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

```
0000     1          .TITLE  OTS$POWCJ - COMPLEX ** INTEGER*4 power routine
0000     2          .IDENT  /1-004/         ; File OTSPOWCJ.MAR  Edit: SBL1004
0000     3  ;******************************************************************************
0000     4  ;*                                                                            *
0000     5  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                   *
0000     6  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
0000     7  ;*  ALL RIGHTS RESERVED.                                                      *
0000     8  ;*                                                                            *
0000     9  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED      *
0000    10  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE     *
0000    11  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER      *
0000    12  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY      *
0000    13  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY      *
0000    14  ;*  TRANSFERRED.                                                               *
0000    15  ;*                                                                            *
0000    16  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE      *
0000    17  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
0000    18  ;*  CORPORATION.                                                               *
0000    19  ;*                                                                            *
0000    20  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS      *
0000    21  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                    *
0000    22  ;*                                                                            *
0000    23  ;*                                                                            *
0000    24  ;******************************************************************************
0000    25  ;
0000    26  ;
0000    27  ; FACILITY: Language support library - user callable
0000    28  ;++
0000    29  ; ABSTRACT:
0000    30  ;
0000    31  ;       COMPLEX base to INTEGER*4 power.
0000    32  ;       Floating overflow can occur.
0000    33  ;       Undefined exponentiation can occur if
0000    34  ;       base = (0.,0.) and exp <=0
0000    35  ;
0000    36  ;--
0000    37  ;
0000    38  ; VERSION: 0
0000    39  ;
0000    40  ; HISTORY:
0000    41  ; AUTHOR:
0000    42  ;       Jonathan M. Taylor, 29-jun-77: Version 0
0000    43  ;
0000    44  ; Edit history for version 0
0000    45  ;
```

```
0000     47          .SBTTL  HISTORY         ; Detailed Current Edit History
0000     48
0000     49
0000     50 ; Edit History for Version 0 of OTS$POWCJ
0000     51 ; 0-2 change MTH$$ERROR to MTH$$SIGNAL JMT 5-OCT-77
0000     52 ; 0-6   - Change FOR$FLAG_JACKET to MTH$FLAG_JACKET.  TNH 17-July-78
0000     53 ; 0-7   - Fix bug giving divide fault, or wrong results for
0000     54 ;         some negative powers. Also clean up comments.
0000     55 ;         SPR 20364  SBL 27-Oct-78
0000     56 ; 1-001 - Change version number to 1 and MTH__UNDEXP
0000     57 ;         to MTH$K_UNDEXP.  JBS 07-DEC-78
0000     58 ; 1-002 - Add "_" to the PSECT directive.  JBS 22-DEC-78
0000     59 ; 1-003 - Declare externals.  SBL 17-May-1979
0000     60 ; 1-004 - Use general mode addressing.  SBL 30-Nov-1981
```

```
                     0000     62          .SBTTL  DECLARATIONS
                     0000     63
                     0000     64 ;
                     0000     65 ; INCLUDE FILES:
                     0000     66 ;
                     0000     67
                     0000     68 ; EXTERNAL SYMBOLS:
                     0000     69 ;
                     0000     70
                     0000     71          .DSABL  GBL
                     0000     72          .EXTRN  MTH$$SIGNAL             ; Math error routine
                     0000     73          .EXTRN  OTS$DIVC                ; COMPLEX division routine
                     0000     74          .EXTRN  MTH$K_UNDEXP
                     0000     75
                     0000     76 ;
                     0000     77 ; MACROS:
                     0000     78 ;
                     0000     79
                     0000     80 ;
                     0000     81 ; EQUATED SYMBOLS:
                     0000     82 ;
                00000004     83          base = 4                        ; base input formal - by-value
                0000000C     84          exp = 12                        ; exponent intpu formal - by-value
                     0000     85
                     0000     86 ;
                     0000     87 ; OWN STORAGE:
                     0000     88 ;
                     0000     89
                     0000     90 ;
                     0000     91 ; PSECT DECLARATIONS:
                     0000     92 ;
                     0000     93
                00000000     94          .PSECT  _OTS$CODE PIC,SHR,LONG,EXE,NOWRT
                     0000     95                                          ; program section for OTS$ code
                     0000     96
```

```
0000     98          .SBTTL  OTS$POWCJ - COMPLEX*8 ** INTEGER*4
0000     99
0000    100 ;**-
0000    101 ;++
0000    102 ; FUNCTIONAL DESCRIPTION:
0000    103 ;
0000    104 ;       COMPLEX result = COMPLEX base ** INTEGER*4 exponent
0000    105 ;       The COMPLEX result is given by:
0000    106 ;
0000    107 ;       base            exponent        result
0000    108 ;
0000    109 ;       any               >0            PRODUCT (base * 2**i) where
0000    110 ;                                       i is each non-zero bit in
0000    111 ;                                       exponent.
0000    112 ;       (0., 0.)          <=0           Undefined exponentiation.
0000    113 ;
0000    114 ;       not (0., 0.)      <0            PRODUCT (base * 2**i) where
0000    115 ;                                       i is each non-zero bit in
0000    116 ;                                       !exponent!.
0000    117 ;
0000    118 ;       not (0., 0.)      =0            (1.0, 0.0)
0000    119 ;
0000    120 ;
0000    121 ;       Floating overflow can occur.
0000    122 ;       Undefined exponentiation occurs if base is 0 and
0000    123 ;       exponent is 0 or negative.
0000    124 ;
0000    125 ; CALLING SEQUENCE:
0000    126 ;
0000    127 ;       Power.wfc.v = OTS$POWCJ (base.rfc.v,  exponent.rl.v)
0000    128 ;
0000    129 ; INPUT PARAMETERS:
0000    130 ;       The base input parameter is standard FORTRAN COMPLEX.
0000    131 ;       The exponent input parameter is a signed longword integer.
0000    132 ;       Both input parameters are CALL BY VALUE.
0000    133 ;
0000    134 ; IMPLICIT INPUTS:
0000    135 ;       NONE
0000    136 ;
0000    137 ; OUTPUT PARAMETERS:
0000    138 ;       NONE
0000    139 ;
0000    140 ; IMPLICIT OUTPUTS:
0000    141 ;       NONE
0000    142 ;
0000    143 ; FUNCTION VALUE:
0000    144 ;
0000    145 ;       COMPLEX base ** INTEGER*4 exponent
0000    146 ;
0000    147 ; SIDE EFFECTS:
0000    148 ;
0000    149 ;       SIGNALs SS$_ARITH with floating overflow hardware code if
0000    150 ;       floating overflow.
0000    151 ;       SIGNALs MTH$_UNDEXP (82 = ' UNDEFINED EXPONENTATION') if
0000    152 ;       base is 0 and exponent is 0 or negative.
0000    153 ;
0000    154 ;--
```

```
                                007C  0000   156          .ENTRY  OTS$POWCJ, ^M<R2,R3,R4,R5,R6>
                                      0002   157                                               ; disable integer overflow
                          52    04 AC   7D   0002   158          MOVQ    base(AP), R2         ; R2/R3 gets COMPLEX base
                          54    0C AC   D0   0006   159          MOVL    exp(AP), R4          ; R4 = longword exponent
                                   03   18   000A   160          BGEQ    1$                   ; R4 = ! exponent !
                             54    54   CE   000C   161          MNEGL   R4, R4
                          0C 54    00   E5   000F   162 1$:      BBCC    #0, R4, EVEN         ; branch if even and clear low bit
                             50    52   7D   0013   163          MOVQ    R2, R0               ; R0/R1 = initial result
                    54    54    FF 8F   9C   0016   164          ROTL    #-1, R4, R4          ; R4 = unsigned_exponent / 2
                                   58   13   001B   165          BEQL    DONE                 ; done if exponent was 1
                                   27   11   001D   166          BRB     SQUAR1               ; else use rest of exponent
                                        001F   167
                                        001F   168 EVEN:
                             50    08   70   001F   169          MOVD    #1, R0               ; R0/R1 = initial result
                    54    54    FF 8F   9C   0022   170          ROTL    #-1, R4, R4          ; R4 = unsigned_exponent / 2
                                   1D   12   0027   171          BNEQ    SQUAR1               ; branch if exponent not 0
                                   52   53   0029   172          TSTF    R2                   ; exponent was 0, test RP(base)
                                   48   12   002B   173          BNEQ    DONE                 ; done if non-0, answer is 1.0
                                   53   53   002D   174          TSTF    R3                   ; IP(base) better not be zero
                                   44   12   002F   175          BNEQ    DONE                 ; it isn't return 1.0
                                        0031   176
                                        0031   177 UNDEFINED:
                       50    01    0F   79   0031   178          ASHQ    #15, #1, R0          ; return R0 = reserved operand
                          7E    00'8F   9A   0035   179          MOVZBL  #MTH$K_UNDEXP, -(SP) ; FORTRAN error number
                00000000'GF    01   FB   0039   180          CALLS   #1, G^MTH$$SIGNAL    ; convert to 32-bit condition code
                                        0040   181                                               ; and SIGNAL MTH$_UNDEXP
                                   04   0040   182          RET
                                        0041   183
                                        0041   184 SQUAR:
                    54    54    FF 8F   78   0041   185          ASHL    #-1, R4, R4          ; R4 = |reduced exponent| / 2
                                        0046   186 ;
                                        0046   187 ; R2/R3 = square current base
                                        0046   188 ;
                                        0046   189 SQUAR1:
                       55    53    52   45   0046   190          MULF3   R2, R3, R5           ; R5 = tmp = RP(base)*IP(base)
                             52    52   44   004A   191          MULF    R2, R2               ; R2 = RP(base)**2
                             53    53   44   004D   192          MULF    R3, R3               ; R3 = IP(base)**2
                             52    53   42   0050   193          SUBF    R3, R2               ; R2 = RP(base)**2 - IP(base)**2
                       53    55    55   41   0053   194          ADDF3   R5, R5, R3           ; R3 = 2*RP(base)*IP(base)
                             E7 54    E9   0057   195          BLBC    R4, SQUAR            ; branch if next exponent bit is 0
                                        005A   196 ;
                                        005A   197 ; R0/R1 = partial result * current power of base
                                        005A   198 ;
                       55    53    50   45   005A   199          MULF3   R0, R3, R5           ; R5 = tmp = RP(part) * IP(base)
                             50    52   44   005E   200          MULF    R2, R0               ; R0 = RP(part) * RP(base)
                       56    53    51   45   0061   201          MULF3   R1, R3, R6           ; R6 = tmp = IP(part) * IP(base)
                             50    56   42   0065   202          SUBF    R6, R0               ; R0=RP(part)*RP(base)-IP(part)*IP(base)
                             51    52   44   0068   203          MULF    R2, R1               ; R1 = IP(part) * RP(base)
                             51    55   40   006B   204          ADDF    R5, R1               ; R1=IP(part)*RP(base)+RP(part)*IP(base)
                    54    54    FF 8F   78   006E   205          ASHL    #-1, R4, R4          ; R4 = |reduced exponent| / 2
                                   D1   12   0073   206          BNEQ    SQUAR1               ; loop if more exponent bits left
                                        0075   207 DONE:
                             0C AC    D5   0075   208          TSTL    exp(AP)              ; test exponent sign
                                   15   18   0078   209          BGEQ    POWCJ                ; done if positive
                                   50   53   007A   210          TSTF    R0                   ; test RP(result)
                                   04   12   007C   211          BNEQ    RECIP                ; if non-0, OK to take reciprocal
                                   51   53   007E   212          TSTF    R1                   ; RP(result) was 0, test IP(result)     ;***-
```

```
                     AF   13  0080   213          BEQL    UNDEFINED               ; undefined (0.0+0.0i) ** -n
                                 0082   214 RECIP:
                   7E    50  7D  0082   215          MOVQ    R0, -(SP)               ; second arg pair is divisor
                   7E    08  70  0085   216          MOVD    S^#1, -(SP)             ; push (1.,0.) on stack
          00000000'GF    04  FB  0088   217          CALLS   #4, G^OTS$DIVC          ; R0/R1 = reciprocal
                                 008F   218 POWCJ:                                                          ;***-
                         04      008F   219          RET                             ; result in R0/R1
                                 0090   220
                                 0090   221          .END
```

```
BASE            = 00000004
DONE              00000075 R      01
EVEN              0000001F R      01
EXP             = 0000000C
MTH$$SIGNAL       ********   X    00
MTH$K_UNDEXP      ********   X    00
OTS$DIVC          ********   X    00
OTS$POWCJ         00000000 RG     01
POWCJ             0000008F R      01
RECIP             00000082 R      01
SQUAR             00000041 R      01
SQUAR1            00000046 R      01
UNDEFINED         00000031 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	00000090 (144.)	01 (1.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.09	00:00:00.89
Command processing	122	00:00:00.45	00:00:06.31
Pass 1	75	00:00:00.69	00:00:04.26
Symbol table sort	0	00:00:00.01	00:00:00.10
Pass 2	54	00:00:00.46	00:00:01.69
Symbol table output	2	00:00:00.02	00:00:00.02
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	289	00:00:01.76	00:00:13.35

The working set limit was 900 pages.
3043 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.
221 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$:OTSPOWCJ/OBJ=OBJ$:OTSPOWCJ MSRC$:OTSPOWCJ/UPDATE=(ENH$:OTSPOWCJ)
```

OTSMULCD
LIS

OTSPOWCGC
LIS

OTSDIVC
LIS

OTSPOWDD
LIS

OTSPOWCC
LIS

OTSPOWGG
LIS

OTSPOWGJ
LIS

OTSPOWCDJ
LIS

MTHTAN
LIS

MTHVECTOR
LIS

OTSDIVCG
LIS

OTSPOWCJ
LIS

OTSPOWDLJ
LIS

MTHTANH
LIS

OTSMULCG
LIS

OTSPOWCGJ
LIS

OTSPOWDJ
LIS

OTSDIVCD
LIS

OTSPOWCDC
LIS