```
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**MTHDATANH

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

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
```

MTH
VAX

Mac

_$

0 (

The

MA(

Table of contents

```
0000          .TITLE  MTH$DATANH        ; Double Precision Hyperbolic Arctangent
0000          .IDENT /2-003/            ; File: MTHDATANH.MAR  Edit: PDG2003
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$DATANH returns the double precision hyperbolic arctangent of the
0000    ; double precision argument.  The call is standard call-by-reference.
0000    ;
0000    ;--
0000    ;
0000    ; VERSION: 2
0000    ;
0000    ; HISTORY:
0000    ; AUTHOR:
0000    ;       Peter D Gilbert, 23-Jul-81: Version 2
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;
```

```
0000          .SBTTL  HISTORY ; Detailed Current Edit History
0000
0000   ; VERSION 1
0000   ;
0000   ; 1-001 - Original from PL/I math library
0000   ;
0000   ; Edit History for Version 02 of MTH$DATANH
0000   ;
0000   ; 2-000   Original       July 1981
0000   ; 2-001 - Change MOVZBL to CVTBL when accessing MTH$$AB_ALOG_V.  PDG 2-Dec-1981
0000   ; 2-002 - Change RSB to RET after error exit.  PDG 6-Jan-1981
0000   ; 2-003 - Repair problem with POLY instruction.  PDG 19-Mar-1982
0000   ;
```

```
          0000          .SBTTL  DECLARATIONS    ; Declarative Part of Module
          0000
          0000   ;
          0000   ; INCLUDE FILES:        MTHJACKET.MAR
          0000   ;
          0000
          0000   ;
          0000   ; EXTERNAL SYMBOLS:
          0000   ;
          0000          .DSABL  GLOBAL
          0000          .SHOW   BINARY,CALLS,CONDITIONALS,DEFINITIONS,EXPANSIONS
          0000          .EXTRN  MTH$K_INVARGMAT
          0000          .EXTRN  MTH$$SIGNAL
          0000          .EXTRN  MTH$$AB_ALOG_V
          0000          .EXTRN  MTH$$AB_D_FHI
          0000
          0000   ;
          0000   ; EQUATED SYMBOLS:
          0000   ;
          0000
          0000   ;
          0000   ; MACROS:
          0000   ;
          0000
          0000          .MACRO  OPDEF   X, OP, SH
          0000          .OPDEF  ADDX    ^X00@SH+OP,R'X,M'X
          0000          .OPDEF  ADDX3   ^X01@SH+OP,R'X,R'X,W'X
          0000          .OPDEF  SUBX    ^X02@SH+OP,R'X,M'X
          0000          .OPDEF  SUBX3   ^X03@SH+OP,R'X,R'X,W'X
          0000          .OPDEF  MULX    ^X04@SH+OP,R'X,M'X
          0000          .OPDEF  MULX3   ^X05@SH+OP,R'X,R'X,W'X
          0000          .OPDEF  DIVX    ^X06@SH+OP,R'X,M'X
          0000          .OPDEF  DIVX3   ^X07@SH+OP,R'X,R'X,W'X
          0000          .OPDEF  CVTWX   ^X0D@SH+OP,RW,W'X
          0000          .OPDEF  POLYX   ^X15@SH+OP,R'X,RW,AB
          0000          .OPDEF  MOVX    ^X007D,RQ,WQ              ; MOVQ
          0000          .OPDEF  MOVAX   ^X007E,AQ,WL              ; MOVAQ
          0000          .ENDM
          0000
          0000          OPDEF   D, <^X0060>, 0
          0000          .OPDEF  ADDX    ^X00@0+^X0060,RD,MD
          0000          .OPDEF  ADDX3   ^X01@0+^X0060,RD,RD,WD
