```
MMM          MMM  TTTTTTTTTTTTTTT   HHH          HHH  RRRRRRRRRRR     TTTTTTTTTTTTTTT   LLL
MMM          MMM  TTTTTTTTTTTTTTT   HHH          HHH  RRRRRRRRRRRR    TTTTTTTTTTTTTTT   LLL
MMM          MMM  TTTTTTTTTTTTTTT   HHH          HHH  RRRRRRRRRRRR    TTTTTTTTTTTTTTT   LLL
MMMMMM    MMMMMM        TTT         HHH          HHH  RRR         RRR       TTT         LLL
MMMMMM    MMMMMM        TTT         HHH          HHH  RRR         RRR       TTT         LLL
MMMMMM    MMMMMM        TTT         HHH          HHH  RRR         RRR       TTT         LLL
MMM   MMM   MMM         TTT         HHH          HHH  RRR         RRR       TTT         LLL
MMM   MMM   MMM         TTT         HHH          HHH  RRR         RRR       TTT         LLL
MMM   MMM   MMM         TTT         HHH          HHH  RRR         RRR       TTT         LLL
MMM         MMM         TTT         HHHHHHHHHHHHHHHH  RRRRRRRRRRR           TTT         LLL
MMM         MMM         TTT         HHHHHHHHHHHHHHHH  RRRRRRRRRRRR          TTT         LLL
MMM         MMM         TTT         HHHHHHHHHHHHHHHH  RRRRRRRRRRRR          TTT         LLL
MMM         MMM         TTT         HHH          HHH  RRR   RRR             TTT         LLL
MMM         MMM         TTT         HHH          HHH  RRR   RRR             TTT         LLL
MMM         MMM         TTT         HHH          HHH  RRR   RRR             TTT         LLL
MMM         MMM         TTT         HHH          HHH  RRR      RRR          TTT         LLL
MMM         MMM         TTT         HHH          HHH  RRR      RRR          TTT         LLL
MMM         MMM         TTT         HHH          HHH  RRR      RRR          TTT         LLL
MMM         MMM         TTT         HHH          HHH  RRR         RRR       TTT         LLLLLLLLLLLLLLLL
MMM         MMM         TTT         HHH          HHH  RRR         RRR       TTT         LLLLLLLLLLLLLLLL
MMM         MMM         TTT         HHH          HHH  RRR         RRR       TTT         LLLLLLLLLLLLLLLL
```

```
MM      MM  TTTTTTTTTT  HH      HH    AAAAAA    TTTTTTTTTT    AAAAAA    NN      NN
MM      MM  TTTTTTTTTT  HH      HH    AAAAAA    TTTTTTTTTT    AAAAAA    NN      NN
MMMM  MMMM      TT      HH      HH  AA      AA      TT      AA      AA  NN      NN
MMMM  MMMM      TT      HH      HH  AA      AA      TT      AA      AA  NN      NN
MM  MM  MM      TT      HH      HH  AA      AA      TT      AA      AA  NNNN    NN
MM  MM  MM      TT      HH      HH  AA      AA      TT      AA      AA  NNNN    NN
MM      MM      TT      HHHHHHHHHH  AA      AA      TT      AA      AA  NN  NN  NN
MM      MM      TT      HHHHHHHHHH  AA      AA      TT      AA      AA  NN  NN  NN
MM      MM      TT      HH      HH  AAAAAAAAAA      TT      AAAAAAAAAA  NN    NNNN
MM      MM      TT      HH      HH  AAAAAAAAAA      TT      AAAAAAAAAA  NN    NNNN
MM      MM      TT      HH      HH  AA      AA      TT      AA      AA  NN      NN      ....
MM      MM      TT      HH      HH  AA      AA      TT      AA      AA  NN      NN      ....
MM      MM      TT      HH      HH  AA      AA      TT      AA      AA  NN      NN      ....
MM      MM      TT      HH      HH  AA      AA      TT      AA      AA  NN      NN      ....

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
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  MTHSATAN        ; Floating Point Arc Tangent Functions
0000     2                                  ; (ATAN,ATAN2,ATAND,ATAND2)
0000     3          .IDENT /2-005/          ; File: MTHATAN.MAR  EDIT:  RNH2005
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$ATAN is a function which returns the floating point arc tangent
0000    34 ; value (in radians) of its single precision floating point argument.
0000    35 ; MTH$ATAN2  is two arguments floating point arctangent. The call is
0000    36 ; standard call-by-reference.
0000    37 ; MTH$ATAN_R4  is  a  special  routine  which is the same as MTH$ATAN
0000    38 ; except a faster non-standard JSB call is  used with the argument in
0000    39 ; R0 and no registers are saved.
0000    40 ;
0000    41 ; MTH$ATAND is a function which returns the floating point arc tangent
0000    42 ; value  (in degrees) of its single precision floating point argument.
0000    43 ; MTH$ATAND2 is two arguments floating point arctangent.  The  call is
0000    44 ; standard call-by-reference.
0000    45 ; MTH$ATAND_R4  is  a  special  routine which is the same as MTH$ATAND
0000    46 ; except a faster non-standard JSB call is  used  with the argument in
0000    47 ; R0 and no registers are saved.
0000    48 ;
0000    49 ;--
0000    50 ;
0000    51 ; VERSION: 01
0000    52 ;
0000    53 ; HISTORY:
0000    54 ; AUTHOR:
0000    55 ;       Peter Yuo, 15-Oct-76: Version 01
0000    56 ;
0000    57 ; MODIFIED BY:
```

```
0000     58 ;
0000     59 ; 01-1  Peter Yuo, 22-May-77
0000     60 ;
0000     61 ;--
0000     62 ;
0000     63 ; VERSION: 02
0000     64 ;
0000     65 ; HISTORY:
0000     66 ; AUTHOR:
0000     67 ;       Bob Hanek, 04-Jun-81: Version 02
0000     68 ;
0000     69 ; MODIFIED BY:
0000     70 ;
0000     71 ;
```

```
0000     73          .SBTTL  HISTORY · Detai'ed Current Edit History
0000     74
0000     75
0000     76 ; ALGORITHMIC DIFFERENCES FROM FP-11/C ROUTINE:
0000     77 ;       1. To avoid various flags subroutine calls have been used.
0000     78 ;
0000     79 ; Edit History for Version 01 of MTH$ATANATAN2
0000     80 ;
0000     81 ; 01-1  Code saving from code review March 1977
0000     82 ;       Add jacket routine to MTH$ATAN2
0000     83 ;       Use proper form of literals so MTH$ATAN2 will work
0000     84 ;
0000     85 ; 01-5  Signal (0, 0) as INVALID ARG TO MATH LIBRARY.  TNH 16-June-78
0000     86 ; 01-6  Move .ENTRY mask definition to module header.  TNH 14-Aug-78
0000     87 ; 1-007 - Update version number and copyright notice.  JBS 16-NOV-78
0000     88 ; 1-008 - Change symbol MTH__INVARG to MTH$K_INVARGMAT.  JBS 07-DEC-78
0000     89 ; 1-009 - Add "_" to the PSECT directive.  JBS 21-DEC-78
0000     90 ; 1-010 - Define all global symbols using .EXTRN.  JBS 19-JUN-1979
0000     91 ; 1-011 - Added degree entry points. RNH 15-MAR-1981
0000     92 ;
0000     93 ;
0000     94 ; Edit History for Version 02 of MTH$ATANATAN2
0000     95 ;
0000     96 ;--
0000     97 ; 2-002 - Use G^ addressing for externals.  SBL 24-Aug-1981
0000     98 ; 2-003 - Added vectored entry point MTH$$AB_ATAN_V.  RNH 29-Sep-81
0000     99 ; 2-004 - Changed MTH$ATAND2 to MTH$ATAN2D to conform to original spec.
0000    100 ;        RNH 05-Oct-81
0000    101 ; 2-005 - Un-did previous edit to conform with PL/1 names.
