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

FILEID**MTHGMOD

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

Table of contents

```
0000     1          .TITLE  MTH$GMOD
0000     2          .IDENT  /3-001/                 ; File: MTHGMOD.MAR Edit: JCW3001
0000     3
0000     4  ;
0000     5  ;*****************************************************************************
0000     6  ;*                                                                           *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16  ;*  TRANSFERRED.                                                             *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                             *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24  ;*                                                                           *
0000    25  ;*                                                                           *
0000    26  ;*****************************************************************************
0000    27  ;
0000    28
0000    29  ;++
0000    30  ; FACILITY: MATH LIBRARY
0000    31  ;
0000    32  ; ABSTRACT:
0000    33  ;
0000    34  ;       This module contains the routine MTH$GMOD:
0000    35  ;       It returns the remainder of the division of arg1/arg2 using
0000    36  ;       the following equation:
0000    37  ;               arg1 - (int(arg1/arg2))*arg2
0000    38  ;
0000    39  ;
0000    40  ;--
0000    41  ;
0000    42  ; AUTHOR: Jeffrey C. Wiener, CREATION DATE: 21-DEC-1982
0000    43  ;
0000    44  ; MODIFIED BY:
0000    45  ;
0000    46  ;
0000    47  ;--
0000    48
0000    49          .SBTTL  HISTORY                         ; Detailed Current Edit History
0000    50  ;
0000    51  ; Edit History for Version 3.0:
0000    52  ;
0000    53  ; 3-001 Original version of complete re-write          JCW  21-DEC-82
0000    54  ;
```

```
                     0000     56          .SBTTL  DECLARATIONS
                     0000     57 ;
                     0000     58 ; INCLUDE FILES:
                     0000     59 ;
                     0000     60 ;       NONE
                     0000     61 ;
                     0000     62 ; EXTERNAL SYMBOLS:
                     0000     63 ;
                     0000     64          .DSABL  GBL             ; Force all external symbols to be declared
                     0000     65          .EXTRN  MTH$$SIGNAL
                     0000     66          .EXTRN  MTH$K_FLOUNDMAT
                     0000     67          .EXTRN  MTH$K_INVARGMAT
                     0000     68 ;
                     0000     69 ; LIBRARY MACROS CALLS:
                     0000     70 ;
                     0000     71          $SFDEF                          ; Define SF$ (stack frame) symbols
                     0000     72 ;
                     0000     73 ; EQUATED SYMBOLS:
                     0000     74 ;
00000340             0000     75          EXP_52    = ^X00000340          ; 52*2^4
FFFF07FF             0000     76          HIGH_MASK = ^XFFFF07FF
                     0000     77 ;
                     0000     78 ; OWN STORAGE:
                     0000     79 ;
                     0000     80 ;       NONE
                     0000     81 ;
                     0000     82 ; PSECT DECLARATIONS:
                     0000     83 ;
                00000000     84          .PSECT  _MTH$CODE               PIC, SHR, LONG, EXE, NOWRT
                     0000     85
                     0000     86 ; CONSTANTS:
                     0000     87
                     0000     88 TWO_EXP_52:
00000000 00004350    0000     89          .LONG   ^X00004350, ^X0         ; 2**52
```

```
                     0008    91          .SBTTL  MTH$GMOD - G REAL*8 remainder
                     0008    92 ;++
                     0008    93 ; FUNCTIONAL DESCRIPTION:
                     0008    94 ;
                     0008    95 ;       Return the remainder of arg1/arg2 in G_floating point format
                     0008    96 ;       Remainder = arg1 - (int(arg1/arg2))*arg2
                     0008    97 ;
                     0008    98 ; The algorithm used to evaluate the GMOD function is as follows:
                     0008    99 ;
                     0008   100 ;               X = the first argument.
                     0008   101 ;               Y = the second argument.
                     0008   102 ;       step 1. m = the exponent of Y.
                     0008   103 ;               n = the exponent of X.
                     0008   104 ;               c = n - m
                     0008   105 ;               If c < 0, end with result = X.
                     0008   106 ;       step 2. I = the fractional part of X.
                     0008   107 ;               J = the fractional part of Y.
                     0008   108 ;               If I >= J, I = I - J
                     0008   109 ;               Go to step 5.
                     0008   110 ;       step 3. L = 2^(p-1)*I, where p = 53 for G_floating numbers.
