```
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       HHHHHHHHHHHHHHH  RRRRRRRRRRR         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    AAAAAA    MM      MM    OOOOOO    DDDDDDDD
MM      MM  TTTTTTTTTT  HH      HH    AAAAAA    MM      MM    OOOOOO    DDDDDDDD
MMMM  MMMM      TT      HH      HH   AA    AA   MMMM  MMMM   OO    OO  DD      DD
MMMM  MMMM      TT      HH      HH   AA    AA   MMMM  MMMM   OO    OO  DD      DD
MM  MM  MM      TT      HH      HH   AA    AA   MM  MM  MM   OO    OO  DD      DD
MM  MM  MM      TT      HH      HH   AA    AA   MM  MM  MM   OO    OO  DD      DD
MM      MM      TT      HHHHHHHHHH   AA    AA   MM      MM   OO    OO  DD      DD
MM      MM      TT      HHHHHHHHHH   AA    AA   MM      MM   OO    OO  DD      DD
MM      MM      TT      HH      HH   AAAAAAAAAA   MM      MM   OO    OO  DD      DD
MM      MM      TT      HH      HH   AAAAAAAAAA   MM      MM   OO    OO  DD      DD
MM      MM      TT      HH      HH   AA    AA   MM      MM   OO    OO  DD      DD
MM      MM      TT      HH      HH   AA    AA   MM      MM   OO    OO  DD      DD
MM      MM      TT      HH      HH   AA    AA   MM      MM    OOOOOO   DDDDDDDD
MM      MM      TT      HH      HH   AA    AA   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$AMOD
0000     2          .IDENT  /3-001/                 ; File: MTHAMOD.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$AMOD:
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: Bob Hanek, CREATION DATE: 21-DEC-1982
0000    43 ;
0000    44 ; MODIFIED BY:
0000    45 ;       Jeffrey C. Wiener, 29-DEC-82
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 a complete rewrite          JCW  29-DEC-82
0000    54 ;
```

```
                        .SBTTL  DECLARATIONS
                ;
                ; INCLUDE FILES:
                ;
                ;       NONE
                ;
                ; EXTERNAL SYMBOLS:
                ;
                        .DSABL  GBL             ; Force all external symbols to be declared
                        .EXTRN  MTH$$SIGNAL
                        .EXTRN  MTH$K_FLOUNDMAT
                        .EXTRN  MTH$K_INVARGMAT
                ;
                ; LIBRARY MACROS CALLS:
                ;
                        $SFDEF                          ; Define SF$ (stack frame) symbols
                ;
                ; EQUATED SYMBOLS:
                ;
                ;       NONE
                ;
                ; OWN STORAGE:
                ;
                ;       NONE
                ;
                ; PSECT DECLARATIONS:
                ;
00000000        .PSECT  _MTH$CODE               PIC, SHR, LONG, EXE, NOWRT

                ;
                ; CONSTANTS:
                ;
                ;       NONE
                ;
```

```
                        .SBTTL  MTH$AMOD - F REAL*4 remainder
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       Return the remainder of arg1/arg2 in F_floating point format
                ;       Remainder = arg1 - (int(arg1/arg2))*arg2
                ;
                ; The algorithm used to evaluate the AMOD function is as follows:
                ;
                ;               X = the first argument.
                ;               Y = the second argument.
                ;       step 1. m = the exponent of Y.
                ;               n = the exponent of X.
                ;               c = n - m
                ;               If c < 0, end with result = X.
                ;       step 2. I = the fractional part of X*2^24
                ;               J = the fractional part of Y*2^24
                ;               If I >= J, I = I - J
                ;       step 3. c = c - 31
                ;               If c < 0 go to step 7.
                ;       step 4. L = 2^31*I
                ;               I = L - J*int(L/J)
                ;               c = c - 31
                ;               If c >= 0 go to step 4.
                ;       step 5. c = c + 31
                ;               If c >= 0 go to step 7.
                ;       step 6. L = 2^c*I
                ;               I = L - J*int(L/J)
                ;       step 7. Result = 2^(m-24) * I
                ;
                ; CALLING SEQUENCE:
                ;
                ;       Remainder.wf.v = MTH$AMOD (dividend.rf.r, divisor.rf.r)
                ;
                ; INPUT PARAMETERS:
                ;
                ;       The two input parameters are F_floating-point values. They are
                ;       passed by reference.
                ;
00000004        DIVIDEND = 4                            ; Dividend = X in the algorithm.
00000008        DIVISOR  = 8                            ; Divisor  = Y in the algorithm.
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       NONE
                ;
                ; FUNCTION VALUE:
                ;
                ;       Remainder of the division of arg1/arg2 is returned as an
                ;       F_floating point value.
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       NONE
                ;
                ; COMPLETION CODES:
                ;
```

```
                                0000   147 ;       NONE
                                0000   148 ;
                                0000   149 ; SIDE EFFECTS:
                                0000   150 ;