0000    102 ;       - Modified small argument logic to avoid an FPA bug in the POLY
0000    103 ;         instruction.  RNH 17-Dec-81
```

```
                     0000   105          .SBTTL  DECLARATIONS    ; Declarative Part of Module
                     0000   106
                     0000   107 ;
                     0000   108 ; INCLUDE FILES:         MTHJACKET.MAR
                     0000   109 ;
                     0000   110 ;
                     0000   111 ; EQUATED SYMBOLS:
                     0000   112 ;
0000401C             0000   113          ACMASK = ^M<IV, R2, R3, R4>     ; .ENTRY register mask, int ovf enabled
0FDB40C9             0000   114          LF_PI_OVER_2_HI = ^X0FDB40C9    ; High 24 bits of pi/2
BD2EB43B             0000   115          LF_PI_OVER_2_LO = ^XBD2EB43B    ; Low 24 bits of pi/2
0FDBC0C9             0000   116          LF_MPI_OVER_2   = ^X0FDBC0C9    ; -pi/2
0FDB40C9             0000   117          LF_PI_OVER_2    = ^X0FDB40C9    ; pi/2
00004434             0000   118          LF_180          = ^X00004434    ; 90
0000C3B4             0000   119          LF_M90          = ^X0000C3B4    ; -90
0FDB4149             0000   120          LF_PI           = ^X0FDB4149    ; pi
000043B4             0000   121          LF_90           = ^X000043B4    ; 90
                     0000   122
                     0000   123 ;
                     0000   124 ; MACROS:       none
                     0000   125 ;
                     0000   126 ; PSECT DECLARATIONS:
                     0000   127
                00000000     128          .PSECT  _MTH$CODE       PIC,SHR,LONG,EXE,NOWRT
                     0000   129                                          ; program section for math routines
                     0000   130 ;
                     0000   131 ; OWN STORAGE:  none
                     0000   132 ;
                     0000   133 ; EXTERNALS:
                     0000   134 ;
                     0000   135          .EXTRN  MTH$$SIGNAL             ; Signal a severe error
                     0000   136          .EXTRN  MTH$K_INVARGMAT         ; Invalid argument to math library
                     0000   137          .DSABL  GBL                     ; No other externals allowed
                     0000   138 ;
                     0000   139 ; CONSTANTS:
                     0000   140
                     0000   141 ;
                     0000   142 ;   The MTH$$AB_ATAN table is a table of byte entries used to obtain an index
                     0000   143 ;   into the ATAN_TABLE.  MTH$$AB_ATAN is indexed using the low order bits of
                     0000   144 ;   the  exponent  field  and the high order bits of the fraction field.  The
                     0000   145 ;   MTH$$AB_ATAN table is independent of the data type and is used by all  of
                     0000   146 ;   the arctangent routines.
                     0000   147 ;
                     0000   148 ;
                     0000   149 MTH$$AB_ATAN_V::                     ; Simulated vector entry for G and
                 00000004'   150          .LONG   MTH$$AB_ATAN-.     ;   H routines
                     0004   151 MTH$$AB_ATAN::
 09 06 06 03 03 00 00 00     0004   152          .BYTE   ^X00, ^X00, ^X00, ^X03, ^X03, ^X06, ^X06, ^X09
 12 0F 0F 0C 0C 0C 09 09     000C   153          .BYTE   ^X09, ^X09, ^X0C, ^X0C, ^X0C, ^X0F, ^X0F, ^X12
 18 18 18 15 15 15 12 12     0014   154          .BYTE   ^X12, ^X12, ^X15, ^X15, ^X15, ^X18, ^X18, ^X18
 21 1E 1E 1E 1B 1B 1B 1B     001C   155          .BYTE   ^X1B, ^X1B, ^X1B, ^X1B, ^X1E, ^X1E, ^X1E, ^X21
 24 24 24 24 21 21 21 21     0024   156          .BYTE   ^X21, ^X21, ^X21, ^X21, ^X24, ^X24, ^X24, ^X24
 27 27 27 27 27 27 24 24     002C   157          .BYTE   ^X24, ^X24, ^X27, ^X27, ^X27, ^X27, ^X27, ^X27
    27 27 27 27 27 27 27     0034   158          .BYTE   ^X27, ^X27, ^X27, ^X27, ^X27, ^X27, ^X27
                             003B   159
                             003B   160
```

```
                     003B   162 ;
                     003B   163 ;
                     003B   164 ;   ************************  Constants for ATAN  ****************************
                     003B   165 ;
                     003B   166 ;
                     003B   167 ;   Each entry of the ATAN_TABLE contains  the the values of XHI, ATAN_XHI_LO
                     003B   168 ;   and ATAN_XHI_HI respectively.  The table is indexed by a pointer obtained
                     003B   169 ;   from  the MTH$$AB_ATAN table.  NOTE: For performance reasons it is import-
                     003B   170 ;   ant to have the ATAN_TABLE longword aligned.
                     003B   171 ;
                     003B   172 ;
                     003B   173          .ALIGN  LONG
                     003C   174
                     003C   175 ATAN_TABLE:
                     003C   176 ; Entry  0
F87E3ED7             003C   177          .LONG   ^XF87E3ED7      ; 0.105454430E+00
0A99B21B             0040   178          .LONG   ^X0A99B21B      ; -.225614927E-08
2CE73ED7             0044   179          .LONG   ^X2CE73ED7      ; 0.105066113E+00
                     0048   180 ; Entry  1
FB703F03             0048   181          .LONG   ^XFB703F03      ; 0.128888845E+00
F372325E             004C   182          .LONG   ^XF372325E      ; 0.324436344E-08
422F3F03             0050   183          .LONG   ^X422F3F03      ; 0.128182158E+00
                     0054   184 ; Entry  2
F63E3F1F             0054   185          .LONG   ^XF63E3F1F      ; 0.156212777E+00
B326B253             0058   186          .LONG   ^XB326B253      ; -.308063752E-08
ADF13F1E             005C   187          .LONG   ^XADF13F1E      ; 0.154960409E+00
                     0060   188 ; Entry  3
E4EC3F47             0060   189          .LONG   ^XE4EC3F47      ; 0.195209205E+00
4A61B2AA             0064   190          .LONG   ^X4A61B2AA      ; -.495610708E-08
69623F45             0068   191          .LONG   ^X69623F45      ; 0.192784816E+00
                     006C   192 ; Entry  4
C3D13F7F             006C   193          .LONG   ^XC3D13F7F      ; 0.249770418E+00
678D3298             0070   194          .LONG   ^X678D3298      ; 0.443555459E-08
A30A3F7A             0074   195          .LONG   ^XA30A3F7A      ; 0.244762570E+00
                     0078   196 ; Entry  5
DB973F9F             0078   197          .LONG   ^XDB973F9F      ; 0.312222213E+00
315D3323             007C   198          .LONG   ^X315D3323      ; 0.949907264E-08
F28D3F9A             0080   199          .LONG   ^XF28D3F9A      ; 0.302631766E+00
                     0084   200 ; Entry  6
9E8E3FC7             0084   201          .LONG   ^X9E8E3FC7      ; 0.389881551E+00
F1A6335E             0088   202          .LONG   ^XF1A6335E      ; 0.129770452E-07
56713FBE             008C   203          .LONG   ^X56713FBE      ; 0.371753246E+00
                     0090   204 ; Entry  7
33B63FFF             0090   205          .LONG   ^X33B63FFF      ; 0.498441398E+00
C9A0B31E             0094   206          .LONG   ^XC9A0B31E      ; -.924265464E-08
BFB03FEC             0098   207          .LONG   ^XBFB03FEC      ; 0.462399960E+00
                     009C   208 ; Entry  8
F8EB4026             009C   209          .LONG   ^XF8EB4026      ; 0.652235687E+00
4695B38C             00A0   210          .LONG   ^X4695B38C      ; -.163302420E-07
F4394013             00A4   211          .LONG   ^XF4394013      ; 0.577945292E+00
                     00A8   212 ; Entry  9
0712405E             00A8   213          .LONG   ^X712405E       ; 0.867295384E+00
6A34B399             00AC   214          .LONG   ^X6A34B399      ; -.178598398E-07
E62C4036             00B0   215          .LONG   ^XE62C4036      ; 0.714449644E+00
                     00B4   216 ; Entry 10
CBD84095             00B4   217          .LONG   ^XCBD84095      ; 0.117028332E+01
99953108             00B8   218          .LONG   ^X99953108      ; 0.496947650E-09
```

```
1B62405D  00BC    219          .LONG   ^X1B62405D      ; 0.863699079E+00
          00C0    220 ; Entry 11
8DEB40D2  00C0    221          .LONG   ^X8DEB40D2      ; 0.164495599E+01
DBF9B3F3  00C4    222          .LONG   ^XDBF9B3F3      ; -.283889552E-07
25414083  00C8    223          .LONG   ^X25414083      ; 0.102457440E+01
          00CC    224 ; Entry 12
88054124  00CC    225          .LONG   ^X88054124      ; 0.257080197E+01
888B3433  00D0    226          .LONG   ^X888B3433      ; 0.418008703E-07
93CA4099  00D4    227          .LONG   ^X93CA4099      ; 0.119982266E+01
          00D8    228 ; Entry 13
7D0E41AB  00D8    229          .LONG   ^X7D0E41AB      ; 0.535901546E+01
DF7F33C8  00DC    230          .LONG   ^XDF7F33C8      ; 0.233846986E-07
72D240B1  00E0    231          .LONG   ^X72D240B1      ; 0.138631654E+01
          00E4    232
          00E4    233 ;
          00E4    234 ; Tables to be used in POLYF for computing ATAN.  ATANTAB1 is obtained
          00E4    235 ; from Hart et. al. (No. 4900).  ATANTAB2 is the same as ATANTAB1 except
          00E4    236 ; that C0 is set to 0
          00E4    237
          00E4    238 ATANTAB1:
B0AA3F4A  00E4    239          .LONG   ^XB0AA3F4A      ; C2 =  .19793954
A9B0BFAA  00E8    240          .LONG   ^XA9B0BFAA      ; C1 = -.33332586
00004080  00EC    241          .LONG   ^X00004080      ; C0 = 1.00000000
00000003  00F0    242 ATANLEN1 = .- ATANTAB1/4
          00F0    243
          00F0    244 ATANTAB2:
B0AA3F4A  00F0    245          .LONG   ^XB0AA3F4A      ; C2 =  .19793954
A9B0BFAA  00F4    246          .LONG   ^XA9B0BFAA      ; C1 = -.33332586
00000000  00F8    247          .LONG   ^X00000000      ; C0 =  .00000000
00000003  00FC    248 ATANLEN2 = .- ATANTAB2/4
          00FC    249
          00FC    250
```

```
                     00FC   252 ;
                     00FC   253 ;
                     00FC   254 ;   ************************  Constants for ATAND  ************************
                     00FC   255 ;
                     00FC   256 ;
                     00FC   257 ;   Each  entry of the ATAND_TABLE contains the the values of XHI, ATAND_XHI_LO
                     00FC   258 ;   and ATAND_XHI_HI respectively.  The table is indexed by a pointer  obtained
                     00FC   259 ;   from the MTH$$AB_ATAN table.  NOTE: for performance reasons it is important
                     00FC   260 ;   to have the ATAND_TABLE longword aligned.
