```
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       HHHHHHHHHHHHHHH  RRRRRRRRRRR         TTT         LLL
MMM       MMM           TTT       HHHHHHHHHHHHHHH  RRRRRRRRRRRR        TTT         LLL
MMM       MMM           TTT       HHHHHHHHHHHHHHH  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         LLLLLLLLLLLLLLL
MMM       MMM           TTT       HHH          HHH  RRR         RRR     TTT         LLLLLLLLLLLLLLL
MMM       MMM           TTT       HHH          HHH  RRR         RRR     TTT         LLLLLLLLLLLLLLL
```

FILEID**OTSPOWHJ

```
  OOOOOO    TTTTTTTTTT    SSSSSSSS  PPPPPPPP      OOOOOO    WW      WW  HH      HH          JJ
  OOOOOO    TTTTTTTTTT    SSSSSSSS  PPPPPPPP      OOOOOO    WW      WW  HH      HH          JJ
OO      OO      TT      SS          PP      PP  OO      OO  WW      WW  HH      HH          JJ
OO      OO      TT      SS          PP      PP  OO      OO  WW      WW  HH      HH          JJ
OO      OO      TT      SS          PP      PP  OO      OO  WW      WW  HH      HH          JJ
OO      OO      TT      SS          PP      PP  OO      OO  WW      WW  HH      HH          JJ
OO      OO      TT        SSSSSS    PPPPPPPP    OO      OO  WW      WW  HHHHHHHHHH          JJ
OO      OO      TT        SSSSSS    PPPPPPPP    OO      OO  WW      WW  HHHHHHHHHH          JJ
OO      OO      TT              SS  PP          OO      OO  WW  WW  WW  HH      HH  JJ      JJ
OO      OO      TT              SS  PP          OO      OO  WW  WW  WW  HH      HH  JJ      JJ
OO      OO      TT              SS  PP          OO      OO  WWWW  WWWW  HH      HH  JJ      JJ      ....
OO      OO      TT              SS  PP          OO      OO  WWWW  WWWW  HH      HH  JJ      JJ      ....
  OOOOOO        TT      SSSSSSSS    PP            OOOOOO    WW      WW  HH      HH    JJJJJJ        ....
  OOOOOO        TT      SSSSSSSS    PP            OOOOOO    WW      WW  HH      HH    JJJJJJ        ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  OTS$POWHJ - REAL*16 ** INTEGER*4 power routine
0000     2          .IDENT  /1-004/                 ; File: OTSPOWHJ.MAR  Edit: SBL1004
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 ;       REAL*16 base to INTEGER*4 power.
0000    34 ;       Floating overflow and underflow can occur.
0000    35 ;       Undefined exponentation can occur if base is 0 and power
0000    36 ;       is 0 or negative.
0000    37 ;
0000    38 ;
0000    39 ;--
0000    40 ;
0000    41 ; VERSION: 1
0000    42 ;
0000    43 ; HISTORY:
0000    44 ; AUTHOR:
0000    45 ;       Steven B. Lionel, 1-Jun-79: Version 1
0000    46 ;
0000    47 ;
0000    48 ;
0000    49 ;
```

```
                                    J 3
0000     51          .SBTTL  HISTORY         ; Detailed Current Edit History
0000     52
0000     53
0000     54 ; Edit History for Version 1 of OTS$POWHJ
0000     55 ; 1-001 - Adapted from OTS$POWDJ version 1-001.  SBL 01-Jun-79
0000     56 ; 1-002 - Recast some comments.  JBS 30-JUL-1979
0000     57 ; 1-003 - Add handlers to catch SS$_FLTOVF and SS$_FLTDIV, and signal
0000     58 ;         MTH$_FLOOVEMAT or MTH$_FLOUNDMAT instead, depending on the context.
0000     59 ;         Also disable IV and change BLBS/ASHL at EXPGTR to BBSC/ROTL for
0000     60 ;         uniformity with OTS$POWRJ.  JAW 26-Feb-1980.
