F 13

```
MMM         MMM  TTTTTTTTTTTTTTT  HHH         HHH  RRRRRRRRRRRR     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        HHHHHHHHHHHHHHH  RRRRRRRRRRRR           TTT        LLL
MMM         MMM        TTT        HHHHHHHHHHHHHHH  RRRRRRRRRRRR           TTT        LLL
MMM         MMM        TTT        HHHHHHHHHHHHHHH  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        LLLLLLLLLLLLLLL
MMM         MMM        TTT        HHH         HHH  RRR         RRR        TTT        LLLLLLLLLLLLLLL
MMM         MMM        TTT        HHH         HHH  RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

FILEID**MTHATAN

```
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


```
(2)    73     HISTORY : Detailed Current Edit History
(3)    105    DECLARATIONS    : Declarative Part of Module
(6)    342    MTH$ATAN - Standard Single Precision Floating Arc Tangent
(7)    412    MTH$ATAN2 - Standard floating Arctangent With 2 Arguments
(8)    506    MTH$ATAN_R4 - Special ATAN routine
(9)    657    MTH$ATAND - Standard Single Precision Floating Arc Tangent
(10)   727    MTH$ATAND2 - Standard floating Arctangent With 2 Arguments
(11)   822    MTH$ATAND_R4 - Special ATAND routine
```

```
0000          .TITLE  MTH$ATAN         ; Floating Point Arc Tangent Functions
0000                                   ; (ATAN,ATAN2,ATAND,ATAND2)
0000          .IDENT /2-005/           ; File: MTHATAN.MAR  EDIT:  RNH2005
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000    ;
0000    ; FACILITY: MATH LIBRARY
0000    ;++
0000    ; ABSTRACT:
0000    ;
0000    ; MTH$ATAN is a function which returns the floating point arc tangent
0000    ; value (in radians) of its single precision floating point argument.
0000    ; MTH$ATAN2  is two arguments floating point arctangent. The call is
0000    ; standard call-by-reference.
0000    ; MTH$ATAN_R4  is  a  special  routine  which is the same as MTH$ATAN
0000    ; except a faster non-standard JSB call is  used with the argument in
0000    ; R0 and no registers are saved.
0000    ;
0000    ; MTH$ATAND is a function which returns the floating point arc tangent
0000    ; value  (in degrees) of its single precision floating point argument.
0000    ; MTH$ATAND2 is two arguments floating point arctangent.  The  call is
0000    ; standard call-by-reference.
0000    ; MTH$ATAND_R4  is  a  special  routine which is the same as MTH$ATAND
0000    ; except a faster non-standard JSB call is  used  with the argument in
0000    ; R0 and no registers are saved.
0000    ;
0000    ;--
0000    ;
0000    ; VERSION: 01
0000    ;
0000    ; HISTORY:
0000    ; AUTHOR:
0000    ;       Peter Yuo, 15-Oct-76: Version 01
0000    ;
0000    ; MODIFIED BY:
```

```
0000     ;
0000     ; 01-1  Peter Yuo, 22-May-77
0000     ;
0000     ;--
0000     ;
0000     ; VERSION: 02
0000     ;
0000     ; HISTORY:
0000     ; AUTHOR:
0000     ;       Bob Hanek, 04-Jun-81: Version 02
0000     ;
0000     ; MODIFIED BY:
0000     ;
0000     ;
```

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

```
                               0000          .SBTTL  DECLARATIONS     ; Declarative Part of Module
                               0000
                               0000   ;
                               0000   ; INCLUDE FILES:           MTHJACKET.MAR
                               0000   ;
                               0000   ;
                               0000   ; EQUATED SYMBOLS:
                               0000   ;
                     0000401C  0000          ACMASK = ^M<IV, R2, R3, R4>       ; .ENTRY register mask, int ovf enabled
                     0FDB40C9  0000          LF_PI_OVER_2_HI = ^X0FDB40C9      ; High 24 bits of pi/2
                     BD2EB43B  0000          LF_PI_OVER_2_LO = ^XBD2EB43B      ; Low 24 bits of pi/2
                     0FDBC0C9  0000          LF_MPI_OVER_2   = ^X0FDBC0C9      ; -pi/2
                     0FDB40C9  0000          LF_PI_OVER_2    = ^X0FDB40C9      ; pi/2
                     00004434  0000          LF_180          = ^X00004434      ; 90
                     0000C3B4  0000          LF_M90          = ^X0000C3B4      ; -90
                     0FDB4149  0000          LF_PI           = ^X0FDB4149      ; pi
                     000043B4  0000          LF_90           = ^X000043B4      ; 90
                               0000
                               0000   ;
                               0000   ; MACROS:         none
                               0000   ;
                               0000   ; PSECT DECLARATIONS:
                               0000
                           00000000          .PSECT  _MTH$CODE         PIC,SHR,LONG,EXE,NOWRT
                               0000                                    ; program section for math routines
                               0000   ;
                               0000   ; OWN STORAGE:  none
                               0000   ;
                               0000   ; EXTERNALS:
                               0000   ;
                               0000          .EXTRN  MTH$$SIGNAL               ; Signal a severe error
                               0000          .EXTRN  MTH$K_INVARGMAT           ; Invalid argument to math library
                               0000          .DSABL  GBL                       ; No other externals allowed
                               0000   ;
                               0000   ; CONSTANTS:
                               0000
                               0000   ;
                               0000   ;   The MTH$$AB_ATAN table is a table of byte entries used to obtain an index
                               0000   ;   into the ATAN_TABLE.  MTH$$AB_ATAN is indexed using the low order bits of
                               0000   ;   the  exponent  field  and the high order bits of the fraction field.  The
                               0000   ;   MTH$$AB_ATAN table is independent of the data type and is used by all  of
                               0000   ;   the arctangent routines.
                               0000   ;
                               0000
                               0000  MTH$$AB_ATAN_V::                  ; Simulated vector entry for G and
                     00000004' 0000          .LONG   MTH$$AB_ATAN-.    ;   H routines
                               0004  MTH$$AB_ATAN::
09 06 06 03 03 00 00 00        0004          .BYTE   ^X00, ^X00, ^X00, ^X03, ^X03, ^X06, ^X06, ^X09
12 0F 0F 0C 0C 0C 09 09        000C          .BYTE   ^X09, ^X09, ^X0C, ^X0C, ^X0C, ^X0F, ^X0F, ^X12
18 18 18 15 15 15 12 12        0014          .BYTE   ^X12, ^X12, ^X15, ^X15, ^X15, ^X18, ^X18, ^X18
21 1E 1E 1E 1B 1B 1B 1B        001C          .BYTE   ^X1B, ^X1B, ^X1B, ^X1B, ^X1E, ^X1E, ^X1E, ^X21
24 24 24 24 21 21 21 21        0024          .BYTE   ^X21, ^X21, ^X21, ^X21, ^X24, ^X24, ^X24, ^X24
27 27 27 27 27 27 24 24        002C          .BYTE   ^X24, ^X24, ^X27, ^X27, ^X27, ^X27, ^X27, ^X27
   27 27 27 27 27 27 27        0034          .BYTE   ^X27, ^X27, ^X27, ^X27, ^X27, ^X27, ^X27
                               003B
                               003B
```

```
                003B     ;
                003B     ;
                003B     ;      *************************  Constants for ATAN  *****************************
                003B     ;
                003B     ;
                003B     ;      Each entry of the ATAN_TABLE contains  the the values of XHI, ATAN_XHI_LO
                003B     ;      and ATAN_XHI_HI respectively.  The table is indexed by a pointer obtained
                003B     ;      from  the MTH$$AB_ATAN table.  NOTE: For performance reasons it is import-
                003B     ;      ant to have the ATAN_TABLE longword aligned.
