```
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  DDDDDDDD    LL          UU      UU
  000000    TTTTTTTTTT    SSSSSSSS  PPPPPPPP      000000    WW      WW  DDDDDDDD    LL          UU      UU
00      00      TT      SS        PP      PP  00      00  WW      WW  DD      DD  LL          UU      UU
00      00      TT      SS        PP      PP  00      00  WW      WW  DD      DD  LL          UU      UU
00      00      TT      SS        PP      PP  00      00  WW      WW  DD      DD  LL          UU      UU
00      00      TT      SS        PP      PP  00      00  WW      WW  DD      DD  LL          UU      UU
00      00      TT        SSSSSS  PPPPPPPP    00      00  WW      WW  DD      DD  LL          UU      UU
00      00      TT        SSSSSS  PPPPPPPP    00      00  WW      WW  DD      DD  LL          UU      UU
00      00      TT              SS  PP        00      00  WW  WW  WW  DD      DD  LL          UU      UU
00      00      TT              SS  PP        00      00  WW  WW  WW  DD      DD  LL          UU      UU
00      00      TT              SS  PP        00      00  WWWW  WWWW  DD      DD  LL          UU      UU      ....
00      00      TT              SS  PP        00      00  WWWW  WWWW  DD      DD  LL          UU      UU      ....
  000000        TT      SSSSSSSS  PP          000000    WW      WW  DDDDDDDD    LLLLLLLLLL  UUUUUUUUUU      ....
  000000        TT      SSSSSSSS  PP          000000    WW      WW  DDDDDDDD    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$POWDLU - D_floating ** unsigned integer power routine

```
0000     1          .TITLE  OTS$POWDLU - D_floating ** unsigned integer power routine
0000     2          .IDENT  /1-001/                 ; File: OTSPOWDLU.MAR Edit: SBL1001
0000     3
0000     4 ;
000C     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 ;++
0000    30 ; FACILITY: Language support procedures, Mathematics Division
0000    31 ;
0000    32 ; ABSTRACT:
0000    33 ;
0000    34 ;       This module contains OTS$POWDLU, a procedure which raises a
0000    35 ;       D_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 ;--
```

```
0000     46          .SBTTL  DECLARATIONS
0000     47 ;
0000     48 ; LIBRARY MACRO CALLS:
0000     49 ;
0000     50          $SSDEF                  ; System error codes
0000     51          $CHFDEF                 ; Define condition handler symbols.
0000     52          $SFDEF                  ; Define stack frame symbols.
0000     53          $PSLDEF                 ; Define program status longword
0000     54                                  ; symbols.
0000     55 ;
0000     56 ; EXTERNAL DECLARATIONS:
0000     57 ;
0000     58           DSABL  GBL             ; Force all external symbols to be declared
0000     59          .EXTRN  MTH$K_UNDEXP    ; Undefined exponentiation
0000     60          .EXTRN  MTH$K_FLOOVEMAT ; Floating overflow in Math Library
0000     61          .EXTRN  MTH$K_FLOUNDMAT ; Floating underflow in Math Library
0000     62          .EXTRN  MTH$$SIGNAL     ; Math error routine
0000     63 ;
0000     64 ; MACROS:
0000     65 ;
0000     66 ;        NONE
0000     67 ;
0000     68 ; EQUATED SYMBOLS:
0000     69 ;
0000     70 ;        NONE
0000     71 ;
0000     72 ; OWN STORAGE:
0000     73 ;
0000     74 ;        NONE
0000     75 ;
0000     76 ; PSECT DECLARATIONS:
0000     77 ;
00000000 78          .PSECT _OTS$CODE PIC, USR, CON, REL, LCL, SHR, -
0000     79                      EXE, RD, NOWRT, LONG
0000     80
```

