```
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
```

```
UU      UU  VV      VV  XX      XX  EEEEEEEEEE  XX      XX  PPPPPPPP
UU      UU  VV      VV  XX      XX  EEEEEEEEEE  XX      XX  PPPPPPPP
UU      UU  VV      VV  XX      XX  EE          XX      XX  PP      PP
UU      UU  VV      VV  XX      XX  EE          XX      XX  PP      PP
UU      UU  VV      VV    XX  XX    EE            XX  XX    PP      PP
UU      UU  VV      VV    XX  XX    EE            XX  XX    PP      PP
UU      UU  VV      VV      XX      EEEEEEEE        XX      PPPPPPPP
UU      UU  VV      VV      XX      EEEEEEEE        XX      PPPPPPPP
UU      UU  VV      VV    XX  XX    EE            XX  XX    PP
UU      UU  VV      VV    XX  XX    EE            XX  XX    PP
UU      UU    VV  VV     XX      XX  EE          XX      XX  PP              ....
UU      UU    VV  VV     XX      XX  EE          XX      XX  PP              ....
UUUUUUUUUU      VV       XX      XX  EEEEEEEEEE  XX      XX  PP              ....
UUUUUUUUUU      VV       XX      XX  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  UVX$EXP          ; Single Precision Floating Exponential TABG
0000     2                                   ; Function (EXP)
0000     3          .IDENT /1-012/           ; File: MTHEXP.MAR  Edit: JCW1012
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 ; 1-012 Jeffrey C. Wiener 22-Feb-83
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     84 ; 1-012 - Changed all references to D_floating point operations to
0000     85 ;         G_floating point operations. This was done to conform to
0000     86 ;         the policy that G_floating, not D_floating, should be used
0000     87 ;         to back-up F_floating point operations. I also corrected
0000     88 ;         some comments.  JCW 22-FEB-83
```

```
                     0000    90          .SBTTL  DECLARATIONS            ; Declarative Part of Module
                     0000    91
                     0000    92 ;
                     0000    93 ; INCLUDE FILES:        MTHJACKET.MAR
                     0000    94 ;
                     0000    95
                     0000    96 ;
                     0000    97 ; EXTERNAL SYMBOLS:
                     0000    98 ;
                     0000    99          .DSABL  GBL
                     0000   100          .EXTRN  MTH$K_FLOUNDMAT
                     0000   101          .EXTRN  MTH$K_FLOOVEMAT
                     0000   102          .EXTRN  MTH$$SIGNAL             ; SIGNAL SEVERE error
                     0000   103          .EXTRN  MTH$$JACKET_TST
                     0000   104
                     0000   105 ; EQUATED SYMBOLS:
                     0000   106
0000401C             0000   107          ACMASK  = ^M<IV, R2, R3, R4>    ; register saving mask and IV enable
00000029             0000   108          X_51    =       ^051            ; Extension for operand in EMODF
                     0000   109 ;
                     0000   110 ; MACROS:
                     0000   111          $SFDEF                          ; define SF$ (stack frame) symbols
                     0000   112 ;
                     0000   113 ; PSECT DECLARATIONS:
                     0000   114
                 00000000   115          .PSECT  _MTH$CODE       PIC,SHR,LONG,EXE,NOWRT
                     0000   116                                          ; program section for math routines
                     0000   117 ;
                     0000   118 ; OWN STORAGE:  none
                     0000   119 ;
                     0000   120 ; CONSTANTS:
                     0000   121
                     0000   122 ;
                     0000   123 ; Table to be used for scaling. These constants here have been
                     0000   124 ; verified by M. Payne using her program ROOT16 on PDP-10.
                     0000   125 ;
                     0000   126 TABLE:
0000 4080            0000   127          .WORD   ^0040200,0              ; 2**(0/16) = 1.0