                     00FC   261 ;
                     00FC   262
                     00FC   263 ATAND_TABLE:
                     00FC   264 ; Entry  0
F87E3ED7             00FC   265          .LONG   ^XF87E3ED7      ; 0.105454430E+00
B1EF354F             0100   266          .LONG   ^XB1EF354F      ; 0.193431092E-06
A29141C0             0104   267          .LONG   ^XA29141C0      ; 0.601984453E+01
                     0108   268 ; Entry  1
FB703F03             0108   269          .LONG   ^XFB703F03      ; 0.128888845E+00
FE1BB4C3             010C   270          .LONG   ^XFE1BB4C3      ; -.912661662E-07
047B41EB             0110   271          .LONG   ^X047B41EB      ; 0.734429693E+01
                     0114   272 ; Entry  2
F63E3F1F             0114   273          .LONG   ^XF63E3F1F      ; 0.156212777E+00
AF4932FD             0118   274          .LONG   ^XAF4932FD      ; 0.738319672E-08
0EA7420E             011C   275          .LONG   ^X0EA7420E      ; 0.887857723E+01
                     0120   276 ; Entry  3
E4EC3F47             0120   277          .LONG   ^XE4EC3F47      ; 0.195209205E+00
3623B5A7             0124   278          .LONG   ^X3623B5A7      ; -.311455636E-06
BB6B4230             0128   279          .LONG   ^XBB6B4230      ; 0.110457563E+02
                     012C   280 ; Entry  4
C3D13F7F             012C   281          .LONG   ^XC3D13F7F      ; 0.249770418E+00
4257B5C1             0130   282          .LONG   ^X4257B5C1      ; -.359973200E-06
61BE4260             0134   283          .LONG   ^X61BE4260      ; 0.140238628E+02
                     0138   284 ; Entry  5
DB973F9F             0138   285          .LONG   ^XDB973F9F      ; 0.312222213E+00
300C351B             013C   286          .LONG   ^X300C351B      ; 0.144529793E-06
B758428A             0140   287          .LONG   ^XB758428A      ; 0.173395233E+02
                     0144   288 ; Entry  6
9E8E3FC7             0144   289          .LONG   ^X9E8E3FC7      ; 0.389881551E+00
CFDD35A6             0148   290          .LONG   ^XCFDD35A6      ; 0.310711499E-06
662E42AA             014C   291          .LONG   ^X662E42AA      ; 0.212998924E+02
                     0150   292 ; Entry  7
33B63FFF             0150   293          .LONG   ^X33B63FFF      ; 0.498441398E+00
F7DAB674             0154   294          .LONG   ^XF7DAB674      ; -.912577548E-06
F2D342D3             0158   295          .LONG   ^XF2D342D3      ; 0.264935665E+02
                     015C   296 ; Entry  8
F8EB4026             015C   297          .LONG   ^XF8EB4026      ; 0.652235687E+00
ADBDB6DF             0160   298          .LONG   ^XADBDB6DF      ; -.166653592E-05
748F4304             0164   299          .LONG   ^X748F4304      ; 0.331138268E+02
                     0168   300 ; Entry  9
0712405E             0168   301          .LONG   ^X0712405E      ; 0.867295384E+00
BD24359B             016C   302          .LONG   ^XBD24359B      ; 0.290086177E-06
BD634323             0170   303          .LONG   ^XBD634323      ; 0.409349480E+02
                     0174   304 ; Entry 10
CBD84095             0174   305          .LONG   ^XCBD84095      ; 0.117028332E+01
B637B668             0178   306          .LONG   ^XB637B668      ; -.866918924E-06
F1FC4345             017C   307          .LONG   ^XF1FC4345      ; 0.494863129E+02
                     0180   308 ; Entry 11
```

```
8DEB40D2  0180   309          .LONG   ^X8DEB40D2      ; 0.164495599E+01
9881B6CC  0184   310          .LONG   ^X9881B6CC      ; -.152435689E-05
D0AE436A  0188   311          .LONG   ^XD0AE436A      ; 0.587037888E+02
          018C   312 ; Entry 12
88054124  018C   313          .LONG   ^X88054124      ; 0.257080197E+01
41353761  0190   314          .LONG   ^X41353761      ; 0.335655682E-05
7D534389  0194   315          .LONG   ^X7D534389      ; 0.687447739E+02
          0198   316 ; Entry 13
7D0E41AB  0198   317          .LONG   ^X7D0E41AB      ; 0.535901546E+01
FECD377E  019C   318          .LONG   ^XFECD377E      ; 0.379972630E-05
DC34439E  01A0   319          .LONG   ^XDC34439E      ; 0.794300842E+02
          01A4   320
          01A4   321 ;
          01A4   322 ; Tables to be used in POLYF for computing ATAND.  ATANDTAB1 is obtained
          01A4   323 ; by multiplying the coefficients given in Hart et. al. (No. 4900) by
          01A4   324 ; 180/pi.  ATANDTAB2 is the same as ATANDTAB1 except that C0 is set to
          01A4   325 ; 180/pi - 64 instead of 180/pi.
          01A4   326
          01A4   327 ATANDTAB1:
75264235  01A4   328          .LONG   ^X75264235      ; C2 =  11.341100693
C90BC298  01A8   329          .LONG   ^XC90BC298      ; C1 = -19.098165512
2EE14365  01AC   330          .LONG   ^X2EE14365      ; C0 =  57.295780182
00000003  01B0   331 ATANDLEN1 = .- ATANDTAB1/4
          01B0   332
          01B0   333 ATANDTAB2:
75264235  01B0   334          .LONG   ^X75264235      ; C2 =  11.341100693
C90BC298  01B4   335          .LONG   ^XC90BC298      ; C1 = -19.098165512
          01B8   336 PI_OV_180_M_64:
88F9C1D6  01B8   337          .LONG   ^X88F9C1D6      ; C0 =  -6.704219818
00000003  01BC   338 ATANDLEN2 = .- ATANDTAB2/4
          01BC   339
          01BC   340
```

```
                     01BC    342          .SBTTL  MTH$ATAN - Standard Single Precision Floating Arc Tangent
                     01BC    343
                     01BC    344
                     01BC    345 ;++
                     01BC    346 ; FUNCTIONAL DESCRIPTION:
                     01BC    347 ;
                     01BC    348 ; ATAN - single precision floating point function
                     01BC    349 ;
                     01BC    350 ; ATAN is computed using the following steps:
                     01BC    351 ;
                     01BC    352 ;    1. If X > 11 then
                     01BC    353 ;       a. Let W = 1/X.
                     01BC    354 ;       b. Compute ATAN(W) = W*P(W**2), where P is a polynomial of
                     01BC    355 ;          degree 2.
                     01BC    356 ;       c. Set ATAN(X) = pi/2 - ATAN(W)
                     01BC    357 ;    2. If 3/32 =< X =< 11 then
                     01BC    358 ;       a. Obtain XHI by table look-up.
                     01BC    359 ;       b. Compute Z = (X - XHI)/(1 + X*XHI).
                     01BC    360 ;       c. Compute ATAN(Z) = Z*P(Z**2), where P is a polynomial of
                     01BC    361 ;          degree 2.
                     01BC    362 ;       d. Obtain ATAN(XHI) by table look-up.  ATAN(XHI) will have
                     01BC    363 ;          two parts - the high order bits, ATAN_XHI_HI, and the low
                     01BC    364 ;          order bits, ATAN_XHI_LO.
                     01BC    365 ;       e. Compute ATAN(X) = ATAN_XHI_HI + (ATAN_XHI_LO + ATAN(Z)).
                     01BC    366 ;    3. If 0 =< X < 3/32 then
                     01BC    367 ;       a. Compute ATAN(X) = X + X*Q(X**2), where Q is a polynomial
                     01BC    368 ;          of degree 2.