                003B     ;
                003B     ;
                003B                .ALIGN  LONG
                003C
                003C     ATAN_TABLE:
                003C     ; Entry  0
      F87E3ED7  003C                .LONG   ^XF87E3ED7          ; 0.105454430E+00
      0A99B21B  0040                .LONG   ^X0A99B21B          ; -.225614927E-08
      2CE73ED7  0044                .LONG   ^X2CE73ED7          ; 0.105066113E+00
                0048     ; Entry  1
      FB703F03  0048                .LONG   ^XFB703F03          ; 0.128888845E+00
      F372325E  004C                .LONG   ^XF372325E          ; 0.324436344E-08
      422F3F03  0050                .LONG   ^X422F3F03          ; 0.128182158E+00
                0054     ; Entry  2
      F63E3F1F  0054                .LONG   ^XF63E3F1F          ; 0.156212777E+00
      B326B253  0058                .LONG   ^XB326B253          ; -.308063752E-08
      ADF13F1E  005C                .LONG   ^XADF13F1E          ; 0.154960409E+00
                0060     ; Entry  3
      E4EC3F47  0060                .LONG   ^XE4EC3F47          ; 0.195209205E+00
      4A61B2AA  0064                .LONG   ^X4A61B2AA          ; -.495610708E-08
      69623F45  0068                .LONG   ^X69623F45          ; 0.192784816E+00
                006C     ; Entry  4
      C3D13F7F  006C                .LONG   ^XC3D13F7F          ; 0.249770418E+00
      678D3298  0070                .LONG   ^X678D3298          ; 0.443555459E-08
      A30A3F7A  0074                .LONG   ^XA30A3F7A          ; 0.244762570E+00
                0078     ; Entry  5
      DB973F9F  0078                .LONG   ^XDB973F9F          ; 0.312222213E+00
      315D3323  007C                .LONG   ^X315D3323          ; 0.949907264E-08
      F28D3F9A  0080                .LONG   ^XF28D3F9A          ; 0.302631766E+00
                0084     ; Entry  6
      9E8E3FC7  0084                .LONG   ^X9E8E3FC7          ; 0.389881551E+00
      F1A6335E  0088                .LONG   ^XF1A6335E          ; 0.129770452E-07
      56713FBE  008C                .LONG   ^X56713FBE          ; 0.371753246E+00
                0090     ; Entry  7
      33B63FFF  0090                .LONG   ^X33B63FFF          ; 0.498441398E+00
      C9A0B31E  0094                .LONG   ^XC9A0B31E          ; -.924265464E-08
      BFB03FEC  0098                .LONG   ^XBFB03FEC          ; 0.462399960E+00
                009C     ; Entry  8
      F8EB4026  009C                .LONG   ^XF8EB4026          ; 0.652235687E+00
      4695B38C  00A0                .LONG   ^X4695B38C          ; -.163302420E-07
      F4394013  00A4                .LONG   ^XF4394013          ; 0.577945292E+00
                00A8     ; Entry  9
      0712405E  00A8                .LONG   ^X712405E           ; 0.867295384E+00
      6A34B399  00AC                .LONG   ^X6A34B399          ; -.178598398E-07
      E62C4036  00B0                .LONG   ^XE62C4036          ; 0.714449644E+00
                00B4     ; Entry 10
      CBD84095  00B4                .LONG   ^XCBD84095          ; 0.117028332E+01
      99953108  00B8                .LONG   ^X99953108          ; 0.496947650E-09
```

```
1B62405D  00BC          .LONG   ^X1B62405D      ; 0.863699079E+00
          00C0  ; Entry 11
8DEB40D2  00C0          .LONG   ^X8DEB40D2      ; 0.164495599E+01
DBF9B3F3  00C4          .LONG   ^XDBF9B3F3      ; -.283889552E-07
25414083  00C8          .LONG   ^X25414083      ; 0.102457440E+01
          00CC  ; Entry 12
88054124  00CC          .LONG   ^X88054124      ; 0.257080197E+01
888B3433  00D0          .LONG   ^X888B3433      ; 0.418008703E-07
93CA4099  00D4          .LONG   ^X93CA4099      ; 0.119982266E+01
          00D8  ; Entry 13
7D0E41AB  00D8          .LONG   ^X7D0E41AB      ; 0.535901546E+01
DF7F33C8  00DC          .LONG   ^XDF7F33C8      ; 0.233846986E-07
72D240B1  00E0          .LONG   ^X72D240B1      ; 0.138631654E+01
          00E4
          00E4  ;
          00E4  ; Tables to be used in POLYF for computing ATAN.  ATANTAB1 is obtained
          00E4  ; from Hart et. al. (No. 4900).  ATANTAB2 is the same as ATANTAB1 except
          00E4  ; that C0 is set to 0
          00E4
          00E4  ATANTAB1:
B0AA3F4A  00E4          .LONG   ^XB0AA3F4A      ; C2 =  .19793954
A9B0BFAA  00E8          .LONG   ^XA9B0BFAA      ; C1 = -.33332586
00004080  00EC          .LONG   ^X00004080      ; C0 = 1.00000000
00000003  00F0  ATANLEN1 = .- ATANTAB1/4
          00F0
          00F0  ATANTAB2:
B0AA3F4A  00F0          .LONG   ^XB0AA3F4A      ; C2 =  .19793954
A9B0BFAA  00F4          .LONG   ^XA9B0BFAA      ; C1 = -.33332586
00000000  00F8          .LONG   ^X00000000      ; C0 =  .00000000
00000003  00FC  ATANLEN2 = .- ATANTAB2/4
          00FC
          00FC
```

```
                00FC   ;
                00FC   ;
                00FC   ;       **************************  Constants for ATAND  ***************************
                00FC   ;
                00FC   ;
                00FC   ;       Each  entry of the ATAND_TABLE contains the the values of XHI, ATAND_XHI_LO
                00FC   ;       and ATAND_XHI_HI respectively.  The table is indexed by a pointer  obtained
                00FC   ;       from the MTH$$AB_ATAN table.  NOTE: For performance reasons it is important
                00FC   ;       to have the ATAND_TABLE longword aligned.
