```
MMM         MMM  TTTTTTTTTTTTTTT  HHH         HHH  RRRRRRRRRRRR     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        HHHHHHHHHHHHHHH  RRRRRRRRRRRR           TTT        LLL
MMM         MMM        TTT        HHHHHHHHHHHHHHH  RRRRRRRRRRRR           TTT        LLL
MMM         MMM        TTT        HHHHHHHHHHHHHHH  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        LLLLLLLLLLLLLLL
MMM         MMM        TTT        HHH         HHH  RRR         RRR        TTT        LLLLLLLLLLLLLLL
MMM         MMM        TTT        HHH         HHH  RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

```
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  MM        TT      HH      HH  GG          SS          QQ      QQ  RR      RR      TT
MM  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  QQ  RR  RR          TT
MM        MM      TT      HH      HH  GG    GGGGGG        SS  QQ  QQ  QQ  RR  RR          TT
MM        MM      TT      HH      HH  GG        GG        SS  QQ    QQ    RR    RR        TT      ....
MM        MM      TT      HH      HH  GG        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
```

```
0000          .TITLE  MTH$GSQRT        ; G Floating Point Square Root routine
0000                                   ; (GSQRT)
0000          .IDENT /1-004/           ; File: MTHGSQRT.MAR    EDIT: RNH1004
0000     ;
0000     ;****************************************************************************
0000     ;*                                                                          *
0000     ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000     ;*  ALL RIGHTS RESERVED.                                                    *
0000     ;*                                                                          *
0000     ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000     ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000     ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000     ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000     ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000     ;*  TRANSFERRED.                                                            *
0000     ;*                                                                          *
0000     ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000     ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000     ;*  CORPORATION.                                                            *
0000     ;*                                                                          *
0000     ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000     ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000     ;*                                                                          *
0000     ;*                                                                          *
0000     ;****************************************************************************
0000     ;
0000     ;
0000     ; FACILITY: MATH LIBRARY
0000     ;++
0000     ; ABSTRACT:
0000     ;
0000     ; MTH$GSQRT is a function which returns the G-floating square root of
0000     ; its G-floating argument.  The call is standard call-by-reference.
0000     ;
0000     ; MTH$GSQRT_R5  is  a  special routine  which  is the same as MTH$GSQRT
0000     ; except  a  faster non-standard JSB call is used with the  argument in
0000     ; R0 and no registers are saved.
0000     ;
0000     ;--
0000     ;
0000     ; VERSION: 1
0000     ;
0000     ; HISTORY:
0000     ; AUTHOR:
0000     ;       Steven B. Lionel, 08-Jan-79: Version 1
0000     ;
0000     ; MODIFIED BY:
0000     ;
0000     ;
0000     ;
```

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

```
                              0000          .SBTTL  DECLARATIONS    ; Declarative Part of Module
                              0000
                              0000    ;
                              0000    ; INCLUDE FILES:       OTSPARAMS.MAR
                              0000    ;
                              0000    ; EXTERNAL SYMBOLS:
                              0000    ;
                              0000          .DSABL  GBL                       ; Declare all externals
                              0000          .EXTRN  MTH$$SIGNAL               ; SIGNAL SEVERE error
                              0000          .EXTRN  MTH$K_SQUROONEG           ; Error code
                              0000
                              0000    ;
                              0000    ; EQUATED SYMBOLS:
                              0000    ;
                              0000
                    0000403C  0000          ACMASK = ^M<IV, R2, R3, R4, R5> ; register save mask and IV enable
                              0000    ;
                              0000    ; MACROS:       none
                              0000    ;
                              0000    ; PSECT DECLARATIONS:
                              0000
                          00000000          .PSECT  _MTH$CODE       PIC,SHR,LONG,EXE,NOWRT
                              0000                                      ; program section for math routines
                              0000    ;
                              0000    ; OWN STORAGE:
                              0000    ;
                              0000    ;
                              0000    ; Constants A and B chosen for k = odd
                              0000    ;
                              0000    LG_ODD_A_E511:
 0000 A000 8279 5FFA          0000          .WORD   ^0057772, ^0101171, ^0120000, ^0000000
                              0008    LG_ODD_B_EM511:
 0000 4000 0789 2003          0008          .WORD   ^0020003, ^0003611, ^0040000, ^0000000
                              0010    ;
                              0010    ; Constants A and B chosen for k = even
                              0010    ;
                              0010    LG_EVEN_A:
 0000 4000 BEC3 4002          0010          .WORD   ^0040002, ^0137303, ^0040000, ^0000000
                              0018    LG_EVEN_B_EM512:
 0000 6000 E964 1FFA          0018          .WORD   ^0017772, ^0164544, ^0060000, ^0000000
```

```
          0020          .SBTTL  MTH$GSQRT  - Standard G-Floating GSQRT
          0020
          0020
          0020  ;++
          0020  ; FUNCTIONAL DESCRIPTION:
          0020  ;
          0020  ; GSQRT  - G-floating point function
          0020  ;
          0020  ; GSQRT(X) is computed using the following approximation technique:
          0020  ;
          0020  ;       If X < 0, error.  If X = 0, return GSQRT(X) = 0.
          0020  ;
          0020  ;       Let X = 2**K * F where F is the fractional part.
          0020  ;
          0020  ;       If K = even, X = 2**(2P) * F,
          0020  ;               GSQRT(X) = 2**P * GSQRT(F), 1/2 =< F < 1
          0020  ;
          0020  ;       If K = odd, X = 2**(2P+1) * F = 2**(2P+2) * (F/2),
          0020  ;               GSQRT(X) = 2**(P+1) * GSQRT(F/2), 1/4 =< F/2 < 1/2.
          0020  ;
          0020  ;       Let F' = A*F + B,
          0020  ;                       A = 0.453730314(octal),
          0020  ;                       B = 0.327226214(octal), for K = even.
          0020  ;               = A*(F/2) + B,
          0020  ;                       A = 0.650117146(octal),
          0020  ;                       B = 0.230170444(octal), for K = odd.
          0020  ;       and
          0020  ;           K' = P,       for K = even
          0020  ;              = P + 1    for K = odd.
          0020  ;
          0020  ;       Let Y0 = 2**K' * F' as a straight line approximation within the
          0020  ;       given interval using coefficients A and B which minimize the
          0020  ;       absolute error at the midpoint and endpoint.
          0020  ;
          0020  ;       Starting with Y0, three Newton-Raphson iterations are performed.
          0020  ;
          0020  ;       Y[n+1] = (1/2) * ( Y[n] + X/Y[n])
          0020  ;
          0020  ;       The relative error is < 10**-17.
          0020  ;
          0020  ; CALLING SEQUENCE:
          0020  ;
          0020  ;       gsqrt.wg.v = MTH$GSQRT(x.rg.r)
          0020  ;
          0020  ; INPUT PARAMETERS:
          0020
00000004  0020          LONG = 4                        ; define longword multiplier
00000004  0020          x = 1 * LONG                    ; Contents of x is the argument
          0020
          0020  ; IMPLICIT INPUTS:      none
          0020  ;
          0020  ; OUTPUT PARAMETERS:
          0020  ;
          0020  ;       VALUE:  G-floating square root of the argument
          0020  ;
          0020  ; IMPLICIT OUTPUTS:     none
          0020  ;
```

```
                        0020  166 ; COMPLETION CODES:     none
                        0020  167 ;
                        0020  168 ; SIDE EFFECTS:
                        0020  169 ;
                        0020  170 ; Signals: MTH$_SQUROONEG if X < 0.0 with reserved operand in R0/R1
                        0020  171 ; (copied to the signal mechanism vector CHF$L_MCH_R0/R1 by LIB$SIGNAL).
                        0020  172 ; Associated message is: "SQUARE ROOT OF NEGATIVE VALUE". Result is reserved
                        0020  173 ; operand -0.0 unless a user supplied (or any) error handler changes CHF$L_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                                         ; case of an error in special routine
      50   04 BC 50FD   0029  186        MOVG    @x(AP), R0                 ; R0/R1 = arg
            01   10     002E  187        BSBB    MTH$GSQRT_R5               ; call kernel GSQRT rountine
                 04     0030  188        RET                                ; return - result in R0/R1
```

```
                    0031          .SBTTL  MTH$GSQRT_R5  - Special GSQRT routine
                    0031
                    0031  ; Special GSQRT - used by the standard routine, and directly.
                    0031  ;
                    0031  ; CALLING SEQUENCE:
                    0031  ;       save anything in R2:R5
                    0031  ;       MOVG    ..., R0                 ; input in R0/R1
                    0031  ;       JSB     MTH$GSQRT_R5
                    0031  ;       return with result in R0/R1
                    0031  ; Note: This routine is written to avoid causing any integer overflows, floating
                    0031  ; overflows, floating underflows or divide by 0 conditions, whether enabled or
                    0031  ; not.
                    0031  ;
                    0031  ; REGISTERS USED:
                    0031  ;       R0/R1 - Floating argument then result
                    0031  ;       R2/R3 - scratch
                    0031  ;       R4/R5 - hold X during calc of F', K'.
                    0031
                    0031  MTH$GSQRT_R5::                          ; JSB routine for GSQRT
      54    50 50FD 0031          MOVG    R0, R4                  ; test sign of X and save it in R4/R5.
         54      15 0035          BLEQ    ZERO_NEG                ; branch to ZERO_NEG if X =< 0
                    0037  ;
                    0037  ; X > 0
                    0037  ;
      52    50   3C 0037  POS:    MOVZWL  R0, R2                  ; isolate low 16 bits in R2
                    003A                                          ; (sign, exp, 4 fract bits)
      52    1F   8A 003A          BICB2   #^X1F, R2               ; clear fraction, 1 exp bit
      50    52   AA 003D          BICW    R2, R0                  ; clear exponent bits in R0
   0C 50    04   E1 0040          BBC     #4, R0, EVEN            ; branch if exponent even
   50   B8 AF 44FD  0044          MULG2   LG_ODD_A_E511, R0       ; add 511 (half of bias) to
                    0049                                          ; (exponent-2)
                    0049                                          ; and start approximation calc
   50   BB AF 40FD  0049          ADDG2   LG_ODD_B_EM511, R0      ; R0 = (first approx) * 2**-512
         0F      11 004E          BRB     ADJUST                  ; go adjust
                    0050
                    0050  EVEN:
 50   2000 8F    A0 0050          ADDW2   #^X2000, R0             ; exp is 0 - make it 512
  50    B7 AF 44FD  0055          MULG2   LG_EVEN_A, R0
  50    BA AF 40FD  005A          ADDG2   LG_EVEN_B_EM512, R0     ; R0 = (first approx) * 2**-512
                    005F  ADJUST:
 52   52    1F   9C 005F          ROTL    #31, R2, R2             ; divide R2 (exp+bias) by 2,
                    0063                                          ; giving (exp/2+512)
      50    52   A0 0063          ADDW    R2, R0                  ; insert exp/2 in first approx and
                    0066                                          ; re-bias it.
                    0066
                    0066  ; first iteration, all done in G floating because of exponent range
                    0066  ;
 52   54    50 47FD 0066          DIVG3   R0, R4, R2              ; R2 = X/Y0
      50    52 40FD 006B          ADDG2   R2, R0                  ; R0 = Y0 + X/Y0
      50    10   A2 006F          SUBW    #^X10, R0               ; R0 = Y1 = (1/2)(Y0 + X/Y0)
                    0072                                          ; no overflow possible
                    0072
                    0072  ; second iteration
                    0072  ;
 52   54    50 47FD 0072          DIVG3   R0, R4, R2              ; R2/R3 = X/Y1
      50    52 40FD 0077          ADDG2   R2, R0                  ; R0/R1 = Y1 + higher part(X/Y1)
      50    10   A2 007B          SUBW    #^X10, R0               ; R0/R1 = Y2 = (1/2) (Y1+X/Y1)
```

```
                          007E
                          007E     ; third iteration
                          007E     ;
     52    54    50 47FD  007E             DIVG3   R0, R4, R2             ; R2/R3 = X/Y2
           50    52 40FD  0083             ADDG2   R2, R0                 ; R0 = Y2+X/Y2
           50    10   A2  0087             SUBW    #^X10, R0              ; R0/R1 = GSQRT(X) =
                          008A                                            ; (1/2) (Y2+X/Y2)
                      05  008A     SQRTX:  RSB                            ; return, R0/R1 = result
                          008B
                          008B     ; X =< 0
                          008B     ;
                          008B     ZERO_NEG:
                 FD   13  008B             BEQL    SQRTX                  ; return with R0 = result = 0
                 6E   DD  008D             PUSHL   (SP)                   ; return PC from JSB routine
        7E    00'8F   9A  008F             MOVZBL  #MTH$K_SQUROONEG, -(SP) ; condition value
     50    01    0F   79  0093             ASHQ    #15, #1, R0            ; R0/R1 = result = reserved operand -0.0
                          0097                                            ; R0/R1 goes to signal mechanism vector
                          0097                                            ; (CHF$L_MCH_R0/R1) so error handler
                          0097                                            ; can modify the result.
   00000000'GF   02   FB  0097             CALLS   #2, G^MTH$$SIGNAL      ; signal error and use real user's PC
                          009E                                            ; independent of CALL vs JSB
                      05  009E             RSB                            ; return - R0 restored from CHF$L_MCH_R0/R1
                          009F
                          009F             .END
```

```
ACMASK           = 0000403C
ADJUST             0000005F R      01
EVEN               0000005C 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
SQRTX              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   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
_MTH$CODE                       0000009F  (   159.)  01 (   1.)    PIC   USR   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
```

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

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

MTH
VAX
88
The
MAC

MTHGSIGN
LIS

MTHHCOSH
LIS

MTHHACOS
LIS

MTHGPROD
LIS

MTHGSINCO
LIS

MTHHASIN
LIS

MTHHATAN
LIS

MTHGTAN
LIS

MTHGTANH
LIS

MTHGSQRT
LIS

MTHHEXP
LIS

MTHGSINH
LIS

MTHHATANH
LIS