                     01BC    369 ;    4. If X < 0 then
                     01BC    370 ;       a. Compute Y = ATAN(|X|) using steps 1 to 3.
                     01BC    371 ;       b. Set ATAN(X) = -Y.
                     01BC    372 ;
                     01BC    373 ;
                     01BC    374 ; CALLING SEQUENCE:
                     01BC    375 ;
                     01BC    376 ;       Arctangent.wf.v = MTH$ATAN(x.rf.r)
                     01BC    377 ;
                     01BC    378 ; INPUT PARAMETERS:
                     01BC    379
00000004             01BC    380          LONG = 4                        ; define longword multiplier
00000004             01BC    381          x = 1 * LONG                    ; x is an angle in radians
                     01BC    382 ;
                     01BC    383 ; IMPLICIT INPUTS:      none
                     01BC    384 ;
                     01BC    385 ; OUTPUT PARAMETERS:
                     01BC    386 ;
                     01BC    387 ;       VALUE:  floating arctangent angle of the argument
                     01BC    388 ;
                     01BC    389 ; IMPLICIT OUTPUTS:     none
                     01BC    390 ;
                     01BC    391 ; SIDE EFFECTS:
                     01BC    392 ;
                     01BC    393 ; Signals:      none
                     01BC    394 ;
                     01BC    395 ; NOTE: This procedure disables floating point underflow and integer
                     01BC    396 ; overflow, causes no floating overflow or other arithmetic traps, and
                     01BC    397 ; preserves enables across the call.
                     01BC    398 ;
```

```
                        01BC    399 ;---
                        01BC    400
                        01BC    401
                   401C 01BC    402         .ENTRY  MTH$ATAN, ACMASK        ; standard call-by-reference entry
                        01BE    403                                         ; disable DV (and FU), enable IV
                        01BE    404         MTH$FLAG_JACKET                 ; flag that this is a jacket procedure
                        01BE
    6D  00000000'GF      9E 01BE            MOVAB   G^MTH$$JACKET_HND, (FP)
                        01C5                                                ; set handler address to jacket
                        01C5                                                ; handler
                        01C5
                        01C5    405                                         ; in case of an error in special JSB
                        01C5    406                                         ; routine
    50  04 BC          50 01C5    407         MOVF    @x(AP), R0              ; R0 = arg
          72            10 01C9    408         BSBB    MTH$ATAN_R4             ; call special ATAN rountine
                        04 01CB    409         RET                             ; return - result in R0
                        01CC    410
```

```
                                01CC   412          .SBTTL  MTH$ATAN2 - Standard floating Arctangent With 2 Arguments
                                01CC   413 ;++
                                01CC   414 ; FUNCTIONAL DESCRIPTION:
                                01CC   415 ;
                                01CC   416 ; ATAN2  - single precision floating point function
                                01CC   417 ;
                                01CC   418 ; ATAN2(X,Y) is computed as following:
                                01CC   419 ;
                                01CC   420 ;       If Y = 0 or X/Y > 2**25, ATAN2(X,Y) = PI/2 * (sign X)
                                01CC   421 ;       If Y > 0 and X/Y =< 2**25, ATAN2(X,Y) = ATAN(X/Y)
                                01CC   422 ;       If Y < 0 and X/Y =< 2**25, ATAN2(X,Y) = PI * (sign X) + ATAN(X/Y)
                                01CC   423 ;
                                01CC   424 ;
                                01CC   425 ; CALLING SEQUENCE:
                                01CC   426 ;
                                01CC   427 ;       Arctangent2.wf.v = MTH$ATAN2(x.rf.r, y.rf.r)
                                01CC   428 ;
                                01CC   429 ; INPUT PARAMETERS:
                                01CC   430
                    00000004    01CC   431          x = 1 * LONG                    ; x is the first argument
                    00000008    01CC   432          y = 2 * LONG                    ; y is the second argument
                                01CC   433
                                01CC   434 ; SIDE EFFECTS: See description of MTH$ATAN
                                01CC   435 ;
                                01CC   436 ;--
                                01CC   437
                                01CC   438
                       401C     01CC   439          .ENTRY  MTH$ATAN2 ,ACMASK       ; standard call-by-reference entry
                                01CE   440                                          ; disable DV (and FU), enable IV
                                01CE   441          MTH$FLAG_JACKET                 ; flag that this is jacket procedure
                                01CE
     6D    00000000'GF    9E   01CE               MOVAB   G^MTH$$JACKET_HND, (FP)
                                01D5                                               ; set handler address to jacket
                                01D5                                               ; handler
                                01D5
                                01D5   442                                         ; in case of an error in special JSB routine
           50    04 BC    50   01D5   443          MOVF    @x(AP), R0              ; R0 = arg1
           51    08 BC    50   01D9   444          MOVF    @y(AP), R1              ; R1 = arg2
                                01DD   445 ;
                                01DD   446 ; Test if Y = 0 or X/Y > 2**25
                                01DD   447 ;
                          35    13   01DD   448          BEQL    INF                ; branch to INF if Y = 0
     52    50      807F    8F   AB   01DF   449          BICW3   #^X807F, R0, R2    ; R2 = exponent(X)
     53    51      807F    8F   AB   01E5   450          BICW3   #^X807F, R1, R3    ; R3 = exponent(Y)
                 52          53   A2   01EB   451          SUBW    R3, R2             ; R2 = exponent(X) - exponent(Y)
           0D00  8F          52   B1   01EE   452          CMPW    R2, #26*128        ; compare R2 with 26
                 1F          14   01F3   453          BGTR    INF                ; if X/Y > 2**25, branch to INF
                                01F5   454 ;
                                01F5   455 ; Test if Y > 0 or Y < 0
                                01F5   456 ;
                             51 B5   01F5   457          TSTW    R1                 ; test the sign of Y
                             18 14   01F7   458          BGTR    A2PLUS             ; branch to A2PLUS if Y > 0
                             50 B5   01F9   459          TSTW    R0                 ; test the sign of X
                             0A 18   01FB   460          BGEQ    A1PLUS             ; branch to A1PLUS if X >= 0
                                01FD   461 ;
                                01FD   462 ; Y < 0 and X < 0 and X/Y =< 2**25
                                01FD   463 ;
```

```
                 3B   10   01FD    464          BSBB    MTH$ATAN_R4D                ; R0 = ATAN(X/Y)
     50  0FDB4149 8F   42   01FF    465          SUBF    #LF_PI, R0                  ; R0 = -PI + ATAN(X/Y)
                 04   0206    466          RET                                 ; return
                      0207    467 ;
                      0207    468 ; Y < 0 and X > 0 and X/Y =< 2**25
                      0207    469 ;
                      0207    470 A1PLUS:
                 31   10   0207    471          BSBB    MTH$ATAN_R4D                ; R0 = ATAN(X/Y)
     50  0FDB4149 8F   40   0209    472          ADDF    #LF_PI, R0                  ; R0 = PI + ATAN(X/Y)
                 04   0210    473          RET                                 ; return
                      0211    474 ;
                      0211    475 ; Y > 0 and X/Y =< 2**25
                      0211    476 ;
                      0211    477 A2PLUS:
                 27   10   0211    478          BSBB    MTH$ATAN_R4D                ; R0 = ATAN(X/Y)
                 04   0213    479          RET                                 ; return
                      0214    480 ;
                      0214    481 ; Y = 0 or X/Y > 2**25
                      0214    482 ;
                      0214    483 INF:
                 50   B5   0214    484          TSTW    R0                          ; test the sign of X
                 0A   14   0216    485          BGTR    1$                          ; branch if X > 0
                 10   13   0218    486          BEQL    2$                          ; branch if X = 0
     50  0FDBC0C9 8F   50   021A    487          MOVF    #LF_MPI_OVER_2, R0          ; R0 = ATAN(X/Y) = -PI/2
                 04   0221    488          RET                                 ; return
                      0222    489
     50  0FDB40C9 8F   50   0222    490 1$:      MOVF    #LF_PI_OVER_2, R0           ; R0 = ATAN(X/Y) = PI/2
                 04   0229    491          RET                                 ; return
                      022A    492
                      022A    493 ;+
                      022A    494 ; Here if X = 0 and Y = 0.  Signal INVALID ARG TO MATH LIBRARY
                      022A    495 ; as a SEVERE error.
                      022A    496 ;-
                      022A    497
       50    01    0F   79   022A    498 2$:      ASHQ    #15, #1, R0                 ; R0/R1 = reserved operand which
                      022E    499                                             ; is copied to CHF$L_MCH_SAVR0/R1
                      022E    500                                             ; so a handler can fixup if wants
                      022E    501                                             ; to continue
         7E  00'8F    9A   022E    502          MOVZBL  #MTH$K_INVARGMAT, -(SP)     ; code for INVALID ARG TO MATH LIBRARY
00000000'GF    01    FB   0232    503          CALLS   #1, G^MTH$$SIGNAL           ; signal SEVERE error
                 04   0239    504          RET                                 ; return if handler continues
```

```
                                023A   506          .SBTTL  MTH$ATAN_R4 - Special ATAN routine
                                023A   507
                                023A   508 ; Special ATAN - used by the standard routine, and directly.