AAC3 4085            0004   128          .WORD   ^0040205,^0125303       ; 2**(1/16)
95C2 408B            0008   129          .WORD   ^0040213,^0112702       ; 2**(2/16)
C3D3 4091            000C   130          .WORD   ^0040221,^0141723       ; 2**(3/16)
37F0 4098            0010   131          .WORD   ^0040230,^0033760       ; 2**(4/16)
F532 409E            0014   132          .WORD   ^0040236,^0172462       ; 2**(5/16)
FED7 40A5            0018   133          .WORD   ^0040245,^0177327       ; 2**(6/16)
583F 40AD            001C   134          .WORD   ^0040255,^0054077       ; 2**(7/16)
04F3 40B5            0020   135          .WORD   ^0040265,^0002363       ; 2**(8/16)
08A4 40BD            0024   136          .WORD   ^0040275,^0004244       ; 2**(9/16)
672A 40C5            0028   137          .WORD   ^0040305,^0063452       ; 2**(10/16)
248C 40CE            002C   138          .WORD   ^0040316,^0022214       ; 2**(11/16)
44FD 40D7            0030   139          .WORD   ^0040327,^0042375       ; 2**(12/16)
CCDF 40E0            0034   140          .WORD   ^0040340,^0146337       ; 2**(13/16)
C0C7 40EA            0038   141          .WORD   ^0040352,^0140307       ; 2**(14/16)
257D 40F5            003C   142          .WORD   ^0040365,^0022575       ; 2**(15/16)
                     0040   143 ;
                     0040   144 ;        This was used when D_float backed up F_floating
                     0040   145 ;
                     0040   146 ; Table to be used for scaling. These constants here have been
```

```
                     0040   147 ; verified by M. Payne using her program ROOT16 on PDP-10.
                     0040   148 ;
                     0040   149 ;TABDB:  .WORD   ^0040200,0              ; 2**(0/16) = 1.0
                     0040   150 ;         .WORD   0,0
                     0040   151 ;         .WORD   ^0040205,^0125303      ; 2**(1/16)
                     0040   152 ;         .WORD   ^0063714,^0044173
                     0040   153 ;         .WORD   ^0040213,^0112701      ; 2**(2/16)
                     0040   154 ;         .WORD   ^0161752,^0105727
                     0040   155 ;         .WORD   ^0040221,^0141723      ; 2**(3/16)
                     0040   156 ;         .WORD   ^0071653,^0010703
                     0040   157 ;         .WORD   ^0040230,^0033760      ; 2**(4/16)
                     0040   158 ;         .WORD   ^0050615,^0134251
                     0040   159 ;         .WORD   ^0040236,^0172462      ; 2**(5/16)
                     0040   160 ;         .WORD   ^0060221,^0120422
                     0040   161 ;         .WORD   ^0040245,^0177326      ; 2**(6/16)
                     0040   162 ;         .WORD   ^0124661,^0050471
                     0040   163 ;         .WORD   ^0040255,^0054076      ; 2**(7/16)
                     0040   164 ;         .WORD   ^0165102,^0120513
                     0040   165 ;         .WORD   ^0040265,^0002363      ; 2**(8/16)
                     0040   166 ;         .WORD   ^0031771,^0157145
                     0040   167 ;         .WORD   ^0040275,^0004243      ; 2**(9/16)
                     0040   168 ;         .WORD   ^0117530,^0006067
                     0040   169 ;         .WORD   ^0040305,^0063452      ; 2**(10/16)
                     0040   170 ;         .WORD   ^0010525,^0003333
                     0040   171 ;         .WORD   ^0040316,^0022214      ; 2**(11/16)
                     0040   172 ;         .WORD   ^0012437,^0102201
                     0040   173 ;         .WORD   ^0040327,^0042374      ; 2**(12/16)
                     0040   174 ;         .WORD   ^0145326,^0116553
                     0040   175 ;         .WORD   ^0040340,^0146336      ; 2**(13/16)
                     0040   176 ;         .WORD   ^0166052,^0112341
                     0040   177 ;         .WORD   ^0040352,^0140306      ; 2**(14/16)
                     0040   178 ;         .WORD   ^0163735,^0022071
