```
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
```

FILEID**UVXSQRT

```
UU      UU  VV      VV  XX      XX    SSSSSSSS    QQQQQQ    RRRRRRRR    TTTTTTTTTT
UU      UU  VV      VV  XX      XX    SSSSSSSS    QQQQQQ    RRRRRRRR    TTTTTTTTTT
UU      UU  VV      VV  XX      XX  SS          QQ      QQ  RR      RR      TT
UU      UU  VV      VV  XX      XX  SS          QQ      QQ  RR      RR      TT
UU      UU  VV      VV    XX  XX    SS          QQ      QQ  RR      RR      TT
UU      UU  VV      VV    XX  XX    SS          QQ      QQ  RR      RR      TT
UU      UU  VV      VV      XX        SSSSSS    QQ      QQ  RRRRRRRR      TT
UU      UU  VV      VV      XX        SSSSSS    QQ      QQ  RRRRRRRR      TT
UU      UU  VV      VV    XX  XX            SS  QQ  QQ  QQ  RR  RR        TT
UU      UU  VV      VV    XX  XX            SS  QQ  QQ  QQ  RR  RR        TT
UU      UU    VV  VV      XX      XX        SS  QQ    QQ    RR    RR      TT          ....
UU      UU    VV  VV      XX      XX        SS  QQ    QQ    RR    RR      TT          ....
UUUUUUUUUU      VV        XX      XX  SSSSSSSS    QQQQ  QQ  RR      RR    TT          ....
UUUUUUUUUU      VV        XX      XX  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  UVX$SQRT        ; Floating Point Square Root routine
0000     2                                  ; (SQRT)
0000     3          .IDENT /1-016/          ; File: MTHSQRT.MAR     EDIT JCW1016
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 ; 1-016 Jeffrey C. Wiener 22-Feb-83
0000    53 ;
```

```
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$$SIGNAL in order to allow user modify error result.
0000     71 ; 01-6  Return -0.0 on negative arg.  TNH 20-Dec-77
0000     72 ; 01-7  Edit in Rich Lary's code bums.  JSB routine is now _R3.  JMT 19-Jan-78
0000     73 ; 01-9  Move .ENTRY symbol to module header.  TNH 14-Aug-78
0000     74 ; 1-010 - Put version number in standard format: three digit edit
0000     75 ;          numbers.  Also, update the copyright notice.  JBS 16-NOV-78
0000     76 ; 1-011 - Change MTH__SQUROONEG to MTH$K_SQUROONEG.  JBS 07-DEC-78
0000     77 ; 1-012 - Add "_" to the PSECT directive.  JBS 22-DEC-78
0000     78 ; 1-013 - Declare externals.  SBL 17-May-1979
0000     79 ; 1-014 - Move MTH$SQRT_R2 to separate module (MTHSQRTR2.MAR) and
0000     80 ;          replace with MTH$SQRT_R3.  JAW 26-Sep-1979.
0000     81 ; 1-015 - Changed W^ to G^ in call to MTH$$SIGNAL RNH 09-Sept-1981
0000     82 ; 1-016 - Changed all references to D_floating to G_floating. This was
0000     83 ;          done because G_floating, not D_floating, is to be used as
0000     84 ;          back-up to F_floating. JCW 22-FEB-83
```

```
                     0000     86          .SBTTL  DECLARATIONS    ; Declarative Part of Module
                     0000     87
                     0000     88 ;
                     0000     89 ; INCLUDE FILES:
                     0000     90 ;
                     0000     91
                     0000     92 ;
                     0000     93 ; EXTERNAL SYMBOLS:
                     0000     94 ;
                     0000     95          .DSABL  GBL
                     0000     96          .EXTRN  MTH$K_SQUROONEG
                     0000     97          .EXTRN  MTH$$SIGNAL
                     0000     98
                     0000     99 ;
                     0000    100 ; EQUATED SYMBOLS:
                     0000    101 ;
                     0000    102
0000400C             0000    103          ACMASK = ^M<IV, R2, R3>         ; register save mask and IV enable
                     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$SQRT  - Standard Single Precision Floating SQRT

                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ; SQRT  - single precision floating point function
                ;
                ; SQRT(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,
                ;                SQRT(X) = 2**P * SQRT(F), 1/2 =< F < 1
                ;
                ;       If K = odd, X = 2**(2P+1) * F = 2**(2P+2) * (F/2),
                ;                SQRT(X) = 2**(P+1) * SQRT(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, two Newton-Raphson iterations are performed.
                ;
                ;       Y[n+1] = (1/2) * ( Y[n] + X/Y[n])
                ;
                ;       The relative error is < 10**-8.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       sqrt.wf.v = MTH$SQRT(x.rf.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:  floating 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$_SQUROONEG if X < 0.0 with reserved operand in R0 (copied to
                                0000   188 ; the signal mechanism vector CHF$L_MCH_R0/R1 by LIB$SIGNAL).
                                0000   189 ; Associated message is: "SQUARE ROOT OF NEGATIVE VALUE". Result is reserved
                                0000   190 ; operand -0.? unless a user supplied (or any) error handler changes CHF$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
                           400C 0000   199         .ENTRY  MTH$SQRT, ACMASK        ; standard call-by-reference entry
                                0002   200                                         ; disable DV (and FU), enable IV
                                0002   201         MTH$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       50   0009   203         MOVF    @x(AP), R0                  ; R0 = arg
                   01   10   000D   204         BSBB    MTH$SQRT_R3             ; call specail SQRT rountine
                        04   000F   205         RET                             ; return - result in R0
                                0010   206
```

```
                                0010   208          .SBTTL  MTH$SQRT_R3  - JSB SQRT routine
                                0010   209
                                0010   210 ; JSB SQRT - used by the standard, and directly.
                                0010   211 ;
                                0010   212 ; CALLING SEQUENCE:
                                0010   213 ;       save anything in R0:R2
                                0010   214 ;       MOVF    ...,R0                  ; input in R0
                                0010   215 ;       JSB     MTH$SQRT_R3
                                0010   216 ;       return with result in R0
                                0010   217 ;
                                0010   218 ; Note: This routine is written to avoid any integer overflows, floating overflows,
                                0010   219 ; floating underflows or divide by 0 conditions, whether enabled or not.
                                0010   220 ;
                                0010   221 ; REGISTERS USED:
                                0010   222 ;       R0 - Floating argument then result
                                0010   223 ;       R1 - X saved for use during iteration
                                0010   224 ;       R2 - scratch
                                0010   225
                                0010   226 MTH$SQRT_R3::                          ; JSB routine for SQRT
          51    50    50        0010   227          MOVF    R0, R1                  ; test sign of X and save it in R1.
                62    15        0013   228          BLEQ    ZERO_NEG                ; branch to ZERO_NEG if X =< 0
                                0015   229 ;
                                0015   230 ; X > 0
                                0015   231 ;
                                0015   232 POS:
          52    50    3C        0015   233          MOVZWL  R0, R2                  ; isolate low 16 bits (sign,exp,>fract) in R
                52    94        0018   234          CLRB    R2                      ; R2 now has sign and left 7 exp bits
          50    52    AA        001A   235          BICW    R2, R0                  ; clear sign and left 7 exp bits
                50    95        001D   236          TSTB    R0                      ; check low bit of exp
                10    12        001F   237          BGEQ    EVEN                    ; and branch if 1
    50    13CD5FD4    8F    44  0021   238          MULF    #LF_ODD_A_E63, R0       ; add 64 (half of bias) to (exponent-2)
                                0028   239                                          ; and start approximation calc
    50    3C4A2018    8F    40  0028   240          ADDF    #LF_ODD_B_EM63, R0      ; R0 = (first approx) * 2**-64
                13    11        002F   241          BRB     ADJUST                  ; go adjust
                                0031   242
                                0031   243 EVEN:
    50    2000    8F    A0      0031   244          ADDW    #^X2000, R0             ; exp is 0 - make it 64 (2**-64) for legalit
    50    F61A4015    8F    44  0036   245          MULF    #LF_EVEN_A, R0
    50    4B231FD7    8F    40  003D   246          ADDF    #LF_EVEN_B_EM64, R0     ; R0 = (first approx) * 2**-64
                                0044   247 ADJUST:
    52    52    1F    9C      0044   248          ROTL    #31, R2, R2             ; divide R2 (exp+bias) by 2,
                                0048   249                                          ; giving (exp/2+64)
          50    52    A0      0048   250          ADDW    R2, R0                  ; insert exp/2 in first approx and
                                004B   251                                          ; re-bias it.
                                004B   252
                                004B   253 ; first iteration - single precision is sufficient
                                004B   254 ;
    52    51    50    47      004B   255          DIVF3   R0, R1, R2              ; R2 = X/Y0
          50    52    40      004F   256          ADDF    R2, R0                  ; R0 = Y0 + X/Y0
    50    0080    8F    A2    0052   257          SUBW    #^X80, R0               ; R0 = Y1 = (1/2)(Y0 + X/Y0)
                                0057   258                                          ; no overflow possible
                                0057   259
                                0057   260 ; second iteration, do in double precision to get truncated( rather than
                                0057   261 ; rounded) result.
                                0057   262 ;
                                0057   263 ;;;     CLRL    R2                      ; lower part (X) = 0
                                0057   264 ;;;     DIVD    R0, R1                  ; divide Y1 into X with low-order
```

```
                        0057   265                                     ; 32 bits of Y1 garbage.  This doesn't
                        0057   266                                     ; effect accuracy, since Y1 inacurate
                        0057   267                                     ; anyway.
                        0057   268 ;;;     Using DIVG instead we have
                        0057   269 ;;;     DIVG2   R0 ,R1              ; divide Y1 into X with low-order 53-24=
                        0057   270                                     ; 29 bits of Y1 garbage.  This doesn't
                        0057   271                                     ; effect accuracy, since Y1 inacurate
                        0057   272                                     ; anyway.
                        0057   273
        52    51 99FD   0057   274         CVTFG   R1, R2               ; convert and copy X into R2/R3
              51   D4   005B   275         CLRL    R1                   ; clear low part of Y1
        7E    50 99FD   005D   276         CVTFG   R0, -(SP)
        52    8E 46FD   0061   277         DIVG2   (SP)+, R2            ; divide Y1 into X
                        0065   278 ;
                        0065   279 ; Need to CVTGF without rounding. To do this clear the rounding bit.
                        0065   280 ;  The rounding bit is bit 12 of the 3rd word(word 2 starting from 0).
                        0065   281 ;
  53    1FFF 8F   AA    0065   282         BICW2   #^X1FFF, R3          ; Clear the rounding bit so that
        52    52 33FD   006A   283         CVTGF   R2, R2               ;  CVTGF truncates instead of rounds.
                        006E   284 ;
                        006E   285 ; Back to original code.
                        006E   286 ;
        50    52   40   006E   287         ADDF    R2, R0               ; R0 = Y1 + higher part(X/Y1)
  50    0080 8F   A2    0071   288         SUBW    #^X80, R0            ; R0 = SQRT(X) = (1/2) (Y1 + X/Y1)
                  05    0076   289 SQRTX:  RSB                          ; return, R0 = result
                        0077   290
                        0077   291 ; X =< 0
                        0077   292 ;
                        0077   293 ZERO_NEG:
              FD   13   0077   294         BEQL    SQRTX                ; return with R0 = result = 0
              6E   DD   0079   295         PUSHL   (SP)                 ; return PC from JSB routine
    7E    00'8F   9A   007B   296         MOVZBL  #MTH$K_SQUROONEG, -(SP) ; condition value
  50    01   0F   78   007F   297         ASHL    #15, #1, R0          ; R0 = result = reserved operand -0.0
                        0083   298                                     ; R0 goes to signal mechanism vector
                        0083   299                                     ; (CHF$L_MCH_R0/R1) so error handler
                        0083   300                                     ; can modify the result.
00000000'GF     02    FB   0083   301         CALLS   #2, G^MTH$$SIGNAL    ; signal error and use real user's PC
                        008A   302                                     ; independent of CALL vs JSB
                  05    008A   303         RSB                          ; return - R0 restored from CHF$L_MCH_R0/R1
                        008B   304
                        008B   305         .END
```

```
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             00000076 R      01
X               = 00000004
ZERO_NEG          00000077 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	0000008B (139.)	01 (1.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.11	00:00:00.68
Command processing	107	00:00:00.64	00:00:02.67
Pass 1	82	00:00:00.81	00:00:03.90
Symbol table sort	0	00:00:00.00	00:00:00.00
Pass 2	68	00:00:00.68	00:00:02.45
Symbol table output	3	00:00:00.03	00:00:00.21
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	293	00:00:02.29	00:00:09.93

The working set limit was 900 pages.
4312 bytes (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.
365 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$:UVXSQRT/OBJ=OBJ$:UVXSQRT MSRC$:MTHJACKET/UPDATE=(ENH$:MTHJACKET)+MSRC$:

NCP
MAP

UUXSQRTR2
LIS

UUXSQRT
LIS

NCPDEF
SDL

UUXSINCOS
LIS

NMADEF
SDL

NCP