                     0008   111 ;       step 4. T = L/J
                     0008   112 ;               T = [T+2^(p-1)]-2^(p-1).   T is int(L/J) or int(L/J)+1
                     0008   113 ;               I = L - J * T
                     0008   114 ;               If I < 0, I = I + J        T was int(L/J)+1
                     0008   115 ;       step 5. c = c - (p-1)
                     0008   116 ;               If c > 0 go to step 3.
                     0008   117 ;       step 6. If c = -(p-1) go to step 9.
                     0008   118 ;       step 7. L = 2^(p-1+c) * I
                     0008   119 ;       step 8. I = L - J * T
                     0008   120 ;       step 9. Result = 2^m * I
                     0008   121 ;
                     0008   122 ; CALLING SEQUENCE:
                     0008   123 ;
                     0008   124 ;       Remainder.wg.v = MTH$GMOD (dividend.rg.r, divisor.rg.r)
                     0008   125 ;
                     0008   126 ; INPUT PARAMETERS:
                     0008   127 ;
                     0008   128 ;       The two input parameters are G_floating-point values. They are
                     0008   129 ;       passed by reference.
                     0008   130 ;
00000004             0008   131         DIVIDEND = 4                            ; Dividend = X in the algorithm.
00000008             0008   132         DIVISOR  = 8                            ; Divisor  = Y in the algorithm.
                     0008   133
                     0008   134 ; IMPLICIT INPUTS:
                     0008   135 ;
                     0008   136 ;       NONE
                     0008   137 ;
                     0008   138 ; FUNCTION VALUE:
                     0008   139 ;
                     0008   140 ;       Remainder of the division arg1/arg2 is returned as a
                     0008   141 ;       G_floating point value.
                     0008   142 ;
                     0008   143 ; IMPLICIT OUTPUTS:
                     0008   144 ;
                     0008   145 ;       NONE
                     0008   146 ;
                     0008   147 ; COMPLETION CODES:
```

```
                                0008   148 ;
                                0008   149 ;       NONE
                                0008   150 ;
                                0008   151 ; SIDE EFFECTS:
                                0008   152 ;
                                0008   153 ;       MTH$_INVARGMAT - Invalid argument to math library if the divisor is zero.
                                0008   154 ;       MTH$_FLOUNDMAT - Floating underflow in math library is signaled if
                                0008   155 ;          the FU bit is set in the callers PSL.
                                0008   156 ;
                                0008   157 ;--
                                0008   158
                         01FC   0008   159         .ENTRY  MTH$GMOD,       ^M<R2, R3, R4, R5, R6, R7, R8>
                                000A   160
          52   08 BC 50FD      000A   161         MOVG    @DIVISOR(AP), R2                ; R2/R3 = Y
                57   13        000F   162         BEQL    ERROR                           ; Y=0. Division by zero
          50   04 BC 50FD      0011   163         MOVG    @DIVIDEND(AP), R0               ; R0/R1 = X
                                0016   164
    56    52   FFFF800F 8F CB  0016   165         BICL3   #^XFFFF800F, R2, R6             ; R6=m is the biased exponent of Y
    58    50   FFFF800F 8F CB  001E   166         BICL3   #^XFFFF800F, R0, R8             ; R8=n is the biased exponent of X
                                0026   167
             58   56   C2      0026   168         SUBL2   R6, R8                          ; R4 = c = n-m unbiased
                  01   18      0029   169         BGEQ    STEP_2                          ; Result is X if X<Y, ie, if c<0
                       04      002B   170         RET                                     ; R0/R1 = X
                               002C   171
                       56 DD   002C   172 STEP_2: PUSHL   R6                              ; push m onto the stack
                               002E   173
          52   FFF0 8F AA      002E   174         BICW2   #^XFFF0, R2                     ; R2/R3 = J = unbiased !fract(Y)!
          52   4000 8F AC      0033   175         XORW    #^X4000, R2                     ; J = properly biased !fract(Y)!
                               0038   176
          50   FFF0 8F AA      0038   177         BICW2   #^XFFF0, R0                     ; R0/R1 = I = unbiased !fract(X)!
          50   4000 8F AC      003D   178         XORW    #^X4000, R0                     ; I = properly biased !fract(X)!
                               0042   179
                               0042   180 ;+
                               0042   181 ;
                               0042   182 ;       In STEP_4 and STEP_8 the calculation of  I = L - J*int(L/J) must be
                               0042   183 ;       computed as precisely as possible. To do this we will need to write J as
                               0042   184 ;                J = J1 + J2
                               0042   185 ;       where J1 = the high 26 bits of J and J2 = J - J1, the low 27 bits of J.
