

[illegible]

3

Sy

MT

MT

MT

MT

MT

MT
MTMT
MTMT
MTMT
MTMT
MT

MT
MT

MT
MT

MT
MT

MT
MT

MT
MT

MT
MT

MT
MT

MT
MT

MT

MT
MT

MT
MTMT
MT

M1

MI
MIMI
MIM1
M1

10

1

1

10



```

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
LLLLLLLLLLLL IIIIII          SSSSSSSS
LLLLLLLLLLLL IIIIII          SSSSSSSS

```

(1) 49
(2) 55
(3) 90

HISTORY
DECLARATIONS
MTH\$HMOD - H REAL*16 remainder

; Detailed Current Edit History


```

0000 1      .TITLE MTH$HMOD
0000 2      .IDENT /3-002/
0000 3      ; File: MTHHMOD.MAR Edit: JCW3002
0000 4
0000 5      :*****
0000 6      :
0000 7      :*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8      :*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9      :*  ALL RIGHTS RESERVED.
0000 10     :
0000 11     :*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12     :*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13     :*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14     :*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15     :*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16     :*  TRANSFERRED.
0000 17     :
0000 18     :*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19     :*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20     :*  CORPORATION.
0000 21     :
0000 22     :*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23     :*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24     :
0000 25     :*****
0000 26     :
0000 27     :
0000 28     :
0000 29     :++
0000 30     : FACILITY: MATH LIBRARY
0000 31     :
0000 32     : ABSTRACT:
0000 33     :
0000 34     :   This module contains the routine MTH$HMOD:
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
0000 50     :
0000 51     : 3-001 Original version of complete re-write
0000 52     : 3-002 DIVIDEND changed to equal 8 and DIVISOR changed to equal 4. JCW 14-Jun-83
0000 53     :

```

; Detailed Current Edit History

JCW 21-DEC-82

JCW 14-Jun-83

DECLARATIONS

```
0000 55 .SBTTL DECLARATIONS
0000 56 :
0000 57 : INCLUDE FILES:
0000 58 :
0000 59 : NONE
0000 60 :
0000 61 : EXTERNAL SYMBOLS:
0000 62 :
0000 63 : .DSABL GBL ; Force all external symbols to be declared
0000 64 : .EXTRN MTH$$SIGNAL
0000 65 : .EXTRN MTH$K_FLOUNDMAT
0000 66 : .EXTRN MTH$K_INVARGMAT
0000 67 :
0000 68 : LIBRARY MACROS CALLS:
0000 69 :
0000 70 : $SFDEF ; Define SF$ (stack frame) symbols
0000 71 :
0000 72 : EQUATED SYMBOLS:
0000 73 :
00000070 0000 74 : EXP_112 = ^X00000070 ; 112*2^0
FFFF01FF 0000 75 : HIGH_MASK = ^XFFFF01FF
0000 76 :
0000 77 : OWN STORAGE:
0000 78 :
0000 79 : NONE
0000 80 :
0000 81 : PSECT DECLARATIONS:
0000 82 :
00000000 0000 83 : .PSECT _MTH$CODE PIC, SHR, LONG, EXE, NOWRT
0000 84 :
0000 85 : CONSTANTS:
0000 86 :
00000000 00000000 00000000 00004071 0000 87 TWO_EXP_112:
0000 88 : .LONG ^X00004071, ^X0, ^X0, ^X0 ; 2**112
```


