


```
MM      MM      TTTTTTTTTT  HH      HH      GGGGGGGG  SSSSSSSS  QQQQQQ  RRRRRRRR  TTTTTTTTTT
MM      MM      TTTTTTTTTT  HH      HH      GGGGGGGG  SSSSSSSS  QQQQQQ  RRRRRRRR  TTTTTTTTTT
MMMM    MMMM    TT          HH      HH      GG          SS          QQ      QQ      RR      RR      TT
MMMM    MMMM    TT          HH      HH      GG          SS          QQ      QQ      RR      RR      TT
MM      MM      TT          HH      HH      GG          SS          QQ      QQ      RR      RR      TT
MM      MM      TT          HH      HH      GG          SS          QQ      QQ      RR      RR      TT
MM      MM      TT          HHHHHHHHHH  GG          SSSSSS  QQ      QQ      RRRRRRRR  TT
MM      MM      TT          HHHHHHHHHH  GG          SSSSSS  QQ      QQ      RRRRRRRR  TT
MM      MM      TT          HH      HH      GG  GGGGGG  SS          QQ      QQ      RR      RR      TT
MM      MM      TT          HH      HH      GG  GGGGGG  SS          QQ      QQ      RR      RR      TT
MM      MM      TT          HH      HH      GG          SS          QQ      QQ      RR      RR      TT
MM      MM      TT          HH      HH      GG          SS          QQ      QQ      RR      RR      TT
MM      MM      TT          HH      HH      GG          SS          QQ      QQ      RR      RR      TT
MM      MM      TT          HH      HH      GGGGGG  SSSSSSSS  QQQQ  QQ      RR      RR      TT
MM      MM      TT          HH      HH      GGGGGG  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
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS
```