                                0000   151 ;       MTH$_INVARGMAT - Invalid argument to math library if the divisor is zero.
                                0000   152 ;       MTH$_FLOUNDMAT - Floating underflow in math library is signaled if
                                0000   153 ;          the FU bit is set in the callers PSL.
                                0000   154 ;
                                0000   155 ;--
                                0000   156
                         001C   0000   157           .ENTRY MTH$AMOD,        ^M<R2, R3, R4>
                                0002   158
              52    08 BC      50   0002   159           MOVF    @DIVISOR(AP), R2                ; R2 = Y, the divisor
           52    8000 8F      AA   0006   160           BICW2   #^X8000, R2                     ; R2 = |Y|
                      10      12   000B   161           BNEQ    START                           ; |Y| <> 0
           50    01    0F      78   000D   162           ASHL    #15, #1, R0                     ; |Y|=0. Division by zero case
             7E    00'8F      9A   0011   163           MOVZBL  #MTH$K_INVARGMAT, -(SP)         ; error code
     00000000'GF    01      FB   0015   164           CALLS   #1, G^MTH$$SIGNAL               ; signal the error
                    04      04   001C   165           RET
                            001D   166
50    04 BC    00008000 8F    CB   001D   167 START:  BICL3   #^X8000, @DIVIDEND(AP), R0      ; R0 = |X|
                            0026   168
54    52    FFFF007F 8F    CB   0026   169           BICL3   #^XFFFF007F, R2, R4             ; R4 = m the exponent of Y
53    50    FFFF007F 8F    CB   002E   170           BICL3   #^XFFFF007F, R0, R3             ; R3 = n the exponent of X
            53    54      C2   0036   171           SUBL2   R4, R3                          ; R3 = c = m-n
                            0039   172                                                   ; plus some low order bits
                    69      19   0039   173           BLSS    GET_SIGN                        ; If c<0 then |X| > |Y| and the
                            003B   174                                                   ;   result is X
                            003B   175 ;+
                            003B   176 ;
                            003B   177 ;       STEP_2
                            003B   178 ;       Extract the fraction part of X*2^24, called I, and the
                            003B   179 ;       fractional part of Y*2^24, called J.
                            003B   180 ;
                            003B   181 ;       After the exponent bits are removed from the internal F_floating
                            003B   182 ;       point representation, the hidden bit needs to be added into the
                            003B   183 ;       internal representation since the number is to be converted to
                            003B   184 ;       an integer value.
                            003B   185 ;
                            003B   186 ;-
                            003B   187
     50    7F80 8F      AA   003B   188           BICW    #^X7F80, R0                     ; Clear the exponent field
50     00000080 8F      C0   0040   189           ADDL2   #^X80, R0                       ; Replace hidden bit
     50    50    10      9C   0047   190           ROTL    #16, R0, R0                     ; Convert to integer (R0 = I)
                            004B   191
     52    7F80 8F      AA   004B   192           BICW    #^X7F80, R2                     ; Clear the exponent field
52     00000080 8F      C0   0050   193           ADDL2   #^X80, R2                       ; Replace hidden bit
     52    52    10      9C   0057   194           ROTL    #16, R2, R2                     ; Convert to integer (R2 = J)
                            005B   195
           52    50      D1   005B   196           CMPL    R0, R2                          ; Compare I and J
                 03      19   005E   197           BLSS    STEP_3                          ; Branch if I < J
           50    52      C2   0060   198           SUBL2   R2, R0                          ; I <-- I - J
                            0063   199
                            0063   200 ;+
                            0063   201 ;
                            0063   202 ;       STEP_3
                            0063   203 ;       Convert c = exponent of X - exponent of Y into an integer.
```

```
                                0063   204 ;       Subtract 31 from c in order to determine if an iteration
                                0063   205 ;       of the algorithm is needed. If c-31>=0 then go to STEP_5.
                                0063   206 ;
                                0063   207 ;-
                                0063   208
        53   53    F9 8F   9C  0063   209 STEP_3: ROTL    #-7, R3, R3                    ; Convert c to an integer value
                  53 1F   A2  0068   210         SUBW    #31, R3                        ; Check shift count, c = c-31
                     1A   19  006B   211         BLSS    STEP_5                         ; Branch, if c < 0
                                006D   212
                                006D   213 ;+
                                006D   214 ;
                                006D   215 ;       STEP_4
                                006D   216 ;       Compute I = L - J*int(2^c*I/J) by rem(2^c*I, J) since I and
