


```

MM      MM      TTTTTTTTTT  HH      HH      HH      HH      SSSSSSSS  QQQQQQ  RRRRRRRR  TTTTTTTTTT
MM      MM      TTTTTTTTTT  HH      HH      HH      HH      SSSSSSSS  QQQQQQ  RRRRRRRR  TTTTTTTTTT
MMM     MMM     TT          HH      HH      HH      HH      SS          QQ      QQ      RR      RR      TT
MMM     MMM     TT          HH      HH      HH      HH      SS          QQ      QQ      RR      RR      TT
MM      MM      TT          HH      HH      HH      HH      SS          QQ      QQ      RR      RR      TT
MM      MM      TT          HH      HH      HH      HH      SS          QQ      QQ      RR      RR      TT
MM      MM      TT          HHHHHHHHHH  HHHHHHHHHH  SSSSSS  QQ      QQ      RRRRRRRR  TT
MM      MM      TT          HHHHHHHHHH  HHHHHHHHHH  SSSSSS  QQ      QQ      RRRRRRRR  TT
MM      MM      TT          HH      HH      HH      HH      SS          QQ  QQ  QQ  RR  RR  TT
MM      MM      TT          HH      HH      HH      HH      SS          QQ  QQ  QQ  RR  RR  TT
MM      MM      TT          HH      HH      HH      HH      SS          QQ  QQ  QQ  RR  RR  TT
MM      MM      TT          HH      HH      HH      HH      SSSSSSSS  QQQQ  QQ  RR      RR  TT
MM      MM      TT          HH      HH      HH      HH      SSSSSSSS  QQQQ  QQ  RR      RR  TT

```

```

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
LLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLL  IIIIII  SSSSSSSS

```

(2)	56	HISTORY	; Detailed Current Edit History
(3)	63	DECLARATIONS	; Declarative Part of Module
(4)	103	MTH\$HSQRT	- Standard H-Floating SQRT
(5)	194	MTH\$HSQRT_R8	- Special HSQRT Routine

```

0000 1      .TITLE  MTH$HSQRT      ; H Floating Point Square Root routine
0000 2      ;                               ; (HSQRT)
0000 3      .IDENT  /1-001/      ; File: MTH$HSQRT.MAR   EDIT: JAW1001
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     ; FACILITY: MATH LIBRARY
0000 30     ;
0000 31     ;++
0000 32     ;
0000 33     ; ABSTRACT:
0000 34     ;
0000 35     ; MTH$HSQRT is a function which returns the H-floating square root of
0000 36     ; its H-floating argument. The call is standard call-by-reference.
0000 37     ;
0000 38     ; MTH$HSQRT_RB is a special routine which is the same as MTH$HSQRT
0000 39     ; except that a faster non-standard JSB call is used with the argument
0000 40     ; in R0/R3 and no registers are saved.
0000 41     ;
0000 42     ;--
0000 43     ;
0000 44     ; VERSION: 1
0000 45     ;
0000 46     ; HISTORY:
0000 47     ;
0000 48     ; AUTHOR:
0000 49     ;      John A. Wheeler, 25-Jul-79: Version 1
0000 50     ;
0000 51     ; MODIFIED BY:
0000 52     ;
0000 53     ;
0000 54     ;

```

MTH\$HSQRT
1-001

L 11
; H Floating Point Square Root routine 16-SEP-1984 01:40:33 VAX/VMS Macro V04-00
HISTORY ; Detailed Current Edit Histor 6-SEP-1984 11:25:45 [MTHRTL.SRC]MTHHSQRT.MAR;1

Page 2
(2)