          0000          .OPDEF  SUBX    ^X02@0+^X0060,RD,MD
          0000          .OPDEF  SUBX3   ^X03@0+^X0060,RD,RD,WD
          0000          .OPDEF  MULX    ^X04@0+^X0060,RD,MD
          0000          .OPDEF  MULX3   ^X05@0+^X0060,RD,RD,WD
          0000          .OPDEF  DIVX    ^X06@0+^X0060,RD,MD
          0000          .OPDEF  DIVX3   ^X07@0+^X0060,RD,RD,WD
          0000          .OPDEF  CVTWX   ^X0D@0+^X0060,RW,WD
          0000          .OPDEF  POLYX   ^X15@0+^X0060,RD,RW,AB
          0000          .OPDEF  MOVX    ^X007D,RQ,WQ              ; MOVQ
          0000          .OPDEF  MOVAX   ^X007E,AQ,WL              ; MOVAQ
          0000
          0000
00000007  0000          F_EXP = 7       ; Bit offset to exponent
00000007  0000          X_EXP = 7       ; Bit offset to exponent
          0000
```

```
    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  ; CONSTANTS:
                            0000  ;
                            0000
                  000001FC  0000        ACMASK = ^M<R2,R3,R4,R5,R6,R7,R8>
                            0000                                        ; register entry mask and integer
                            0000                                        ; overflow enable
                            0000
                            0000  ;+  The "128" in the constants is used to shift the unbiased exponent
                            0000  ;   right 7 places so that the rightmost bit is at bit 0.
                            0000  ;-
                            0000
7200F7B1 72173CB1           0000  LN2_HI: .QUAD   ^X7200F7B172173CB1    ; (Hi 48 bits of ln2)/128
8ECEAF27 75E62B81           0008  LN2_LO: .QUAD   ^X8ECEAF2775E62B81    ; (Low bits of ln2)/128
                            0010
                            0010  LOGTAB1:                              ; Constants for q(z).  Generated using
                            0010                                        ; eq. 6.3.10 of Hart et. al. (sin(2a)
                            0010                                        ; = 1/32)
1E51DE52 4D00BECD           0010        .QUAD   ^X1E51DE524D00BECD      ; C9 =  -0.10024452856511271
44C1F2BD 0E683EE4           0018        .QUAD   ^X44C1F2BD0E683EE4      ; C8 =   0.11135560980588577
9B9BEC78 FFDBBEFF           0020        .QUAD   ^X9B9BEC78FFDBBEFF      ; C7 =  -0.12499973121073342
4BB28A46 49143F12           0028        .QUAD   ^X4BB28A4649143F12      ; C6 =   0.14285690397225510
6C93AD01 AAAABF2A           0030        .QUAD   ^X6C93AD01AAAABF2A      ; C5 =  -0.16666666680280835
C92CCE8D CCCC3F4C           0038        .QUAD   ^XC92CCE8DCCCC3F4C      ; C4 =   0.20000000010208756
DCE9FFFF FFFFBF7F           0040        .QUAD   ^XDCE9FFFFFFFFBF7F      ; C3 =  -0.24999999999996883
A0A5AAAA AAAA3FAA           0048        .QUAD   ^XA0A5AAAAAAAA3FAA      ; C2 =   0.33333333333331553
00000000 0000C000           0050        .QUAD   ^X000000000000C000      ; C1 =  -0.50000000000000000
                            0058        ;
                            0058        ; Remove this constant, and do another multiply in-line.