(2)	53
(3)	68
(4)	109
(5)	190

HISTORY : Detailed Current Edit History
DECLARATIONS : Declarative Part of Module
MTH\$GSQRT - Standard G-Floating GSQRT
MTH\$GSQRT_R5 - Special GSQRT routine

```
0000 1      .TITLE  MTH$GSQRT      ; G Floating Point Square Root routine
0000 2      ;      (GSQRT)
0000 3      .IDENT /1-004/        ; File: MTHGSQRT.MAR      EDIT: RNH1004
0000 4
0000 5      *****
0000 6      *
0000 7      *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8      *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9      *  ALL RIGHTS RESERVED.
0000 10     *
0000 11     *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12     *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13     *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14     *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15     *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16     *  TRANSFERRED.
0000 17     *
0000 18     *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19     *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20     *  CORPORATION.
0000 21     *
0000 22     *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23     *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24     *
0000 25     *
0000 26     *****
0000 27
0000 28
0000 29     FACILITY: MATH LIBRARY
0000 30     ++
0000 31     ABSTRACT:
0000 32
0000 33     MTH$GSQRT is a function which returns the G-floating square root of
0000 34     its G-floating argument. The call is standard call-by-reference.
0000 35
0000 36     MTH$GSQRT_R5 is a special routine which is the same as MTH$GSQRT
0000 37     except a faster non-standard JSB call is used with the argument in
0000 38     R0 and no registers are saved.
0000 39
0000 40     --
0000 41
0000 42     VERSION: 1
0000 43
0000 44     HISTORY:
0000 45     AUTHOR:
0000 46             Steven B. Lionel, 08-Jan-79: Version 1
0000 47
0000 48     MODIFIED BY:
0000 49
0000 50
0000 51
```

```
0000 53      .SBTTL HISTORY ; Detailed Current Edit History
0000 54
0000 55
0000 56 : ALGORITHMIC DIFFERENCES FROM FP-11/C ROUTINE:
0000 57 : \DSQRT used for comparison, FP-11 has no G-float. \
0000 58 : 1. Last DIVD is rounded instead of truncated. Results should be
0000 59 : correct within 2 LSB's.
0000 60
0000 61 : Edit History for Version 1 of MTH$GSQRT
0000 62
0000 63 : 1-001 - Adapted from MTH$DSQRT version 1-010. SBL 08-Jan-79
0000 64 : 1-002 - Correct some typos in comments. JBS 30-JUL-1979
0000 65 : 1-003 - Use ASHQ, not ASHL, to generate reserved operand. JAW 12-Oct-1979
0000 66 : 1-004 - Change W^ to G^ on call to MTH$$SIGNAL RNH 09-Sept-1981
```

```
0000 68      .SBITL DECLARATIONS      ; Declarative Part of Module
0000 69
0000 70 :
0000 71 : INCLUDE FILES:      OTSPARAMS.MAR
0000 72 :
0000 73 : EXTERNAL SYMBOLS:
0000 74 :
0000 75      .DSABL  GBL      ; Declare all externals
0000 76      .EXTRN  MTH$$SIGNAL ; SIGNAL SEVERE error
0000 77      .EXTRN  MTH$K_SQUROONEG ; Error code
0000 78
0000 79 :
0000 80 : EQUATED SYMBOLS:
0000 81 :
0000 82 :
0000403C 0000 83      ACMASK = ^M<IV, R2, R3, R4, R5> ; register save mask and IV enable
0000 84 :
0000 85 : MACROS:      none
0000 86 :
0000 87 : PSECT DECLARATIONS:
0000 88 :
00000000 0000 89      .PSECT  _MTH$CODE      PIC,SHR,LONG,EXE,NOWRT
0000 90      ; program section for math routines
0000 91 :
0000 92 : OWN STORAGE:
0000 93 :
0000 94 :
0000 95 : Constants A and B chosen for k = odd
0000 96 :
0000 97 LG_ODD_A_E511:
0000 98      .WORD      ^0057772, ^0101171, ^0120000, ^0000000
00008 0008 99 LG_ODD_B_EM511:
0000 100      .WORD      ^0020003, ^0003611, ^0040000, ^0000000
0010 101 :
0010 102 : Constants A and B chosen for k = even
0010 103 :
0010 104 LG_EVEN_A:
0000 105      .WORD      ^0040002, ^0137303, ^0040000, ^0000000
0018 106 LG_EVEN_B_EM512:
0000 107      .WORD      ^0017772, ^0164544, ^0060000, ^0000000
```

```
0020 109 .SBTTL MTH$GSQRT - Standard G-Floating GSQRT
0020 110
0020 111
0020 112 :++
0020 113 : FUNCTIONAL DESCRIPTION:
0020 114 :
0020 115 : GSQRT - G-floating point function
0020 116 :
0020 117 : GSQRT(X) is computed using the following approximation technique:
0020 118 :
0020 119 : If X < 0, error. If X = 0, return GSQRT(X) = 0.
0020 120 :
0020 121 : Let X = 2**K * F where F is the fractional part.
0020 122 :
0020 123 : If K = even, X = 2**(2P) * F,
0020 124 : GSQRT(X) = 2**P * GSQRT(F), 1/2 <= F < 1
0020 125 :
0020 126 : If K = odd, X = 2**(2P+1) * F = 2**(2P+2) * (F/2),
0020 127 : GSQRT(X) = 2**(P+1) * GSQRT(F/2), 1/4 <= F/2 < 1/2.
0020 128 :
0020 129 : Let F' = A*F + B,
0020 130 : A = 0.453730314(octal),
0020 131 : B = 0.327226214(octal), for K = even.
0020 132 : = A*(F/2) + B,
0020 133 : A = 0.650117146(octal),
0020 134 : B = 0.230170444(octal), for K = odd.
0020 135 :
0020 136 : and
0020 137 : K' = P, for K = even
0020 138 : = P + 1 for K = odd.
0020 139 :
0020 140 : Let Y0 = 2**K' * F' as a straight line approximation within the
0020 141 : given interval using coefficients A and B which minimize the
