```
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
```

```
MM      MM  TTTTTTTTTT  HH      HH  HH      HH  MM      MM    OOOOOO    DDDDDDDD
MM      MM  TTTTTTTTTT  HH      HH  HH      HH  MM      MM    OOOOOO    DDDDDDDD
MMMM  MMMM      TT      HH      HH  HH      HH  MMMM  MMMM  OO      OO  DD      DD
MMMM  MMMM      TT      HH      HH  HH      HH  MMMM  MMMM  OO      OO  DD      DD
MM  MM  MM      TT      HH      HH  HH      HH  MM  MM  MM  OO      OO  DD      DD
MM  MM  MM      TT      HH      HH  HH      HH  MM  MM  MM  OO      OO  DD      DD
MM      MM      TT      HHHHHHHHHH  HHHHHHHHHH  MM      MM  OO      OO  DD      DD
MM      MM      TT      HHHHHHHHHH  HHHHHHHHHH  MM      MM  OO      OO  DD      DD
MM      MM      TT      HH      HH  HH      HH  MM      MM  OO      OO  DD      DD
MM      MM      TT      HH      HH  HH      HH  MM      MM  OO      OO  DD      DD
MM      MM      TT      HH      HH  HH      HH  MM      MM  OO      OO  DD      DD
MM      MM      TT      HH      HH  HH      HH  MM      MM  OO      OO  DD      DD      ....
MM      MM      TT      HH      HH  HH      HH  MM      MM    OOOOOO    DDDDDDDD        ....
MM      MM      TT      HH      HH  HH      HH  MM      MM    OOOOOO    DDDDDDDD        ....

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
1-0

Table of contents


```
(1)      49       HISTORY                          ; Detailed Current Edit History
(2)      55       DECLARATIONS
(3)      90       MTH$HMOD - H REAL*16 remainder
```


MTH
1-0

```
0000          .TITLE  MTH$HMOD
0000          .IDENT  /3-002/                ; File: MTHHMOD.MAR Edit: JCW3002
0000
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    ;++
0000    ; FACILITY: MATH LIBRARY
0000    ;
0000    ; ABSTRACT:
0000    ;
0000    ;       This module contains the routine MTH$HMOD:
0000    ;       It returns the remainder of the division of arg1/arg2 using
0000    ;       the following equation:
0000    ;               arg1 - (int(arg1/arg2))*arg2
0000    ;
0000    ;
0000    ;--
0000    ;
0000    ; AUTHOR: Jeffrey C. Wiener, CREATION DATE: 21-DEC-1982
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;
0000    ;--
0000
0000          .SBTTL  HISTORY                        ; Detailed Current Edit History
0000    ;
0000    ; 3-001 Original version of complete re-write          JCW  21-DEC-82
0000    ; 3-002 DIVIDEND changed to equal 8 and DIVISOR changed to equal 4. JCW 14-Jun-83
0000    ;
```

```
                                        0000          .SBTTL  DECLARATIONS
                                        0000    ;
                                        0000    ; INCLUDE FILES:
                                        0000    ;
                                        0000          NONE
                                        0000    ;
                                        0000    ; EXTERNAL SYMBOLS:
                                        0000    ;
                                        0000          .DSABL  GBL             ; Force all external symbols to be declared
                                        0000          .EXTRN  MTH$$SIGNAL
                                        0000          .EXTRN  MTH$K_FLOUNDMAT
                                        0000          .EXTRN  MTH$K_INVARGMAT
                                        0000    ;
                                        0000    ; LIBRARY MACROS CALLS:
                                        0000    ;
                                        0000          $SFDEF                          ; Define SF$ (stack frame) symbols
                                        0000    ;
                                        0000    ; EQUATED SYMBOLS:
                                        0000    ;
                               00000070 0000          EXP_112   = ^X00000070              ; 112*2^0
                               FFFF01FF 0000          HIGH_MASK = ^XFFFF01FF
                                        0000    ;
                                        0000    ; OWN STORAGE:
                                        0000    ;
                                        0000          NONE
                                        0000    ;
                                        0000    ; PSECT DECLARATIONS:
                                        0000    ;
                                    00000000          .PSECT  _MTH$CODE               PIC, SHR, LONG, EXE, NOWRT
                                        0000
                                        0000    ; CONSTANTS:
                                        0000
                                        0000    TWO_EXP_112:
00000000 00000000 00000000 00004071     0000          .LONG   ^X00004071, ^X0, ^X0, ^X0       ; 2**112
```

