D 8

```
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
```

```
_$

Sy
--
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI

LI
LI
LI
LI
LI
LI
LI

LI
LI
LI

LI
LI
LI
LI
LI
LI
```

```
**FILE**ID**OTSCVTRT                                    D  6
```

```
  OOOOOO    TTTTTTTTTT    SSSSSSSS    CCCCCCCC  VV      VV  TTTTTTTTTT  RRRRRRRR    TTTTTTTTTT
  OOOOOO    TTTTTTTTTT    SSSSSSSS    CCCCCCCC  VV      VV  TTTTTTTTTT  RRRRRRRR    TTTTTTTTTT
OO      OO      TT      SS          CC          VV      VV      TT      RR      RR      TT
OO      OO      TT      SS          CC          VV      VV      TT      RR      RR      TT
OO      OO      TT      SS          CC          VV      VV      TT      RR      RR      TT
OO      OO      TT      SS          CC          VV      VV      TT      RR      RR      TT
OO      OO      TT        SSSSSS    CC          VV      VV      TT      RRRRRRRR        TT
OO      OO      TT        SSSSSS    CC          VV      VV      TT      RRRRRRRR        TT
OO      OO      TT              SS  CC          VV      VV      TT      RR  RR          TT
OO      OO      TT              SS  CC          VV      VV      TT      RR  RR          TT
OO      OO      TT              SS  CC            VV  VV        TT      RR    RR        TT          ....
OO      OO      TT              SS  CC            VV  VV        TT      RR    RR        TT          ....
  OOOOOO        TT      SSSSSSSS      CCCCCCCC      VV          TT      RR      RR      TT          ....
  OOOOOO        TT      SSSSSSSS      CCCCCCCC      VV          TT      RR      RR      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
```

Table of contents

```
0000          .TITLE  OTS$$CVTRT - Kernel Convert real (G and H) to text
0000          .IDENT  /1-012/               ; File: OTSCVTRT.MAR  Edit: LEB1012
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
000)    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0C00    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000    ;
0000    ;++
0000    ; FACILITY: Language-independent Support Library
0000    ;
0000    ; ABSTRACT:
0000    ;
0000    ;       A routine to convert G and H floating values to a string of
0000    ;       ASCII digits and an exponent.  It is meant to be used as
0000    ;       a base for floating point output conversion routines.
0000    ;
000C    ; ENVIRONMENT: User Mode, AST Reentrant
0000    ;
0000    ;--
0000    ; AUTHOR: Tom Eggers and Steven Lionel, CREATION DATE: 25-Jun-1979
0000    ;
```

```
0000            .SBTTL  Edit History
0000    ;
0000    ; 1-001 - Original.  Algorithm implemented by Tom Eggers.  SBL 25-Jun-1979
0000    ; 1-002 - Remove STRING_LEN from frame.  SBL 11-Jul-1979
0000    ; 1-003 - Keep sign when right rounding to zero.  SBL 16-July-1979
0000
0000    ; 1-004 - When using RT_RND, if rounding would be to the right of the
0000    ;         number of significant digits, use the latter.  SBL 27-July-1979
0000    ; 1-005 - Add CVT_HANDLER for correct processing of reserved operands.
0000    ;         SBL 8-Jan-1980
0000    ; 1-006 - Don't loop if a reserved operand gets replaced by another.  SBL 29-Oct-81
0000    ; 1-007 - Fix bug introduced by 1-006 for G_floating.  SBL 4-Feb-1982
0000    ; 1-008 - Add "#" for a literal, missing for three years!  SBL 12-May-1983
0000    ; 1-009 - Extract the code that reciprocates pointer to an item in the
0000    ;         OTS$$A_CVT_TAB table from OTS$$CVT_MUL and include it in calls
0000    ;         from OTSCVTRT.  This allows OTSCVTTR and ADACVTNL to call
0000    ;         OTS$$CVT_MUL in a simpler fashion.  It also saves a few u-seconds
0000    ;         in these calls.  FM 13-MAY-83
0000    ; 1-010 - Add a JSB routine that returns address of OTS$$A_CVT_TAB table.  This
0000    ;         routine will serve the purpose that modules outside the sharable
0000    ;         image (LIBRTL) that contains this routine will be able to reference
0000    ;         this table.  This routine, named OTS$$RET_A_CVT_TAB_R1, is also
0000    ;         added to LIBRTL's vector.  FM 13-MAY-83
0000    ; 1-011 - Removed the CVTLP and CVTPS instructions to improve the performance
0000    ;         of this routine. Instead, EDIV instructions were used. I also
0000    ;         fixed some comments.   JCW 1-NOV-1983
0000    ; 1-012 - Move tables to position after PSECT declaration.  LEB 22-Mar-1984
0000    ;
```

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

```
          00D0
          00D0
          00D0   ; Stack frame offsets from R7
          00D0   ;; Common frame for kernel convert routines
FFFFFFF8  00D0           PACKED = -8                    ; Temp for packed representation
FFFFFFF4  00D0           FLAGS = PACKED - 4             ; Flags for outer and inner routines
FFFFFFF0  00D0           SIG_DIGITS = FLAGS - 4         ; Significant digits
FFFFFFEC  00D0           STRING_ADDR = SIG_DIGITS - 4   ; Address of temp string
FFFFFFE8  00D0           SIGN = STRING_ADDR - 4         ; Sign
FFFFFFE4  00D0           DEC_EXP = SIGN - 4             ; Decimal exponent
FFFFFFE0  00D0           OFFSET = DEC_EXP - 4           ; Offset
FFFFFFDC  00D0           RT_RND = OFFSET - 4            ; Right round point
FFFFFFDC  00D0           COMMON_FRAME = RT_RND          ; Common frame size
          00D0
          00D0
          00D0                                  ; BINNUM HOLDS THE 4 LONG-WORDS OF
          00D0                                  ; THE BINARY FRACTION. IT IS INITIALIZED
          00D0                                  ; WITH THE ''STRAIGHTENED OUT'' FRACTION
          00D0                                  ; BITS FROM THE H-FLOATING NUMBER.
          00D0                                  ; BINNUM+0<0> IS THE LEAST SIGNIFICANT BIT
          00D0                                  ; BINNUM+12<31> IS THE MOST SIG BIT
00000000  00D0           BINNUM = 0
00000010  00D0           INT = BINNUM + 16      ; INT MUST BE 1ST WORD AFTER THE 4
          00D0                                  ; LONGWORDS OF BINNUM. IT IS USED TO CATCH
          00D0                                  ; THE BINARY FOR THE 9 DECIMAL DIGITS
          00D0                                  ; WHEN BINNUM IS MULTIPLIED BY 10**9.
00000000  00D0           .IF NE, <BINNUM+16-INT>
          00D0                   .ERROR         ; INT MUST FOLLOW THE 4 L-WORDS OF BINNUM
          00D0           .ENDC
          00D0
00000014  00D0           BINEXP = INT + 4       ; THE BINARY EXPONENT. IT IS INITIALIZED
          00D0                                  ; FROM THE H-FLOATING EXPONENT.
00000018  00D0           PRODF_4  = BINEXP + 4  ; A TEMPORARY FOR HELPING WITH THE
          00D0                                  ; 4X4 MULTIPLE PRECISION MULTIPLY.
          00D0                                  ; THIS WORD NEVER GETS ALL
          00D0                                  ; THE APPROPRIATE CROSS-PRODUCTS ADDED IN
          00D0                                  ; AND IS NOT REALLY PART OF THE RESULT.
          00D0                                  ; IT'S HERE BECAUSE 'EMUL'' ALWAYS GIVES
          00D0                                  ; DOUBLE L-WORD PRODUCTS EVEN WHEN THE LOW
          00D0                                  ; WORD ISN'T NEEDED (WANTED).
          00D0
0000001C  00D0           PRODF = PRODF_4 + 4    ; THE 4 LONG-WORDS OF PRODF MUST START
          00D0                                  ; JUST AFTER PRODF_4 (WHICH IS ALWAYS
          00D0                                  ; USED AS PRODF-4).
00000000  00D0           .IF NE, <PRODF_4+4-PRODF>
          00D0                   .ERROR         ; PRODF MUST FOLLOW THE L-WORD OF PRODF_4
          00D0           .ENDC
          00D0
0000002C  00D0           CRY = PRODF + 16       ; USED FOR A ''CARRY SAVE'' MULTIPLY.
          00D0
0000003C  00D0           LOCAL_FRAME = CRY + 16 ; SIZE OF DATA AREA TO ALLOCATE ON STACK
          00D0
          00D0
```

```
                      00D0           .SBTTL  OTS$$CVT_x_T  - Convert G and H to text
                      00D0    ;++
                      00D0    ; FUNCTIONAL DESCRIPTION:
                      00D0    ;
                      00D0    ;       This routine converts a G or H floating point value to a string
                      00D0    ;       of ASCII digits.  It is intended to form the base of a
                      00D0    ;       language's floating point output conversion routine.