0020 142 : absolute error at the midpoint and endpoint.
0020 143 :
0020 144 : Starting with Y0, three Newton-Raphson iterations are performed.
0020 145 : Y[n+1] = (1/2) * ( Y[n] + X/Y[n] )
0020 146 :
0020 147 : The relative error is < 10**-17.
0020 148 :
0020 149 : CALLING SEQUENCE:
0020 150 :
0020 151 : gsqrt.wg.v = MTH$GSQRT(x.rg.r)
0020 152 :
0020 153 : INPUT PARAMETERS:
0020 154 :
00000004 0020 155 : LONG = 4 ; define longword multiplier
00000004 0020 156 : x = 1 * LONG ; Contents of x is the argument
0020 157 :
0020 158 : IMPLICIT INPUTS: none
0020 159 :
0020 160 : OUTPUT PARAMETERS:
0020 161 :
0020 162 : VALUE: G-floating square root of the argument
0020 163 :
0020 164 : IMPLICIT OUTPUTS: none
0020 165 :
```

```
0020 166 ; COMPLETION CODES: none
0020 167 ;
0020 168 ; SIDE EFFECTS:
0020 169 ;
0020 170 ; Signals: MTH$_SQURDNEG if X < 0.0 with reserved operand in R0/R1
0020 171 ; (copied to the signal mechanism vector CHF$_MCH_R0/R1 by LIB$$_SIGNAL).
0020 172 ; Associated message is: "SQUARE ROOT OF NEGATIVE VALUE". Result is reserved
0020 173 ; operand -0.0 unless a user supplied (or any) error handler changes CHF$_MCH_R0/R1
0020 174 ;
0020 175 ; NOTE: This procedure disables floating point underflow, enables integer
0020 176 ; overflow, causes no floating overflow or other arithmetic traps, and
0020 177 ; preserves enables across the call.
0020 178 ;
0020 179 ;---
0020 180
0020 181
403C 0020 182 .ENTRY MTH$GSQRT, ACMASK ; standard call-by-reference entry
0022 183 ; disable DV (and FU), enable IV
0022 184 MTH$FLAG_JACKET ; flag that this is a jacket procedure in
0022
6D 00000000'GF 9E 0022 MOVAB G^MTH$$JACKET_HND, (FP)
0029 ; set handler address to jacket
0029 ; handler
0029
0029 185
50 04 BC 50FD 0029 186 MOVG @x(AP), R0 ; case of an error in special routine
01 10 002E 187 BSBB MTH$GSQRT_R5 ; R0/R1 = arg
04 0030 188 RET ; call kernel GSQRT routine
; return - result in R0/R1
```

```
0031 190 .SBTTL MTH$GSQRT_R5 - Special GSQRT routine
0031 191
0031 192 ; Special GSQRT - used by the standard routine, and directly.
0031 193 ;
0031 194 CALLING SEQUENCE:
0031 195 ; save anything in R2:R5
0031 196 ; MOVG R0 ; input in R0/R1
0031 197 ; JSB MTH$GSQRT_R5
0031 198 ; return with result in R0/R1
0031 199 ; Note: This routine is written to avoid causing any integer overflows, floating
0031 200 ; overflows, floating underflows or divide by 0 conditions, whether enabled or
0031 201 ; not.
0031 202 ;
0031 203 REGISTERS USED:
0031 204 ; R0/R1 - Floating argument then result
0031 205 ; R2/R3 - scratch
0031 206 ; R4/R5 - hold X during calc of F', K'.
0031 207
0031 208 MTH$GSQRT_R5::
0031 209 ; JSB routine for GSQRT
0031 209 MOVG R0, R4 ; test sign of X and save it in R4/R5.
0035 210 BLEQ ZERO_NEG ; branch to ZERO_NEG if X <= 0
0037 211 ;
0037 212 ; X > 0
0037 213 ;
0037 214 POS: MOVZWL R0, R2 ; isolate low 16 bits in R2
003A 215 ; (sign, exp, 4 fract bits)
003A 216 BICB2 #^X1F, R2 ; clear fraction, 1 exp bit
003D 217 BICW R2, R0 ; clear exponent bits in R0
0040 218 BBC #4, R0, EVEN ; branch if exponent even
0044 219 MULG2 LG_ODD_A_E511, R0 ; add 511 (half of bias) to
0049 220 ; (exponent-2)
0049 221 ; and start approximation calc
0049 222 ADDG2 LG_ODD_B_EM511, R0 ; R0 = (first approx) * 2**512
004E 223 BRB ADJUST ; go adjust
0050 224
0050 225 EVEN:
0050 226 ADDW2 #^X2000, R0 ; exp is 0 - make it 512
0055 227 MULG2 LG_EVEN_A, R0
005A 228 ADDG2 LG_EVEN_B_EM512, R0 ; R0 = (first approx) * 2**512
005F 229 ADJUST:
005F 230 ROTL #31, R2, R2 ; divide R2 (exp+bias) by 2,
0063 231 ; giving (exp/2+512)
0063 232 ADDW R2, R0 ; insert exp/2 in first approx and
0066 233 ; re-bias it.
0066 234
0066 235 ; first iteration, all done in G floating because of exponent range
0066 236 ;
0066 237 DIVG3 R0, R4, R2 ; R2 = X/Y0
006B 238 ADDG2 R2, R0 ; R0 = Y0 + X/Y0
006F 239 SUBW #^X10, R0 ; R0 = Y1 = (1/2)(Y0 + X/Y0)
0072 240 ; no overflow possible
0072 241
0072 242 ; second iteration
0072 243 ;
0072 244 DIVG3 R0, R4, R2 ; R2/R3 = X/Y1
0077 245 ADDG2 R2, R0 ; R0/R1 = Y1 + higher_part(X/Y1)
007B 246 SUBW #^X10, R0 ; R0/R1 = Y2 = (1/2)(Y1+X/Y1)
```

```
007E 247
007E 248 ; third iteration
007E 249 ;
52 54 50 47FD 007E 250 DIVG3 R0, R4, R2 ; R2/R3 = X/Y2
50 52 40FD 0083 251 ADDG2 R2, R0 ; R0 = Y2+X/Y2
50 10 A2 0087 252 SUBW #*X10, R0 ; R0/R1 = GSQRT(X) =
008A 253 ; (1/2) (Y2+X/Y2)
05 008A 254 SQRTX: RSB ; return, R0/R1 = result
008B 255
008B 256 ; X =< 0
008B 257 ;
008B 258 ZERO_NEG:
008B 259 BEQL SQRTX ; return with R0 = result = 0
008D 260 PUSHL (SP) ; return PC from JSB routine
7E 00 8F 9A 008F 261 MOVZBL #MTH$K_SQUROONEG, -(SP) ; condition value
50 01 0F 79 0093 262 ASHQ #15, #T, R0 ; R0/R1 = result = reserved operand -0.0
0097 263 ; R0/R1 goes to signal mechanism vector
0097 264 ; (CHF$L_MCH_R0/R1) so error handler
0097 265 ; can modify the result.
00000000'GF 02 FB 0097 266 CALLS #2, G^MTH$$SIGNAL ; signal error and use real user's PC
009E 267 ; independent of CALL vs JSB
05 009E 268 RSB ; return - R0 restored from CHF$L_MCH_R0/R1
009F 269
009F 270 .END
```

