```
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         HHHHHHHHHHHHHHHH  RRRRRRRRRRR           TTT         LLL
MMM         MMM         TTT         HHHHHHHHHHHHHHHH  RRRRRRRRRRRR          TTT         LLL
MMM         MMM         TTT         HHHHHHHHHHHHHHHH  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         LLLLLLLLLLLLLLLL
MMM         MMM         TTT         HHH          HHH  RRR         RRR       TTT         LLLLLLLLLLLLLLLL
MMM         MMM         TTT         HHH          HHH  RRR         RRR       TTT         LLLLLLLLLLLLLLLL
```

```
MM      MM  TTTTTTTTTT  HH      HH  DDDDDDDD     SSSSSSSS    QQQQQQ    RRRRRRRR    TTTTTTTTTT
MM      MM  TTTTTTTTTT  HH      HH  DDDDDDDD     SSSSSSSS    QQQQQQ    RRRRRRRR    TTTTTTTTTT
MMMM  MMMM      TT      HH      HH  DD      DD  SS          QQ      QQ  RR      RR      TT
MMMM  MMMM      TT      HH      HH  DD      DD  SS          QQ      QQ  RR      RR      TT
MM  MM  MM      TT      HH      HH  DD      DD  SS          QQ      QQ  RR      RR      TT
MM  MM  MM      TT      HH      HH  DD      DD  SS          QQ      QQ  RR      RR      TT
MM      MM      TT      HHHHHHHHHH  DD      DD    SSSSSS    QQ      QQ  RRRRRRRR        TT
MM      MM      TT      HHHHHHHHHH  DD      DD    SSSSSS    QQ      QQ  RRRRRRRR        TT
MM      MM      TT      HH      HH  DD      DD          SS  QQ  QQ  QQ  RR  RR          TT
MM      MM      TT      HH      HH  DD      DD          SS  QQ  QQ  QQ  RR  RR          TT
MM      MM      TT      HH      HH  DD      DD          SS  QQ    QQ    RR    RR        TT
MM      MM      TT      HH      HH  DD      DD          SS  QQ    QQ    RR    RR        TT
MM      MM      TT      HH      HH  DDDDDDDD    SSSSSSSS    QQQQ  QQ  RR      RR      TT
MM      MM      TT      HH      HH  DDDDDDDD    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
```

Table of contents

```
0000     1          .TITLE  MTH$DSQRT       ; Double Floating Point Square Root routine
0000     2                                  ; (DSQRT)
0000     3          .IDENT /1-015/          ; File: MTHDSQRT.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$DSQRT 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$DSQRT_R5  is  a  special routine  which  is the same as MTH$DSQRT
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: 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 ;
```

```
0000     55          .SBTTL  HISTORY ; Detailed Current Edit History
0000     56
0000     57
0000     58 ; ALGORITHMIC DIFFERENCES FROM FP-11/C ROUTINE:
0000     59 ;       1. Last DIVD is rounded instead of truncated. Results should be
0000     60 ;          correct within 2 LSB's.
0000     61 ;
0000     62 ; Edit History for Version 01 of MTH$DSQRT
0000     63 ;
0000     64 ; 01-1  Code saving after code review March 1977
0000     65 ; 01-2  ROTL shift in garbage into highest bit. Use ASHL instead.
0000     66 ;       ADDL instruction after ADJUST has been changed into ADDW to prevent
0000     67 ;       overflow if R1<31:16> = FFFF and R0<31:16> = FFFF
0000     68
0000     69 ; 01-3  Finish error handling 10-June-1977
0000     70 ; 0-4   MTH$$ERROR changed to MTH$$SIGNAL.
0000     71 ;       MTH$_... changed to MTH__....
0000     72 ;       Changed error handling mechanism. Put error result in R0:R1 before
0000     73 ;       calling MTH$$SIGNAL in order to allow user modify error result.
