```
LLL                 IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                 IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                 IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                    III        BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                    III        BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                    III        BBB         BBB  RRR   RRR              TTT        LLL
LLL                    III        BBB         BBB  RRR   RRR              TTT        LLL
LLL                    III        BBB         BBB  RRR   RRR              TTT        LLL
LLL                    III        BBB         BBB  RRR      RRR           TTT        LLL
LLL                    III        BBB         BBB  RRR      RRR           TTT        LLL
LLL                    III        BBB         BBB  RRR      RRR           TTT        LLL
LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

```
  000000    TTTTTTTTTT    SSSSSSSS    CCCCCCCC  VV      VV  TTTTTTTTTT  DDDDDDDD    TTTTTTTTTT
  000000    TTTTTTTTTT    SSSSSSSS    CCCCCCCC  VV      VV  TTTTTTTTTT  DDDDDDDD    TTTTTTTTTT
00      00      TT      SS          CC          VV      VV      TT      DD      DD      TT
00      00      TT      SS          CC          VV      VV      TT      DD      DD      TT
00      00      TT      SS          CC          VV      VV      TT      DD      DD      TT
00      00      TT      SS          CC          VV      VV      TT      DD      DD      TT
00      00      TT        SSSSSS    CC          VV      VV      TT      DD      DD      TT
00      00      TT        SSSSSS    CC          VV      VV      TT      DD      DD      TT
00      00      TT              SS  CC          VV      VV      TT      DD      DD      TT
00      00      TT              SS  CC          VV      VV      TT      DD      DD      TT
00      00      TT              SS  CC            VV  VV        TT      DD      DD      TT      ....
00      00      TT              SS  CC            VV  VV        TT      DD      DD      TT      ....
  000000        TT      SSSSSSSS      CCCCCCCC      VV          TT      DDDDDDDD        TT      ....
  000000        TT      SSSSSSSS      CCCCCCCC      VV          TT      DDDDDDDD        TT      ....

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$$CVTDT
0000     2          .IDENT  /1-017/                 ; File: OTSCVTDT.MAR  Edit: LEB1017
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 ;++
0000    30 ; FACILITY: Language-independent Support Library
0000    31 ;
0000    32 ; ABSTRACT:
0000    33 ;
0000    34 ;       A routine to convert an F or D-floating value to a string of
0000    35 ;       ASCII digits and an exponent.  It is meant to be used as
0000    36 ;       a base for floating point output conversion routines.
0000    37 ;
0000    38 ; ENVIRONMENT: User Mode, AST Reentrant
0000    39 ;
0000    40 ;--
0000    41 ; AUTHOR: Steven B. Lionel, CREATION DATE: 24-May-1979
0000    42 ;
```

```
0000     44          .SBTTL  Edit History
0000     45 ;
0000     46 ; 1-001 - Original.  Numeric conversion algorithm by Tryggve Fossum.
0000     47 ;         SBL 24-May-1979
0000     48 ; 1-002 - Make routine an _R8 so as to conform with OTS$$CVTRT.  SBL 3-Jul-1979
0000     49 ; 1-003 - Add extra longword to stack frame to prevent clobbering of
0000     50 ;         saved info.  SBL 8-Jul-1979
0000     51 ; 1-004 - Don't use R9 or R10 at all.  SBL 11-Jul-1979
0000     52 ; 1-005 - On right-rounding to zero, don't change the sign. SBL 16-Jul-1979
0000     53 ; 1-006 - Fix typo in stack frame setup.  SBL 23-July-1979
0000     54 ; 1-007 - Modify rounding algorithm so that if RT_RND would cause
0000     55 ;         rounding to the right of the number of significant digits,
0000     56 ;         the latter is used instead.  This is at the request of
0000     57 ;         BASIC - the situation can not occur in FORTRAN.  SBL 27-Jul-1979
0000     58 ; 1-008 - Clear 96 bits ahead of fraction instead of 63.  SBL 30-July-1979
0000     59 ; 1-009 - Speed improvements. Clear 64 bits ahead of fraction.  Use
0000     60 ;         register in inner convert loop.  SBL 21-Jan-1980
0000     61 ; 1-010 - Compute number of fraction longwords correctly at INIT_FRACT,
0000     62 ;         to assure accurate low-order digits.  JAW 21-Jul-1981
0000     63 ; 1-011 - Make sure all bits between significand and binary point are
0000     64 ;         cleared when value is an integer, to assure accurate low-order
0000     65 ;         digits.  JAW 26-Jul-1981
0000     66 ; 1-012 - If we find a reserved operand, return zero if it doesn't get
0000     67 ;         replaced by a non-reserved value.  SBL 29-Oct-81
0000     68 ; 1-013 - Add entry for F_floating.  SBL 29-Oct-1982
0000     69 ; 1-014 - Remove CVTFD instruction from OTS$$CVT_F_T_R8.  SBL 27-Apr-1983
0000     70 ; 1-015 - Fix bug introduced by 1-014.  SBL 17-May-1983
0000     71 ; 1-016 - Removed the CVTLP, CVTPS, and SKPC instructions to improve the
0000     72 ;         performance of this routine. Instead, EDIV instructions were
0000     73 ;         used. I also fixed a couple of comments.  JCW 31-OCT-1983
0000     74 ; 1-017 - Move tables after PSECT definition.  LEB 22-Mar-1984
0000     75
```