MTH\$HMOD - H REAL*16 remainder

```
0010 90 .SBTTL MTH$HMOD - H REAL*16 remainder
0010 91 :++
0010 92 : FUNCTIONAL DESCRIPTION:
0010 93 :
0010 94 : Return the remainder of arg1/arg2 in H_floating point format
0010 95 : Remainder = arg1 - (int(arg1/arg2))*arg2
0010 96 :
0010 97 : The algorithm used to evaluate the HMOD function is as follows:
0010 98 :
0010 99 : X = the first argument.
0010 100 : Y = the second argument.
0010 101 : step 1. m = the exponent of Y.
0010 102 : n = the exponent of X.
0010 103 : c = n - m
0010 104 : If c < 0, end with result = X.
0010 105 : step 2. I = the fractional part of X.
0010 106 : J = the fractional part of Y.
0010 107 : If I >= J, I = I - J
0010 108 : Go to step 5.
0010 109 : step 3. L = 2^(p-1)*I, where p = 113 for H_floating numbers.
0010 110 : step 4. T = L/J
0010 111 : T = [T+2^(p-1)]-2^(p-1). T is int(L/J) or int(L/J)+1
0010 112 : I = L - J * T
0010 113 : If I < 0, I = I + J T was int(L/J)+1
0010 114 : step 5. c = c - (p-1)
0010 115 : If c > 0 go to step 3.
0010 116 : step 6. If c = -(p-1) go to step 9.
0010 117 : step 7. L = 2^(p-1+c) * I
0010 118 : step 8. I = L - J * T
0010 119 : step 9. Result = 2^m * I
0010 120 :
0010 121 : CALLING SEQUENCE:
0010 122 :
0010 123 : CALL MTH$HMOD (remainder.wh.r, dividend.rh.r, divisor.rh.r)
0010 124 :
0010 125 : INPUT PARAMETERS:
0010 126 :
0010 127 : The two input parameters are H_floating-point values.
0010 128 :
00000008 0010 129 : DIVIDEND = 8 ; Dividend = X in the algorithm.
0000000C 0010 130 : DIVISOR = 12 ; Divisor = Y in the algorithm.
0010 131 :
0010 132 : IMPLICIT INPUTS:
0010 133 :
0010 134 : NONE
0010 135 :
0010 136 : OUTPUT PARAMETERS:
0010 137 :
0010 138 : Remainder is the remainder of the division of
0010 139 : arg1/arg2, returned as an H_floating point value.
0010 140 :
0010 141 : IMPLICIT OUTPUTS:
0010 142 :
0010 143 : NONE
0010 144 :
0010 145 : COMPLETION CODES:
0010 146 :
```

MTH\$HMOD - H REAL*16 remainder

```
0010 147 : NONE
0010 148 :
0010 149 : SIDE EFFECTS:
0010 150 :
0010 151 : MTH$_INVARGMAT - Invalid argument to math library if the divisor is zero.
0010 152 : MTH$_FLOUNDMAT - Floating underflow in math library is signaled if
0010 153 : the FU bit is set in the callers PSL.
0010 154 :
0010 155 :--
0010 156 :
0010 157 :.ENTRY MTH$HMOD, ^M<R2, R3, R4, R5, R6, R7, R8>
0012 158 :
0012 159 :MOVH @DIVISOR(AP), R4 ; R4/R7 = Y
0017 160 :BEQL ERROR ; Y=0. Division by zero
0019 161 :MOVH @DIVIDEND(AP), R0 ; R0/R3 = X
001E 162 :
001E 163 :BICL3 #^XFFFF8000, R4, -(SP) ; (SP)=m is the biased exponent of Y
0026 164 :BICL3 #^XFFFF8000, R0, R8 ; R8=n is the biased exponent of X
002E 165 :
002E 166 :SUBL2 (SP), R8 ; R8 = c = n-m unbiased
0031 167 :BGEQ STEP_2 ; Result is X if X<Y, ie, if c<0
0033 168 :MOVO R0, @4(AP) ; @4(AP) = X
0038 169 :RET
0039 170 :
0039 171 :STEP_2: MOVW #^X4000, R4 ; R4/R7 = J = biased !fract(Y)!
003E 172 :
003E 173 :MOVW #^X4000, R0 ; R0/R3 = I = biased !fract(X)!
0043 174 :
0043 175 :
0043 176 :+
0043 177 :
0043 178 :In STEP_4 and STEP_8 the calculation of I = L - J*int(L/J) must be
0043 179 :computed as precisely as possible. To do this we will need to write J as
0043 180 :J = J1 + J2
0043 181 :where J1 = the high 56 bits of J and J2 = J - J1, the low 57 bits of J.
0043 182 :
0043 183 :HIGH_MASK is used to extract the 7 bits of J from longword3 that belong
0043 184 :to JT.
0043 185 :
0043 186 :-
0043 187 :
0043 188 :
0043 189 :MOVO R4, -(SP) ; (SP) = J
0047 190 :BICL #HIGH_MASK, 8(SP)
004F 191 :CLRL 12(SP) ; (SP) = J1 replaces the value
0052 192 : ; of J on the top of SP
0052 193 :SUBH3 (SP), R4, -(SP) ; (SP) = J2 = J - J1
0057 194 :
0057 195 :CMPH R0, R4 ; If I<J
005B 196 :BGEQ I_GEQ_J
005D 197 :BRW STEP_5 ; go to STEP_5
0060 198 :I_GEQ_J:
0060 199 :SUBH2 R4, R0 ; else I = I-J
0064 200 :BLEQ OVER
0066 201 :BRW STEP_5 ; go to STEP_5 if I>0, or
0069 202 : ; else the algorithm ends
0069 203 :OVER: TSTW @DIVIDEND(AP) ; the sign of the result is
```


MTH\$HMOD - H REAL*16 remainder

```

      04 18 006C 204      BGEQ  DONE      ; the same as the sign of
      50 50 72FD 205      MNEGH  R0, R0    ; the first argument, A.
04 BC 50 7DFD 206 DONE:  MOVQ  R0, @4(AP)  ; Return the correct result
      04 0077 207      RET
      50 01 0F 79 0078 208      ERROR:  ASHQ  #15, #1, R0      ; Y=0. Reserved operand
      52 7C 007C 210      CLRL  R2
      7E 00'8F 9A 007E 211      MOVZBL #MTH$K INVARGMAT, -(SP) ; error code
00000000'GF 01 FB 0082 212      CALLS  #1, G^MTH$$SIGNAL      ; signal the error
      04 BC 50 7DFD 0089 213      MOVQ  R0, @4(AP)            ; Return the correct result
      04 008E 214      RET
      50 00000070 8F C0 008F 215 STEP_3: ADDL2  #EXP_112, R0      ; R0/R3 = L = 2**(p-1)*I
      0096 216
      0096 217
      0096 218
      0096 219 ;+
      0096 220
      0096 221 STEP_4:
      0096 222 2^(p-1) = 2^(112) is added and then subtracted from
      0096 223 T = int(L/J) to ensure that T = chopped(L/J) or chopped(L/J)+1
      0096 224
      0096 225 ;--
      0096 226
      54 7E 54 7DFD 0096 227      MOVQ  R4, -(SP)                ; save J for use in STEP_5
      54 50 54 67FD 009A 228      DIVH3  R4, R0, R4              ; R4/R7 = T = L/J
      54 FF5C CF 60FD 009F 229      ADDH2  TWO_EXP_112, R4      ; R4/R7 = T = T+2**(p-1)
      54 FF56 CF 62FD 00A5 230      SUBH2  TWO_EXP_112, R4      ; T-2**(p-1) = L/J chopped
      00AB 231 ; or L/J chopped + 1
      00AB 232
      00AB 233 ;+
      00AB 234
      00AB 235 The calculation of I = L - J*int(L/J) must be computed as precisely
      00AB 236 as possible. To do this we will need to write T as
      00AB 237 T = Z1 + Z2
      00AB 238 where Z1 = the high 56 bits of T and Z2 = T - Z1, the low 57 bits of T.
      00AB 239 Now, using J = J1 + J2,
      00AB 240
      00AB 241 L - J * int(L/J) = L - (J1 + J2) * (Z1 + Z2)
      00AB 242 = L - (Z1 * J1) - (Z1 * J2)
      00AB 243 = L - (Z2 * J1) - (Z2 * J2)
      00AB 244 = L - (Z1 * J) - (Z2 * J)
      00AB 245
      00AB 246
      00AB 247 ;--
      00AB 248
      08 AE FFFF01FF 8F CA 00AB 249      MOVQ  R4, -(SP)                ; Stack Z = INT(L/J)
      0C AE D4 00AF 250      BICL  #HIGH_MASK, 8(SP)            ; Start to form Z1
      7E 54 6E 63FD 00BA 251      CLRL  12(SP)                  ; (SP) = Z1
      54 10 AE 40 AE 65FD 00BF 252      SUBH3  (SP), R4, -(SP)      ; (SP) = Z2
      50 54 62FD 00C6 253      MULH3  64(SP), 16(SP), R4        ; Compute Z1*J1
      54 30 AE 10 AE 65FD 00CA 254      SUBH2  R4, R0              ; R0/R3 = L - Z1*J1
      50 54 62FD 00D1 255      MULH3  16(SP), 48(SP), R4        ; R4/R7 = Z1*J2
      54 40 AE 6E 65FD 00D5 256      SUBH2  R4, R0              ; R0/R3 = L - Z1*J
      50 54 62FD 00DB 257      MULH3  (SP), 64(SP), R4          ; R4/R7 = Z2*J1
      54 30 AE 6E 65FD 00DF 258      SUBH2  R4, R0              ; R0/R3 = L - Z1*J - Z2*J1
      50 54 62FD 00E5 259      MULH3  (SP), 48(SP), R4          ; R4/R7 = Z2*J2
      50 54 62FD 00E5 260      SUBH2  R4, R0              ; R0/R3 = L - Z*J
```