```
          0010         .SBTTL  MTH$HMOD - H REAL*16 remainder
          0010  ;++
          0010  ; FUNCTIONAL DESCRIPTION:
          0010  ;
          0010  ;      Return the remainder of arg1/arg2 in H_floating point format
          0010  ;      Remainder = arg1 - (int(arg1/arg2))*arg2
          0010  ;
          0010  ; The algorithm used to evaluate the HMOD function is as follows:
          0010  ;
          0010  ;              X = the first argument.
          0010  ;              Y = the second argument.
          0010  ;      step 1. m = the exponent of Y.
          0010  ;              n = the exponent of X.
          0010  ;              c = n - m
          0010  ;              If c < 0, end with result = X.
          0010  ;      step 2. I = the fractional part of X.
          0010  ;              J = the fractional part of Y.
          0010  ;              If I >= J, I = I - J
          0010  ;              Go to step 5.
          0010  ;      step 3. L = 2^(p-1)*I, where p = 113 for H_floating numbers.
          0010  ;      step 4. T = L/J
          0010  ;              T = [T+2^(p-1)]-2^(p-1).   T is int(L/J) or int(L/J)+1
          0010  ;              I = L - J * T
          0010  ;              If I < 0, I = I + J        T was int(L/J)+1
          0010  ;      step 5. c = c - (p-1)
          0010  ;              If c > 0 go to step 3.
          0010  ;      step 6. If c = -(p-1) go to step 9.
          0010  ;      step 7. L = 2^(p-1+c) * I
          0010  ;      step 8. I = L - J * T
          0010  ;      step 9. Result = 2^m * I
          0010  ;
          0010  ; CALLING SEQUENCE:
          0010  ;
          0010  ;      CALL MTH$HMOD (remainder.wh.r, dividend.rh.r, divisor.rh.r)
          0010  ;
          0010  ; INPUT PARAMETERS:
          0010  ;
          0010  ;      The two input parameters are H_floating-point values.
          0010  ;
00000008  0010         DIVIDEND = 8                          ; Dividend = X in the algorithm.
0000000C  0010         DIVISOR  = 12                         ; Divisor  = Y in the algorithm.
          0010
          0010  ; IMPLICIT INPUTS:
          0010  ;
          0010  ;      NONE
          0010  ;
          0010  ; OUTPUT PARAMETERS:
          0010  ;
          0010  ;      Remainder is the remainder of the division of
          0010  ;      arg1/arg2, returned as an H_Floating point value.
          0010  ;
          0010  ; IMPLICIT OUTPUTS:
          0010  ;
          0010  ;      NONE
          0010  ;
          0010  ; COMPLETION CODES:
          0010  ;
```