                00FC   ;
                00FC
                00FC   ATAND_TABLE:
                00FC   ; Entry  0
      F87E3ED7  00FC                 .LONG      ^XF87E3ED7          ; 0.105454430E+00
      B1EF354F  0100                 .LONG      ^XB1EF354F          ; 0.193431092E-06
      A29141C0  0104                 .LONG      ^XA29141C0          ; 0.601984453E+01
                0108   ; Entry  1
      FB703F03  0108                 .LONG      ^XFB703F03          ; 0.128888845E+00
      FE1BB4C3  010C                 .LONG      ^XFE1BB4C3          ; -.9126616662E-07
      047B41EB  0110                 .LONG      ^X047B41EB          ; 0.734429693E+01
                0114   ; Entry  2
      F63E3F1F  0114                 .LONG      ^XF63E3F1F          ; 0.156212777E+00
      AF4932FD  0118                 .LONG      ^XAF4932FD          ; 0.738319672E-08
      0EA7420E  011C                 .LONG      ^X0EA7420E          ; 0.887857723E+01
                0120   ; Entry  3
      E4EC3F47  0120                 .LONG      ^XE4EC3F47          ; 0.195209205E+00
      3623B5A7  0124                 .LONG      ^X3623B5A7          ; -.311455636E-06
      BB6B4230  0128                 .LONG      ^XBB6B4230          ; 0.110457563E+02
                012C   ; Entry  4
      C3D13F7F  012C                 .LONG      ^XC3D13F7F          ; 0.249770418E+00
      4257B5C1  0130                 .LONG      ^X4257B5C1          ; -.359973200E-06
      61BE4260  0134                 .LONG      ^X61BE4260          ; 0.140238628E+02
                0138   ; Entry  5
      DB973F9F  0138                 .LONG      ^XDB973F9F          ; 0.312222213E+00
      300C351B  013C                 .LONG      ^X300C351B          ; 0.144529793E-06
      B758428A  0140                 .LONG      ^XB758428A          ; 0.173395233E+02
                0144   ; Entry  6
      9E8E3FC7  0144                 .LONG      ^X9E8E3FC7          ; 0.389881551E+00
      CFDD35A6  0148                 .LONG      ^XCFDD35A6          ; 0.310711499E-06
      662E42AA  014C                 .LONG      ^X662E42AA          ; 0.212998924E+02
                0150   ; Entry  7
      33B63FFF  0150                 .LONG      ^X33B63FFF          ; 0.498441398E+00
      F7DAB674  0154                 .LONG      ^XF7DAB674          ; -.912577548E-06
      F2D342D3  0158                 .LONG      ^XF2D342D3          ; 0.264935665E+02
                015C   ; Entry  8
      F8EB4026  015C                 .LONG      ^XF8EB4026          ; 0.652235687E+00
      ADBDB6DF  0160                 .LONG      ^XADBDB6DF          ; -.166653592E-05
      748F4304  0164                 .LONG      ^X748F4304          ; 0.331138268E+02
                0168   ; Entry  9
      0712405E  0168                 .LONG      ^X0712405E          ; 0.867295384E+00
      BD24359B  016C                 .LONG      ^XBD24359B          ; 0.290086177E-06
      BD634323  0170                 .LONG      ^XBD634323          ; 0.409349480E+02
                0174   ; Entry 10
      CBD84095  0174                 .LONG      ^XCBD84095          ; 0.117028332E+01
      B637B668  0178                 .LONG      ^XB637B668          ; -.866918924E-06
      F1FC4345  017C                 .LONG      ^XF1FC4345          ; 0.494863129E+02
                0180   ; Entry 11
```

```
8DEB40D2  0180          .LONG   ^X8DEB40D2          ; 0.164495599E+01
9881B6CC  0184          .LONG   ^X9881B6CC          ; -.152435689E-05
D0AE436A  0188          .LONG   ^XD0AE436A          ; 0.587037888E+02
          018C   ; Entry 12
88054124  018C          .LONG   ^X88054124          ; 0.257080197E+01
41353761  0190          .LONG   ^X41353761          ; 0.335655682E-05
7D534389  0194          .LONG   ^X7D534389          ; 0.687447739E+02
          0198   ; Entry 13
7D0E41AB  0198          .LONG   ^X7D0E41AB          ; 0.535901546E+01
FECD377E  019C          .LONG   ^XFECD377E          ; 0.379972630E-05
DC34439E  01A0          .LONG   ^XDC34439E          ; 0.794300842E+02
          01A4
          01A4   ;
          01A4   ; Tables to be used in POLYF for computing ATAND.  ATANDTAB1 is obtained
          01A4   ; by multiplying the coefficients given in Hart et. al. (No. 4900) by
          01A4   ; 180/pi.  ATANDTAB2 is the same as ATANDTAB1 except that C0 is set to
          01A4   ; 180/pi - 64 instead of 180/pi.
          01A4
          01A4   ATANDTAB1:
75264235  01A4          .LONG   ^X75264235          ; C2 =  11.341100693
C90BC298  01A8          .LONG   ^XC90BC298          ; C1 = -19.098165512
2EE14365  01AC          .LONG   ^X2EE14365          ; C0 =  57.295780182
00000003  01B0   ATANDLEN1 = .- ATANDTAB1/4
          01B0
          01B0   ATANDTAB2:
75264235  01B0          .LONG   ^X75264235          ; C2 =  11.341100693
C90BC298  01B4          .LONG   ^XC90BC298          ; C1 = -19.098165512
          01B8   PI_OV_180_M_64:
88F9C1D6  01B8          .LONG   ^X88F9C1D6          ; C0 =  -6.704219818
00000003  01BC   ATANDLEN2 = .- ATANDTAB2/4
          01BC
          01BC
```

```
          01BC          .SBTTL  MTH$ATAN - Standard Single Precision Floating Arc Tangent
          01BC
          01BC
          01BC   ;++
          01BC   ; FUNCTIONAL DESCRIPTION:
          01BC   ;
          01BC   ; ATAN - single precision floating point function
          01BC   ;
          01BC   ; ATAN is computed using the following steps:
          01BC   ;
          01BC   ;    1. If X > 11 then
          01BC   ;        a. Let W = 1/X.
          01BC   ;        b. Compute ATAN(W) = W*P(W**2), where P is a polynomial of
          01BC   ;           degree 2.
          01BC   ;        c. Set ATAN(X) = pi/2 - ATAN(W)
          01BC   ;    2. If 3/32 =< X =< 11 then
          01BC   ;        a. Obtain XHI by table look-up.
          01BC   ;        b. Compute Z = (X - XHI)/(1 + X*XHI).
          01BC   ;        c. Compute ATAN(Z) = Z*P(Z**2), where P is a polynomial of
          01BC   ;           degree 2.
          01BC   ;        d. Obtain ATAN(XHI) by table look-up.  ATAN(XHI) will have
          01BC   ;           two parts - the high order bits, ATAN_XHI_HI, and the low
          01BC   ;           order bits, ATAN_XHI_LO.
          01BC   ;        e. Compute ATAN(X) = ATAN_XHI_HI + (ATAN_XHI_LO + ATAN(Z)).
          01BC   ;    3. If 0 =< X < 3/32 then
          01BC   ;        a. Compute ATAN(X) = X + X*Q(X**2), where Q is a polynomial
          01BC   ;           of degree 2.
          01BC   ;    4. If X < 0 then
          01BC   ;        a. Compute Y = ATAN(|X|) using steps 1 to 3.
          01BC   ;        b. Set ATAN(X) = -Y.
          01BC   ;
          01BC   ;
          01BC   ; CALLING SEQUENCE:
          01BC   ;
          01BC   ;       Arctangent.wf.v = MTH$ATAN(x.rf.r)
          01BC   ;
          01BC   ; INPUT PARAMETERS:
          01BC
00000004  01BC          LONG = 4                        ; define longword multiplier
00000004  01BC          x = 1 * LONG                    ; x is an angle in radians
          01BC   ;
          01BC   ; IMPLICIT INPUTS:      none
          01BC   ;
          01BC   ; OUTPUT PARAMETERS:
          01BC   ;
          01BC   ;       VALUE:  floating arctangent angle of the argument
          01BC   ;
          01BC   ; IMPLICIT OUTPUTS:     none
          01BC   ;
          01BC   ; SIDE EFFECTS:
          01BC   ;
          01BC   ; Signals:      none
          01BC   ;
          01BC   ; NOTE: This procedure disables floating point underflow and integer
          01BC   ; overflow, causes no floating overflow or other arithmetic traps, and
          01BC   ; preserves enables across the call.