0000     74 ; 01-5  - Return -0.0 on negative arg.  TNH 20-Dec-77
0000     75 ; 01-6  - Remove undefined $15 global.  TNH 20-Dec-77
0000     76 ; 01-7  - Add Rich Lary's code bums.  JMT 26-Jan-78
0000     77 ; 01-9  - Move .ENTRY symbol definition to module header.  TNH 14-Aug-78
0000     78 ; 1-010 - Update version number and copyright notice.  JBS 16-NOV-78
0000     79 ; 1-011 - Change symbol MTH__SQUROONEG to MTH$K_SQUROONEG.  JBS 07-DEC-78
0000     80 ; 1-012 - Add "_" to the PSECT directive.  JBS 22-DEC-78
0000     81 ; 1-013 - Declare externals.  SBL 17-May-1979
0000     82 ; 1-014 - Use ASHQ, not ASHL, to generate reserved operand.  JAW 12-Oct-1979
0000     83 ; 1-015 - Changed W^ to G^ in call to MTH$$SIGNAL RNH 09-Sept-1981
```

```
                     0000     85          .SBTTL  DECLARATIONS    ; Declarative Part of Module
                     0000     86
                     0000     87 ;
                     0000     88 ; INCLUDE FILES:        MTHJACKET.MAR
                     0000     89 ;
                     0000     90
                     0000     91 ;
                     0000     92 ; EXTERNAL SYMBOLS:
                     0000     93 ;
                     0000     94          .DSABL  GBL
                     0000     95          .EXTRN  MTH$K_SQUROONEG
                     0000     96          .EXTRN  MTH$$SIGNAL             ; SIGNAL SEVERE error
                     0000     97
                     0000     98 ;
                     0000     99 ; EQUATED SYMBOLS:
                     0000    100 ;
                     0000    101
0000403C             0000    102          ACMASK = ^M<IV, R2, R3, R4, R5> ; register save mask and IV enable
                     0000    103 ;
                     0000    104 ; MACROS:       none
                     0000    105 ;
                     0000    106 ; PSECT DECLARATIONS:
                     0000    107
                00000000     108          .PSECT  _MTH$CODE       PIC,SHR,LONG,EXE,NOWRT
                     0000    109                                          ; program section for math routines
                     0000    110 ;
                     0000    111 ; OWN STORAGE:  none
                     0000    112 ;
                     0000    113 ; CONSTANTS:
                     0000    114
                     0000    115 ;
                     0000    116 ; Constants A and B chosen for k = odd
                     0000    117 ;
13CD5FD4             0000    118          LF_ODD_A_E63    =       ^X13CD5FD4
3C4A2018             0000    119          LF_ODD_B_EM63   =       ^X3C4A2018
                     0000    120 ;
                     0000    121 ; Constants A and B chosen for k = even
                     0000    122 ;
F61A4015             0000    123          LF_EVEN_A       =       ^XF61A4015
4B231FD7             0000    124          LF_EVEN_B_EM64  =       ^X4B231FD7
```

```
                        .SBTTL  MTH$DSQRT  - Standard Double Precision Floating DSQRT


;++
; FUNCTIONAL DESCRIPTION:
;
; DSQRT  - double precision floating point function
;
; DSQRT(X) is computed using the following approximation technique:
;
;       If X <= 0 , error.  Let X = |X|.
;
;       Let X = 2**K * F where F is the fractional part.
;
;       If K = even, X = 2**(2P) * F,
;               DSQRT(X) = 2**P * DSQRT(F), 1/2 =< F < 1
;
;       If K = odd, X = 2**(2P+1) * F = 2**(2P+2) * (F/2),
;               DSQRT(X) = 2**(P+1) * DSQRT(F/2), 1/4 =< F/2 < 1/2.
;
;       Let F' = A*F + B,
;                       A = 0.453730314(octal),
;                       B = 0.327226214(octal), for K = even.
;              = A*(F/2) + B,
;                       A = 0.650117146(octal),
;                       B = 0.230170444(octal), for K = odd.
;       and
;           K' = P,       for K = even
;              = P'+ 1    for K = odd.
;
;       Let Y0 = 2**K' * F' as a staight line approximation wthin the
;       given interval using coefficients A and B which minimize the
;       absolute error at the midpoint and endpoint.
;
;       Starting with Y0, three Newton-Raphson iterations are performed.
;
;       Y[n+1] = (1/2) * ( Y[n] + X/Y[n])
;
;       The relative error is < 10**-17.
;
; CALLING SEQUENCE:
;
;       dsqrt.wd.v = MTH$DSQRT(x.rd.r)
;
; INPUT PARAMETERS:
;
00000004        LONG = 4                        ; define longword multiplier
00000004        x = 1 * LONG                    ; Contents of x is the argument
;
; IMPLICIT INPUTS:      none
;
; OUTPUT PARAMETERS:
;
;       VALUE:  double precision square root of the argument
;
; IMPLICIT OUTPUTS:     none
;
```

```
                                0000   183 ; COMPLETION CODES:     none
                                0000   184 ;
                                0000   185 ; SIDE EFFECTS:
                                0000   186 ;
                                0000   187 ; Signals: MTH$_SQUROONEG if X < 0.0 with reserved operand in R0/R1
                                0000   188 ; (copied to the signal mechanism vector CHF$L_MCH_R0/R1 by LIB$SIGNAL).
