```
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          JJ          JJ
  000000    TTTTTTTTTT    SSSSSSSS  PPPPPPPP      000000    WW      WW          JJ          JJ
00      00      TT      SS          PP      PP  00      00  WW      WW          JJ          JJ
00      00      TT      SS          PP      PP  00      00  WW      WW          JJ          JJ
00      00      TT      SS          PP      PP  00      00  WW      WW          JJ          JJ
00      00      TT      SS          PP      PP  00      00  WW      WW          JJ          JJ
00      00      TT        SSSSSS    PPPPPPPP    00      00  WW      WW          JJ          JJ
00      00      TT        SSSSSS    PPPPPPPP    00      00  WW      WW          JJ          JJ
00      00      TT              SS  PP          00      00  WW  WW  WW  JJ      JJ  JJ      JJ
00      00      TT              SS  PP          00      00  WW  WW  WW  JJ      JJ  JJ      JJ
00      00      TT              SS  PP          00      00  WWWW  WWWW  JJ      JJ  JJ      JJ      ....
00      00      TT              SS  PP          00      00  WWWW  WWWW  JJ      JJ  JJ      JJ      ....
  000000        TT      SSSSSSSS    PP            000000    WW      WW    JJJJJJ      JJJJJJ        ....
  000000        TT      SSSSSSSS    PP            000000    WW      WW    JJJJJJ      JJJJJJ        ....

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  OTS$POWJJ - INTEGER*4 ** INTEGER*4 power routine
0000     2          .IDENT  /1-005/          ; File OTSPOWJJ.MAR  Edit: SBL1005
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  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: Language support library - user callable
0000    30 ;++
0000    31 ; ABSTRACT:
0000    32 ;
0000    33 ;       Integer longword base to integer longword power.
0000    34 ;       Integer overflow can occur if the result exceeds a longword.
0000    35 ;       Undefined exponentation can occur if base is 0 and power is 0 or negative.
0000    36 ;
0000    37 ;
0000    38 ;--
000C    39 ;
0000    40 ; VERSION: 0
0000    41 ;
0000    42 ; HISTORY:
0000    43 ; AUTHOR:
0000    44 ;       Thomas N. Hastings, 5-May-77: Version 0
0000    45 ;
0000    46 ; MODIFIED BY: SUSAN HUBBARD AZIBERT
0000    47 ;
0000    48 ;
0000    49 ;
```

```
0000     51          .SBTTL  HISTORY         ; Detailed Current Edit History
0000     52
0000     53
0000     54 ; Edit History for Version 01 of OTS$POWJJ
0000     55 ; version 04 - changed module name to forpowJJ
0000     56 ; version 06 - changed error routine from MTH$ERROR to MTH$$ERROR
0000     57 ; version 07 - removed W^ on call to MTH$$ERROR, save code with MOVZBL.
0000     58 ; version 08 - change MTH$$ERROR to MTH$$SIGNAL - JMT
0000     59 ; 0-11  - fix case instruction bug.  JMT 28-Feb-78
0000     60 ; 1-001 - Update version number and copyright notice.  JBS 16-NOV-78
0000     61 ; 1-002 - Change MTH_UNDEXP to MTH$K_UNDEXP.  JBS 07-DEC-78
0000     62 ; 1-003 - Add "_" to the PSECT directive.  JBS 22-DEC-78
0000     63 ; 1-004 - Declare externals.  SBL 17-May-1979
0000     64 ; 1-005 - Use general mode addressing.  SBL 30-Nov-1981
```

```
                     0000     66          .SBTTL  DECLARATIONS
                     0000     67
                     0000     68 ;
                     0000     69 ; INCLUDE FILES:
                     0000     70 ;
                     0000     71
                     0000     72 ;
                     0000     73 ; EXTERNAL SYMBOLS:
                     0000     74 ;
                     0000     75
                     0000     76          .DSABL  GBL
                     0000     77          .EXTRN  MTH$K_UNDEXP
                     0000     78          .EXTRN MTH$$SIGNAL                  ; Math error routine
                     0000     79 ;
                     0000     80 ; MACROS:
                     0000     81 ;
                     0000     82
                     0000     83 ;
                     0000     84 ; EQUATED SYMBOLS:
                     0000     85 ;
                     0000     86          base = 4                            ; base input formal - by-value
                     0000     87          exp = 8                             ; exponent intpu formal - by-value
                     0000     88
                     0000     89 ;
                     0000     90 ; OWN STORAGE:
                     0000     91 ;
                     0000     92
                     0000     93 ;
                     0000     94 ; PSECT DECLARATIONS:
                     0000     95 ;
                     0000     96
                     0000     97          .PSECT  _OTS$CODE PIC,SHR,LONG,EXE,NOWRT
                     0000     98                                              ; program section for OTS$ code
                     0000     99
```

```
0000    101          .SBTTL  OTS$POWJJ - longword to power longword giving longword result
0000    102
0000    103 ;++
0000    104 ; FUNCTIONAL DESCRIPTION:
0000    105 ;
0000    106 ;       Signed longword result = signed longword base ** signed longword exponent
0000    107 ;       The signed longword result is given by:
0000    108 ;
0000    109 ;       base    exponent        result
0000    110 ;
0000    111 ;       any     > 0             product (base * 2**i) where i is each
0000    112 ;                               non-zero bit position in exponent
0000    113 ;
0000    114 ;       > 0     = 0             1
0000    115 ;       = 0     = 0             Undefined exponentation
0000    116 ;       < 0     = 0             1
0000    117 ;
0000    118 ;       > 1     < 0             0
0000    119 ;       = 1     < 0             1
0000    120 ;       = 0     < 0             Undefined exponentation
0000    121 ;       = -1    < 0 and even    1
0000    122 ;       = -1    < 0 and odd     -1
0000    123 ;       < -1    < 0             1
0000    124 ;
0000    125 ;       Integer overflow can occur.
