F 13

```
MMM        MMM  TTTTTTTTTTTTTTT HHH        HHH  RRRRRRRRRRR     TTTTTTTTTTTTTTT LLL
MMM        MMM  TTTTTTTTTTTTTTT HHH        HHH  RRRRRRRRRRR     TTTTTTTTTTTTTTT LLL
MMM        MMM  TTTTTTTTTTTTTTT HHH        HHH  RRRRRRRRRRR     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       HHHHHHHHHHHHHH  RRRRRRRRRRR          TTT        LLL
MMM        MMM        TTT       HHHHHHHHHHHHHH  RRRRRRRRRRR          TTT        LLL
MMM        MMM        TTT       HHHHHHHHHHHHHH  RRRRRRRRRRR          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        LLLLLLLLLLLLLL
MMM        MMM        TTT       HHH        HHH  RRR        RRR       TTT        LLLLLLLLLLLLLL
MMM        MMM        TTT       HHH        HHH  RRR        RRR       TTT        LLLLLLLLLLLLLL
```

FILEID**MTHDATAN

F 15

```
MM      MM  TTTTTTTTTT  HH      HH  DDDDDDDD      AAAAAA    TTTTTTTTTT    AAAAAA    NN      NN
MM      MM  TTTTTTTTTT  HH      HH  DDDDDDDD      AAAAAA    TTTTTTTTTT    AAAAAA    NN      NN
MMMM  MMMM      TT      HH      HH  DD      DD  AA      AA      TT      AA      AA  NN      NN
MMMM  MMMM      TT      HH      HH  DD      DD  AA      AA      TT      AA      AA  NN      NN
MM  MM  MM      TT      HH      HH  DD      DD  AA      AA      TT      AA      AA  NNNN    NN
MM  MM  MM      TT      HH      HH  DD      DD  AA      AA      TT      AA      AA  NNNN    NN
MM      MM      TT      HHHHHHHHHH  DD      DD  AA      AA      TT      AA      AA  NN  NN  NN
MM      MM      TT      HHHHHHHHHH  DD      DD  AA      AA      TT      AA      AA  NN  NN  NN
MM      MM      TT      HH      HH  DD      DD  AAAAAAAAAA      TT      AAAAAAAAAA  NN    NNNN
MM      MM      TT      HH      HH  DD      DD  AAAAAAAAAA      TT      AAAAAAAAAA  NN    NNNN
MM      MM      TT      HH      HH  DD      DD  AA      AA      TT      AA      AA  NN      NN    ::::
MM      MM      TT      HH      HH  DD      DD  AA      AA      TT      AA      AA  NN      NN    ::::
MM      MM      TT      HH      HH  DDDDDDDD    AA      AA      TT      AA      AA  NN      NN    ::::
MM      MM      TT      HH      HH  DDDDDDDD    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        .TITLE  MTH$DATAN       ; Floating Point Arc Tangent Functions
0000                                ; (DATAN,DATAN2,DATAND,DATAND2)
0000        .IDENT /2-004/          ; File: MTHDATAN.MAR  EDIT: RNH2004
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$DATAN is a function which returns the floating point arctangent
0000  ; value (in radians) of its double precision floating point argument.
0000  ; MTH$DATAN2 is two argument double floating arctangent. The  call is
0000  ; standard call-by-reference.
0000  ; MTH$DATAN_R7  is  a  special routine which is the same as MTH$DATAN
0000  ; except a  faster non-standard JSB call is used with the argument in
0000  ; R0 and no registers are saved.
0000  ;
0000  ; MTH$DATAND is a function which returns the floating point arctangent
0000  ; value (in degrees) of its double precision  floating point argument.
0000  ; MTH$DATAND2 is two  argument double floating arctangent. The call is
0000  ; standard call-by-reference.
0000  ; MTH$DATAND_R7 is  a  special routine which is the same as MTH$DATAND
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     ; VERSION: 02
0000     ;
0000     ; HISTORY:
0000     ; AUTHOR:
0000     ;         Bob Hanek, 05-Jun-81: Version 02
0000     ;
0000     ; MODIFIED BY:
0000     ;
0000     ;
```

```
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$DATANDATAN2
0000    ;
0000
0000
0000    ; 01-1  Code saving after code review March 1977
0000    ;       In DATAN2, fix references to OWN constants so DATAN2 will work.
0000    ; 01-3  In MTH$DATAN2, comparison of exponents of arguments X and
0000    ;       Y is with 58 instead of 26.
0000    ;
0000    ; 01-8  - Signal INVALID ARG TO MATH LIBRARY if x=y=0.  TNH 16-June-78
0000    ; 01-9  - Fix comments.  TNH 16-June-78
0000    ; 01-10 - Move .ENTRY mask to module header.  TNH 14-Aug-78
0000    ; 1-011 - Update version number and copyright notice.  JBS 16-NOV-78
0000    ; 1-012 - Change MTH__INVARG to MTH$K_INVARGMAT.  JBS 07-DEC-78
0000    ; 1-013 - Add "_" to the PSECT directive.  JBS 22-DEC-78
0000    ; 1-014 - Declare externals.  SBL 17-May-1979
0000    ; 1-015 - Added deree entry points.  RNH 15-MAR-1981
0000    ;
0000    ;
0000    ; Edit History for Version 01 of MTH$DATANDATAN2
0000    ;
0000    ;
0000    ; 2-002 - Use G^ addressing for externals.  SBL 24-Aug-1981
0000    ; 2-003 - Changed MTH$DATAND2 entry to MTH$DATAN2D in order to conform
0000    ;         to the original specification.  RNH 05-Oct-81
0000    ; 2-004 - Un-did previous edit to be consistent with PL/1
0000    ;       - Modified small argument processing to avoid a microcode bug
0000    ;         in the FPA.  RNH 18-Dec-81
```

```
                     0000          .SBTTL  DECLARATIONS     ; Declarative Part of Module
                     0000
                     0000  ;
                     0000  ; INCLUDE FILES:         MTHJACKET.MAR, MTHATAN.MAR
                     0000  ;
                     0000
                     0000  ;
                     0000  ; EXTERNAL SYMBOLS:
                     0000  ;
                     0000          .DSABL  GBL
                     0000          .EXTRN  MTH$K_INVARGMAT
                     0000          .EXTRN  MTH$$SIGNAL                ; Signal SEVERE error
                     0000          .EXTRN  MTH$$AB_ATAN               ; Gobal table used by all Arctangent
                     0000                                             ;     routines.  Part of MTHATAN.MAR
                     0000  ;
                     0000
                     0000  ; EQUATED SYMBOLS:
                     0000  ;
           000040FC  0000          ACMASK = ^M<IV, R2, R3, R4, R5, R6, R7> ; .ENTRY register mask, int
                     0000                                             ;  ovf enabled
                     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   ;       **************************** Constants for DATAN ****************************
                    0000   ;
                    0000   ;
                    0000   ;       Each  entry of the DATAN_TABLE contains the the values of XHI, DATAN_XHI_LO
                    0000   ;       and DATAN_XHI_HI respectively.  The table is indexed by a pointer  obtained
                    0000   ;       from the MTH$$AB_ATAN table. The MTH$$AB_ATAN table is common to all of the
                    0000   ;       arctangent  routines  and is included as part of the MTHATAN module.  NOTE:
                    0000   ;       For performance reasons it is important to have  the  DATAN_TABLE  longword
                    0000   ;       aligned.
                    0000   ;
                    0000   ;
                    0000           .ALIGN  LONG
                    0000
                    0000   DATAN_TABLE:
                    0000   ; Entry  0
00000000 F87E3ED7   0000           .QUAD   ^X00000000F87E3ED7      ;  0.10545442998409271E+00
E21C5BB4 E52DA277   0008           .QUAD   ^XE21C5BB4E52DA277      ; -0.83990168661711120E-18
B377B27A 2CE63ED7   0010           .QUAD   ^XB377B27A2CE63ED7      ;  0.10506611091781236E+00
                    0018   ; Entry  1
00000000 FB703F03   0018           .QUAD   ^X00000000FB703F03      ;  0.12888884544372559E+00
8EC75105 6B81A001   0020           .QUAD   ^X8EC751056B81A001      ; -0.27405738718612654E-19
DC7B37BC 422F3F03   0028           .QUAD   ^XDC7B37BC422F3F03      ;  0.12818216111847079E+00
                    0030   ; Entry  2
00000000 F63E3F1F   0030           .QUAD   ^X00000000F63E3F1F      ;  0.15621277689933777E+00
215F05DE B08D22D7   0038           .QUAD   ^X215F05DEB08D22D7      ;  0.14615699319155353E-17
3692CB13 ADF03F1E   0040           .QUAD   ^X3692CB13ADF03F1E      ;  0.15496040572616338E+00
                    0048   ; Entry  3
00000000 E4EC3F47   0048           .QUAD   ^X00000000E4EC3F47      ;  0.19520920515060425E+00
7B536972 03519EEA   0050           .QUAD   ^X7B53697203519EEA      ; -0.61942715015325782E-20
CF73AADA 69613F45   0058           .QUAD   ^XCF73AADA69613F45      ;  0.19278481107058050E+00
                    0060   ; Entry  4
00000000 C3D13F7F   0060           .QUAD   ^X00000000C3D13F7F      ;  0.24977041780948639E+00
B4E3255F 9F63A232   0068           .QUAD   ^XB4E3255F9F63A232      ; -0.60519693165102660E-18
C6A74C33 A30A3F7A   0070           .QUAD   ^XC6A74C33A30A3F7A      ;  0.24476257410146354E+00
                    0078   ; Entry  5
00000000 DB973F9F   0078           .QUAD   ^X00000000DB973F9F      ;  0.31222221255302429E+00
94AB79A5 57201F4A   0080           .QUAD   ^X94AB79A557201F4A      ;  0.10711808370652331E-19
AEB45198 F28D3F9A   0088           .QUAD   ^XAEB45198F28D3F9A      ;  0.30263177510309217E+00
                    0090   ; Entry  6
00000000 9E8E3FC7   0090           .QUAD   ^X000000009E8E3FC7      ;  0.38988155126571655E+00
37BDC0BB D813A29B   0098           .QUAD   ^X37BDC0BBD813A29B      ; -0.10560403693868137E-17
D2E26F78 56713FBE   00A0           .QUAD   ^XD2E26F7856713FBE      ;  0.37175325856916232E+00