                               0042   186 ;
                               0042   187 ;       HIGH_MASK is used to extract the 5 bits of J from longword2 that belong
                               0042   188 ;       to J1.
                               0042   189 ;
                               0042   190 ;-
                               0042   191
             7E   52   7D      0042   192         MOVQ    R2, -(SP)                       ; (SP) = J
04 AE     FFFF07FF 8F CA      0045   193         BICL    #HIGH_MASK, 4(SP)               ; (SP) = J1 replaces the value
                               004D   194                                                 ;   on the top of SP.
       7E    52   6E 43FD     004D   195         SUBG3   (SP), R2, -(SP)                 ; (SP) = J2 = J - J1
                               0052   196
             52   50 51FD      0052   197         CMPG    R0, R2                          ; If I<J
                  73   19      0056   198         BLSS    STEP_5                          ; go to STEP_5
             50   52 42FD      0058   199         SUBG2   R2, R0                          ; else I = I-J
                  6D   14      005C   200         BGTR    STEP_5                          ; go to STEP_5 if I>0, or
                               005E   201                                                 ;   else the algorithm ends
               04 BC   B5      005E   202         TSTW    @DIVIDEND(AP)                   ; the sign of the result is
                  04   18      0061   203         BGEQ    DONE                            ; the same as the sign of
             50   50 52FD      0063   204         MNEGG   R0, R0                          ; the first argument, A.
```

```
                     04  0067   205 DONE:   RET
                         0068   206
          50  01  0F   79  0068   207 ERROR:  ASHQ    #15, #1, R0                     ; Y=0. Reserved operand.
            7E  00'8F   9A  006C   208         MOVZBL  #MTH$K_INVARGMAT, -(SP)         ; error code
00000000'GF     01   FB  0070   209         CALLS   #1, G^MTH$$SIGNAL               ; signal the error
                     04  0077   210         RET
                         0078   211
50  00000340 8F      C0  0078   212 STEP_3: ADDL2   #EXP_52, R0                      ; R0/R1 = L = 2**(p-1)*I
                         007F   213
                         007F   214 ;+
                         007F   215 ;
                         007F   216 ;       STEP_4:
                         007F   217 ;       2^(p-1) = 2^(52) is added and then subtracted from
                         007F   218 ;       T = int(L/J) to ensure that T = chopped(L/J) or chopped(L/J)+1
                         007F   219 ;
                         007F   220 ;-
                         007F   221
56  50   52 47FD   007F   222         DIVG3   R2, R0, R6                      ; R6/R7 = T = L/J
56  FF77 CF 40FD   0084   223         ADDG2   TWO_EXP_52, R6                  ; R6/R7 = T = T+2**(p-1)
56  FF71 CF 42FD   008A   224         SUBG2   TWO_EXP_52, R6                  ; T-2**(p-1) = L/J chopped
                         0090   225
                         0090   226 ;+
                         0090   227 ;
                         0090   228 ;       The calculation of  I = L - J*int(L/J) must be computed as precisely
                         0090   229 ;       as possible. To do this we will need to write T as
                         0090   230 ;                T = Z1 + Z2
                         0090   231 ;       where Z1 = the high 26 bits of T and Z2 = T - Z1, the low 27 bits of T.