```
                                  0010   ;       NONE
                                  0010   ;
                                  0010   ; SIDE EFFECTS:
                                  0010   ;
                                  0010   ;       MTH$_INVARGMAT - Invalid argument to math library if the divisor is zero.
                                  0010   ;       MTH$_FLOUNDMAT - Floating underflow in math library is signaled if
                                  0010   ;          the FU bit is set in the callers PSL.
                                  0010   ;
                                  0010   ;--
                                  0010
                            01FC  0010           .ENTRY  MTH$HMOD,      ^M<R2, R3, R4, R5, R6, R7, R8>
                                  0012
            54     0C BC 70FD     0012           MOVH    @DIVISOR(AP), R4              ; R4/R7 = Y
                      5F   13     0017           BEQL    ERROR                         ; Y=0. Division by zero
            50     08 BC 70FD     0019           MOVH    @DIVIDEND(AP), R0             ; R0/R3 = X
                                  001E
  7E    54   FFFF8000 8F   CB     001E           BICL3   #^XFFFF8000, R4, -(SP)        ; (SP)=m is the biased exponent of Y
  58    50   FFFF8000 8F   CB     0026           BICL3   #^XFFFF8000, R0, R8           ; R8=n is the biased exponent of X
                                  002E
               58    6E    C2     002E           SUBL2   (SP), R8                      ; R8 = c = n-m unbiased
                     06    18     0031           BGEQ    STEP_2                        ; Result is X if X<Y, ie, if c<0
            04 BC    50 7DFD      0033           MOVO    R0, @4(AP)                    ; @4(AP) = X
                           04     0038           RET
                                  0039
          54     4000 8F   B0     0039   STEP_2: MOVW    #^X4000, R4                   ; R4/R7 = J = biased |fract(Y)|
                                  003E
          50     4000 8F   B0     003E           MOVW    #^X4000, R0                   ; R0/R3 = I = biased |fract(X)|
                                  0043
                                  0043
                                  0043   ;+
                                  0043   ;
                                  0043   ;       In STEP_4 and STEP_8 the calculation of  I = L - J*int(L/J) must be
                                  0043   ;       computed as precisely as possible. To do this we will need to write J as
                                  0043   ;               J = J1 + J2
                                  0043   ;       where J1 = the high 56 bits of J and J2 = J - J1, the low 57 bits of J.
                                  0043   ;
                                  0043   ;       HIGH_MASK is used to extract the 7 bits of J from longword3 that belong
                                  0043   ;       to J1.
                                  0043   ;
                                  0043   ;-
                                  0043
                                  0043
                 7E    54 7DFD    0043           MOVO    R4, -(SP)                     ; (SP) = J
       08 AE   FFFF01FF 8F   CA   0047           BICL    #HIGH_MASK, 8(SP)             ;
                 0C AE     D4     004F           CLRL    12(SP)                        ; (SP) = J1 replaces the value
                                  0052                                                 ;   of J on the top of SP
          7E    54    6E 63FD     0052           SUBH3   (SP), R4, -(SP)               ; (SP) = J2 = J - J1
                                  0057
                54    50 71FD     0057           CMPH    R0, R4                        ; If I<J
                      03   18     005B           BGEQ    I_GEQ_J
                    009D   31     005D           BRW     STEP_5                        ; go to STEP_5
                                  0060   I_GEQ_J:
                50    54 62FD     0060           SUBH2   R4, R0                        ; else I = I-J
                      03   15     0064           BLEQ    OVER
                    0094   31     0066           BRW     STEP_5                        ; go to STEP_5 if I>0, or
                                  0069                                                 ;   else the algorithm ends
                   08 BC   B5     0069   OVER:   TSTW    @DIVIDEND(AP)                 ; the sign of the result is
```