MTH\$HMOD - H REAL*16 remainder

```

5E 20 C0 00E9 261 ADDL2 #32, SP ; remove Z1,Z2,J1,J2 from stack
54 8E 7DFD 00EC 262 MOVO (SP)+, R4 ; R4/R7 = J
50 B5 00F0 263 TSTW R0
09 14 00F2 264 BGTR STEP_5 ; If R0/R3>0 the algoritm cotinues
03 19 00F4 265 BLSS SUBTRACT_J
0093 31 00F6 266 BRW RETURN ; If R0/R3=0 the algorithm ends
50 54 60FD 00F9 267 SUBTRACT_J: ; Add J back in because you had
00FD 268 ADDH2 R4, R0 ; T = chopped(L/J)+1
00FD 269
00FD 270
58 00000070 8F C2 00FD 271 STEP_5: SUBL2 #EXP_112, R8 ;c = c-(p-1) = c-112
89 18 0104 272 BGEQ STEP_3
58 00000070 8F C0 0106 273 ADDL2 #EXP_112, R8 ; c = (p-1)+c = 112+c
010D 274
010D 275 ;+
010D 276
010D 277 The next two lines of code are STEP_6 and STEP_7.
010D 278
010D 279 ;+
010D 280
50 67 13 010D 281 BEQL STEP_9
58 C0 010F 282 ADDL2 R8, R0 ; L = I*2^(c+t)
0112 283
0112 284 ;+
0112 285 2^(p-1) = 2^(55) is added and then subtracted from
0112 286 T = int(L/J) to ensure that T = chopped(L/J) or chopped(L/J)+1
0112 287 ;+
0112 288
54 7E 54 7DFD 0112 289 MOVO R4, -(SP) ; Save J
54 50 54 67FD 0116 290 DIVH3 R4, R0, R4 ; R4/R7 = T = L/J
54 FEEO CF 60FD 011B 291 ADDH2 TWO_EXP_112, R4 ; R4/R7 = T = T+2**(p-1)
54 FEDA CF 62FD 0121 292 SUBH2 TWO_EXP_112, R4 ; T-2**(p-1) = L/J chopped
0127 293 ; or L/J chopped + 1
0127 294 ;+
0127 295
0127 296 STEP_8:
0127 297 The calculation of I = L - J*int(L/J) must be computed as precisely
0127 298 as possible. To do this we will need to write T as
0127 299 T = Z1 + Z2
0127 300 where Z1 = the high 56 bits of T and Z2 = T - Z1, the low 57 bits of T.
0127 301 Now, using J = J1 + J2,
0127 302
0127 303 L - J * int(L/J) = L - (J1 + J2) * (Z1 + Z2)
0127 304 = L - (Z1 * J1) - (Z1 * J2)
0127 305 = L - (Z2 * J1) - (Z2 * J2)
0127 306 = L - (Z1 * J) - (Z2 * J)
0127 307
0127 308
0127 309
0127 310 ;+
0127 311
0127 312
08 AE 7E 54 7DFD 0127 312 MOVO R4, -(SP) ; Stack Z = INT(L/J)
FFFF01FF 8F CA 012B 313 BICL #HIGH_MASK, 8(SP) ; Start to form Z1
0C AE D4 0133 314 CLRL 12(SP) ; (SP) = Z1
7E 54 6E 63FD 0136 315 SUBH3 (SP), R4, -(SP) ; (SP) = Z2
54 10 AE 40 AE 65FD 013B 316 MULH3 64(SP), 16(SP), R4 ; Compute Z1*J1
50 54 62FD 0142 317 SUBH2 R4, R0 ; R0/R3 = L - Z1*J1