                         0090   232 ;
                         0090   233 ;       Now, using J = J1 + J2,
                         0090   234 ;
                         0090   235 ;              L - J * int(L/J) = L - (J1 + J2) * (Z1 + Z2)
                         0090   236 ;                               = L - (Z1 * J1) - (Z1 * J2)
                         0090   237 ;                                   - (Z2 * J1)   (Z2 * J2)
                         0090   238 ;                               = L - (Z1 * J ) = (Z2 * J )
                         0090   239 ;
                         0090   240 ;-
                         0090   241
          54  56   D0    0090   242         MOVL    R6, R4                          ;
55  57  FFFF07FF 8F   CB  0093   243         BICL3   #HIGH_MASK, R7, R5              ; R4/R5 = Z1
          56  54 42FD   009B   244         SUBG2   R4, R6                          ; R6/R7 = Z2
 7E   08 AE  54 45FD   009F   245         MULG3   R4, 8(SP), -(SP)                ; Compute Z1*J1
          50  8E 42FD   00A5   246         SUBG2   (SP)+, R0                       ; R0/R1 = L - Z1*J1
          54  6E 44FD   00A9   247         MULG2   (SP), R4                        ; R4/R5 = Z1*J2
          50  54 42FD   00AD   248         SUBG2   R4, R0                          ; R0/R1 = L - Z1*J
 54   56   08 AE  45FD   00B1   249         MULG3   8(SP), R6, R4                   ; R4/R5 = Z2*J1
          50  54 42FD   00B7   250         SUBG2   R4, R0                          ; R0/R1 = L - Z1*J - Z2*J1
          56  6E 44FD   00BB   251         MULG2   (SP), R6                        ; R6/R7 = Z2*J2
          50  56 42FD   00BF   252         SUBG2   R6, R0                          ; R0/R1 = L - Z*J
              06   14   00C3   253         BGTR    STEP_5
              7B   13   00C5   254         BEQL    RETURN                          ; If R0/R1=0 the algorithm ends
          50  52 40FD   00C7   255         ADDG2   R2, R0                          ; Add J back in because you had
                         00CB   256                                                 ;   T=chopped(L/J)+1
                         00CB   257
 58  00000340 8F   C2  00CB   258 STEP_5: SUBL2   #EXP_52, R8                     ;c = c-(p-1) = c-52
              A4   18   00D2   259         BGEQ    STEP_3
 58  00000340 8F   C0  00D4   260         ADDL2   #EXP_52, R8                     ; c = (p-1)+c = 52+c
                         00DB   261
```

```
                                00DB   262 ;+
                                00DB   263 ;
                                00DB   264 ;       The next two lines of code are STEP_6 and STEP_7.
                                00DB   265 ;
                                00DB   266 ;-
                                00DB   267
                          4F    13    00DB   268         BEQL    STEP_9                          ;
                    50    58    C0    00DD   269         ADDL2   R8, R0                          ; L = I*2^(c+t)
                                00E0   270
                                00E0   271 ;+
                                00E0   272 ;       2^(p-1) = 2^(55) is added and then subtracted from
                                00E0   273 ;       T = int(L/J) to ensure that T = chopped(L/J) or chopped(L/J)+1
                                00E0   274 ;-
                                00E0   275
              56    50    52 47FD   00E0   276         DIVG3   R2, R0, R6                      ; R6/R7 = T = L/J
              56    FF16  CF 40FD   00E5   277         ADDG2   TWO_EXP_52, R6                  ; R6/R7 = T = T+2**(p-1)
              56    FF10  CF 42FD   00EB   278         SUBG2   TWO_EXP_52, R6                  ; T-2**(p-1) = L/J chopped
                                00F1   279                                                 ;    or L/J chopped + 1
                                00F1   280 ;+
                                00F1   281 ;
                                00F1   282 ;       STEP_8:
                                00F1   283 ;
                                00F1   284 ;       The calculation of  I = L - J*int(L/J) must be computed as precisely
                                00F1   285 ;       as possible. To do this we will need to write T as
                                00F1   286 ;                  T = Z1 + Z2
                                00F1   287 ;       where Z1 = the high 26 bits of T and Z2 = T - Z1, the low 27 bits of T.
                                00F1   288 ;
                                00F1   289 ;       Now, using J = J1 + J2,
                                00F1   290 ;
                                00F1   291 ;                L - J * int(L/J) = L - (J1 + J2) * (Z1 + Z2)
                                00F1   292 ;                                 = L - (Z1 * J1) - (Z1 * J2)
                                00F1   293 ;                                   - (Z2 * J1)   (Z2 * J2)
                                00F1   294 ;                                 = L - (Z1 * J ) = (Z2 * J )
                                00F1   295 ;
                                00F1   296 ;-