                                006D   217 ;       J were scaled to integer values.
                                006D   218 ;
                                006D   219 ;-
                                006D   220
        51   50    FF 8F   9C  006D   221 STEP_4: ROTL    #-1, R0, R1                    ; R0/R1=2^31*I. This and the next
  50    51   7FFFFFFF 8F   CB  0072   222         BICL3   #^X7FFFFFFF, R1, R0            ; two instructions are equivalent
                                007A   223                                                ; to ASHQ #31,R0,R0, but are faster
             51   50    C2  007A   224         SUBL2   R0, R1                         ; R0/R1 contains L = 2^31*I
        50   51   50    52  7B  007D   225         EDIV    R2, R0, R1, R0                 ; R0 = rem(2^31*I,J)
             53   1F    A2  0082   226         SUBW2   #31, R3                        ; Check shift count, c = c-31
                  E6    18  0085   227         BGEQ    STEP_4                         ; Branch if c >=0
                                0087   228
             53   1F    A0  0087   229 STEP_5: ADDW2   #31, R3                        ; Restore shift count, c = c+31
                  0B    13  008A   230         BEQL    STEP_7                         ; If zero, branch to STEP_7
                                008C   231 ;+
                                008C   232 ;
                                008C   233 ;       STEP_6
                                008C   234 ;       Compute I = L - J*int(2^c*I/J) by rem(2^c*I, J) since I and
                                008C   235 ;       J were scaled to integer values.
                                008C   236 ;
                                008C   237 ;-
             51   D4    008C   238         CLRL    R1                             ;
     50      50   53    79  008E   239         ASHQ    R3, R0, R0                     ; R0 = 2^c*I
  50 51      50   52    7B  0092   240         EDIV    R2, R0, R1, R0                 ; R0 = rem(2^c*I, J)
                                0097   241
             50   50    4E  0097   242 STEP_7: CVTLF   R0, R0                         ; Convert I to floating point
        50   4C00 8F    A2  009A   243         SUBW2   #^X4C00, R0                    ; R0 = 2^(-24) * I
             50   54    A0  009F   244         ADDW2   R4, R0                         ; R0 = 2^(m-24) * I
                  09    19  00A2   245         BLSS    UNDERFLOW                      ; Branch if underflow occured
                                00A4   246 GET_SIGN:
           04 BC   B5  00A4   247         TSTW    @DIVIDEND(AP)                  ; Check for sign of result
                  03    18  00A7   248         BGEQ    RETURN                         ;  and adjust answer accordingly
             50   50    52  00A9   249         MNEGF   R0, R0                         ;
                  04    00AC   250 RETURN: RET
                                00AD   251
                                00AD   252 UNDERFLOW:
             50   D4    00AD   253         CLRL    R0                             ; set up default result to 0.0
     0D 04 AD    06    E1  00AF   254         BBC     #SF$V_FU, SF$W_SAVE_PSW(FP), NO_FU
                                00B4   255                                                ; Branch if caller has not enabled F
        00000000'8F      DD  00B4   256         PUSHL   #MTH$K_FLOUNDMAT               ; Report MTH$_FLOUNDMAT
  00000000'GF    01      FB  00BA   257         CALLS   #1, G^MTH$$SIGNAL              ; Signal the error
                  04    00C1   258 NO_FU:  RET                                    ; Return
                                00C2   259
                                00C2   260         .END
```

```
DIVIDEND        =  00000004
DIVISOR         =  00000008
GET_SIGN           000000A4  R      02
MTH$$SIGNAL        ********    X    00
MTH$AMOD           00000000  RG     02
MTH$K_FLOUNDMAT    ********    X    00
MTH$K_INVARGMAT    ********    X    00
NO_FU              000000C1  R      02
RETURN             000000AC  R      02
SF$V_FU         =  00000006
SF$W_SAVE_PSW   =  00000004
START              0000001D  R      02
STEP_3             00000063  R      02
STEP_4             0000006D  R      02
STEP_5             00000087  R      02
STEP_7             00000097  R      02
UNDERFLOW          000000AD  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	000000C2 (194.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.10	00:00:01.11
Command processing	122	00:00:00.50	00:00:03.31
Pass 1	115	00:00:01.17	00:00:05.46
Symbol table sort	0	00:00:00.02	00:00:00.05
Pass 2	56	00:00:00.60	00:00:03.58
Symbol table output	3	00:00:00.02	00:00:00.03
Psect synopsis output	2	00:00:00.03	00:00:00.05
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	330	00:00:02.44	00:00:13.59

```
The working set limit was 1050 pages.
5219 bytes (11 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 44 non-local and 0 local symbols.
260 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$:MTHAMOD/OBJ=OBJ$:MTHAMOD MSRC$:MTHAMOD/UPDATE=(ENH$:MTHAMOD)
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MTH4OVPI
LIS

MTHABS
LIS

MTHAINT
LIS

MTHAMOD
LIS

MTHERR
SDL

MTHASIN
LIS

MTHCDABS
LIS

MTHATAN
LIS

MTHATANH
LIS

MTHCDLOG
LIS

MTHJACKET
MAR

MTHBITOPS
LIS

MTHALOG
LIS

MTHDEF
FOR

MTHANINT
LIS

MTHCABS
LIS

MTHACOS
LIS

MTHCDEXP
LIS