                                0000   189 ; Associated message is: "SQUARE ROOT OF NEGATIVE VALUE". Result is reserved
                                0000   190 ; operand -0.0 unless a user supplied (or any) error handler changes CHF$L_MCH_R0/R1
                                0000   191 ;
                                0000   192 ; NOTE: This procedure disables floating point underflow, enables integer
                                0000   193 ; overflow, causes no floating overflow or other arithmetic traps, and
                                0000   194 ; preserves enables across the call.
                                0000   195 ;
                                0000   196 ;---
                                0000   197
                                0000   198
                           403C 0000   199         .ENTRY  MTH$DSQRT, ACMASK       ; standard call-by-reference entry
                                0002   200                                         ; disable DV (and FU), enable IV
                                0002   201         MTH$FLAG_JACKET                 ; flag that this is a jacket procedure in
                                0002                                            ;
    6D   00000000'GF    9E      0002                MOVAB   G^MTH$$JACKET_HND, (FP)
                                0009                                            ; set handler address to jacket
                                0009                                            ; handler
                                0009
                                0009   202                                      ; case of an error in special routine
     50   04 BC    70          0009   203         MOVD    @x(AP), R0              ; R0/R1 = arg
             01    10          000D   204         BSBB    MTH$DSQRT_R5            ; call kernel DSQRT rountine
                   04          000F   205         RET                             ; return - result in R0/R1
```

```
                                0010   207          .SBTTL  MTH$DSQRT_R5  - Special DSQRT routine
                                0010   208
                                0010   209 ; Special DSQRT - used by the standard routine, and directly.
                                0010   210 ;
                                0010   211 ; CALLING SEQUENCE:
                                0010   212 ;       save anything in R2:R5
                                0010   213 ;       MOVD    ...,R0                  ; input in R0/R1
                                0010   214 ;       JSB     MTH$DSQRT_R5
                                0010   215 ;       return with result in R0/R1
                                0010   216 ; Note: This routine is written to avoid causing any integer overflows, floating
                                0010   217 ; overflows, floating underflows or divide by 0 conditions, whether enabled or
                                0010   218 ; not.
                                0010   219 ;
                                0010   220 ; REGISTERS USED:
                                0010   221 ;       R0/R1 - floating argument then result
                                0010   222 ;       R2/R3 - scratch
                                0010   223 ;       R4/R5 - hold X during calc of F', K'.
                                0010   224
                                0010   225 MTH$DSQRT_R5::                          ; JSB routine for DSQRT
          54    50    70      0010   226          MOVD    R0, R4                  ; test sign of X and save it in R4/R5.
                5D    15      0013   227          BLEQ    ZERO_NEG                ; branch to ZERO_NEG if X =< 0
                              0015   228 ;
                              0015   229 ; X > 0
                              0015   230 ;
          51    50    3C      0015   231 POS:     MOVZWL  R0, R1                  ; isolate low 16 bits (sign,exp,>fract) in R
                51    94      0018   232          CLRB    R1                      ; R1 now has sign and left 7 exp bits
          50    51    AA      001A   233          BICW    R1, R0                  ; clear sign and left 7 exp bits
                50    95      001D   234          TSTB    R0                      ; check low bit of exp
                10    18      001F   235          BGEQ    EVEN                    ; and branch if 1
    50  3CD5FD4 8F    44      0021   236          MULF    #LF_ODD_A_E63, R0       ; add 64 (half of bias) to (exponent-2)
                              0028   237                                          ; and start approximation calc
    50  3C4A2018 8F    40      0028   238          ADDF    #LF_ODD_B_EM63, R0      ; R0 = (first approx) * 2**-64
                13    11      002F   239          BRB     ADJUST                  ; go adjust
                              0031   240
                              0031   241 EVEN:
       50   2000 8F    A0      0031   242          ADDW    #^X2000, R0             ; exp is 0 - make it 64 (2**-64) for legalit
    50  F61A4015 8F    44      0036   243          MULF    #LF_EVEN_A, R0
    50   4B231FD7 8F    40      003D   244          ADDF    #LF_EVEN_B_EM64, R0     ; R0 = (first approx) * 2**-64
                              0044   245 ADJUST:
       51   51    1F    9C      0044   246          ROTL    #31, R1, R1             ; divide R1 (exp+bias) by 2,
                              0048   247                                          ; giving (exp/2+64)
            50    51    A0      0048   248          ADDW    R1, R0                  ; insert exp/2 in first approx and
                              0048   249                                          ; re-bias it.
                              0048   250
                              0048   251 ; first iteration - single precision is sufficient
                              0048   252 ;
       51   54    50    47      0048   253          DIVF3   R0, R4, R1              ; R1 = X/Y0
            50    51    40      004F   254          ADDF    R1, R0                  ; R0 = Y0 + X/Y0
       50   0080 8F    A2      0052   255          SUBW    #^X80, R0               ; R0 = Y1 = (1/2)(Y0 + X/Y0)
                              0057   256                                          ; no overflow possible
                              0057   257
                              0057   258 ; second iteration, do in double precision to get truncated( rather than
                              0057   259 ; rounded) result.
                              0057   260 ;
                51    D4      0057   261          CLRL    R1                      ; lower part (X) = 0
       52   54    50    67      0059   262          DIVD3   R0, R4, R2              ; R2/R3 = X/Y1
            50    52    60      005D   263          ADDD    R2, R0                  ; R0/R1 = Y1 + higher_part(X/Y1)
```

```
                50   0080 8F    A2  0063   264          SUBW    #^X80, R0               ; R0/R1 = Y2 = (1/2) (Y1+X/Y1)
                                    0065   265
                                    0065   266 ; third iteration, do in double precision.
                                    0065   267 ;
                52   54    50     67  0065   268          DIVD3   R0, R4, R2              ; R2/R3 = X/Y2
                     50    52     60  0069   269          ADDD    R2, R0                  ; R0 = Y2+X/Y2
                50   0080 8F    A2  006C   270          SUBW    #^X80, R0               ; R0/R1 = DSQRT(X) =
                                    0071   271                                          ; (1/2) (Y2+X/Y2)
                                05  0071   272 SQRTX:   RSB                             ; return, R0/R1 = result
                                    0072   273
                                    0072   274 ; X =< 0
                                    0072   275 ;
                                    0072   276 ZERO_NEG:
                          FD    13  0072   277          BEQL    SQRTX                   ; return with R0 = result = 0
                          6E    DD  0074   278          PUSHL   (SP)                    ; return PC from JSB routine
                   7E     00'8F    9A  0076   279          MOVZBL  #MTH$K_SQUROONEG, -(SP) ; condition value
                50   01     0F    79  007A   280          ASHQ    #15, #T, R0             ; R0/R1 = result = reserved operand -0.0
                                    007E   281                                          ; R0/R1 goes to signal mechanism vector
                                    007E   282                                          ; (CHF$L_MCH_R0/R1) so error handler
                                    007E   283                                          ; can modify the result.
0C000000'GF     02  FB  007E   284          CALLS   #2, G^MTH$$SIGNAL               ; signal error and use real user's PC
                                    0085   285                                          ; independent of CALL vs JSB
                                05  0085   286          RSB                             ; return - R0 restored from CHF$L_MCH_R0/R1
                                    0086   287
                                    0086   288          .END
```

```
ACMASK            = 0000403C
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$DSQRT           00000000 RG     01
MTH$DSQRT_R5        00000010 RG     01
MTH$K_SQUROONEG     ********    X   00
POS                 00000015 R      01
SQRTX               00000071 R      01
X                 = 00000004
ZERO_NEG            00000072 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	00000086 (134.)	01 (1.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.08	00:00:01.27
Command processing	106	00:00:00.63	00:00:05.92
Pass 1	82	00:00:00.88	00:00:04.28
Symbol table sort	0	00:00:00.01	00:00:00.01
Pass 2	64	00:00:00.68	00:00:03.11
Symbol table output	3	00:00:00.04	00:00:00.16
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	288	00:00:02.33	00:00:14.77

```
The working set limit was 900 pages.
4182 byies (9 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.
348 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
_255DUA28:[SYSLIB]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$:MTHDSQRT/OBJ=OBJ$:MTHDSQRT MSRC$:MTHJACKET/UPDATE=(ENH$:MTHJACKET)+MSRC
```

MTHDCOSH LIS

MTHDMIN LIS

MTHDLOG LIS

MTHDSINCO LIS

MTHDATANH LIS

MTHDNINT LIS

MTHDSQRT LIS

MTHDCONJG LIS

MTHDINT LIS

MTHDMAX1 LIS

MTHDSIGN LIS

MTHDIM LIS

MTHDMOD LIS

MTHDSINH LIS

MTHDEXP LIS

MTHDFLOOR LIS

MTHDPROD LIS

MTHGCONJG
LIS

MTHGINT
LIS

MTHGMOD
LIS

MTHEXP
LIS

MTHFLOOR
LIS

MTHDTAN
LIS

MTHDTANH
LIS

MTHGEXP
LIS

MTHGFLOOR
LIS

MTHGMIN1
LIS

MTHGCOSH
LIS

MTHGLOG
LIS

MTHGACOS
LIS

MTHGASIN
LIS

MTHGNINT
LIS

MTHGATAN
LIS

MTHGATANH
LIS

MTHGMAX1
LIS