          01BC   ;
```

```
                         01BC   ;---
                         01BC
                         01BC
                   401C  01BC         .ENTRY  MTH$ATAN, ACMASK            ; standard call-by-reference entry
                         01BE                                             ; disable DV (and FU), enable IV
                         01BE         MTH$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                                             ; in case of an error in special JSB
                         01C5                                             ; routine
        50   04 BC   50  01C5         MOVF    @x(AP), R0                  ; R0 = arg
                72   10  01C9         BSBB    MTH$ATAN_R4                 ; call special ATAN rountine
                     04  01CB         RET                                 ; return - result in R0
                         01CC
```

```
                              01CC        .SBTTL  MTH$ATAN2 - Standard floating Arctangent With 2 Arguments
                              01CC  ;++
                              01CC  ; FUNCTIONAL DESCRIPTION:
                              01CC  ;
                              01CC  ; ATAN2  - single precision floating point function
                              01CC  ;
                              01CC  ; ATAN2(X,Y) is computed as following:
                              01CC  ;
                              01CC  ;       If Y = 0 or X/Y > 2**25, ATAN2(X,Y) = PI/2 * (sign X)
                              01CC  ;       If Y > 0 and X/Y =< 2**25, ATAN2(X,Y) = ATAN(X/Y)
                              01CC  ;       If Y < 0 and X/Y =< 2**25, ATAN2(X,Y) = PI * (sign X) + ATAN(X/Y)
                              01CC  ;
                              01CC  ;
                              01CC  ; CALLING SEQUENCE:
                              01CC  ;
                              01CC  ;       Arctangent2.wf.v = MTH$ATAN2(x.rf.r, y.rf.r)
                              01CC  ;
                              01CC  ; INPUT PARAMETERS:
                              01CC  ;
                    00000004  01CC        x = 1 * LONG                          ; x is the first argument
                    00000008  01CC        y = 2 * LONG                          ; y is the second argument
                              01CC  ;
                              01CC  ; SIDE EFFECTS: See description of MTH$ATAN
                              01CC  ;
                              01CC  ;--
                              01CC
                              01CC
                        401C  01CC        .ENTRY  MTH$ATAN2 ,ACMASK             ; standard call-by-reference entry
                              01CE                                              ; disable DV (and FU), enable IV
                              01CE        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                                              ; in case of an error in special JSB routine
          50    04 BC     50  01D5        MOVF    @x(AP), R0                    ; R0 = arg1
          51    08 BC     50  01D9        MOVF    @y(AP), R1                    ; R1 = arg2
                              01DD  ;
                              01DD  ; Test if Y = 0 or X/Y > 2**25
                              01DD  ;
                  35      13  01DD        BEQL    INF                           ; branch to INF if Y = 0
   52    50    807F 8F    AB  01DF        BICW3   #^X807F, R0, R2               ; R2 = exponent(X)
   53    51    807F 8F    AB  01E5        BICW3   #^X807F, R1, R3               ; R3 = exponent(Y)
                  52  53  A2  01EB        SUBW    R3, R2                        ; R2 = exponent(X) - exponent(Y)
         0D00 8F      52  B1  01EE        CMPW    R2, #26*128                   ; compare R2 with 26
                  1F      14  01F3        BGTR    INF                           ; if X/Y > 2**25, branch to INF
                              01F5  ;
                              01F5  ; Test if Y > 0 or Y < 0
                              01F5  ;
                  51      B5  01F5        TSTW    R1                            ; test the sign of Y
                  18      14  01F7        BGTR    A2PLUS                        ; branch to A2PLUS if Y > 0
                  50      B5  01F9        TSTW    R0                            ; test the sign of X
                  0A      18  01FB        BGEQ    A1PLUS                        ; branch to A1PLUS if X >= 0
                              01FD  ;
                              01FD  ; Y < 0 and X < 0 and X/Y =< 2**25
                              01FD  ;
```

```
                     3B   10  01FD          BSBB    MTH$ATAN_R4D             ; R0 = ATAN(X/Y)
 50     0FDB4149 8F   42  01FF          SUBF    #LF_PI, R0               ; R0 = -PI + ATAN(X/Y)
                      04  0206          RET                              ; return
                          0207   ;
                          0207   ; Y < 0 and X > 0 and X/Y =< 2**25
                          0207   ;
                          0207   A1PLUS:
                     31   10  0207          BSBB    MTH$ATAN_R4D             ; R0 = ATAN(X/Y)
 50     0FDB4149 8F   40  0209          ADDF    #LF_PI, R0               ; R0 = PI + ATAN(X/Y)
                      04  0210          RET                              ; return
                          0211   ;
                          0211   ; Y > 0 and X/Y =< 2**25
                          0211   ;
                          0211   A2PLUS:
                     27   10  0211          BSBB    MTH$ATAN_R4D             ; R0 = ATAN(X/Y)
                          04  0213          RET                              ; return
                          0214   ;
                          0214   ; Y = 0 or X/Y > 2**25
                          0214   ;
                          0214   INF:
                     50   B5  0214          TSTW    R0                       ; test the sign of X
                     0A   14  0216          BGTR    1$                       ; branch if X > 0
                     10   13  0218          BEQL    2$                       ; branch if X = 0
 50     0FDBC0C9 8F   50  021A          MOVF    #LF_MPI_OVER_2, R0       ; R0 = ATAN(X/Y) = -PI/2
                      04  0221          RET                              ; return
                          0222
 50     0FDB40C9 8F   50  0222   1$:    MOVF    #LF_PI_OVER_2, R0        ; R0 = ATAN(X/Y) = PI/2
                      04  0229          RET                              ; return
                          022A
                          022A   ;+
                          022A   ; Here if X = 0 and Y = 0.  Signal INVALID ARG TO MATH LIBRARY
                          022A   ; as a SEVERE error.
                          022A   ;-
                          022A
     50    01    0F   79  022A   2$:    ASHQ    #15, #1, R0              ; R0/R1 = reserved operand which
                          022E                                           ; is copied to CHF$L_MCH_SAVR0/R1
                          022E                                           ; so a handler can fixup if wants
                          022E                                           ; to continue
      7E    00'8F     9A  022E          MOVZBL  #MTH$K_INVARGMAT, -(SP)  ; code for INVALID ARG TO MATH LIBRARY
 00000000'GF    01    FB  0232          CALLS   #1, G^MTH$$SIGNAL        ; signal SEVERE error
                      04  0239          RET                              ; return if handler continues
```

```
                          023A          .SBTTL  MTH$ATAN_R4 - Special ATAN routine
                          023A
                          023A  ; Special ATAN - used by the standard routine, and directly.