MTH
1-0

0000 56 .SBTTL HISTORY ; Detailed Current Edit History
0000 57
0000 58 ; Edit History for Version 1 of MTH\$HSQRT
0000 59 ;
0000 60 ; 1-001 - Adapted from MTH\$SQRT version 1-013, MTH\$DSQRT version 1-013,
0000 61 ; and MTH\$GSQRT version 1-001. JAW 25-Jul-79.

```

0000 63      .SBTTL  DECLARATIONS          ; Declarative Part of Module
0000 64
0000 65      :
0000 66      : INCLUDE FILES:
0000 67      :
0000 68      : EXTERNAL SYMBOLS:
0000 69      :
0000 70      .DSABL  GBL                  ; Declare all externals
0000 71      .EXTRN  MTH$$SIGNAL         ; SIGNAL SEVERE error
0000 72      .EXTRN  MTH$K_SQUROONEG    ; Error code
0000 73
0000 74      :
0000 75      : EQUATED SYMBOLS:
0000 76      :
0000 77
0000 78      000041FC ACMASK = ^M<IV, R2, R3, R4, R5, R6, R7, R8>
0000 79      ; Register save mask and IV enable
0000 80
0000 81      :
0000 82      : MACROS:      None
0000 83      :
0000 84      : PSECT DECLARATIONS:
0000 85
0000 86      00000000 .PSECT  _MTH$CODE      PIC,SHR,LONG,EXE,NOWRT
0000 87      ; Program section for math routines
0000 88
0000 89      :
0000 90      : OWN STORAGE:
0000 91      :
0000 92      :
0000 93      : Constants A and B chosen for k odd
0000 94      :
0000 95      13CD4054 LF_ODD_A = ^X13CD4054      ; Decimal: 0.8284271
0000 96      3C4A3F98 LF_ODD_B = ^X3C4A3F98      ; Decimal: 0.2973349
0000 97      :
0000 98      : Constants A and B chosen for k even
0000 99      :
0000 100     F61A4015 LF_EVEN_A = ^XF61A4015     ; Decimal: 0.5857865
0000 101     4B233FD7 LF_EVEN_B = ^X4B233FD7     ; Decimal: 0.4204951

```

```

0000 103      .SBTTL MTH$HSQRT - Standard H-Floating SQRT
0000 104
0000 105
0000 106 :++
0000 107 : FUNCTIONAL DESCRIPTION:
0000 108 :
0000 109 : HSQRT - H-floating point square root function
0000 110 :
0000 111 : HSQRT(X) is computed using the following approximation technique:
0000 112 :
0000 113 :   If X < 0, signal error.  If X = 0, return HSQRT(X) = 0.
0000 114 :
0000 115 :   Let X = 2**K * F where F is the fractional part.
0000 116 :
0000 117 :   If K is even, X = 2**(2P) * F,
0000 118 :                   HSQRT(X) = 2**P * HSQRT(F), 1/2 =< F < 1
0000 119 :
0000 120 :   If K is odd, X = 2**(2P+1) * F = 2**(2P+2) * (F/2),
0000 121 :                   HSQRT(X) = 2**(P+1) * HSQRT(F/2), 1/4 =< F/2 < 1/2.
0000 122 :
0000 123 :   Let F' = A*F + B,
0000 124 :                   A = 0.453730314(octal),
0000 125 :                   B = 0.327226214(octal), for K even.
0000 126 :                   = A*(F/2) + B,
0000 127 :                   A = 0.650117146(octal),
0000 128 :                   B = 0.230170444(octal), for K odd.
0000 129 :
0000 130 :   and
0000 130 :       K' = P,      for K even
0000 131 :           = P + 1  for K odd.
0000 132 :
0000 133 :   Let Y0 = 2**K' * F' as a straight line approximation within the
0000 134 :   given interval using coefficients A and B which minimize the
0000 135 :   absolute error at the midpoint and endpoints.
0000 136 :
0000 137 :   Starting with Y0, five Newton-Raphson iterations are performed.
0000 138 :
0000 139 :   Y[n+1] = (1/2) * ( Y[n] + X/Y[n] )
0000 140 :
0000 141 :   The relative error is < 10**-17.
0000 142 :
0000 143 : CALLING SEQUENCE:
0000 144 :
0000 145 :   hsqrt.wh.v = MTH$HSQRT(x.rh.r)
0000 146 :
0000 147 :   -or-
0000 148 :
0000 149 :   CALL MTH$HSQRT(hsqrt.wh.r, x.rh.r)
0000 150 :
0000 151 :   Because an H-floating result cannot be expressed in 64 bits, it is
0000 152 :   returned as the first argument, with the input parameter displaced
0000 153 :   to the second argument, in accordance with the Procedure Calling
0000 154 :   Standard.
0000 155 :
0000 156 : INPUT PARAMETERS:
0000 157 :
00000004 0000 158 : LONG = 4 ; Define longword multiplier
00000008 0000 159 : x = 2 * LONG ; Contents of x is the argument