                    00A8   ; Entry  7
00000000 33B63FFF   00A8           .QUAD   ^X0000000033B63FFF      ;  0.49844139814376831E+00
E8718A35 E17DA331   00B0           .QUAD   ^XE8718A35E17DA331      ; -0.24107346684847003E-17
3007B09B BFAF3FEC   00B8           .QUAD   ^X3007B09BBFAF3FEC      ;  0.46239995032155439E+00
                    00C0   ; Entry  8
00000000 F8EB4026   00C0           .QUAD   ^X00000000F8EB4026      ;  0.65223568677902222E+00
05D855B7 DF45A3BB   00C8           .QUAD   ^X05D855B7DF45A3BB      ; -0.50922848759855227E-17
B5B6B9DC F4384013   00D0           .QUAD   ^XB5B6B9DCF4384013      ;  0.57794527566576090E+00
                    00D8   ; Entry  9
00000000 0712405E   00D8           .QUAD   ^X000000000712405E      ;  0.86729538440704346E+00
F2BF420E DB20A3C7   00E0           .QUAD   ^XF2BF420EDB20A3C7      ; -0.54171066766593593E-17
E63CB34A E62B4036   00E8           .QUAD   ^XE63CB34AE62B4036      ;  0.71444962622890612E+00
                    00F0   ; Entry 10
00000000 CBD84095   00F0           .QUAD   ^X00000000CBD84095      ;  0.11702833175659180E+01
```

```
69749B08 2D47A3EC  00F8          .QUAD   ^X69749B082D47A3EC       ; -0.64015869933736197E-17
66530222 1B62405D  0100          .QUAD   ^X665302221B62405D       ;  0.86369907905682312E+00
                   0108   ; Entry 11
00000000 8DEB40D2  0108          .QUAD   ^X000000008DEB40D2       ;  0.16449559926986694E+01
A226B1AA 552C242C  0110          .QUAD   ^XA226B1AA552C242C       ;  0.93421751035229859E-17
01D0C309 25404083  0118          .QUAD   ^X01D0C30925404083       ;  0.10245743706054911E+01
                   0120   ; Entry 12
00000000 88054124  0120          .QUAD   ^X0000000088054124       ;  0.25708019733428955E+01
D3691BCA CD08244B  0128          .QUAD   ^XD3691BCACD08244B       ;  0.11048069196521280E-16
45BD59C4 93CA4099  0130          .QUAD   ^X45BD59C493CA4099       ;  0.11998227060617363E+01
                   0138   ; Entry 13
00000000 7D0E41AB  0138          .QUAD   ^X000000007D0E41AB       ;  0.53590154647827148E+01
73B91758 4359A428  0140          .QUAD   ^X73B917584359A428       ; -0.91215597452526939E-17
DFB53237 72D240B1  0148          .QUAD   ^XDFB5323772D240B1       ;  0.13863165612417540E+01
                   0150
                   0150   ;
                   0150   ; Tables to be used in POLYD for computing DATAN:  DATANTAB1 is obtained
                   0150   ; from Hart et. al. (No. 4904).  DATANTAB2 is the same as DATANTAB1 except
                   0150   ; that C0 is set to 0
                   0150   ;
                   0150
                   0150   DATANTAB1:
                   0150
4B4F7B0B FCA13E98  0150          .QUAD   ^X4B4F7B0BFCA13E98       ; C6 = 0.74700604980000000E-01
BA534D4C 1F19BEBA  0158          .QUAD   ^XBA534D4C1F19BEBA       ; C5 = -.90879628821850000E-01
D5B0D0E5 8E1E3EE3  0160          .QUAD   ^XD5B0D0E58E1E3EE3       ; C4 = 0.11111091685300320E+00
EEBF86F9 4924BF12  0168          .QUAD   ^XEEBF86F94924BF12       ; C3 = -.14285714219884826E+00
200FCCC8 CCCC3F4C  0170          .QUAD   ^X200FCCC8CCCC3F4C       ; C2 = 0.19999999999893708E+00
AA4EAAAA AAAABFAA  0178          .QUAD   ^XAA4EAAAAAAAABFAA       ; C1 = -.33333333333333269E+00
00000000 00004080  0180          .QUAD   ^X0000000000004080       ; C0 = 0.10000000000000000E+01
         00000007  0188   DATANLEN1 = .- DATANTAB1/8
                   0188
                   0188   DATANTAB2:
4B4F7B0B FCA13E98  0188          .QUAD   ^X4B4F7B0BFCA13E98       ; C6 = 0.74700604980000000E-01
BA534D4C 1F19BEBA  0190          .QUAD   ^XBA534D4C1F19BEBA       ; C5 = -.90879628821850000E-01
D5B0D0E5 8E1E3EE3  0198          .QUAD   ^XD5B0D0E58E1E3EE3       ; C4 = 0.11111091685300320E+00
EEBF86F9 4924BF12  01A0          .QUAD   ^XEEBF86F94924BF12       ; C3 = -.14285714219884826E+00
200FCCC8 CCCC3F4C  01A8          .QUAD   ^X200FCCC8CCCC3F4C       ; C2 = 0.19999999999893708E+00
AA4EAAAA AAAABFAA  01B0          .QUAD   ^XAA4EAAAAAAAABFAA       ; C1 = -.33333333333333269E+00
00000000 00000000  01B8          .QUAD   ^X0000000000000000       ; C0 = 0.00000000000000000E+00
         00000007  01C0   DATANLEN2 = .- DATANTAB2/8
                   01C0
                   01C0   D_PI:
68C2A221 0FDA4149  01C0          .QUAD   ^X68C2A2210FDA4149       ; pi
                   01C8   D_PI_OVER_2:
68C2A221 0FDA40C9  01C8          .QUAD   ^X68C2A2210FDA40C9       ; pi/2
                   01D0   D_MPI_OVER_2:
68C2A221 0FDAC0C9  01D0          .QUAD   ^X68C2A2210FDAC0C9       ; -pi/2
                   01D8   D_PI_OVER_2_HI:
68C2A221 0FDA40C9  01D8          .QUAD   ^X68C2A2210FDA40C9       ; High order bits of pi/2
                   01E0   D_PI_OVER_2_LO:
03708A2E 131923D3  01E0          .QUAD   ^X03708A2E131923D3       ; Low order bits of pi/2
                   01E8
```

```
                          01E8  ;
                          01E8  ; ************************** Constants for ATAND **********************************
                          01E8  ;
                          01E8  ; Each entry of the DATAND_TABLE contains the the values of XHI, DATAND_XHI_LO
                          01E8  ; and DATAND_XHI_HI respectively.  The table is indexed by a pointer  obtained
                          01E8  ; from  the MTH$$AB_ATAN table. The MTH$$AB_ATAN table is common to all of the
                          01E8  ; arctangent routines and is included as part of the MTHATAN module. NOTE: For
                          01E8  ; performance reasons it is important to have the DATAN_TABLE longword aligned.
                          01E8  ;
                          01E8  ;
                          01E8
                          01E8  DATAND_TABLE:
                          01E8  ; Entry  0
00000000 F87E3ED7         01E8            .QUAD    ^X00000000F87E3ED7      ;  0.10545442998409271E+00
EC84E32B 2B2BA44F         01F0            .QUAD    ^XEC84E32B2B2BA44F      ; -0.11230634392205251E-16
F76467D8 A29141C0         01F8            .QUAD    ^XF76467D8A29141C0      ;  0.60198447254440279E+01
                          0200  ; Entry  1
00000000 FB703F03         0200            .QUAD    ^X00000000FB703F03      ;  0.12888884544372559E+00
96F9C4C8 A0012420         0208            .QUAD    ^X96F9C4C8A0012420      ;  0.87075001607967749E-17
795BCF00 047A41EB         0210            .QUAD    ^X795BCF00047A41EB      ;  0.73442968409542958E+01
                          0218  ; Entry  2
00000000 F63E3F1F         0218            .QUAD    ^X00000000F63E3F1F      ;  0.15621277689933777E+00
FC66745A F63822CA         0220            .QUAD    ^XFC66745AF63822CA      ;  0.13753226458048320E-17
5E9101FB 0EA7420E         0228            .QUAD    ^X5E9101FB0EA7420E      ;  0.88785772397440363E+01
                          0230  ; Entry  3
00000000 E4EC3F47         0230            .QUAD    ^X00000000E4EC3F47      ;  0.19520920515060425E+00
728CB36C 241C25C2         0238            .QUAD    ^X728CB36C241C25C2      ;  0.84195264883526611E-16
EE5FAC64 BB6A4230         0240            .QUAD    ^XEE5FAC64BB6A4230      ;  0.11045756028571212E+02
                          0248  ; Entry  4
00000000 C3D13F7F         0248            .QUAD    ^X00000000C3D13F7F      ;  0.24977041780948639E+00
72036DE9 3B89A5D5         0250            .QUAD    ^X72036DE93B89A5D5      ; -0.92474884414648262E-16
D4B09F5E 61BD4260         0258            .QUAD    ^XD4B09F5E61BD4260      ;  0.14023862478771928E+02
                          0260  ; Entry  5
00000000 DB973F9F         0260            .QUAD    ^X00000000DB973F9F      ;  0.31222221255302429E+00
B8A22FB8 0616A565         0268            .QUAD    ^XB8A22FB80616A565      ; -0.49661615106200334E-16
018A1366 B758428A         0270            .QUAD    ^X018A1366B758428A      ;  0.17339523459959485E+02
                          0278  ; Entry  6
00000000 9E8E3FC7         0278            .QUAD    ^X000000009E8E3FC7      ;  0.38988155126571655E+00
344BEAED E7C2A489         0280            .QUAD    ^X344BEAEDE7C2A489      ; -0.14951724532714388E-16
F73829B3 662E42AA         0288            .QUAD    ^XF73829B3662E42AA      ;  0.21299892736248605E+02
                          0290  ; Entry  7
00000000 33B63FFF         0290            .QUAD    ^X0000000033B63FFF      ;  0.49844139814376831E+00
752E7920 CC7825F9         0298            .QUAD    ^X752E7920CC7825F9      ;  0.10833292304647813E-15
13348584 F2D242D3         02A0            .QUAD    ^X13348584F2D242D3      ;  0.26493565600483999E+02
                          02A8  ; Entry  8
00000000 F8EB4026         02A8            .QUAD    ^X00000000F8EB4026      ;  0.65223568677902222E+00
906EBE73 31EC258D         02B0            .QUAD    ^X906EBE7331EC258D      ;  0.61233578397063759E-16
214E9029 748E4304         02B8            .QUAD    ^X214E9029748E4304      ;  0.33113825085173017E+02
                          02C0  ; Entry  9
00000000 0712405E         02C0            .QUAD    ^X000000000712405E      ;  0.86729538440704346E+00
61B2D72E 6F942641         02C8            .QUAD    ^X61B2D72E6F942641      ;  0.16777886794863949E-15