```
                     0000     77          .SBTTL  DECLARATIONS
                     0000     78 ;
                     0000     79 ; INCLUDE FILES:
                     0000     80 ;
                     0000     81
                     0000     82 ;
                     0000     83 ; EXTERNAL DECLARATIONS:
                     0000     84 ;
                     0000     85          .DSABL  GBL                             ; Prevent undeclared
                     0000     86                                                  ; symbols from being
                     0000     87                                                  ; automatically global.
                     0000     88
                     0000     89 ;
                     0000     90 ; MACROS:
                     0000     91 ;
                     0000     92
                     0000     93 ;
                     0000     94 ; EQUATED SYMBOLS:
                     0000     95 ;
                     0000     96
                     0000     97 ;
                     0000     98 ; PSECT DECLARATIONS:
                     0000     99 ;
                00000000    100          .PSECT _OTS$CODE PIC, USR, CON, REL, LCL, SHR, -
                     0000    101                      EXE, RD, NOWRT, LONG
                     0000    102
                     0000    103 ;
                     0000    104 ; OWN STORAGE:
                     0000    105 ;
                     0000    106 ; CONSTANTS
                     0000    107 ;
                     0000    108
                     0000    109 ASCII_ZEROES:
        30303030 30303030    0000    110          .QUAD   ^X3030303030303030             ; 8 copies of the character 0
                     0008    111
3430 3330 3230 3130 3030     0008    112 TABLE:   .WORD   ^X3030, ^X3130, ^X3230, ^X3330, ^X3430
3930 3830 3730 3630 3530     0012    113          .WORD   ^X3530, ^X3630, ^X3730, ^X3830, ^X3930
3431 3331 3231 3131 3031     001C    114          .WORD   ^X3031, ^X3131, ^X3231, ^X3331, ^X3431
3931 3831 3731 3631 3531     0026    115          .WORD   ^X3531, ^X3631, ^X3731, ^X3831, ^X3931
3432 3332 3232 3132 3032     0030    116          .WORD   ^X3032, ^X3132, ^X3232, ^X3332, ^X3432
3932 3832 3732 3632 3532     003A    117          .WORD   ^X3532, ^X3632, ^X3732, ^X3832, ^X3932
3433 3333 3233 3133 3033     0044    118          .WORD   ^X3033, ^X3133, ^X3233, ^X3333, ^X3433
3933 3833 3733 3633 3533     004E    119          .WORD   ^X3533, ^X3633, ^X3733, ^X3833, ^X3933
3434 3334 3234 3134 3034     0058    120          .WORD   ^X3034, ^X3134, ^X3234, ^X3334, ^X3434
3934 3834 3734 3634 3534     0062    121          .WORD   ^X3534, ^X3634, ^X3734, ^X3834, ^X3934
3435 3335 3235 3135 3035     006C    122          .WORD   ^X3035, ^X3135, ^X3235, ^X3335, ^X3435
3935 3835 3735 3635 3535     0076    123          .WORD   ^X3535, ^X3635, ^X3735, ^X3835, ^X3935
3436 3336 3236 3136 3036     0080    124          .WORD   ^X3036, ^X3136, ^X3236, ^X3336, ^X3436
3936 3836 3736 3636 3536     008A    125          .WORD   ^X3536, ^X3636, ^X3736, ^X3836, ^X3936
3437 3337 3237 3137 3037     0094    126          .WORD   ^X3037, ^X3137, ^X3237, ^X3337, ^X3437
3937 3837 3737 3637 3537     009E    127          .WORD   ^X3537, ^X3637, ^X3737, ^X3837, ^X3937
3438 3338 3238 3138 3038     00A8    128          .WORD   ^X3038, ^X3138, ^X3238, ^X3338, ^X3438
3938 3838 3738 3638 3538     00B2    129          .WORD   ^X3538, ^X3638, ^X3738, ^X3838, ^X3938
3439 3339 3239 3139 3039     00BC    130          .WORD   ^X3039, ^X3139, ^X3239, ^X3339, ^X3439
3939 3839 3739 3639 3539     00C6    131          .WORD   ^X3539, ^X3639, ^X3739, ^X3839, ^X3939
                             00D0    132
                             00D0    133 ; Stack frame offsets from R7
```

```
                00D0    134 ;; Common frame for kernel convert routines
FFFFFFF8        00D0    135         PACKED = -8                     ; Temp for packed representation
FFFFFFF4        00D0    136         FLAGS = PACKED - 4              ; Flags for outer and inner routines
FFFFFFF0        00D0    137         SIG_DIGITS = FLAGS - 4          ; Significant digits
FFFFFFEC        00D0    138         STRING_ADDR = SIG_DIGITS - 4    ; Address of temp string
FFFFFFE8        00D0    139         SIGN = STRING_ADDR - 4          ; Sign
FFFFFFE4        00D0    140         DEC_EXP = SIGN - 4              ; Decimal exponent
FFFFFFE0        00D0    141         OFFSET = DEC_EXP - 4            ; Offset
FFFFFFDC        00D0    142         RT_RND = OFFSET - 4             ; Right round point
FFFFFFDC        00D0    143         COMMON_FRAME = RT_RND           ; Common frame size
                00D0    144
                00D0    145 ;+
                00D0    146 ; Inner routine frame pointed to by R8 during conversion
                00D0    147 ;-
FFFFFFF0        00D0    148         INT_HI = -16                    ; Highest integer part
FFFFFFE4        00D0    149         BIN_PT = INT_HI - 12            ; Binary point
FFFFFFC8        00D0    150         FRACT_LIM = BIN_PT - 28         ; Lowest fraction bits
FFFFFFB4        00D0    151         DIGITS = FRACT_LIM - 20         ; Digits radix 10**9
FFFFFFB0        00D0    152         BIN_EXP = DIGITS - 4            ; Saved binary exponent
FFFFFFAC        00D0    153         LONG_COUNT = BIN_EXP - 4        ; Longword count
FFFFFFA8        00D0    154         TEMP = LONG_COUNT - 4           ; Temporary
FFFFFFA8        00D0    155         LOCAL_FRAME = TEMP              ; Local frame size
                00D0    156
                00D0    157
```