                     0040   179 ;         .WORD   ^0040365,^0022575      ; 2**(15/16)
                     0040   180 ;         .WORD   ^0012444,^0103314
                     0040   181
                     0040   182 ;
                     0040   183 ; Table to be used for scaling. These constants are truncated
                     0040   184 ; conversions of the D_floating constants above to G_floating
                     0040   185 ; values.
                     0040   186 ;
                     0040   187
00000000 00004010    0040   188 TABGB:  .QUAD   ^X0000000000004010      ; 2**(0/16) = 1.0
890F6CF9 B5584010    0048   189         .QUAD   ^X890F6CF9B5584010      ; 2**(1/16)
517A3C7D 72884011    0050   190         .QUAD   ^X517A3C7D72B84011      ; 2**(2/16)
62386E75 387A4012    0058   191         .QUAD   ^X62386E75387A4012      ; 2**(3/16)
B7150A31 06FE4013    0060   192         .QUAD   ^XB7150A3106FE4013      ; 2**(4/16)
34224C12 DEA64013    0068   193         .QUAD   ^X34224C12DEA64013      ; 2**(5/16)
2A27D536 BFDA4014    0070   194         .QUAD   ^X2A27D536BFDA4014      ; 2**(6/16)
5429DD48 AB074015    0078   195         .QUAD   ^X5429DD48AB074015      ; 2**(7/16)
3BCC667F A09E4016    0080   196         .QUAD   ^X3BCC667FA09E4016      ; 2**(8/16)
018673EB A1144017    0088   197         .QUAD   ^X018673EBA1144017      ; 2**(9/16)
A0DB422A ACE54018    0090   198         .QUAD   ^XA0DB422AACE54018      ; 2**(10/16)
F09082A3 C4914019    0098   199         .QUAD   ^XF09082A3C4914019      ; 2**(11/16)
D3AD995A E89F401A    00A0   200         .QUAD   ^XD3AD995AE89F401A      ; 2**(12/16)
529CDD85 199B401C    00A8   201         .QUAD   ^X529CDD85199B401C      ; 2**(13/16)
A487DCFB 5818401D    00B0   202         .QUAD   ^XA487DCFB5818401D      ; 2**(14/16)
90D9A2A4 A4AF401E    00B8   203         .QUAD   ^X90D9A2A4A4AF401E      ; 2**(15/16)
```

```
                     00C0   204
                     00C0   205 ;
                     00C0   206 ; Polynomial coefficient tables for POLYF.
                     00C0   207 ;
                     00C0   208 EXPTAB:
9924 351D            00C0   209         .WORD   ^0032435,^0114444       ;
5F1B 3863            00C4   210         .WORD   ^0034143,^0057433       ;
FDF0 3B75            00C8   211         .WORD   ^0035565,^0176760       ;
7217 3E31            00CC   212         .WORD   ^0037061,^0071027       ;
0000 0000            00D0   213         .WORD   0,0                     ; 0.0
 00000005            00D4   214 EXPLEN = <.-EXPTAB>/4
                     00D4   215
                     00D4   216 EXPTB1:
B333 3E2A            00D4   217         .WORD   ^0037052,^0131463       ;
B555 3F2A            00D8   218         .WORD   ^0037452,^0132525       ;
FFFF 3FFF            00DC   219         .WORD   ^0037777,^0177777       ;
FFFF 407F            00E0   220         .WORD   ^0040177,^0177777       ;
0000 4080            00E4   221         .WORD   ^0040200,0              ; 1.00000000254251
 00000005            00E8   222 EXPLN1 = <.-EXPTB1>/4
                     00E8   223
                     00E8   224 F_16LOG2_E:                          ; LOG2(E) * 16
AA3B 42B8            00E8   225         .WORD   ^0041270,^0125073
                     00EC   226 F_LN2_OV_16_HI:                      ; High 13 bits ln2/16
 70003E31            00EC   227         .LONG   ^X70003E31
                     00F0   228 F_LN2_OV_16_LO:                      ; Low bits of ln2/16
 FDF43705            00F0   229         .LONG   ^XFDF43705
                     00F4   230
                     00F4   231
```

```
                     00F4    233          .SBTTL  MTH$EXP - Standard Single Precision Floating EXP
                     00F4    234
                     00F4    235
                     00F4    236 ;++
                     00F4    237 ; FUNCTIONAL DESCRIPTION:
                     00F4    238 ;
                     00F4    239 ; EXP  - single precision floating point function
                     00F4    240 ;
                     00F4    241 ; EXP(X) is computed using Chebyshev approximation 1001: about
                     00F4    242 ; 27 bit accuracy.
                     00F4    243 ;
                     00F4    244 ;
                     00F4    245 ; CALLING SEQUENCE:
                     00F4    246 ;
                     00F4    247 ;       Exponential.wf.v = MTH$EXP(x.rf.r)
                     00F4    248 ;
                     00F4    249 ; INPUT PARAMETERS:
                     00F4    250
00000004             00F4    251          LONG = 4                        ; define longword multiplier
00000004             00F4    252          x = 1 * LONG                    ; Contents of x is the argument
                     00F4    253
                     00F4    254 ; IMPLICIT INPUTS:      none
                     00F4    255 ;
                     00F4    256 ; OUTPUT PARAMETERS:
                     00F4    257 ;
                     00F4    258 ;       VALUE:  floating exponential of the argument
                     00F4    259 ;
                     00F4    260 ; IMPLICIT OUTPUTS:     none
                     00F4    261 ;
                     00F4    262 ; SIDE EFFECTS:
                     00F4    263 ;
                     00F4    264 ; Signals: MTH$_FLOOVEMAT if X > 88.028 with reserved operand in R0/R1 (copied
                     00F4    265 ; to the signal mechanism vector CHF$L_MCH_R0/R1 by LIB$SIGNAL).  Associated
                     00F4    266 ; message is: "FLOATING OVERFLOW IN MATH LIBRARY". Result is reserved operand
                     00F4    267 ; -0.0 unless a user supplied (or any) error handler changes CHF$L_MCH_R0/R1.
                     00F4    268 ;       MTH$_FLOUNDMAT if X =< -89.416 and caller has hardware enable set.
                     00F4    269 ; The result is set to +0.0. Associated message is: "FLOATING UNDERFLOW
                     00F4    270 ; IN MATH LIBRARY"
                     00F4    271 ;
                     00F4    272 ; NOTE: This procedure disables floating point underflow, enable integer
                     00F4    273 ; overflow, causes no floating overflow or other arithmetic traps, and
                     00F4    274 ; preserves enables across the call.
                     00F4    275 ;
                     00F4    276 ;---
                     00F4    277
                     00F4    278
                 401C 00F4    279          .ENTRY  MTH$EXP, ACMASK ; standard call-by-reference entry
                     00F6    280                                  ; disable DV (and FU), enable IV
                     00F6    281          MTH$FLAG_JACKET         ; flag that this is a jacket procedure
                     00F6    282
6D   00000000'GF  9E 00F6                 MOVAB   G^MTH$$JACKET_HND, (FP)
                     00FD                                         ; set handler address to jacket
                     00FD                                         ; handler
                     00FD
                     00FD    283                                  ; in case of an error in special JSB
                     00FD    284                                  ; routine
   50  04 BC   50    00FD              MOVF    @x(AP), R0          ; R0 = user's arg
```

```
01   10  0101   285          BSBB    MTH$EXP_R4              ; R0 = special EXP(R0)
     04  0103   286          RET                             ; return - result in R0
```

```
                     0104   288          .SBTTL  MTH$EXP_R4 - Special EXP routine
                     0104   289
                     0104   290 ; Special EXP - used by the standard, and direct interfaces.
                     0104   291 ;
                     0104   292 ; CALLING SEQUENCE:
                     0104   293 ;       save anything needed in R0:R4
                     0104   294 ;       MOVF    ...,R0                  ; input in R0
                     0104   295 ;       JSB     MTH$EXP_R4
                     0104   296 ;       return with result in R0
                     0104   297
                     0104   298 ; Note: This routine is written to avoid causing any integer overflows,
                     0104   299 ; floating overflows, or floating underflows or divide by 0 conditions,
                     0104   300 ; whether enabled or  not.
                     0104   301 ;
                     0104   302 ; REGISTERS USED:
                     0104   303 ;       R0 - floating argument, then result
                     0104   304 ;       R2 - diddled exponent
                     0104   305 ;       R3 - scratch
                     0104   306 ;       R4 - integer part of X * LOG2(E)* 16
                     0104   307
                     0104   308
                     0104   309 MTH$EXP_R4::                             ; special EXP routine
 52   50   8000 8F   AB  0104   310 OVUND:  BICW3   #^X8000, R0, R2       ; Preliminary test for over/underflow
 53   52   3E00 8F   A3  010A   311         SUBW3   #^X3E00, R2, R3       ; R3 = (4+exponent) + 1st 7 fract bits
      53   05B0 8F   B1  0110   312         CMPW    #^X5B0, R3            ; Compare |X| with 88
                   62   1F  0115   313         BLSSU   SMTST                 ; to more tests if LSSU
                     0117   314                                           ;  else, -4 < unbiased exp < 8
                     0117   315                                           ;  no exceptions in EMODF or POLYF
                     0117   316
           4280 8F   52   B1  0117   317         CMPW    R2, #^X4280           ; Check for loss of significance in
                     011C   318                                           ; EMOD ( |X| >= 2**4 )
                   20   19  011C   319         BLSS    EVAL                  ; No loss of significance
                     011E   320
                     011E   321 ;
                     011E   322 ; |X| >= 2**4.  EMOD will lose significance so the interger and fractional
                     011E   323 ; parts of X*16/ln2 must be obtained in seperate steps.
                     011E   324 ;
      51   50   C7 AF   45  011E   325         MULF3   F_16LOG2_E, R0, R1    ; Get integer part of X*16/ln2 in
           54   51   4A  0123   326         CVTFL   R1, R4                ;    R4 (=I+J) as a longword and in
           51   54   4E  0126   327         CVTLF   R4, R1                ;    R1 in F format
      52   51   C0 AF   45  0129   328         MULF3   F_LN2_OV_16_HI, R1, R2 ; Get fraction part of X*16/ln2 =
           50   52   42  012E   329         SUBF    R2, R0                ;    16/ln2*[ X - (I+J)*ln2/16 ]
           51   BC AF   44  0131   330         MULF    F_LN2_OV_16_LO, R1    ;    in R0.
           50   51   42  0135   331         SUBF    R1, R0                ;
           50   AD AF   44  0138   332         MULF    F_16LOG2_E, R0        ;
                07   11  013C   333         BRB     POLY                  ;
                     013E   334
50   54   50   29   A7 AF   54  013E   335 EVAL:   EMODF   F_16LOG2_E, #X_51, R0, R4, R0
                     0145   336                                           ; get X*16*LG2(E) with
                     0145   337                                           ;  integer part in R4 (=I+J)
                     0145   338                                           ;  fraction = W in R0/R1
     FF75 CF   04   50   55  0145   339 POLY:   POLYF   R0, #EXPLEN-1, EXPTAB ; evaluate polynomial ap-
                     014B   340                                           ;  proximation with POLY.
                     014B   341                                           ;  5 coefficients.
52   54   FFFFFFF0 8F   CB  014B   342         BICL3   #-16, R4, R2          ; R2 = J
     50   FEAB CF42   44  0153   343         MULF    TABLE[R2], R0         ; else MUL by 2**(J/16)
                     0159   344 ;
```

```
                                0159   345 ; Replace the following two lines of code with the corresponding
                                0159   346 ; G_floating code.
                                0159   347 ;
                                0159   348 ;       ADDD    TABDB[R2], R0           ; add in DP 2**(J/16)
                                0159   349 ;       CVTDF   R0, R0
                                0159   350 ;
                     50   50 99FD  0159   351         CVTFG   R0, R0
                50   FEDD CF42 40FD  015D   352         ADDG2   TABGB[R2], R0           ; add in DP 2**(J/16)
                     50   50 33FD  0164   353         CVTGF   R0, R0
                     54   0F   CA  0168   354         BICL    #15, R4                 ; R4 = I
                          0B   13  016B   355         BEQL    RETURN                  ; if I=0, then done
                     50   6044 7E  016D   356         MOVAQ   (R0)[R4], R0            ; shift I to exp position and
                                0171   357                                         ; MUL by 2**I by exponent addition
                   007F 8F   50   B1  0171   358         CMPW    R0, #^X7F               ; test for over/underflow
                             3D   15  0176   359         BLEQ    EXCEPT                  ; see what exception is if neg or = 0
                                  05  0178   360 RETURN: RSB                             ; otherwise return result in R0
                                0179   361
                                0179   362 SMTST:
                             12   19  0179   363         BLSS    TOOBIG                  ; exception if exp+4 > 11
                   3400 8F   52   B1  017B   364         CMPW    R2, #^X3400             ; eliminate underflow if exp <-24
                             07   19  0180   365         BLSS    ONE                     ; bypass if E**ARG = 1
                FF4C CF    04   50   55  0182   366         POLYF   R0, #EXPLN1-1, EXPTB1   ; evaluate alternate polynomial
                                  05  0188   367         RSB
                                0189   368
                     50   08   50   D0  0189   369 ONE:    MOVF    S^#1.0, R0              ; E**ARG =1; store it
                                  05  018C   370         RSB                             ; and return
                                018D   371
                                018D   372
                                018D   373 ;
                                018D   374 ; Handlers for software detected over/underflow conditions follow
                                018D   375 ;
                     50   53   018D   376 TOOBIG: TSTF    R0                      ; if big ARG > 0 goto OVERFLOW
                     28   18   018F   377         BGEQ    OVER
                                0191   378 ;
                                0191   379 ; Underflow; if user has FU set, signal error.  Always return 0.0
                                0191   380 ;
                                0191   381 UNDER:
                     52   DC   0191   382         MOVPSL  R2                      ; R2 = user's or jacket routine's PSL
00000000'GF          00   FB   0193   383         CALLS   #0, G^MTH$$JACKET_TST   ; R0 = TRUE if JSB from jacket routine
                04   50   E9  019A   384         BLBC    R0, 10$                 ; branch if user did JSB
             52 04   AD   3C  019D   385         MOVZWL  SF$W_SAVE_PSW(FP), R2   ; get user PSL saved by CALL
                     50   D4  01A1   386 10$:    CLRL    R0                      ; R0 = result. LIB$SIGNAL will save in
                                01A3   387                                         ; CHF$L_MCH_R0/R1 so any handler can fixup
             0D 52   06   E1  01A3   388         BBC     #6, R2, 20$             ; has user enabled floating underflow?
                     6E   DD  01A7   389         PUSHL   (SP)                    ; yes, return PC from special routine
             7E   00'8F   9A  01A9   390         MOVZBL  #MTH$K_FLOUNDMAT, -(SP) ; trap code for hardware floating underflow
                                01AD   391                                         ; convert to MTH$_FLOUNDMAT (32-bit VAX-11
                                01AD   392                                         ; exception code)
00000000'GF          02   FB   01AD   393         CALLS   #2, G^MTH$$SIGNAL       ; signal (condition, PC)
                          05   01B4   394 20$:    RSB                             ; return
                                01B5   395
                                01B5   396 EXCEPT:
                     54   D5   01B5   397         TSTL    R4                      ; test sign of I; if I < 0
                     D8   19   01B7   398         BLSS    UNDER                   ; go to underflow handler
                                01B9   399 ;
                                01B9   400 ; Signal floating overflow, return reserved operand, -0.0
                                01B9   401 ;
```

```
                     01B9   402 OVER:                                          ; else process for overflow
                 6E  DD  01B9   403          PUSHL   (SP)                      ; return PC from special routine
              7E  00'8F  9A  01BB   404          MOVZBL  #MTH$K_FLOOVEMAT, -(SP) ; hardware floating overflow
           50  01  0F  79  01BF   405          ASHQ    #15, #1, R0               ; R0 = result = reserved operand
                        01C3   406                                             ; -0.0. R0 will be copied to
                        01C3   407                                             ; signal mechanism vector (CHF$L_MCH_R0/R1)
                        01C3   408                                             ; so can be fixed up by any error
                        01C3   409                                             ; handler
00000000'GF  02  FB  01C3   410          CALLS   #2, G^MTH$$SIGNAL              ; signal (condition, PC)
                  05  01CA   411          RSB                                   ; return - R0 restored from CHF$L_MCH_R0/R1
                        01CB   412
                        01CB   413
                        01CB   414          .END
```

```
ACMASK          = 0000401C
EVAL              0000013E R      02
EXCEPT            000001B5 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               00000189 R      02
OVER              000001B9 R      02
OVUND             00000104 R      02
POLY              00000145 R      02
RETURN            00000178 R      02
SF$W_SAVE_PSW   = 00000004
SMTST             00000179 R      02
TABGB             00000040 R      02
TABLE             00000000 R      02
TOOBIG            0000018D R      02
UNDER             00000191 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 | 000001CB ( 459.) | 02 ( 2.) | PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG |

## Performance indicators

| Phase | Page faults | CPU Time | Elapsed Time |
|---|---|---|---|
| Initialization | 30 | 00:00:00.10 | 00:00:01.07 |
| Command processing | 116 | 00:00:00.69 | 00:00:03.15 |
| Pass 1 | 127 | 00:00:01.74 | 00:00:05.81 |
| Symbol table sort | 0 | 00:00:00.05 | 00:00:00.44 |
| Pass 2 | 86 | 00:00:00.99 | 00:00:04.45 |
| Symbol table output | 4 | 00:00:00.04 | 00:00:00.18 |
| Psect synopsis output | 3 | 00:00:00.03 | 00:00:00.08 |
| Cross-reference output | 0 | 00:00:00.00 | 00:00:00.00 |
| Assembler run totals | 368 | 00:00:03.64 | 00:00:15.19 |

The working set limit was 1050 pages.
7898 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.
474 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 |
|---|---|
| _$255$DUA28:[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$:UVXEXP/OBJ=OBJ$:UVXEXP MSRC$:MTHJACKET/UPDATE=(ENH$:MTHJACKET)+MSRC$:UV
```

OTSPOWHH
LIS

OTSPOWII
LIS

OTSPOWRJ
LIS

UUXPOWGG
LIS

OTSPOWGLU
LIS

OTSPOWHLU
LIS

OTSPOWHJ
LIS

UUXPOWCJ
LIS

UUXPOWRR
LIS

OTSPOWLUL
LIS

OTSPOWRR
LIS

OTSPOWJJ
LIS

UUXEXP
LIS

UUXGSINCO
LIS

OTSPOWRLU
LIS