```
               04    18  006C              BGEQ    DONE                            ; the same as the sign of
          50   50  72FD  006E              MNEGH   R0, R0                          ; the first argument, A.
       04 BC   50  7DFD  0072  DONE:       MOVO    R0, @4(AP)                      ; Return the correct result
                     04  0077              RET
                         0078
    50    01   0F    79  0078  ERROR:      ASHQ    #15, #1, R0                     ; Y=0. Reserved operand
               52    7C  007C              CLRQ    R2
      7E   00'8F     9A  007E              MOVZBL  #MTH$K_INVARGMAT, -(SP)         ; error code
 00000000'GF   01    FB  0082              CALLS   #1, G^MTH$$SIGNAL               ; signal the error
       04 BC   50  7DFD  0089              MOVO    R0, @4(AP)                      ; Return the correct result
                     04  008E              RET
                         008F
 50   00000070 8F    C0  008F  STEP_3:     ADDL2   #EXP_112, R0                    ; R0/R3 = L = 2**(p-1)*I
                         0096
                         0096
                         0096  ;+
                         0096  ;
                         0096  ;           STEP_4:
                         0096  ;           2^(p-1) = 2^(112) is added and then subtracted from
                         0096  ;           T = int(L/J) to ensure that T = chopped(L/J) or chopped(L/J)+1
                         0096  ;
                         0096  ;-
                         0096
          7E   54  7DFD  0096              MOVO    R4, -(SP)                       ; save J for use in STEP_5
     54   50   54  67FD  009A              DIVH3   R4, R0, R4                      ; R4/R7 = T = L/J
     54   FF5C CF  60FD  009F              ADDH2   TWO_EXP_112, R4                 ; R4/R7 = T = T+2**(p-1)
     54   FF56 CF  62FD  00A5              SUBH2   TWO_EXP_112, R4                 ; T-2**(p-1) = L/J chopped
                         00AB                                                      ;    or L/J chopped + 1
                         00AB
                         00AB  ;+
                         00AB  ;
                         00AB  ;           The calculation of  I = L - J*int(L/J) must be computed as precisely
                         00AB  ;           as possible. To do this we will need to write T as
                         00AB  ;                   T = Z1 + Z2
                         00AB  ;           where Z1 = the high 56 bits of T and Z2 = T - Z1, the low 57 bits of T.
                         00AB  ;
                         00AB  ;           Now, using J = J1 + J2,
                         00AB  ;
                         00AB  ;                   L - J * int(L/J) = L - (J1 + J2) * (Z1 + Z2)
                         00AB  ;                                    = L - (Z1 * J1) - (Z1 * J2)
                         00AB  ;                                        - (Z2 * J1) - (Z2 * J2)
                         00AB  ;                                    = L - (Z1 * J ) - (Z2 * J )
                         00AB  ;
                         00AB  ;-
                         00AB
          7E   54  7DFD  00AB              MOVO    R4, -(SP)                       ; Stack Z = INT(L/J)
 08 AE   FFFF01FF 8F     CA  00AF              BICL    #HIGH_MASK, 8(SP)               ; Start to form Z1
            0C AE    D4  00B7              CLRL    12(SP)                          ; (SP) = Z1
     7E   54   6E  63FD  00BA              SUBH3   (SP), R4, -(SP)                 ; (SP) = Z2
 54  10 AE  40 AE  65FD  00BF              MULH3   64(SP), 16(SP), R4              ; Compute Z1*J1
          50   54  62FD  00C6              SUBH2   R4, R0                          ; R0/R3 = L - Z1*J1
 54  30 AE  10 AE  65FD  00CA              MULH3   16(SP), 48(SP), R4              ; R4/R7 = Z1*J2
          50   54  62FD  00D1              SUBH2   R4, R0                          ; R0/R3 = L - Z1*J
     54   40 AE    6E  65FD  00D5          MULH3   (SP), 64(SP), R4                ; R4/R7 = Z2*J1
          50   54  62FD  00DB              SUBH2   R4, R0                          ; R0/R3 = L - Z1*J - Z2*J1
     54   30 AE    6E  65FD  00DF          MULH3   (SP), 48(SP), R4                ; R4/R7 = Z2*J2
          50   54  62FD  00E5              SUBH2   R4, R0                          ; R0/R3 = L - Z*J
```

