```
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**MTHEXP

```
MM      MM  TTTTTTTTTT  HH      HH  EEEEEEEEEE  XX      XX  PPPPPPPP
MM      MM  TTTTTTTTTT  HH      HH  EEEEEEEEEE  XX      XX  PPPPPPPP
MMMM  MMMM      TT      HH      HH  EE          XX      XX  PP      PP
MMMM  MMMM      TT      HH      HH  EE          XX      XX  PP      PP
MM  MM  MM      TT      HH      HH  EE            XX  XX    PP      PP
MM  MM  MM      TT      HH      HH  EE            XX  XX    PP      PP
MM      MM      TT      HHHHHHHHHH  EEEEEEEE        XX      PPPPPPPP
MM      MM      TT      HHHHHHHHHH  EEEEEEEE        XX      PPPPPPPP
MM      MM      TT      HH      HH  EE            XX  XX    PP
MM      MM      TT      HH      HH  EE            XX  XX    PP
MM      MM      TT      HH      HH  EE          XX      XX  PP              ....
MM      MM      TT      HH      HH  EE          XX      XX  PP              ....
MM      MM      TT      HH      HH  EEEEEEEEEE  XX      XX  PP              ....
MM      MM      TT      HH      HH  EEEEEEEEEE  XX      XX  PP              ....

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$EXP          ; Single Precision Floating Exponrntial
0000     2                                   ; Function (EXP)
0000     3          .IDENT /1-011/           ; File: MTHEXP.MAR  Edit: RNH1011
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$EXP is a function which returns the single floating point
0000    34 ; exponential of its single precision floating point argument.
0000    35 ; The call is standard call-by-reference.
0000    36 ;
0000    37 ;--
0000    38 ;
0000    39 ; VERSION: 0
0000    40 ;
0000    41 ; HISTORY:
0000    42 ; AUTHOR:
0000    43 ;       Peter Yuo, 15-Oct-76: Version 0
0000    44 ;
0000    45 ; MODIFIED BY:
0000    46 ;
0000    47 ; 0-1   Peter Yuo, 22-May-77
0000    48 ;
0000    49 ;
```