A4871377 BD634323         02D0            .QUAD    ^XA4871377BD634323      ;  0.40934948257615480E+02
                          02D8  ; Entry 10
00000000 CBD84095         02D8            .QUAD    ^X00000000CBD84095      ;  0.11702833175659180E+01
F3FAEAF9 FCD82585         02E0            .QUAD    ^XF3FAEAF9FCD82585      ;  0.58107895623740531E-16
722AC5D2 F1FB4345         02E8            .QUAD    ^X722AC5D2F1FB4345      ;  0.49486311999291994E+02
                          02F0  ; Entry 11
```

```
00000000 8DEB40D2  02F0          .QUAD   ^X000000008DEB40D2      ;  0.16449559926986694E+01
E4C2D4E7 9E22A6C5  02F8          .QUAD   ^XE4C2D4E79E22A6C5      ; -0.34281209639921420E-15
BF6999B3 D0AD436A  0300          .QUAD   ^XBF6999B3D0AD436A      ;  0.58703787232967309E+02
                   0308    ; Entry 12
00000000 88054124  0308          .QUAD   ^X0000000088054124      ;  0.25708019733428955E+01
BEDD635C 359CA65C  0310          .QUAD   ^XBEDD635C359CA65C      ; -0.19100122312198548E-15
9ABE70A0 7D534389  0318          .QUAD   ^X9ABE70A07D534389      ;  0.68744777221303021E+02
                   0320    ; Entry 13
00000000 7D0E41AB  0320          .QUAD   ^X000000007D0E41AB      ;  0.53590154647827148E+01
58110A04 52C3A4B5  0328          .QUAD   ^X58110A0452C3A4B5      ; -0.19659110337997096E-16
66B57F7F DC34439E  0330          .QUAD   ^X66B57F7FDC34439E      ;  0.79430088028242020E+02
                   0338
                   0338    ;
                   0338    ; Tables to be used in POLYD for computing DATAND:  DATANDTAB1 is obtained
                   0338    ; by multiplying the coefficients given in Hart et. al. (No. 4904) by
                   0338    ; 180/pi.  DATANDTAB2 is the same as DATANDTAB1 except that C0 is set to
                   0338    ; 180/pi - 64 instead of 180/pi.
                   0338    ;
                   0338
                   0338    DATANDTAB1:
B22B334C F6004188  0338          .QUAD   ^XB22B334CF6004188      ; C6 = 0.42800293924279392E+01
270AAD65 9FE6C1A6  0340          .QUAD   ^X270AAD659FE6C1A6      ; C5 = -.52070191752074788E+01
F448F26A B7CC41CB  0348          .QUAD   ^XF448F26AB7CC41CB      ; C4 = 0.63661865935060939E+01
404149F1 F637C202  0350          .QUAD   ^X404149F1F637C202      ; C3 = -.81851113212942581E+01
DD37DC13 58B34237  0358          .QUAD   ^XDD37DC1358B34237      ; C2 = 0.11459155902555563E+02
5F813769 C9EBC298  0360          .QUAD   ^X5F813769C9EBC298      ; C1 = -.19098593171027404E+02
0FBED31E 2EE04365  0368          .QUAD   ^X0FBED31E2EE04365      ; C0 = 0.57295779513082321E+02
         00000007  0370    DATANDLEN1 = .- DATANDTAB1/8
                   0370
                   0370    DATANDTAB2:
B22B334C F6004188  0370          .QUAD   ^XB22B334CF6004188      ; C6 = 0.42800293924279392E+01
270AAD65 9FE6C1A6  0378          .QUAD   ^X270AAD659FE6C1A6      ; C5 = -.52070191752074788E+01
F448F26A B7CC41CB  0380          .QUAD   ^XF448F26AB7CC41CB      ; C4 = 0.63661865935060939E+01
404149F1 F637C202  0388          .QUAD   ^X404149F1F637C202      ; C3 = -.81851113212942581E+01
DD37DC13 58B34237  0390          .QUAD   ^XDD37DC1358B34237      ; C2 = 0.11459155902555563E+02
5F813769 C9EBC298  0398          .QUAD   ^X5F813769C9EBC298      ; C1 = -.19098593171027404E+02
                   03A0    D_PI_OV_180_M_64:
8212670F 88F9C1D6  03A0          .QUAD   ^X8212670F88F9C1D6      ; C0 = -.67042204869176791E+01
         00000007  03A8    DATANDLEN2 = .- DATANDTAB2/8
                   03A8
                   03A8
                   03A8    D_90:
00000000 000043B4  03A8          .QUAD   ^X00000000000043B4      ; 90.
                   03B0    D_M90:
00000000 0000C3B4  03B0          .QUAD   ^X000000000000C3B4      ; -90.
                   03B8    D_180:
00000000 00004434  03B8          .QUAD   ^X0000000000004434      ; 180
                   03C0
```

```
                03C0          .SBTTL  MTH$DATAN - Standard Single Precision Floating Arc Tangent
                03C0
                03C0
                03C0    ;++
                03C0    ; FUNCTIONAL DESCRIPTION:
                03C0    ;
                03C0    ; DATAN - double precision floating point function
                03C0    ;
                03C0    ; DATAN is computed using the following steps:
                03C0    ;
                03C0    ;    1. If X > 11 then
                03C0    ;          a. Let W = 1/X.
                03C0    ;          b. Compute DATAN(W) = W*P(W**2), where P is a polynomial of
                03C0    ;             degree 6.
                03C0    ;          c. Set DATAN(X) = pi/2 - DATAN(W)
                03C0    ;    2. If 3/32 =< X =< 11 then
                03C0    ;          a. Obtain XHI by table look-up.
                03C0    ;          b. Compute Z = (X - XHI)/(1 + X*XHI).
                03C0    ;          c. Compute DATAN(Z) = Z*P(Z**2), where P is a polynomial of
                03C0    ;             degree 6.
                03C0    ;          d. Obtain DATAN(XHI) by table look-up.  DATAN(XHI) will have
                03C0    ;             two parts - the high order bits, DATAN_XHI_HI, and the low
                03C0    ;             order bits, DATAN_XHI_LO.
                03C0    ;          e. Compute DATAN(X) = DATAN_XHI_HI + (DATAN_XHI_LO + DATAN(Z)).
                03C0    ;    3. If 0 =< X < 3/32 then
                03C0    ;          a. Compute DATAN(X) = X + X*Q(X**2), where Q is a polynomial
                03C0    ;             of degree 6.
                03C0    ;    4. If X < 0 then
                03C0    ;          a. Compute Y = DATAN(|X|) using steps 1 to 3.
                03C0    ;          b. Set DATAN(X) = -Y.
                03C0    ;
                03C0    ;
                03C0    ; CALLING SEQUENCE:
                03C0    ;
                03C0    ;       Arctangent.wd.v = MTH$DATAN(x.rd.r)
                03C0    ;
                03C0    ; INPUT PARAMETERS:
                03C0
       00000004 03C0          LONG = 4                          ; define longword multiplier
       00000004 03C0          x = 1 * LONG                      ; x is an angle in radians
                03C0    ;
                03C0    ; IMPLICIT INPUTS:      none
                03C0    ;
                03C0    ; OUTPUT PARAMETERS:
                03C0    ;
                03C0    ;       VALUE:  double precision floating arctangent angle of the argument
                03C0    ;
                03C0    ; IMPLICIT OUTPUTS:     none
                03C0    ;
                03C0    ; SIDE EFFECTS:
                03C0    ;
                03C0    ; Signals:      none
                03C0    ;
                03C0    ; NOTE: This procedure disables floating point underflow, enable integer
                03C0    ; overflow, causes no floating overflow or other arithmetic traps, and
                03C0    ; preserves enables across the call.
                03C0    ;
```

```
                            03C0  ;---
                            03C0
                            03C0
                     40FC   03C0           .ENTRY  MTH$DATAN, ACMASK          ; standard call-by-reference entry
                            03C2                                              ; disable DV (and FU), enable IV
                            03C2           MTH$FLAG_JACKET                    ; flag that this is a jacket procedure
                            03C2
       6D  00000000'GF  9E  03C2           MOVAB   G^MTH$$JACKET_HND, (FP)
                            03C9                                              ; set handler address to jacket
                            03C9                                              ; handler
                            03C9
                            03C9                                              ; in case of an error in special JSB
                            03C9                                              ; routine
          50    04 BC   70  03C9           MOVD    @x(AP), R0                 ; R0/R1 = arg
                   6A   10  03CD           BSBB    MTH$DATAN_R7               ; call special DATAN rountine
                        04  03CF           RET                                ; return - result in R0
                            03D0
```

```
                             03D0          .SBTTL  MTH$DATAN2 - Standard Double Floating Arctangent With 2 Arguments
                             03D0   ;++
                             03D0   ; FUNCTIONAL DESCRIPTION:
                             03D0   ;
                             03D0   ; DATAN2  - double precision floating point function
                             03D0   ;
                             03D0   ; DATAN2(X,Y) is computed as following:
                             03D0   ;
                             03D0   ;       If Y = 0 or X/Y > 2**57, DATAN2(X,Y) = PI/2 * (sign X)
                             03D0   ;       If Y > 0 and X/Y =< 2**57, DATAN2(X,Y) = DATAN(X/Y)
                             03D0   ;       If Y < 0 and X/Y =< 2**57, DATAN2(X,Y) = PI * (sign X) + DATAN(X/Y)
                             03D0   ;
                             03D0   ;