```
                 5E    20    C0  00E9            ADDL2   #32, SP                         ; remove Z1,Z2,J1,J2 from stack
                 54    8E  7DFD  00EC            MOVO    (SP)+, R4                       ; R4/R7 = J
                       50    B5  00F0            TSTW    R0
                       09    14  00F2            BGTR    STEP_5                          ; If R0/R3>0 the algoritm cotinues
                       03    19  00F4            BLSS    SUBTRACT_J
                     0093    31  00F6            BRW     RETURN                          ; If R0/R3=0 the algorithm ends
                                 00F9  SUBTRACT_J:
                 50    54  60FD  00F9            ADDH2   R4, R0                          ; Add J back in because you had
                                 00FD                                                    ;   T = chopped(L/J)+1
                                 00FD
      58   00000070 8F       C2  00FD  STEP_5:  SUBL2   #EXP_112, R8                    ;c = c-(p-1) = c-112
                       89    18  0104            BGEQ    STEP_3
      58   00000070 8F       C0  0106            ADDL2   #EXP_112, R8                    ; c = (p-1)+c = 112+c
                                 010D
                                 010D  ;+
                                 010D  ;
                                 010D  ;         The next two lines of code are STEP_6 and STEP_7.
                                 010D  ;
                                 010D  ;-
                                 010D
                       67    13  010D            BEQL    STEP_9                          ;
                 50    58    C0  010F            ADDL2   R8, R0                          ; L = I*2^(c+t)
                                 0112
                                 0112  ;+
                                 0112  ;         2^(p-1) = 2^(55) is added and then subtracted from
                                 0112  ;         T = int(L/J) to ensure that T = chopped(L/J) or chopped(L/J)+1
                                 0112  ;-
                                 0112
                 7E    54  7DFD  0112            MOVO    R4, -(SP)                       ; Save J
          54     50    54  67FD  0116            DIVH3   R4, R0, R4                      ; R4/R7 = T = L/J
          54   FEE0 CF     60FD  011B            ADDH2   TWO_EXP_112, R4                 ; R4/R7 = T = T+2**(p-1)
          54   FEDA CF     62FD  0121            SUBH2   TWO_EXP_112, R4                 ; T-2**(p-1) = L/J chopped
                                 0127                                                    ;     or L/J chopped + 1
                                 0127  ;+
                                 0127  ;
                                 0127  ;         STEP_8:
                                 0127  ;
                                 0127  ;         The calculation of  I = L - J*int(L/J) must be computed as precisely
                                 0127  ;         as possible. To do this we will need to write T as
                                 0127  ;                 T = Z1 + Z2
                                 0127  ;         where Z1 = the high 56 bits of T and Z2 = T - Z1, the low 57 bits of T.
                                 0127  ;
                                 0127  ;         Now, using J = J1 + J2,
                                 0127  ;
                                 0127  ;                 L - J * int(L/J) = L - (J1 + J2) * (Z1 + Z2)
                                 0127  ;                                  = L - (Z1 * J1) - (Z1 * J2)
                                 0127  ;                                      - (Z2 * J1) _ (Z2 * J2)
                                 0127  ;                                  = L - (Z1 * J ) = (Z2 * J )
                                 0127  ;
                                 0127  ;-
                                 0127
                 7E    54  7DFD  0127            MOVO     R4, -(SP)                      ; Stack Z = INT(L/J)
    08 AE     FFFF01FF 8F    CA  012B            BICL    #HIGH_MASK, 8(SP)               ; Start to form Z1
                    0C AE    D4  0133            CLRL    12(SP)                          ; (SP) = Z1
          7E     54    6E  63FD  0136            SUBH3   (SP), R4, -(SP)                 ; (SP) = Z2
   54   10 AE    40 AE     65FD  013B            MULH3   64(SP), 16(SP), R4              ; Compute Z1*J1
                 50    54  62FD  0142            SUBH2   R4, R0                          ; R0/R3 = L - Z1*J1
```

```
54   30 AE    10 AE 65FD  0146            MULH3   16(SP), 48(SP), R4          ; R4/R7 = Z1*J2
           50    54 62FD  014D            SUBH2   R4, R0                      ; R0/R3 = L - Z1*J
  54    40 AE    6E 65FD  0151            MULH3   (SP), 64(SP), R4            ; R4/R7 = Z2*J1
           50    54 62FD  0157            SUBH2   R4, R0                      ; R0/R3 = L - Z1*J - Z2*J1
  54    30 AE    6E 65FD  015B            MULH3   (SP), 48(SP), R4            ; R4/R7 = Z2*J2
           50    54 62FD  0161            SUBH2   R4, R0                      ; R0/R3 = L - Z*J
           5E    20   C0  0165            ADDL2   #32, SP                     ; Remove Z1 and Z2 from the stack
           54    8E 7DFD  0168            MOVO    (SP)+, R4                   ; Restore J
                 50   B5  016C            TSTW    R0
                 06   14  016E            BGTR    STEP_9
                 1A   13  0170            BEQL    RETURN                      ; If R0/R3=0 the algorithm ends
           50    54 60FD  0172            ADDH2   R4, R0                      ; Add J back in because you had
                          0176                                                ;   T = chopped(L/J)+1
                          0176
  20 AE    4000 8F    A2  0176  STEP_9:   SUBW2   #^X4000, 32(SP)             ; Remove the bias from m
       50    20 AE    A0  017C            ADDW2   32(SP), R0                  ;  and form R0/R1 = 2^m*L
                 10   19  0180            BLSS    UNDERFLOW
                          0182
                          0182  TEST_SIGN:
              08 BC   B5  0182            TSTW    @DIVIDEND(AP)               ; the sign of the result is
                 05   18  0185            BGEQ    RETURN                      ; the same as the sign of
     50    8000 8F    A8  0187            BISW2   #^X8000, R0                 ; the first argument, X.
      04 BC      50 7DFD  018C  RETURN:   MOVO    R0, @4(AP)                  ; Return the correct result
                      04  0191            RET
                          0192
                          0192  UNDERFLOW:
              04 BC 7CFD  0192            CLRO    @4(AP)                      ; Set up default result to 0.0
    0D 04 AD     06   E1  0196            BBC     #SF$V_FU, SF$W_SAVE_PSW(FP), NO_FU
                          019B                                                ; Branch if caller has not enabled F
      00000000'8F     DD  019B            PUSHL   #MTH$K_FLOUNDMAT            ; Report MTH$_FLOUNDMAT
  00000000'GF    01   FB  01A1            CALLS   #1, G^MTH$$SIGNAL           ; Signal the condition
                      04  01A8  NO_FU:    RET                                 ; Return
                          01A9
                          01A9            .END
```