```

```

0000 160
0000 161 ; IMPLICIT INPUTS:      None
0000 162 ;
0000 163 ; OUTPUT PARAMETERS:
0000 164 ;
00000004 0000 165 ;          hsqrt = 1 * LONG          ; Contents of hsqrt is the result
0000 166 ;
0000 167 ; IMPLICIT OUTPUTS:      None
0000 168 ;
0000 169 ; COMPLETION CODES:      None
0000 170 ;
0000 171 ; SIDE EFFECTS:
0000 172 ;
0000 173 ; Signals: MTH$_SQURONEG if x < 0.0 with reserved operand in R0/R3
0000 174 ; (R0/R1 copied to the signal mechanism vector CHF$L_MCH_R0/R1 by LIB$SIGNAL).
0000 175 ; Associated message is: "SQUARE ROOT OF NEGATIVE VALUE". Result is reserved
0000 176 ; operand -0.0 unless a user supplied (or any) error handler changes CHF$L_MCH_R0/R1
0000 177 ;
0000 178 ; NOTE: This procedure disables floating point underflow, enables integer
0000 179 ; overflow, causes no floating overflow or other arithmetic traps, and
0000 180 ; preserves enables across the call.
0000 181 ;
0000 182 ;---
0000 183
0000 184
41FC 0000 185 ; .ENTRY MTH$HSQRT, ACMASK ; Standard call-by-reference entry
0002 186 ; ; Disable DV (and FU), enable IV
0002 187 ; MTH$FLAG_JACKET ; Flag that this is a jacket procedure in
6D 00000000'GF 9E 0002 ; MOVAB G^MTH$$JACKET_HND, (FP)
0009 ; ; set handler address to jacket
0009 ; handler
0009
50 08 BC 70FD 0009 188 ; case of an error in special routine
000E 189 ; MOVH @x(AP), R0 ; Fetch x from second argument
04 BC 50 7DFD 0010 190 ; BSBB MTH$HSQRT_R8 ; Call kernel HSQRT routine
0015 191 ; MOVO R0, @hsqrt(AP) ; Store result in first argument
0015 192 ; RET ; Return to caller