```
                     00D0   159          .SBTTL  OTS$$CVT_D_T  - Convert D floating to text
                     00D0   160 ;++
                     00D0   161 ; FUNCTIONAL DESCRIPTION:
                     00D0   162 ;
                     00D0   163 ;       This routine converts a D-floating point value to a string
                     00D0   164 ;       of ASCII digits.  It is intended to form the base of a
                     00D0   165 ;       language's floating point output conversion routine.
                     00D0   166 ;
                     00D0   167 ;       OTS$$CVT_F_T_R8 converts F_floating.
                     00D0   168 ;
                     00D0   169 ; CALLING SEQUENCE:
                     00D0   170 ;
                     00D0   171 ;       MOVAB   common_frame, R1        ; See common_frame definition above
                     00D0   172 ;       MOVL    string_length, STRING_LEN(R1)
                     00D0   173 ;       MOVL    string_address, STRING_ADDR(R1)
                     00D0   174 ;       MOVL    sig_digits, SIG_DIGITS(R1)
                     00D0   175 ;       MOVL    user_flags, FLAGS(R1)
                     00D0   176 ;       MOVL    rt_round, RT_RND(R1)    ;  Optional
                     00D0   177 ;       MOVAB   value, R0
                     00D0   178 ;       JSB     OTS$$CVT_D_T_R8 or OTS$$CVT_F_T_R8
                     00D0   179 ;       ; outputs are:
                     00D0   180 ;       ;       R1 = unchanged
                     00D0   181 ;       ;       OFFSET(R1) - offset
                     00D0   182 ;       ;       DEC_EXP(R1) - decimal exponent
                     00D0   183 ;       ;       SIGN(R1) - sign
                     00D0   184 ;
                     00D0   185 ; INPUT PARAMETERS:
                     00D0   186 ;
                     00D0   187 ;       VALUE                           ; F or D-floating value to be converted
                     00D0   188 ;       SIG_DIGITS(R1)                  ; Number of significant digits to
                     00D0   189 ;                                       ; generate.  If neither V_TRUNCATE
                     00D0   190 ;                                       ; or V_ROUND_RIGHT is set, the
                     00D0   191 ;                                       ; value will be rounded to this
                     00D0   192 ;                                       ; many digits.
                     00D0   193 ;       FLAGS(R1)                       ; Caller supplied flags:
00000018             00D0   194 ;           V_TRUNCATE = 24             ; Truncate, don't round.
00000019             00D0   195 ;           V_ROUND_RIGHT = 25          ; Round "rt_round" digits to
                     00D0   196 ;                                       ; right of decimal point.
                     00D0   197 ;       RT_RND(R1)                      ; Number of places to the right
                     00D0   198 ;                                       ; of the decimal point to round
                     00D0   199 ;                                       ; after.  Ignored if V_ROUND_RIGHT
                     00D0   200 ;                                       ; is clear.  The rounding takes
                     00D0   201 ;                                       ; place after the specified number
                     00D0   202 ;                                       ; of significant digits if that
                     00D0   203 ;                                       ; would be farther to the left.
                     00D0   204 ;
                     00D0   205 ; IMPLICIT INPUTS:
                     00D0   206 ;
                     00D0   207 ;       NONE
                     00D0   208 ;
                     00D0   209 ; OUTPUT PARAMETERS:
                     00D0   210 ;
                     00D0   211 ;       out_string                      ; String with result.  It will
                     00D0   212 ;                                       ; Not have valid digits after the
                     00D0   213 ;                                       ; requested number of significant
                     00D0   214 ;                                       ; digits.
                     00D0   215 ;                                       ; The length MUST be at least:
```

```
00D0   216                                             ; (9*INT((sig_digits+17)/9))+2
00D0   217 ;       offset                              ; The offset into out_string at
00D0   218                                             ; which the first significant digit
00D0   219                                             ; may be found.  It is guaranteed
00D0   220                                             ; to be between 0 and 9.
00D0   221 ;       exponent                            ; The signed decimal exponent of
00D0   222                                             ; the value, assuming a radix point
00D0   223                                             ; immediately to the left of the
00D0   224                                             ; most significant digit.
00D0   225 ;       sign                                ; -1 if the value is negative
00D0   226                                             ; 0 if the value is zero
00D0   227                                             ; 1 if the value is positive
00D0   228 ;
00D0   229 ; IMPLICIT OUTPUTS:
00D0   230 ;
00D0   231 ;       NONE
00D0   232 ;
00D0   233 ; SIDE EFFECTS:
00D0   234 ;
00D0   235 ;       Alters registers R0 through R8.
00D0   236 ;
00D0   237 ;       SS$_ROPRAND     - If the value is a reserved operand
00D0   238 ;       SS$_ACCVIO      , or other nasty errors if the length of
00D0   239 ;                         out_string is not enough (see formula above).
00D0   240 ;                         This routine does not check the length, it
00D0   241 ;                         is up to the caller to insure the correct
00D0   242 ;                         length is present.
00D0   243 ;
00D0   244 ;--
00D0   245
```

```
                                00D0   247          .SBTTL  OTS$$CVT_D_T_R8
                                00D0   248
                                00D0   249 ;+
                                00D0   250 ; JSB entry point
                                00D0   251 ;-
                                00D0   252
                                00D0   253 OTS$$CVT_F_T_R8::
      57    51    D0            00D0   254          MOVL    R1, R7                  ; Use R7 as common frame pointer
            51    D4            00D3   255          CLRL    R1                      ; Clear high part of value
      50    60    50            00D5   256          MOVF    (R0), R0                ; Fetch and test for zero
            06    11            00D8   257          BRB     COMMON_FD               ; Join common code
                                00DA   258
                                00DA   259 OTS$$CVT_D_T_R8::
      57    51    D0            00DA   260          MOVL    R1, R7                  ; Use R7 as common frame pointer
      50    60    70            00DD   261          MOVD    (R0), R0                ; Fetch and test for zero
                                00E0   262                                          ; and for reserved operand
                                00E0   263 COMMON_FD:
            0E    14            00E0   264          BGTR    VAL_POS                 ; Value is positive
            06    19            00E2   265          BLSS    VAL_NEG                 ; Value is negative
            7D    10            00E4   266          BSBB    ZERO                    ; Value is zero
      51    57    D0            00E6   267          MOVL    R7, R1                  ; Restore R1
                  05            00E9   268          RSB                             ; Return to caller
                                00EA   269
                                00EA   270 VAL_NEG:
   E8 A7    01    CE            00EA   271          MNEGL   #1, SIGN(R7)            ; Set negative sign
            04    11            00EE   272          BRB     EXTRACT                 ; Continue
                                00F0   273 VAL_POS:
   E8 A7    01    D0            00F0   274          MOVL    #1, SIGN(R7)            ; Set positive sign
                                00F4   275
                                00F4   276 EXTRACT:
         58    5E    D0         00F4   277          MOVL    SP, R8                  ; R8 points to local frame
      5E    A8    AE            00F7   278          MOVAB   LOCAL_FRAME(SP), SP     ; Set up local frame
      54    E4    A8            00FB   279          MOVAB   BIN_PT(R8), R4          ; R4 points to binary point
   52    50    08    07         00FF   280          EXTZV   #7, #8, R0, R2          ; Extract exponent
                     5D 13      0104   281          BEQL    ZERO                    ; Still reserved operand; give up
      50    50    10            0106   282          ROTL    #16, R0, R0             ; Make into proper fraction
      51    51    10            010A   283          ROTL    #16, R1, R1
   52    00000080  8F            010E   284          SUBL2   #128, R2                ; Remove bias
                     05 18      0115   285          BGEQ    10$                     ; If value is less than 1,
               F8    A4         0117   286          CLRQ    -8(R4)                  ;  clear some fraction bits
                                011A   287                                          ;  in case value is < 2**-64.
                     02 11      011A   288          BRB     20$
                     64    7C   011C   289 10$:     CLRQ    (R4)                    ; If value is greater than 1,
                                011E   290                                          ;  clear some integer bits
                                011E   291                                          ;  in case value is >= 2**88.
F5 A4    20    52    00    F0   011E   292 20$:     INSV    #0, R2, #32, -11(R4)    ; Create fixed point binary
F9 A4    20    52    51    F0   0124   293          INSV    R1, R2, #32, -7(R4)     ; value with enough surrounding
   50    00800000    8F    C8   012A   294          BISL2   #^X800000, R0
FD A4    18    52    50    F0   0131   295          INSV    R0, R2, #24, -3(R4)     ; zeroes as "guard digits".
   64    20    52    00    F0   0137   296          INSV    #0, R2, #32, (R4)
04 A4    20    52    00    F0   013C   297          INSV    #0, R2, #32, +4(R4)
         B0 A8    52    D0   0142   298          MOVL    R2, BIN_EXP(R8)         ; Save binary exponent
                     2D 15      0146   299          BLEQ    FRACT_ONLY              ; If less than 1...
         56    B4 A8    9E     0148   300          MOVAB   DIGITS(R8), R6          ; R6 points to scratch area
                     57 DD      014C   301          PUSHL   R7                      ; Save R7 so we can use as temp
      55    52    FB    8F     014E   302          ASHL    #-5, R2, R5             ; How many integer longwords?
         AC A8    55    D0   0153   303          MOVL    R5, LONG_COUNT(R8)
```

```
                     07   15  0157   304          BLEQ    ONE_LONG                ; 1 longword
                03   55   91  0159   305          CMPB    R5, #3
                     29   19  015C   306          BLSS    INT_LOOP                ; 2 or 3 longwords
                     1C   11  015E   307          BRB     FOUR_LONG               ; Four longwords
                              0160   308
                              0160   309 ONE_LONG:
                   007B   31  0160   310          BRW     INT_NEXT
                              0163   311
                              0163   312 ZERO:
                51   EC A7   D0  0163   313          MOVL    STRING_ADDR(R7), R1     ; Get string address
61   F0 A7   30   6E   00   2C  0167   314          MOVC5   #0, (SP), #^A/0/, SIG_DIGITS(R7), (R1) ; Zero fill string
                   E0 A7   7C  016E   315          CLRQ    OFFSET(R7)              ; Clear offset and exponent
                   E8 A7   D4  0171   316          CLRL    SIGN(R7)                ; Zero has sign of zero
                           05  0174   317          RSB                             ; Return to caller
                              0175   318
                              0175   319 FRACT_ONLY:
                AC A8   01   CE  0175   320          MNEGL   #1, LONG_COUNT(R8)      ; To note that there is no integer part
                   00E0   31  0179   321          BRW     FORMAT                  ; Go directly to formatter
                              017C   322
```

```
                                017C   324          .SBTTL  Numeric conversion routines
                                017C   325
                                017C   326 ;+
                                017C   327 ; This is the portion which converts the integer part of the value
                                017C   328 ; to 1-5 longwords of radix 10**9.  This is done by repeated division
                                017C   329 ; by 10**9.
                                017C   330 ;-
                                017C   331 FOUR_LONG:
                          51    D4   017C   332          CLRL    R1                          ; High part of dividend
                    54    F0 A8   DE   017E   333          MOVAL   INT_HI(R8), R4              ; Use R4 as address pointer
                       50    64   D0   0182   334          MOVL    (R4),R0                     ; Low part of dividend
                             14   11   0185   335          BRB     INT_DIV
                                0187   336 INT_LOOP:
                 54    E4 A845   DE   0187   337          MOVAL   BIN_PT(R8)[R5], R4          ; Get address pointer
                       04 A4   D5   018C   338          TSTL    4(R4)                       ; Are we missing some bits?
                             07   13   018F   339          BEQL    10$                         ; No if zero
                             55   D6   0191   340          INCL    R5                          ; Back up one longword
                          AC A8   D6   0193   341          INCL    LONG_COUNT(R8)              ; Bump longword counter
                             EF   11   0196   342          BRB     INT_LOOP                    ; And try again
                       50    64   7D   0198   343 10$:     MOVQ    (R4), R0                    ; Get first quadword of dividend
                                019B   344 INT_DIV:
 51    64    50  3B9ACA00 8F   7B   019B   345          EDIV    #^D1000000000,R0,(R4),R1
                       04 A4   D4   01A4   346          CLRL    4(R4)                       ; Since this is really a
                                01A7   347                                                  ; quadword quotient, zero the
                                01A7   348                                                  ; higher longword.
                          57    55   D0   01A7   349          MOVL    R5, R7                      ; R7 is inner loop counter
       53    51  1DCD6500 8F   C3   01AA   350 30$:     SUBL3   #^D500000000,R1,R3          ; Is this dividend too large ?
                             15   19   01B2   351          BLSS    40$                         ; Skip adjustment if not
                          52    74   D0   01B4   352          MOVL    -(R4), R2                   ; Low part of dividend
 51    64    52  3B9ACA00 8F   7B   01B7   353          EDIV    #^D1000000000,R2,(R4),R1
                                01C0   354                                                  ; Divide by 10**9
          64  80000000 8F   C8   01C0   355          BISL    #^X80000000,(R4)            ; Set high bit
                             0C   11   01C7   356          BRB     60$
                          50    74   D0   01C9   357 40$:     MOVL    -(R4), R0                   ; Get low part of dividend
    51    64    50  3B9ACA00 8F   7B   01CC   358          EDIV    #^D1000000000,R0,(R4),R1
                                01D5   359                                                  ; Divide and store result in (R4)
                       D2 57   F5   01D5   360 60$:     SOBGTR  R7, 30$                     ; Loop back
                    86    51   D0   01D8   361          MOVL    R1, (R6)+                   ; Store result on stack
                       A9 55   F5   01DB   362          SOBGTR  R5, INT_LOOP                ; loop back if not done
                                01DE   363 INT_NEXT:
                 52    AC A8   D0   01DE   364          MOVL    LONG_COUNT(R8), R2
                 50    E4 A8   7C   01E2   365          MOVQ    BIN_PT(R8),R0               ; Low part of next dividend
 86    04 A6    50  3B9ACA00 8F   7B   01E6   366          EDIV    #^D1000000000,R0,4(R6),(R6)+
                             02   13   01F0   367          BEQL    10$                         ; Branch if high longword is 0
                             52   D6   01F2   368          INCL    R2                          ; Convert one more longword
                    B4 A842   D5   01F4   369 10$:     TSTL    DIGITS(R8)[R2]              ; Find first non-zero longword
                             04   12   01F8   370          BNEQ    20$                         ; Found.  Go format them.
                             52   D7   01FA   371          DECL    R2                          ; Not found.  Try next one.
                             F6   11   01FC   372          BRB     10$
                 AC A8       52   D0   01FE   373 20$:     MOVL    R2, LONG_COUNT(R8)          ; Save longword count
                       57    8E   D0   0202   374          MOVL    (SP)+, R7                   ; Restore R7 from where saved
                             55   11   0205   375          BRB     FORMAT
                                0207   376
                                0207   377 ;+
                                0207   378 ; This routine initializes the pointer for getting fraction digits.
                                0207   379 ; The number of fraction longwords is calculated and is stored in
                                0207   380 ; LONG_COUNT(R8) for future calls.
```

```
                     0207   381 ;-
                     0207   382 INIT_FRACT:
 50   B0 A8  00000057 8F   C3  0207   383         SUBL3   #<56+32-1>, BIN_EXP(R8), R0
       AC A8    50    20   C7  0210   384         DIVL3   #32, R0, LONG_COUNT(R8)
                     0215   385
                     0215   386 ;+
                     0215   387 ; This routine gets the next nine fraction digits.  It is smart
                     0215   388 ; enough not to do EMULs on zero values.
                     0215   389 ;-
                     0215   390 GET_FRACT:
              51     D4   0215   391         CLRL    R1                      ; Result is initially zero
        52    AC A8   D0   0217   392         MOVL    LONG_COUNT(R8), R2      ; Get number of fraction longwords
              1A     18   021B   393         BGEQ    30$                     ; If not negative, return
     53   E4 A842   DE   021D   394 5$:      MOVAL   BIN_PT(R8)[R2], R3      ; Get address of lowest longword
        50   63     D0   0222   395 10$:     MOVL    (R3), R0                ; Get the longword
              11     15   0225   396         BLEQ    40$                     ; Beware of overflow on EMUL
 50   51   50  3B9ACA00 8F   7A  0227   397         EMUL    #^D1000000000, R0, R1, R0
        83   50     D0   0230   398         MOVL    R0, (R3)+               ; Store result
             52     D6   0233   399         INCL    R2                      ; 1 less longword
             EB     19   0235   400         BLSS    10$                     ; Loop back if more
                    05   0237   401 30$:     RSB
              18     13   0238   402 40$:     BEQL    60$                     ; Don't multiply a zero
 50   51   50  3B9ACA00 8F   7A  023A   403         EMUL    #^D1000000000, R0, R1, R0
       51   3B9ACA00 8F   C0  0243   404         ADDL2   #^D1000000000, R1       ; To prevent overflow
        83   50     D0   024A   405         MOVL    R0, (R3)+               ; Store result
             52     D6   024D   406         INCL    R2                      ; 1 less longword
             D1     19   024F   407         BLSS    10$                     ; Loop back if more
                    05   0251   408         RSB
        83   51     D0   0252   409 60$:     MOVL    R1, (R3)+               ; Store current product
             51     D4   0255   410         CLRL    R1
             52     D6   0257   411         INCL    R2                      ; 1 less longword
             C7     19   0259   412         BLSS    10$                     ; Loop back if more
                    05   025B   413         RSB
```

```
                                025C   415          .SBTTL  Character formatting routines
                                025C   416 ;+
                                025C   417 ; After all the integer portion of the value has been converted to
                                025C   418 ; longwords and stored, the integer part is then converted to
                                025C   419 ; characters and the fraction part, if any, is converted.
                                025C   420 ;-
                                025C   421 FORMAT:
                     55   EC A7    D0 025C   422          MOVL    STRING_ADDR(R7), R5     ; Get string address
                          B5   30    90 0260   423          MOVB    #^A/0/, (R5)+           ; Set first character to '0'
                56   F0 A7   01    C1 0263   424          ADDL3   #1, SIG_DIGITS(R7), R6  ; Generate at least one extra digit
                     50   AC A8    D0 0268   425          MOVL    LONG_COUNT(R8), R0      ; How many integer longwords?
                             03    18 026C   426          BGEQ    1$
                          00F7     31 026E   427          BRW     NO_INT                  ; If none, skip this part
                          55   09   C0 0271   428 1$:      ADDL2   #9, R5                  ; R5 will store least signif digit
                                         0274   429                                  ;  (lsd) in the high order byte.
                          53   55    D0 0274   430          MOVL    R5, R3                  ; save the old address
                F7 A5     FD85 CF    7D 0277   431          MOVQ    ASCII_ZEROES, -9(R5)    ; Initialize the string to contain 30's
                                         027D   432                                  ;  the 9th byte will be filled below
                   51   B4 A840      D0 027D   433          MOVL    DIGITS(R8)[R0], R1      ; R1/R2 must be a quadword for
                               52    D4 0282   434          CLRL    R2                      ;  the EDIV
          54   51   51   00000064 8F   7B 0284   435          EDIV    #100, R1, R1, R4        ; extract two lsd
                               31    13 028D   436          BEQL    60$
                     75   FD74 CF44   B0 028F   437          MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
          54   51   51   00000064 8F   7B 0295   438          EDIV    #100, R1, R1, R4        ; extract two lsd
                               20    13 029E   439          BEQL    60$
                     75   FD63 CF44   B0 02A0   440          MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
          54   51   51   00000064 8F   7B 02A6   441          EDIV    #100, R1, R1, R4        ; extract two lsd
                               0F    13 02AF   442          BEQL    60$
                     75   FD52 CF44   B0 02B1   443          MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
          54   51   51   00000064 8F   7B 02B7   444          EDIV    #100, R1, R1, R4        ; extract two lsd
                     75   FD43 CF44   B0 02C0   445 60$:      MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
                        75   51   30   81 02C6   446          ADDB3   #^A/0/, R1, -(R5)       ; character rep needed for last number
                                         02CA   447 ;
                                         02CA   448 ; Numbers are stored as characters as follows: low order byte is the most
                                         02CA   449 ; significant digit (character), while the high order byte is the least signif
                                         02CA   450 ; digit (character). The storage took place from the high oder digit to the
                                         02CA   451 ; low order digit. Since we used an EDIV by 100, 0,1, or 2 zeroes may be
                                         02CA   452 ; located at (R5). R0 is to contain the number of nonzero digits (not char 30)
                                         02CA   453 ; between (R3) and (R5). If R1<>0 then R0=9. If R1=0, there is at least one
                                         02CA   454 ; zero at (R5) and possibly another at (R5)-1. For example, 12 --> 323130
                                         02CA   455 ; while 102 --> 3230313030.
                                         02CA   456 ;
               50   53   55   C3 02CA   457          SUBL3   R5, R3, R0
                         51   D5 02CE   458          TSTL    R1
                         0A   12 02D0   459          BNEQ    98$
                         50   D7 02D2   460          DECL    R0                      ; At least on leading 30.
                    30   01   A5 02D4   461          CMPB    1(R5), #^A/0/           ; there still could be 1 more 0
                         02   12 02D8   462          BNEQ    98$                     ; 102 --> 3230313030 by the above
                                         02DA   463                                  ;  we've already seen rightmost 30
                         50   D7 02DA   464          DECL    R0                      ;  if there is another, subt 1.
                                         02DC   465                                  ;  There can be no more consect 0's
          E0 A7    0A   50   C3 02DC   466 98$:      SUBL3   R0, #10, OFFSET(R7)
          51    AC A8   09   C5 02E1   467          MULL3   #9, LONG_COUNT(R8), R1  ; Calculate exponent
          E4 A7    51   50   C1 02E6   468          ADDL3   R0, R1, DEC_EXP(R7)     ; Store exponent
                   55   53   D0 02EB   469          MOVL    R3, R5                  ; Move string pointer up by 9
                   56   50   C2 02EE   470          SUBL2   R0, R6                  ; Decrease # of digits left to produce
                                         02F1   471 OUT_LOOP:
```

```
                          72   15  02F1   472          BLEQ    OUT_ROUND                ; Done if no more sig. digits
                       AC A8   D7  02F3   473          DECL    LONG_COUNT(R8)           ; Decrement longword count
                  50   AC A8   D0  02F6   474          MOVL    LONG_COUNT(R8), R0
                          03   18  02FA   475          BGEQ    1$
                        00F8   31  02FC   476          BRW     OUT_FRACT                ; Do fraction part if time
                     55   09   C0  02FF   477 1$:      ADDL2   #9, R5                   ; R5 will store least signif digit
                                  0302   478                                           ;  (lsd) in the high order byte.
                     53   55   D0  0302   479          MOVL    R5, R3                   ; save the old address
                F7 A5   FCF7 CF   7D  0305   480          MOVQ    ASCII_ZEROES, -9(R5)     ; Initialize the string to contain 30's
                                  030B   481                                           ;  the 9th byte will be filled below
                   51   B4 A840   D0  030B   482          MOVL    DIGITS(R8)[R0], R1       ;
                        0A   51   D1  0310   483          CMPL    R1, #^X0000000A          ; if R1 < 10 you may skip the EDIV
                             44   19  0313   484          BLSS    70$                      ;
                             52   D4  0315   485          CLRL    R2                       ; R1/R2 must be a quadword for the EDIV
54   51   51   00000064 8F   7B  0317   486          EDIV    #100, R1, R1, R4         ; extract two lsd
                             31   13  0320   487          BEQL    60$
                  75   FCE1 CF44   B0  0322   488          MOVW    TABLE[R4], -(R5)         ; load correct char rep of the 2 digits
54   51   51   00000064 8F   7B  0328   489          EDIV    #100, R1, R1, R4         ; extract two lsd
                             20   13  0331   490          BEQL    60$
                  75   FCD0 CF44   B0  0333   491          MOVW    TABLE[R4], -(R5)         ; load correct char rep of the 2 digits
54   51   51   00000064 8F   7B  0339   492          EDIV    #100, R1, R1, R4         ; extract two lsd
                             0F   13  0342   493          BEQL    60$
                  75   FCBF CF44   B0  0344   494          MOVW    TABLE[R4], -(R5)         ; load correct char rep of the 2 digits
54   51   51   00000064 8F   7B  034A   495          EDIV    #100, R1, R1, R4         ; extract two lsd
                  75   FCB0 CF44   B0  0353   496 60$:     MOVW    TABLE[R4], -(R5)         ; load correct char rep of the 2 digits
                     75   51   30  81  0359   497 70$:     ADDB3   #^A/0/, R1, -(R5)        ; character rep needed for last number
                          55   53  D0  035D   498          MOVL    R3, R5                   ; Move string pointer up by 9
                          56   09  C2  0360   499          SUBL2   #9, R6                   ; Adjust # of sig. digits
                               8C  11  0363   500          BRB     OUT_LOOP
                                       0365   501
                                       0365   502 OUT_ROUND:
                                       0365   503 ;        BRB     ROUND
                             0157   31  0365   504          BRW     ROUND
                                       0368   505
                                       0368   506 ;+
                                       0368   507 ; This code is executed if the value is less than 1.
                                       0368   508 ;-
                                       0368   509 NO_INT:
                             FE9C   30  0368   510          BSBW    INIT_FRACT               ; Initialize the pointers
                                       036B   511                                           ; and get first 9 digits.
                          E4 A7   D4  036B   512          CLRL    DEC_EXP(R7)              ; Calculate exponent
                     E4 A7   09   C2  036E   513 10$:     SUBL2   #9, DEC_EXP(R7)          ; Its 9 smaller now
                             51   D5  0372   514          TSTL    R1                       ; Are digits zero?
                             05   12  0374   515          BNEQ    20$
                             FE9C   30  0376   516          BSBW    GET_FRACT                ; Get next 9 digits
                                F3   11  0379   517          BRB     10$                      ; And try again
                          55   09   C0  037B   518 20$:     ADDL2   #9, R5                   ; R5 will store least signif digit
                                       037E   519                                           ;  (lsd) in the high order byte.
                          53   55   D0  037E   520          MOVL    R5, R3                   ; save the old address
                F7 A5   FC7B CF   7D  0381   521          MOVQ    ASCII_ZEROES, -9(R5)     ; Initialize the string to contain 30's
                                       0387   522                                           ;  the 9th byte will be filled below
                        0A   51   D1  0387   523          CMPL    R1, #^X0000000A          ; if R1 < 10 you may skip the EDIV
                             44   19  038A   524          BLSS    70$                      ;
                             52   D4  038C   525          CLRL    R2                       ; R1/R2 must be a quadword for the EDIV
54   51   51   00000064 8F   7B  038E   526          EDIV    #100, R1, R1, R4         ; extract two lsd
                             31   13  0397   527          BEQL    60$
                  75   FC5A CF44   B0  0399   528          MOVW    TABLE[R4], -(R5)         ; load correct char rep of the 2 digits
```

```
                00000064 8F   7B  039F   529          EDIV    #100, R1, R1, R4        ; extract two lsd
                         20   13  03A8   530          BEQL    60$
                   FC59 CF44  B0  03AA   531          MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
                00000064 8F   7B  03B0   532          EDIV    #100, R1, R1, R4        ; extract two lsd
                         0F   13  03B9   533          BEQL    60$
                   FC48 CF44  B0  03BB   534          MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
                00000064 8F   7B  03C1   535          EDIV    #100, R1, R1, R4        ; extract two lsd
                   FC39 CF44  B0  03CA   536 60$:     MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
                      51   30   81  03D0   537 70$:     ADDB3   #^A/0/, R1, -(R5)       ; character rep needed for last number
                                  03D4   538 ;
                                  03D4   539 ; Numbers are stored as characters as follows: low order byte is the most
                                  03D4   540 ; significant digit (character), while the high order byte is the least signif
                                  03D4   541 ; digit (character). The storaye took place from the high oder digit to the
                                  03D4   542 ; low order digit. Since we used an EDIV by 100, 0,1, or 2 zeroes may be
                                  03D4   543 ; located at (R5). R0 is to contain the number of nonzero digits (not char 30)
                                  03D4   544 ; between (R3) and (R5). If R1<>0 then R0=9. If R1=0, there is at least one
                                  03D4   545 ; zero at (R5) and possibly another at (R5)-1. For example, 12 --> 323130
                                  03D4   546 ; while 102 --> 3230313030.
                                  03D4   547 ;
                   55   53   C3   03D4   548          SUBL3   R5, R3, R0
                        51   D5   03D8   549          TSTL    R1
                        0A   12   03DA   550          BNEQ    98$
                        50   D7   03DC   551          DECL    R0                      ; At least on leading 30.
                   30   01 A5   91   03DE   552          CMPB    1(R5), #^A/0/           ; there still could be 1 more 0
                             02   12   03E2   553          BNEQ    98$                     ; 102 --> 3230313030 by the above
                                  03E4   554                                          ;  we've already seen rightmost 30
                        50   D7   03E4   555          DECL    R0                      ;  if there is another, subt 1.
                                  03E6   556                                          ;  There can be no more consect 0's
                0A   50   C3   03E6   557 98$:     SUBL3   R0, #10, OFFSET(R7)
                   E4 A7   50   C0   03EB   558          ADDL2   R0, DEC_EXP(R7)         ; Calculate exponent
                        55   53   D0   03EF   559          MOVL    R3, R5                  ; Move string pointer up by 9
                        56   50   C2   03F2   560          SUBL2   R0, R6                  ; Adjust # of sig. digits
                             62   11   03F5   561          BRB     FRACT_LOOP              ; Get 9 more
                                  03F7   562
                                  03F7   563 ;+
                                  03F7   564 ; This code starts the fraction portion if the integer portion exists.
                                  03F7   565 ;-
                                  03F7   566 OUT_FRACT:
                      FE0D   30   03F7   567          BSBW    INIT_FRACT              ; Initialize and get 9 digits
                   55   09   C0   03FA   568          ADDL2   #9, R5                  ; R5 will store least signif digit
                                  03FD   569                                          ;  (lsd) in the high order byte.
                   53   55   D0   03FD   570          MOVL    R5, R3                  ; save the old address
                F7 A5   FBFC CF   7D   0400   571          MOVQ    ASCII_ZEROES, -9(R5)    ; Initialize the string to contain 30's
                                  0406   572                                          ;  the 9th byte will be filled below
                   0A   51   D1   0406   573          CMPL    R1, #^X0000000A         ; if R1 < 10 you may skip the EDIV
                             44   19   0409   574          BLSS    70$                     ;
                             52   D4   040B   575          CLRL    R2                      ; R1/R2 must be a quadword for the EDIV
54   51   51   00000064 8F   7B  040D   576          EDIV    #100, R1, R1, R4        ; extract two lsd
                         31   13  0416   577          BEQL    60$
                   FBEB CF44  B0  0418   578          MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
54   51   51   00000064 8F   7B  041E   579          EDIV    #100, R1, R1, R4        ; extract two lsd
                         20   13  0427   580          BEQL    60$
                   FBDA CF44  B0  0429   581          MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
54   51   51   00000064 8F   7B  042F   582          EDIV    #100, R1, R1, R4        ; extract two lsd
                         0F   13  0438   583          BEQL    60$
                   FBC9 CF44  B0  043A   584          MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
54   51   51   00000064 8F   7B  0440   585          EDIV    #100, R1, R1, R4        ; extract two lsd
```

```
                75   FBBA CF44    B0  0449   586 60$:    MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
                  75   51   30    81  044F   587 70$:    ADDB3   #^A/0/, R1, -(R5)       ; character rep needed for last number
                       55   53    D0  0453   588          MOVL    R3, R5                  ; Move string pointer up by 9
                       56   09    C2  0456   589          SUBL2   #9, R6                  ; Adjust # of sig. digits
                                    0459   590 FRACT_LOOP:
                            64    15  0459   591          BLEQ    ROUND                   ; If not, finish
                          FDB7    30  045B   592          BSBW    GET_FRACT               ; Get 9 more digits
                       55   09    C0  045E   593          ADDL2   #9,R5                   ; R5 will store least signif digit
                                    0461   594                                          ;  (lsd) in the high order byte.
                       53   55    D0  0461   595          MOVL    R5, R3                  ; save the old address
                F7 A5  FB98 CF    7D  0464   596          MOVQ    ASCII_ZEROES, -9(R5)    ; Initialize the string to contain 30's
                                    046A   597                                          ;  the 9th byte will be filled below
                       0A   51    D1  046A   598          CMPL    R1, #^X0000000A         ; if R1 < 10 you may skip the EDIV
                            44    19  046D   599          BLSS    70$                     ;
                            52    D4  046F   600          CLRL    R2                      ; R1/R2 must be a quadword for the EDIV
    54   51   51   00000064 8F    7B  0471   601          EDIV    #100, R1, R1, R4        ; extract two lsd
                            31    13  047A   602          BEQL    60$
                75   FB87 CF44    B0  047C   603          MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
    54   51   51   00000064 8F    7B  0482   604          EDIV    #100, R1, R1, R4        ; extract two lsd
                            20    13  048B   605          BEQL    60$
                75   FB76 CF44    B0  048D   606          MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
    54   51   51   00000064 8F    7B  0493   607          EDIV    #100, R1, R1, R4        ; extract two lsd
                            0F    13  049C   608          BEQL    60$
                75   FB65 CF44    B0  049E   609          MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
    54   51   51   00000064 8F    7B  04A4   610          EDIV    #100, R1, R1, R4        ; extract two lsd
                  75   FB56 CF44    B0  04AD   611 60$:    MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
                    75   51   30    81  04B3   612 70$:    ADDB3   #^A/0/, R1, -(R5)       ; character rep needed for last number
                         55   53    D0  04B7   613          MOVL    R3, R5                  ; Move string pointer up by 9
                         56   09    C2  04BA   614          SUBL2   #9, R6                  ; Adjust # of sig. digits
                              9A    11  04BD   615          BRB     FRACT_LOOP              ; Loop back for more
                                    04BF   616
                                    04BF   617 ;+
                                    04BF   618 ; This routine rounds the value to the given number of significant
                                    04BF   619 ; digits, unless flag V_TRUNCATE is on.  If so, the value is truncated
                                    04BF   620 ; at the next digit.
                                    04BF   621 ;-
                                    04BF   622 ROUND:
                              56    D7  04BF   623          DECL    R6
                         55   56    C0  04C1   624          ADDL2   R6, R5                  ; Find least significant + 1
                41 F4 A7   18    E0  04C4   625          BBS     #V_TRUNCATE, FLAGS(R7), FINIS   ; Truncate if desired
                17 F4 A7   19    E1  04C9   626          BBC     #V_ROUND_RIGHT, FLAGS(R7), 5$   ; Round to right of dec pt?
50   DC A7    E4 A7    C1  04CE   627          ADDL3   DEC_EXP(R7), RT_RND(R7), R0     ; Yes, find it
                              34    19  04D4   628          BLSS    FINIS                   ; Done if rounds to zero
                   F0 A7    50    D1  04D6   629          CMPL    R0, SIG_DIGITS(R7)      ; Round to right of # sig digits?
                            09    18  04DA   630          BGEQ    5$                      ; Yes, round to significant digits
                   50   E0 A7    C0  04DC   631          ADDL2   OFFSET(R7), R0          ; Finish calculation
          55   EC A7    50    C1  04E0   632          ADDL3   R0, STRING_ADDR(R7), R5 ; Get rounding character address
                   35    65    91  04E5   633 5$:     CMPB    (R5), #^A/5/            ; Round?
                         20    19  04E8   634          BLSS    FINIS                   ; No, just finish
                         50    55  04EA   635          MOVL    R5, R0                  ; Save position
                         39    70  91  04ED   636 10$:    CMPB    -(R0), #^A/9/           ; If this is a 9...
                               05  19  04F0   637          BLSS    20$
                         60    30  90  04F2   638          MOVB    #^A/0/, (R0)            ; Then it becomes a zero
                               F6  11  04F5   639          BRB     10$                     ; And we continue
                               60  96  04F7   640 20$:    INCB    (R0)                    ; Else this is last carry
                   50   EC A7    C2  04F9   641          SUBL2   STRING_ADDR(R7), R0     ; Do we need to change offset
                   E0 A7    50    D1  04FD   642          CMPL    R0, OFFSET(R7)          ; and exponent?
```

```
                     07  18  0501   643          BGEQ    FINIS                   ; No
                E0 A7 50  D0  0503   644          MOVL    R0, OFFSET(R7)          ; Yes, set new offset
                   E4 A7  D6  0507   645          INCL    DEC_EXP(R7)             ; Set new exponent
                          050A   646
                          050A   647 ;+
                          050A   648 ; All done.
                          050A   649 ;-
                          050A   650 FINIS:
                5E     08 BF  C2  050A   651          SUBL2   #LOCAL_FRAME, SP        ; Restore stack pointer
                     57  D0  0511   652          MOVL    R7, R1                  ; Restore common frame pointer
                     05  0514   653          RSB                             ; Return to caller
                         0515   654
                         0515   655          .END
```

```
ASCII_ZEROES        00000000 R      01
BIN_EXP           = FFFFFFB0
BIN_PT            = FFFFFFE4
COMMON_FD           000000E0 R      01
DEC_EXP           = FFFFFFF4
DIGITS            = FFFFFFB4
EXTRACT             000000F4 R      01
FINIS               0000050A R      01
FLAGS             = FFFFFFF4
FORMAT              0000025C R      01
FOUR_LONG           0000017C R      01
FRACT_LIM         = FFFFFFC8
FRACT_LOOP          00000459 R      01
FRACT_ONLY          00000175 R      01
GET_FRACT           00000215 R      01
INIT_FRACT          00000207 R      01
INT_DIV             0000019B R      01
INT_HI            = FFFFFFF0
INT_LOOP            00000187 R      01
INT_NEXT            000001DE R      01
LOCAL_FRAME       = FFFFFFA8
LONG_COUNT        = FFFFFFAC
NO_INT              00000368 R      01
OFFSET            = FFFFFFE0
ONE_LONG            00000160 R      01
OTS$$CVT_D_T_R8     000000DA RG     01
OTS$$CVT_F_T_R8     000000D0 RG     01
OUT_FRACT           000003F7 R      01
OUT_LOOP            000002F1 R      01
OUT_ROUND           00000365 R      01
PACKED            = FFFFFFF8
ROUND               000004BF R      01
RT_RND            = FFFFFFDC
SIGN              = FFFFFFE8
SIG_DIGITS        = FFFFFFF0
STRING_ADDR       = FFFFFFEC
TABLE               00000008 R      01
TEMP              = FFFFFFA8
VAL_NEG             000000EA R      01
VAL_POS             000000F0 R      01
V_ROUND_RIGHT     = 00000019
V_TRUNCATE        = 00000018
ZERO                00000163 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	00000515 (1301.)	01 (1.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.02	00:00:01.62
Command processing	135	00:00:00.30	00:00:02.78
Pass 1	94	00:00:01.28	00:00:07.73
Symbol table sort	0	00:00:00.03	00:00:00.03
Pass 2	130	00:00:00.86	00:00:04.28
Symbol table output	5	00:00:00.03	00:00:00.34
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	401	00:00:02.54	00:00:16.80

The working set limit was 1200 pages.
13641 bytes (27 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 44 non-local and 31 local symbols.
655 source lines were read in Pass 1, producing 10 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$:OTSCVTDT/OBJ=OBJ$:OTSCVTDT MSRC$:OTSCVTDT/UPDATE=(ENH$:OTSCVTDT)
```

OTSCCB
LIS

OTSCCBDAT
LIS

OTSCVTDP
LIS

OTSCNVOUT
LIS

OTSCVTFP
LIS

OTSCVTLT
LIS

LIBVECTR2
LIS

LIBWAIT
LIS

OTSCLOSEF
LIS

OTSCVTHP
LIS

LIBVECTOR
LIS

OTSCVTDT
LIS

LIBVM
LIS

OTSCVTGP
LIS