                          023A  ;
                          023A  ; CALLING SEQUENCES:
                          023A  ;       save anything needed in R0:R4
                          023A  ;       MOVF    ..., R0                       ; input in R0
                          023A  ;       JSB     MTH$ATAN_R4
                          023A  ;       return with result in R0
                          023A  ;
                          023A  ; Note:  This  routine  is written to avoid causing any integer overflows,
                          023A  ;    floating overflows, or floating underflows or divide by 0 conditions,
                          023A  ;    whether enabled or not.
                          023A  ;
                          023A  ; REGISTERS USED:
                          023A  ;       R0 - Floating argument then result
                          023A  ;       R0:R3 - POLYF
                          023A  ;       R4 - Pointer into ATAN_TABLE
                          023A
                          023A
                          023A
                          023A  MTH$ATAN_R4D:
             50   51   46 023A          DIVF    R1, R0                        ; for our own use only!
                          023D  MTH$ATAN_R4::                                 ; Special ATAN routine
                  50   53 023D          TSTF    R0                            ; R4 = X = argument
                  74   19 023F          BLSS    NEG_ARG                       ; Branch to negative argument logic
                          0241  ;
                          0241  ; Argument is positive
                          0241  ;
  54   50   3EC0 8F    A3 0241          SUBW3   #^X3EC0, R0, R4               ; Argument is less than 3/32,
                  45   19 0247          BLSS    SMALL                         ;  branch to small argument logic
       54   036F 8F    B1 0249          CMPW    #^X036F, R4                   ; Argument is greater that 11,
                  41   19 024E          BLSS    LARGE_ARG                     ;  branch to large argument logic
                          0250  ;
                          0250  ; Logic for positive medium sized arguments.  Get pointer into ATAN_TABLE.
                          0250  ;
   54   54    FC 8F    9C 0250          ROTL    #-4, R4, R4                   ; R4 = index into MTH$$AB_ATAN table
   54   FFFFFF00 8F    CA 0255          BICL    #-256, R4                     ; zero high order bits of index
    54   FDA3 CF44     90 025C          MOVB    MTH$$AB_ATAN[R4], R4          ; R4 = offset into ATAN_TABLE
    54   FDD5 CF44     DE 0262          MOVAL   ATAN_TABLE[R4], R4            ; R4 = pointer to XHI
                          0268  ;
                          0268  ; Compute Z
                          0268  ;
             51   84   D0 0268          MOVL    (R4)+, R1                     ; R1 = XHI
        52   50   51   45 026B          MULF3   R1, R0, R2                    ; R2 = X*XHI
             52   08   40 026F          ADDF    #1, R2                        ; R2 = 1 + X*XHI
             50   51   42 0272          SUBF    R1, R0                        ; R0 = X - XHI
             50   52   46 0275          DIVF    R2, R0                        ; R0 = Z = (X - XHI)/(1 + X*XHI)
                          0278  ;
                          0278  ; Evaluate Z*P(Z**2)
                          0278  ;
             7E   50   D0 0278          MOVL    R0, -(SP)                     ; Push Z onto the stack
             50   50   44 027B          MULF    R0, R0                        ; R0 = Z**2
   FE60 CF   02   50   55 027E          POLYF   R0, #ATANLEN1-1, ATANTAB1
                          0284                                                ; R0 = P(Z**2)
             50   8E   44 0284          MULF    (SP)+, R0                     ; R0 = ATAN(Z) = Z*P(Z**2)
             50   84   40 0287          ADDF    (R4)+, R0                     ; R0 = ATAN_XHI_LO + ATAN(Z)
```

```
            50     64     40  028A          ADDF    (R4), R0                ; R0 = ATAN(X) = ATAN_XHI_HI +
                              028D                                          ;      (ATAN_XHI_LO + ATAN(Z))
                          05  028D          RSB                             ; Return
                              028E
                              028E
                0096      31  028E SMALL:   BRW     SMALL_ARG               ; Dummy label used to avoid adding
                              0291                                          ;   an extra insrtuction in the
                              0291                                          ;   medium argument logic
                              0291 ;
                              0291 ; Large positive argument logic.
                              0291 ;
                              0291
                              0291 LARGE_ARG:
   54   0000C080 8F    50     47  0291          DIVF3   R0, #-1, R4             ; R4 = -W = -1/X
           50     54     54   45  0299          MULF3   R4, R4, R0              ; R0 = W**2
      FE41 CF     02     50   55  029D          POLYF   R0, #ATANLEN1-1, ATANTAB1
                              02A3                                          ; R0 = P(W**2)
            50     54     44  02A3          MULF    R4, R0                  ; R0 = ATAN(W) = -W*P(W**2)
     50   BD2EB43B 8F     40  02A6          ADDF    #LF_PI_OVER_2_LO, R0    ;
     50   0FDB40C9 8F     40  02AD          ADDF    #LF_PI_OVER_2_HI, R0    ; R0 = ATAN(X) = PI/2 - ATAN(W)
                          05  02B4          RSB                             ; Return
                              02B5
                              02B5 ;
                              02B5 ; Logic for negative arguments
                              02B5 ;
                              02B5
                              02B5 NEG_ARG:
   54     50   BEC0 8F    A3  02B5          SUBW3   #^XBEC0, R0, R4         ; Argument is less than 3/32,
                     6A   19  02BB          BLSS    SMALL_ARG               ;  branch to small argument logic
          54   036F 8F    B1  02BD          CMPW    #^X036F, R4             ; Argument is greater than 11,
                     3F   19  02C2          BLSS    N_LARGE_ARG             ;  branch to large argument logic
                              02C4 ;
                              02C4 ; Logic for negative medium sized arguments.  Get index into ATAN_TABLE.
                              02C4 ;
   54     54     FC 8F    9C  02C4          ROTL    #-4, R4, R4             ; R4 = index into MTH$$AB_ATAN table
   54   FFFFFF00 8F       CA  02C9          BICL    #-256, R4               ; clear high order (unused) bits of ind
    54      FD2F CF44     90  02D0          MOVB    MTH$$AB_ATAN[R4], R4    ; R4 = offset into ATAN_TABLE
    54      FD61 CF44     DE  02D6          MOVAL   ATAN_TABLE[R4], R4      ; R4 = pointer to XHI
                              02DC ;
                              02DC ; Compute Z
                              02DC ;
            51     84     D0  02DC          MOVL    (R4)+, R1               ; R1 = XHI
     52     50     51     45  02DF          MULF3   R1, R0, R2              ; R2 = X*XHI
     52     08     52     43  02E3          SUBF3   R2, #1, R2              ; R2 = 1 - X*XHI = 1 + X*(-XHI)
            50     51     40  02E7          ADDF    R1, R0                  ; R0 = X + XHI = X - (-XHI)
            50     52     46  02EA          DIVF    R2, R0                  ; R0 = Z
                              02ED ;
                              02ED ; Evaluate Z*P(Z**2)
                              02ED ;
            7E     50     D0  02ED          MOVL    R0, -(SP)               ; Push Z onto the stack
            50     50     44  02F0          MULF    R0, R0                  ; R0 = Z**2
      FDEB CF     02     50   55  02F3          POLYF   R0, #ATANLEN1-1, ATANTAB1
                              02F9                                          ; R0 = P(Z**2)
            50     8E     44  02F9          MULF    (SP)+, R0               ; R0 = ATAN(Z) = Z*P(Z**2)
            50     84     42  02FC          SUBF    (R4)+, R0               ; R0 = ATAN_XHI_LO + ATAN(Z)
            50     64     42  02FF          SUBF    (R4), R0                ; R0 = ATAN(X) = ATAN_XHI_HI +
                              0302                                          ;      (ATAN_XHI_LO + ATAN(Z))
```

```
                      05 0302          RSB                              ; Return
                         0303  ;
                         0303  ; Logic for large negative  arguments
                         0303  ;
                         0303
                         0303  N_LARGE_ARG:
54  0000C080 8F    50 47 0303          DIVF3   R0, #-1, R4              ; R4 = W = 1/|X|
          50 54    54 45 030B          MULF3   R4, R4, R0               ; R0 = W**2
     FDCF CF 02    50 55 030F          POLYF   R0, #ATANLEN1-1, ATANTAB1
                         0315                                           ; R0 = P(W**2)
             50    54 44 0315          MULF    R4, R0                   ; R0 = ATAN(W) = W*P(W**2)
  50 BD2EB43B 8F   42    0318          SUBF    #LF_PI_OVER_2_LO, R0     ;
  50 0FDB40C9 8F   42    031F          SUBF    #LF_PI_OVER_2_HI, R0     ; R0 = ATAN(X) = ATAN(W) - PI/2
                      05 0326          RSB                              ; Return
                         0327
                         0327  ;
                         0327  ; Small argument logic.