```



```

0016 194      .SBTTL MTH$HSQRT_R8 - Special HSQRT Routine
0016 195
0016 196 MTH$HSQRT_R8::
54 50 70FD 0016 197      MOVH   R0, R4      ; Test value of x and save in R4/R7
      03 14 001A 198      BGTR   POS      ; If x > 0, proceed with computation
0098 31 001C 199      BRW    ZERO_NEG ; Otherwise handle separately
001F 200
001F 201      ;
001F 202      ; Here if x > 0.
001F 203      ;
001F 204
22 50 E9 001F 205 POS:   BLBC   R0, EVEN ; Branch on odd/even exponent
0022 206
58 50 3FFF 8F A3 0022 207 ODD:   SUBW3  #^X3FFF, R0, R8 ; Save scale factor in R8
50 50 3FFF 8F B0 0028 208      MOVW   #^X3FFF, R0 ; Scale x
      54 50 B0 002D 209      MOVW   R0, R4 ; Scale copy of x in R4
      50 50 F6FD 0030 210      CVTHF  R0, R0 ; Convert x to single precision
50 13CD4054 8F 44 0034 211      MULF2  #LF_ODD_A, R0 ; Compute initial approximation
50 3C4A3F98 8F 40 003B 212      ADDF2  #LF_ODD_B, R0 ;
      20 11 0042 213      BRB    ITERATE ; Go iterate
0044 214
58 50 4000 8F A3 0044 215 EVEN:  SUBW3  #^X4000, R0, R8 ; Save scale factor in R8
50 50 4000 8F B0 004A 216      MOVW   #^X4000, R0 ; Scale x
      54 50 B0 004F 217      MOVW   R0, R4 ; Scale copy of x in R4
      50 50 F6FD 0052 218      CVTHF  R0, R0 ; Convert x to single precision
50 F61A4015 8F 44 0056 219      MULF2  #LF_EVEN_A, R0 ; Compute first approximation
50 4B233FD7 8F 40 005D 220      ADDF2  #LF_EVEN_B, R0 ;
0064 221
0064 222 ITERATE:
0064 223
0064 224 ;
0064 225 ; first iteration - single precision
0064 226 ;
0064 227
52 51 54 F6FD 0064 228      CVTHF  R4, R1 ; R1 = X
50 51 50 47 0068 229      DIVF3  R0, R1, R2 ; R2 = X/Y0
50 50 52 40 006C 230      ADDF2  R2, R0 ; R0 = Y0 + X/Y0
50 0080 8F A2 006F 231      SUBW2  #^X80, R0 ; R0 = Y1 = (1/2)(Y0 + X/Y0)
0074 232 ; No overflow possible
0074 233
0074 234 ;
0074 235 ; second iteration - single precision
0074 236 ;
0074 237
52 51 50 47 0074 238      DIVF3  R0, R1, R2 ; R2 = X/Y1
50 50 52 40 0078 239      ADDF2  R2, R0 ; R0 = Y1 + X/Y1
50 0080 8F A2 007B 240      SUBW2  #^X80, R0 ; R0 = Y2 = (1/2)(Y1 + X/Y1)
0080 241
0080 242 ;
0080 243 ; third iteration - double precision
0080 244 ;
0080 245
52 54 76FD 0080 246      CVTHG  R4, R2 ; R2/R3 = X
50 50 99FD 0084 247      CVTFG  R0, R0 ; R0/R1 = Y2
52 50 46FD 0088 248      DIVG2  R0, R2 ; R2 = X/Y2
50 52 40FD 008C 249      ADDG2  R2, R0 ; R0 = Y2 + X/Y2
50 10 A2 0090 250      SUBW2  #^X10, R0 ; R0 = Y3 = (1/2)(Y2 + X/Y2)