                            0058        ;
                            0058  ;;;;;;; .QUAD ^X0000000000000000      ; C0 =   0.00000000000000000
                  00000008  0058  LOGLEN1 = .-LOGTAB1/8 - 1             ; no. of floating point entries
                            0058
                            0058
                            0058  LOGTAB2:                              ; Constants for p(z*z). Generated using
                            0058                                        ; eq. 6.3.11 of Hart et. al. (sin(2a) =
                            0058                                        ; (b - 1)/(b + 1) where b = 2**(1/7))
88B900ED 70B23F3B           0058        .QUAD   ^X88B900ED70B23F3B      ; C5 =  0.18304708605445150
5D2C3E00 8D3C3F63           0060        .QUAD   ^X5D2C3E008D3C3F63      ; C4 =  0.22221845749308248
185BC1CE 49243F92           0068        .QUAD   ^X185BC1CE49243F92      ; C3 =  0.28571429124626553
1CEFCCC4 CCCC3FCC           0070        .QUAD   ^X1CEFCCC4CCCC3FCC      ; C2 =  0.39999999999604962
AB02AAAA AAAA402A           0078        .QUAD   ^XAB02AAAAAAAA402A      ; C1 =  0.66666666666666787
00000000 00004100           0080        .QUAD   ^X0000000000004100      ; C0 =  2.00000000000000000
                  00000005  0088  LOGLEN2 = .-LOGTAB2/8 - 1
```

```
          0088         .SBTTL  MTH$DATANH      - Double Precision Hyperbolic Arctangent
          0088
          0088   ;++
          0088   ; FUNCTIONAL DESCRIPTION:
          0088   ;
          0088   ; DATANH  - Double precision floating point function
          0088   ;
          0088   ; DATANH(X) is computed using the following approximation technique:
          0088   ;
          0088   ;       If |X| >= 1.0, error.  Otherwise
          0088   ;
          0088   ;       Let (1+X)/(1-X) = f * (2**n), where 1/2 <= f < 1
          0088   ;
          0088   ;       If n is greater than or equal to 1 then
          0088   ;           set N = n - 1 and F1 = 2*f.
          0088   ;       Else
          0088   ;           set N = n and F = f.
          0088   ;
          0088   ;       If |F - 1| < 2**-5 then
          0088   ;           2*atanh(X) =  N*ln(2) + W + W*P(W),
          0088   ;               where W = ((1+F)/(1-F))*2**N - 1,
          0088   ;               and P is a polynomial of degree F=5,D=9.
          0088   ;       Else
          0088   ;           Obtain FHI (roughly equal to F) from table lookup.
          0088   ;           2*atanh(X) = ln((1+X)/(1-X)) = N*ln(2) + ln(FHI) + Z+Q(Z*Z),
          0088   ;               where Q is a polynomial of degree F=2,D=5,
          0088   ;               where Z = (F - FHI)/(F + FHI)
          0088   ;               where F = (2**-N)*(1+X)/(1-X)
          0088   ;           Z is computed by:
          0088   ;               Z = (X-D)/(1-X*D)
          0088   ;               where Y = FHI*2**N
          0088   ;               where D = (Y-1)/(Y+1)
          0088   ;           Note that Z may be computed in a variety of ways:
          0088   ;               Z = [(1+X) - Y*(1-X)]/[(1+X) + Y*(1-X)]
          0088   ;               Z = [1 + X - Y + X*Y]/[1 + X + Y - X*Y]
          0088   ;               Z = [1 - Y + X + X*Y]/[1 + Y + X - X*Y]
          0088   ;               Z = [(1-Y) + X*(1+Y)]/[(1+Y) + X*(1-Y)]
          0088   ;
          0088   ;       NOTE:   The quantities ln(A=FHI) and ln2 are used in the above
          0088   ;               equations in two parts - a high part (containing the
          0088   ;               high order bits) and a low part (containing the low
          0088   ;               order bits.  In the code the high and low parts of the
          0088   ;               constants are indicated by a _HI and _LO suffix respec-
          0088   ;               tively.  The values were chosen such that N*LN2_HI +
          0088   ;               LN_FHI_HI is exactly representable.
          0088   ;
          0088   ; CALLING SEQUENCE:
          0088   ;
          0088   ;       datanh.wd.v = MTH$DATANH(x.rd.r)
          0088   ;
          0088   ; INPUT PARAMETERS:
          0088
00000004  0088         x = 4                           ; Contents of x is the argument
          0088
          0088   ; IMPLICIT INPUTS:      none
          0088   ;
          0088   ; OUTPUT PARAMETERS:
```

```
                              0088  ;
                              0088  ;       VALUE:  Double precision hyperbolic arctangent of the argument
                              0088  ;
                              0088  ; IMPLICIT OUTPUTS:     none
                              0088  ;
                              0088  ; COMPLETION CODES:     none
                              0088  ;
                              0088  ; SIDE EFFECTS:
                              0088  ;
                              0088  ; Signals: MTH$K_INVARGMAT if !X! >= 1.0 with reserved operand in R0 (copied to
                              0088  ; the signal mechanism vector CHF$L_MCH_R0/R1 by LIB$SIGNAL).
                              0088  ; Associated message is: "Floating overflow in math library". Result is
                              0088  ; reserved operand -0.0 unless a user supplied (or any) error handler changes
                              0088  ; CHF$L_MCH_R0/R1.
                              0088  ;
                              0088  ; NOTE: This procedure disables floating point underflow and integer
                              0088  ; overflow, causes no floating overflow or other arithmetic traps, and
                              0088  ; preserves enables across the call.
                              0088  ;
                              0088  ; Note: This routine is written to avoid causing any integer overflows,
                              0088  ; floating overflows, or floating underflows or divide by 0 conditions,
                              0088  ; whether enabled or not.
                              0088  ;
                              0088  ;---
                              0088
               0135     31    0088  ERR:    BRW     ERROR
                              008B
                      01FC    008B          .ENTRY  MTH$DATANH, ACMASK       ; standard call-by-reference entry
                              008D                                           ; disable DV (and FU), enable IV
         50      04 BC  7D    008D          MOVX    @X(AP), R0               ; R0 = arg
                              0091
      52     08     50  43    0091          SUBF3   R0, S^#1.0, R2           ; R2 = 1-X (okay to trunc towards 0)
                    F1  15    0095          BLEQ    ERR                      ; ATANH(X) is not defined for X>-1
      56     08     50  41    0097          ADDF3   R0, S^#1.0, R6           ; R6 = 1+X (okay to truc towards 0)
                    EB  15    009B          BLEQ    ERR                      ; ATANH(X) is not defined for X<=-1
             56     52  46    009D          DIVF2   R2, R6                   ; R6 = approximation to (1+X)/(1-X)
   58    00000000'GF    9E    00A0          MOVAB   G^MTH$$AB_ALOG_V, R8
             58     68  C0    00A7          ADDL2   (R8), R8                 ; R8 = address of ALOG table
   55    56   007F 8F   AB    00AA          BICW3   #1@F_EXP-1, R6, R5       ; R5 = Biased exponent
         55   4000 8F   A2    00B0          SUBW    #^X4000, R5              ; R5 = Unbiased exponent
                    64  15    00B5          BLEQ    NEG_EXP                  ; Branch to processing for n=<0
                              00B7
         55   0080 8F   A2    00B7          SUBW    #1@F_EXP, R5             ; Exponent is positive, R5 = N = n - 1
             56     55  A2    00BC          SUBW    R5, R6                   ; R6 = F = 2f
             56     56  9A    00BF          MOVZBL  R6, R6                   ; R6 = index into ALOG table
              00000000        00C2          .IF     NE, F_EXP-X_EXP
                              00C2          DIVW2   #1@<F_EXP-X_EXP>, R5     ; Shift R5 to scale X-floating
                              00C2          .ENDC
             7E     55  6D    00C2          CVTWX   R5, -(SP)                ; Push N onto the stack
          58      6846  98    00C5          CVTBL   (R8)[R6], R8             ; R8 = offset into FHI tables
                    5D  19    00C9          BLSS    LN_1_PLUS_W              ; Branch to handle F close to 1
   58    00000000'GF48  7E    00CB          MOVAX   G^MTH$$AB_D_FHI[R8], R8  ; R8 = Address of FHI
             56     88  7D    00D3          MOVX    (R8)+, R6                ; R6 = FHI
                              00D6  ;
                              00D6  ; Compute Z = (F - FHI)/(F + FHI)
                              00D6  ;                 Z = [(1+X) - Y*(1-X)]/[(1+X) + Y*(1-X)]
                              00D6  ;                 Z = [1 + X - Y + X*Y]/[1 + X + Y - X*Y]
```

```
                          00D6   ;       where Y = FHI*2**N, roughly equal to (1+X)/(1-X)
                          00D6   ;
         56   55   A0     00D6           ADDW    R5, R6                    ; R6 = FHI * 2**N = SFHI
    52   08   56   63     00D9           SUBX3   R6, S^#1.0, R2            ; R2 = 1 - SFHI
         56   08   60     00DD           ADDX    S^#1.0, R6                ; R6 = 1 + SFHI
         52   56   66     00E0           DIVX    R6, R2                    ; R2 = (1-SFHI)/(1+SFHI) = D
    56   52   50   61     00E3           ADDX3   R0, R2, R6                ; R6 = D + X
         52   50   64     00E7           MULX    R0, R2                    ; R2 = D * X
         52   08   60     00EA           ADDX    S^#1.0, R2                ; R2 = 1 + D*X
         56   52   66     00ED           DIVX    R2, R6                    ; R6 = (D+X)/(1+D*X) = Z
                          00F0   ;
                          00F0   ; Compute Z**2, P(Z**2) and Z*P(Z**2)
                          00F0   ;
    50   56   56   65     00F0           MULX3   R6, R6, R0                ; R0 = Z**2
FF5E CF  05   50   75     00F4           POLYX   R0, #LOGLEN2, LOGTAB2     ; R0 = P(Z**2)
         50   56   64     00FA           MULX    R6, R0                    ; R0 = Z*P(Z**2)
                          00FD   ;
                          00FD   ; Compute B = N*LN2_LO + LN_FHI_LO + Z*P(Z*Z)
                          00FD   ;
 52   FF06 CF   6E   65   00FD           MULX3   (SP), LN2_LO, R2          ; R2 = N*LN2_LO
         52   88   60     0103           ADDX    (R8)+, R2                 ; R2 = N*LN2_LO + LN_FHI_LO
         50   52   60     0106           ADDX    R2, R0                    ; R0 = B
                          0109   ;
                          0109   ; Compute A = N*LN2_HI + LN_FHI_HI and ALOG(X)
                          0109   ;
 52   FEF2 CF   8E   65   0109           MULX3   (SP)+, LN2_HI, R2         ; R2 = N*LN2_HI
         52   68   60     010F           ADDX    (R8), R2                  ; R2 = A = N*LN2_HI + LN_FHI_HI
         50   52   60     0112           ADDX    R2, R0                    ; R0 = A + B = ALOG(X)
    50   0080 8F   A2     0115           SUBW2   #1@X_EXP, R0              ; Divide by 2
                   04     011A           RET
                          011B
                          011B   NEG_EXP:
         56   55   A2     011B           SUBW    R5, R6                    ; R6 = F = 2f
         56   56   9A     011E           MOVZBL  R6, R6                    ; R6 = index into ALOG table
             00000000     0121           .IF     NE, F_EXP-X_EXP
                          0121           DIVW2   #1@<F_EXP-X_EXP>, R5      ; Shift R5 to scale X-floating
                          0121           .ENDC
         7E   55   6D     0121           CVTWX   R5, -(SP)                 ; Push N onto the stack
    58   6846      98     0124           CVTBL   (R8)[R6], R8              ; R8 = offset into FHI tables
                          0128   LN_1_PLUS_W:
              50   19     0128           BLSS    LN_1_PLUS                 ; Branch to handle F close to 1
58   00000000'GF48 7E     012A           MOVAX   G^MTH$$AB_D_FHI[R8], R8   ; R8 = Address of FHI
         56   68   7D     0132           MOVX    (R8), R6                  ; R6 = FHI
                          0135   ;
                          0135   ; Compute Z = (F - FHI)/(F + FHI)
                          0135   ;               Z = [(1+X) - Y*(1-X)]/[(1+X) + Y*(1-X)]
                          0135   ;               Z = [1 + X - Y + X*Y]/[1 + X + Y - X*Y]
                          0135   ;       where Y = FHI*2**N, roughly equal to (1+X)/(1-X)
                          0135   ;
         56   55   A0     0135           ADDW    R5, R6                    ; R6 = FHI * 2**N = SFHI
    52   08   56   63     0138           SUBX3   R6, S^#1.0, R2            ; R2 = 1 - SFHI
         56   08   60     013C           ADDX    S^#1.0, R6                ; R6 = 1 + SFHI
         52   56   66     013F           DIVX    R6, R2                    ; R2 = (1-SFHI)/(1+SFHI) = D
    56   52   50   61     0142           ADDX3   R0, R2, R6                ; R6 = D + X
         52   50   64     0146           MULX    R0, R2                    ; R2 = D * X
         52   08   60     0149           ADDX    S^#1.0, R2                ; R2 = 1 + D*X
         56   52   66     014C           DIVX    R2, R6                    ; R6 = (D+X)/(1+D*X) = Z
```

```
                             014F      ;
                             014F      ; Compute Z**2, P(Z**2) and Z*P(Z**2)
                             014F      ;
        50    56    56   65  014F              MULX3   R6, R6, R0              ; R0 = Z**2
  FEFF  CF    05    50   75  0153              POLYX   R0, #LOGLEN2, LOGTAB2   ; R0 = P(Z**2)
              50    56   64  0159              MULX    R6, R0                  ; R0 = Z*P(Z**2)
                             015C      ;
                             015C      ; Compute B = N*LN2_LO + LN_FHI_LO + Z*P(Z*Z)
                             015C      ;
  52    FEA7  CF    6E   65  015C              MULX3   (SP), LN2_LO, R2        ; R2 = N*LN2_LO
              52    78   60  0162              ADDX    -(R8), R2               ; R2 = N*LN2_LO + LN_FHI_LO
              50    52   60  0165              ADDX    R2, R0                  ; R0 = B
                             0168      ;
                             0168      ; Compute A = N*LN2_HI + LN_FHI_HI and ALOG(X)
                             0168      ;
  52    FE93  CF    8E   65  0168              MULX3   (SP)+, LN2_HI, R2       ; R2 = N*LN2_HI
              52    78   62  016E              SUBX    -(R8), R2               ; R2 = A = N*LN2_HI + LN_FHI_HI
              50    52   60  0171              ADDX    R2, R0                  ; R0 = A + B = ALOG(X)
        50    0080  8F   A2  0174              SUBW2   #1ax_EXP, R0            ; Divide by 2
                         04  0179              RET
                             017A
                             017A      ;
                             017A      ; Special logic for F close to 1
                             017A      ;
                             017A
                             017A      LN_1_PLUS:
        56    08    50   63  017A              SUBX3   R0, S^#1.0, R6          ; R6 = 1-X
                    55   B5  017E              TSTW    R5                      ; Determine which way to calculate W
                    0F   13  0180              BEQL    10$
        56    10    56   67  0182              DIVX3   R6, S^#2.0, R6          ; R6 = 2/(1-X)
              56    08   62  0186              SUBX    S^#1.0, R6              ; R6 = (1+X)/(1-X)
              56    55   A2  0189              SUBW    R5, R6                  ; Scale R6
              56    08   62  018C              SUBX    S^#1.0, R6              ; R6 = W
                    09   11  018F              BRB     20$
        56    50    56   67  0191      10$:    DIVX3   R6, R0, R6              ; R6 = X / (1-X)
        56    0080  8F   A0  0195              ADDW    #1ax_EXP, R6            ; R6 = W = 2*X/(1-X) = (1+X)/(1-X) - 1
  FE70  CF    08    56   75  019A      20$:    POLYX   R6, #LOGLEN1, LOGTAB1   ; R0 = Q(W)
              50    56   64  01A0              MULX    R6, R0                  ; Finish computing Q(W)
              50    56   64  01A3              MULX    R6, R0                  ; R0 = W*Q(W)
  52    FE5D  CF    6E   65  01A6              MULX3   (SP), LN2_LO, R2        ; R2 = N*LN2_LO
              50    52   60  01AC              ADDX    R2, R0                  ; R0 = N*LN2_LO + W*Q(W)
              50    56   60  01AF              ADDX    R6, R0                  ; R0 = N*LN2_LO + W*Q(W) + W
        6E    FE4A  CF   64  01B2              MULX    LN2_HI, (SP)            ; (SP) = N*LN2_HI
              50    8E   60  01B7              ADDX    (SP)+, R0               ; R0 = ALOG(X)
        50    0080  8F   A2  01BA              SUBW2   #1ax_EXP, R0            ; Divide by 2
                         04  01BF              RET
                             01C0      ;
                             01C0      ; X <= 0.0, signal error
                             01C0      ;
        7E    00'8F      9A  01C0      ERROR:  MOVZBL  #MTH$K_INVARGMAT, -(SP) ; condition value
        50    01    0F   79  01C4              ASHQ    #15, #1, R0             ; R0 = result = reserved operand -0.0
                             01C8                                              ; goes to signal mechanism vector
                             01C8                                              ; (CHF$L_MCH_R0/R1) so error handler
                             01C8                                              ; can modify the result.
   00000000'GF      01   FB  01C8              CALLS   #1, G^MTH$$SIGNAL       ; signal error and use real user's PC
                             01CF                                              ; independent of CALL vs JSB
                         04  01CF              RET                             ; return - R0 restored from
```

```
01D0   386                                     ; CHF$L_MCH_R0/R1
01D0   387          .END
```

```
ACMASK           = 000001FC
ERR                000000088 R      01
ERROR              000001C0 R       01
F_EXP            = 00000007
LN2_HI             00000000 R       01
LN2_LO             00000008 R       01
LN_1_PLUS          0000017A R       01
LN_1_PLUS_W        00000128 R       01
LOGLEN1          = 00000008
LOGLEN2          = 00000005
LOGTAB1            00000010 R       01
LOGTAB2            00000058 R       01
MTH$$AB_ALOG_V     ********      X  00
MTH$$AB_D_FHI      ********      X  00
MTH$$SIGNAL        ********      X  00
MTH$DATANH         0000008B RG      01
MTH$K_INVARGMAT    ********      X  00
NEG_EXP            0000011B R       01
X                = 00000004
X_EXP            = 00000007
```

```
                                   +-----------------+
                                   ! 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                        000001D0  (   464.)  01 (  1.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD  NOWRT NOVEC LONG
```

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

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                   30      00:00:00.09    00:00:00.73
Command processing              157      00:00:00.72    00:00:04.94
Pass 1                          101      00:00:01.37    00:00:05.19
Symbol table sort                 0      00:00:00.01    00:00:00.01
Pass 2                           82      00:00:00.93    00:00:03.49
Symbol table output               3      00:00:00.04    00:00:00.12
Psect synopsis output             2      00:00:00.02    00:00:00.02
Cross-reference output            0      00:00:00.00    00:00:00.00
Assembler run totals            377      00:00:03.20    00:00:14.54
```

The working set limit was 1050 pages.
6942 bytes (14 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 20 non-local and 2 local symbols.
447 source lines were read in Pass 1, producing 11 object records in Pass 2.
3 pages of virtual memory were used to define 2 macros.

```
+-----------------------------+
! 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$:MTHDATANH/OBJ=OBJ$:MTHDATANH MSRC$:MTHJACKET/UPDATE=(ENH$:MTHJACKET)+MS
```

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