LF_1_OV_45, #X_1_OV_45, R0, R3, R4
    54  53  16 53 E9 06F3 1359 : R3 = I = integer part of ix!/45
    54  54  01 C1 06F6 1361 : Branch if octant bits are even
    54  54  54 4E 06FA 1362 : R4 = I + 1
    54  00004334 8F 44 06FD 1363 :
    54  54  50 42 0704 1364 : R4 = 45*(I+1)
    53  F8 8F 8A 0707 1365 : R4 = Y
    05  070B 1366 : Save only last three octant bits
    070C 1367
    54  54  53 4E 070C 1368 EVEN: CVTLF R3, R4 : R4 = I
    54  0000C334 8F 44 070F 1369 : MULF #LF_M45, R4 : R4 = -45*I
    54  54  50 40 0716 1370 : ADDF R0, R4 : R4 = Y
    53  F8 8F 8A 0719 1371 : BICB #^XF8, R3 : Save only last three octant bits
    05  071D 1372
    071E 1373
    
```

```

071E 1375
071E 1376          .SBTTL RADIAN_POLYNOMIALS          ; Polynomials for arguments in radians
071E 1377
071E 1378
071E 1379
071E 1380          ;
071E 1381          ; Polynomial evaluation for COS(Y) for Y in radians
071E 1382          ;
071E 1383
071E 1384 P_COS_R:
          53 8000 8F AA 071E 1385          BICW      #^X8000, R3          ; R3/R4 = !Y!
          53 4000 8F B1 0723 1386          CMPW      #^X4000, R3          ; Compare 1/2 with !Y!
          1D 18 0723 1387          BGEQ      LEQL_HALF          ; Sufficent overhang is available
          53 53 64 072A 1388 NEEDS_DOUBLE:
          53 53 64 072A 1389          MULD      R3, R3          ; R3/R4 = Y*Y
          53 DD 072D 1390 NEEDS_DOUBLE SINCOS:
          53 DD 072D 1391          PUSHL     R3          ; Save high half of Y*Y
          53 04 53 55 072F 1392          POLYF     R3, #COSLENR2-1, COSTBR2: R0 = Y^4*Q(Y*Y)
          53 8E 00000080 8F C3 0735 1393          SUBL3     #^X80, (SP)+, R3          ; R3/R4 = Y*Y/2
          53 08 62 073D 1394          SUBD      #1, R3          ; R3/R4 = Y^2/2 - 1
          50 53 62 0740 1395          SUBD      R3, R0          ; R0/R1 = COS(Y)
          50 50 76 0743 1396          CVTDF     R0, R0          ; R0 = COS(Y)
          05 0746 1397          RSB
          54 53 76 0747 1398 LEQL_HALF:
          54 54 44 074A 1400          CVTDF     R3, R4          ; R4 = !Y!
          54 54 44 074A 1400          MULF     R4, R4          ; R4 = Y*Y
          F8CD CF 03 54 55 074D 1401          POLYF     R4, #COSLENR1-1, COSTBR1: R0 = COS(Y)
          05 0753 1402          RSB
          0754 1403
          0754 1404
          0754 1405          ;
          0754 1406          ; Polynomial evaluation for -COS(Y)
          0754 1407          ;
          0754 1408
          0754 1409 N_COS_R:
          53 8000 8F AA 0754 1410          BICW      #^X8000, R3          ; R3/R4 = !Y!
          53 4000 8F B1 0759 1411          CMPW      #^X4000, R3          ; Compare 1/2 with !Y!
          1D 18 075E 1412          BGEQ      1$          ; Sufficent overhang is available
          53 53 64 0760 1413          MULD      R3, R3          ; R3/R4 = Y*Y
          53 DD 0763 1414          PUSHL     R3          ; Save high half of Y*Y
          53 04 53 55 0765 1415          POLYF     R3, #COSLENR2-1, COSTBR2: R0 = Y^4*Q(Y*Y)
          53 8E 00000080 8F C3 076B 1416          SUBL3     #^X80, (SP)+, R3          ; R3/R4 = Y*Y/2
          53 08 62 0773 1417          SUBD      #1, R3          ; R3/R4 = -(1 - Y^2/2)
          53 50 62 0776 1418          SUBD      R0, R3          ; R3/R4 = -COS(Y)
          50 53 76 0779 1419          CVTDF     R3, R0          ; R0 = -COS(Y)
          05 077C 1420          RSB
          54 53 76 077D 1421          1$:
          54 54 44 0780 1423          CVTDF     R3, R4          ; R4 = !Y!
          54 54 44 0780 1423          MULF     R4, R4          ; R4 = Y*Y
          F897 CF 03 54 55 0783 1424          POLYF     R4, #COSLENR1-1, COSTBR1: R0 = COS(Y)
          50 8000 8F AC 0789 1425          XORW2     #^X8000, R0          ; R0 = -COS(Y)
          05 078E 1426          RSB
          078F 1427
          078F 1428          ;
          078F 1429          ; Polynomial evaluation for -SIN(Y)
          078F 1430          ;
          078F 1431

```