                                023A   509 ;
                                023A   510 ; CALLING SEQUENCES:
                                023A   511 ;       save anything needed in R0:R4
                                023A   512 ;       MOVF    ...,R0                  ; input in R0
                                023A   513 ;       JSB     MTH$ATAN_R4
                                023A   514 ;       return with result in R0
                                023A   515 ;
                                023A   516 ; Note:  This  routine  is written to avoid causing any integer overflows,
                                023A   517 ;    floating overflows, or floating underflows or divide by 0 conditions,
                                023A   518 ;    whether enabled or not.
                                023A   519 ;
                                023A   520 ; REGISTERS USED:
                                023A   521 ;       R0 - Floating argument then result
                                023A   522 ;       R0:R3 - POLYF
                                023A   523 ;       R4 - Pointer into ATAN_TABLE
                                023A   524
                                023A   525
                                023A   526
                                023A   527 MTH$ATAN_R4D:
                     50   51    46   023A   528          DIVF    R1, R0                  ; for our own use only!
                                023D   529 MTH$ATAN_R4::                               ; Special ATAN routine
                          50    53   023D   530          TSTF    R0                      ; R4 = X = argument
                          74    19   023F   531          BLSS    NEG_ARG                 ; Branch to negative argument logic
                                0241   532 ;
                                0241   533 ; Argument is positive
                                0241   534 ;
                54   50   3EC0 8F    A3   0241   535          SUBW3   #^X3EC0, R0, R4         ; Argument is less than 3/32,
                          45    19   0247   536          BLSS    SMALL                   ;  branch to small argument logic
                     54   036F 8F    B1   0249   537          CMPW    #^X036F, R4             ; Argument is greater that 11,
                          41    19   024E   538          BLSS    LARGE_ARG               ;  branch to large argument logic
                                0250   539 ;
                                0250   540 ; Logic for positive medium sized arguments.  Get pointer into ATAN_TABLE.
                                0250   541 ;
                 54   54    FC 8F    9C   0250   542          ROTL    #-4, R4, R4             ; R4 = index into MTH$$AB_ATAN table
                54   FFFFFF00 8F    CA   0255   543          BICL    #-256, R4               ; zero high order bits of index
                  54    FDA3 CF44    90   025C   544          MOVB    MTH$$AB_ATAN[R4], R4    ; R4 = offset into ATAN_TABLE
                  54    FDD5 CF44    DE   0262   545          MOVAL   ATAN_TABLE[R4], R4      ; R4 = pointer to XHI
                                0268   546 ;
                                0268   547 ; Compute Z
                                0268   548 ;
                     51    84    D0   0268   549          MOVL    (R4)+, R1               ; R1 = XHI
                52   50    51    45   026B   550          MULF3   R1, R0, R2              ; R2 = X*XHI
                          52    08    40   026F   551          ADDF    #1, R2                  ; R2 = 1 + X*XHI
                          50    51    42   0272   552          SUBF    R1, R0                  ; R0 = X - XHI
                          50    52    46   0275   553          DIVF    R2, R0                  ; R0 = Z = (X - XHI)/(1 + X*XHI)
                                0278   554 ;
                                0278   555 ; Evaluate Z*P(Z**2)
                                0278   556 ;
                     7E   50    D0   0278   557          MOVL    R0, -(SP)               ; Push Z onto the stack
                     50   50    44   027B   558          MULF    R0, R0                  ; R0 = Z**2
           FE60 CF   02   50    55   027E   559          POLYF   R0, #ATANLEN1-1, ATANTAB1
                                0284   560                                              ; R0 = P(Z**2)
                     50   8E    44   0284   561          MULF    (SP)+, R0               ; R0 = ATAN(Z) = Z*P(Z**2)
                     50   84    40   0287   562          ADDF    (R4)+, R0               ; R0 = ATAN_XHI_LO + ATAN(Z)
```

```
                     50    64    40   028A    563          ADDF    (R4), R0                ; R0 = ATAN(X) = ATAN_XHI_HI +
                                     028D    564                                           ;      (ATAN_XHI_LO + ATAN(Z))
                           05        028D    565          RSB                              ; Return
                                     028E    566
                                     028E    567
                        0096    31   028E    568 SMALL:   BRW     SMALL_ARG                ; Dummy label used to avoid adding
                                     0291    569                                           ;   an extra insrtuction in the
                                     0291    570                                           ;   medium argument logic
                                     0291    571 ;
                                     0291    572 ; Large positive argument logic.
                                     0291    573 ;
                                     0291    574
                                     0291    575 LARGE_ARG:
     54   0000C080 8F    50    47   0291    576          DIVF3   R0, #-1, R4               ; R4 = -W = -1/X
               50    54    54    45   0299    577          MULF3   R4, R4, R0               ; R0 = W**2
          FE41 CF    02    50    55   029D    578          POLYF   R0, #ATANLEN1-1, ATANTAB1
                                     02A3    579                                           ; R0 = P(W**2)
                     50    54    44   02A3    580          MULF    R4, R0                   ; R0 = ATAN(W) = -W*P(W**2)
           50    BD2EB43B 8F    40   02A6    581          ADDF    #LF_PI_OVER_2_LO, R0     ;
           50    0FDB40C9 8F    40   02AD    582          ADDF    #LF_PI_OVER_2_HI, R0     ; R0 = ATAN(X) = PI/2 - ATAN(W)
                           05        02B4    583          RSB                              ; Return
                                     02B5    584
                                     02B5    585 ;
                                     02B5    586 ; Logic for negative arguments
                                     02B5    587 ;
                                     02B5    588
                                     02B5    589 NEG_ARG:
          54   50    BEC0 8F    A3   02B5    590          SUBW3   #^XBEC0, R0, R4          ; Argument is less than 3/32,
                           6A    19   02BB    591          BLSS    SMALL_ARG                ;  branch to small argument logic
               54    036F 8F    B1   02BD    592          CMPW    #^X036F, R4              ; Argument is greater than 11,
                           3F    19   02C2    593          BLSS    N_LARGE_ARG              ;  branch to large argument logic
                                     02C4    594 ;
                                     02C4    595 ; Logic for negative medium sized arguments.  Get index into ATAN_TABLE.
                                     02C4    596 ;
      54   54    FC 8F    9C   02C4    597          ROTL    #-4, R4, R4              ; R4 = index into MTH$$AB_ATAN table
     54   FFFFFF00 8F    CA   02C9    598          BICL    #-256, R4                ; clear high order (unused) bits of ind
        54   FD2F CF44    90   02D0    599          MOVB    MTH$$AB_ATAN[R4], R4     ; R4 = offset into ATAN_TABLE
        54   FD61 CF44    DE   02D6    600          MOVAL   ATAN_TABLE[R4], R4       ; R4 = pointer to XHI
                                     02DC    601 ;
                                     02DC    602 ; Compute Z
                                     02DC    603 ;
               51    84    D0   02DC    604          MOVL    (R4)+, R1                ; R1 = XHI
          52   50    51    45   02DF    605          MULF3   R1, R0, R2               ; R2 = X*XHI
          52   08    52    43   02E3    606          SUBF3   R2, #1, R2               ; R2 = 1 - X*XHI = 1 + X*(-XHI)
               50    51    40   02E7    607          ADDF    R1, R0                   ; R0 = X + XHI = X - (-XHI)
               50    52    46   02EA    608          DIVF    R2, R0                   ; R0 = Z
                                     02ED    609 ;
                                     02ED    610 ; Evaluate Z*P(Z**2)
                                     02ED    611 ;
               7E    50    D0   02ED    612          MOVL    R0, -(SP)                ; Push Z onto the stack
               50    50    44   02F0    613          MULF    R0, R0                   ; R0 = Z**2
    FDEB CF    02    50    55   02F3    614          POLYF   R0, #ATANLEN1-1, ATANTAB1
                                     02F9    615                                           ; R0 = P(Z**2)
               50    8E    44   02F9    616          MULF    (SP)+, R0                ; R0 = ATAN(Z) = Z*P(Z**2)
               50    84    42   02FC    617          SUBF    (R4)+, R0                ; R0 = ATAN_XHI_LO + ATAN(Z)
               50    64    42   02FF    618          SUBF    (R4), R0                 ; R0 = ATAN(X) = ATAN_XHI_HI +
                                     0302    619                                           ;      (ATAN_XHI_LO + ATAN(Z))
```

```
                     05  0302   620          RSB                                 ; Return
                         0303   621 ;
                         0303   622 ; Logic for large negative  arguments
                         0303   623 ;
                         0303   624
                         0303   625 N_LARGE_ARG:
54   0000C080 8F    50   47 0303   626          DIVF3   R0, #-1, R4                 ; R4 = W = 1/|X|
          50    54   54   45 030B   627          MULF3   R4, R4, R0                  ; R0 = W**2
     FDCF CF    02   50   55 030F   628          POLYF   R0, #ATANLEN1-1, ATANTAB1
                         0315   629                                              ; R0 = P(W**2)
               50    54   44 0315   630          MULF    R4, R0                      ; R0 = ATAN(W) = W*P(W**2)
     50   BD2EB43B 8F    42 0318   631          SUBF    #LF_PI_OVER_2_LO, R0        ;
     50   0FDB40C9 8F    42 031F   632          SUBF    #LF_PI_OVER_2_HI, R0        ; R0 = ATAN(X) = ATAN(W) - PI/2
                     05  0326   633          RSB                                 ; Return
                         0327   634
                         0327   635 ;
                         0327   636 ; Small argument logic.