```

```

0093 251
0093 252
0093 253 : fourth iteration - quadruple precision
0093 254
0093 255
7E 50 50 56FD 0093 256 CVTGH R0, R0 ; R0/R3 = Y3
54 50 67FD 0097 257 DIVH3 R0, R4, -(SP) ; Stack = X/Y3
50 6E 60FD 009C 258 ADDH2 (SP), R0 ; R0 = Y3 + X/Y3
50 B7 00A0 259 DECW R0 ; R0 = Y4 = (1/2)(Y3 + X/Y3)
00A2 260
00A2 261
00A2 262 : fifth iteration - quadruple precision
00A2 263
00A2 264
6E 54 50 67FD 00A2 265 DIVH3 R0, R4, (SP) ; Stack = X/Y4
50 6E 60FD 00A7 266 ADDH2 (SP), R0 ; R0 = Y4 + X/Y4
50 B7 00AB 267 DECW R0 ; R0 = Y5 = (1/2)(Y4 + X/Y4)
00AD 268
00AD 269 ; scale result by multiplying by scale_factor/2
00AD 270
58 02 A6 00AD 271 DIVW2 #2, R8 ; Divide scale factor by 2
50 58 A0 00B0 272 ADDW2 R8, R0 ; Scale the result
5E 10 C0 00B3 273 ADDL2 #16, SP ; Reset stack pointer
00B6 274 SQRX: RSB ; Return with result in R0/R3
00B7 275
00B7 276 ; Here if x <= 0
00B7 277
00B7 278 ZERO_NEG:
7E FD 13 00B7 279 BEQL SQRX ; Return with result = 0
6E DD 00B9 280 PUSHL (SP) ; Return PC from JSB routine
50 01 00'8F 9A 00BB 281 MOVZBL #MTH$K_SQUROONEG, -(SP) ; Condition value
50 01 0F 79 00BF 282 ASHQ #15, #T, R0 ; R0/R3 = result = reserved operand -0.0
00C3 283 ; R0/R1 goes to signal mechanism vector
00C3 284 ; (CHF$L_MCH_R0/R1) so error handler
00C3 285 ; can modify the result.
0000000'GF 52 7C 00C3 286 CLRQ R2
02 FB 00C5 287 CALLS #2, G*MTH$$SIGNAL ; Signal error and use real user's PC
00CC 288 ; Independent of CALL vs JSB
05 00CC 289 RSB ; Return - R0 restored from CHF$L_MCH_R0/R1
00CD 290
00CD 291 .END

```

```

ACMASK      = 000041FC
EVEN        = 00000044 R 01
HSQRT      = 00000004
ITERATE    = 00000064 R 01
LF_EVEN_A  = F61A4015
LF_EVEN_B  = 4B233FD7
LF_ODD_A   = 13CD4054
LF_ODD_B   = 3C4A3F98
LONG       = 00000004
MTH$$JACKET_HND ***** X 01
MTH$$SIGNAL ***** X 00
MTH$HSQRT  = 00000000 RG 01
MTH$HSQRT_R8 = 00000016 RG 01
MTH$K_SQUROONEG ***** X 00
ODD        = 00000022 R 01
POS        = 0000001F R 01
SQRTX     = 000000B6 R 01
X          = 00000008
ZERO_NEG   = 000000B7 R 01
    
```

 ! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes													
. ABS	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE			
_MTH\$CODE	000000CD (205.)	01 (1.)	PIC	USR	CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC	LONG			

 ! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.07	00:00:01.19
Command processing	119	00:00:00.60	00:00:04.17
Pass 1	87	00:00:00.83	00:00:04.03
Symbol table sort	0	00:00:00.00	00:00:00.21
Pass 2	64	00:00:00.72	00:00:02.46
Symbol table output	3	00:00:00.02	00:00:00.02
Psect synopsis output	2	00:00:00.01	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	306	00:00:02.26	00:00:12.11

The working set limit was 900 pages.
 4673 bytes (10 pages) of virtual memory were used to buffer the intermediate code.
 There were 10 pages of symbol table space allocated to hold 19 non-local and 0 local symbols.
 351 source lines were read in Pass 1, producing 11 object records in Pass 2.
 1 page of virtual memory was used to define 1 macro.

! Macro library statistics !

Macro library name

Macros defined

_S255SDUA28:[SYS]IB]STARLET.MLB;2

0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS\$:MTHHSQRT/OBJ=OBJ\$:MTHHSQRT MSRC\$:MTHJACKET/UPDATE=(ENHS:MTHJACKET)+MSRC

A grid of 100 small panels, each containing a different type of data visualization or report. The panels are arranged in a 10x10 grid. Each panel has a title in the top-left corner, such as 'MTHHINT LIS', 'MTHHINT LIS', 'MTHHINT LIS', 'MTHHINT LIS', 'MTHHINT LIS', 'MTHHINT LIS', 'MTHHINT LIS', 'MTHHINT LIS', 'MTHHINT LIS', 'MTHHINT LIS'. The content of the panels varies, including tables of data, bar charts, and text-based reports. The overall appearance is that of a technical manual or a collection of sample outputs from a data processing system.