                             03D0   ; CALLING SEQUENCE:
                             03D0   ;
                             03D0   ;       Arctangent2.wd.v = MTH$DATAN2(x.rd.r, y.rd.r)
                             03D0   ;
                             03D0   ; INPUT PARAMETERS:
                             03D0
                 00000004    03D0          x = 1 * LONG                    ; x is the first argument
                 00000008    03D0          y = 2 * LONG                    ; y is the second argument
                             03D0
                             03D0   ; SIDE EFFECTS: See description of MTH$DATAN
                             03D0   ;
                             03D0   ;--
                             03D0
                             03D0
                     40FC    03D0          .ENTRY  MTH$DATAN2, ACMASK      ; standard call-by-reference entry
                             03D2                                          ; disable DV (and FU), enable IV
                             03D2          MTH$FLAG_JACKET                 ; flag that this is a jacket procedure
                             03D2
    6D   00000000'GF   9E    03D2          MOVAB   G^MTH$$JACKET_HND, (FP)
                             03D9                                          ; set handler address to jacket
                             03D9                                          ; handler
                             03D9
                             03D9                                          ; in case of an error in special JSB
                             03D9                                          ; routine
          50    04 BC  70    03D9          MOVD    @x(AP), R0              ; R0/R1 = arg1
          52    08 BC  70    03DD          MOVD    @y(AP), R2              ; R2/R3 = arg2
                             03E1   ;
                             03E1   ; Test if Y = 0 or X/Y > 2**57
                             03E1   ;
                 31    13    03E1          BEQL    INF                     ; branch to INF if Y = 0
 54    50   807F 8F    AB    03E3          BICW3   #^X807F, R0, R4         ; R4 = exponent(X)
 55    52   807F 8F    AB    03E9          BICW3   #^X807F, R2, R5         ; R5 = exponent(Y)
           54    55    A2    03EF          SUBW    R5, R4                  ; R4 = exponent(X) - exponent(Y)
       1D00 8F   54    B1    03F2          CMPW    R4, #58*128             ; compare R4 with 58
                 1B    14    03F7          BGTR    INF                     ; if X/Y > 2**57, branch to INF
                             03F9   ;
                             03F9   ; Test if Y > 0 or Y < 0
                             03F9   ;
                 52    B5    03F9          TSTW    R2                      ; test the sign of Y
                 14    14    03FB          BGTR    A2PLUS                  ; branch to A2PLUS if Y > 0
                 50    B5    03FD          TSTW    R0                      ; test the sign of X
                 08    18    03FF          BGEQ    A1PLUS                  ; branch to A1PLUS if X >= 0
                             0401   ;
                             0401   ; Y < 0 and X < 0 and X/Y =< 2**57
```

```
                              0401  ;
                33    10      0401          BSBB    MTH$DATAN_R7D           ; R0/R1 = DATAN(X/Y)
        50    FDB9 CF    62   0403          SUBD    D_PI, R0                ; R0/R1 = -PI + DATAN(X/Y)
                      04      0408          RET                             ; return
                              0409  ;
                              0409  ; Y < 0 and X > 0 and X/Y =< 2**57
                              0409  ;
                              0409  A1PLUS:
                2B    10      0409          BSBB    MTH$DATAN_R7D           ; R0/R1 = DATAN(X/Y)
        50    FDB1 CF    60   040B          ADDD    D_PI, R0                ; R0/R1 = PI + DATAN(X/Y)
                      04      0410          RET                             ; return
                              0411  ;
                              0411  ; Y > 0 and X/Y =< 2**57
                              0411  ;
                              0411  A2PLUS:
                23    10      0411          BSBB    MTH$DATAN_R7D           ; R0/R1 = DATAN(X/Y)
                      04      0413          RET                             ; return
                              0414  ;
                              0414  ; Y = 0 or X/Y > 2**57
                              0414  ;
                              0414  INF:
                50    B5      0414          TSTW    R0                      ; test the sign of X
                08    14      0416          BGTR    1$                      ; branch if X > 0
                0C    13      0418          BEQL    2$                      ; branch if X = 0
        50    FDB2 CF    70   041A          MOVD    D_MPI_OVER_2, R0        ; R0/R1 = DATAN(X/Y) = -PI/2
                      04      041F          RET                             ; return
                              0420
        50    FDA4 CF    70   0420  1$:     MOVD    D_PI_OVER_2, R0         ; R0/R1 = DATAN(X/Y) = PI/2
                      04      0425          RET                             ; return
                              0426
                              0426  ;+
                              0426  ; Here if both X = 0 and Y = 0.  Signal INVALID ARG TO MATH LIBRARY
                              0426  ;-
                              0426
        50    01    0F   79   0426  2$:     ASHQ    #15, #1, R0             ; R0/R1 = reserved operand, copied
                              042A                                          ; to CHF$L_MCH_SAVR0/R1 so handlers
                              042A                                          ; can change if they want to continue.
        7E    00'8F      9A   042A          MOVZBL  #MTH$K_INVARGMAT, -(SP) ; code for INVALID ARG TO MATH LIBRARY
 00000000'GF    01       FB   042E          CALLS   #1, G^MTH$$SIGNAL       ; Signal SEVERE error
                      04      0435          RET                             ; return if a handler says SS$_CONTINUE
```

```
                        0436        .SBTTL  MTH$DATAN_R7 - Special DATAN routine
                        0436
                        0436  ; Special DATAN - used by the standard routine, and directly.
                        0436  ;
                        0436  ; CALLING SEQUENCES:
                        0436  ;         save anything needed in R0:R7
                        0436  ;         MOVD    ..., R0                     ; input in R0/R1
                        0436  ;         JSB     MTH$DATAN_R7
                        0436  ;         return with result in R0/R1
                        0436  ; Note:  This  routine  is written to avoid causing any integer overflows,
                        0436  ;    floating overflows, or floating underflows or divide by 0 conditions,
                        0436  ;    whether enabled or not.
                        0436  ;
                        0436  ; REGISTERS USED:
                        0436  ;         R0/R1 - Floating argument then result
                        0436  ;         R0:R5 - POLYD
                        0436  ;         R6    - Pointer into DATAN_TABLE
                        0436  ;         R6/R7 - Y during POLYD
                        0436
                        0436
                        0436  MTH$DATAN_R7D:                                  ; for local use only!
          50   52   66  0436          DIVD    R2, R0
                        0439  MTH$DATAN_R7::                                  ; Special DATAN routine
               50   53  0439          TSTF    R0                              ; R6 = X = argument
               76   19  043B          BLSS    NEG_ARG                         ; Branch to negative argument logic
                        043D  ;
                        043D  ; Argument is positive
                        043D  ;
56   50   3EC0 8F   A3  043D          SUBW3   #^X3EC0, R0, R6                 ; Argument is less than 3/32,
               47   19  0443          BLSS    SMALL                           ;  branch to small argument logic
     56   036F 8F   B1  0445          CMPW    #^X036F, R6                     ; Argument is greater that 11,
               43   19  044A          BLSS    LARGE_ARG                       ;  branch to large argument logic
                        044C  ;
                        044C  ; Logic for positive medium sized arguments.  Get pointer into DATAN_TABLE.
                        044C  ;
  56   56     FC 8F   9C  044C          ROTL    #-4, R6, R6                     ; R6 = index into MTH$$AB_ATAN table
   56   FFFFFF00 8F   CA  0451          BICL    #-256, R6                       ; zero high order bits of index
56   00000000'GF46    90  0458          MOVB    G^MTH$$AB_ATAN[R6], R6          ; R6 = offset into DATAN_TABLE
     56   FB9B CF46   7E  0460          MOVAQ   DATAN_TABLE[R6], R6             ; R6 = pointer to XHI
                        0466  ;
                        0466  ; Compute Z
                        0466  ;
          52   86   7D  0466          MOVQ    (R6)+, R2                       ; R2 = XHI
     54   50   52   65  0469          MULD3   R2, R0, R4                      ; R4 = X*XHI
          54   08   60  046D          ADDD    #1, R4                          ; R4 = 1 + X*XHI
          50   52   62  0470          SUBD    R2, R0                          ; R0 = X - XHI
          50   54   66  0473          DIVD    R4, R0                          ; R0 = Z = (X - XHI)/(1 + X*XHI)
                        0476  ;
                        0476  ; Evaluate Z*P(Z**2)
                        0476  ;
          7E   50   7D  0476          MOVQ    R0, -(SP)                       ; Push Z onto the stack
          50   50   64  0479          MULD    R0, R0                          ; R0 = Z**2
FCCE CF   06   50   75  047C          POLYD   R0, #DATANLEN1-1, DATANTAB1
                        0482                                                  ; R0 = P(Z**2)
          50   8E   64  0482          MULD    (SP)+, R0                       ; R0 = DATAN(Z) = Z*P(Z**2)
          50   86   60  0485          ADDD    (R6)+, R0                       ; R0 = DATAN_XHI_LO + DATAN(Z)
          50   66   60  0488          ADDD    (R6), R0                        ; R0 = DATAN(X) = DATAN_XHI_HI +
```

```
                                   048B                                                   ;         (DATAN_XHI_LO + DATAN(Z))
                               05  048B                     RSB                           ; Return
                                   048C
                                   048C
                    0098       31  048C  SMALL:    BRW      SMALL_ARG                     ; Dummy label used to avoid adding
                                   048F                                                   ;   an extra insrtuction in the
                                   048F                                                   ;   medium argument logic
                                   048F  ;
                                   048F  ; Large positive argument logic.
                                   048F  ;
                                   048F
                                   048F  LARGE_ARG:
56    00000000 0000C080 8F     50  67  048F         DIVD3    R0, #-1, R6                   ; R6 = -W = -1/X
                  50    56     56  65  049B         MULD3    R6, R6, R0                    ; R0 = W**2
             FCAB CF    06     50  75  049F         POLYD    R0, #DATANLEN1-1, DATANTAB1
                                   04A5                                                   ; R0 = P(W**2)
                  50    56     64  04A5             MULD     R6, R0                        ; R0 = DATAN(W) = -W*P(W**2)
             50   FD34 CF      60  04A8             ADDD     D_PI_OVER_2_LO, R0            ;
             50   FD27 CF      60  04AD             ADDD     D_PI_OVER_2_HI, R0            ; R0 = DATAN(X) = PI/2 - DATAN(W)
                               05  04B2             RSB                                    ; Return
                                   04B3
                                   04B3  ;
                                   04B3  ; Logic for negative arguments
                                   04B3  ;
                                   04B3
                                   04B3  NEG_ARG:
          56    50   BEC0 8F   A3  04B3             SUBW3    #^XBEC0, R0, R6               ; Argument is less than 3/32,
                           6C  19  04B9             BLSS     SMALL_ARG                     ;  branch to small argument logic
             56      036F 8F   B1  04BB             CMPW     #^X036F, R6                   ; Argument is greater than 11,
                           41  19  04C0             BLSS     N_LARGE_ARG                   ;  branch to large argument logic
                                   04C2  ;
                                   04C2  ; Logic for negative medium sized arguments.  Get index into DATAN_TABLE.
                                   04C2  ;
        56    56     FC 8F     9C  04C2             ROTL     #-4, R6, R6                   ; R6 = index into MTH$$AB_ATAN table
         56    FFFFFF00 8F     CA  04C7             BICL     #-256, R6                     ; clear high order (unused) bits of ind
      56    00000000'GF46      90  04CE             MOVB     G^MTH$$AB_ATAN[R6], R6        ; R6 = offset into DATAN_TABLE
           56     FB25 CF46    7E  04D6             MOVAQ    DATAN_TABLE[R6], R6           ; R6 = pointer to XHI
                                   04DC  ;
                                   04DC  ; Compute Z
                                   04DC  ;
                  52    86     7D  04DC             MOVQ     (R6)+, R2                     ; R2 = XHI
             54   50    52     65  04DF             MULD3    R2, R0, R4                    ; R4 = X*XHI
             54   08    54     63  04E3             SUBD3    R4, #1, R4                    ; R4 = 1 - X*XHI = 1 + X*(-XHI)
                  50    52     60  04E7             ADDD     R2, R0                        ; R0 = X + XHI = X - (-XHI)
                  50    54     66  04EA             DIVD     R4, R0                        ; R0 = Z
                                   04ED  ;
                                   04ED  ; Evaluate Z*P(Z**2)
                                   04ED  ;
                  7E    50     7D  04ED             MOVQ     R0, -(SP)                     ; Push Z onto the stack
                  50    50     64  04F0             MULD     R0, R0                        ; R0 = Z**2
             FC57 CF    06     50  75  04F3         POLYD    R0, #DATANLEN1-1, DATANTAB1
                                   04F9                                                   ; R0 = P(Z**2)
                  50    8E     64  04F9             MULD     (SP)+, R0                     ; R0 = DATAN(Z) = Z*P(Z**2)
                  50    86     62  04FC             SUBD     (R6)+, R0                     ; R0 = DATAN_XHI_LO + DATAN(Z)
                  50    66     62  04FF             SUBD     (R6), R0                      ; R0 = DATAN(X) = DATAN_XHI_HI +
                                   0502                                                   ;         (DATAN_XHI_LO + DATAN(Z))
                               05  0502             RSB                                    ; Return
```

```
                                      0503  ;
                                      0503  ; Logic for large negative  arguments
                                      0503  ;
                                      0503
                                      0503  N_LARGE_ARG:
56   00000000 0000C080 8F   50   67   0503          DIVD3   R0, #-1, R6              ; R6 = W = 1/!X!
                   50  56   56   65   050F          MULD3   R6, R6, R0               ; R0 = W**2
              FC37 CF  06   50   75   0513          POLYD   R0, #DATANLEN1-1, DATANTAB1
                                      0519                                           ; R0 = P(W**2)
                       50   56   64   0519          MULD    R6, R0                   ; R0 = DATAN(W) = W*P(W**2)
                   50  FCC0 CF   62   051C          SUBD    D_PI_OVER_2_LO, R0       ;
                   50  FCB3 CF   62   0521          SUBD    D_PI_OVER_2_HI, R0       ; R0 = DATAN(X) = DATAN(W) - PI/2
                                 05   0526          RSB                              ; Return
                                      0527
                                      0527  ;
                                      0527  ; Small argument logic.
                                      0527  ;
                                      0527
                                      0527  SMALL_ARG:
                       56   50   7D   0527          MOVQ    R0, R6                   ; R6 = argument = X
                   50  8000 8F   AA   052A          BICW    #^X8000, R0              ; R0 = !X!
                   50  3280 8F   B1   052F          CMPW    #^X3280, R0              ; Compare 2^-28 to !X!
                            04   19   0534          BLSS    1$                       ; Branch to Polyinomial evaluation
                       50   56   7D   0536          MOVQ    R6, R0                   ; Return with answer equal to argument
                                 05   0539          RSB                              ;
                                      053A
                       50   50   64   053A  1$:     MULD    R0, R0                   ; R0 = X**2
              FC45 CF  06   50   75   053D          POLYD   R0, #DATANLEN2-1, DATANTAB2
                                      0543                                           ; R0 = Q(X**2)
                       50   56   64   0543          MULD    R6, R0                   ; R0 = X*Q(X**2)
                       50   56   60   0546          ADDD    R6, R0                   ; R0 = DATAN(X) = X + X*Q(X**2)
                                 05   0549          RSB                              ; Return
                                      054A
```

```
          054A          .SBTTL  MTH$DATAND - Standard Single Precision Floating Arc Tangent