```
                     0000     82          .SBTTL  OTS$POWDLU - D_floating ** unsigned integer power routine
                     0000     83 ;++
                     0000     84 ; FUNCTIONAL DESCRIPTION:
                     0000     85 ;
                     0000     86 ;       Raise a D_floating value to an unsigned integer power giving
                     0000     87 ;       a D_floating result.
                     0000     88 ;
                     0000     89 ;       The result is given by:
                     0000     90 ;
                     0000     91 ;       base    exponent        result
                     0000     92 ;
                     0000     93 ;       any     > 0             product (base * 2**i) where i is each
                     0000     94 ;                               non-zero bit position in exponent
                     0000     95 ;
                     0000     96 ;       > 0     = 0             1.0
                     0000     97 ;       = 0     = 0             Undefined exponentation
                     0000     98 ;       < 0     = 0             1.0
                     0000     99 ;
                     0000    100 ;
                     0000    101 ; CALLING SEQUENCE:
                     0000    102 ;
                     0000    103 ;       result.wd.v = OTS$POWDLU (base.rd.v, exponent.rlu.v)
                     0000    104 ;
                     0000    105 ; FORMAL PARAMETERS:
                     0000    106 ;
                     0000    107
00000004             0000    108         base = 4        ; D_floating base.  Note that this is passed by
                     0000    109                         ; immediate value in two argument list positions,
                     0000    110                         ; which is a violation of the calling standard.
                     0000    111                         ; This is allowed for language support procedures.
                     0000    112
0000000C             0000    113         exponent = 12   ; Unsigned longword integer exponent
                     0000    114
                     0000    115 ;
                     0000    116 ; IMPLICIT INPUTS:
                     0000    117 ;
                     0000    118 ;       NONE
                     0000    119 ;
                     0000    120 ; IMPLICIT OUTPUTS:
                     0000    121 ;
                     0000    122 ;       NONE
                     0000    123 ;
                     0000    124 ; ROUTINE VALUE:
                     0000    125 ;
                     0000    126 ;       The base raised to the exponent's power.
                     0000    127 ;
                     0000    128 ; SIDE EFFECTS:
                     0000    129 ;
                     0000    130 ;       May signal:
                     0000    131 ;               MTH$_FLOOVEMAT - floating overflow in Math Library
                     0000    132 ;               MTH$_FLOUNDMAT - floating underflow in Math Library, only
                     0000    133 ;                                if the caller has FU enabled.
                     0000    134 ;               MTH$_UNDEXP    - undefined exponentiation, if both base and
                     0000    135 ;                                exponent are zero.
                     0000    136 ;
                     0000    137 ;--
                     0000    138
```

```
                     001C  0000   139          .ENTRY OTS$POWDLU, ^M<R2,R3,R4> ; Entry point
                           0002   140
          6D   64'AF   9E  0002   141          MOVAB   B^EXC_HANDLER, (FP)     ; Translate exceptions to MTH$ errors
             0040 8F   B8  0006   142          BISPSW  #PSL$M_FU               ; Enable floating underflow detection
             50   08   70  000A   143          MOVD    #1, R0                  ; R0/R1 = initial result
          52   04 AC   70  000D   144          MOVD    base(AP), R2            ; R2/R3 = base
          54   0C AC   D0  0011   145          MOVL    exponent(AP), R4        ; R4 = exponent
                   05   12  0015   146          BNEQ    BEGIN                   ; Skip if not zero
                   52   73  0017   147          TSTD    R2                      ; Is base also zero?
                   20   13  0019   148          BEQL    UNDEFINED               ; If so, error.  Otherwise, answer is
                        001B   149                                              ; 1.0.
                   04   001B   150          RET                             ; With result of 1.0
                        001C   151
                        001C   152 ;+
                        001C   153 ; Do the first iteration here so that we can clear the high-order exponent
                        001C   154 ; bit.  Afterwards, that bit will be zero so we don't have to worry about
                        001C   155 ; it.
                        001C   156 ;-
                        001C   157
                        001C   158 BEGIN:
          03 54    00   E5  001C   159          BBCC    #0, R4, 10$             ; Skip initial multiply
             50    52   70  0020   160          MOVD    R2, R0                  ; New partial result
    54    54    FF 8F   9C  0023   161 10$:     ROTL    #-1, R4, R4             ; Get new low exponent bit and clear
                        0028   162                                              ; high bit
                   10   13  0028   163          BEQL    DONE                    ; Is that all?  If so, we're done.
                        002A   164
                        002A   165 ;+
                        002A   166 ; Each time we get here we know that there's at least one more exponent bit
                        002A   167 ; left, so square our current power of the base.
                        002A   168 ;-
                        002A   169
                        002A   170 LOOP:
             52   52   64  002A   171          MULD2   R2, R2                  ; Square base.  This may overflow or
                        002D   172                                              ; underflow.  In either case, the
                        002D   173                                              ; final result would also overflow
                        002D   174                                              ; or underflow.  Our exception
                        002D   175                                              ; handler will catch these cases.
             03 54   E9  002D   176          BLBC    R4, NEXT_BIT            ; See if this exponent bit is on.
                        0030   177                                              ; If so, multiply base by product.
             50   52   64  0030   178          MULD2   R2, R0                  ; This too may overflow or underflow,
                        0033   179                                              ; which will be caught by the handler.
                        0033   180 NEXT_BIT:
    54    54    FF 8F   78  0033   181          ASHL    #-1, R4, R4             ; Get next bit of exponent.
                   F0   12  0038   182          BNEQ    LOOP                    ; Keep going if not all done.
                        003A   183
                        003A   184 DONE:
                   04   003A   185          RET                             ; Return with answer in R0/R1.
                        003B   186
                        003B   187 ;+
                        003B   188 ; Undefined exponentation error - 0**0
                        003B   189 ;-
                        003B   190
                        003B   191 UNDEFINED:
       7E   00'8F   9A  003B   192          MOVZBL  #MTH$K_UNDEXP, -(SP)    ; Indicate undefined exponentiation.
    50    01   0F   79  003F   193          ASHQ    #15, #T, R0             ; R0/R1 = reserved floating operand
                   15   11  0043   194          BRB     SIGNAL_ERROR
                        0045   195
```

```
                                0045   196 ;+
                                0045   197 ; EXC_HANDLER unwinds here if we got a floating overflow.
                                0045   198 ;-
                                0045   199
                                0045   200 OVERFLOW:
      7E    00'8F    9A          0045   201         MOVZBL  #MTH$K_FLOOVEMAT, -(SP) ; Floating overflow in Math Library
50    01       0F    79          0049   202         ASHQ    #15, #1, R0             ; R0/R1 = reserved floating operand
                0B    11          004D   203         BRB     SIGNAL_ERROR
                                004F   204
                                004F   205 ;+
                                004F   206 ; EXC_HANDLER unwinds here if we got a floating underflow.
                                004F   207 ; If our caller enabled FU, then signal FLOUNDMAT, with a result of zero.
                                004F   208 ; Otherwise quietly return zero.
                                004F   209 ;-
                                004F   210
                                004F   211 UNDERFLOW:
                50    7C        004F   212         CLRD    R0                      ; Initial result
   E4 04 AD     06    E1        0051   213         BBC     #PSL$V_FU, SF$W_SAVE_PSW(FP), DONE      ; Just return if
                                0056   214                                         ; caller disabled underflow.
      7E    00'8F    9A          0056   215         MOVZBL  #MTH$K_FLOUNDMAT, -(SP) ; Floating underflow in Math Library
                                005A   216 ;       BRB     SIGNAL_ERROR
                                005A   217
                                005A   218 ;+
                                005A   219 ; Signal a MTH$ error whose code has already been pushed on the stack.
                                005A   220 ; R0/R1 has either a reserved operand or a zero, depending on the
                                005A   221 ; exception type.
                                005A   222 ;-
                                005A   223
                                005A   224 SIGNAL_ERROR:
                6D    D4        005A   225         CLRL    (FP)                    ; Cancel our condition handler
00000000'GF     01    FB        005C   226         CALLS   #1, G^MTH$$SIGNAL       ; convert to 32-bit condition code
                                0063   227                                         ; and signal a MTH$ error.  Omit
                                0063   228                                         ; second argument to show that this
                                0063   229                                         ; is a CALL entry point.
                04              0063   230         RET                             ; Return to caller with result in
                                0064   231                                         ; R0/R1
                                0064   232
```