                         0327  ;
                         0327
                         0327  SMALL_ARG:
             54    50 D0 0327          MOVL    R0, R4                   ; R4 = argument = X
     50 8000 8F       AA 032A          BICW    #^X8000, R0              ; R0 = |X|
     50 3A00 8F       B1 032F          CMPW    #^X3A00, R0              ; Compare 2^-13 to |X|
                   04 19 0334          BLSS    1$                       ; Branch to polynomial evaluation
             50    54 D0 0336          MOVL    R4, R0                   ; No POLY needed.  Answer = X
                      05 0339          RSB                              ;
                         033A
             50    50 44 033A  1$:     MULF    R0, R0                   ; R0 = X**2
     FDAD CF 02    50 55 033D          POLYF   R0, #ATANLEN2-1, ATANTAB2
                         0343                                           ; R0 = Q(X**2)
             50    54 44 0343          MULF    R4, R0                   ; R0 = X*Q(X**2)
             50    54 40 0346          ADDF    R4, R0                   ; R0 = ATAN(X) = X + X*Q(X**2)
                      05 0349          RSB                              ; Return
                         034A
                         034A
                         034A
```

```
                 034A        .SBTTL  MTH$ATAND - Standard Single Precision Floating Arc Tangent
                 034A
                 034A
                 034A   ;++
                 034A   ; FUNCTIONAL DESCRIPTION:
                 034A   ;
                 034A   ; ATAND - single precision floating point function
                 034A   ;
                 034A   ; ATAND is computed using the following steps:
                 034A   ;
                 034A   ;    1. If X > 11 then
                 034A   ;       a. Let W = 1/X.
                 034A   ;       b. Compute ATAND(W) = W*P(W**2), where P is a polynomial of
                 034A   ;          degree 2.
                 034A   ;       c. Set ATAND(X) = pi/2 - ATAND(W)
                 034A   ;    2. If 3/32 =< X =< 11 then
                 034A   ;       a. Obtain XHI by table look-up.
                 034A   ;       b. Compute Z = (X - XHI)/(1 + X*XHI).
                 034A   ;       c. Compute ATAND(Z) = Z*P(Z**2), where P is a polynomial of
                 034A   ;          degree 2.
                 034A   ;       d. Obtain ATAND(XHI) by table look-up.  ATAND(XHI) will have
                 034A   ;          two parts - the high order bits, ATAND_XHI_HI, and the low
                 034A   ;          order bits, ATAND_XHI_LO.
                 034A   ;       e. Compute ATAND(X) = ATAND_XHI_HI + (ATAND_XHI_LO + ATAND(Z)).
                 034A   ;    3. If 0 =< X < 3/32 then
                 034A   ;       a. Compute ATAND(X) = 64*X + X*Q(X**2), where Q is a polynomial
                 034A   ;          of degree 2.
                 034A   ;    4. If X < 0 then
                 034A   ;       a. Compute Y = ATAND(|X|) using steps 1 to 3.
                 034A   ;       b. Set ATAND(X) = -Y.
                 034A   ;
                 034A   ;
                 034A   ; CALLING SEQUENCE:
                 034A   ;
                 034A   ;       Arctangent.wf.v = MTH$ATAND(x.rf.r)
                 034A   ;
                 034A   ; INPUT PARAMETERS:
                 034A
        00000004 034A           LONG = 4                        ; define longword multiplier
        00000004 034A           x = 1 * LONG                    ; x is an angle in radians
                 034A   ;
                 034A   ; IMPLICIT INPUTS:      none
                 034A   ;
                 034A   ; OUTPUT PARAMETERS:
                 034A   ;
                 034A   ;       VALUE:  floating arctangent angle of the argument (in degrees)
                 034A   ;
                 034A   ; IMPLICIT OUTPUTS:     none
                 034A   ;
                 034A   ; SIDE EFFECTS:
                 034A   ;
                 034A   ; Signals:      none
                 034A   ;
                 034A   ; NOTE: This procedure disables floating point underflow and integer
                 034A   ; overflow, causes no floating overflow or other arithmetic traps, and
                 034A   ; preserves enables across the call.
