```
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    IIIIII      IIIIII
  000000    TTTTTTTTTT    SSSSSSSS  PPPPPPPP      000000    WW      WW    IIIIII      IIIIII
00      00      TT      SS          PP      PP  00      00  WW      WW      II          II
00      00      TT      SS          PP      PP  00      00  WW      WW      II          II
00      00      TT      SS          PP      PP  00      00  WW      WW      II          II
00      00      TT      SS          PP      PP  00      00  WW      WW      II          II
00      00      TT        SSSSSS    PPPPPPPP    00      00  WW      WW      II          II
00      00      TT        SSSSSS    PPPPPPPP    00      00  WW      WW      II          II
00      00      TT              SS  PP          00      00  WW  WW  WW      II          II
00      00      TT              SS  PP          00      00  WW  WW  WW      II          II
00      00      TT              SS  PP          00      00  WWWW  WWWW      II          II      ....
00      00      TT              SS  PP          00      00  WWWW  WWWW      II          II      ....
  000000        TT      SSSSSSSS    PP            000000    WW      WW    IIIIII      IIIIII    ....
  000000        TT      SSSSSSSS    PP            000000    WW      WW    IIIIII      IIIIII    ....

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$POWII - INTEGER*2 ** INTEGER*2 power routine
0000     2          .IDENT  /1-006/         ; File OTSPOWII.MAR  Edit: SBL1006
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 word base to integer word power.
0000    34 ;       Integer overflow can occur if the result exceeds a word.
0000    35 ;       Undefined exponentation can occur if base is 0 and power is 0 or negative.
0000    36 ;
0000    37 ;
0000    38 ;--
0000    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 0 of OTS$POWII
0000     55 ; version 04 - changed module name to forpowII
0000     56 ; version 05 - changed error handler from MTH$ERROR to MTH$$ERROR
0000     57 ; version 07 - changed error handler to MTH$$SIGNAL
0000     58 ; 0-10  - fixed bug in case instruction.  JMT 28-Feb-78
0000     59 ; 1-001 Update version number and copyright notice.  JBS 16-NOV-78
0000     60 ; 1-002 - Change MTH__UNDEXP to MTH$K_UNDEXP.  JBS 07-DEC-78
0000     61 ; 1-003 - Add "_" to the PSECT directive.  JBS 22-DEC-78
0000     62 ; 1-004 - Use 32-bit addresses to refer to externals.  JBS 28-JAN-1979
0000     63 ; 1-005 - Declare externals.  SBL 17-May-1979
0000     64 ; 1-006 - 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$POWII - Word to power word giving word result
0000    102
0000    103 ;++
0000    104 ; FUNCTIONAL DESCRIPTION:
0000    105 ;
0000    106 ;       Signed word result = signed word base ** signed word exponent
0000    107 ;       The signed word result is given by:
0000    108 ;
0000    109 ;       base    exponent        result
0000    110 ;
0000    111 ;       any     > 0             product (base * 2**i) where i is each
0000    112 ;                               non-zero bit position in exponent
0000    113 ;
0000    114 ;       > 0     = 0             1
0000    115 ;       = 0     = 0             Undefined exponentation
0000    116 ;       < 0     = 0             1
0000    117 ;
0000    118 ;       > 1     < 0             0
0000    119 ;       = 1     < 0             1
0000    120 ;       = 0     < 0             Undefined exponentation
0000    121 ;       = -1    < 0 and even    1
0000    122 ;       = -1    < 0 and odd     -1
0000    123 ;       < -1    < 0             1
0000    124 ;
0000    125 ;       Integer overflow can occur.
0000    126 ;       Undefined exponentiation occurs if base is 0 and
0000    127 ;       exponent is 0 or negative.
0000    128 ;
0000    129 ; CALLING SEQUENCE:
0000    130 ;
0000    131 ;       Power.ww.v = OTS$POWII (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 ;       Word 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 = ' UNDEFINED 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$POWII, ^M<IV, R2>    ; enable integer overflow
                50    01   D0   0002   161          MOVL    #1, R0                  ; R0 = initial result
            52  08 AC   32   0005   162          CVTWL   exp(AP), R2             ; R2 = exponent
                      1D   15   0009   163          BLEQ    EXPLEQ                  ; branch if exponent =< 0
                                000B   164
                                000B   165 ;+
                                000B   166 ; Exponent > 0.
