```
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        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
```

FILEID**OTSCVTLT

D 15

OT
1-

```
  000000    TTTTTTTTTT    SSSSSSSS    CCCCCCCC  VV        VV  TTTTTTTTTT  LL          TTTTTTTTTT
  000000    TTTTTTTTTT    SSSSSSSS    CCCCCCCC  VV        VV  TTTTTTTTTT  LL          TTTTTTTTTT
00      00      TT      SS          CC          VV        VV      TT      LL              TT
00      00      TT      SS          CC          VV        VV      TT      LL              TT
00      00      TT      SS          CC          VV        VV      TT      LL              TT
00      00      TT      SS          CC          VV        VV      TT      LL              TT
00      00      TT        SSSSSS    CC          VV        VV      TT      LL              TT
00      00      TT        SSSSSS    CC          VV        VV      TT      LL              TT
00      00      TT              SS  CC          VV        VV      TT      LL              TT
00      00      TT              SS  CC          VV        VV      TT      LL              TT
00      00      TT              SS  CC            VV    VV        TT      LL              TT        ....
00      00      TT              SS  CC            VV    VV        TT      LL              TT        ....
  000000        TT      SSSSSSSS      CCCCCCCC      VV            TT      LLLLLLLLLL      TT        ....
  000000        TT      SSSSSSSS      CCCCCCCC      VV            TT      LLLLLLLLLL      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            .TITLE  OTS$CVTLT - Convert longword to text, O, Z, L, B, U, I formats
0000            .IDENT  /1-014/                 ; File: OTSCVTLT.MAR  Edit: MDL1014
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  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   *
0000    ;*  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    ; Routines to convert values of any length to text using O (octal),
0000    ; Z (hexadecimal), B (binary) and L (logical) formats.  Also routines to
0000    ; convert byte, word and longword integers to text using I (signed decimal)
0000    ; and unsigned decimal formats.
0000    ;
0000    ; ENVIRONMENT: User Mode, AST Reentrant
0000    ;
0000    ;--
0000    ; AUTHOR: Steven B. Lionel, CREATION DATE: 21-Mar-1979
0000    ;
```

```
0000          .SBTTL  Edit History
0000    ;
0000    ; 1-001 - Original.  Replaces FOR$CNVOI.  SBL 22-Mar-1979
0000    ; 1-002 - Move V_FORCEPLUS to bit position 1.  SBL 25-July-1979
0000    ; 1-003 - Speed Improvements.  New I format logic.  Use CASEB in
0000    ;         INITIALIZE.  SBL 28-Dec-1979
0000    ; 1-004 - Fix bug in CASE.  SBL 31-Dec-1979
0000    ; 1-005 - Do correct thing for >128 arguments.  SBL 31-Dec-1979
0000    ; 1-006 - Minor code improvements courtesy of Rich Grove.  SBL 2-Jan-1980
0000    ; 1-007 - More minor code improvements.  Make value_size of zero an error.
0000    ;         SBL 3-Jan-1980
0000    ; 1-008 - Fix bug where OTS$CVT_L_TI of 3 arguments doesnt fetch
0000    ;         the value.  SBL 13-March-1980
0000    ; 1-009 - Add OTS$CVT_L_TB.  SBL 6-Nov-1980
0000    ; 1-010 - Make OTS$CVT_L_TI produce a blank field when value is zero and
0000    ;         int_digits is zero.  SPR 11-37827  JAW  22-May-1981
0000    ; 1-011 - Add bit_offset and flags parameter to B, O and Z conversions.  SBL 6-July-
0000    ; 1-012 - Reverse order of bit offset and flags parameters in B, O and Z.  SBL 30-Oc
0000    ; 1-013 - Add OTS$CVT_L_TU.  SBL 27-Apr-1983
0000    ; 1-014 - fix bug where OTS$CVT_L_TU of 4 or 5 arguments doesnt fetch the
0000    ;         value.  MDL 25-May-1984
```

```
                                     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         .EXTRN  OTS$_OUTCONERR                 ; error code
                                     0000
                                     0000    ;
                                     0000    ; MACROS:
                                     0000    ;
                                     0000
                                     0000    ;
                                     0000    ; EQUATED SYMBOLS:
                                     0000    ;
                                     0000
                                     0000    ;
                                     0000    ; PSECT DECLARATIONS:
                                     0000    ;
                                 00000000         .PSECT _OTS$CODE PIC, USR, CON, REL, LCL, SHR, -
                                     0000                         EXE, RD, NOWRT, LONG
                                     0000
                                     0000    ;
                                     0000    ; OWN STORAGE:
                                     0000    ;
                                     0000
                                     0000    LETTERS:
42 41 39 38 37 36 35 34 33 32 31 30  0000         .ASCII  /0123456789ABCDEF/     ; Characters for output
                         46 45 44 43  000C
                                     0010
```

```
                  0010          .SBTTL  OTS$CVT_L_TO - Long to text, O format
                  0010   ;++
                  0010   ; FUNCTIONAL DESCRIPTION:
                  0010   ;
                  0010   ;      This routine converts its input value to a text representation,
                  0010   ;      using base 8 (octal).  The input value may be of any length.