```
                     0064   234 ;+
                     0064   235 ; EXC_HANDLER - This condition handler gets control if we got an exception
                     0064   236 ; while performing the exponentiation.  If it is a floating overflow,
                     0064   237 ; then continue execution at OVERFLOW, to cause MTH$_FLOOVEMAT to be
                     0064   238 ; signalled.  Similarly, continue at UNDERFLOW if an underflow was seen.
                     0064   239 ; Otherwise resignal.
                     0064   240 ;-
                     0064   241
                     0064   242 EXC_HANDLER:
                0000 0064   243         .WORD   ^M<>                            ; Entry point
                     0066   244
      50    08 AC    D0 0066   245         MOVL    CHF$L_MCHARGLST(AP), R0 ; Get mechanism arguments list
            08 A0    D5 006A   246         TSTL    CHF$L_MCH_DEPTH(R0)     ; At depth zero?
               24    12 006D   247         BNEQ    RESIGNAL                ; If not, resignal
      50    04 AC    D0 006F   248         MOVL    CHF$L_SIGARGLST(AP), R0 ; Get signal arguments list
      51    04 A0    D0 0073   249         MOVL    CHF$L_SIG_NAME(R0), R1  ; Get signal name
  04B4 8F     51    B1 0077   250         CMPW    R1, #SS$_FLTOVF_F       ; Overflow fault?
               1B    13 007C   251         BEQL    GOTO_OVERFLOW           ; If so, continue at OVERFLOW
  048C 8F     51    B1 007E   252         CMPW    R1, #SS$_FLTOVF         ; Overflow trap?
               14    13 0083   253         BEQL    GOTO_OVERFLOW           ; If so, continue at OVERFLOW
  04C4 8F     51    B1 0085   254         CMPW    R1, #SS$_FLTUND_F       ; Underflow fault?
               18    13 008A   255         BEQL    GOTO_UNDERFLOW          ; If so, continue at UNDERFLOW
  049C 8F     51    B1 008C   256         CMPW    R1, #SS$_FLTUND         ; Underflow trap?
               11    13 0091   257         BEQL    GOTO_UNDERFLOW          ; If so, continue at UNDERFLOW
                     0093   258 RESIGNAL:
      50  0918 8F    3C 0093   259         MOVZWL  #SS$_RESIGNAL, R0       ; Resignal exception
               04    04 0098   260         RET                             ; Return to VMS exception dispatcher
                     0099   261
                     0099   262 GOTO_OVERFLOW:
            51    60    D0 0099   263         MOVL    CHF$L_SIG_ARGS(R0), R1  ; Get number of signal arguments
  FC A041     A6 AF    DE 009C   264         MOVAL   B^OVERFLOW, -4(R0)[R1]  ; Move address of OVERFLOW routine
                     00A2   265                                             ; to PC in signal argument list
               09    11 00A2   266         BRB     CONTINUE                ; Continue execution
                     00A4   267
                     00A4   268 GOTO_UNDERFLOW:
            51    60    D0 00A4   269         MOVL    CHF$L_SIG_ARGS(R0), R1  ; Get number of signal arguments
  FC A041     A5 AF    DE 00A7   270         MOVAL   B^UNDERFLOW, -4(R0)[R1] ; Move address of UNDERFLOW routine
                     00AD   271                                             ; to PC in signal argument list
                     00AD   272 ;       BRB     CONTINUE                ; Continue execution
                     00AD   273
                     00AD   274 CONTINUE:
            50    01    3C 00AD   275         MOVZWL  #SS$_CONTINUE, R0       ; Continue at PC in signal list
                     04    04 00B0   276         RET                             ; Return to VMS exception dispatcher
                     00B1   277
                     00B1   278
                     00B1   279         .END                            ; End of module OTS$POWDLU
```

```
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                        = 0000000C
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$K_UNDEXP                      ******** *    X 00
NEXT_BIT                          00000033 R      02
OTS$POWDLU                        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                    = 00000484
SS$_FLTUND                      = 0000049C
SS$_FLTUND_F                    = 00000494
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	29	00:00:00.09	00:00:01.01
Command processing	112	00:00:00.53	00:00:03.14
Pass 1	215	00:00:05.02	00:00:14.85
Symbol table sort	0	00:00:00.78	00:00:01.25
Pass 2	61	00:00:01.15	00:00:03.04
Symbol table output	6	00:00:00.05	00:00:00.13

Psect synopsis output	2	00:00:00.03	00:00:00.04
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	427	00:00:07.65	00:00:23.48

The working set limit was 1200 pages.
26577 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.
279 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$:OTSPOWDLU/OBJ=OBJ$:OTSPOWDLU MSRC$:OTSPOWDLU/UPDATE=(ENH$:OTSPOWDLU)

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