                                000B   167 ; Scan each exponent bit from right, squaring base each time thru loop.
                                000B   168 ; For each 1-bit in exponent, multiply current base into partial result.
                                000B   169 ;-
                                000B   170
            51  04 AC   B0   000B   171          MOVW    base(AP), R1            ; R1 = base
                   0B 52   E8   000F   172          BLBS    R2, PARTIAL             ; branch if exponent is odd
        52  52  FF 8F   78   0012   173 SQUAR:  ASHL    #-1, R2, R2             ; R2 = exponent/2
              51   51   A4   0017   174 SQUAR1: MULW    R1, R1                  ; R1 = current power of base
                                001A   175                                          ; integer overflow will trap
                                001A   176                                          ; and SIGNAL SS$_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   A4   001D   179          MULW    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   B0   002A   198          MOVW    base(AP), R1            ; R1 = base
                19   13   002E   199          BEQL    UNDEFINED               ; undefined if base = 0 too
                16   11   0030   200          BRB     POWIIX                  ; 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   FFFF 8F   04 AC   AF  0032   216         CASEW   base(AP), #-1, #2        ; Case on value of base
                  0009' 0039   217 10$:      .WORD   MINUS1-10$                ; [-1]: return R0 = -1 or 1 depending
                        003B   218                                             ; on exponent being odd or even
                  0010' 003B   219         .WORD   UNDEFINED-10$              ; [0]: Undefined exponentation
                  000F' 003D   220         .WORD   POWIIX-10$                 ; [+1]: return R0 = 1
                     50   D4  003F   221         CLRL    R0                     ; [< -1 or > +1]: return R0 = 0
                     04   0041   222         RET
                          0042   223
                          0042   224 MINUS1:
                  03 52   E9  0042   225         BLBC    R2, POWIIX             ; if exponent is even, return R0 = 1
                50   01   CE  0045   226         MNEGL   #1, R0                 ; else return R0 = -1
                     04   0048   227 POWIIX: RET                                ; return
                          0049   228
                          0049   229 ;+
                          0049   230 ; Undefined exponentation error - 0**0 or 0**(-n)
                          0049   231 ;-
                          0049   232
                          0049   233 UNDEFINED:
                     50   D4  0049   234         CLRL    R0                     ; return result = 0 if error
             7E   00'8F   9A  004B   235         MOVZBL  #MTH$K_UNDEXP, -(SP)   ; FORTRAN error #
00000000'GF   01   FB  004F   236         CALLS   #1, G^MTH$$SIGNAL              ; convert to 32-bit condition code
                          0056   237                                             ; and SIGNAL FOR$_UNDEXP
                     04   0056   238         RET
                          0057   239
                          0057   240         .END
```

```
BASE            = 00000004
EXP             = 00000008
EXPLEQ            00000028 R      01
EXPLSS            00000032 R      01
MINUS1            00000042 R      01
MTH$$SIGNAL       ********   X    00
MTH$K_UNDEXP      ********   X    00
OTS$POWII         00000000 RG     01
PARTIAL           0000001D R      01
POWIIX            00000048 R      01
SQUAR             00000012 R      01
SQUAR1            00000017 R      01
UNDEFINED         00000049 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	00000057 (87.)	01 (1.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.06	00:00:01.01
Command processing	120	00:00:00.55	00:00:03.65
Pass 1	68	00:00:00.55	00:00:02.23
Symbol table sort	0	00:00:00.01	00:00:00.01
Pass 2	56	00:00:00.53	00:00:02.51
Symbol table output	2	00:00:00.02	00:00:00.02
Psect synopsis output	3	00:00:00.02	00:00:00.11
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	282	00:00:01.76	00:00:09.60

The working set limit was 900 pages.
2719 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.
240 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$:OTSPOWII/OBJ=OBJ$:OTSPOWII MSRC$:OTSPOWII/UPDATE=(ENH$:OTSPOWII)
```

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