                  0010   ;
                  0010   ;      OTS$CVT_L_TO supports FORTRAN Ow and Ow.m output conversion.
                  0010   ;
                  0010   ;      A separate entry point FOR$CNV_OUT_O is provided for compatibility
                  0010   ;      with previous releases.  Note that the input value for
                  0010   ;      OTS$CVT_L_TO is by reference while that for FOR$CNV_OUT_O is
                  0010   ;      by value.
                  0010   ;
                  0010   ; CALLING SEQUENCE:
                  0010   ;
                  0010   ;      status.wlc.v = OTS$CVT_L_TO (value.rx.r, out_string.wt.ds
                  0010   ;                                   [, int_digits.rl.v
                  0010   ;                                   [, value_size.rl.v
                  0010   ;                                   [, flags.rbu.v
                  0010   ;                                   [, bit_offset.rl.v]]]])
                  0010   ;
                  0010   ;      status.wlc.v = FOR$CNV_OUT_O (value.rl.v, out_string.wt.ds)
                  0010   ;
                  0010   ;
                  0010   ; INPUT PARAMETERS:
                  0010   ;
        00000004  0010          value = 4               ; Input value to be converted to text
        0000000C  0010          int_digits = 12         ; Minimum number of digits to be produced.
                  0010                                  ; If actual number of significant digits
                  0010                                  ; is smaller, leading zeroes will be
                  0010                                  ; produced.  If int_digits is zero
                  0010                                  ; and value is zero, a blank field will
                  0010                                  ; result.  The default is 1.
        00000010  0010          value_size = 16         ; The size of value in bytes.  The
                  0010                                  ; default is 4 if this argument is not present.
                  0010                                  ; If V_SIZE_IN_BITS set, value_size is in units of b
        00000014  0010          flags      = 20         ; Caller supplied flags
        00000002  0010            V_SIZE_IN_BITS = 2    ; "value_size" is in bits rather than bytes.
        00000018  0010          bit_offset = 24         ; Starting bit position.  Default is 0.
                  0010
                  0010   ;
                  0010   ; IMPLICIT INPUTS:
                  0010   ;
                  0010          NONE
                  0010   ;
                  0010   ; OUTPUT PARAMETERS:
                  0010   ;
        00000008  0010          out_string = 8          ; Output string by descriptor.
                  0010   ;
                  0010   ; IMPLICIT OUTPUTS:
                  0010   ;
                  0010          NONE
                  0010   ;
                  0010   ; COMPLETION CODES:
                  0010   ;
                  0010          SS$_NORMAL      - Successful completion.
```

```
                           0010  :        OTS$_OUTCONERR  - Output conversion error.  The converted value
                           0010  :                          does not fit in the field provided.  The field
                           0010  :                          is filled with asterisks.  This error is also
                           0010  :                          given if value_size is not positive.
                           0010  :
                           0010  : SIDE EFFECTS:
                           0010  :
                           0010  :        NONE
                           0010  :
                           0010  :--
                           0010
                     00FC  0010          .ENTRY  FOR$CNV_OUT_O, ^M<R2,R3,R4,R5,R6,R7>
                           0012
        54   04 AC    9E   0012          MOVAB   value(AP), R4           ; Address of value
               015B   30   0016          BSBW    INITIALIZE              ; Set up default values
           53   20    D0   0019          MOVL    #32, R3                 ; Value MUST be 4 bytes!