                         0327   637 ;
                         0327   638
                         0327   639 SMALL_ARG:
               54    50   D0   0327   640          MOVL    R0, R4                      ; R4 = argument = X
          50   8000 8F    AA   032A   641          BICW    #^X8000, R0                 ; R0 = |X|
          50   3A00 8F    B1   032F   642          CMPW    #^X3A00, R0                 ; Compare 2^-13 to |X|
                    04   19   0334   643          BLSS    1$                          ; Branch to polynomial evaluation
               50    54   D0   0336   644          MOVL    R4, R0                      ; No POLY needed.  Answer = X
                     05  0339   645          RSB                                 ;
                         033A   646
               50    50   44   033A   647 1$:      MULF    R0, R0                      ; R0 = X**2
     FDAD CF    02   50   55 033D   648          POLYF   R0, #ATANLEN2-1, ATANTAB2
                         0343   649                                              ; R0 = Q(X**2)
               50    54   44 0343   650          MULF    R4, R0                      ; R0 = X*Q(X**2)
               50    54   40 0346   651          ADDF    R4, R0                      ; R0 = ATAN(X) = X + X*Q(X**2)
                     05  0349   652          RSB                                 ; Return
                         034A   653
                         034A   654
                         034A   655
```

```
                     034A   657          .SBTTL  MTH$ATAND - Standard Single Precision Floating Arc Tangent
                     034A   658
                     034A   659
                     034A   660 ;++
                     034A   661 ; FUNCTIONAL DESCRIPTION:
                     034A   662 ;
                     034A   663 ; ATAND - single precision floating point function
                     034A   664 ;
                     034A   665 ; ATAND is computed using the following steps:
                     034A   666 ;
                     034A   667 ;    1. If X > 11 then
                     034A   668 ;       a. Let W = 1/X.
                     034A   669 ;       b. Compute ATAND(W) = W*P(W**2), where P is a polynomial of
                     034A   670 ;          degree 2.
                     034A   671 ;       c. Set ATAND(X) = pi/2 - ATAND(W)
                     034A   672 ;    2. If 3/32 =< X =< 11 then
                     034A   673 ;       a. Obtain XHI by table look-up.
                     034A   674 ;       b. Compute Z = (X - XHI)/(1 + X*XHI).
                     034A   675 ;       c. Compute ATAND(Z) = Z*P(Z**2), where P is a polynomial of
                     034A   676 ;          degree 2.
                     034A   677 ;       d. Obtain ATAND(XHI) by table look-up.  ATAND(XHI) will have
                     034A   678 ;          two parts - the high order bits, ATAND_XHI_HI, and the low
                     034A   679 ;          order bits, ATAND_XHI_LO.
                     034A   680 ;       e. Compute ATAND(X) = ATAND_XHI_HI + (ATAND_XHI_LO + ATAND(Z)).
                     034A   681 ;    3. If 0 =< X < 3/32 then
                     034A   682 ;       a. Compute ATAND(X) = 64*X + X*Q(X**2), where Q is a polynomial
                     034A   683 ;          of degree 2.
                     034A   684 ;    4. If X < 0 then
                     034A   685 ;       a. Compute Y = ATAND(|X|) using steps 1 to 3.
                     034A   686 ;       b. Set ATAND(X) = -Y.
                     034A   687 ;
                     034A   688 ;
                     034A   689 ; CALLING SEQUENCE:
                     034A   690 ;
                     034A   691 ;       Arctangent.wf.v = MTH$ATAND(x.rf.r)
                     034A   692 ;
                     034A   693 ; INPUT PARAMETERS:
                     034A   694
00000004             034A   695          LONG = 4                            ; define longword multiplier
00000004             034A   696          x = 1 * LONG                        ; x is an angle in radians
                     034A   697 ;
                     034A   698 ; IMPLICIT INPUTS:       none
                     034A   699 ;
                     034A   700 ; OUTPUT PARAMETERS:
                     034A   701 ;
                     034A   702 ;       VALUE:  floating arctangent angle of the argument (in degrees)
                     034A   703 ;
                     034A   704 ; IMPLICIT OUTPUTS:      none
                     034A   705 ;
                     034A   706 ; SIDE EFFECTS:
                     034A   707 ;
                     034A   708 ; Signals:       none
                     034A   709 ;
                     034A   710 ; NOTE: This procedure disables floating point underflow and integer
                     034A   711 ; overflow, causes no floating overflow or other arithmetic traps, and
                     034A   712 ; preserves enables across the call.
                     034A   713 ;
```

```
                                034A   714 ;---
                                034A   715
                                034A   716
                         401C   034A   717          .ENTRY  MTH$ATAND, ACMASK       ; standard call-by-reference entry
                                034C   718                                          ; disable DV (and FU), enable IV
                                034C   719          MTH$FLAG_JACKET                 ; flag that this is a jacket procedure
                                034C
    6D  00000000'GF     9E      034C               MOVAB   G^MTH$$JACKET_HND, (FP)
                                0353                                                ; set handler address to jacket
                                0353                                                ; handler
                                0353
                                0353   720                                          ; in case of an error in special JSB
                                0353   721                                          ; routine
     50  04 BC     50           0353   722          MOVF    @x(AP), R0              ; R0 = arg
                   72  10       0357   723          BSBB    MTH$ATAND_R4            ; call special ATAND rountine
                       04       0359   724          RET                             ; return - result in R0
                                035A   725
```

```
                                035A   727          .SBTTL  MTH$ATAND2 - Standard floating Arctangent With 2 Arguments
                                035A   728 ;++
                                035A   729 ; FUNCTIONAL DESCRIPTION:
                                035A   730 ;
                                035A   731 ; ATAND2  - single precision floating point function
                                035A   732 ;
                                035A   733 ; ATAND2(X,Y) is computed as following:
                                035A   734 ;
                                035A   735 ;       If Y = 0 or X/Y > 2**25, ATAND2(X,Y) = 90 * (sign X)
                                035A   736 ;       If Y > 0 and X/Y =< 2**25, ATAND2(X,Y) = ATAND(X/Y)
                                035A   737 ;       If Y < 0 and X/Y =< 2**25, ATAND2(X,Y) = 180 * (sign X) + ATAND(X/Y)
                                035A   738 ;
                                035A   739 ;
                                035A   740 ; CALLING SEQUENCE:
                                035A   741 ;
                                035A   742 ;       Arctangent2.wf.v = MTH$ATAND2(x.rf.r, y.rf.r)
                                035A   743 ;
                                035A   744 ; INPUT PARAMETERS:
                                035A   745
                     00000004   035A   746          x = 1 * LONG                    ; x is the first argument
                     00000008   035A   747          y = 2 * LONG                    ; y is the second argument
                                035A   748
                                035A   749 ; SIDE EFFECTS: See description of MTH$ATAND
                                035A   750 ;
                                035A   751 ;--
                                035A   752
                                035A   753
                        401C    035A   754          .ENTRY  MTH$ATAND2 ,ACMASK      ; standard call-by-reference entry
                                035C   755                                          ; disable DV (and FU), enable IV
                                035C   756          MTH$FLAG_JACKET                 ; flag that this is jacket procedure
                                035C
          6D    00000000'GF     9E    035C                MOVAB   G^MTH$$JACKET_HND, (FP)
                                0363                                              ; set handler address to jacket
                                0363                                              ; handler
                                0363
                                0363   757                                        ; in case of an error in special JSB
                                0363   758                                        ; routine
           50     04 BC    50  0363   759          MOVF    @x(AP), R0             ; R0 = arg1
           51     08 BC    50  0367   760          MOVF    @y(AP), R1             ; R1 = arg2
                                036B   761 ;
                                036B   762 ; Test if Y = 0 or X/Y > 2**25
                                036B   763 ;
                          35    13    036B   764          BEQL    INFD           ; branch to INFD if Y = 0
     52    50     807F   8F    AB    036D   765          BICW3   #^X807F, R0, R2 ; R2 = exponent(X)
     53    51     807F   8F    AB    0373   766          BICW3   #^X807F, R1, R3 ; R3 = exponent(Y)
                 52     53    A2    0379   767          SUBW    R3, R2          ; R2 = exponent(X) - exponent(Y)
          0D00   8F     52    B1    037C   768          CMPW    R2, #26*128     ; compare R2 with 26
                          1F    14    0381   769          BGTR    INFD           ; if X/Y > 2**25, branch to INFD
                                0383   770 ;
                                0383   771 ; Test if Y > 0 or Y < 0
                                0383   772 ;
                          51    B5    0383   773          TSTW    R1             ; test the sign of Y
                          18    14    0385   774          BGTR    A2PLUSD        ; branch to A2PLUSD if Y > 0
                          50    B5    0387   775          TSTW    R0             ; test the sign of X
                          0A    18    0389   776          BGEQ    A1PLUSD        ; branch to A1PLUSD if X >- 0
                                038B   777 ;
                                038B   778 ; Y < 0 and X < 0 and X/Y =< 2**25
```

```
                                038B   779 ;
                          3B    038B   780          BSBB    MTH$ATAND_R4D                   ; R0 = ATAND(X/Y)
      50  00004434       8F    038D   781          SUBF    #LF_180, R0                     ; R0 = -180 + ATAND(X/Y)
                          04    0394   782          RET                                     ; return
                                0395   783 ;
                                0395   784 ; Y < 0 and X > 0 and X/Y =< 2**25
                                0395   785 ;
                                0395   786 A1PLUSD:
                          31    0395   787          BSBB    MTH$ATAND_R4D                   ; R0 = ATAND(X/Y)
      50  00004434       8F    0397   788          ADDF    #LF_180, R0                     ; R0 = 180 + ATAND(X/Y)
                          04    039E   789          RET                                     ; return
                                039F   790 ;
                                039F   791 ; Y > 0 and X/Y =< 2**25
                                039F   792 ;
                                039F   793 A2PLUSD:
                          27    039F   794          BSBB    MTH$ATAND_R4D                   ; R0 = ATAND(X/Y)
                          04    03A1   795          RET                                     ; return
                                03A2   796 ;
                                03A2   797 ; Y = 0 or X/Y > 2**25
                                03A2   798 ;
                                03A2   799 INFD:
                          50    B5     03A2   800          TSTW    R0                      ; test the sign of X
                          0A    14     03A4   801          BGTR    1$                      ; branch if X > 0
                          10    13     03A6   802          BEQL    2$                      ; branch if X = 0
      50  0000C3B4       8F    50     03A8   803          MOVF    #LF_M90, R0             ; R0 = ATAND(X/Y) = -90
                          04    03AF   804          RET                                     ; return
                                03B0   805
      50  000043B4       8F    50     03B0   806 1$:      MOVF    #LF_90, R0              ; R0 = ATAND(X/Y) = 90
                          04    03B7   807          RET                                     ; return
                                03B8   808
                                03B8   809 ;+
                                03B8   810 ; Here if X = 0 and Y = 0.  Signal INVALID ARG TO MATH LIBRARY
                                03B8   811 ; as a SEVERE error.
                                03B8   812 ;-
                                03B8   813
        50  01    0F    79      03B8   814 2$:      ASHQ    #15, #1, R0                     ; R0/R1 = reserved operand which
                                03BC   815                                                 ; is col80ed to CHF$L_MCH_SAVR0/R1
                                03BC   816                                                 ; so a handler can fixup if wants
                                03BC   817                                                 ; to continue
          7E    00'8F    9A      03BC   818          MOVZBL  #MTH$K_INVARGMAT, -(SP)         ; code for INVALID ARG TO MATH LIBRARY
  00000000'GF    01       FB     03C0   819          CALLS   #1, G^MTH$$SIGNAL               ; signal SEVERE error
                          04    03C7   820          RET                                     ; return if handler continues
```

```
                        03C8    822         .SBTTL  MTH$ATAND_R4 - Special ATAND routine
                        03C8    823
                        03C8    824 ; Special ATAND - used by the standard routine, and directly.
                        03C8    825 ;
                        03C8    826 ; CALLING SEQUENCES:
                        03C8    827 ;       save anything needed in R0:R4
                        03C8    828 ;       MOVF    ...,R0                  ; input in R0
                        03C8    829 ;       JSB     MTH$ATAND_R4
                        03C8    830 ;       return with result in R0
                        03C8    831 ;
                        03C8    832 ; Note: This routine is  written  to  avoid causing any integer overflows,
                        03C8    833 ;   floating overflows, or floating underflows or divide by 0 conditions,
                        03C8    834 ;   whether enabled or not.
                        03C8    835 ;
                        03C8    836 ; REGISTERS USED:
                        03C8    837 ;       R0 - Floating argument then result
                        03C8    838 ;       R0:R3 - POLYF
                        03C8    839 ;       R4 - Pointer into ATAND_TABLE
                        03C8    840
                        03C8    841
                        03C8    842 MTH$ATAND_R4D:
            50  51  46  03C8    843         DIVF    R1, R0                  ; for our own use only!
                        03CB    844 MTH$ATAND_R4::                          ; Special ATAND routine
                50  53  03CB    845         TSTF    R0                      ; R4 = X = argument
                6D  19  03CD    846         BLSS    NEG_ARGD                ; Branch to negative argument logic
                        03CF    847 ;
                        03CF    848 ; Argument is positive
                        03CF    849 ;
      54    50  3EC0 8F  A3  03CF    850         SUBW3   #^X3EC0, R0, R4         ; Argument is less than 3/32,
                45  19  03D5    851         BLSS    SMALLD                  ;  branch to small argument logic
           54   036F 8F  B1  03D7    852         CMPW    #^X036F, R4             ; Argument is greater that 11,
                41  19  03DC    853         BLSS    LARGE_ARGD              ;  branch to large argument logic
                        03DE    854 ;
                        03DE    855 ; Logic for positive medium sized arguments.  Get pointer into ATAND_TABLE.
                        03DE    856 ;
 54   54    FC 8F    9C  03DE    857         ROTL    #-4, R4, R4             ; R4 = index into MTH$$AB_ATAN table
54   FFFFFF00 8F    CA  03E3    858         BICL    #-256, R4               ; zero high order bits of index
  54   FC15 CF44    90  03EA    859         MOVB    MTH$$AB_ATAN[R4], R4    ; R4 = offset into ATAND_TABLE
  54   FD07 CF44    DE  03F0    860         MOVAL   ATAND_TABLE[R4], R4     ; R4 = pointer to XHI
                        03F6    861 ;
                        03F6    862 ; Compute z
                        03F6    863 ;
           51   84   D0  03F6    864         MOVL    (R4)+, R1               ; R1 = XHI
      52   50   51   45  03F9    865         MULF3   R1, R0, R2              ; R2 = X*XHI
           52   08   40  03FD    866         ADDF    #1, R2                  ; R2 = 1 + X*XHI
           50   51   42  0400    867         SUBF    R1, R0                  ; R0 = X - XHI
           50   52   46  0403    868         DIVF    R2, R0                  ; R0 = Z = (X - XHI)/(1 + X*XHI)
                        0406    869 ;
                        0406    870 ; Evaluate Z*P(Z**2)
                        0406    871 ;
           7E   50   D0  0406    872         MOVL    R0, -(SP)               ; Push Z onto the stack
           50   50   44  0409    873         MULF    R0, R0                  ; R0 = Z**2
FD92 CF    02   50   55  040C    874         POLYF   R0, #ATANDLEN1-1, ATANDTAB1
                        0412    875                                         ; R0 = P(Z**2)
           50   8E   44  0412    876         MULF    (SP)+, R0               ; R0 = ATAND(Z) = Z*Q(Z**2)
           50   84   40  0415    877         ADDF    (R4)+, R0               ; R0 = ATAND_XHI_LO + ATAND(Z)
           50   64   40  0418    878         ADDF    (R4), R0                ; R0 = ATAND(X) = ATAND_XHI_HI +
```

```
                                041B   879                                     ;       (ATAND_XHI_LO + ATAND(Z))
                             05 041B   880             RSB                   ; Return
                                041C   881
                                041C   882
                          0088  31 041C   883 SMALLD:  BRW     SMALL_ARGD      ; Dummy label used to avoid adding
                                041F   884                                     ;   an extra insrtuction in the
                                041F   885                                     ;   medium argument logic
                                041F   886 ;
                                041F   887 ; Large positive argument logic.
                                041F   888 ;
                                041F   889
                                041F   890 LARGE_ARGD:
                54 0000C080 8F  50  47 041F   891             DIVF3   R0, #-1, R4     ; R4 = -W = -1/X
                      50    54  54  45 0427   892             MULF3   R4, R4, R0      ; R0 = W**2
                FD73  CF    02  50  55 042B   893             POLYF   R0, #ATANDLEN1-1, ATANDTAB1
                                0431   894                                     ; R0 = P(W**2)
                          50    54  44 0431   895             MULF    R4, R0          ; R0 = -ATAND(Z) = -Z*P(W**2)
                 50   000043B4  8F  40 0434   896             ADDF    #LF_90, R0      ; R0 = ATAND(X) = 90 - ATAND(Z)
                             05 043B   897             RSB                   ; Return
                                043C   898
                                043C   899 ;
                                043C   900 ; Logic for negative arguments
                                043C   901 ;
                                043C   902
                                043C   903 NEG_ARGD:
                54    50    BEC0  8F  A3 043C   904             SUBW3   #^XBEC0, R0, R4 ; Argument is less than 3/32,
                                  63  19 0442   905             BLSS    SMALL_ARGD      ;  branch to small argument logic
                      54    036F  8F  B1 0444   906             CMPW    #^X036F, R4     ; Argument is greater than 11,
                                  3F  19 0449   907             BLSS    N_LARGE_ARGD    ;  branch to large argument logic
                                044B   908 ;
                                044B   909 ; Logic for negative medium sized arguments.  Get index into ATAND_TABLE.