                      00D0    ;
                      00D0    ;
                      00D0    ; CALLING SEQUENCE:
                      00D0    ;
                      00D0    ;       MOVAB   common_frame, R1          ; See common_frame definition above
                      00D0    ;       MOVL    string_address, STRING_ADDR(R1)
                      00D0    ;       MOVL    sig_digits, SIG_DIGITS(R1)
                      00D0    ;       MOVL    user_flags, FLAGS(R1)
                      00D0    ;       MOVL    rt_round, RT_RND(R1)      ;  Optional
                      00D0    ;       MOVAB   value, R0
                      00D0    ;       JSB     OTS$$CVT_x_T_R8           ; x is the datatype, G or H
                      00D0    ;       ; outputs are:
                      00D0    ;       ;       OFFSET(R1) - offset
                      00D0    ;       ;       DEC_EXP(R1) - decimal exponent
                      00D0    ;       ;       SIGN(R1) - sign
                      00D0    ;
                      00D0    ; INPUT PARAMETERS:
                      00D0    ;
                      00D0    ;       VALUE                             ; floating value to be converted
                      00D0    ;       SIG_DIGITS(R1)                    ; Number of significant digits to
                      00D0    ;                                         ; generate.  If neither V_TRUNCATE
                      00D0    ;                                         ; or V_ROUND_RIGHT is set, the
                      00D0    ;                                         ; value will be rounded to this
                      00D0    ;                                         ; many digits.
                      00D0    ;       FLAGS(R1)                         ; Caller supplied flags:
             00000018 00D0              V_TRUNCATE = 24           ; Truncate, don't round.
             00000019 00D0              V_ROUND_RIGHT = 25        ; Round "rt_round" digits to
                      00D0                                        ; right of decimal point.
                      00D0    ;       RT_RND(R1)                        ; Number of places to the right
                      00D0    ;                                         ; of the decimal point to round
                      00D0    ;                                         ; after.  Ignored if V_ROUND_RIGHT
                      00D0    ;                                         ; is clear.
                      00D0    ;
                      00D0    ; IMPLICIT INPUTS:
                      00D0    ;
                      00D0    ;       NONE
                      00D0    ;
                      00D0    ; OUTPUT PARAMETERS:
                      00D0    ;
                      00D0    ;       out_string                        ; String with result.  It will
                      00D0    ;                                         ; Not have valid digits after the
                      00D0    ;                                         ; requested number of significant
                      00D0    ;                                         ; digits.
                      00D0    ;                                         ; The length MUST be at least:
                      00D0                                              ; (9*INT((sig_digits+8)/9))+2
                      00D0    ;       offset                            ; The offset into out_string at
                      00D0                                              ; which the first significant digit
                      00D0                                              ; may be found.  It is guaranteed
                      00D0                                              ; to be either 0 or 1.
                      00D0    ;       exponent                          ; The signed decimal exponent of
```

```
0000                                                    ; the value, assuming a radix point
0000                                                    ; immediately to the left of the
0000                                                    ; most significant digit.
0000   ;       sign                                     ; -1 if the value is negative
0000                                                    ; 0 if the value is zero
0000                                                    ; 1 if the value is positive
0000   ;
0000   ; IMPLICIT OUTPUTS:
0000   ;
0000   ;       NONE
0000   ;
0000   ; SIDE EFFECTS:
0000   ;
0000   ;       Alters registers R0 through R8.
0000   ;
0000   ;       SS$_ROPRAND     - If the value is a reserved operand
0000   ;       SS$_ACCVIO      , or other nasty errors if the length of
0000   ;                         out_string is not enough (see formula above).
0000   ;                         This routine does not check the length, it
0000   ;                         is up to the caller to insure the correct
0000   ;                         length is present.
0000   ;
0000   ;--
0000
```

```
                                   00D0          .SBTTL  OTS$$CVT_H_T_R8
                                   00D0
                                   00D0    ;+
                                   00D0    ; JSB entry point
                                   00D0    ;-
                                   00D0
                                   00D0    OTS$$CVT_H_T_R8::
            57   51    D0          00D0            MOVL    R1, R7                    ; Use R7 as base
             E4 A7     D4          00D3            CLRL    DEC_EXP(R7)               ; INIT DECIMAL EXPONENT
                                   00D6    TSTVAL_H:
            51   60    32          00D6            CVTWL   (R0), R1                  ; Test for zero and negative
                 03    12          00D9            BNEQ    10$                       ; Not zero
               032D    31          00DB            BRW     ZERO                      ; Is zero
                 06    19          00DE    10$:    BLSS    NEG_VAL_H                 ; Negative?
         E8 A7   01    D0          00E0            MOVL    #1, SIGN(R7)              ; No, set sign
                 2A    11          00E4            BRB     NOTRES_H                  ; Continue
                                   00E6    NEG_VAL_H:
    51    51   0F   00    EF       00E6            EXTZV   #0, #15, R1, R1           ; Reserved operand?
                 1F    12          00EB            BNEQ    10$                       ; No
            58   6D    D0          00ED            MOVL    (FP), R8                  ; Save handler address
   00000101'EF   00    FB          00F0            CALLS   #0, 5$                    ; Reserved operand
       8000 8F   60    B1          00F7            CMPW    (R0), #^X8000             ; Still reserved?
                 D8    12          00FC            BNEQ    TSTVAL_H                  ; No, try again
               030A    31          00FE            BRW     ZERO                      ; Yes, call it zero and quit
                     0000          0101    5$:     .WORD   ^M<>
        6D    0420'CF  9E          0103            MOVAB   W^CVT_HANDLER, (FP)       ; Enable condition handler
               60 73FD             0108            TSTH    (R0)                      ; Force reserved operand fault
                    04             010B            RET                               ; Continue
                                   010C
         E8 A7   01    CE          010C    10$:    MNEGL   #1, SIGN(R7)              ; Set negative sign
                                   0110    NOTRES_H:
            5E   3C    C2          0110            SUBL2   #LOCAL_FRAME, SP          ; ALLOCATE LOCAL DATA ON STACK
            58   5E    D0          0113            MOVL    SP, R8                    ; SETUP POINTER TO LOCAL DATA AREA
 14 A8   51   00004000 8F  C3      0116            SUBL3   #^X4000, R1, BINEXP(R8) ; REMOVE EXCESS FROM EXPONENT
                                   011F
                                   011F                    ; PICK UP H-FLOATING FRACTION AND STORE AS A LEFT
                                   011F                    ; NORMALIZED UNSIGNED 4-LONGWORD INTEGER WITH THE BINARY
                                   011F                    ; POINT BETWEEN BITS 32 & 31 OF 'BINNUM+12'
                                   011F
      54   02 A0   10    9C        011F            ROTL    #16, 2(R0), R4            ; GET BYTES #5,4,3,2; STORE 3,2,5,4
      53   06 A0   10    9C        0124            ROTL    #16, 6(R0), R3            ; GET 9,8,7,6; STORE 7,6,9,8
      52   0A A0   10    9C        0129            ROTL    #16, 10(R0), R2           ; GET 13,12,11,10; STORE 11,10,13,12
                                   012E
         51   0E A0      3C        012E            MOVZWL  14(R0), R1                ; GET Z,Z,15,14
      51   51      10    9C        0132            ROTL    #16, R1, R1               ; STORE 15,14,Z,Z
                                   0136
                                   0136                    ; DENORMALIZE BY 1 BIT TO INSERT
                                   0136                    ; THE HIDDEN BIT. THIS WILL LEAVE 15 GUARD BITS.
                                   0136
      68    51   20   01   EE      0136            EXTV    #1, #32, R1, BINNUM+0(R8)
   04 A8    52   20   01   EE      013B            EXTV    #1, #32, R2, BINNUM+4(R8)
   08 A8    53   20   01   EE      0141            EXTV    #1, #32, R3, BINNUM+8(R8)
      54    54   1F   01   EF      0147            EXTZV   #1, #31, R4, R4
 0C A8   54   80000000 8F  C9      014C            BISL3   #^X80000000, R4, BINNUM+12(R8)  ; AND SET HIDDEN BIT
               0076        31      0155            BRW     BEGSRC           ; Now convert the value
```

```
                                  0158            .SBTTL  OTS$$CVT_G_T_R8
                                  0158
                                  0158    ;+
                                  0158    ; JSB entry point
                                  0158    ;-
                                  0158
                                  0158    OTS$$CVT_G_T_R8::
                 57   51     D0   0158            MOVL    R1, R7                    ; Use R7 as base
                 E4 A7       D4   015B            CLRL    DEC_EXP(R7)               ; INIT DECIMAL EXPONENT
                                  015E    TSTVAL_G:
          51   60   0C   04  EE   015E            EXTV    #4, #12, (R0), R1         ; Test for zero and negative
                         03  12   0163            BNEQ    10$                       ; Not zero
                       02A3  31   0165            BRW     ZERO                      ; Is zero
                         06  19   0168    10$:    BLSS    NEG_VAL_G                 ; Negative?
                 E8 A7   01  D0   016A            MOVL    #1, SIGN(R7)              ; No, set sign
                         2B  11   016E            BRB     NOTRES_G                  ; Continue
                                  0170    NEG_VAL_G:
          51   51   0B   00  EF   0170            EXTZV   #0, #11, R1, R1           ; Reserved operand?