                09    11   001C          BRB     COMMON_O                ; Go to common routine
                           001E
                     00FC  001E          .ENTRY OTS$CVT_L_TO , ^M<R2,R3,R4,R5,R6,R7>
                           0020
        54   04 AC    D0   0020          MOVL    value(AP), R4           ; Address of value
               014D   30   0024          BSBW    INITIALIZE              ; Set up default values
                           0027
                           0027
                           0027  COMMON_O:
           56   51    D0   0027          MOVL    R1, R6                  ; last set char address
     01    01   53    CF   002A  10$:    CASEL   R3, #1, #1              ; Select on bits remaining
                   0006'   002E  1$:     .WORD   11$-1$                  ; 1 bit
                   001B'   0030          .WORD   12$-1$                  ; 2 bits
                2A    11   0032          BRB     13$                     ; 3 or more bits
                           0034                                          ; can't be zero
57   64    01   55    EF   0034  11$:    EXTZV   R5, #1, (R4), R7        ; extract one bit
                3E    13   0039          BEQL    EXIT_O                  ; if zero, exit
              0176    30   003B          BSBW    ZERO_FILL               ; fill with zero to this point
     71    BE AF47    90   003E          MOVB    LETTERS[R7], -(R1)      ; move character
                52    D7   0043          DECL    R2                      ; decrement digits count
                56    D7   0045          DECL    R6                      ; decrement place holder
                30    11   0047          BRB     EXIT_O
57   64    02   55    EF   0049  12$:    EXTZV   R5, #2, (R4), R7        ; extract 2 bits
                29    15   004E          BLEQ    EXIT_O                  ; if zero, finish
              0161    30   0050          BSBW    ZERO_FILL               ; fill with zeroes
     71    A9 AF47    90   0053          MOVB    LETTERS[R7], -(R1)      ; move character
                52    D7   0058          DECL    R2                      ; decrement digits count
                56    D7   005A          DECL    R6                      ; decrement place holder
                1B    11   005C          BRB     EXIT_O                  ; exit
57   64    03   55    EF   005E  13$:    EXTZV   R5, #3, (R4), R7        ; extract 3 bits
                0A    13   0063          BEQL    40$                     ; skip insert if zero
              014C    30   0065          BSBW    ZERO_FILL               ; fill with zeroes
     76    94 AF47    90   0068          MOVB    LETTERS[R7], -(R6)      ; move character
                52    D7   006D          DECL    R2                      ; decrement digits count
           55   03    C0   006F  40$:    ADDL2   #3, R5                  ; increment position
                51    D7   0072          DECL    R1                      ; decrement character pointer
           53   03    C2   0074          SUBL2   #3, R3                  ; decrement count
                B1    1A   0077          BGTRU   10$                     ; continue if not done
                           0079
                           0079  EXIT_O:
                52    D5   0079          TSTL    R2                      ; more zeroes to fill?
```

```
                 09   15  007B              BLEQ    50$                         ; no
     51   56     52   C3  007D              SUBL3   R2, R6, R1                  ; insert R2 zeroes
                 50   D6  0081              INCL    R0                          ; we aren't writing a char here
               012E   30  0083              BSBW    ZERO_FILL                   ; fill with zeroes
                 50   D5  0086      50$:    TSTL    R0                          ; Blank fill?
                 06   15  0088              BLEQ    70$                         ; No
          76     20   90  008A      60$:    MOVB    #^A/ /, -(R6)               ; Move a blank
             FA  50   F5  008D              SOBGTR  R0, 60$                     ; Loop till done
          50     01   D0  0090      70$:    MOVL    #1, R0                      ; SS$_NORMAL
                      04  0093              RET                                 ; exit
          0094         .SBTTL  OTS$CVT_L_TZ - Long to text, Z format
          0094  ;++
          0094  ; FUNCTIONAL DESCRIPTION:
          0094  ;
          0094  ;      This routine converts its input value to a text representation,
          0094  ;      using base 16 (hexadecimal).  The input value may be of any length.
          0094  ;
          0094  ;      OTS$CVT_L_TZ supports FORTRAN Zw and Zw.m output conversion.