                                00F1   297
                    54    56    D0    00F1   298         MOVL    R6, R4
55    57    FFFF07FF  8F    CB    00F4   299         BICL3   #HIGH_MASK, R7, R5              ; R4/R5 = Z1
                    56    54 42FD   00FC   300         SUBG2   R4, R6                          ; R6/R7 = Z2
      7E    08 AE   54    45FD    0100   301         MULG3   R4, 8(SP), -(SP)                ; Compute Z1*J1
                    50    8E 42FD   0106   302         SUBG2   (SP)+, R0                       ; R0/R1 = L - Z1*J1
                    54    6E 44FD   010A   303         MULG2   (SP), R4                        ; R4/R5 = Z1*J2
                    50    54 42FD   010E   304         SUBG2   R4, R0                          ; R0/R1 = L - Z1*J
      54    56      08 AE 45FD     0112   305         MULG3   8(SP), R6, R4                   ; R4/R5 = Z2*J1
                    50    54 42FD   0118   306         SUBG2   R4, R0                          ; R0/R1 = L - Z1*J - Z2*J1
                    56    6E 44FD   011C   307         MULG2   (SP), R6                        ; R6/R7 = Z2*J2
                    50    56 42FD   0120   308         SUBG2   R6, R0                          ; R0/R1 = L - Z*J
                          06   14    0124   309         BGTR    STEP_9
                          1A   13    0126   310         BEQL    RETURN                          ; If R0/R1=0 the algorithm ends
                    50    52 40FD   0128   311         ADDG2   R2, R0                          ; Add J back in because you had
                                012C   312                                                 ;   T=chopped(L/J)+1
                                012C   313
      10 AE   4000  8F    A2    012C   314 STEP_9:  SUBW    #^X4000, 16(SP)                 ; Remove the bias from m and
            50    10 AE   A0    0132   315         ADDW    16(SP), R0                      ;   form R0/R1 = 2^m*L
                          0B   19    0136   316         BLSS    UNDERFLOW                       ; Branch if underflow
                                0138   317 TEST_SIGN:
                 04 BC    B5    0138   318         TSTW    @DIVIDEND(AP)                   ; the sign of the result is
```

```
                 05   18  013B   319          BGEQ    RETURN                          ; the same as the sign of
          50     8000 8F  A8  013D   320          BISW2   #^X8000, R0                 ; the first argument, X.
                      04  0142   321 RETURN:    RET
                          0143   322
                          0143   323 UNDERFLOW:
                 50   7C  0143   324          CLRQ    R0                              ; Set up default result to 0.0
             0D 04 AD 06  E1  0145   325          BBC     #SF$V_FU, SF$W_SAVE_PSW(FP), NO_FU
                          014A   326                                                  ; Branch if caller has not enabled F
           00000000'8F   DD  014A   327          PUSHL   #MTH$K_FLOUNDMAT            ; Report MTH$_FLOUNDMAT
        00000000'GF      01  FB  0150   328          CALLS   #1, G^MTH$$SIGNAL       ; Signal the condition
                      04  0157   329 NO_FU:    RET
                          0158   330
                          0158   331          .END
```

```
DIVIDEND          =  00000004
DIVISOR           =  00000008
DONE                 00000067  R      02
ERROR                00000068  R      02
EXP_52            =  00000340
HIGH_MASK         =  FFFF07FF
MTH$$SIGNAL          ********    X    00
MTH$GMOD             00000008  RG     02
MTH$K_FLOUNDMAT      ********    X    00
MTH$K_INVARGMAT      ********    X    00
NO_FU                00000157  R      02
RETURN               00000142  R      02
SF$V_FU           =  00000006
SF$W_SAVE_PSW     =  00000004
STEP_2               0000002C  R      02
STEP_3               00000078  R      02
STEP_5               000000CB  R      02
STEP_9               0000012C  R      02
TEST_SIGN            00000138  R      02
TWO_EXP_52           00000000  R      02
UNDERFLOW            00000143  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	00000158 (344.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.11	00:00:00.36
Command processing	103	00:00:00.50	00:00:03.40
Pass 1	116	00:00:01.34	00:00:05.30
Symbol table sort	0	00:00:00.02	00:00:00.02
Pass 2	71	00:00:00.77	00:00:02.99
Symbol table output	3	00:00:00.03	00:00:00.03
Psect synopsis output	2	00:00:00.02	00:00:00.11
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	326	00:00:02.81	00:00:12.24

```
The working set limit was 900 pages.
6542 bytes (13 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 48 non-local and 0 local symbols.
331 source lines were read in Pass 1, producing 13 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$:MTHGMOD/OBJ=OBJ$:MTHGMOD MSRC$:MTHGMOD/UPDATE=(ENH$:MTHGMOD)
```

MTHGCONJG
LIS

MTHGINT
LIS

MTHGMOD
LIS

MTHEXP
LIS

MTHFLOOR
LIS

MTHDTAN
LIS

MTHDTANH
LIS

MTHGEXP
LIS

MTHGFLOOR
LIS

MTHGMIN1
LIS

MTHGCOSH
LIS

MTHGLOG
LIS

MTHGACOS
LIS

MTHGASIN
LIS

MTHGNINT
LIS

MTHGATAN
LIS

MTHGATANH
LIS

MTHGMAX1
LIS