                         20  12   0175            BNEQ    10$                       ; No
            0000018C'EF  00  FB   0177            CALLS   #0, 5$                    ; Reserved operand
  00000800 8F   60   0C  04  ED   017E            CMPZV   #4, #12, (R0), #^X800     ; Still reserved?
                         D5  12   0187            BNEQ    TSTVAL_G                  ; No, try again
                       027F  31   0189            BRW     ZERO                      ; Still reserved, call it zero
                       0000       018C    5$:     .WORD   ^M<>
            6D     0420'CF   9E   018E            MOVAB   W^CVT_HANDLER, (FP)       ; Enable condition handler
                    60 53FD       0193            TSTG    (R0)                      ; Force reserved operand fault
                         04       0196            RET                               ; Continue
                                  0197
                 E8 A7   01  CE   0197    10$:    MNEGL   #1, SIGN(R7)              ; Set negative sign
                                  019B    NOTRES_G:
                   5E    3C  C2   019B            SUBL2   #LOCAL_FRAME, SP          ; ALLOCATE LOCAL DATA ON STACK
                   58    5E  D0   019E            MOVL    SP, R8                    ; SETUP POINTER TO LOCAL DATA AREA
     14 A8   51    00000400  8F   C3   01A1       SUBL3   #^X400, R1, BINEXP(R8)    ; REMOVE EXCESS FROM EXPONENT
                                  01AA
                                  01AA                    ; PICK UP G-FLOATING FRACTION AND STORE AS A LEFT
                                  01AA                    ; NORMALIZED UNSIGNED 4-LONGWORD INTEGER WITH THE BINARY
                                  01AA                    ; POINT BETWEEN BITS 32 & 31 OF "BINNUM+12"
                                  01AA
               54   60   10  9C   01AA            ROTL    #16, (R0), R4             ; Get high fraction
          53   04 A0     10  9C   01AE            ROTL    #16, 4(R0), R3            ; Get low fraction
                                  01B3
                                  01B3                    ; DENORMALIZE BY 1 BIT TO INSERT
                                  01B3                    ; THE HIDDEN BIT.
                                  01B3
                         68  7C   01B3            CLRQ    BINNUM+0(R8)              ; Clear low order bits
                         52  D4   01B5            CLRL    R2
       08 A8   52   20   15  EE   01B7            EXTV    #21, #32, R2, BINNUM+8(R8)
          54   53   1F   15  EF   01BD            EXTZV   #21, #31, R3, R4
     0C A8   54    80000000  8F   C9   01C2       BISL3   #^X80000000, R4, BINNUM+12(R8)  ; AND SET HIDDEN BIT
                       0000  31   01CB            BRW     BEGSRC          ; Now convert the value
```

```
                               01CE                         ; NOW SEARCH THE POWER-OF-TEN TABLE TO FIND
                               01CE                         ; AN ENTRY CLOSE TO THE VALUE STORED
                               01CE                         ; IN BINEXP & BINNUM. THEN DIVIDE (OR RATHER
                               01CE                         ; MULTIPLY BY THE RECIPROCAL) BINEXP & BINNUM
                               01CE                         ; BY THAT TABLE ENTRY TO GET THE RESULTANT
                               01CE                         ; FRACTION INTO THE RANGE:
                               01CE                         ;       1.0 .GT. (FRACTION * 2** EXPONENT) .GE. 0.1
                               01CE
                               01CE                         ; THE TABLE SEARCH IS BROKEN INTO THREE PIECES: THE
                               01CE                         ; BIG NUMBER EXPONENTIAL SEARCH (STARTING AT BIGEXP),
                               01CE                         ; THE SMALL NUMBER EXPONENTIAL SEARCH (STARTING AT
                               01CE                         ; SMLEXP), AND THE MIDDLE NUMBER SEARCH OF THE LINEAR
                               01CE                         ; PORTION OF THE TABLE (STARTING AT SRCLIN).
                               01CE
       52   000005C7'EF   3E   01CE   BEGSRC: MOVAW   TM16, R2                  ; GET 1ST ADR OF LINEAR TABLE
       14 A8    0010'C2   B1   01D5           CMPW    T_BEXP(R2), BINEXP(R8)    ; COMPARE WITH ENTRY'S BIN EXP
                     11   14   01DB           BGTR    SMLEXP                    ; BRANCH FOR SMALL NUMBERS
       14 A8    0290'C2   B1   01DD           CMPW    <T16-TM16>+T_BEXP(R2), BINEXP(R8)
                               01E3                                             ; COMPARE WITH LAST LINEAR ENTRY
                     49   14   01E3           BGTR    SRCLIN                    ; BRANCH FOR LINEAR SEARCH
                               01E5
                               01E5                         ; THE TWO SEARCHES WHICH FOLLOW (BIGEXP & SMLEXP) FIND
                               01E5                         ; THE TABLE ENTRY CLOSEST TO THE NUMBER STORED IN
                               01E5                         ; BINEXP(R8). THIS TABLE ENTRY IS USED TO DIVIDE (OR
                               01E5                         ; MULTIPLY BY THE RECIPROCAL) BINEXP & BINNUM.
                               01E5
       52   00000847'EF   3E   01E5   BIGEXP: MOVAW   T16, R2                   ; EXPONENTIAL SEARCH FOR BIG NUMBERS
                     07   11   01EC           BRB     BIGEX1
                               01EE
       52   00000527'EF   3E   01EE   SMLEXP: MOVAW   TSMALL, R2                ; EXPONENTIAL SEARCH FOR SMALL NUMBERS
          50    0010'C2   32   01F5   BIGEX1: CVTWL   T_BEXP(R2), R0            ; GET POWER-OF-2 FROM TABLE
       51   50     FF 8F  78   01FA           ASHL    #-1, R0, R1               ; FOR LARGE, CALC: 1.5*ENTRY
                     08   18   01FF           BGEQ    BIGEX2                    ; XFER FOR BIG NUMS (POSITIVE EXPONENT)
       51   51     FF 8F  78   0201           ASHL    #-1, R1, R1               ; FOR SMALL, CALC: .75*ENTRY
                 51  51   CE   0206           MNEGL   R1, R1
                 50  51   C0   0209   BIGEX2: ADDL2   R1, R0                    ; FORM .75*ENTRY OR 1.5*ENTRY
                               020C                                             ; R0 NOW CONTAINS VALUE HALF WAY
                               020C                                             ; BETWEEN THIS AND NEXT ENTRY.
             14 A8   50   B1   020C           CMPW    R0, BINEXP(R8)            ; IS THIS CLOSEST TABLE ENTRY?
                     09   18   0210           BGEQ    BIGEX3                    ; IF YES, XFER
       52   00000014'8F   C0   0212           ADDL2   #<T1-T0>, R2              ; NO, GO LOOK AT NEXT ENTRY
                     DA   11   0219           BRB     BIGEX1
                               021B
                               021B   BIGEX3:
                               021B                         ; FIND THE RECIPROCAL TABLE ENTRY POINTED TO BY R2.
                               021B                         ; R2 CONTAINS THE BASE (T0) PLUS AN "INDEX". THE
                               021B                         ; RECIPROCAL ENTRY HAS AN ADR OF "T0-INDEX" WHICH
                               021B                         ; IS CALCULATED BY 2*T0-(T0+INDEX), OR 2*T0-R2.
                               021B
       51   00000707'EF   DE   021B           MOVAL   T0, R1                    ; GET BASE ADR
                 51  51   C0   0222           ADDL2   R1, R1                    ; 2*BASE
          52     51  52   C3   0225           SUBL3   R2, R1, R2                ; GET ADR OF RECIPROCAL ENTRY
                               0229
                   0256   30   0229           BSBW    RMUL                      ; YES, GO MUL BY RECIPROCAL
                     A0   11   022C           BRB     BEGSRC                    ;   AND GO TRY AGAIN
```

```
                                022E  SRCLIN:
                                022E                               ; THE CONVERSION WILL TAKE PLACE FROM THE LINEAR (IN
                                022E                               ; POWERS OF TEN) PART OF THE TABLE.
                                022E                               ; The DECIMAL_EXPONENT = 1 + LOG10(2) * (BIN_EXP - 1). Use this
                                022E                               ; approximation to get the 1st probe into the table.
                                022E                               ; This approx may be 1 small, but no more than that.
                                022E                               ; The approx has been tested exhaustively over the
                                022E                               ; range -106 .LE. BIN_EXP .LE. +108 and always works
                                022E                               ; except for BIN_EXP=1 which has a special code hack.
                                022E
 51    14 A8     01      C3     022E              SUBL3   #1, BINEXP(R8), R1          ; GET (BINEXP - 1)
                 0E      13     0233              BEQL    SRCL1                       ; IF BINEXP=+1, RETURN 0 (hack)
 51    000004D1  8F      C4     0235              MULL2   #1233, R1                   ; 1233 = 4096 * LOG10(2)
 51    51     F4 8F      78     023C              ASHL    #-12, R1, R1                ; REMOVE THE 4096 FACTOR
                 51      D6     0241              INCL    R1                          ; FINAL +1
                                0243
 51    00000014'8F       C4     0243  SRCL1:      MULL2   #<T1-T0>, R1                ; MUL BY SIZE OF TABLE ENTRY
             52  51      C0     024A              ADDL2   R1, R2                      ; GET INDEX*size+TM16
 52    00000140'8F       C0     024D              ADDL2   #<T0-TM16>, R2              ; GET INDEX*size+T0
                                0254
  14 A8     0010'C2      B1     0254              CMPW    T_BEXP(R2), BINEXP(R8)      ; COMPARE EXPONENTS
                 27      14     025A              BGTR    FOUND                       ; XFER IF ENTRY .GT. BINNUM
                                025C
                                025C                          ; THE NEXT INSTRUCTION IS COMMENTED OUT. IT CAN NOT XFER.