```
0000     51          .SBTTL  HISTORY ; Detailed Current Edit History
0000     52
0000     53 ; Edit History for Version 0 of MTH$EXPDEXP
0000     54 ;
0000     55 ; 0-1    Code saving after code review March 1977
0000     56 ; 0-2    Finish error handling 10-June-1977
0000     57 ; 0-4    change RET to RSB in ERROR; fix undefined GLOBLs
0000     58 ; 0-5    add $SRMDEF macro
0000     59 ; 0-6    MTH$$ERROR changed to MTH$$SIGNAL.
0000     60 ;        MTH$_... changed to MTH__....
0000     61 ;        Changed error handling mechanism. Put error result in R0 before
0000     62 ;        calling MTH$$SIGNAL in order to allow user modify error result.
0000     63 ; 0-7    Declared PSECTs and use SF$W_SAVE_PSW.  TNH 20-Dec-77
0000     64 ; 0-8    Invoke $SFDEF.  TNH 20-Dec-77
0000     65 ;
0000     66 ; Edit History for Version 1 of MTH$EXP
0000     67 ;
0000     68 ; 1-1    Split single and double precision routines into two parts;
0000     69 ;        Used more accurate and faster algorithms provided by M. Payne.
0000     70 ;        JMT 23-Jan-78
0000     71 ; 1-3    Fixed bug causing unexpected integer overflow.  JMT 24-MAR-78
0000     72 ; 1-4    Fixes for better accuracy.  Use ADDD.   JMT 11-Apr-78
0000     73 ; 1-5    Move .ENTRY mask definition to module header.  TNH 14-Aug-78
0000     74 ; 1-006 - Update version number and copyright notice.  JBS 16-NOV-78
0000     75 ; 1-007 - Change symbols MTH__FLOUNDMAT and MTH__FLOOVEMAT to
0000     76 ;           MTH$K_FLOUNDMAT and MTH$K_FLOOVEMAT, respectively.
0000     77 ;           JBS 07-DEC-78
0000     78 ; 1-008 - Add "_" to the PSECT directive.  JBS 22-DEC-78
0000     79 ; 1-009 - Declare externals.  SBL 17-May-1979
0000     80 ; 1-010 - Included logic to avoid the loss in significance in EMOD for
0000     81 ;           arguments greater than 2**4.  RNH 23-JUN-81
0000     82 ; 1-011 - Chaneged W^ to G^ in calls to MTH$$SIGNAL and MTH$$JACKET_TST
0000     83 ;           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$K_FLOUNDMAT
                     0000     96          .EXTRN  MTH$K_FLOOVEMAT
                     0000     97          .EXTRN  MTH$$SIGNAL                 ; SIGNAL SEVERE error
                     0000     98          .EXTRN  MTH$$JACKET_TST
                     0000     99
                     0000    100 ; EQUATED SYMBOLS:
                     0000    101
                0000401C     0000    102          ACMASK  = ^M<IV, R2, R3, R4>    ; register saving mask and IV enable
                00000029     0000    103          X_51    =       ^051            ; Extension for operand in EMODF
                     0000    104 ;
                     0000    105 ; MACROS:
                     0000    106          $SFDEF                              ; define SF$ (stack frame) symbols
                     0000    107 ;
                     0000    108 ; PSECT DECLARATIONS:
                     0000    109
                00000000     0000    110          .PSECT  _MTH$CODE       PIC,SHR,LONG,EXE,NOWRT
                     0000    111                                              ; program section for math routines
                     0000    112 ;
                     0000    113 ; OWN STORAGE:  none
                     0000    114 ;
                     0000    115 ; CONSTANTS:
                     0000    116
                     0000    117 ;
                     0000    118 ; Table to be used for scaling. These constants here have beer
                     0000    119 ; verified by M. Payne using her program ROOT16 on PDP-10.
                     0000    120 ;
                     0000    121 TABLE:
C000 4080            0000    122          .WORD   ^0040200,0                  ; 2**(0/16) = 1.0
AAC3 4085            0004    123          .WORD   ^0040205,^0125303           ; 2**(1/16)
95C2 408B            0008    124          .WORD   ^0040213,^0112702           ; 2**(2/16)
C3D3 4091            000C    125          .WORD   ^0040221,^0141723           ; 2**(3/16)
37F0 4098            0010    126          .WORD   ^0040230,^0033760           ; 2**(4/16)
F532 409E            0014    127          .WORD   ^0040236,^0172462           ; 2**(5/16)
FED7 40A5            0018    128          .WORD   ^0040245,^0177327           ; 2**(6/16)
583F 40AD            001C    129          .WORD   ^0040255,^0054077           ; 2**(7/16)
04F3 40B5            0020    130          .WORD   ^0040265,^0002363           ; 2**(8/16)
08A4 40BD            0024    131          .WORD   ^0040275,^0004244           ; 2**(9/16)
672A 40C5            0028    132          .WORD   ^0040305,^0063452           ; 2**(10/16)
248C 40CE            002C    133          .WORD   ^0040316,^0022214           ; 2**(11/16)
44FD 40D7            0030    134          .WORD   ^0040327,^0042375           ; 2**(12/16)
CCDF 40E0            0034    135          .WORD   ^0040340,^0146337           ; 2**(13/16)
C0C7 40EA            0038    136          .WORD   ^0040352,^0140307           ; 2**(14/16)
257D 40F5            003C    137          .WORD   ^0040365,^0022575           ; 2**(15/16)
                     0040    138 ;
                     0040    139 ; Table to be used for scaling. These constants here have been
                     0040    140 ; verified by M. Payne using her program ROOT16 on PDP-10.
                     0040    141 ;
```

```
                                        142 TABDB:   .WORD   ^0040200,0              ; 2**(0/16) = 1.0
0000 0000                               143          .WORD   0,0
AAC3 4085                               144          .WORD   ^0040205,^0125303       ; 2**(1/16)
487B 67CC                               145          .WORD   ^0063714,^0044173
95C1 408B                               146          .WORD   ^0040213,^0112701       ; 2**(2/16)
8BD7 E3EA                               147          .WORD   ^0161752,^0105727
C3D3 4091                               148          .WORD   ^0040221,^0141723       ; 2**(3/16)
11C3 73AB                               149          .WORD   ^0071653,^0010703
37F0 4098                               150          .WORD   ^0040230,^0033760       ; 2**(4/16)
B8A9 518D                               151          .WORD   ^0050615,^0134251
F532 409E                               152          .WORD   ^0040236,^0172462       ; 2**(5/16)
A112 6091                               153          .WORD   ^0060221,^0120422
FED6 40A5                               154          .WORD   ^0040245,^0177326       ; 2**(6/16)
5139 A9B1                               155          .WORD   ^0124661,^0050471
583E 40AD                               156          .WORD   ^0040255,^0054076       ; 2**(7/16)
A14B EA42                               157          .WORD   ^0165102,^0120513
04F3 40B5                               158          .WORD   ^0040265,^0002363       ; 2**(8/16)
DE65 33F9                               159          .WORD   ^0031771,^0157145
08A3 40BD                               160          .WORD   ^0040275,^0004243       ; 2**(9/16)
0C37 9F58                               161          .WORD   ^0117530,^0006067
672A 40C5                               162          .WORD   ^0040305,^0063452       ; 2**(10/16)
06DB 1155                               163          .WORD   ^0010525,^0003333
248C 40CE                               164          .WORD   ^0040316,^0022214       ; 2**(11/16)
8481 151F                               165          .WORD   ^0012437,^0102201
44FC 40D7                               166          .WORD   ^0040327,^0042374       ; 2**(12/16)
9D6B CAD6                               167          .WORD   ^0145326,^0116553
CCDE 40E0                               168          .WORD   ^0040340,^0146336       ; 2**(13/16)
94E1 EC2A                               169          .WORD   ^0166052,^0112341
C0C6 40EA                               170          .WORD   ^0040352,^0140306       ; 2**(14/16)
2439 E7DD                               171          .WORD   ^0163735,^0022071
257D 40F5                               172          .WORD   ^0040365,^0022575       ; 2**(15/16)
86CC 1524                               173          .WORD   ^0012444,^0103314
                                        174 ;
                                        175 ; Polynomial coefficient tables for POLYF.
                                        176 ;
                                        177 EXPTAB:
9924 351D                               178          .WORD   ^0032435,^0114444       ;
5F1B 3863                               179          .WORD   ^0034143,^0057433       ;
FDF0 3B75                               180          .WORD   ^0035565,^0176760       ;
7217 3E31                               181          .WORD   ^0037061,^0071027       ;
0000 0000                               182          .WORD   0,0                     ; 0.0
 00000005                               183 EXPLEN = <.-EXPTAB>/4
                                        184
                                        185 EXPTB1:
B333 3E2A                               186          .WORD   ^0037052,^0131463       ;
B555 3F2A                               187          .WORD   ^0037452,^0132525       ;
FFFF 3FFF                               188          .WORD   ^0037777,^0177777       ;
FFFF 407F                               189          .WORD   ^0040177,^0177777       ;
0000 4080                               190          .WORD   ^0040200,0              ; 1.00000000254251
 00000005                               191 EXPLN1 = <.-EXPTB1>/4
                                        192
                                        193 F_16LOG2_E:                              ; LOG2(E) * 16
AA3B 42B8                               194          .WORD   ^0041270,^0125073
                                        195 F_LN2_OV_16_HI:                          ; High 13 bits ln2/16
 70003E31                               196          .LONG   ^X70003E31
                                        197 F_LN2_OV_16_LO:                          ; Low bits of ln2/16
 FDF43705                               198          .LONG   ^XFDF43705
```

```
                00F4    199
                00F4    200
```

```
                        00F4   202          .SBTTL  MTH$EXP - Standard Single Precision Floating EXP
                        00F4   203
                        00F4   204
                        00F4   205 ;++
                        00F4   206 ; FUNCTIONAL DESCRIPTION:
                        00F4   207 ;
                        00F4   208 ; EXP  - single precision floating point function
                        00F4   209 ;
                        00F4   210 ; EXP(X) is computed using Chebyshev approximation 1001: about
                        00F4   211 ; 27 bit accuracy.
                        00F4   212 ;
                        00F4   213 ;
                        00F4   214 ; CALLING SEQUENCE:
                        00F4   215 ;
                        00F4   216 ;       Exponential.wf.v = MTH$EXP(x.rf.r)
                        00F4   217 ;
                        00F4   218 ; INPUT PARAMETERS:
                        00F4   219
                00000004 00F4   220          LONG = 4                          , define longword multiplier
                00000004 00F4   221          x = 1 * LONG                      ; Contents of x is the argument
                        00F4   222
                        00F4   223 ; IMPLICIT INPUTS:      none
                        00F4   224 ;
                        00F4   225 ; OUTPUT PARAMETERS:
                        00F4   226 ;
                        00F4   227 ;       VALUE:  floating exponential of the argument
                        00F4   228 ;
                        00F4   229 ; IMPLICIT OUTPUTS:     none
                        00F4   230 ;
                        00F4   231 ; SIDE EFFECTS:
                        00F4   232 ;
                        00F4   233 ; Signals: MTH$_FLOOVEMAT if X > 88.028 with reserved operand in R0/R1 (copied
                        00F4   234 ; to the signal mechanism vector CHF$L_MCH_R0/R1 by LIB$SIGNAL).  Associated
                        00F4   235 ; message is: "FLOATING OVERFLOW IN MATH LIBRARY". Result is reserved operand
                        00F4   236 ; -0.0 unless a user supplied (or any) error handler changes CHF$L_MCH_R0/R1.
                        00F4   237 ;       MTH$_FLOUNDMAT if X =< -89.416 and caller has hardware enable set.
                        00F4   238 ; The result is set to +0.0. Associated message is: "FLOATING UNDERFLOW
                        00F4   239 ; IN MATH LIBRARY"
                        00F4   240 ;
                        00F4   241 ; NOTE: This procedure disables floating point underflow, enable integer
                        00F4   242 ; overflow, causes no floating overflow or other arithmetic traps, and
                        00F4   243 ; preserves enables across the call.
                        00F4   244 ;
                        00F4   245 ;---
                        00F4   246
                        00F4   247
                   401C 00F4   248          .ENTRY  MTH$EXP, ACMASK ; standard call-by-reference entry
                        00F6   249                                   ; disable DV (and FU), enable IV
                        00F6   250          MTH$FLAG_JACKET          ; flag that this is a jacket procedure
                        00F6
    6D  00000000'GF  9E 00F6                MOVAB   G^MTH$$JACKET_HND, (FP)
                        00FD                                         ; set handler address to jacket
                        00FD                                         ; handler
                        00FD
                        00FD   251                                   ; in case of an error in special JSB
                        00FD   252                                   ; routine
    50  04 BC    50  00FD   253          MOVF    @x(AP), R0           ; R0 = user's arg
```

```
01   10  0101   254         BSBB    MTH$EXP_R4              ; R0 = special EXP(R0)
     04  0103   255         RET                             ; return - result in R0
```

```
                                0104    257          .SBTTL  MTH$EXP_R4 - Special EXP routine
                                0104    258
                                0104    259 ; Special EXP - used by the standard, and direct interfaces.
                                0104    260 ;
                                0104    261 ; CALLING SEQUENCE:
                                0104    262 ;       save anything needed in R0:R4
                                0104    263 ;       MOVF    ...,R0                  ; input in R0
                                0104    264 ;       JSB     MTH$EXP_R4
                                0104    265 ;       return with result in R0
                                0104    266
                                0104    267 ; Note: This routine is written to avoid causing any integer overflows,
                                0104    268 ; floating overflows, or floating underflows or divide by 0 conditions,
                                0104    269 ; whether enabled or  not.
                                0104    270 ;
                                0104    271 ; REGISTERS USED:
                                0104    272 ;       R0 - floating argument, then result
                                0104    273 ;       R2 - diddled exponent
                                0104    274 ;       R3 - scratch
                                0104    275 ;       R4 - integer part of X * LOG2(E)* 16
                                0104    276
                                0104    277
                                0104    278 MTH$EXP_R4::                             ; special EXP routine
     52   50   8000 8F   AB     0104    279 OVUND:  BICW3   #^X8000, R0, R2          ; Preliminary test for over/underflow
     53   52   3E00 8F   A3     010A    280         SUBW3   #^X3E00, R2, R3          ; R3 = (4+exponent) + 1st 7 fract bits
          53   05B0 8F   B1     0110    281         CMPW    #^X5B0, R3               ; Compare |X| with 88
                       5C   1F  0115    282         BLSSU   SMTST                    ; to more tests if LSSU
                                0117    283                                          ;  else, -4 < unbiased exp < 8
                                0117    284                                          ;  no exceptions in EMODF or POLYF
                                0117    285
               4280 8F   52     B1     0117    286         CMPW    R2, #^X4280      ; Check for loss of significance in
                                011C    287                                          ; EMOD ( |X| >= 2**4
                       20   19  011C    288         BLSS    EVAL                     ; No loss of significance
                                011E    289
                                011E    290 ;
                                011E    291 ; |X| >= 2**4.  EMOD will lose significance so the interger and fractional
                                011E    292 ; parts of X*16/ln2 must be obtained in seperate steps.
                                011E    293 ;
          51   50    C7 AF     45     011E    294         MULF3   F_16LOG2_E, R0, R1   ; Get integer part of X*16/ln2 in
                  54   51      4A     0123    295         CVTFL   R1, R4               ;    R4 (=I+J) as a longword and in
                  51   54      4E     0126    296         CVTLF   R4, R1               ;    R1 in F format
          52   51    C0 AF     45     0129    297         MULF3   F_LN2_OV_16_HI, R1, R2 ; Get fraction part of X*16/ln2 =
                  50   52      42     012E    298         SUBF    R2, R0               ;    16/ln2*[ X - (I+J)*ln2/16 ]
                  51    BC AF  44     0131    299         MULF    F_LN2_OV_16_LO, R1   ;    in R0.
                  50   51      42     0135    300         SUBF    R1, R0               ;
                  50    AD AF  44     0138    301         MULF    F_16LOG2_E, R0       ;
                       07   11      013C    302         BRB     POLY                 ;
                                013E    303
50   54   50   29   A7 AF     54     013E    304 EVAL:   EMODF   F_16LOG2_E, #X_51, R0, R4, R0
                                0145    305                                          ; get X*16*LG2(E) with
                                0145    306                                          ;  integer part in R4 (=I+J)
                                0145    307                                          ;  fraction = W in R0/R1
          FF75 CF   04   50     55     0145    308 POLY:   POLYF   R0, #EXPLEN-1, EXPTAB ; evaluate polynomial ap-
                                014B    309                                          ;  proximation with POLY.
                                014B    310                                          ;  5 coefficients.
     52   54   FFFFFFF0 8F     CB     014B    311         BICL3   #-16, R4, R2         ; R2 = J
               50   FEA8 CF42 44     0153    312         MULF    TABLE[R2], R0        ; else MUL by 2**(J/16)
               50   FEE2 CF42 60     0159    313         ADDD    TABDB[R2], R0        ; add in DP 2**(J/16)
```

```
                50   50   76   015F   314          CVTDF   R0, R0
                54   0F   CA   0162   315          BICL    #15, R4                 ; R4 = I
                     0B   13   0165   316          BEQL    RETURN                  ; if I=0, then done
                50   6044 7E   0167   317          MOVAQ   (R0)[R4], R0            ; shift I to exp position and
                                016B   318                                          ; MUL by 2**I by exponent addition
             007F 8F   50   B1   016B   319          CMPW    R0, #^X7F               ; test for over/underflow
                     3D   15   0170   320          BLEQ    EXCEPT                  ; see what exception is if neg or = 0
                          05   0172   321 RETURN:   RSB                             ; otherwise return result in R0
                                0173   322
                                0173   323 SMTST:
                     12   19   0173   324          BLSS    TOOBIG                  ; exception if exp+4 > 11
             3400 8F   52   B1   0175   325          CMPW    R2, #^X3400             ; eliminate underflow if exp <-24
                     07   19   017A   326          BLSS    ONE                     ; bypass if E**ARG = 1
         FF52 CF   04   50   55   017C   327          POLYF   R0, #EXPLN1-1, EXPTB1   ; evaluate alternate polynomial
                          05   0182   328          RSB
                                0183   329
                50   08   50   0183   330 ONE:     MOVF    S^#1.0, R0              ; E**ARG =1; store it
                          05   0186   331          RSB                             ; and return
                                0187   332
                                0187   333
                                0187   334 ;
                                0187   335 ; Handlers for software detected over/underflow conditions follow
                                0187   336 ;
                50   53   0187   337 TOOBIG:  TSTF    R0                      ; if big ARG > 0 goto OVERFLOW
                     28   18   0189   338          BGEQ    OVER
                                018B   339 ;
                                018B   340 ; Underflow; if user has FU set, signal error.  Always return 0.0
                                018B   341 ;
                                018B   342 UNDER:
                52   DC   018B   343          MOVPSL  R2                      ; R2 = user's or jacket routine's PSL
00000000'GF     00   FB   018D   344          CALLS   #0, G^MTH$$JACKET_TST   ; R0 = TRUE if JSB from jacket routine
                04   50   E9   0194   345          BLBC    R0, 10$                 ; branch if user did JSB
           52   04   AD   3C   0197   346          MOVZWL  SF$W_SAVE_PSW(FP), R2   ; get user PSL saved by CALL
                50   D4   019B   347 10$:     CLRL    R0                      ; R0 = result. LIB$SIGNAL will save in
                                019D   348                                          ; CHF$L_MCH_R0/R1 so any handler can fixup
           0D   52   06   E1   019D   349          BBC     #6, R2, 20$             ; has user enabled floating underflow?
                     6E   DD   01A1   350          PUSHL   (SP)                    ; yes, return PC from special routine
           7E   00'8F   9A   01A3   351          MOVZBL  #MTH$K_FLOUNDMAT, -(SP) ; trap code for hardware floating underflow
                                01A7   352                                          ; convert to MTH$_FLOUNDMAT (32-bit VAX-11
                                01A7   353                                          ; exception code)
00000000'GF     02   FB   01A7   354          CALLS   #2, G^MTH$$SIGNAL       ; signal (condition, PC)
                          05   01AE   355 20$:     RSB                             ; return
                                01AF   356
                                01AF   357 EXCEPT:
                54   D5   01AF   358          TSTL    R4                      ; test sign of I; if I < 0
                     D8   19   01B1   359          BLSS    UNDER                   ; go to underflow handler
                                01B3   360 ;
                                01B3   361 ; Signal floating overflow, return reserved operand, -0.0
                                01B3   362 ;
                                01B3   363 OVER:                                ; else process for overflow
                     6E   DD   01B3   364          PUSHL   (SP)                    ; return PC from special routine
             7E   00'8F   9A   01B5   365          MOVZBL  #MTH$K_FLOOVEMAT, -(SP) ; hardware floating overflow
          50   01   0F   79   01B9   366          ASHQ    #15, #1, R0             ; R0 = result = reserved operand
                                01BD   367                                          ; -0.0. R0 will be copied to
                                01BD   368                                          ; signal mechanism vector (CHF$L_MCH_R0/R1)
                                01BD   369                                          ; so can be fixed up by any error
                                01BD   370                                          ; handler
```

```
                00000000'GF   02   FB  01BD    371          CALLS   #2, G^MTH$$SIGNAL       ; signal (condition, PC)
                              05   01C4    372          RSB                             ; return - R0 restoredd from CHF$L_MCH_R0/R1
                                   01C5    373
                                   01C5    374
                                   01C5    375          .END
```

```
ACMASK          =  0000401C
EVAL               0000013E  R      02
EXCEPT             000001AF  R      02
EXPLEN          =  00000005
EXPLN1          =  00000005
EXPTAB             000000C0  R      02
EXPTB1             000000D4  R      02
F_16LOG2_E         000000E8  R      02
F_LN2_OV_16_HI     000000EC  R      02
F_LN2_OV_16_LO     000000F0  R      02
LONG            =  00000004
MTH$$JACKET_HND    ********    X    02
MTH$$JACKET_TST    ********    X    00
MTH$$SIGNAL        ********    X    00
MTH$EXP            000000F4  RG     02
MTH$EXP_R4         00000104  RG     02
MTH$K_FLOOVEMAT    ********    X    00
MTH$K_FLOUNDMAT    ********    X    00
ONE                00000183  R      02
OVER               000001B3  R      02
OVUND              00000104  R      02
POLY               00000145  R      02
RETURN             00000172  R      02
SF$W_SAVE_PSW   =  00000004
SMTST              00000173  R      02
TABDB              00000040  R      02
TABLE              00000000  R      02
TOOBIG             00000187  R      02
UNDER              0000018B  R      02
X               =  00000004
X_51            =  00000029
```

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_MTH$CODE	000001C5 (453.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.10	00:00:01.06
Command processing	113	00:00:00.76	00:00:07.24
Pass 1	123	00:00:01.80	00:00:08.55
Symbol table sort	0	00:00:00.05	00:00:00.30
Pass 2	80	00:00:00.94	00:00:04.12
Symbol table output	4	00:00:00.05	00:00:00.07
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	357	00:00:03.72	00:00:21.37

The working set limit was 1050 pages.
7720 bytes (16 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 59 non-local and 2 local symbols.
435 source lines were read in Pass 1, producing 13 object records in Pass 2.
9 pages of virtual memory were used to define 8 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
_255DUA28:[SYSLIB]STARLET.MLB;2	4

88 GETS were required to define 4 macros.

There were no errors, warnings or information messages.

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

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