          054A
          054A
          054A    ;++
          054A    ; FUNCTIONAL DESCRIPTION:
          054A    ;
          054A    ; DATAND - double precision floating point function
          054A    ;
          054A    ; DATAN is computed using the following steps:
          054A    ;
          054A    ;     1. If X > 11 then
          054A    ;        a. Let W = 1/X.
          054A    ;        b. Compute DATAN(W) = W*P(W**2), where P is a polynomial of
          054A    ;           degree 6.
          054A    ;        c. Set DATAN(X) = pi/2 - DATAN(W)
          054A    ;     2. If 3/32 =< X =< 11 then
          054A    ;        a. Obtain XHI by table look-up.
          054A    ;        b. Compute Z = (X - XHI)/(1 + X*XHI).
          054A    ;        c. Compute DATAN(Z) = Z*P(Z**2), where P is a polynomial of
          054A    ;           degree 6.
          054A    ;        d. Obtain DATAN(XHI) by table look-up.  DATAN(XHI) will have
          054A    ;           two parts - the high order bits, DATAN_XHI_HI, and the low
          054A    ;           order bits, DATAN_XHI_LO.
          054A    ;        e. Compute DATAN(X) = DATAN_XHI_HI + (DATAN_XHI_LO + DATAN(Z)).
          054A    ;     3. If 0 =< X < 3/32 then
          054A    ;        a. Compute DATAN(X) = X + X*Q(X**2), where Q is a polynomial
          054A    ;           of degree 6.
          054A    ;     4. If X < 0 then
          054A    ;        a. Compute Y = DATAN(|X|) using steps 1 to 3.
          054A    ;        b. Set DATAN(X) = -Y.
          054A    ;
          054A    ; CALLING SEQUENCE:
          054A    ;
          054A    ;       Arctangent.wd.v = MTH$DATAND(x.rd.r)
          054A    ;
          054A    ; INPUT PARAMETERS:
          054A
00000004  054A          LONG = 4                        ; define longword multiplier
00000004  054A          x = 1 * LONG                    ; x is an angle in radians
          054A    ;
          054A    ; IMPLICIT INPUTS:      none
          054A    ;
          054A    ; OUTPUT PARAMETERS:
          054A    ;
          054A    ;       VALUE:  double precision floating arctangent angle of the argument
          054A    ;
          054A    ; IMPLICIT OUTPUTS:     none
          054A    ;
          054A    ; SIDE EFFECTS:
          054A    ;
          054A    ; Signals:      none
          054A    ;
          054A    ; NOTE: This  procedure disables floating point underflow, enable integer
          054A    ;    overflow, causes no floating overflow or other arithmetic traps, and
          054A    ;    preserves enables across the call.
          054A    ;
          054A    ;---
```

```
                            054A   730
                            054A   731
                     40FC   054A   732          .ENTRY  MTH$DATAND, ACMASK        ; standard call-by-reference entry
                            054C   733                                            ; disable DV (and FU), enable IV
                            054C   734          MTH$FLAG_JACKET                   ; flag that this is a jacket procedure
                            054C
       6D   00000000'GF  9E 054C                MOVAB   G^MTH$$JACKET_HND, (FP)
                            0553                                                  ; set handler address to jacket
                            0553                                                  ; handler
                            0553
                            0553   735                                            ; in case of an error in special JSB
                            0553   736                                            ; routine
            50    04 BC  70 0553   737          MOVD    @x(AP), R0                ; R0/R1 = arg
                      6A 10 0557   738          BSBB    MTH$DATAND_R7             ; call special DATAND rountine
                         04 0559   739          RET                               ; return - result in R0
                            055A   740
```

```
                          055A            .SBTTL  MTH$DATAND2 - Standard Double Floating Arctangent With 2 Arguments
                          055A      ;++
                          055A      ; FUNCTIONAL DESCRIPTION:
                          055A      ;
                          055A      ; DATAND2  - double precision floating point function
                          055A      ;
                          055A      ; DATAND2(X,Y) is computed as following:
                          055A      ;
                          055A      ;       If Y = 0 or X/Y > 2**57, DATAND2(X,Y) = 90 * (sign X)
                          055A      ;       If Y > 0 and X/Y =< 2**57, DATAND2(X,Y) = DATAND(X/Y)
                          055A      ;       If Y < 0 and X/Y =< 2**57, DATAND2(X,Y) = 180 * (sign X) + DATAND(X/Y)
                          055A      ;
                          055A      ;
                          055A      ; CALLING SEQUENCE:
                          055A      ;
                          055A      ;       Arctangent2.wd.v = MTH$DATAND2(x.rd.r, y.rd.r)
                          055A      ;
                          055A      ; INPUT PARAMETERS:
                          055A
              00000004    055A              x = 1 * LONG                    ; x is the first argument
              00000008    055A              y = 2 * LONG                    ; y is the second argument
                          055A
                          055A      ; SIDE EFFECTS: See description of MTH$DATAND
                          055A      ;
                          055A      ;--
                          055A
                          055A
                  40FC    055A              .ENTRY  MTH$DATAND2, ACMASK     ; standard call-by-reference entry
                          055C                                              ; disable DV (and FU), enable IV
                          055C              MTH$FLAG_JACKET                 ; flag that this is a jacket procedure
                          055C
   6D   00000000'GF   9E  055C              MOVAB   G^MTH$$JACKET_HND, (FP)
                          0563                                              ; set handler address to jacket
                          0563                                              ; handler
                          0563
                          0563                                              ; in case of an error in special JSB
                          0563                                              ; routine
        50    04 BC   70  0563              MOVD    @x(AP), R0              ; R0/R1 = arg1
        52    08 BC   70  0567              MOVD    @y(AP), R2              ; R2/R3 = arg2
                          056B      ;
                          056B      ; Test if Y = 0 or X/Y > 2**57
                          056B      ;
              31      13  056B              BEQL    INF_DEG                 ; branch to INF_DEG if Y = 0
   54   50   807F 8F  AB  056D              BICW3   #^X807F, R0, R4         ; R4 = exponent(X)
   55   52   807F 8F  AB  0573              BICW3   #^X807F, R2, R5         ; R5 = exponent(Y)
              54      55  A2  0579          SUBW    R5, R4                  ; R4 = exponent(X) - exponent(Y)
        1D00 8F   54  B1  057C              CMPW    R4, #58*128             ; compare R4 with 58
              1B      14  0581              BGTR    INF_DEG                 ; if X/Y > 2**57, branch to INF_DEG
                          0583      ;
                          0583      ; Test if Y > 0 or Y < 0
                          0583      ;
              52      B5  0583              TSTW    R2                      ; test the sign of Y
              14      14  0585              BGTR    A2PLUSD                 ; branch to A2PLUSD if Y > 0
              50      B5  0587              TSTW    R0                      ; test the sign of X
              08      18  0589              BGEQ    A1PLUSD                 ; branch to A1PLUSD if X >= 0
                          058B      ;
                          058B      ; Y < 0 and X < 0 and X/Y =< 2**57
```

```
                           058B  ;
               33    10    058B               BSBB    MTH$DATAND_R7D          ; R0/R1 = DATAND(X/Y)
      50    FE27 CF  62    058D               SUBD    D_180, R0               ; R0/R1 = -180 + DATAND(X/Y)
                     04    0592               RET                             ; return
                           0593  ;
                           0593  ; Y < 0 and X > 0 and X/Y =< 2**57
                           0593  ;
                           0593  A1PLUSD:
               2B    10    0593               BSBB    MTH$DATAND_R7D          ; R0/R1 = DATAND(X/Y)
      50    FE1F CF  60    0595               ADDD    D_180, R0               ; R0/R1 = 180 + DATAND(X/Y)
                     04    059A               RET                             ; return
                           059B  ;
                           059B  ; Y > 0 and X/Y =< 2**57
                           059B  ;
                           059B  A2PLUSD:
               23    10    059B               BSBB    MTH$DATAND_R7D          ; R0/R1 = DATAND(X/Y)
                     04    059D               RET                             ; return
                           059E  ;
                           059E  ; Y = 0 or X/Y > 2**57
                           059E  ;
                           059E  INF_DEG:
               50    B5    059E               TSTW    R0                      ; test the sign of X
               08    14    05A0               BGTR    1$                      ; branch if X > 0
               0C    13    05A2               BEQL    2$                      ; branch if X = 0
      50    FE08 CF  70    05A4               MOVD    D_M90, R0       ; R0/R1 = DATAND(X/Y) = -90
                     04    05A9               RET                             ; return
                           05AA
      50    FDFA CF  70    05AA  1$:          MOVD    D_90, R0                ; R0/R1 = DATAND(X/Y) = 90
                     04    05AF               RET                             ; return
                           05B0
                           05B0  ;+
                           05B0  ; Here if both X = 0 and Y = 0.  Signal INVALID ARG TO MATH LIBRARY
                           05B0  ;-
                           05B0
      50    01   0F  79    05B0  2$:          ASHQ    #15, #1, R0             ; R0/R1 = reserved operand, co180ed
                           05B4                                               ; to CHF$L_MCH_SAVR0/R1 so handlers
                           05B4                                               ; can change if they want to continue.
       7E    00'8F   9A    05B4               MOVZBL  #MTH$K_INVARGMAT, -(SP) ; code for INVALID ARG TO MATH LIBRARY
 00000000'GF    01   FB    05B8               CALLS   #1, G^MTH$$SIGNAL       ; Signal SEVERE error
                     04    05BF               RET                             ; return if a handler says SS$_CONTINUE
```

```
                              05C0        .SBTTL  MTH$DATAND_R7 - Special DATAND routine
                              05C0
                              05C0  ; Special DATAND - used by the standard routine, and directly.
                              05C0  ;
                              05C0  ; CALLING SEQUENCES:
                              05C0  ;       save anything needed in R0:R7
                              05C0  ;       MOVD    ..., R0                         ; input in R0/R1
                              05C0  ;       JSB     MTH$DATAND_R7
                              05C0  ;       return with result in R0/R1
                              05C0  ; Note: This routine is written to avoid causing any  integer  overflows,
                              05C0  ;   floating overflows, or floating underflows or divide by 0 conditions,
                              05C0  ;   whether enabled or not.
                              05C0  ;
                              05C0  ; REGISTERS USED:
                              05C0  ;       R0/R1 - Floating argument then result
                              05C0  ;       R0:R5 - POLYD
                              05C0  ;       R6    - Pointer into DATAND_TABLE
                              05C0  ;       R6/R7 - Y during POLYD
                              05C0
                              05C0
                              05C0 MTH$DATAND_R7D:                               ; for local use only!
            50    52    66    05C0        DIVD    R2, R0
                              05C3 MTH$DATAND_R7::                               ; Special DATAND routine
                  50    53    05C3        TSTF    R0                            ; R6 = X = argument
                  71    19    05C5        BLSS    NEG_ARGD                      ; Branch to negative argument logic
                              05C7  ;
                              05C7  ; Argument is positive
                              05C7  ;
  56    50    3EC0 8F    A3   05C7        SUBW3   #^X3EC0, R0, R6               ; Argument is less than 3/32,
                  47    19    05CD        BLSS    SMALLD                        ;  branch to small argument logic
        56    036F 8F    B1   05CF        CMPW    #^X036F, R6                   ; Argument is greater that 11,
                  43    19    05D4        BLSS    LARGE_ARGD                    ;  branch to large argument logic
                              05D6  ;
                              05D6  ; Logic for positive medium sized arguments.  Get pointer into DATAND_TABLE.
                              05D6  ;
   56    56    FC 8F     9C   05D6        ROTL    #-4, R6, R6                   ; R6 = index into AB_ATAN table
  56    FFFFFF00 8F      CA   05DB        BICL    #-256, R6                     ; zero high order bits of index
 56   00000000'GF46      90   05E2        MOVB    G^MTH$$AB_ATAN[R6], R6        ; R6 = offset into DATAND_TABLE
     56    FBF9 CF46     7E   05EA        MOVAQ   DATAND_TABLE[R6], R6          ; R6 = pointer to XHI
                              05F0  ;
                              05F0  ; Compute z
                              05F0  ;
            52    86    7D    05F0        MOVQ    (R6)+, R2                     ; R2 = XHI
      54    50    52    65    05F3        MULD3   R2, R0, R4                    ; R4 = X*XHI
            54    08    60    05F7        ADDD    #1, R4                        ; R4 = 1 + X*XHI
            50    52    62    05FA        SUBD    R2, R0                        ; R0 = X - XHI
            50    54    66    05FD        DIVD    R4, R0                        ; R0 = Z = (X - XHI)/(1 + X*XHI)
                              0600  ;
                              0600  ; Evaluate Z*P(Z**2)
                              0600  ;
            7E    50    7D    0600        MOVQ    R0, -(SP)                     ; Push Z onto the stack
            50    50    64    0603        MULD    R0, R0                        ; R0 = Z**2
 FD2C CF    06    50    75    0606        POLYD   R0, #DATANDLEN1-1, DATANDTAB1
                              060C                                              ; R0 = P(Z**2)
            50    8E    64    060C        MULD    (SP)+, R0                     ; R0 = DATAND(Z) = Z*Q(Z**2)
            50    86    60    060F        ADDD    (R6)+, R0                     ; R0 = DATAND_XHI_LO + DATAND(Z)
            50    66    60    0612        ADDD    (R6), R0                      ; R0 = DATAND(X) = DATAND_XHI_HI +
```

```
                                    0615                                                ;        (DATAND_XHI_LO + DATAND(Z))
                                05  0615                   RSB                          ; Return
                                    0616
                                    0616
                    008E        31  0616  SMALLD:  BRW     SMALL_ARGD                   ; Dummy label used to avoid adding
                                    0619                                                ;    an extra insrtuction in the
                                    0619                                                ;    medium argument logic
                                    0619  ;
                                    0619  ; Large positive argument logic.
                                    0619  ;
                                    0619
                                    0619  LARGE_ARGD:
56   00000000 0000C080 8F    50     67  0619           DIVD3   R0, #-1, R6              ; R6 = -W = -1/X
           50     56   56           65  0625           MULD3   R6, R6, R0               ; R0 = W**2
      FD09 CF     06   50           75  0629           POLYD   R0, #DATANDLEN1-1, DATANDTAB1
                                    062F                                                ; R0 = P(W**2)
           50     56                64  062F           MULD    R6, R0                   ; R0 = -DATAND(Z) = -Z*P(W**2)
           50   FD72 CF             60  0632           ADDD    D_90, R0                 ; R0 = DATAND(X) = 90 - DATAND(Z)
                                    05  0637           RSB                              ; Return
                                    0638
                                    0638  ;
                                    0638  ; Logic for negative arguments
                                    0638  ;
                                    0638
                                    0638  NEG_ARGD:
     56   50   BEC0 8F              A3  0638           SUBW3   #^XBEC0, R0, R6          ; Argument is less than 3/32,
                        67          19  063E           BLSS    SMALL_ARGD               ;  branch to small argument logic
          56   036F 8F              B1  0640           CMPW    #^X036F, R6              ; Argument is greater than 11,
                        41          19  0645           BLSS    N_LARGE_ARGD             ;  branch to large argument logic
                                    0647  ;
                                    0647  ; Logic for negative medium sized arguments.  Get index into DATAND_TABLE.
                                    0647  ;
      56   56     FC 8F             9C  0647           ROTL    #-4, R6, R6              ; R6 = index into MTH$$AB_ATAN table
     56    FFFFFF00 8F              CA  064C           BICL    #-256, R6                ; clear high order (unused) bits of ind
  56    00000000'GF46               90  0653           MOVB    G^MTH$$AB_ATAN[R6], R6   ; R6 = offset into DATAN_TABLE
        56   FB88 CF46              7E  065B           MOVAQ   DATAND_TABLE[R6], R6     ; R6 = pointer to XHI
                                    0661  ;
                                    0661  ; Compute Z
                                    0661  ;
                52     86           7D  0661           MOVQ    (R6)+, R2                ; R2 = XHI
          54    50     52           65  0664           MULD3   R2, R0, R4               ; R4 = X*XHI
          54    08     54           63  0668           SUBD3   R4, #1, R4               ; R4 = 1 - X*XHI = 1 + X*(-XHI)
                50     52           60  066C           ADDD    R2, R0                   ; R0 = X + XHI = X - (-XHI)
                50     54           66  066F           DIVD    R4, R0                   ; R0 = Z
                                    0672  ;
                                    0672  ; Evaluate Z*P(Z**2)
                                    0672  ;
                7E     50           7D  0672           MOVQ    R0, -(SP)                ; Push Z onto the stack
                50     50           64  0675           MULD    R0, R0                   ; R0 = Z**2
      FCBA CF   06     50           75  0678           POLYD   R0, #DATANDLEN1-1, DATANDTAB1
                                    067E                                                ; R0 = P(Z**2)
                50     8E           64  067E           MULD    (SP)+, R0                ; R0 = DATAND(Z) = Z*P(Z**2)
                50     86           62  0681           SUBD    (R6)+, R0                ; R0 = DATAND_XHI_LO + DATAND(Z)
                50     66           62  0684           SUBD    (R6), R0                 ; R0 = DATAND(X) = DATAND_XHI_HI +
                                    0687                                                ;        (DATAND_XHI_LO + DATAND(Z))
                                05  0687           RSB                              ; Return
                                    0688  ;
```

```
                                      0688   949 ; Logic for large negative  arguments
                                      0688   950 ;
                                      0688   951
                                      0688   952 N_LARGE_ARGD:
56  00000000 0000C080 8F    50    67  0688   953         DIVD3   R0, #-1, R6                 ; R6 = W = 1/|X|
                      50    56    65  0694   954         MULD3   R6, R6, R0                  ; R0 = W**2
                 FC9A CF    06    75  0698   955         POLYD   R0, #DATANDLEN1-1, DATANDTAB1
                                      069E   956                                             ; R0 = P(W**2)
                      50    56    64  069E   957         MULD    R6, R0                      ; R0 = DATAND(W) = W*P(W**2)
               50   FD03 CF       62  06A1   958         SUBD    D_90, R0                    ; R0 = DATAND(X) = DATAND(W) - 90
                                  05  06A6   959         RSB                                 ; Return
                                      06A7   960
                                      06A7   961 ;
                                      06A7   962 ; Small argument logic.
                                      06A7   963 ;
                                      06A7   964
                                      06A7   965 SMALL_ARGD:
                      56    50    70  06A7   966         MOVD    R0, R6                      ; R6 = argument = X
                            28    13  06AA   967         BEQL    3$                          ;
               50   8000 8F       AA  06AC   968         BICW    #^X8000, R0                 ; R0 = |X|
               50   3280 8F       B1  06B1   969         CMPW    #^X3280, R0                 ; Compare 2^-28 to |X|
                            08    19  06B6   970         BLSS    1$                          ; Branch to Polyinomial evaluation
          50   56   FCE4 CF       65  06B8   971         MULD3   D_PI_OV_180_M_64, R6 ,R0; R0 = X*(pi/180 - 64)
                            0C    11  06BE   972         BRB 2$                              ;
                      50    50    64  06C0   973 1$:     MULD    R0, R0                      ; R0 = X**2
            FCA7 CF   06    50    75  06C3   974         POLYD   R0, #DATANDLEN2-1, DATANDTAB2
                                      06C9   975                                             ; R0 = Q(X**2)
                      50    56    64  06C9   976         MULD    R6, R0                      ; R0 = X*Q(X**2)
               56   0300 8F       A0  06CC   977 2$:     ADDW    #^X300, R6                  ; R6 = X*2**6
                      50    56    60  06D1   978         ADDD    R6, R0                      ; R0 = DATAND(X) = X*2**6 + X*Q(X**2)
                                  05  06D4   979 3$:     RSB                                 ; Return
                                      06D5   980
                                      06D5   981
                                      06D5   982         .END
```

```
A1PLUS                  00000409 R      01
A1PLUSD                 00000593 R      01
A2PLUS                  00000411 R      01
A2PLUSD                 0000059B R      01
ACMASK                = 000040FC
DATANDLEN1            = 00000007
DATANDLEN2            = 00000007
DATANDTAB1              00000338 R      01
DATANDTAB2              00000370 R      01
DATAND_TABLE            000001E8 R      01
DATANLEN1             = 00000007
DATANLEN2             = 00000007
DATANTAB1               00000150 R      01
DATANTAB2               00000188 R      01
DATAN_TABLE             00000000 R      01
D_180                   000003B8 R      01
D_90                    000003A8 R      01
D_M90                   000003B0 R      01
D_MPI_OVER_2            000001D0 R      01
D_PI                    000001C0 R      01
D_PI_OVER_2             000001C8 R      01
D_PI_OVER_2_HI          000001D8 R      01
D_PI_OVER_2_LO          000001E0 R      01
D_PI_OV_180_M_64        000003A0 R      01
INF                     00000414 R      01
INF_DEG                 0000059E R      01
LARGE_ARG               0000048F R      01
LARGE_ARGD              00000619 R      01
LONG                  = 00000004
MTH$$AB_ATAN            ********    X   00
MTH$$JACKET_HND         ********    X   01
MTH$$SIGNAL             ********    X   00
MTH$DATAN               000003C0 RG     01
MTH$DATAN2              000003D0 RG     01
MTH$DATAND              0000054A RG     01
MTH$DATAND2             0000055A RG     01
MTH$DATAND_R7           000005C3 RG     01
MTH$DATAND_R7D          000005C0 R      01
MTH$DATAN_R7            00000439 RG     01
MTH$DATAN_R7D           00000436 R      01
MTH$K_INVARGMAT         ********    X   00
NEG_ARG                 000004B3 R      01
NEG_ARGD                00000638 R      01
N_LARGE_ARG             00000503 R      01
N_LARGE_ARGD            00000688 R      01
SMALL                   0000048C R      01
SMALLD                  00000616 R      01
SMALL_ARG               00000527 R      01
SMALL_ARGD              000006A7 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                       000006D5  (  1749.)  01 (   1.)    PIC    USR    CON    REL    LCL   SHR   EXE   RD   NOWRT NOVEC LONG

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

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     30    00:00:00.10    00:00:01.09
Command processing                152    00:00:00.70    00:00:03.81
Pass 1                            115    00:00:02.50    00:00:07.34
Symbol table sort                   0    00:00:00.02    00:00:00.20
Pass 2                            178    00:00:02.11    00:00:06.41
Symbol table output                 8    00:00:00.05    00:00:00.08
Psect synopsis output               2    00:00:00.02    00:00:00.03
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals              487    00:00:05.51    00:00:19.18

The working set limit was 1050 pages.
16195 bytes (32 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 51 non-local and 8 local symbols.
1042 source lines were read in Pass 1, producing 22 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$:MTHDATAN/OBJ=OBJ$:MTHDATAN MSRC$:MTHJACKET/UPDATE=(ENH$:MTHJACKET)+MSRC
```

0258 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MTHCVTDG LIS

MTHDACOS LIS

MTHCGABS LIS

MTHCDSINC LIS

MTHCLOG LIS

MTHDASIN LIS

MTHCGLOG LIS

MTHCONVER LIS

MTHDATAN LIS

MTHCSQRT LIS

MTHCEXP LIS

MTHCGSQRT LIS

MTHCGEXP LIS

MTHCSINCO LIS

MTHCONJG LIS

MTHCDSQRT LIS

MTHCGSINC LIS

MTHCOSH LIS

0259 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MTHDCOSH
LIS

MTHDMIN1
LIS

MTHDLOG
LIS

MTHDSINCO
LIS

MTHDATANH
LIS

MTHDNINT
LIS

MTHDSQRT
LIS

MTHDCONJG
LIS

MTHDINT
LIS

MTHDMAX1
LIS

MTHDSIGN
LIS

MTHDIM
LIS

MTHDMOD
LIS

MTHDSINH
LIS

MTHDEXP
LIS

MTHDFLOOR
LIS

MTHDPROD
LIS