```

```
54 30 AE 10 AE 65FD 0146 318 MULH3 16(SP), 48(SP), R4 ; R4/R7 = Z1*J2
54 50 54 62FD 014D 319 SUBH2 R4, R0 ; R0/R3 = L - Z1*J
54 40 AE 6E 65FD 0151 320 MULH3 (SP), 64(SP), R4 ; R4/R7 = Z2*J1
54 50 54 62FD 0157 321 SUBH2 R4, R0 ; R0/R3 = L - Z1*J - Z2*J1
54 30 AE 6E 65FD 015B 322 MULH3 (SP), 48(SP), R4 ; R4/R7 = Z2*J2
54 50 54 62FD 0161 323 SUBH2 R4, R0 ; R0/R3 = L - Z*J
54 5E 20 C0 0165 324 ADDL2 #32, SP ; Remove Z1 and Z2 from the stack
54 8E 7DFD 0168 325 MOVO (SP)+, R4 ; Restore J
54 50 B5 016C 326 TSTW R0
54 06 14 016E 327 BGTR STEP 9
54 1A 13 0170 328 BEQL RETURN
54 50 54 60FD 0172 329 ADDH2 R4, R0 ; If R0/R3=0 the algorithm ends
; Add J back in because you had
; T = chopped(L/J)+1
20 AE 4000 8F A2 0176 331 STEP_9: SUBW2 #^X4000, 32(SP) ; Remove the bias from m
54 50 20 AE A0 017C 333 ADDW2 32(SP), R0 ; and form R0/R1 = 2^m*L
54 10 19 0180 334 BLSS UNDERFLOW
54 08 BC B5 0182 335 TEST_SIGN:
54 05 18 0182 336 TSTW @DIVIDEND(AP) ; the sign of the result is
54 8000 8F A8 0185 337 BGEQ RETURN ; the same as the sign of
54 04 BC 50 7DFD 0187 338 BLSW2 #^X8000, R0 ; the first argument, X.
54 04 04 018C 339 RETURN: MOVO R0, @4(AP) ; Return the correct result
54 04 04 0191 340 RET
54 04 04 0192 341 UNDERFLOW:
54 04 BC 7CFD 0192 342 CLRO @4(AP) ; Set up default result to 0.0
54 04 AD 06 E1 0196 343 BBC #SF$V_FU, SF$W_SAVE_PSW(FP), NO_FU
54 00000000'8F DD 019B 344 PUSHL #MTH$K_FLOUNDMAT ; Branch if caller has not enabled F
54 00000000'GF 01 FB 019B 345 CALLS #1, G^MTH$$SIGNAL ; Report MTH$_FLOUNDMAT
54 01A8 347 NO_FU: RET ; Signal the condition
54 01A9 348 RET ; Return
54 01A9 349 .END
54 01A9 350
54 01A9 351
```


MTH\$HMOD
Symbol table

H 5

16-SEP-1984 01:38:00 VAX/VMS Macro V04-00
6-SEP-1984 11:25:13 [MTHRTL.SRC]MTHHMOD.MAR;1

Page 8
(3)

```

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
\$AB\$\$	00000000 (0.)	01 (1.)	NOPIC USR
_MTH\$CODE	000001A9 (425.)	02 (2.)	PIC USR

CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
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

_S255SDUA28:[SYSLIB]STARLET.MLB;2

Macros defined

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