```

      53      8000 8F      AC      078F 1432 N_SIN_R:
      078F 1433      XORW  #^X8000, R3
      0794 1434
      0794 1435      :
      0794 1436      : Polynomial evaluation for SIN(Y)
      0794 1437      :
      0794 1438
      0794 1439 P_SIN_R:
F8A5 CF      53      53      DD      0794 1440      PUSHL  R3      : Save high half of Y
      03      53      44      0796 1441      MULF   R3, R3      : R3 = Y*Y
      53      8E      55      0799 1442      POLYF  R3, #SINLENR-1, SINTBR : R0 = P(Y*Y)
      50      53      44      079F 1443      MOVL   (SP)+, R3      : R3/R4 = Y
      50      53      60      07A2 1444      MULF   R3, R0      : R0 = Y*P(Y*Y)
      50      50      76      07A5 1445      ADDD   R3, R0      : R0/R1 = SIN(Y)
      05      05      07AB 1446      CVTDF  R0, R0      : R0 = SIN(Y)
      07AC 1447
      07AC 1448
      07AC 1449
      07AC 1450
  
```

```

                                .SBTTL CYCLE_POLYNOMIALS ; Polynomials for arguments in cycles
07AC 1452
07AC 1453
07AC 1454
07AC 1455
07AC 1456 ;
07AC 1457 ; Polynomial evaluation for COS(Y) for Y in cycles
07AC 1458 ;
07AC 1459 ;
07AC 1460 P_COS_C:
53 F9834022 8F 51 07AC 1461 CMPF #LF_2_OV_PI, R3 ; Compare 2/pi with |Y|
                                BGEQ LEQL_2_OV_PI ; Sufficient overhang is available
                                MULF R3, R3 ; R3/R4 = Y*Y
                                PUSHL R3 ; Save high half of Y*Y
53 F8A8 CF 04 53 55 07BA 1465 POLYF R3, #COSLENC2-1, COSTBC2; R0 = Y^4*Q(Y*Y)
8E 0000100 8F C3 07C0 1466 SUBL3 #^X100, (SP)+, R3 ; R3/R4 = Y*Y/4
                                SUBD #1, R3 ; R3/R4 = Y^2/4 - 1
                                SUBD R3, R0 ; R0/R1 = COS(Y)
                                CVTDF R0, R0 ; R0 = COS(Y)
                                RSB
07D1 1470
07D2 1471 LEQL_2_OV_PI:
F876 CF 54 53 76 07D2 1472 CVTDF R3, R4 ; R4 = |Y|
                                MULF R4, R4 ; R4 = Y*Y
                                POLYF R4, #COSLENC1-1, COSTBC1; R0 = COS(Y)
                                RSB
07DF 1476
07DF 1477
07DF 1478 ;
07DF 1479 ; Polynomial evaluation for -COS(Y)
07DF 1480 ;
07DF 1481 ;
07DF 1482 N_COS_C:
53 F9834022 8F 51 07DF 1483 CMPF #LF_2_OV_PI, R3 ; Compare 2/pi with |Y|
                                BGEQ 1$ ; Sufficient overhang is available
                                MULF R3, R3 ; R3/R4 = Y*Y
                                PUSHL R3 ; Save high half of Y*Y
53 F875 CF 04 53 55 07ED 1487 POLYF R3, #COSLENC2-1, COSTBC2; R0 = Y^4*Q(Y*Y)
8E 0000100 8F C3 07F3 1488 SUBL3 #^X100, (SP)+, R3 ; R3/R4 = Y*Y/4
                                SUBD #1, R3 ; R3/R4 = -(1 - Y^2/4)
                                SUBD R0, R3 ; R3/R4 = -COS(Y)
                                CVTDF R3, R0 ; R0 = -COS(Y)
                                RSB
                                1$:
                                CVTDF R3, R4 ; R4 = |Y|
                                MULF R4, R4 ; R4 = Y*Y
                                POLYF R4, #COSLENC1-1, COSTBC1; R0 = COS(Y)
                                XORW2 #^X8000, R0 ; R0 = -COS(Y)
                                RSB
0816 1498
0817 1499
0817 1500 ;
0817 1501 ; Polynomial evaluation for -SIN(Y)
0817 1502 ;
0817 1503 ;
0817 1504 N_SIN_C:
53 8000 8F AC 0817 1505 XORW #^X8000, R3 ; R3/R4 = - Y
081C 1506
081C 1507 ;
081C 1508 ; Polynomial evaluation for SIN(Y)

```