```
DIVIDEND        = 00000008
DIVISOR         = 0000000C
DONE              00000072 R      02
ERROR             00000078 R      02
EXP_112         = 00000070
HIGH_MASK       = FFFF01FF
I_GEQ_J           00000060 R      02
MTH$$SIGNAL       ********    X   00
MTH$HMOD          00000010 RG     02
MTH$K_FLOUNDMAT   ********    X   00
MTH$K_INVARGMAT   ********    X   00
NO_FU             000001A8 R      02
OVER              00000069 R      02
RETURN            0000018C R      02
SF$V_FU         = 00000006
SF$W_SAVE_PSW   = 00000004
STEP_2            00000039 R      02
STEP_3            0000008F R      02
STEP_5            000000FD R      02
STEP_9            00000176 R      02
SUBTRACT_J        000000F9 R      02
TEST_SIGN         00000182 R      02
TWO_EXP_112       00000000 R      02
UNDERFLOW         00000192 R      02
```

```
                                        +-----------------+
                                        ! Psect synopsis !
                                        +-----------------+

PSECT name                      Allocation            PSECT No.  Attributes
----------                      ----------            ---------  ----------
.  ABS  .                       00000000  (     0.)   00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                           00000000  (     0.)   01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
_MTH$CODE                       000001A9  (   425.)   02 (  2.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD   NOWRT NOVEC LONG
```

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

Phase                     Page faults    CPU Time        Elapsed Time
-----                     -----------    --------        ------------
Initialization                     29    00:00:00.10     00:00:00.62
Command processing                121    00:00:00.49     00:00:02.61
Pass 1                            120    00:00:01.47     00:00:08.35
Symbol table sort                   0    00:00:00.03     00:00:00.08
Pass 2                             77    00:00:00.75     00:00:02.72
Symbol table output                 3    00:00:00.03     00:00:00.03
Psect synopsis output               2    00:00:00.02     00:00:00.17
Cross-reference output              0    00:00:00.00     00:00:00.00
Assembler run totals              354    00:00:02.89     00:00:14.60

The working set limit was 900 pages.
7106 bytes (14 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 0 local symbols.
351 source lines were read in Pass 1, producing 14 object records in Pass 2.
8 pages of virtual memory were used to define 7 macros.
```

```
+---------------------------+
! Macro library statistics !
+---------------------------+

Macro library name                       Macros defined
------------------                       --------------
_$255$DUA28:[SYSLIB]STARLET.MLB;2              4

88 GETS were required to define 4 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS$:MTHHMOD/OBJ=OBJ$:MTHHMOD MSRC$:MTHHMOD/UPDATE=(ENH$:MTHHMOD)
```

0262 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MTHIISIGN
LIS

MTHHFLOOR
LIS

MTHHSIGN
LIS

MTHHMIN1
LIS

MTHHLOG
LIS

MTHHTAN
LIS

MTHIIDNNT
LIS

MTHIIHNNT
LIS

MTHHSQRT
LIS

MTHIMAX0
LIS

MTHHNINT
LIS

MTHHSINH
LIS

MTHHTANH
LIS

MTHHINT
LIS

MTHHMAX1
LIS

MTHHSINCO
LIS

MTHHMOD
LIS

MTHIIGNNT
LIS