          0094  ;
          0094  ;      A separate entry point FOR$CNV_OUT_Z is provided for compatibility
          0094  ;      with previous releases.  Note that the input value for
          0094  ;      OTS$CVT_L_TZ is by reference while that for FOR$CNV_OUT_Z is
          0094  ;      by value.
          0094  ;
          0094  ; CALLING SEQUENCE:
          0094  ;
          0094  ;      status.wlc.v = OTS$CVT_L_TZ (value.rx.r, out_string.wt.ds
          0094  ;                                 [, int_digits.rl.v
          0094  ;                                 [, value_size.rl.v
          0094  ;                                 [, flags.rbu.v
          0094  ;                                 [, bit_offset.rl.v]]]])
          0094  ;
          0094  ;      status.wlc.v = FOR$CNV_OUT_Z (value.rl.v, out_string.wt.ds)
          0094  ;
          0094  ;
          0094  ; INPUT PARAMETERS:
          0094  ;
00000004  0094         value = 4               ; Input value to be converted to text
0000000C  0094         int_digits = 12         ; Minimum number of digits to be produced.
          0094                                 ; If actual number of significant digits
          0094                                 ; is smaller, leading zeroes will be
          0094                                 ; produced.  If int_digits is zero
          0094                                 ; and value is zero, a blank field will
          0094                                 ; result.  The default is 1.
00000010  0094         value_size = 16         ; The size of value in bytes.  The
          0094                                 ; default is 4 if this argument is not present.
          0094                                 ; If flags bit V_SIZE_IN_BITS is set, value_size
          0094                                 ; is the number of bits in the value.
00000014  0094         flags = 20              ; Caller supplied flags.  Defined bits are:
00000002  0094           V_SIZE_IN_BITS = 2    ; "value_size" is in units of bits
00000018  0094         bit_offset = 24         ; Offset of value in bits.  Default is zero.
          0094
          0094  ;
          0094  ; IMPLICIT INPUTS:
          0094  ;
          0094  ;      NONE
          0094  ;
          0094  ; OUTPUT PARAMETERS:
          0094  ;
00000008  0094         out_string = 8          ; Output string by descriptor.
          0094  ;
          0094  ; IMPLICIT OUTPUTS:
          0094  ;
          0094  ;      NONE
          0094  ;
          0094  ; COMPLETION CODES:
          0094  ;
```

```
                         0094  ;         SS$_NORMAL       - Successful completion.
                         0094  ;         OTS$_OUTCONERR   - Output conversion error.  The converted value
                         0094  ;                            does not fit in the field provided.  The field
                         0094  ;                            is filled with asterisks.  This error is also given
                         0094  ;                            if value_size is not positive.
                         0094  ;
                         0094  ; SIDE EFFECTS:
                         0094  ;
                         0094  ;         NONE
                         0094  ;
                         0094  ;--
                         0094
                   00FC  0094            .ENTRY  FOR$CNV_OUT_Z, ^M<R2,R3,R4,R5,R6,R7>
                         0096
       54    04 AC   9E  0096            MOVAB   value(AP), R4            ; Address of value
              00D7   30  009A            BSBW    INITIALIZE               ; Set up default values
          53    20   D0  009D            MOVL    #32, R3                  ; Value MUST be 4 bytes!