```

      7E 53 7D 081C 1509 ;
      53 53 44 081C 1510 ;
      53 53 C81C 1511 P_SIN_C:
      50 6E 55 081C 1512 MOVQ R3, -(SP) ; Save argument
53 6E 00000100 8F 44 081F 1513 MULF R3, R3 ; R3 = Y*Y
      53 53 55 0822 1514 POLYF R3, #SINLENC-1, SINTBC ; R0 = P(Y*Y)
      50 50 44 0828 1515 MULF (SP), R0 ; R0 = Y*P(Y*Y)
      53 53 C3 082B 1516 SUBL3 #*X100, (SP), R3 ; R3/R4 = Y/4
      50 53 62 0833 1517 SUBD (SP)+, R3 ; R3/R4 = 3/4*Y
      50 53 62 0836 1518 SUBD R3, R0 ; R0/R1 = SIN(Y)
      76 0839 1519 CVTDF R0, R0 ; R0 = SIN(Y)
      05 083C 1520 RSB
      083D 1521
```



```

08F5 1587
08F5 1588
08F5 1589      .SBTTL  DEGENERATE_SOLUTIONS
08F5 1590
08F5 1591 P_ONE:
50 08 50 08F5 1592      MOVF  #1, R0      ; Answer is 1
05 08F8 1593      RSB
08F9 1594
08F9 1595
08F9 1596 N_ONE:
50 0000C080 8F 50 08F9 1597      MOVF  #-1, R0     ; Answer is -1
05 0900 1598      RSB
0901 1599
0901 1600
0901 1601 UNFL:
0901 1602      :
0901 1603      : Underflow; if user has FU set, signal error.  Always return 0.0
0901 1604      :
0901 1605      :
00000000'GF 52 DC 0901 1605      MOVPSL R2      ; R2 = user's or jacket routine's PSL
04 00 FB 0903 1606      CALLS #0, G^MTH$$JACKET_TST ; R0 = TRUE if JSB from jacket routine
52 04 50 E9 090A 1607      BLBC  R0, 10$      ; branch if user did JSB
04 AD 3C 090D 1608      MOVZWL SF$W_SAVE_PSW(FP), R2 ; get user PSL saved by CALL
50 D4 0911 1609 10$:      CLRL  R0      ; R0 = result. LIB$SIGNAL will save in
0913 1610      ; CHF$L_MCH_R0/R1 so any handler can
0913 1611      ; fixup
0D 52 06 E1 0913 1612      BBC  #6, R2, 20$      ; has user enabled floating underflow?
6E DD 0917 1613      PUSHL (SP)      ; yes, return PC from special routine
7E 00'8F 9A 0919 1614      MOVZBL #MTH$K_FLOUNDMAT, -(SP) ; trap code for hardware floating
091D 1615      ; underflow convert to MTH$_FLOUNDMAT
091D 1616      ; (32-bit VAX-11 exception code)
00000000'GF 02 FB 091D 1617      CALLS #2, G^MTH$$SIGNAL ; signal (condition, PC)
05 0924 1618 20$:      RSB      ; return
0925 1619
0925 1620      .END
    