                                044B   910 ;
              54    54     FC  8F  9C 044B   911             ROTL    #-4, R4, R4     ; R4 = index into MTH$$AB_ATAN table
            54    FFFFFF00  8F  CA 0450   912             BICL    #-256, R4       ; clear high order (unused) bits of ind
              54    FBA8  CF44  90 0457   913             MOVB    MTH$$AB_ATAN[R4], R4    ; R4 = offset into ATAN_TABLE
              54    FC9A  CF44  DE 045D   914             MOVAL   ATAND_TABLE[R4], R4     ; R4 = pointer to XHI
                                0463   915 ;
                                0463   916 ; Compute Z
                                0463   917 ;
                      51    84  D0 0463   918             MOVL    (R4)+, R1       ; R1 = XHI
                52    50    51  45 0466   919             MULF3   R1, R0, R2      ; R2 = X*XHI
                52    08    52  43 046A   920             SUBF3   R2, #1, R2      ; R2 = 1 - X*XHI = 1 + X*(-XHI)
                      50    51  40 046E   921             ADDF    R1, R0          ; R0 = X + XHI = X - (-XHI)
                      50    52  46 0471   922             DIVF    R2, R0          ; R0 = Z
                                0474   923 ;
                                0474   924 ; Evaluate Z*P(Z**2)
                                0474   925 ;
                      7E    50  D0 0474   926             MOVL    R0, -(SP)       ; Push Z onto the stack
                      50    50  44 0477   927             MULF    R0, R0          ; R0 = Z**2
            FD24 CF   02    50  55 047A   928             POLYF   R0, #ATANDLEN1-1, ATANDTAB1
                                0480   929                                     ; R0 = P(Z**2)
                      50    8E  44 0480   930             MULF    (SP)+, R0       ; R0 = ATAND(Z) = Z*P(Z**2)
                      50    84  42 0483   931             SUBF    (R4)+, R0       ; R0 = ATAND_XHI_LO + ATAND(Z)
                      50    64  42 0486   932             SUBF    (R4), R0        ; R0 = ATAND(X) = ATAND_XHI_HI +
                                0489   933                                     ;       (ATAND_XHI_LO + ATAND(Z))
                             05 0489   934             RSB                   ; Return
                                048A   935 ;
```

```
                                048A   936 ; Logic for large negative  arguments
                                048A   937 ;
                                048A   938
                                048A   939 N_LARGE_ARGD:
54   0000C080 8F    50    47   048A   940         DIVF3   R0, #-1, R4              ; R4 = W = 1/!X!
          50    54    54    45   0492   941         MULF3   R4, R4, R0             ; R0 = W**2
     FD08 CF    02    50    55   0496   942         POLYF   R0, #ATANDLEN1-1, ATANDTAB1
                                049C   943                                        ; R0 = P(W**2)
               50    54    44   049C   944         MULF    R4, R0                 ; R0 = ATAND(W) = W*P(W**2)
     50   000043B4 8F    42   049F   945         SUBF    #LF_90, R0             ; R0 = ATAND(X) = ATAND(W) - 90
                           05   04A6   946         RSB                            ; Return
                                04A7   947
                                04A7   948 ;
                                04A7   949 ; Small argument logic.
                                04A7   950 ;
                                04A7   951
                                04A7   952 SMALL_ARGD:
               54    50    50   04A7   953         MOVF    R0, R4                 ; R4 = argument = X
                           28    13   04AA   954         BEQL    3$             ;
          50   8000 8F    AA   04AC   955         BICW    #^X8000, R0            ; R0 = !X!
          50   3A00 8F    B1   04B1   956         CMPW    #^X3A00, R0            ; Compare 2^-13 to !X!
                           08    19   04B6   957         BLSS    1$             ; Needs POLY
     50   54   FCFC CF    45   04B8   958         MULF3   PI_OV_180_M_64, R4, R0 ; R0 = X*[pi/180 - 64]
                           0C    11   04BE   959         BRB     2$
                                04C0   960
               50    50    44   04C0   961 1$:     MULF    R0, R0                 ; R0 = X**2
     FCE7 CF    02    50    55   04C3   962         POLYF   R0, #ATANDLEN2-1, ATANDTAB2
                                04C9   963                                        ; R0 = Q(X**2)
               50    54    44   04C9   964         MULF    R4, R0                 ; R0 = X*Q(X**2)
          54   0300 8F    A0   04CC   965 2$:     ADDW    #^X300, R4             ; R4 = X*2**6
               50    54    40   04D1   966         ADDF    R4, R0                 ; R0 = ATAND(X) = X*2**6 + X*Q(X**2)
                           05   04D4   967 3$:     RSB                            ; Return
                                04D5   968
                                04D5   969
                                04D5   970
                                04D5   971         .END
```

```
A1PLUS              00000207 R      01
A1PLUSD             00000395 R      01
A2PLUS              00000211 R      01
A2PLUSD             0000039F R      01
ACMASK            = 0000401C
ATANDLEN1         = 00000003
ATANDLEN2         = 00000003
ATANDTAB1           000001A4 R      01
ATANDTAB2           000001B0 R      01
ATAND_TABLE         000000FC R      01
ATANLEN1          = 00000003
ATANLEN2          = 00000003
ATANTAB1            000000E4 R      01
ATANTAB2            000000F0 R      01
ATAN_TABLE          0000003C R      01
INF                 0000021C R      01
INFD                000003A2 R      01
LARGE_ARG           00000291 R      01
LARGE_ARGD          0000041F R      01
LF_180            = 00004434
LF_90             = 000043B4
LF_M90            = 0000C3B4
LF_MPI_OVER_2     = 0FDBC0C9
LF_PI             = 0FDB4149
LF_PI_OVER_2      = 0FDB40C9
LF_PI_OVER_2_HI   = 0FDB40C9
LF_PI_OVER_2_LO   = BD2EB43B
LONG              = 00000004
MTH$$AB_ATAN        00000004 RG     01
MTH$$AB_ATAN_V      00000000 RG     01
MTH$$JACKET_AND     ********    X   01
MTH$$SIGNAL         ********    X   01
MTH$ATAN            000001BC RG     01
MTH$ATAN2           000001CC RG     01
MTH$ATAND           0000034A RG     01
MTH$ATAND2          0000035A RG     01
MTH$ATAND_R4        000003CB RG     01
MTH$ATAND_R4D       000003C8 R      01
MTH$ATAN_R4         0000023D RG     01
MTH$ATAN_R4D        0000023A R      01
MTH$K_INVARGMAT     ********    X   01
NEG_ARG             000002B5 R      01
NEG_ARGD            0000043C R      01
N_LARGE_ARG         00000303 R      01
N_LARGE_ARGD        0000048A R      01
PI_OV_180_M_64      000001B8 R      01
SMALL               0000028E R      01
SMALLD              0000041C R      01
SMALL_ARG           00000327 R      01
SMALL_ARGD          000004A7 R      01
X                 = 00000004
Y                 = 00000008
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
_MTH$CODE	000004D5 (1237.)	01 (1.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.10	00:00:00.31
Command processing	110	00:00:00.54	00:00:02.86
Pass 1	109	00:00:02.56	00:00:09.46
Symbol table sort	0	00:00:00.02	00:00:00.02
Pass 2	220	00:00:02.00	00:00:09.39
Symbol table output	6	00:00:00.06	00:00:00.22
Psect synopsis output	3	00:00:00.02	00:00:00.06
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	479	00:00:05.30	00:00:22.33

The working set limit was 1050 pages.
15739 bytes (31 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 52 non-local and 8 local symbols.
1031 source lines were read in Pass 1, producing 21 object records in Pass 2.
1 page of virtual memory was used to define 1 macro.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYSLIB]STARLET.MLB;2	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

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

MTH4OVPI
LIS

MTHABS
LIS

MTHAINT
LIS

MTHAMOD
LIS

MTHERR
SDL

MTHASIN
LIS

MTHCDABS
LIS

MTHATAN
LIS

MTHATANH
LIS

MTHCDLOG
LIS

MTHJACKET
MAR

MTHBITOPS
LIS

MTHALOG
LIS

MTHDEF
FOR

MTHANINT
LIS

MTHCABS
LIS

MTHACOS
LIS

MTHCDEXP
LIS