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

(2) 51 HISTORY ; Detailed Current Edit History
(3) 72 DECLARATIONS
(4) 112 OTS$POWRJ - floating to power longword giving floating result

```
0000     1          .TITLE  OTS$POWRJ - REAL ** INTEGER*4 power routine
0000     2          .IDENT  /1-006/                 ; File: OTSPOWRJ.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 ;       Floating base to integer longword power.
0000    34 ;       Floating overflow and underflow can occur.
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 ; Ed t History for Version 0 of OTS$POWRJ
0000     55 ; version 05 - changed module name to forpowRJ
0000     56 ; version 07 - changed error handler from MTH$ERROR to MTH$$ERROR
0000     57 ; version 08 - removed W^ from MTH$$ERROR, saved code with MOVZBL.
0000     58 ;               removed infinite loop with largest negative integer exponent.
0000     59
0000     60 ; version 09 - changed MTH$$ERROR to MTH$$SIGNAL - JMT
0000     61 ; 1-001 - Update version number and copyright notice.  The previous
0000     62 ;          version number was 0-10.  JBS 16-NOV-78.
0000     63 ; 1-002 - Change MTH__UNDEXP to MTH$K_UNDEXP.  JBS 07-DEC-78
0000     64 ; 1-003 - Add "_" to the PSECT directive.  JBS 22-DEC-78
0000     65 ; 1-004 - Declare externals.  SBL 17-May-1979
0000     66 ; 1-005 - Add handlers to catch SS$_FLTOVF and SS$_FLTDIV, and signal
0000     67 ;          MTH$_FLOOVEMAT or MTH$_FLOUNDMAT instead, depending on the context.
0000     68 ;          JAW 26-Feb-1980.
0000     69 ; 1-006 - Use general mode addressing.  SBL 30-Nov-1981
0000     70 ;
```

```
                     0000     72          .SBTTL  DECLARATIONS
                     0000     73
                     0000     74 ;
                     0000     75 ; INCLUDE FILES:
                     0000     76 ;
                     0000     77
                     0000     78 ;
                     0000     79 ; EXTERNAL SYMBOLS:
                     0000     80 ;
                     0000     81          .DSABL  GBL
                     0000     82          .EXTRN  MTH$K_UNDEXP, MTH$K_FLOOVEMAT, MTH$K_FLOUNDMAT
                     0000     83          .EXTRN  MTH$$SIGNAL             ; Math error routine
                     0000     84          .EXTRN  SS$_FLTOVF, SS$_FLTOVF_F, SS$_FLTDIV, SS$_FLTDIV_F, SS$_CONTINUE
                     0000     85
                     0000     86 ;
                     0000     87 ; MACROS:
                     0000     88 ;
                     0000     89          $CHFDEF                         ; Define condition handler symbols.
                     0000     90          $SFDEF                          ; Define stack frame symbols.
                     0000     91          $PSLDEF                         ; Define program status longword
                     0000     92                                          ; symbols.
                     0000     93
                     0000     94 ;
                     0000     95 ; EQUATED SYMBOLS:
                     0000     96 ;
00000004             0000     97          base = 4                        ; base input formal - by-value
00000008             0000     98          exp = 8                         ; exponent intpu formal - by-value
                     0000     99
                     0000    100 ;
                     0000    101 ; OWN STORAGE:
                     0000    102 ;
                     0000    103
                     0000    104 ;
                     0000    105 ; PSECT DECLARATIONS:
                     0000    106 ;
                     0000    107
                 00000000    108          .PSECT  _OTS$CODE PIC,SHR,LONG,EXE,NOWRT
                     0000    109                                          ; program section for OTS$ code
                     0000    110
```

```
0000    112          .SBTTL  OTS$POWRJ - floating to power longword giving floating result
0000    113
0000    114 ;++
0000    115 ; FUNCTIONAL DESCRIPTION:
0000    116 ;
0000    117 ;       Floating result = floating base ** signed longword exponent
0000    118 ;       The floating result is given by:
0000    119 ;
0000    120 ;       base    exponent        result
0000    121 ;
0000    122 ;       any     > 0             product (base * 2**i) where i is each
0000    123 ;                               non-zero bit position in exponent
0000    124 ;
0000    125 ;       > 0     = 0             1.0
0000    126 ;       = 0     = 0             Undefined exponentation
0000    127 ;       < 0     = 0             1.0
0000    128 ;
0000    129 ;       > 0     < 0             1.0 / product (base * 2**i)
0000    130 ;                               where i is each non-zero bit position
0000    131 ;                               in !exponent!
0000    132 ;       = 0     < 0             Undefined exponentation
0000    133 ;       < 0     < 0             1.0 / product (base * 2**i)
0000    134 ;                               where i is each non-zero bit position
0000    135 ;                               in !exponent!
0000    136 ;
0000    137 ;       Floating overflow can occur on either of the two MULF's.  If this
0000    138 ;       happens when the exponent is less than zero, the exception is caught by
0000    139 ;       a local condition handler named EXC_HNDLR_UNDER, which sets the result
0000    140 ;       to 0.0 and either signals MTH$_FLOUNDMAT (if FU is enabled in the
0000    141 ;       caller's PSW) or continues at POWRJX.  If it happens when the exponent
0000    142 ;       is greater than zero, the exception is caught by a local condition
0000    143 ;       handler named EXC_HNDLR_OVER, which sets the result to the reserved
0000    144 ;       operand (-0.0) and signals MTH$_FLOOVEMAT.
0000    145 ;
0000    146 ;       Floating overflow and floating divide by zero can occur on the DIVF.
0000    147 ;       These exceptions are caught by EXC_HNDLR_OVER, which sets the result to
0000    148 ;       the reserved operand (-0.0) and signals MTH$_FLOOVEMAT.
0000    149 ;
0000    150 ;       Undefined exponentiation occurs if base is 0 and
0000    151 ;       exponent is 0 or negative.
0000    152 ;
0000    153 ; CALLING SEQUENCE:
0000    154 ;
0000    155 ;       Power.wf.v = OTS$POWRJ (base.rf.v,  exponent.rl.v)
0000    156 ;
0000    157 ; INPUT PARAMETERS:
0000    158 ;       NONE
0000    159 ;
0000    160 ; IMPLICIT INPUTS:
0000    161 ;       The setting of FU in the caller's PSW.
0000    162 ;
0000    163 ; OUTPUT PARAMETERS:
0000    164 ;       NONE
0000    165 ;
0000    166 ; IMPLICIT OUTPUTS:
0000    167 ;       NONE
0000    168 ;
```

```
                                0000   169 ; FUNCTION VALUE:
                                0000   170 ;
                                0000   171 ;       Floating base ** exponent
                                0000   172 ;
                                0000   173 ; SIDE EFFECTS:
                                0000   174 ;
                                0000   175 ;       Signals MTH$_FLOOVEMAT if floating overflow occurs on either of the two
                                0000   176 ;           MULF's when exponent > 0, or if floating overflow or divide by zero
                                0000   177 ;           occurs on the DIVF.
                                0000   178 ;       Signals MTH$_FLOUNDMAT if floating overflow occurs on either of the two
                                0000   179 ;           MULF's when exponent < 0 and caller has FU enabled.
                                0000   180 ;       Signals MTH$_UNDEXP (82 = ' UNDEFINED exponentation') if
                                0000   181 ;           base is 0 and exponent is 0 or negative.
                                0000   182 ;
                                0000   183 ;--
                                0000   184
                                0000   185
                                0000   186
                        0004    0000   187         .ENTRY OTS$POWRJ, ^M<R2>            ; disable integer overflow
                                0002   188                                             ; occurs on largest negative integer exp
6D      66'AF     9E            0002   189         MOVAB   B^EXC_HNDLR_OVER, (FP)      ; Translate exceptions to
                                0006   190                                             ; MTH$_FLOOVEMAT.
   50     08     50            0006   191         MOVF    #1, R0                      ; R0 = initial result
51     04 AC     50            0009   192         MOVF    base(AP), R1                ; R1 = base
52     08 AC     D0            000D   193         MOVL    exp(AP), R2                 ; R2 = exponent
                                0011   194                                             ; Note: integer overflow can occur
                                0011   195                                             ; on largest neagtive integer exponent.
                                0011   196                                             ; However, R2 is correct unsigned 32-bit val
                                0011   197                                             ; Use ROTL -1 rather than ASHL -1 below.
        0D     14               0011   198         BGTR    EXPGTR                      ; branch if exponent > 0
6D      57'AF     9E            0013   199         MOVAB   B^EXC_HNDLR_UNDER, (FP)     ; Translate exceptions to
                                0017   200                                             ; MTH$_FLOUNDMAT.
                                0017   201
        51     53               0017   202         TSTF    R1                          ; test base
        2C     13               0019   203         BEQL    UNDEFINED                   ; undefined 0**0 or 0**(-n)
   52   52     CE               001B   204         MNEGL   R2, R2                      ; R2 = |exponent|
        26     13               001E   205         BEQL    POWRJX                      ; if exponent is 0, return R0 = 1.0
                                0020   206
                                0020   207 ;+
                                0020   208 ; Exponent is > 0 or (exponent is =< 0 and base is not = 0 -- use |exponent|)
                                0020   209 ;-
                                0020   210
    0B 52     00     E4         0020   211 EXPGTR: BBSC    #0, R2, PARTIAL             ; branch if |exponent| is odd
                                0024   212                                             ; and clear low order bit
52  52   FF 8F     9C           0024   213 SQUAR:  ROTL    #-1, R2, R2                 ; R2 = |32-bit unsigned exponent|/2
   51   51     44               0029   214 SQUAR1: MULF    R1, R1                      ; R1 = current power of base
                                002C   215                                             ; Floating overflow will trap or fault
                                002C   216                                             ; and signal SS$_FLTOVF or SS$_FLTOVF_F.
     F5 52     E9               002C   217         BLBC    R2, SQUAR                   ; branch if next bit in |exponent| is 0
                                002F   218
                                002F   219 ;+
                                002F   220 ; Here when bit i of |exponent| is a 1.
                                002F   221 ; Partial result = partial result *  (base * 2**i)
                                002F   222 ;-
                                002F   223
                                002F   224 PARTIAL:
   50   51     44               002F   225         MULF    R1, R0                      ; R0 = new partial result
```

```
                52   52    FF 8F    78   0032   226          ASHL    #-1, R2, R2                 ; R2 = |exponent|/2
                               F0    12   0037   227          BNEQ    SQUAR1                      ; loopback if more exponent bits are 1
                                     0039   228
                          08 AC     D5   0039   229          TSTL    exp(AP)                     ; test sign of exponent
                                08   14   003C   230          BGTR    POWRJX                      ; if exponent > 0, return R0
                     6D   66'AF     9E   003E   231          MOVAB   B^EXC_HNDLR_OVER, (FP)      ; Translate exceptions to
                                     0042   232                                              ; MTH$_FLOOVEMAT.
                50   08   50       47   0042   233          DIVF3   R0, #1, R0                  ; R0 = 1.0/result
                                04   0046   234 POWRJX:: RET                                   ; return, result in R0
                                     0047   235
                                     0047   236 ;+
                                     0047   237 ; Undefined exponentation error - 0**0 or 0**(-n)
                                     0047   238 ;-
                                     0047   239
                                     0047   240 UNDEFINED:
                50   01   0F       78   0047   241          ASHL    #15, #1, R0                 ; R0 = reserved floating operand
                  7E   00'8F       9A   004B   242          MOVZBL  #MTH$K_UNDEXP, -(SP)        ; Indicate undefined exponentiation.
          00000000'GF   01       FB   004F   243          CALLS   #1, G^MTH$$SIGNAL           ; convert to 32-bit condition code
                                     0056   244                                              ; and SIGNAL MTH$_UNDEXP
                                     0056   245                                              ; Note: 2nd arg not needed since
                                     0056   246                                              ; no JSB OTS$POWRJ
                                04   0056   247          RET                                 ; return
                                     0057   248
                                     0057   249 ;+
                                     0057   250 ; The following handler is established to process exceptions which imply
                                     0057   251 ; underflow of the final result (floating overflow in either of the two MULF's
                                     0057   252 ; when exp < 0).  On the occurrence of such an exception, the handler signals
                                     0057   253 ; MTH$_FLOUNDMAT.
                                     0057   254 ;-
                                     0057   255
                                     0057   256 EXC_HNDLR_UNDER:
                          001C       0057   257          .WORD   ^M<R2, R3, R4>              ; Entry mask
                     2B   10       0059   258          BSBB    SETUP                       ; Set up R0:R3 and identify condition.
                                     005B   259                                              ; Return only if FLTOVF or FLTDIV.
          1E 04 A2     06   E1   005B   260          BBC     #PSL$V_FU, SF$W_SAVE_PSW(R2), CON_U
                                     0060   261                                              ; Branch if caller has not enabled FU.
                54   00'8F       9A   0060   262          MOVZBL  #MTH$K_FLOUNDMAT, R4        ; Report MTH$_FLOUNDMAT, not SS$_FLTOVF.
                          0C       11   0064   263          BRB     DO_SIG
                                     0066   264
                                     0066   265 ;+
                                     0066   266 ; The following handler is established to process exceptions which imply
                                     0066   267 ; overflow of the final result (floating overflow in either of the two MULF's
                                     0066   268 ; when exp > 0, floating overflow in the DIVF, or floating divide by zero in the
                                     0066   269 ; DIVF).  On the occurrence of such an exception, the handler signals
                                     0066   270 ; MTH$_FLOOVEMAT.
                                     0066   271 ;-
                                     0066   272
                                     0066   273 EXC_HNDLR_OVER:
                          001C       0066   274          .WORD   ^M<R2, R3, R4>              ; Entry mask
                     1C   10       0068   275          BSBB    SETUP                       ; Set up R0:R3 and identify condition.
                                     006A   276                                              ; Return only if FLTOVF or FLTDIV.
                50   01   0F       78   006A   277          ASHL    #15, #1, R0                 ; Make the default result -0.0.
                  54   00'8F       9A   006E   278          MOVZBL  #MTH$K_FLOOVEMAT, R4        ; Report MTH$_FLOOVEMAT, not SS$_FLTxxx.
                                     0072   279
                     10 A2       DD   0072   280 DO_SIG:: PUSHL   SF$L_SAVE_PC(R2)           ; Report caller's PC, not exception PC.
                          54       DD   0075   281          PUSHL   R4                          ; Report MTH$_xxx, not SS$_xxx.
          00000000'GF   02       FB   0077   282          CALLS   #2, G^MTH$$SIGNAL           ; Signal the condition.
```

```
                                0086   287 ;+
                                0086   288 ; Common setup routine for handlers.  Returns normally if exception was FLTOVF,
                                0086   289 ; FLTOVF_F, FLTDIV, or FLTDIV_F.  If the exception was anything else, it
                                0086   290 ; executes a RET, causing an exit from the handler with R0 = 0, which is
                                0086   291 ; equivalent to $SS$_RESIGNAL.  In the case of a normal return (FLTOVF, FLTOVF_F,
                                0086   292 ; FLTDIV, or FLTDIV_F) it sets up R0:R3 as follows:
                                0086   293 ;       R0/R1:  0
                                0086   294 ;       R2:     address of establisher's frame
                                0086   295 ;       R3:     address of mechanism array
                                0086   296 ;-
                                0086   297
                           50   7C   0086   298 SETUP:  CLRQ    R0                              ; Set default result to 0.0.
                      52   04 AC   7D   0088   299          MOVQ    CHF$L_SIGARGLST(AP), R2 ; R2 = address of signal array
                                      008C   300                                          ; R3 = address of mechanism array
                0000'8F    04 A2   B1   008C   301          CMPW    CHF$L_SIG_NAME(R2), #SS$_FLTOVF
                                      0092   302                                          ; Was it a floating overflow trap?
                           18   13   0092   303          BEQL    DO_RSB                  ; Branch if yes.
                0000'8F    04 A2   B1   0094   304          CMPW    CHF$L_SIG_NAME(R2), #SS$_FLTOVF_F
                                      009A   305                                          ; Or a floating overflow fault?
                           10   13   009A   306          BEQL    DO_RSB                  ; Branch if yes.
                0000'8F    04 A2   B1   009C   307          CMPW    CHF$L_SIG_NAME(R2), #SS$_FLTDIV
                                      00A2   308                                          ; Or a floating divide by zero trap?
                           08   13   00A2   309          BEQL    DO_RSB                  ; Branch if yes.
                0000'8F    04 A2   B1   00A4   310          CMPW    CHF$L_SIG_NAME(R2), #SS$_FLTDIV_F
                                      00AA   311                                          ; Or a floating divide by zero fault?
                           D9   12   00AA   312          BNEQ    DO_RET                  ; None of the above: return from handler
                                      00AC   313                                          ; with R0 = 0.
          08 A2    97 AF   9E   00AC   314 DO_RSB: MOVAB   B^POWRJX, CHF$L_SIG_NAME+4(R2)
                                      00B1   315                                          ; Change return PC to POWRJX.
             52    04 A3   D0   00B1   316          MOVL    CHF$L_MCH_FRAME(R3), R2 ; R2 = address of establisher's frame
                           05   00B5   317          RSB                                 ; Return.
                                00B6   318
                                00B6   319          .END
```

```
BASE                =  00000004
CHF$L_MCH_FRAME     =  00000004
CHF$L_MCH_SAVR0     =  0000000C
CHF$L_SIGARGLST     =  00000004
CHF$L_SIG_NAME      =  00000004
CON_U                  0000007E  R      02
DO_RET                 00000085  R      02
DO_RSB                 000000AC  R      02
DO_SIG                 00000072  R      02
EXC_HNDLR_OVER         00000066  R      02
EXC_HNDLR_UNDER        00000057  R      02
EXP                 =  00000008
EXPGTR                 00000020  R      02
MTH$$SIGNAL            ********    X    00
MTH$K_FLOOVEMAT        ********    X    00
MTH$K_FLOUNDMAT        ********    X    00
MTH$K_UNDEXP           ********    X    00
OTS$POWRJ              00000000  RG     02
PARTIAL                0000002F  R      02
POWRJX                 00000046  R      02
PSL$V_FU            =  00000006
SETUP                  00000086  R      02
SF$L_SAVE_PC        =  00000010
SF$W_SAVE_PSW       =  00000004
SQUAR                  00000024  R      02
SQUAR1                 00000029  R      02
SS$_CONTINUE           ********    X    00
SS$_FLTDIV             ********    X    00
SS$_FLTDIV_F           ********    X    00
SS$_FLTOVF             ********    X    00
SS$_FLTOVF_F           ********    X    00
UNDEFINED              00000047  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	000000B6 (182.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.06	00:00:00.96
Command processing	111	00:00:00.54	00:00:04.51
Pass 1	137	00:00:01.86	00:00:06.28
Symbol table sort	0	00:00:00.11	00:00:00.19
Pass 2	74	00:00:00.81	00:00:02.42
Symbol table output	4	00:00:00.06	00:00:00.20
Psect synopsis output	3	00:00:00.01	00:00:00.10
Cross-reference output	0	00:00:00.00	00:00:00.00

```
Assembler run totals              361     00:00:03.47    00:00:14.67

The working set limit was 900 pages.
8678 bytes (17 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 106 non-local and 0 local symbols.
319 source lines were read in Pass 1, producing 13 object records in Pass 2.
10 pages of virtual memory were used to define 9 macros.

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

Macro library name                       Macros defined
------------------                       --------------
_$255$DUA28:[SYSLIB]STARLET.MLB;2                 6

148 GETS were required to define 6 macros.

There were no errors, warnings or information messages.

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

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