```

M
V

M
-
O
T
M

MTH\$SINCOS
Symbol table

CONVERT	0000053F	R	02	MTH\$\$JACKET_HND	*****	X	02
CONVERT_OA	00000637	RR	02	MTH\$\$JACKET_TST	*****	X	00
CONVERT_OB	0000063A	RR	02	MTH\$\$SIGNAL	*****	X	00
CONVERT_1A	000005D6	RR	02	MTH\$AL_4_OV_PI	*****	X	00
CONVERT_1B	000005D9	R	02	MTH\$COS	000000F0	RG	02
COSD	= 0000000C			MTH\$COSD	00000134	RG	02
COSDLN1	= 00000003			MTH\$COSD_R4	000003F1	RG	02
COSDLN2	= 00000004			MTH\$COS_R4	000002B6	RG	02
COSDTB1	0000008C	R	02	MTH\$K_FLOUNDMAT	*****	X	00
COSDTB2	0000009C	R	02	MTH\$SIN	000000DC	RG	02
COSINE	= 0000000C			MTH\$SINCOS	000000C0	RG	02
COSLENC1	= 00000005			MTH\$SINCOSD	00000104	RG	02
COSLENC2	= 00000005			MTH\$SINCOSD_R5	0000033D	RG	02
COSLENR1	= 00000004			MTH\$SINCOS_R5	00000148	RG	02
COSLENR2	= 00000005			MTH\$SIND	00000120	RG	02
COSTBC1	00000054	R	02	MTH\$SIND_R4	0000038D	RG	02
COSTBC2	00000068	RR	02	MTH\$SIN_R4	0000020F	RG	02
COSTBR1	00000020	RR	02	M_COS	000002D5	R	02
COSTBR2	00000030	RR	02	M_SIN	0000023A	R	02
DEGENERATE_CASE_COS	00000329	R	02	NEEDS_DOUBLE	0000072A	R	02
DEGENERATE_CASE_SIN	000002A2	R	02	NEEDS_DOUBLE_SINCOS	0000072D	R	02
D_2_PI	00000018	R	02	NEG_SIND	0000039B	R	02
D_2_TO_64	00000488	RR	02	N_COS_C	000007DF	R	02
D_3_PI_OV_2	00000010	RR	02	N_COS_D	00000889	R	02
D_PI	00000008	RR	02	N_COS_R	00000754	R	02
D_PI_OV_2	00000000	RR	02	N_ONE	000008F9	R	02
EVAL_COSD	00000405	R	02	N_SIN_C	00000817	R	02
EVAL_SIND	000003AD	R	02	N_SIN_D	000008DA	R	02
EVEN	0000070C	R	02	N_SIN_R	0000078F	R	02
GEN_MORE_BITS	0000065A	R	02	POS_SIN	00000222	R	02
LARGE_COS	0000030B	RR	02	POS_SINCOS	0000015C	R	02
LARGE_SIN	00000284	RR	02	POS_SIND	000003A0	R	02
LARGE_SINCOS	000001EB	RR	02	POWER_MOD_360_0	0000066C	R	02
LAST_STEP	000006E8	RR	02	POWER_MOD_360_1	00000684	R	02
LEADING_ONES	00000585	RR	02	PSLSM_IV	= 00000020		
LEADING_ZEROS	00000605	RR	02	P_COS_C	000007AC	R	02
LEQL_2_OV_PI	000007D2	RR	02	P_COS_D	0000083D	RR	02
LEQL_HALF	00000747	R	02	P_COS_R	0000071E	RR	02
LF_1_OV_45	= 0B603DB6			P_ONE	000008F5	RR	02
LF_2_OV_PI	= F9834022			P_SIN_C	0000081C	RR	02
LF_3_PI_OV_4	= CBE44116			P_SIN_D	000008DD	RR	02
LF_45	= 00004334			P_SIN_R	00000794	RR	02
LF_5_PI_OV_4	= 53D1417B			REDUCE_DEGREES	0000069C	RR	02
LF_7_PI_OV_4	= EDDF41AF			REDUCE_LARGE	00000490	RR	02
LF_90_OV_PI	= 2EE142E5			REDUCE_MEDIUM	0000042C	RR	02
LF_9_PI_OV_4	= 31D641E2			RESTORE	00000654	RR	02
LF_CONVERT	= A3513BEF			RETURN	C00008F4	RR	02
LF_M45	= 0000C334			SFSW_SAVE_PSW	= 00000004		
LF_PI_OV_4	= 0FDB4049			SIN	0000021D	R	02
LF_SMALLD	= 2EE13D65			SINCOS	00000157	R	02
LF_SMALLEST_DEG	= 2EE10365			SINCOSD	00000350	R	02
LONG	= 00000004			SIND	= 00000008		
LOOP_0	00000610	R	02	SINDLN	= 00000003		
LOOP_1	00000590	RR	02	SINDTB	000000B0	R	02
L_COS	00000311	R	02	SINE	= 00000008		
L_INT_WEIGHT	= 00001E80			SINLENC	= 00000004		
L_SIN	0000028A	R	02	SINLENR	= 00000004		

MTH\$SINCOS
Symbol table