                                025C  ;           BLSS    SMALL                       ; XFER IF ENTRY TOO SMALL
     0C A8     0C A2     D1     025C              CMPL    12(R2), BINNUM+12(R8)       ; COMPARE HIGH-ORDER FRACTION
                 20      1A     0261              BGTRU   FOUND
                 17      1F     0263              BLSSU   SMALL
     08 A8     08 A2     D1     0265              CMPL    8(R2), BINNUM+8(R8)
                 17      1A     026A              BGTRU   FOUND
                 0E      1F     026C              BLSSU   SMALL
     04 A8     04 A2     D1     026E              CMPL    4(R2), BINNUM+4(R8)
                 0E      1A     0273              BGTRU   FOUND
                 05      1F     0275              BLSSU   SMALL
            68   62      D1     0277              CMPL    0(R2), BINNUM+0(R8)         ; COMPARE LOW-ORDER FRACTION
                 07      1A     027A              BGTRU   FOUND
 52    00000014'8F       C0     027C  SMALL:      ADDL2   #<T1-T0>, R2                ; ADVANCE TO NEXT TABLE ITEM
                                0283
                                0283                               ; FINAL CHECK FOR DEBUGGING. REMOVE THESE NEXT THREE
                                0283                               ; INSTRUCTIONS AFTER ALL THE TESTING IS DONE. (OR
                                0283                               ; LEAVE THEM IN-- THEY DON'T REALLY HURT.)
                                0283
                                0283  ;           CMPW    T_BEXP(R2), BINEXP(R8)      ; FINAL SIZE CHECK
                                0283  ;           BGTR    FOUND
                                0283  ;           HALT                                ; BAD INDEX FORMULA
                                0283
 50    00000707'EF       3E     0283  FOUND:      MOVAW   T0, R0                      ; GET TABLE BASE ADR
            50   52      D1     028A              CMPL    R2, R0
                 0A      13     028D              BEQL    MULDUN                      ; IF 0, DON'T MUL BY 1.0
                                028F
                                028F                               ; FIND THE RECIPROCAL TABLE ENTRY POINTED TO BY R2.
                                028F                               ; R2 CONTAINS THE BASE (T0) PLUS AN "INDEX". THE
                                028F                               ; RECIPROCAL ENTRY HAS AN ADR OF "T0-INDEX" WHICH
                                028F                               ; IS CALCULATED BY 2*T0-(T0+INDEX), OR 2*T0-R2.
                                028F
            50   50      C0     028F              ADDL2   R0, R0                      ; 2*BASE
     52     50   52      C3     0292              SUBL3   R2, R0, R2                  ; GET ADR OF RECIPROCAL ENTRY
```

```
                   0296   486
      01F9   30    0296   487          BSBW    RMUL                       ; AND MULTIPLY BY RECIPROCAL
```

```
                         0299 MULDUN:            ; BINEXP SHOULD NOW CONTAIN 0, -1, -2, OR -3.
                         0299                    ; SHIFT BINNUM RIGHT BY THAT NUMBER OF PLACES
                         0299                    ; IN ORDER TO REDUCE BINEXP TO ZERO, THUS
                         0299                    ; FINALLY FINISHING WITH THE BINARY EXPONENT
                         0299                    ; ROUND USING THE BITS SHIFTED OFF TO THE RIGHT
                         0299
        50    14 A8   CE 0299         MNEGL     BINEXP(R8), R0                  ; FIND BIT # FROM BINEXP
                 35   13 029D         BEQL      GETDIG                          ; IF 0, SKIP RIGHT SHIFT
                         029F
        51    50   01   C3 029F       SUBL3     #1, R0, R1                      ; GET POS OF 1ST DISCARDED BIT
     51    68   01   51   EF 02A3     EXTZV     R1, #1, BINNUM+0(R8), R1        ; GET 1ST DISCARDE BIT
                         02A8
     68   68   20   50   EE 02A8      EXTV      R0, #32, BINNUM+0(R8), BINNUM+0(R8)
04 A8  04 A8   20   50   EE 02AD      EXTV      R0, #32, BINNUM+4(R8), BINNUM+4(R8)
08 A8  08 A8   20   50   EE 02B4      EXTV      R0, #32, BINNUM+8(R8), BINNUM+8(R8)
                 10 A8   D4 02BB      CLRL      BINNUM+16(R8)                   ; NEXT EXTV WILL GET 0'S HERE
0C A8  0C A8   20   50   EE 02BE      EXTV      R0, #32, BINNUM+12(R8), BINNUM+12(R8)
                         02C5 ;       CLRL      BINEXP(R8)                      ; BINEXP NOW REDUCED TO ZERO
                         02C5
           68   51   C0 02C5          ADDL2     R1, BINNUM+0(R8)                ; ROUND WITH 1ST DISCARDED BIT
        04 A8   00   D8 02C8          ADWC      #0, BINNUM+4(R8)
        08 A8   00   D8 02CC          ADWC      #0, BINNUM+8(R8)
        0C A8   00   D8 02D0          ADWC      #0, BINNUM+12(R8)
                         02D4
        55    EC A7   D0 02D4 GETDIG: MOVL      STRING_ADDR(R7), R5             ; GET ADR FOR DIGIT STRING
     56   F0 A7   01   C1 02D8        ADDL3     #1, SIG_DIGITS(R7), R6          ; Number of digits wanted
        E0 A7   01   D0 02DD          MOVL      #1, OFFSET(R7)                  ; Initial offset
           85   30   90 02E1          MOVB      #^A/0/, (R5)+                   ; Start out with a zero
                         02E4
                         02E4                    ; NOW MUL THE BINNUM FRACTION BY 10**9 IN ORDER TO
                         02E4                    ; FORCE 9 DIGITS TO THE LEFT OF THE DECIMAL POINT.
                         02E4                    ; THEN CONVERT THAT 9 DIGIT BINARY INTEGER TO A
                         02E4                    ; STRING FOR OUTPUT IN THE FINAL ANSWER. REPEAT
                         02E4                    ; THE PROCESS UNTIL ENOUGH DIGITS ARE OUTPUT.
                         02E4
                         02E4 .MACRO IMUL2 I, R, ?L
                         02E4         EMUL      I, R, #0, R0
                         02E4         TSTL      R
                         02E4         BGEQ      L
                         02E4         ADDL2     I, R1
                         02E4 L:      MOVL      R0, R
                         02E4         ADDL2     R1, 4+R
                         02E4 .ENDM IMUL2
                         02E4
             10 A8   D4 02E4 DIGLUP:  CLRL      INT(R8)                 ; CLEAR FOR DIGITS LEFT OF BIN POINT
                         02E7
                         02E7                    ; MULTIPLY 4-LONG-WORDS BY 10**9, PROPOGATING CARRIES
                         02E7                    ; ACROSS THE LONG-WORD BOUNDARIES.
                         02E7
     52   3B9ACA00 8F   D0 02E7       MOVL      #1000000000, R2                 ; SETUP 10**9
                         02EE
                         02EE         IMUL2     R2, BINNUM+12(R8)
                         0304         IMUL2     R2, BINNUM+8(R8)
        10 A8   00   D8 031A          ADWC      #0, INT(R8)
                         031E         IMUL2     R2, BINNUM+4(R8)
        0C A8   00   D8 0334          ADWC      #0, BINNUM+12(R8)
        10 A8   00   D8 0338          ADWC      #0, INT(R8)
```

```
                                    033C  546         IMUL2   R2, BINNUM+0(R8)
                 08 A8    00    D8  034F  547         ADWC    #0, BINNUM+8(R8)
                 0C A8    00    D8  0353  548         ADWC    #0, BINNUM+12(R8)
                 10 A8    00    D8  0357  549         ADWC    #0, INT(R8)
                                    035B  550
                                    035B  551                 ; CONVERT BINARY NUM NOW LEFT OF DECIMAL POINT INTO
                                    035B  552                 ; 9 PACKED DIGITS.
                                    035B  553
                    55    09    C0  035B  554         ADDL2   #9, R5                  ; R5 will store least signif digit
                                    035E  555                                         ;  (lsd) in the high order byte.
                    53    55    D0  035E  556         MOVL    R5, R3                  ; save the old address
              F7 A5  FC9B CF    7D  0361  557         MOVQ    ASCII_ZEROES, -9(R5)    ; Initialize the string to contain 30's
                                    0367  558                                         ;  the 9th byte will be filled below
                  51   10 A8    D0  0367  559         MOVL    INT(R8), R1             ; R1/R2 must be a quadword for
                          52    D4  036B  560         CLRL    R2                      ;  the EDIV
  54   51   51   00000064 8F    7B  036D  561         EDIV    #100, R1, R1, R4        ; extract two lsd
                          33    13  0376  562         BEQL    10$
              75   FC8B CF44    B0  0378  563         MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
  54   51   51   00000064 8F    7B  037E  564         EDIV    #100, R1, R1, R4        ; extract two lsd
                          22    13  0387  565         BEQL    10$
              75   FC7A CF44    B0  0389  566         MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
  54   51   51   00000064 8F    7B  038F  567         EDIV    #100, R1, R1, R4        ; extract two lsd
                          11    13  0398  568         BEQL    10$
              75   FC69 CF44    B0  039A  569         MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
  54   51   51   00000064 8F    7B  03A0  570         EDIV    #100, R1, R1, R4        ; extract two lsd
                          00    13  03A9  571         BEQL    10$
              75   FC58 CF44    B0  03AB  572 10$:    MOVW    TABLE[R4], -(R5)        ; load correct char rep of the 2 digits
              75    51    30    81  03B1  573         ADDB3   #^A/0/, R1, -(R5)       ; character rep needed for last number
                    55    53    D0  03B5  574         MOVL    R3, R5                  ; Move string pointer up by 9, ie,
                                    03B8  575                                         ;    ADVANCE OUTPUT STRING ADDRESS
                    56    09    C2  03B8  576         SUBL2   #9, R6                  ; 9 more digits
                          03    15  03BB  577         BLEQ    ROUND                   ; Loop for more?
                        FF24    31  03BD  578         BRW     DIGLUP                  ; Yes
                                    03C0  579
                                    03C0  580 ;+
                                    03C0  581 ; This routine rounds the value to the given number of significant
                                    03C0  582 ; digits, unless flag V_TRUNCATE is on.  If so, the value is truncated
                                    03C0  583 ; at the next digit.
                                    03C0  584 ;-
                                    03C0  585 ROUND:
                          56    D7  03C0  586         DECL    R6
                    55    56    C0  03C2  587         ADDL2   R6, R5                  ; Find least significant + 1
              3A F4 A7    18    E0  03C5  588         BBS     #V_TRUNCATE, FLAGS(R7), FINIS   ; Truncate if desired
              15 F4 A7    19    E1  03CA  589         BBC     #V_ROUND_RIGHT, FLAGS(R7), 5$   ; Round to right of dec pt?
          51   DC A7   E4 A7    C1  03CF  590         ADDL3   DEC_EXP(R7), RT_RND(R7), R1     ; Yes, find it
                          2D    19  03D5  591         BLSS    FINIS                   ; Exit if round to zero
                 F0 A7    51    D1  03D7  592         CMPL    R1, SIG_DIGITS(R7)      ; Round to right of # sig digits?
                          07    18  03DB  593         BGEQ    5$                      ; Yes, use number of significant
                                    03DD  594                                         ; digits instead.
                          51    D6  03DD  595         INCL    R1                      ; Finish calculation
            55   EC A7    51    C1  03DF  596         ADDL3   R1, STRING_ADDR(R7), R5 ; Get rounding character address
                    35    65    91  03E4  597 5$:     CMPB    (R5), #^A/5/            ; Round?
                          1B    19  03E7  598         BLSS    FINIS                   ; No, just finish
                    50    55    D0  03E9  599         MOVL    R5, R0                  ; Save position
                    39    70    91  03EC  600 10$:    CMPB    -(R0), #^A/9/           ; If this is a 9...
                          05    19  03EF  601         BLSS    20$
                    60    30    90  03F1  602         MOVB    #^A/0/, (R0)            ; Then it becomes a zero
```

```
                             F6   11  03F4          BRB     10$                         ; And we continue
                             60   96  03F6  20$:    INCB    (R0)                        ; Else this is last carry
                  50     EC  A7   C2  03F8          SUBL2   STRING_ADDR(R7), R0         ; Do we need to change offset
                             06   14  03FC          BGTR    FINIS                       ; No
                         E0  A7   D4  03FE          CLRL    OFFSET(R7)                  ; Yes, set new offset
                         E4  A7   D6  0401          INCL    DEC_EXP(R7)                 ; Set new exponent
                                      0404
                                      0404  ;+
                                      0404  ; All done.
                                      0404  ;-
                                      0404  FINIS:
                        5E   3C   C0  0404          ADDL2   #LOCAL_FRAME, SP                    ; Restore stack pointer
                        51   57   D0  0407          MOVL    R7, R1                      ; Restore frame pointer
                                  05  040A          RSB                                 ; Return to caller
                                      040B
                                      040B  ZERO:
                       51  EC A7  D0  040B          MOVL    STRING_ADDR(R7), R1         ; Get string address
    61    F0 A7    30  6E      00  2C  040F          MOVC5   #0, (SP), #^A/0/, SIG_DIGITS(R7), (R1) ; Zero fill string
                         E0  A7   7C  0416          CLRQ    OFFSET(R7)                  ; Clear offset and exponent
                         E8  A7   D4  0419          CLRL    SIGN(R7)                    ; Clear sign
                        51   57   D0  041C          MOVL    R7, R1                      ; Restore frame pointer
                                  05  041F          RSB                                 ; Return to caller
```

```
                                       0420          .SBTTL  CVT_HANDLER - Local condition handler
                                       0420
                                       0420  ;++
                                       0420  ;
                                       0420  ;       CVT_HANDLER allows OTS$$CVT_G_T_R8 and OTS$$CVT_H_T_R8 to detect
                                       0420  ;       reserved operands using the TSTG and TSTH instructions, regardless of
                                       0420  ;       whether the processor supports those instructions.
                                       0420  ;
                                       0420  ;       When a reserved operand is seen, a TSTG or a TSTH is executed with
                                       0420  ;       the reserved operand at (R0).  If the processor knows
                                       0420  ;       about TSTG or TSTH, a reserved operand fault is signaled.  However,
                                       0420  ;       if it doesn't support TSTG or TSTH, an "opcode reserved to Digital"
                                       0420  ;       fault will occur. CVT_HANDLER turns this into a reserved operand
                                       0420  ;       fault.
                                       0420  ;
                                       0420  ;       If the condition being signaled is not SS$_OPCDEC or if the
                                       0420  ;       signaled instruction is not in the frame that established this
                                       0420  ;       handler, then the exception is resignaled.  A test is made to
                                       0420  ;       see if (R0) is a reserved operand.  It
                                       0420  ;       will be on the initial fault, but might not be if it has been
                                       0420  ;       fixed up by another condition handler (i.e. LIB$FIXUP_FLT).
                                       0420  ;       If it is a reserved operand, the signal name is changed to
                                       0420  ;       SS$_ROPRAND and the exception is resignaled.  Otherwise,
                                       0420  ;       execution continues with the instruction following the TSTx.
                                       0420  ;
                                       0420  ;--
                                       0420
                                       0420  CVT_HANDLER:
                                  0004 0420          .WORD   ^M<R2>
                     50    04 AC    D0 0422          MOVL    4(AP), R0                 ; signal argument list
           0000043C 8F    04 A0    D1 0426          CMPL    CHF$L_SIG_NAME(R0), #SS$_OPCDEC ; Opcode reserved to Digital fault?
                              37    12 042E          BNEQ    RESIGNAL                  ; No, resignal
                     51    08 AC    D0 0430          MOVL    8(AP), R1                 ; mechanism argument list
                           08 A1    D5 0434          TSTL    CHF$L_MCH_DEPTH(R1)       ; Is depth zero?
                              2E    12 0437          BNEQ    RESIGNAL                  ; If not, can't be this routine
                  52    60    01    C3 0439          SUBL3   #1, CHF$L_SIG_ARGS(R0), R2  ; Get position of PC
                       52   6042    DE 043D          MOVAL   (R0)[R2], R2              ; R2 has position of PC
                73FD 8F    00 B2    B1 0441          CMPW    @(R2), #^X73FD            ; TSTH?
                              0E    13 0447          BEQL    10$                       ; Yes
 00000800 8F   0C B1    0C    04    ED 0449          CMPZV   #4, #12, @CHF$L_MCH_SAVR0(R1), #^X800   ; G reserved operand?
                              1A    12 0453          BNEQ    CONTINUE                  ; No, continue execution
                              08    11 0455          BRB     20$
                8000 8F    0C B1    B1 0457  10$:    CMPW    @CHF$L_MCH_SAVR0(R1), #^X8000   ; H reserved operand?
                              10    12 045D          BNEQ    CONTINUE                  ; No, continue execution
            04 A0    00000454 8F    D0 045F  20$:    MOVL    #SS$_ROPRAND, CHF$L_SIG_NAME(R0)  ; Change condition code name
                                       0467  RESIGNAL:
               50    00000918 8F    D0 0467          MOVL    #SS$_RESIGNAL, R0         ; Resignal exception
                                    04 046E          RET
                                       046F
                                       046F  CONTINUE:
                  51    60    01    C3 046F          SUBL3   #1, CHF$L_SIG_ARGS(R0), R1 ; Get position of PC
                        62    03    C0 0473          ADDL2   #3, (R2)                  ; Add length of TSTG or TSTH
                                       0476                                            ; to instruction PC, causing
                                       0476                                            ; next instruction to be executed.
                        50    01    D0 0476          MOVL    #SS$_CONTINUE, R0         ; Continue execution
                                    04 0479          RET
                                       047A
```

```
                               047A   684         .SBTTL  OTS$$RET_A_CVT_TAB_R1
                               047A   685
                               047A   686 ;+
                               047A   687 ; JSB entry point
                               047A   688 ;-
                               047A   689
                               047A   690 OTS$$RET_A_CVT_TAB_R1::
      50   00000707'EF    DE   047A   691         MOVAL   OTS$$A_CVT_TAB, R0              ; Return address of this table
                          05   0481   692         RSB                                     ;  so that from places outside
                               0482   693                                                 ;  of this sharable image,
                               0482   694                                                 ;  through the vector, one can
                               0482   695                                                 ;  use this table.
                               0482   696
                          0482  ; THIS IS THE SUBROUTINE WHICH DOES THE MULTIPLE
                          0482  ; PRECISION MULTIPLIES. IT IS CALLED WITH BSB OR JSB
                          0482  ; WITH R2 CONTAINING A POINTER TO AN APPROPRIATE
                          0482  ; ENTRY IN THE POWER-OF-TEN TABLE. BINEXP & BINNUM
                          0482  ; ARE MULTIPLIED BY THIS ENTRY,
                          0482  ; WITH THE RESULTS GOING TO
                          0482  ; BINEXP & BINNUM, AND DECEXP IS UPDATED WITH THE
                          0482  ; POWER OF TEN VALUE.
                          0482  ; THIS ROUTINE CLOBBERS R0-R1, R3-R6.
                          0482
                          0482
                          0482  OTS$$CVT_MUL::
                          0482  RMUL:                                         ; ENTRY POINT
                          0482
             18 A8   D4   0482          CLRL    PRODF-4(R8)                   ; INIT PRODUCT
             1C A8   7C   0485          CLRQ    PRODF+0(R8)
             24 A8   7C   0488          CLRQ    PRODF+8(R8)
                          048B
             2C A8   7C   048B          CLRQ    CRY+0(R8)                     ; CLEAR CARRIES
             34 A8   7C   048E          CLRQ    CRY+8(R8)
                          0491
                          0491                  ; THIS MACRO HAS THE FUNCTION R=A*B, WITH THE CARRIES
                          0491                  ; GOING INTO THE 4 L-WORDS AT "CRY". A AND B ARE
                          0491                  ; UNSIGNED LONG-WORDS. R IS AN UNSIGNED DOUBLE LONG-WORD.
                          0491                  ; REMOVING THIS MACRO DEFINITION (WHICH IS ONLY USED ONCE),
                          0491                  ; AND EXPANDING THE CODE WHERE IT IS USED, OBSCURES THE FUNCTION.
                          0491
                          0491  .MACRO LMUL A, B, R, ?L1, ?L2, ?L3
                          0491          MOVL    A, R0                         ; Get first operand
                          0491          BEQL    L3                            ; Skip if zero
                          0491          MOVL    B, R1                         ; Get second operand
                          0491          BEQL    L3                            ; Skip if zero
                          0491          EMUL    R0, R1, #0, R0                ; FORM PRODUCT OF A AND B
                          0491          TSTL    A
                          0491          BGEQ    L1
                          0491          ADDL2   B, R1                         ; IF A<0, FIXUP FOR NEG SIGN
                          0491  L1:     TSTL    B
                          0491          BGEQ    L2
                          0491          ADDL2   A, R1                         ; IF B<0, FIXUP FOR NEG SIGN
                          0491  L2:     ADDL2   R0, R                         ; ADD LOW PRODUCT INTO RESULT
                          0491          ADWC    R1, 4+R                       ; ADD HI PRODUCT INTO RESULT
                          0491          ADWC    #0, CRY+8-PRODF+R             ; AND SAVE CARRIES
                          0491  L3:
                          0491          .ENDM LMUL
                          0491
                          0491                  ; THE FOLLOWING LOOP FORMS ALL THE CROSS-PRODUCTS
                          0491                  ; REQUIRED FOR A 4-LONG-WORD BY 4-LONG-WORD MULTIPLY.
                          0491                  ; ONLY THE HIGH 4 LONG-WORDS ARE ACCUMULATED. THE BYTE
                          0491                  ; TABLE AT "BYTAB" SHOWS THE INDICIES USED FOR THE
                          0491                  ; LONG-WORD OPERANDS AND THE RESULTING DOUBLE-LONG-
                          0491                  ; WORD PRODUCTS.
                          0491
   53  000008FB'EF   3E   0491          MOVAW   BYTAB, R3                     ; INIT BYTE TABLE INDEX
             54 83   98   0498  BYTLUP: CVTBL   (R3)+, R4                     ; SETUP 1ST INDEX
                3A   19   049B          BLSS    BYTDUN                        ; AND TEST FOR END
             55 83   98   049D          CVTBL   (R3)+, R5                     ; SETUP 2ND INDEX
             56 83   98   04A0          CVTBL   (R3)+, R6                     ; SETUP 3RD INDEX
```

```
                                     04A3   755
                                     04A3   756            LMUL    BINNUM(R8)[R4], 0(R2)[R5], PRODF_4(R8)[R6]
                          C1    11   04D5   757            BRB     BYTLUP                  ; LOOP
                                     04D7   758
                                     04D7   759 BYTDUN:
                                     04D7   760 ;          INCL    CRY+0(R8)               ; SMALL EXTRA FUDGE
               1C A8    2C A8   C0   04D7   761            ADDL2   CRY+0(R8), PRODF+0(R8)  ; PUT CARRIES INTO SUM
               20 A8    30 A8   D8   04DC   762            ADWC    CRY+4(R8), PRODF+4(R8)
               24 A8    34 A8   D8   04E1   763            ADWC    CRY+8(R8), PRODF+8(R8)
               28 A8    38 A8   D8   04E6   764            ADWC    CRY+12(R8), PRODF+12(R8)
                                     04EB   765
      51   28 A8   01    1F     EF   04EB   766            EXTZV   #31, #1, PRODF+12(R8), R1      ; GET NORMALIZE BIT
                                     04F1   767
                                     04F1   768                    ; NORMALIZED OPERANDS CANNOT PRODUCE A RESULT
                                     04F1   769                    ; UN-NORMALIZED BY MORE THAN ONE BIT POSITION. SO
                                     04F1   770                    ; IF NORM_BIT=1, SHIFT LEFT BY 0
                                     04F1   771                    ; IF NORM_BIT=0, SHIFT LEFT BY 1 AND SUB 1 FROM EXP
                                     04F1   772
                             03 12   04F1   773            BNEQ    NOSUB1                  ; XFER IF NORM_BIT = 1
                          14 A8 D7   04F3   774            DECL    BINEXP(R8)              ; NORM_BIT = 0, SUB 1 FROM EXPONENT
                                     04F6   775
                                     04F6   776                    ; MOVE THE PRODUCT FROM PRODF TO BINNUM, NORMALIZING
                                     04F6   777                    ; IT ONE BIT POSITION IF REQUIRED.
                                     04F6   778
                      51    1F  C0   04F6   779 NOSUB1: ADDL2  #31, R1                        ; DO EXTV'S FROM BIT 31 OR 32
                                     04F9   780
      68   18 A8   20    51     EE   04F9   781            EXTV    R1, #32, PRODF-4(R8), BINNUM+0(R8)     ; SHIFT LEFT 0 OR 1 BIT
   04 A8   1C A8   20    51     EE   04FF   782            EXTV    R1, #32, PRODF+0(R8), BINNUM+4(R8)
   08 A8   20 A8   20    51     EE   0506   783            EXTV    R1, #32, PRODF+4(R8), BINNUM+8(R8)
   0C A8   24 A8   20    51     EE   050D   784            EXTV    R1, #32, PRODF+8(R8), BINNUM+12(R8)
                                     0514   785
               51   0010'C2     32   0514   786            CVTWL   T_BEXP(R2), R1          ; EXTRACT BINARY EXPONENT
                  14 A8   51    C0   0519   787            ADDL2   R1, BINEXP(R8)          ; ADD EXPONENTS FOR MUL
                                     051D   788
                                     051D   789                    ; WHEN CONVERTING FROM REAL TO TEXT:
                                     051D   790                    ; THE BINARY EXPONENT MOVES TOWARD ZERO WHILE THE
                                     051D   791                    ; DECIMAL EXPONENT MOVES AWAY FROM ZERO BY AN AMOUNT
                                     051D   792                    ; ABOUT EQUAL TO LOG(BIN EXP).
                                     051D   793                    ; WHEN CONVERTING FROM TEXT TO REAL:
                                     051D   794                    ; THE DECIMAL EXPONENT MOVES TOWARDS ZERO WHIL THE
                                     051D   795                    ; BINARY EXPONENT MOVES AWAY FROM ZERO.