                 034A   ;
```

```
                         034A   ;---
                         034A
                         034A
                   401C  034A          .ENTRY  MTH$ATAND, ACMASK        ; standard call-by-reference entry
                         034C                                           ; disable DV (and FU), enable IV
                         034C          MTH$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                                           ; in case of an error in special JSB
                         0353                                           ; routine
      50     04 BC   50  0353          MOVF    @x(AP), R0               ; R0 = arg
                 72  10  0357          BSBB    MTH$ATAND_R4             ; call special ATAND rountine
                     04  0359          RET                              ; return - result in R0
                         035A
```

```
                          035A        .SBTTL  MTH$ATAND2 - Standard floating Arctangent With 2 Arguments
                          035A  ;++
                          035A  ; FUNCTIONAL DESCRIPTION:
                          035A  ;
                          035A  ; ATAND2  - single precision floating point function
                          035A  ;
                          035A  ; ATAND2(X,Y) is computed as following:
                          035A  ;
                          035A  ;       If Y = 0 or X/Y > 2**25, ATAND2(X,Y) = 90 * (sign X)
                          035A  ;       If Y > 0 and X/Y =< 2**25, ATAND2(X,Y) = ATAND(X/Y)
                          035A  ;       If Y < 0 and X/Y =< 2**25, ATAND2(X,Y) = 180 * (sign X) + ATAND(X/Y)
                          035A  ;
                          035A  ;
                          035A  ; CALLING SEQUENCE:
                          035A  ;
                          035A  ;       Arctangent2.wf.v = MTH$ATAND2(x.rf.r, y.rf.r)
                          035A  ;
                          035A  ; INPUT PARAMETERS:
                          035A
              00000004    035A          x = 1 * LONG                    ; x is the first argument
              00000008    035A          y = 2 * LONG                    ; y is the second argument
                          035A
                          035A  ; SIDE EFFECTS: See description of MTH$ATAND
                          035A  ;
                          035A  ;--
                          035A
                          035A
                  401C    035A          .ENTRY  MTH$ATAND2 ,ACMASK      ; standard call-by-reference entry
                          035C                                          ; disable DV (and FU), enable IV
                          035C          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                                          ; in case of an error in special JSB
                          0363                                          ; routine
         50    04 BC   50 0363          MOVF    @x(AP), R0              ; R0 = arg1
         51    08 BC   50 0367          MOVF    @y(AP), R1              ; R1 = arg2
                          036B  ;
                          036B  ; Test if Y = 0 or X/Y > 2**25
                          036B  ;
               35      13 036B          BEQL    INFD                    ; branch to INFD if Y = 0
  52   50   807F 8F    AB 036D          BICW3   #^X807F, R0, R2         ; R2 = exponent(X)
  53   51   807F 8F    AB 0373          BICW3   #^X807F, R1, R3         ; R3 = exponent(Y)
               52      53 A2 0379       SUBW    R3, R2                  ; R2 = exponent(X) - exponent(Y)
     0D00 8F   52      B1 037C          CMPW    R2, #26*128             ; compare R2 with 26
               1F      14 0381          BGTR    INFD                    ; if X/Y > 2**25, branch to INFD
                          0383  ;
                          0383  ; Test if Y > 0 or Y < 0
                          0383  ;
               51      B5 0383          TSTW    R1                      ; test the sign of Y
               18      14 0385          BGTR    A2PLUSD                 ; branch to A2PLUSD if Y > 0
               50      B5 0387          TSTW    R0                      ; test the sign of X
               0A      18 0389          BGEQ    A1PLUSD                 ; branch to A1PLUSD if X >= 0
                          038B  ;
                          038B  ; Y < 0 and X < 0 and X/Y =< 2**25
```

```
                            038B     ;
                   3B   10  038B                BSBB    MTH$ATAND_R4D           ; R0 = ATAND(X/Y)
50      00004434   8F   42  038D                SUBF    #LF_180, R0             ; R0 = -180 + ATAND(X/Y)
                        04  0394                RET                             ; return
                            0395     ;
                            0395     ; Y < 0 and X > 0 and X/Y =< 2**25
                            0395     ;
                            0395     A1PLUSD:
                   31   10  0395                BSBB    MTH$ATAND_R4D           ; R0 = ATAND(X/Y)
50      00004434   8F   40  0397                ADDF    #LF_180, R0             ; R0 = 180 + ATAND(X/Y)
                        04  039E                RET                             ; return
                            039F     ;
                            039F     ; Y > 0 and X/Y =< 2**25
                            039F     ;
                            039F     A2PLUSD:
                   27   10  039F                BSBB    MTH$ATAND_R4D           ; R0 = ATAND(X/Y)
                        04  03A1                RET                             ; return
                            03A2     ;
                            03A2     ; Y = 0 or X/Y > 2**25
                            03A2     ;
                            03A2     INFD:
                   50   B5  03A2                TSTW    R0                      ; test the sign of X
                   0A   14  03A4                BGTR    1$                      ; branch if X > 0
                   10   13  03A6                BEQL    2$                      ; branch if X = 0
50      0000C3B4   8F   50  03A8                MOVF    #LF_M90, R0             ; R0 = ATAND(X/Y) = -90
                        04  03AF                RET                             ; return
                            03B0
50      000043B4   8F   50  03B0     1$:        MOVF    #LF_90, R0              ; R0 = ATAND(X/Y) = 90
                        04  03B7                RET                             ; return
                            03B8
                            03B8     ;+
                            03B8     ; Here if X = 0 and Y = 0.  Signal INVALID ARG TO MATH LIBRARY
                            03B8     ; as a SEVERE error.
                            03B8     ;-
                            03B8
     50   01       0F   79  03B8     2$:        ASHQ    #15, #1, R0             ; R0/R1 = reserved operand which
                            03BC                                                ; is co180ed to CHF$L_MCH_SAVR0/R1
                            03BC                                                ; so a handler can fixup if wants
                            03BC                                                ; to continue
        7E     00'8F    9A  03BC                MOVZBL  #MTH$K_INVARGMAT, -(SP) ; code for INVALID ARG TO MATH LIBRARY
00000000'GF        01   FB  03C0                CALLS   #1, G^MTH$$SIGNAL       ; signal SEVERE error
                        04  03C7                RET                             ; return if handler continues
```

```
                                03C8          .SBTTL  MTH$ATAND_R4 - Special ATAND routine
                                03C8
                                03C8   ; Special ATAND - used by the standard routine, and directly.
                                03C8   ;
                                03C8   ; CALLING SEQUENCES:
                                03C8   ;       save anything needed in R0:R4
                                03C8   ;       MOVF    ...,R0                          ; input in R0
                                03C8   ;       JSB     MTH$ATAND_R4
                                03C8   ;       return with result in R0
                                03C8   ;
                                03C8   ; Note: This routine is  written  to  avoid causing any integer overflows,
                                03C8   ;    floating overflows, or floating underflows or divide by 0 conditions,
                                03C8   ;    whether enabled or not.
                                03C8   ;
                                03C8   ; REGISTERS USED:
                                03C8   ;       R0 - Floating argument then result
                                03C8   ;       R0:R3 - POLYF
                                03C8   ;       R4 - Pointer into ATAND_TABLE
                                03C8
                                03C8
                                03C8   MTH$ATAND_R4D:
              50   51   46      03C8          DIVF    R1, R0                  ; for our own use only!
                                03CB   MTH$ATAND_R4::                         ; Special ATAND routine
                   50   53      03CB          TSTF    R0                      ; R4 = X = argument
                   6D   19      03CD          BLSS    NEG_ARGD                ; Branch to negative argument logic
                                03CF   ;
                                03CF   ; Argument is positive
                                03CF   ;
     54   50   3EC0 8F   A3     03CF          SUBW3   #^X3EC0, R0, R4         ; Argument is less than 3/32,
                   45   19      03D5          BLSS    SMALLD                  ;  branch to small argument logic
          54   036F 8F   B1     03D7          CMPW    #^X036F, R4             ; Argument is greater that 11,
                   41   19      03DC          BLSS    LARGE_ARGD              ;  branch to large argument logic
                                03DE   ;
                                03DE   ; Logic for positive medium sized arguments.  Get pointer into ATAND_TABLE.
                                03DE   ;
     54   54     FC 8F   9C     03DE          ROTL    #-4, R4, R4             ; R4 = index into MTH$$AB_ATAN table
     54   FFFFFF00 8F    CA     03E3          BICL    #-256, R4               ; zero high order bits of index
     54    FC15 CF44     90     03EA          MOVB    MTH$$AB_ATAN[R4], R4    ; R4 = offset into ATAND_TABLE
     54    FD07 CF44     DE     03F0          MOVAL   ATAND_TABLE[R4], R4     ; R4 = pointer to XHI
                                03F6   ;
                                03F6   ; Compute z
                                03F6   ;
              51   84   D0      03F6          MOVL    (R4)+, R1               ; R1 = XHI
         52   50   51   45      03F9          MULF3   R1, R0, R2              ; R2 = X*XHI
              52   08   40      03FD          ADDF    #1, R2                  ; R2 = 1 + X*XHI
              50   51   42      0400          SUBF    R1, R0                  ; R0 = X - XHI
              50   52   46      0403          DIVF    R2, R0                  ; R0 = Z = (X - XHI)/(1 + X*XHI)
                                0406   ;
                                0406   ; Evaluate Z*P(Z**2)
                                0406   ;
              7E   50   D0      0406          MOVL    R0, -(SP)               ; Push Z onto the stack
              50   50   44      0409          MULF    R0, R0                  ; R0 = Z**2
     FD92 CF   02   50   55     040C          POLYF   R0, #ATANDLEN1-1, ATANDTAB1
                                0412                                          ; R0 = P(Z**2)
              50   8E   44      0412          MULF    (SP)+, R0               ; R0 = ATAND(Z) = Z*Q(Z**2)
              50   84   40      0415          ADDF    (R4)+, R0               ; R0 = ATAND_XHI_LO + ATAND(Z)
              50   64   40      0418          ADDF    (R4), R0                ; R0 = ATAND(X) = ATAND_XHI_HI +
```

```
                                041B                                              ;        (ATAND_XHI_LO + ATAND(Z))
                            05  041B              RSB                             ; Return
                                041C
                                041C
                 0088   31      041C  SMALLD: BRW       SMALL_ARGD                ; Dummy label used to avoid adding
                                041F                                              ;   an extra insrtuction in the
                                041F                                              ;   medium argument logic
                                041F  ;
                                041F  ; Large positive argument logic.
                                041F  ;
                                041F
                                041F  LARGE_ARGD:
54    0000C080 8F    50     47  041F          DIVF3     R0, #-1, R4               ; R4 = -W = -1/X
            50  54   54     45  0427          MULF3     R4, R4, R0                ; R0 = W**2
      FD73 CF   02   50     55  042B          POLYF     R0, #ATANDLEN1-1, ATANDTAB1
                                0431                                              ; R0 = P(W**2)
            50  54          44  0431          MULF      R4, R0                    ; R0 = -ATAND(Z) = -Z*P(W**2)
50    000043B4 8F           40  0434          ADDF      #LF_90, R0                ; R0 = ATAND(X) = 90 - ATAND(Z)
                            05  043B          RSB                                 ; Return
                                043C
                                043C  ;
                                043C  ; Logic for negative arguments
                                043C  ;
                                043C
                                043C  NEG_ARGD:
54    50    BEC0 8F         A3  043C          SUBW3     #^XBEC0, R0, R4           ; Argument is less than 3/32,
                     63     19  0442          BLSS      SMALL_ARGD                ;  branch to small argument logic
      54    036F 8F         B1  0444          CMPW      #^X036F, R4               ; Argument is greater than 11,
                     3F     19  0449          BLSS      N_LARGE_ARGD              ;  branch to large argument logic
                                044B  ;
                                044B  ; Logic for negative medium sized arguments.  Get index into ATAND_TABLE.
                                044B  ;
 54    54     FC 8F         9C  044B          ROTL      #-4, R4, R4               ; R4 = index into MTH$$AB_ATAN table
 54    FFFFFF00 8F          CA  0450          BICL      #-256, R4                 ; clear high order (unused) bits of ind
  54     FBA8 CF44          90  0457          MOVB      MTH$$AB_ATAN[R4], R4      ; R4 = offset into ATAN_TABLE
  54     FC9A CF44          DE  045D          MOVAL     ATAND_TABLE[R4], R4       ; R4 = pointer to XHI
                                0463  ;
                                0463  ; Compute Z
                                0463  ;
            51   84         D0  0463          MOVL      (R4)+, R1                 ; R1 = XHI
      52    50   51         45  0466          MULF3     R1, R0, R2                ; R2 = X*XHI
      52    08   52         43  046A          SUBF3     R2, #1, R2                ; R2 = 1 - X*XHI = 1 + X*(-XHI)
            50   51         40  046E          ADDF      R1, R0                    ; R0 = X + XHI = X - (-XHI)
            50   52         46  0471          DIVF      R2, R0                    ; R0 = Z
                                0474  ;
                                0474  ; Evaluate Z*P(Z**2)
                                0474  ;
            7E   50         D0  0474          MOVL      R0, -(SP)                 ; Push Z onto the stack
            50   50         44  0477          MULF      R0, R0                    ; R0 = Z**2
      FD24 CF   02   50     55  047A          POLYF     R0, #ATANDLEN1-1, ATANDTAB1
                                0480                                              ; R0 = P(Z**2)
            50   8E         44  0480          MULF      (SP)+, R0                 ; R0 = ATAND(Z) = Z*P(Z**2)
            50   84         42  0483          SUBF      (R4)+, R0                 ; R0 = ATAND_XHI_LO + ATAND(Z)
            50   64         42  0486          SUBF      (R4), R0                  ; R0 = ATAND(X) = ATAND_XHI_HI +
                                0489                                              ;        (ATAND_XHI_LO + ATAND(Z))
                            05  0489          RSB                                 ; Return
                                048A  ;
```

```
                                048A  ; Logic for large negative  arguments
                                048A  ;
                                048A
                                048A  N_LARGE_ARGD:
54   0000C080 8F   50   47      048A          DIVF3   R0, #-1, R4                 ; R4 = W = 1/|X|
           50 54   54   45      0492          MULF3   R4, R4, R0                  ; R0 = W**2
      FD08 CF 02   50   55      0496          POLYF   R0, #ATANDLEN1-1, ATANDTAB1
                                049C                                              ; R0 = P(W**2)
           50 54        44      049C          MULF    R4, R0                      ; R0 = ATAND(W) = W*P(W**2)
  50   000043B4 8F      42      049F          SUBF    #LF_90, R0                  ; R0 = ATAND(X) = ATAND(W) - 90
                        05      04A6          RSB                                 ; Return
                                04A7
                                04A7  ;
                                04A7  ; Small argument logic.
                                04A7  ;
                                04A7
                                04A7  SMALL_ARGD:
              54   50   50      04A7          MOVF    R0, R4                      ; R4 = argument = X
                   28   13      04AA          BEQL    3$                          ;
         50   8000 8F   AA      04AC          BICW    #^X8000, R0                 ; R0 = |X|
         50   3A00 8F   B1      04B1          CMPW    #^X3A00, R0                 ; Compare 2^-13 to |X|
                   08   19      04B6          BLSS    1$                          ; Needs POLY
  50   54   FCFC CF     45      04B8          MULF3   PI_OV_180_M_64, R4, R0      ; R0 = X*[pi/180 - 64]
                   0C   11      04BE          BRB     2$
                                04C0
              50   50   44      04C0  1$:     MULF    R0, R0                      ; R0 = X**2
      FCE7 CF 02   50   55      04C3          POLYF   R0, #ATANDLEN2-1, ATANDTAB2
                                04C9                                              ; R0 = Q(X**2)
              50   54   44      04C9          MULF    R4, R0                      ; R0 = X*Q(X**2)
         54   0300 8F   A0      04CC  2$:     ADDW    #^X300, R4                  ; R4 = X*2**6
              50   54   40      04D1          ADDF    R4, R0                      ; R0 = ATAND(X) = X*2**6 + X*Q(X**2)
                        05      04D4  3$:     RSB                                 ; Return
                                04D5
                                04D5
                                04D5
                                04D5          .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               00000214 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$$JACKET_HND   ********    X  01
MTH$$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
------------------                          --------------
_$255$DUA28:[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$:
```

0257 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

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