```

SINTBC          0000007C R    02
SINTBR          00000044 R    02
SMALL_COS       000002E9 R    02
SMALL_COSD      00000419 R    02
SMALL_SIN       0000024E R    02
SMALL_SINCOS    00000194 R    02
SMALL_SINCOSD   00000375 R    02
SMALL_SIND      000003C1 R    02
UNFL           00000901 R    02
W_45           = 00004334
W_ADJUST       = 00000019
W_MAX_WEIGHT   = 00004000
W_TERM_WEIGHT  = 00001000
X              = 00000004
X_1_OV_45     = 000000B6
  
```

+-----+
! 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	00000925 (2341.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.05	00:00:00.64
Command processing	115	00:00:00.72	00:00:03.68
Pass 1	190	00:00:05.06	00:00:17.24
Symbol table sort	0	00:00:00.23	00:00:00.54
Pass 2	287	00:00:03.33	00:00:13.57
Symbol table output	16	00:00:00.13	00:00:00.48
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	643	00:00:09.54	00:00:36.30

The working set limit was 1500 pages.
 33661 bytes (66 pages) of virtual memory were used to buffer the intermediate code.
 There were 20 pages of symbol table space allocated to hold 195 non-local and 49 local symbols.
 1680 source lines were read in Pass 1, producing 33 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	5

131 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS\$:MTH\$SINCOS/OBJ=OBJ\$:MTH\$SINCOS MSRC\$:MTH\$JACKET/UPDATE=(ENH\$:MTH\$JACKET)+MS

0263 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION CONFIDENTIAL AND PROPRIETARY

The image displays a grid of 100 terminal windows, arranged in 10 rows and 10 columns. Each window shows a different output from a command, likely related to file management or directory listings. The windows are titled with various alphanumeric strings followed by "LIS", such as "MTHMINO LIS", "MTHMSG LIS", "MTHMAX1 LIS", "MTHJIDMNT LIS", "MTHMSGDEF LIS", "MTHJHNT LIS", "MTHSGN LIS", "MTHMAX0 LIS", "MTHMOD LIS", "MTHSIGNAL LIS", "MTHSQRTR2 LIS", "MTHJNINT LIS", "MTHSINCOs LIS", "MTHSQR LIS", "MTHININT LIS", "MTHRANDOM LIS", "MTHJIGNNT LIS", "MTHMINI LIS", "MTHJISGN LIS", "MTHSIGN LIS", and "MTHSINH LIS". Each window contains lines of text, including file names, sizes, and dates, representing the output of a specific command. The overall appearance is that of a multi-tasking environment where numerous processes are running simultaneously.