                                     051D   796
               51   0012'C2     32   051D   797            CVTWL   T_DEXP(R2), R1          ; GET EQUIVALENT DECIMAL EXPONENT
                  E4 A7   51    C2   0522   798            SUBL2   R1, DEC_EXP(R7)         ; AND SUB IT FROM RESULT EXP
                                     0526   799
                                05   0526   800            RSB                             ; RETURN
```

```
                    0527          .SBTTL  TABLES
                    0527
                    0527
                    0527  .MACRO NUMBER A1, A2, A3, A4, A5, A6, A7
                    0527          .LONG ^X'A5+<<^X'A6@-31>&1>, ^X'A4, ^X'A3, ^X'A2
                    0527          .WORD ^D<A1>, ^D<A7>
                    0527  .ENDM NUMBER
                    0527
                    0527                  ; THIS MACRO CREATES A TABLE ENTRY OF THE FOLLOWING FORM:
                    0527
                    0527                  ;       .LONG < LEAST SIG BITS> :     0(R2)
                    0527                  ;       .LONG <    ........    > :     4(R2)
                    0527                  ;       .LONG <    ........    > :     8(R2)
                    0527                  ;       .LONG < MOST SIG BITS  > :    12(R2)
                    0527                  ;       .WORD < BINARY EXP     > :T_BEXP(R2)
                    0527                  ;       .WORD < DECIMAL EXP    > :T_DEXP(R2)
                    0527
           00000010 0527  T_BEXP=16       ; THE BINARY EXPONENT IS BYTES 16-17 OF EACH TABLE ENTRY
           00000012 0527  T_DEXP=18       ; THE DECIMAL EXPONENT IS BYTES 18-19
                    0527
                    0527
                    0527  ;             VALUE = FRACTION * 2**POWER_OF_2 = 10**POWER_OF_10
                    0527
                    0527  ;       THE HEX FRACTION IS STORED AS A 4 LONG-WORD UNSIGNED INTEGER,
                    0527  ;       LEFT NORMALIZED, WITH THE BINARY POINT LEFT OF BIT 31
                    0527  ;       OF THE MOST SIGNIFICANT LONG-WORD.
                    0527
                    0527  ;       THE FRACTION IS GUARANTEED CORRECT FOR THE FOUR HIGH-ORDER
                    0527  ;       LONG-WORDS. ABOUT 16 BITS OF THE FIFTH LOW-ORDER LONG-WORD MAY
                    0527  ;       BE IN ERROR. THE CHECK LINE AT THE BOTTOM OF THE TABLE IS
                    0527  ;       THE PRODUCT OF THE FIRST AND LAST TABLE ENTRIES. IT WOULD
                    0527  ;       EQUAL EXACTLY 1.0 IF EVERY BIT OF THE 5 LONG-WORDS WERE CORRECT.
                    0527
                    0527  ;             DECIMAL,<-------5 LONG-WORD HEX FRACTION----------->, DECIMAL
                    0527  ;              POWER ,<--MSB-------------------------------LSB-->,  POWER
                    0527  ;              OF 2                                                 OF 10
                    0527  TSMALL:
                    0527  ;       NUMBER -27213,D986C20B,686DA869,5D1D4FD8,5B05F4C2,EEF0FB87,-8192
                    0527          NUMBER -13606,A6DD04C8,D2CE9FDE,2DE38123,A1C3CFFC,203028DA,-4096
                    053B          NUMBER  -6803,CEAE534F,34362DE4,492512D4,F2EAD2CB,8263AA10,-2048
                    054F          NUMBER  -3401,A2A682A5,DA57C0BD,87A60158,6BD3F698,F53E881E,-1024
                    0563          NUMBER  -1700,9049EE32,DB23D21C,7132D332,E3F204D4,E73177C2, -512
                    0577          NUMBER   -850,C0314325,637A1939,FA911155,FEFB5308,A23E2B15, -256
                    058B          NUMBER   -425,DDD0467C,64BCE4A0,AC7CB3F6,D05DDBDE,E26CA3DF, -128
                    059F          NUMBER   -212,A87FEA27,A539E9A5,3F2398D7,47B36224,2A1FED70,  -64
                    05B3  ; TM32:
                    05B3          NUMBER   -106,CFB11EAD,453994BA,67DE18ED,A5814AF2, B5B1A20,  -32
                    05C7  ;       NUMBER   -102,81CEB32C,4B43FCF4,80EACF94,8770CED7,4718F05A,  -31
                    05C7  ;       NUMBER    -99,A2425FF7,5E14FC31,A1258379,A94D028D,18DF2C73,  -30
                    05C7  ;       NUMBER    -96,CAD2F7F5,359A3B3E, 96EE458,13A04330,5F16F793,  -29
                    05C7  ;       NUMBER    -93,FD87B5F2,8300CA0D,8BCA9D6E,188853FC,76DCB57B,  -28
                    05C7  ;       NUMBER    -89,9E74D1B7,91E07E48,775EA264,CF55347D,CA49F16F,  -27
                    05C7  ;       NUMBER    -86,C6120625,76589DDA,95364AFE, 32A819D,3CDC6DCD,  -26
                    05C7  ;       NUMBER    -83,F79687AE,D3EEC551,3A83DDBD,83F52204,8C138944,  -25
                    05C7  ;       NUMBER    -79,9ABE14CD,44753B52,C4926A96,72793542,D78C35CE,  -24
                    05C7  ;       NUMBER    -76,C16D9A00,95928A27,75B7053C, F178293,8D6F434A,  -23
                    05C7  ;       NUMBER    -73,F1C90080,BAF72CB1,5324C68B,12DD6338,70CB1420,  -22
```

```
05C7   859 ;        NUMBER    -69,971DA050,74DA7BEE,D3F6FC16,EBCA5E03,467EEC97,  -21
05C7   860 ;        NUMBER    -66,BCE50864,92111AEA,88F4BB1C,A6BCF584,181EA7C0,  -20
05C7   861 ;        NUMBER    -63,EC1E4A7D,B69561A5,2B31E9E3,D06C32E5,1E2651B1,  -19
05C7   862 ;        NUMBER    -59,9392EE8E,921D5D07,3AFF322E,62439FCF,32D7F311,  -18
05C7   863 ;        NUMBER    -56,B877AA32,36A4B449, 9BEFEB9,FAD487C2,FF8DEFDB,  -17
05C7   864 TM16:    NUMBER    -53,E69594BE,C44DE15B,4C2EBE68,7989A9B3,BF716BD5,  -16
05DB   865          NUMBER    -49,901D7CF7,3AB0ACD9, F9D3701,4BF60A10,57A6E369,  -15
05EF   866          NUMBER    -46,B424DC35, 95CD80F,538484C1,9EF38C94,6D909C46,  -14
0603   867          NUMBER    -43,E12E1342,4BB40E13,2865A5F2, 6B06FB9,88F4C35A,  -13
0617   868          NUMBER    -39,8CBCCC09,6F5088CB,F93F87B7,442E45D3,F598FA1C,  -12
062B   869          NUMBER    -36,AFEBFF0B,CB24AAFE,F78F69A5,1539D748,F2FF38A8,  -11
063F   870          NUMBER    -33,DBE6FECE,BDEDD5BE,B573440E,5A884D1B,2FBF06D5,  -10
0653   871          NUMBER    -29,89705F41,36B4A597,31680A88,F8953030,FDD76447,   -9
0667   872          NUMBER    -26,ABCC7711,8461CEFC,FDC20D2B,36BA7C3D,3D4D3D5C,   -8
067B   873          NUMBER    -23,D6BF94D5,E57A42BC,3D329076, 4691B4C,8CA08CB8,   -7
068F   874          NUMBER    -19,8637BD05,AF6C69B5,A63F9A49,C2C1B10F,D7E457F7,   -6
06A3   875          NUMBER    -16,A7C5AC47,1B478423, FCF80DC,33721D53,CDDD6DF6,   -5
06B7   876          NUMBER    -13,D1B71758,E219652B,D3C36113,404EA4A8,C154C978,   -4
06CB   877          NUMBER     -9,83126E97,8D4FDF3B,645A1CAC, 83126E9,78D4FDEE,   -3
06DF   878          NUMBER     -6,A3D70A3D,70A3D70A,3D70A3D7, A3D70A3,D70A3D6C,   -2
06F3   879          NUMBER     -3,CCCCCCCC,CCCCCCCC,CCCCCCCC,CCCCCCCC,CCCCCCCC,   -1
0707   880 OTS$$A_CVT_TAB::
0707   881 T0:      NUMBER      1,80000000,       0,        0,        0,       0,    0
071B   882 T1:      NUMBER      4,A0000000,       0,        0,        0,       0,    1
072F   883          NUMBER      7,C8000000,       0,        0,        0,       0,    2
0743   884          NUMBER     10,FA000000,       0,        0,        0,       0,    3
0757   885          NUMBER     14,9C400000,       0,        0,        0,       0,    4
076B   886          NUMBER     17,C3500000,       0,        0,        0,       0,    5
077F   887          NUMBER     20,F4240000,       0,        0,        0,       0,    6
0793   888          NUMBER     24,98968000,       0,        0,        0,       0,    7
07A7   889          NUMBER     27,BEBC2000,       0,        0,        0,       0,    8
07BB   890          NUMBER     30,EE6B2800,       0,        0,        0,       0,    9
07CF   891          NUMBER     34,9502F900,       0,        0,        0,       0,   10
07E3   892          NUMBER     37,BA43B740,       0,        0,        0,       0,   11
07F7   893          NUMBER     40,E8D4A510,       0,        0,        0,       0,   12
080B   894          NUMBER     44,9184E72A,       0,        0,        0,       0,   13
081F   895          NUMBER     47,B5E620F4,80000000,        0,        0,       0,   14
0833   896          NUMBER     50,E35FA931,A0000000,        0,        0,       0,   15
0847   897 T16:     NUMBER     54,8E1BC9BF, 4000000,        0,        0,       0,   16
085B   898 ;        NUMBER     57,B1A2BC2E,C5000000,        0,        0,       0,   17
085B   899 ;        NUMBER     60,DE0B6B3A,76400000,        0,        0,       0,   18
085B   900 ;        NUMBER     64,8AC72304,89E80000,        0,        0,       0,   19
085B   901 ;        NUMBER     67,AD78EBC5,AC620000,        0,        0,       0,   20
085B   902 ;        NUMBER     70,D8D726B7,177A8000,        0,        0,       0,   21
085B   903 ;        NUMBER     74,87867832,6EAC9000,        0,        0,       0,   22
085B   904 ;        NUMBER     77,A968163F, A57B400,        0,        0,       0,   23
085B   905 ;        NUMBER     80,D3C21BCE,CCEDA100,        0,        0,       0,   24
085B   906 ;        NUMBER     84,84595161,401484A0,        0,        0,       0,   25
085B   907 ;        NUMBER     87,A56FA5B9,9019A5C8,        0,        0,       0,   26
085B   908 ;        NUMBER     90,CECB8F27,F4200F3A,        0,        0,       0,   27
085B   909 ;        NUMBER     94,813F3978,F8940984,40000000,        0,       0,   28
085B   910 ;        NUMBER     97,A18F07D7,36B90BE5,50000000,        0,       0,   29
085B   911 ;        NUMBER    100,C9F2C9CD, 4674EDE,A4000000,        0,       0,   30
085B   912 ;        NUMBER    103,FC6F7C40,45812296,4D000000,        0,       0,   31
085B   913 ; T32:
085B   914          NUMBER    107,9DC5ADA8,2B70B59D,F0200000,        0,       0,   32
086F   915          NUMBER    213,C2781F49,FFCFA6D5,3CBF6B71,C76B25FB,50F80800,   64
```

```
            0883             NUMBER    426,93BA47C9,80E98CDF,C66F336C,36B10137, 234F3FC,  128
            0897             NUMBER    851,AA7EEBFB,9DF9DE8D,DDBB901B,98FEEAB7,851E4CBB,  256
            08AB             NUMBER   1701,E319A0AE,A60E91C6,CC655C54,BC5058F8,9C658389,  512
            08BF             NUMBER   3402,C9767586,81750C17,650D3D28,F18B50CE,526B9865, 1024
            08D3             NUMBER   6804,9E8B3B5D,C53D5DE4,A74D28CE,329ACE52,6A31978C, 2048
            08E7             NUMBER  13607,C4605202,8A20979A,C94C153F,804A4A92,65761F39, 4096
            08FB     ;       NUMBER  27214,96A3A1D1,7FAF211A, C7C2892,305F4E12, 72B205F, 8192
            08FB     ;                   0,FFFFFFFF,FFFFFFFF,FFFFFFFF,FFFFFFFF,FFFF5EB4      ; 1.0 IF EXA
            08FB
            08FB                     ; THIS TABLE CONTAINS THE BYTE INDICIES FOR THE
            08FB                     ; MULTIPLE PRECISION MULTIPLY CROSS PRODUCTS.
            08FB                     ; THE 1ST AND 2ND ENTRIES ON EACH LINE ARE THE INDICIES
            08FB                     ; FOR THE MULTIPLICAND AND THE MULTIPLIER. THE THIRD
            08FB                     ;  ENTRY IS THE PRODUCT INDEX.
            08FB
  00 03 00  08FB     BYTAB:  .BYTE   0,3,0
  00 00 03  08FE             .BYTE   3,0,0
  00 01 02  0901             .BYTE   2,1,0
  00 02 01  0904             .BYTE   1,2,0
  01 03 01  0907             .BYTE   1,3,1
  01 01 03  090A             .BYTE   3,1,1
  01 02 02  090D             .BYTE   2,2,1
  02 03 02  0910             .BYTE   2,3,2
  02 02 03  0913             .BYTE   3,2,2
  03 03 03  0916             .BYTE   3,3,3
        FF  0919             .BYTE   -1              ; END FLAG
            091A
            091A             .END
```

```
ASCII_ZEROES                       00000000 R      02          T1                             0000071B R      02
BEGSRC                             000001CE R      02          T16                            00000847 R      02
BIGEX1                             000001F5 R      02          TABLE                          00000008 R      02
BIGEX2                             00000209 R      02          TM16                           000005C7 R      02
BIGEX3                             0000021B R      02          TSMALL                         00000527 R      02
BIGEXP                             000001E5 R      02          TSTVAL_G                       0000015E R      02
BINEXP                           = 00000014                    TSTVAL_H                       000000D6 R      02
BINNUM                           = 00000000                    V_ROUND_RIGHT                = 00000019
BYTAB                              000008FB R      02          V_TRUNCATE                   = 00000018
BYTDUN                             000004D7 R      02          ZERO                           0000040B R      02
BYTLUP                             00000498 R      02
CHF$L_MCH_DEPTH                  = 00000008
CHF$L_MCH_SAVR0                  = 0000000C
CHF$L_SIG_ARGS                   = 00000000
CHF$L_SIG_NAME                   = 00000004
CONTINUE                           0000046F R      02
CRY                              = 0000002C
CVT_HANDLER                        00000420 R      02
DEC_EXP                          = FFFFFFE4
DIGLUP                             000002E4 R      02
FINIS                              00000404 R      02
FLAGS                            = FFFFFFF4
FOUND                              00000283 R      02
GETDIG                             000002D4 R      02
INT                              = 00000010
LOCAL_FRAME                      = 0000003C
MULDUN                             00000299 R      02
NEG_VAL_G                          00000170 R      02
NEG_VAL_H                          000000E6 R      02
NOSUB1                             000004F6 R      02
NOTRES_G                           0000019B R      02
NOTRES_H                           00000110 R      02
OFFSET                           = FFFFFFE0
OTS$$A_CVT_TAB                     00000707 RG     02
OTS$$CVT_G_T_R8                    00000158 RG     02
OTS$$CVT_H_T_R8                    000000D0 RG     02
OTS$$CVT_MUL                       00000482 RG     02
OTS$$RET_A_CVT_TAB_R1              0000047A RG     02
PACKED                           = FFFFFFF8
PRODF                            = 0000001C
PRODF_4                          = 00000018
RESIGNAL                           00000467 R      02
RMUL                               00000482 R      02
ROUND                              000003C0 R      02
RT_RND                           = FFFFFFDC
SIGN                             = FFFFFFE8
SIG_DIGITS                       = FFFFFFF0
SMALL                              0000027C R      02
SMLEXP                             000001EE R      02
SRCL1                              00000243 R      02
SRCLIN                             0000022E R      02
SS$_CONTINUE                     = 00000001
SS$_OPCDEC                       = 0000043C
SS$_RESIGNAL                     = 00000918
SS$_ROPRAND                      = 00000454
STRING_ADDR                      = FFFFFFEC
TO                                 00000707 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                       0000091A  (  2330.)  02 (  2.)    PIC    USR    CON    REL    LCL   SHR   EXE   RD   NOWRT NOVEC LONG

                                          +------------------------+
                                          ! Performance indicators !
                                          +------------------------+

Phase                        Page faults    CPU Time       Elapsed Time
-----                        -----------    --------       ------------
Initialization                        30    00:00:00.07    00:00:01.80
Command processing                   105    00:00:00.31    00:00:04.59
Pass 1                               228    00:00:04.76    00:00:19.55
Symbol table sort                      0    00:00:00.50    00:00:01.51
Pass 2                               178    00:00:01.55    00:00:06.27
Symbol table output                   10    00:00:00.05    00:00:00.06
Psect synopsis output                  3    00:00:00.01    00:00:00.01
Cross-referance output                 0    00:00:00.00    00:00:00.00
Assembler run totals                 556    00:00:07.26    00:00:33.79

The working set limit was 1200 pages.
38880 bytes (76 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 482 non-local and 19 local symbols.
943 source lines were read in Pass 1, producing 15 object records in Pass 2.
13 pages of virtual memory were used to define 11 macros.

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

Macro library name                       Macros defined
------------------                       --------------
_$255$DUA28:[SYSLIB]STARLET.MLB;2                     5

486 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

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

0212 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

OTSCVTPD
LIS

OTSCVTTF
LIS

OTSCVTRFP
LIS

OTSCVTRT
LIS

OTSCVTTOL
LIS

OTSCVTPG
LIS

OTSCVTTR
LIS

OTSCVTRHP
LIS

OTSLINKAG
LIS

OTSCVTRDP
LIS

OTSCVTTIL
LIS

OTSCVTTLL
LIS

OTSCVTPF
LIS

OTSCVTRGP
LIS

OTSMOVE
LIS

OTSMSG
LIS

OTSCVTPH
LIS

OTSLUN
LIS