0000     61 ; 1-004 - Use word-relative addressing for handler address.  SBL 9-Sep-1981
0000     62 ;
```

```
                     0000     64          .SBTTL  DECLARATIONS
                     0000     65
                     0000     66 ;
                     0000     67 ; INCLUDE FILES:
                     0000     68 ;
                     0000     69
                     0000     70 ;
                     0000     71 ; EXTERNAL SYMBOLS:
                     0000     72 ;
                     0000     73
                     0000     74          .DSABL  GBL
                     0000     75          .EXTRN  MTH$K_UNDEXP, MTH$K_FLOOVEMAT, MTH$K_FLOUNDMAT
                     0000     76          .EXTRN  MTH$$SIGNAL                     ; Math error routine
                     0000     77          .EXTRN  SS$_FLTOVF, SS$_FLTOVF_F, SS$_FLTDIV, SS$_FLTDIV_F, SS$_CONTINUE
                     0000     78 ;
                     0000     79 ; MACROS:
                     0000     80 ;
                     0000     81          $CHFDEF                                 ; Define condition handler symbols.
                     0000     82          $SFDEF                                  ; Define stack frame symbols.
                     0000     83          $PSLDEF                                 ; Define program status longword
                     0000     84                                                  ; symbols.
                     0000     85
                     0000     86 ;
                     0000     87 ; EQUATED SYMBOLS:
                     0000     88 ;
00000004             0000     89          base = 4                                ; base input formal - by-value
00000014             0000     90          exp = 20                                ; exponent input formal - by-value
                     0000     91                                                  ; Note: quad floating by-value violates
                     0000     92                                                  ; calling standard, but ok since this
                     0000     93                                                  ; routine is a code support routine (OTS$)
                     0000     94
                     0000     95 ;
                     0000     96 ; OWN STORAGE:
                     0000     97 ;
                     0000     98
                     0000     99 ;
                     0000    100 ; PSECT DECLARATIONS:
                     0000    101 ;
                     0000    102
                     0000    103          .PSECT  _OTS$CODE PIC,SHR,LONG,EXE,NOWRT
                     0000    104                                                  ; program section for OTS$ code
                     0000    105
```

```
                                L 3
0000    107          .SBTTL  OTS$POWHJ_R3 - REAL*16 ** INTEGER*4
0000    108
0000    109 ;++
0000    110 ; FUNCTIONAL DESCRIPTION:
0000    111 ;
0000    112 ;       REAL*16 result = REAL*16 base ** signed longword exponent
0000    113 ;       The REAL*16 result is given by:
0000    114 ;
0000    115 ;       base    exponent        result
0000    116 ;
0000    117 ;       any     > 0             product (base * 2**i) where i is each
0000    118 ;                               non-zero bit position in exponent
0000    119 ;
0000    120 ;       > 0     = 0             1.0
0000    121 ;       = 0     = 0             Undefined exponentation
0000    122 ;       < 0     = 0             1.0
0000    123 ;
0000    124 ;       > 0     < 0             1.0 / product (base * 2**i)
0000    125 ;                               where i is each non-zero bit position
0000    126 ;                               in !exponent!
0000    127 ;       = 0     < 0             Undefined exponentation
0000    128 ;       < 0     < 0             1.0 / product (base * 2**i)
0000    129 ;                               where i is each non-zero bit position
0000    130 ;                               in !exponent!
0000    131 ;
0000    132 ;       Floating overflow can occur on either of the two MULH's.  If this
0000    133 ;       happens when the exponent is less than zero, the exception is caught by
0000    134 ;       a local condition handler named EXC_HNDLR_UNDER, which sets the result
0000    135 ;       to 0.0 and either signals MTH$_FLOUNDMAT (if FU is enabled in the
0000    136 ;       caller's PSW) or continues at POWHJX.  If it happens when the exponent
0000    137 ;       is greater than zero, the exception is caught by a local condition
0000    138 ;       handler named EXC_HNDLR_OVER, which sets the result to the reserved
0000    139 ;       operand (-0.0) and signals MTH$_FLOOVEMAT.
0000    140 ;
0000    141 ;       Floating overflow and floating divide by zero can occur on the DIVH.
0000    142 ;       These exceptions are caught by EXC_HNDLR_OVER, which sets the result to
0000    143 ;       the reserved operand (-0.0) and signals MTH$_FLOOVEMAT.
0000    144 ;
0000    145 ;       Undefined exponentiation occurs if base is 0 and
0000    146 ;       exponent is 0 or negative.
0000    147 ;
0000    148 ; CALLING SEQUENCE:
0000    149 ;
0000    150 ;       Power.wh.v = OTS$POWHJ_R3 (base.rh.v, exponent.rl.v)
0000    151 ;
0000    152 ; INPUT PARAMETERS:
0000    153 ;       base            - REAL*16 base
0000    154 ;       exponent        - INTEGER*4 exponent
0000    155 ;
0000    156 ; IMPLICIT INPUTS:
0000    157 ;       The setting of FU in the caller's PSW.
0000    158 ;
0000    159 ; OUTPUT PARAMETERS:
0000    160 ;       NONE
0000    161 ;
0000    162 ; IMPLICIT OUTPUTS:
0000    163 ;       NONE
```

```
                     0000   164 ;
                     0000   165 ; FUNCTION VALUE:
                     0000   166 ;
                     0000   167 ;                   WARNING !!
                     0000   168 ;
                     0000   169 ;       REAL*16 result in registers R0 through R3.
                     0000   170 ;       This is a serious violation of the VAX calling standard and can
                     0000   171 ;       kill programs if misused.  Beware!
                     0000   172 ;
                     0000   173 ; SIDE EFFECTS:
                     0000   174 ;
                     0000   175 ;       Destroys registers R2-R3!
                     0000   176 ;       Signals MTH$_FLOOVEMAT if floating overflow occurs on either of the two
                     0000   177 ;           MULH's when exponent > 0, or if floating overflow or divide by zero
                     0000   178 ;           occurs on the DIVH.
                     0000   179 ;       Signals MTH$_FLOUNDMAT if floating overflow occurs on either of the two
                     0000   180 ;           MULH's when exponent < 0 and caller has FU enabled.
                     0000   181 ;       SIGNALs MTH$_UNDEXP (82 = ' UNDEFINED EXPONENTATION') if
                     0000   182 ;       base is 0 and exponent is 0 or negative.
                     0000   183 ;
                     0000   184 ;--
                     0000   185
                     0000   186
                     0000   187
                01F0 0000   188           .ENTRY  OTS$POWHJ_R3, ^M<R4, R5, R6, R7, R8>
                     0002   189                                           ; Don't save registers R2 and R3!
                     0002   190                                           ; Disable integer overflow.  (Occurs on
                     0002   191                                           ; maximum negative exponent.)
        6D 006F'CF   9E 0002   192           MOVAB   W^EXC_HNDLR_OVER, (FP)  ; Translate exceptions to
                     0007   193                                           ; MTH$_FLOOVEMAT.
        50    08 70FD 0007   194           MOVH    #1, R0                  ; R0-R3 = initial result
     54   04 AC 70FD 000B   195           MOVH    base(AP), R4            ; R4-R7 = base
     58   14 AC   D0 0010   196           MOVL    exp(AP), R8             ; R8 = exponent
           0E   14 0014   197           BGTR    EXPGTR                  ; branch if exponent > 0
     6D  60'AF   9E 0016   198           MOVAB   B^EXC_HNDLR_UNDER, (FP) ; Translate exceptions to
                     001A   199                                           ; MTH$_FLOUNDMAT.
                     001A   200
           54 73FD 001A   201           TSTH    R4                      ; test base
           2F   13 001D   202           BEQL    UNDEFINED               ; undefined 0**0 or 0**(-n)
        58   58   CE 001F   203           MNEGL   R8, R8                  ; R8 = |exponent|
           29   13 0022   204           BEQL    POWHJX                  ; if exponent is 0, return R0 = 1.0
                     0024   205
                     0024   206 ;+
                     0024   207 ; Exponent is > 0 or (exponent is =< 0 and base is not = 0 -- use |exponent|)
                     0024   208 ;-
                     0024   209
        0C 58   00   E4 0024   210 EXPGTR: BBSC    #0, R8, PARTIAL         ; branch if |exponent| is odd
     58 58   FF 8F   9C 0028   211 SQUAR:  ROTL    #-1, R8, R8             ; R8 = |exponent|/2
        54   54 64FD 002D   212 SQUAR1: MULH2   R4, R4                  ; R4-R7 = current power of base
                     0031   213                                           ; floating overflow will trap or fault
                     0031   214                                           ; and signal SS$_FLTOVF or SS$_FLTOVF_F.
           F4 58   E9 0031   215           BLBC    R8, SQUAR               ; branch if next bit in |exponent| is 0
                     0034   216
                     0034   217 ;+
                     0034   218 ; Here when bit i of |exponent| is a 1.
                     0034   219 ; Partial result = partial result *  (base * 2**i)
                     0034   220 ;-
```

```
                                0034   221
                                0034   222 PARTIAL:
                     50   54 64FD  0034   223          MULH2   R4, R0                      ; R0-R3 = new partial result
                58   58   FF 8F    78  0038   224          ASHL    #-1, R8, R8                 ; R8 = |exponent|/2
                          EE       12  003D   225          BNEQ    SQUAR1                      ; loopback if more exponent bits are 1
                                   003F   226
                      14 AC       D5  003F   227          TSTL    exp(AP)                     ; test sign of exponent
                         09       14  0042   228          BGTR    POWHJX                      ; if exponent > 0, return R0
                   6D   6F'AF     9E  0044   229          MOVAB   B^EXC_HNDLR_OVER, (FP)      ; Translate exceptions to
                                   0048   230                                              ; MTH$_FLOOVEMAT.
                50   08   50 67FD  0048   231          DIVH3   R0, #1, R0                  ; R0-R3 = 1.0/result
                             04    004D   232 POWHJX: RET                                 ; return, result in R0-R3
                                   004E   233
                                   004E   234 ;+
                                   004E   235 ; Undefined exponentation error - 0**0 or 0**(-n)
                                   004E   236 ;-
                                   004E   237
                                   004E   238 UNDEFINED:
                50   01   0F    79  004E   239          ASHQ    #15, #1, R0                 ; R0-R3 = reserved floating operand
                          52     7C  0052   240          CLRQ    R2                          ; second quadword
                   7E   00'8F     9A  0054   241          MOVZBL  #MTH$K_UNDEXP, -(SP)        ; Indicate undefined exponentiation.
          00000000'GF    01     FB  0058   242          CALLS   #1, G^MTH$$SIGNAL           ; convert to 32-bit condition code
                                   005F   243                                              ; and SIGNAL MTH$_UNDEXP
                                   005F   244                                              ; Note: 2nd arg not needed since no JSB OTS$
                                   005F   245                                              ; is possible.
                             04    005F   246          RET                                 ; return
                                   0060   247
                                   0060   248 ;+
                                   0060   249 ; The following handler is established to process exceptions which imply
                                   0060   250 ; underflow of the final result (floating overflow in either of the two MULH's
                                   0060   251 ; when exp < 0).  On the occurrence of such an exception, the handler signals
                                   0060   252 ; MTH$_FLOUNDMAT.
                                   0060   253 ;-
                                   0060   254
                                   0060   255 EXC_HNDLR_UNDER:
                         0070      0060   256          .WORD   ^M<R4, R5, R6>              ; Entry mask.
                                   0062   257                                              ; Don't save registers R2 and R3.
                      2B   10      0062   258          BSBB    SETUP                       ; Set up R0:R5 and identify condition.
                                   0064   259                                              ; Return only if FLTOVF or FLTDIV.
          1E 04 A4      06   E1    0064   260          BBC     #PSL$V_FU, SF$W_SAVE_PSW(R4), CON_U
                                   0069   261                                              ; Branch if caller has not enabled FU.
                56   00'8F     9A  0069   262          MOVZBL  #MTH$K_FLOUNDMAT, R6        ; Report MTH$_FLOUNDMAT, not SS$_FLTOVF.
                         0C    11  006D   263          BRB     DO_SIG
                                   006F   264
                                   006F   265 ;+
                                   006F   266 ; The following handler is established to process exceptions which imply
                                   006F   267 ; overflow of the final result (floating overflow in either of the two MULH's
                                   006F   268 ; when exp > 0, floating overflow in the DIVH, or floating divide by zero in the
                                   006F   269 ; DIVH).  On the occurrence of such an exception, the handler signals
                                   006F   270 ; MTH$_FLOOVEMAT.
                                   006F   271 ;-
                                   006F   272
                                   006F   273 EXC_HNDLR_OVER:
                         0070      006F   274          .WORD   ^M<R4, R5, R6>              ; Entry mask
                                   0071   275                                              ; Don't save registers R2 and R3.
                      1C   10      0071   276          BSBB    SETUP                       ; Set up R0:R5 and identify condition.
                                   0073   277                                              ; Return only if FLTOVF or FLTDIV.
```

```
      50   01    0F   78  0073            ASHL    #15, #1, R0                 ; Make the default result -0.0.
      56   00'8F      9A  0077            MOVZBL  #MTH$K_FLOOVEMAT, R6        ; Report MTH$_FLOOVEMAT, not SS$_FLTxxx.
                          007B
            10 A4     DD  007B   DO_SIG:  PUSHL   SF$L_SAVE_PC(R4)            ; Report caller's PC, not exception PC.
               56     DD  007E            PUSHL   R6                          ; Report MTH$_xxx, not SS$_xxx.
00000000'GF    02     FB  0080            CALLS   #2, G^MTH$$SIGNAL           ; Signal the condition.
      0C A5    50     7D  0087   CON_U:   MOVQ    R0, CHF$L_MCH_SAVR0(R5)     ; If continued, restore R0 and R1.
                          008B                                                ; R2 and R3 will retain their values.
      50       00'    D0  008B            MOVL    S^#SS$_CONTINUE, R0         ; Continue from the original exception.
               04         008E   DO_RET:  RET                                 ; Exit from handler.
                          008F
                          008F   ;+
                          008F   ; Common setup routine for handlers.  Returns normally if exception was FLTOVF,
                          008F   ; FLTOVF_F, FLTDIV, or FLTDIV_F.  If the exception was anything else, it
                          008F   ; executes a RET, causing an exit from the handler with R0 = 0, which is
                          008F   ; equivalent to SS$_RESIGNAL.  In the case of a normal return (FLTOVF, FLTOVF_F,
                          008F   ; FLTDIV, or FLTDIV_F) it sets up R0:R5 as follows:
                          008F   ;        R0/R3:  0
                          008F   ;        R4:     address of establisher's frame
                          008F   ;        R5:     address of mechanism array
                          008F   ;-
                          008F
               50     7C  008F   SETUP:   CLRQ    R0                          ; Clear first half of default result.
      54    04 AC     7D  0091            MOVQ    CHF$L_SIGARGLST(AP), R4     ; R4 = address of signal array
                          0095                                                ; R5 = address of mechanism array
 0000'8F    04 A4     B1  0095            CMPW    CHF$L_SIG_NAME(R4), #SS$_FLTOVF
                          009B                                                ; Was it a floating overflow trap?
            18        13  009B            BEQL    DO_RSB                      ; Branch if yes.
 0000'8F    04 A4     B1  009D            CMPW    CHF$L_SIG_NAME(R4), #SS$_FLTOVF_F
                          00A3                                                ; Or a floating overflow fault?
            10        13  00A3            BEQL    DO_RSB                      ; Branch if yes.
 0000'8F    04 A4     B1  00A5            CMPW    CHF$L_SIG_NAME(R4), #SS$_FLTDIV
                          00AB                                                ; Or a floating divide by zero trap?
            08        13  00AB            BEQL    DO_RSB                      ; Branch if yes.
 0000'8F    04 A4     B1  00AD            CMPW    CHF$L_SIG_NAME(R4), #SS$_FLTDIV_F
                          00B3                                                ; Or a floating divide by zero fault?
            D9        12  00B3            BNEQ    DO_RET                      ; None of the above: return from handler
                          00B5                                                ; with R0 = 0.
   08 A4    95 AF     9E  00B5   DO_RSB:  MOVAB   B^POWHJX, CHF$L_SIG_NAME+4(R4)
                          00BA                                                ; Change return PC to POWHJX.
      54    04 A5     D0  00BA            MOVL    CHF$L_MCH_FRAME(R5), R4     ; R4 = address of establisher's frame
               52     7C  00BE            CLRQ    R2                          ; Clear second half of default result.
               05         00C0            RSB                                 ; Return.
                          00C1
                          00C1            .END
```

```
BASE                =  00000004
CHF$L_MCH_FRAME     =  00000004
CHF$L_MCH_SAVR0     =  0000000C
CHF$L_SIGARGLST     =  00000004
CHF$L_SIG_NAME      =  00000004
CON_U                  00000087  R      02
DO_RET                 0000008E  R      02
DO_RSB                 000000B5  R      02
DO_SIG                 0000007B  R      02
EXC_HNDLR_OVER         0000006F  R      02
EXC_HNDLR_UNDER        00000060  R      02
EXP                 =  00000014
EXPGTR                 00000024  R      02
MTH$$SIGNAL            ********    X    00
MTH$K_FLOOVEMAT        ********    X    00
MTH$K_FLOUNDMAT        ********    X    00
MTH$K_UNDEXP           ********    X    00
OTS$POWHJ_R3           00000000  RG     02
PARTIAL                00000034  R      02
POWHJX                 0000004D  R      02
PSL$V_FU            =  00000006
SETUP                  0000008F  R      02
SF$L_SAVE_PC        =  00000010
SF$W_SAVE_PSW       =  00000004
SQUAR                  00000028  R      02
SQUAR1                 00000000  R      02
SS$_CONTINUE           ********    X    00
SS$_FLTDIV             ********    X    00
SS$_FLTDIV_F           ********    X    00
SS$_FLTOVF             ********    X    00
SS$_FLTOVF_F           ********    X    00
UNDEFINED              0000004E  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	000000C1 (193.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	38	00:00:00.06	00:00:01.50
Command processing	115	00:00:00.58	00:00:04.93
Pass 1	135	00:00:01.95	00:00:08.08
Symbol table sort	0	00:00:00.10	00:00:00.18
Pass 2	70	00:00:00.89	00:00:02.34
Symbol table output	5	00:00:00.04	00:00:00.07
Psect synopsis output	2	00:00:00.02	00:00:00.07
Cross-reference output	0	00:00:00.00	00:00:00.00

Assembler run totals 367 00:00:03.65 00:00:17.23

The working set limit was 1200 pages.
9201 bytes (18 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.
382 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	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$:OTSPOWHJ/OBJ=OBJ$:OTSPOWHJ MSRC$:MTHJACKET/UPDATE=(ENH$:MTHJACKET)+MSRC
```

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