0000    126 ;       Undefined exponentiation occurs if base is 0 and
0000    127 ;       exponent is 0 or negative.
0000    128 ;
0000    129 ; CALLING SEQUENCE:
0000    130 ;
0000    131 ;       Power.ww.v = OTSS$POWJJ (base.rw.v,  exponent.rw.v)
0000    132 ;
0000    133 ; INPUT PARAMETERS:
0000    134 ;       NONE
0000    135 ;
0000    136 ; IMPLICIT INPUTS:
0000    137 ;       NONE
0000    138 ;
0000    139 ; OUTPUT PARAMETERS:
0000    140 ;       NONE
0000    141 ;
0000    142 ; IMPLICIT OUTPUTS:
0000    143 ;       NONE
0000    144 ;
0000    145 ; FUNCTION VALUE:
0000    146 ;
0000    147 ;       longword integer base ** exponent
0000    148 ;
0000    149 ; SIDE EFFECTS:
0000    150 ;
0000    151 ;       SIGNALs SS$_ARITH with integer overflow hardware code if
0000    152 ;       integer overflow.
0000    153 ;       SIGNALs MTH$_UNDEXP (82 = ' JNDEFINED EXPONENTATION') if
0000    154 ;       base is 0 and exponent is 0 or negative.
0000    155 ;
0000    156 ;--
0000    157
```

```
                                0000   158
                                0000   159
                           4004 0000   160          .ENTRY OTS$POWJJ, ^M<IV, R2>    ; enable integer overflow
                50   01     D0  0002   161          MOVL    #1, R0                  ; R0 = initial result
              52    08 AC    D0  0005   162          MOVL    exp(AP), R2             ; R2 = exponent
                       1D    15  0009   163          BLEQ    EXPLEQ                  ; branch if exponent =< 0
                             000B   164
                             000B   165 ;+
                             000B   166 ; Exponent > 0.
                             000B   167 ; Scan each exponent bit from right, squaring base each time thru loop.
                             000B   168 ; For each 1-bit in exponent, multiply current base into partial result.
                             000B   169 ;-
                             000B   170
              51    04 AC    D0  000B   171          MOVL    base(AP), R1            ; R1 = base
                    0B 52    E8  000F   172          BLBS    R2, PARTIAL             ; branch if exponent is odd
        52    52    FF 8F    78  0012   173 SQUAR:  ASHL    #-1, R2, R2             ; R2 = exponent/2
              51    51       C4  0017   174 SQUAR1: MULL    R1, R1                  ; R1 = current power of base
                             001A   175                                             ; integer overflow will trap
                             001A   176                                             ; and SIGNAL SS$_ARITH
                 F5 52       E9  001A   177          BLBC    R2, SQUAR               ; loop if next bit in exponent is 0
                             001D   178 PARTIAL:                                    ; next bit in exponent is a 1
              50    51       C4  001D   179          MULL    R1, R0                  ; R0 = new partial result
                             0020   180                                             ; integer overflow will trap
                             0020   181                                             ; and SIGNAL SS$_ARITH
        52    52    FF 8F    78  0020   182          ASHL    #-1, R2, R2             ; R2 = exponent/2
                    F0       12  0025   183          BNEQ    SQUAR1                  ; loop if more exponent bits are 1
                             04  0027   184          RET                             ; return, R0<15:0> = base ** exp
                             0028   185                                             ; R0<31:16> = 0
                             0028   186
                             0028   187 ;+
                             0028   188 ; Exponent is =< 0.
                             0028   189 ;-
                             0028   190
                       08    19  0028   191 EXPLEQ: BLSS    EXPLSS                  ; branch if exponent < 0
                             002A   192
                             002A   193 ;+
                             002A   194 ; Exponent is = 0.
                             002A   195 ; Undefined exponentation if base = 0 too, else return 1
                             002A   196 ;-
                             002A   197
      51    04 AC    D0  002A   198          MOVL    base(AP), R1            ; R1 = base
                       1B  13  002E   199          BEQL    UNDEFINED               ; undefined if base = 0 too
                       18  11  0030   200          BRB     POWJJX                  ; return with result = 1
                             0032   201                                             ; since base ** 0 = 1
                             0032   202
                             0032   203 ;+
                             0032   204 ; exponent =< 0.
                             0032   205 ; Result is given by the following table:
                             0032   206 ;
                             0032   207 ;       Base    Result
                             0032   208 ;       <-1     0
                             0032   209 ;       -1      1 or -1 depending on exponent being even or odd
                             0032   210 ;       0       Undefined exponentation
                             0032   211 ;       1       1
                             0032   212 ;       >1      0
                             0032   213 ;-
                             0032   214
```

```
                     0032   215 EXPLSS:
02  FFFFFFFF 8F  04 AC  CF  0032   216         CASEL   base(AP), #-1, #2          ; Case on value of base
                 0009' 003B   217 10$:    .WORD   MINUS1-10$                    ; [-1]: return R0 = -1 or 1 depending
                       003D   218                                                ; on exponent being odd or even
                 0010' 003D   219         .WORD   UNDEFINED-10$                 ; [0]: Undefined exponentation
                 000F' 003F   220         .WORD   POWJJX-10$                    ; [+1]: return R0 = 1
                    50  D4    0041   221         CLRL    R0                    ; [< -1 or > +1]: return R0 = 0
                        04    0043   222         RET
                              0044   223
                   03 52  E9  0044   224 MINUS1: BLBC    R2, POWJJX            ; if exponent is even, return R0 = 1
                50    01  CE  0047   225         MNEGL   #1, R0                ; else return R0 = -1
                        04    004A   226 POWJJX: RET                           ; return
                              004B   227
                              004B   228 ;+
                              004B   229 ; Undefined exponentation error - 0**0 or 0**(-n)
                              004B   230 ;-
                              004B   231
                              004B   232 UNDEFINED:
                    50  D4    004B   233         CLRL    R0                    ; if error, return result = 0
              7E    00'8F  9A  004D   234         MOVZBL  #MTH$K_UNDEXP, -(SP)  ; FORTRAN error #
     00000000'GF    01  FB  0051   235         CALLS   #1, G^MTH$$SIGNAL     ; convert to 32-bit condition code
                              0058   236                                       ; and SIGNAL MTH$_UNDEXP
                              0058   237                                       ; Note: 2nd arg not needed since no
                              0058   238                                       ; JSB OTS$POWRJ is possible.
                        04    0058   239         RET
                              0059   240
                              0059   241
                              0059   242         .END
```

```
BASE            =  00000004
EXP             =  00000008
EXPLEQ             00000028 R      01
EXPLSS             00000032 R      01
MINUS1             00000044 R      01
MTH$$SIGNAL        ********   X    00
MTH$K_UNDEXP       ********   X    00
OTS$POWJJ          00000000 RG     01
PARTIAL            0000001D R      01
POWJJX             0000004A R      01
SQUAR              00000012 R      01
SQUAR1             00000017 R      01
UNDEFINED          0000004B R      01
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
_OTS$CODE	00000059 (89.)	01 (1.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.10	00:00:01.48
Command processing	130	00:00:00.48	00:00:02.34
Pass 1	71	00:00:00.58	00:00:02.34
Symbol table sort	0	00:00:00.00	00:00:00.00
Pass 2	54	00:00:00.52	00:00:02.18
Symbol table output	2	00:00:00.02	00:00:00.21
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	293	00:00:01.72	00:00:08.57

The working set limit was 750 pages.
2737 bytes (6 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 13 non-local and 1 local symbols.
242 source lines were read in Pass 1, producing 11 object records in Pass 2.
0 pages of virtual memory were used to define 0 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYSLIB]STARLET.MLB;2	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

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

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