MTH\$GSQRT
Symbol table

I 7
; G Floating Point Square Root routine

16-SEP-1984 01:31:52
6-SEP-1984 11:24:20

VAX/VMS Macro V04-00
[MTHRTL.SRC]MTHGSQRT.MAR;1

Page 8
(5)

ACMASK = 0000403C
ADJUST 0000005F R 01
EVEN 00000050 R 01
LG_EVEN_A 00000010 R 01
LG_EVEN_B_EM512 00000018 R 01
LG_ODD_A_E511 00000000 R 01
LG_ODD_B_EM511 00000008 R 01
LONG = 00000004
MTH\$\$JACKET_HND ***** X 01
MTH\$\$SIGNAL ***** X 00
MTH\$GSQRT 00000020 RG 01
MTH\$GSQRT_R5 00000031 RG 01
MTH\$K_SQUROONEG ***** X 00
POS 00000037 R 01
SQR TX 0000008A R 01
X = 00000004
ZERO_NEG 0000008B R 01

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

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR
MTH\$CODE	0000009F (159.)	01 (1.)	PIC USR

CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
CON REL LCL SHR EXE RD NOWRT NOVEC LONG

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.08	00:00:00.94
Command processing	109	00:00:00.64	00:00:03.45
Pass 1	82	00:00:00.81	00:00:03.49
Symbol table sort	0	00:00:00.00	00:00:00.00
Pass 2	61	00:00:00.67	00:00:03.13
Symbol table output	2	00:00:00.03	00:00:00.03
Psect synopsis output	3	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	290	00:00:02.26	00:00:11.12

The working set limit was 900 pages.
4084 bytes (8 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 17 non-local and 0 local symbols.
330 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
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	0

MTH\$GSQRT ; G Floating Point Square Root routine J 7 16-SEP-1984 01:31:52 VAX/VMS Macro V04-00 Page 9
VAX-11 Macro Run Statistics 6-SEP-1984 11:24:20 [MTHRTL.SRC]MTHGSQRT.MAR;1 (5)

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

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

MTH
VAX
88
The
MAC

0261 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY