



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

MTH\$SQRT  
Table of contents

; Floating Point Square Root routine L 15

16-SEP-1984 01:51:08 VAX/VMS Macro V04-00

Page 0

(2) 55  
(3) 83  
(4) 123  
(5) 205

HISTORY : Detailed Current Edit History  
DECLARATIONS : Declarative Part of Module  
MTH\$SQRT - Standard Single Precision Floating SQRT  
MTH\$SQRT\_R3 - JSB SQRT routine

MTH\$SQRT  
1-015

M 15  
; Floating Point Square Root routine

16-SEP-1984 01:51:08 VAX/VMS Macro V04-00  
6-SEP-1984 11:27:12 [MTHRTL.SRC]MTH\$SQRT.MAR;1

Page 1  
(1)

```
0000 1 .TITLE MTH$SQRT ; Floating Point Square Root routine
0000 2 ; (SQRT)
0000 3 .IDENT /1-015/ ; File: MTH$SQRT.MAR EDIT RNH1015
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$SQRT is a function which returns the floating point square root
0000 34 of its single precision floating point argument. The call is standard
0000 35 call-by-reference.
0000 36 MTH$SQRT_R3 is a special routine which is the same as MTH$SQRT except
0000 37 a faster non-standard JSB call is used with the argument in R0 and no
0000 38 registers are saved.
0000 39
0000 40 --
0000 41
0000 42 VERSION: 01
0000 43
0000 44 HISTORY:
0000 45 AUTHOR:
0000 46 Peter Yuo, 15-Oct-76: Version 01
0000 47
0000 48 MODIFIED BY:
0000 49
0000 50 01-1 Peter Yuo, 22-May-77
0000 51 01-2 Peter Yuo, 31-May-77
0000 52
0000 53
```

MTH\$SQRT  
1-015

N 15  
; Floating Point Square Root routine 16-SEP-1984 01:51:08 VAX/VMS Macro V04-00  
HISTORY ; Detailed Current Edit History 6-SEP-1984 11:27:12 [MTHRTL.SRC]MTH\$SQRT.MAR;1

Page 2  
(2)

```
0000 55      .SBTTL HISTORY ; Detailed Current Edit History
0000 56
0000 57
0000 58 : ALGORITHMIC DIFFERENCES FROM FP-11/C ROUTINE: none
0000 59
0000 60 : Edit History for Version 01 of MTH$SQRT
0000 61
0000 62 : 01-1 Code saving after code review
0000 63 : 01-2 ROTL shift in garbage into highest bit. Use ASHL instead.
0000 64 : ADDL instruction after ADJUST has been changed into ADDW to prevent
0000 65 : overflow if R1<31:16> = FFFF and R0<31:16> = FFFF
0000 66 : 01-3 Finish error handling 10-June-1977
0000 67 : 01-5 MTH$ERROR changed to MTH$SIGNAL.
0000 68 : MTH$... changed to MTH$.....
0000 69 : Changed error handling mechanism. Put error result in R0 before
0000 70 : calling MTH$SIGNAL in order to allow user modify error result.
0000 71 : 01-6 Return -0.0 on negative arg. TNH 20-Dec-77
0000 72 : 01-7 Edit in Rich Lary's code bums. JSB routine is now R3. JMT 19-Jan-78
0000 73 : 01-9 Move .ENTRY symbol to module header. TNH 14-Aug-78
0000 74 : 1-010 - Put version number in standard format: three digit edit
0000 75 : numbers. Also, update the copyright notice. JBS 16-NOV-78
0000 76 : 1-011 - Change MTH_SQUROONEG to MTH$K_SQUROONEG. JBS 07-DEC-78
0000 77 : 1-012 - Add " to the PSECT directive. JBS 22-DEC-78
0000 78 : 1-013 - Declare externals. SBL 17-May-1979
0000 79 : 1-014 - Move MTH$SQRT R2 to separate module (MTH$SQRT2.MAR) and
0000 80 : replace with MTH$SQRT R3. JAW 26-Sep-1979.
0000 81 : 1-015 - Changed W^ to G^ in call to MTH$SIGNAL RNH 09-Sept-1981
```

```
0000 83 .SBTTL DECLARATIONS ; Declarative Part of Module
0000 84
0000 85 ;
0000 86 ; INCLUDE FILES:
0000 87 ;
0000 88
0000 89 ;
0000 90 ; EXTERNAL SYMBOLS:
0000 91 ;
0000 92 .DSABL GBL
0000 93 .EXTRN MTH$K_SQUROONEG
0000 94 .EXTRN MTH$$SIGNAL
0000 95
0000 96 ;
0000 97 ; EQUATED SYMBOLS:
0000 98 ;
0000 99
0000400C 0000 100 ACMASK = ^M<IV, R2, R3> ; register save mask and IV enable
0000 101 ; MACROS: none
0000 102 ;
0000 103 ; PSECT DECLARATIONS:
0000 104
00000000 0000 105 .PSECT _MTH$CODE PIC,SHR,LONG,EXE,NOWRT
0000 106 ; program section for math routines
0000 107 ;
0000 108 ; OWN STORAGE: none
0000 109 ;
0000 110 ; CONSTANTS:
0000 111 ;
0000 112 ;
0000 113 ; Constants A and B chosen for k = odd
0000 114 ;
13CD5FD4 0000 115 LF_ODD_A_E63 = ^X13CD5FD4
3C4A2018 0000 116 LF_ODD_B_EM63 = ^X3C4A2018
0000 117 ;
0000 118 ; Constants A and B chosen for k = even
0000 119 ;
F61A4015 0000 120 LF_EVEN_A = ^XF61A4015
4B231FD7 0000 121 LF_EVEN_B_EM64 = ^X4B231FD7
```

```
0000 123      .SBTTL MTH$SQRT - Standard Single Precision Floating SQRT
0000 124
0000 125
0000 126 :++
0000 127 : FUNCTIONAL DESCRIPTION:
0000 128 :
0000 129 : SQRT - single precision floating point function
0000 130 :
0000 131 : SQRT(X) is computed using the following approximation technique:
0000 132 :
0000 133 :   If X <= 0 , error. Let X = !X!.
0000 134 :
0000 135 :   Let X = 2**K * F where F is the fractional part.
0000 136 :
0000 137 :   If K = even, X = 2**(2P) * F,
0000 138 :               SQRT(X) = 2**P * SQRT(F), 1/2 <= F < 1
0000 139 :
0000 140 :   If K = odd, X = 2**(2P+1) * F = 2**(2P+2) * (F/2),
0000 141 :               SQRT(X) = 2**(P+1) * SQRT(F/2), 1/4 <= F/2 < 1/2.
0000 142 :
0000 143 :   Let F' = A*F + B,
0000 144 :               A = 0.453730314(octal),
0000 145 :               B = 0.327226214(octal), for K = even.
0000 146 :               = A*(F/2) + B,
0000 147 :               A = 0.650117146(octal),
0000 148 :               B = 0.230170444(octal), for K = odd.
0000 149 :
0000 150 :   and
0000 151 :       K' = P,      for K = even
0000 152 :       K' = P + 1  for K = odd.
0000 153 :
0000 154 :   Let Y0 = 2**K' * F' as a straight line approximation within the
0000 155 :   given interval using coefficients A and B which minimize the
0000 156 :   absolute error at the midpoint and endpoint.
0000 157 :
0000 158 :   Starting with Y0, two Newton-Raphson iterations are performed.
0000 159 :   Y[n+1] = (1/2) * ( Y[n] + X/Y[n] )
0000 160 :
0000 161 :   The relative error is < 10**-8.
0000 162 :
0000 163 : CALLING SEQUENCE:
0000 164 :
0000 165 :   sqrt.wf.v = MTH$SQRT(x.rf.r)
0000 166 :
0000 167 : INPUT PARAMETERS:
0000 168 :
0000 169 :   LONG = 4 ; define longword multiplier
0000 170 :   x = 1 * LONG ; Contents of x is the argument
0000 171 :
0000 172 : IMPLICIT INPUTS: none
0000 173 :
0000 174 : OUTPUT PARAMETERS:
0000 175 :
0000 176 :   VALUE: floating square root of the argument
0000 177 :
0000 178 : IMPLICIT OUTPUTS: none
0000 179 :
```

00000004  
00000004

```
0000 180 : COMPLETION CODES: none
0000 181 :
0000 182 : SIDE EFFECTS:
0000 183 :
0000 184 : Signals: MTH$_SQURDNEG if X < 0.0 with reserved operand in R0 (copied to
0000 185 : the signal mechanism vector CHF$_MCH_R0/R1 by LIB$SIGNAL).
0000 186 : Associated message is: "SQUARE ROOT OF NEGATIVE VALUE". Result is reserved
0000 187 : operand -0.0 unless a user supplied (or any) error handler changes CHF$_MCH_R0/R1
0000 188 :
0000 189 : NOTE: This procedure disables floating point underflow, enables integer
0000 190 : overflow, causes no floating overflow or other arithmetic traps, and
0000 191 : preserves enables across the call.
0000 192 :
0000 193 : ---
0000 194 :
0000 195 :
400C 0000 196 .ENTRY MTH$SQRT, ACMASK ; standard call-by-reference entry
0002 197 ; disable DV (and FU), enable IV
0002 198 MTH$FLAG_JACKET ; flag that this is a jacket procedure in
0002 199
6D 00000000'GF 9E 0002 MOVAB G^MTH$$JACKET_HND, (FP)
0009 ; set handler address to jacket
0009 ; handler
0009 199
50 04 BC 50 0009 200 ; case of an error in special routine
01 10 000D 201 ; R0 = arg
04 000F 202 ; call specail SQRT routine
0010 203 RET ; return - result in R0
```

```
0010 205 .SBTTL MTH$SQRT_R3 - JSB SQRT routine
0010 206
0010 207 ; JSB SQRT - used by the standard, and directly.
0010 208
0010 209 CALLING SEQUENCE:
0010 210 ; save anything in R0:R2
0010 211 MOVF R0 ; input in R0
0010 212 JSB MTH$SQRT_R3
0010 213 ; return with result in R0
0010 214
0010 215 Note: This routine is written to avoid any integer overflows, floating overflows,
0010 216 floating underflows or divide by 0 conditions, whether enabled or not.
0010 217
0010 218 REGISTERS USED:
0010 219 R0 - Floating argument then result
0010 220 R1 - X saved for use during iteration
0010 221 R2 - scratch
0010 222
0010 223 MTH$SQRT_R3:: ; JSB routine for SQRT
51 50 50 0010 224 MOVF R0, R1 ; test sign of X and save it in R1.
53 15 0013 225 BLEQ ZERO_NEG ; branch to ZERO_NEG if X <= 0
0015 226
0015 227 ; X > 0
0015 228
0015 229 POS:
52 50 3C 0015 230 MOVZWL R0, R2 ; isolate low 16 bits (sign,exp,>fract) in R
52 52 94 0018 231 CLRB R2 ; R2 now has sign and left 7 exp bits
50 52 AA 001A 232 BICW R2, R0 ; clear sign and left 7 exp bits
50 50 95 001D 233 TSTB R0 ; check low bit of exp
10 18 001F 234 BGEQ EVEN ; and branch if 1
50 13CD5FD4 8F 44 0021 235 MULF #LF_ODD_A_E63, R0 ; add 64 (half of bias) to (exponent-2)
0028 236 ; and start approximation calc
50 3C4A2018 8F 40 0028 237 ADDF #LF_ODD_B_EM63, R0 ; R0 = (first approx) * 2**-64
13 11 002F 238 BRB ADJUST ; go adjust
0031 239
0031 240 EVEN:
50 50 2000 8F A0 0031 241 ADDW #X2000, R0 ; exp is 0 - make it 64 (2**-64) for legalit
50 F61A4015 8F 44 0036 242 MULF #LF_EVEN_A, R0
50 4B231FD7 8F 40 003D 243 ADDF #LF_EVEN_B_EM64, R0 ; R0 = (first approx) * 2**-64
0044 244 ADJUST:
52 52 1F 9C 0044 245 ROTL #31, R2, R2 ; divide R2 (exp+bias) by 2,
0048 246 ; giving (exp/2+64)
50 52 A0 0048 247 ADDW R2, R0 ; insert exp/2 in first approx and
004B 248 ; re-bias it.
004B 249
004B 250 ; first iteration - single precision is sufficient
004B 251
52 51 50 47 004B 252 DIVF3 R0, R1, R2 ; R2 = X/Y0
50 50 52 40 004F 253 ADDF R2, R0 ; R0 = Y0 + X/Y0
50 0080 8F A2 0052 254 SUBW #X80, R0 ; R0 = Y1 = (1/2)(Y0 + X/Y0)
0057 255 ; no overflow possible
0057 256
0057 257 ; second iteration, do in double precision to get truncated( rather than
0057 258 ; rounded) result.
0057 259
0057 260 CLRL R2 ; lower part (X) = 0
0057 261 DIVD R0, R1 ; divide Y1 into X with low-order
```

```

; 32 bits of Y1 garbage. This doesn't
; effect accuracy, since Y1 innacurate
; anyway.
; convert and copy X into R2/R3
; clear low part of Y1
; divide Y1 into X
; R0 = Y1 + higher_part(X/Y1)
; R0 = SQRT(X) = (T/2) (Y1 + X/Y1)
; return, R0 = result

0057 262
0057 263
0057 264
52 51 56 0057 265 CVTFD R1, R2
52 51 D4 005A 266 CLRL R1
52 50 66 005C 267 DIVD2 R0, R2
50 50 52 40 005F 268 ADDF R2, R0
50 0080 8F A2 0062 269 SUBW #^X80, R0
05 0067 270 SQRTX: RSB
0068 271
0068 272 ; X =< 0
0068 273
0068 274 ZERO_NEG:
0068 275 BEQL SQRTX
7E 00 8F DD 006A 276 PUSHL (SP)
50 01 0F 9A 006C 277 MOVZBL #MTH$K_SQUROONEG, -(SP)
0070 278 ASHL #15, #T, R0
0074 279
0074 280
0074 281
00000000'GF 02 FB 0074 282 CALLS #2, G^MTH$$SIGNAL
007B 283
05 007B 284 RSB
007C 285
007C 286 .END

; return with R0 = result = 0
; return PC from JSB routine
; condition value
; R0 = result = reserved operand -0.0
; R0 goes to signal mechanism vector
; (CHF$MCH_R0/R1) so error handler
; can modify the result.
; signal error and use real user's PC
; independent of CALL vs JSB
; return - R0 restored from CHF$MCH_R0/R1
```

```

ACMASK      = 0000400C
ADJUST      = 00000044 R    01
EVEN        = 00000031 R    01
LF_EVEN_A   = F61A4015
LF_EVEN_B_EM64 = 4B231FD7
LF_ODD_A_E63 = 13CD5FD4
LF_ODD_B_EM63 = 3C4A2018
LONG        = 00000004
MTH$JACKET_HND ***** X    01
MTH$SIGNAL ***** X    00
MTH$K_SQUROONEG ***** X    00
MTH$SQRT    = 00000000 RG   01
MTH$SQRT_R3 = 00000010 RG   01
POS         = 00000015 R    01
SQRTX       = 00000067 R    01
X           = 00000004
ZERO_NEG    = 00000068 R    01

```

```

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

```

| PSECT name | Allocation       | PSECT No. | Attributes |     |     |     |     |       |       |                  |
|------------|------------------|-----------|------------|-----|-----|-----|-----|-------|-------|------------------|
| ABS        | 00000000 ( 0.)   | 00 ( 0.)  | NOPIC      | USR | CON | ABS | LCL | NOSHR | NOEXE | NORD             |
| _MTH\$CODE | 0000007C ( 124.) | 01 ( 1.)  | PIC        | USR | CON | REL | LCL | SHR   | EXE   | RD               |
|            |                  |           |            |     |     |     |     |       |       | NOWRT NOVEC BYTE |
|            |                  |           |            |     |     |     |     |       |       | NOWRT NOVEC LONG |

```

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

```

| Phase                  | Page faults | CPU Time    | Elapsed Time |
|------------------------|-------------|-------------|--------------|
| Initialization         | 29          | 00:00:00.10 | 00:00:00.81  |
| Command processing     | 109         | 00:00:00.72 | 00:00:04.91  |
| Pass 1                 | 82          | 00:00:00.83 | 00:00:03.78  |
| Symbol table sort      | 0           | 00:00:00.00 | 00:00:00.00  |
| Pass 2                 | 64          | 00:00:00.72 | 00:00:03.39  |
| Symbol table output    | 2           | 00:00:00.03 | 00:00:00.41  |
| 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   | 291         | 00:00:02.42 | 00:00:13.43  |

The working set limit was 900 pages.  
4082 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.  
346 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\$SQRT ; Floating Point Square Root routine<sup>H 16</sup>  
VAX-11 Macro Run Statistics

16-SEP-1984 01:51:08 VAX/VMS Macro V04-00  
6-SEP-1984 11:27:12 [MTHRTL.SRC]MTH\$SQRT.MAR;1

Page 9  
(5)

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

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

0263

AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY