```
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
```

```
  000000    TTTTTTTTTT    SSSSSSSS  PPPPPPPP      000000    WW      WW  RRRRRRRR    LL          UU      UU
  000000    TTTTTTTTTT    SSSSSSSS  PPPPPPPP      000000    WW      WW  RRRRRRRR    LL          UU      UU
00      00      TT      SS        PP      PP  00      00  WW      WW  RR      RR  LL          UU      UU
00      00      TT      SS        PP      PP  00      00  WW      WW  RR      RR  LL          UU      UU
00      00      TT      SS        PP      PP  00      00  WW      WW  RR      RR  LL          UU      UU
00      00      TT      SS        PP      PP  00      00  WW      WW  RR      RR  LL          UU      UU
00      00      TT        SSSSSS  PPPPPPPP    00      00  WW      WW  RRRRRRRR    LL          UU      UU
00      00      TT        SSSSSS  PPPPPPPP    00      00  WW      WW  RRRRRRRR    LL          UU      UU
00      00      TT              SS PP          00      00  WW  WW  WW  RR  RR      LL          UU      UU
00      00      TT              SS PP          00      00  WW  WW  WW  RR  RR      LL          UU      UU
00      00      TT              SS PP          00      00  WWWW  WWWW  RR    RR    LL          UU      UU      ....
00      00      TT              SS PP          00      00  WWWW  WWWW  RR    RR    LL          UU      UU      ....
  000000        TT      SSSSSSSS  PP            000000    WW      WW  RR      RR  LLLLLLLLLL  UUUUUUUUUU      ....
  000000        TT      SSSSSSSS  PP            000000    WW      WW  RR      RR  LLLLLLLLLL  UUUUUUUUUU      .. .

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

(2) 46 DECLARATIONS
(3) 82 OTS$POWRLU - F_floating ** unsigned integer power routine

```
0000     1          .TITLE  OTS$POWRLU - F_floating ** unsigned integer power routine
0000     2          .IDENT  /1-001/                 ; File: OTSPOWRLU.MAR Edit: SBL1001
0000     3
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  TH.  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 ;++
0000    30 ; FACILITY: Language support procedures, Mathematics division
0000    31 ;
0000    32 ; ABSTRACT:
0000    33 ;
0000    34 ;       This module contains OTS$POWRLU, a procedure which raises an
0000    35 ;       F_floating value to an unsigned integer power.
0000    36 ;
0000    37 ; ENVIRONMENT: Runs at any access mode, AST Reentrant
0000    38 ;
0000    39 ; AUTHOR: Steven B. Lionel, CREATION DATE: 22-JUL-1981
0000    40 ;
0000    41 ; MODIFIED BY:
0000    42 ;
0000    43 ; 1-001 - Original. SBL 22-JUL-1981
0000    44 ;--
```

```
                        .SBTTL  DECLARATIONS
                ;
                ; LIBRARY MACRO CALLS:
                ;
                        $SSDEF                  ; System error codes
                        $CHFDEF                 ; Define condition handler symbols.
                        $SFDEF                  ; Define stack frame symbols.
                        $PSLDEF                 ; Define program status longword
                                                ; symbols.
                ;
                ; EXTERNAL DECLARATIONS:
                ;
                        .DSABL  GBL             ; Force all external symbols to be declared
                        .EXTRN  MTH$K_UNDEXP    ; Undefined exponentiation
                        .EXTRN  MTH$K_FLOOVEMAT ; Floating overflow in Math Library
                        .EXTRN  MTH$K_FLOUNDMAT ; Floating underflow in Math Library
                        .EXTRN  MTH$$SIGNAL     ; Math error routine
                ;
                ; MACROS:
                ;
                        NONE
                ;
                ; EQUATED SYMBOLS:
                ;
                        NONE
                ;
                ; OWN STORAGE:
                ;
                        NONE
                ;
                ; PSECT DECLARATIONS:
                ;
00000000                .PSECT _OTS$CODE PIC, USR, CON, REL, LCL, SHR, -
                                EXE, RD, NOWRT, LONG
```

```
                     0000     82          .SBTTL  OTS$POWRLU - F_floating ** unsigned integer power routine
                     0000     83 ;++
                     0000     84 ; FUNCTIONAL DESCRIPTION:
                     0000     85 ;
                     0000     86 ;       Raise an F_floating value to an unsigned integer power giving
                     0000     87 ;       an F_floating result.
                     0000     88 ;
                     0000     89 ;       The result is given by:
                     0000     90 ;
                     0000     91 ;       base    exponent        result
                     0000     92 ;
                     0000     93 ;       any     > 0             product (base * 2**i) where i is each
                     0000     94 ;                               non-zero bit position in exponent
                     0000     95 ;
                     0000     96 ;       > 0     = 0             1.0
                     0000     97 ;       = 0     = 0             Undefined exponentation
                     0000     98 ;       < 0     = 0             1.0
                     0000     99 ;
                     0000    100 ;
                     0000    101 ; CALLING SEQUENCE:
                     0000    102 ;
                     0000    103 ;       result.wf.v = OTS$POWRLU (base.rf.v, exponent.rlu.v)
                     0000    104 ;
                     0000    105 ; FORMAL PARAMETERS:
                     0000    106 ;
                     0000    107
00000004             0000    108         base = 4        ; F_floating base
                     0000    109
00000008             0000    110         exponent = 8    ; Unsigned longword integer exponent
                     0000    111
                     0000    112 ;
                     0000    113 ; IMPLICIT INPUTS:
                     0000    114 ;
                     0000    115 ;       NONE
                     0000    116 ;
                     0000    117 ; IMPLICIT OUTPUTS:
                     0000    118 ;
                     0000    119 ;       NONE
                     0000    120 ;
                     0000    121 ; ROUTINE VALUE:
                     0000    122 ;
                     0000    123 ;       The base raised to the exponent's power.
                     0000    124 ;
                     0000    125 ; SIDE EFFECTS:
                     0000    126 ;
                     0000    127 ;       May signal:
                     0000    128 ;               MTH$_FLOOVEMAT - floating overflow in Math Library
                     0000    129 ;               MTH$_FLOUNDMAT - floating underflow in Math Library, only
                     0000    130 ;                                if the caller has FU enabled.
                     0000    131 ;               MTH$_UNDEXP    - undefined exponentiation, if both base and
                     0000    132 ;                                exponent are zero.
                     0000    133 ;
                     0000    134 ;--
                     0000    135
                0004 0000    136         .ENTRY OTS$POWRLU, ^M<R2>       ; Entry point
                     0002    137
6D    64'AF    9E  0002    138         MOVAB   B^EXC_HANDLER, (FP)     ; Translate exceptions to MTH$ errors
```

```
                0040 8F   B8  0006   139          BISPSW  #PSL$M_FU                       ; Enable floating underflow detection
                50   08   50  000A   140          MOVF    #1, R0                          ; R0 = initial result
           51    04 AC   50  000D   141          MOVF    base(AP), R1                    ; R1 = base
           52    08 AC   D0  0011   142          MOVL    exponent(AP), R2                ; R2 = exponent
                     05   12  0015   143          BNEQ    BEGIN                           ; Skip if not zero
                     51   53  0017   144          TSTF    R1                              ; Is base also zero?
                     20   13  0019   145          BEQL    UNDEFINED                       ; If so, error.  Otherwise, answer is
                          001B   146                                                      ; 1.0.
                     04   001B   147          RET                                     ; With result of 1.0
                          001C   148
                          001C   149 ;+
                          001C   150 ; Do the first iteration here so that we can clear the high-order exponent
                          001C   151 ; bit.  Afterwards, that bit will be zero so we don't have to worry about
                          001C   152 ; it.
                          001C   153 ;-
                          001C   154
                          001C   155 BEGIN:
                03 52  00   E5  001C   156          BBCC    #0, R2, 10$                     ; Skip initial multiply
                   50  51   50  0020   157          MOVF    R1, R0                          ; New partial result
           52   52   FF 8F   9C  0023   158 10$:     ROTL    #-1, R2, R2                     ; Get new low exponent bit and clear
                          0028   159                                                      ; high bit
                     10   13  0028   160          BEQL    DONE                            ; Is that all?  If so, we're done.
                          002A   161
                          002A   162 ;+
                          002A   163 ; Each time we get here we know that there's at least one more exponent bit
                          002A   164 ; left, so square our current power of the base.
                          002A   165 ;-
                          002A   166
                          002A   167 LOOP:
                51   51   44  002A   168          MULF2   R1, R1                          ; Square base.  This may overflow or
                          002D   169                                                      ; underflow.  In either case, the
                          002D   170                                                      ; final result would also overflow
                          002D   171                                                      ; or underflow.  Our exception
                          002D   172                                                      ; handler will catch these cases.
                  03 52   E9  002D   173          BLBC    R2, NEXT_BIT                    ; See if this exponent bit is on.
                          0030   174                                                      ; If so, multiply base by product.
                50   51   44  0030   175          MULF2   R1, R0                          ; This too may overflow or underflow,
                          0033   176                                                      ; which will be caught by the handler.
                          0033   177 NEXT_BIT:
           52   52   FF 8F   78  0033   178          ASHL    #-1, R2, R2                     ; Get next bit of exponent.
                     F0   12  0038   179          BNEQ    LOOP                            ; Keep going if not all done.
                          003A   180
                          003A   181 DONE:
                     04   003A   182          RET                                     ; Return with answer in R0.
                          003B   183
                          003B   184 ;+
                          003B   185 ; Undefined exponentation error - 0**0
                          003B   186 ;-
                          003B   187
                          003B   188 UNDEFINED:
              7E   00'8F   9A  003B   189          MOVZBL  #MTH$K_UNDEXP, -(SP)            ; Indicate undefined exponentiation.
           50   01   0F   78  003F   190          ASHL    #15, #1, R0                     ; R0 = reserved floating operand
                     15   11  0043   191          BRB     SIGNAL_ERROR
                          0045   192
                          0045   193 ;+
                          0045   194 ; EXC_HANDLER unwinds here if we got a floating overflow.
                          0045   195 ;-
```

```
                        0045   196
                        0045   197 OVERFLOW:
          7E   00'8F    9A  0045   198         MOVZBL  #MTH$K_FLOOVEMAT, -(SP) ; Floating overflow in Math Library
       50   01    0F    78  0049   199         ASHL    #15, #1, R0             ; R0 = reserved floating operand
                  0B    11  004D   200         BRB     SIGNAL_ERROR
                        004F   201
                        004F   202 ;+
                        004F   203 ; EXC_HANDLER unwinds here if we got a floating underflow.
                        004F   204 ; If our caller enabled FU, then signal FLOUNDMAT, with a result of zero.
                        004F   205 ; Otherwise quietly return zero.
                        004F   206 ;-
                        004F   207
                        004F   208 UNDERFLOW:
                  50    D4  004F   209         CLRF    R0                      ; Initial result
       E4 04 AD   06    E1  0051   210         BBC     #PSL$V_FU, SF$W_SAVE_PSW(FP), DONE      ; Just return if
                        0056   211                                             ; caller disabled underflow.
          7E   00'8F    9A  0056   212         MOVZBL  #MTH$K_FLOUNDMAT, -(SP) ; Floating underflow in Math Library
                        005A   213 ;       BRB     SIGNAL_ERROR
                        005A   214
                        005A   215 ;+
                        005A   216 ; Signal a MTH$ error whose code has already been pushed on the stack.
                        005A   217 ; R0 has either a reserved operand or a zero, depending on the exception type.
                        005A   218 ;-
                        005A   219
                        005A   220 SIGNAL_ERROR:
                  6D    D4  005A   221         CLRL    (FP)                    ; Cancel our condition handler
 00000000'GF      01    FB  005C   222         CALLS   #1, G^MTH$$SIGNAL       ; convert to 32-bit condition code
                        0063   223                                             ; and signal a MTH$ error.  Omit
                        0063   224                                             ; second argument to show that this
                        0063   225                                             ; is a CALL entry point.
                  04    0063   226         RET                             ; Return to caller with result in R0
                        0064   227
```

```
                     0064   229 ;+
                     0064   230 ; EXC_HANDLER - This condition handler gets control if we got an exception
                     0064   231 ; while performing the exponentiation.  If it is a floating overflow,
                     0064   232 ; then continue execution at OVERFLOW, to cause MTH$_FLOOVEMAT to be
                     0064   233 ; signalled.  Similarly, continue at UNDERFLOW if an underflow was seen.
                     0064   234 ; Otherwise resignal.
                     0064   235 ;-
                     0064   236
                     0064   237 EXC_HANDLER:
                0000 0064   238         .WORD   ^M<>                            ; Entry point
                     0066   239
    50   08 AC   D0 0066   240         MOVL    CHF$L_MCHARGLST(AP), R0 ; Get mechanism arguments list
         08 A0   D5 006A   241         TSTL    CHF$L_MCH_DEPTH(R0)     ; At depth zero?
            24   12 006D   242         BNEQ    RESIGNAL                ; If not, resignal
    50   04 AC   D0 006F   243         MOVL    CHF$L_SIGARGLST(AP), R0 ; Get signal arguments list
    51   04 A0   D0 0073   244         MOVL    CHF$L_SIG_NAME(R0), R1  ; Get signal name
04B4 8F     51   B1 0077   245         CMPW    R1, #SS$_FLTOVF_F       ; Overflow fault?
            1B   13 007C   246         BEQL    GOTO_OVERFLOW           ; If so, continue at OVERFLOW
048C 8F     51   B1 007E   247         CMPW    R1, #SS$_FLTOVF         ; Overflow trap?
            14   13 0083   248         BEQL    GOTO_OVERFLOW           ; If so, continue at OVERFLOW
04C4 8F     51   B1 0085   249         CMPW    R1, #SS$_FLTUND_F       ; Underflow fault?
            18   13 008A   250         BEQL    GOTO_UNDERFLOW          ; If so, continue at UNDERFLOW
049C 8F     51   B1 008C   251         CMPW    R1, #SS$_FLTUND         ; Underflow trap?
            11   13 0091   252         BEQL    GOTO_UNDERFLOW          ; If so, continue at UNDERFLOW
                     0093   253 RESIGNAL:
50   0918 8F     3C 0093   254         MOVZWL  #SS$_RESIGNAL, R0       ; Resignal exception
            04   0098   255         RET                             ; Return to VMS exception dispatcher
                     0099   256
                     0099   257 GOTO_OVERFLOW:
    51   60   D0 0099   258         MOVL    CHF$L_SIG_ARGS(R0), R1  ; Get number of signal arguments
FC A041   A6 AF   DE 009C   259         MOVAL   B^OVERFLOW, -4(R0)[R1]  ; Move address of OVERFLOW routine
                     00A2   260                                         ; to PC in signal argument list
            09   11 00A2   261         BRB     CONTINUE                ; Continue execution
                     00A4   262
                     00A4   263 GOTO_UNDERFLOW:
    51   60   D0 00A4   264         MOVL    CHF$L_SIG_ARGS(R0), R1  ; Get number of signal arguments
FC A041   A5 AF   DE 00A7   265         MOVAL   B^UNDERFLOW, -4(R0)[R1] ; Move address of UNDERFLOW routine
                     00AD   266                                         ; to PC in signal argument list
                     00AD   267 ;       BRB     CONTINUE                ; Continue execution
                     00AD   268
                     00AD   269 CONTINUE:
        50   01   3C 00AD   270         MOVZWL  #SS$_CONTINUE, R0       ; Continue at PC in signal list
             04   00B0   271         RET                             ; Return to VMS exception dispatcher
                     00B1   272
                     00B1   273
                     00B1   274         .END                            ; End of module OTS$POWRLU
```

```
                                                I 8

BASE                            = 00000004
BEGIN                             0000001C R      02
CHF$L_MCHARGLST                 = 00000008
CHF$L_MCH_DEPTH                 = 00000008
CHF$L_SIGARGLST                 = 00000004
CHF$L_SIG_ARGS                  = 00000000
CHF$L_SIG_NAME                  = 00000004
CONTINUE                          000000AD R      02
DONE                              0000003A R      02
EXC_HANDLER                       00000064 R      02
EXPONENT                        = 00000008
GOTO_OVERFLOW                     00000099 R      02
GOTO_UNDERFLOW                    000000A4 R      02
LOOP                              0000002A R      02
MTH$$SIGNAL                       ******** X      00
MTH$K_FLOOVEMAT                   ******** X      00
MTH$K_FLOUNDMAT                   ******** X      00
MTH$K_UNDEXP                      ******** X      00
NEXT_BIT                          00000033 R      02
OTS$POWRLU                        00000000 RG     02
OVERFLOW                          00000045 R      02
PSL$M_FU                        = 00000040
PSL$V_FU                        = 00000006
RESIGNAL                          00000093 R      02
SF$W_SAVE_PSW                   = 00000004
SIGNAL_ERROR                      0000005A R      02
SS$_CONTINUE                    = 00000001
SS$_FLTOVF                      = 0000048C
SS$_FLTOVF_F                    = 000004B4
SS$_FLTUND                      = 0000049C
SS$_FLTUND_F                    = 000004C4
SS$_RESIGNAL                    = 00000918
UNDEFINED                         0000003B R      02
UNDERFLOW                         0000004F R      02

                                +----------------+
                                ! 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
_OTS$CODE                       000000B1  (  177.)  02 (  2.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD  NOWRT NOVEC LONG

                                +-------------------------+
                                ! Performance indicators !
                                +-------------------------+

Phase                       Page faults     CPU Time        Elapsed Time
-----                       -----------     --------        ------------
Initialization                       33     00:00:00.08     00:00:00.75
Command processing                  130     00:00:00.52     00:00:02.28
Pass 1                              230     00:00:04.94     00:00:15.28
Symbol table sort                     0     00:00:00.78     00:00:01.84
Pass 2                               70     00:00:01.12     00:00:04.68
Symbol table output                   6     00:00:00.06     00:00:00.18
```

Psect synopsis output	5	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	477	00:00:07.53	00:00:25.03

The working set limit was 1050 pages.
26537 bytes (52 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 508 non-local and 1 local symbols.
274 source lines were read in Pass 1, producing 13 object records in Pass 2.
11 pages of virtual memory were used to define 10 macros.

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

Macro library name	Macros defined
_255DUA28:[SYSLIB]STARLET.MLB;2	7

562 GETS were required to define 7 macros.

There were no errors, warnings or information messages.

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

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