                09   11  00A0            BRB     COMMON_Z                 ; Go to common routine
                         00A2
                   00FC  00A2            .ENTRY  OTS$CVT_L_TZ , ^M<R2,R3,R4,R5,R6,R7>
                         00A4
       54    04 AC   D0  00A4            MOVL    value(AP), R4            ; Address of value
              00C9   30  00A8            BSBW    INITIALIZE               ; Set up default values
                         00AB
                         00AB
                         00AB  COMMON_Z:
          56    51   D0  00AB            MOVL    R1, R6                   ; last set char address
     02   01    53   CF  00AE  10$:      CASEL   R3, #1, #2               ; Select on bits remaining
                 0008'   00B2  1$:       .WORD   11$-1$                   ; 1 bit
                 001E'   00B4            .WORD   12$-1$                   ; 2 bits
                 0034'   00B6            .WORD   13$-1$                   ; 3 bits
                42   11  00B8            BRB     14$                      ; 4 or more bits
                         00BA                                             ; can't be zero
57   64   01    55   EF  00BA  11$:      EXTZV   R5, #1, (R4), R7         ; extract one bit
                57   13  00BF            BEQL    EXIT_Z                   ; if zero, exit
              00F0   30  00C1            BSBW    ZERO_FILL                ; fill with zero to this point
   71   FF37 CF47    90  00C4            MOVB    LETTERS[R7], -(R1)       ; move character
                52   D7  00CA            DECL    R2                       ; decrement digits count
                56   D7  00CC            DECL    R6                       ; decrement place holder
                48   11  00CE            BRB     EXIT_Z
57   64   02    55   EF  00D0  12$:      EXTZV   R5, #2, (R4), R7         ; extract 2 bits
                41   15  00D5            BLEQ    EXIT_Z                   ; if zero, finish
              00DA   30  00D7            BSBW    ZERO_FILL                ; fill with zeroes
   71   FF21 CF47    90  00DA            MOVB    LETTERS[R7], -(R1)       ; move character
                52   D7  00E0            DECL    R2                       ; decrement digits count
                56   D7  00E2            DECL    R6                       ; decrement place holder
                32   11  00E4            BRB     EXIT_Z                   ; exit
57   64   03    55   EF  00E6  13$:      EXTZV   R5, #3, (R4), R7         ; extract 3 bits
                2B   13  00EB            BEQL    EXIT_Z                   ; skip insert if zero
              00C4   30  00ED            BSBW    ZERO_FILL                ; fill with zeroes
   71   FF0B CF47    90  00F0            MOVB    LETTERS[R7], -(R1)       ; move character
                52   D7  00F6            DECL    R2                       ; decrement digits count
                56   D7  00F8            DECL    R6                       ; decrement place holder
                1C   11  00FA            BRB     EXIT_Z
57   64   04    55   EF  00FC  14$:      EXTZV   R5, #4, (R4), R7         ; extract 4 bits
                0B   13  0101            BEQL    40$                      ; skip insert if zero
              00AE   30  0103            BSBW    ZERO_FILL                ; fill with zeroes
       76  FEF5 CF47  90  0106   341          MOVB    LETTERS[R7], -(R6)      ; move character
                 52   D7  010C   342          DECL    R2                      ; decrement digits count
           55    04   C0  010E   343 40$:     ADDL2   #4, R5                  ; increment position
                 51   D7  0111   344          DECL    R1                      ; decrement character pointer
           53    04   C2  0113   345          SUBL2   #4, R3                  ; decrement count
                 96   1A  0116   346          BGTRU   10$                     ; continue if not done
                          0118   347
                          0118   348 EXIT_Z:
                 52   D5  0118   349          TSTL    R2                      ; more zeroes to fill?
                 09   15  011A   350          BLEQ    30$                     ; no
        51  56   52   C3  011C   351          SUBL3   R2, R6, R1              ; insert R2 zeroes
                 50   D6  0120   352          INCL    R0                      ; we aren't writing a char here
               008F   30  0122   353          BSBW    ZERO_FILL               ; fill with zeroes
                 50   D5  0125   354 30$:     TSTL    R0                      ; Blank fill?
                 06   15  0127   355          BLEQ    50$                     ; No
           76    20   90  0129   356 40$:     MOVB    #^A/ /, -(R6)           ; Move a blank
              FA 50   F5  012C   357          SOBGTR  R0, 40$                 ; Loop till done
           50    01   D0  012F   358 50$:     MOVL    #1, R0                  ; SS$_NORMAL
                      04  0132   359          RET                             ; exit
```

```
                0133           .SBTTL  OTS$CVT_L_TB - Long to text, binary format
                0133     ;++
                0133     ; FUNCTIONAL DESCRIPTION:
                0133     ;
                0133     ;        This routine converts its input value to a text representation,
                0133     ;        using base 2 (binary).  The input value may be of any length.
                0133     ;
                0133     ; CALLING SEQUENCE:
                0133     ;
                0133     ;        status.wlc.v = OTS$CVT_L_TB (value.rx.r, out_string.wt.ds
                0133     ;                                     [, int_digits.rl.v
                0133     ;                                     [, value_size.rl.v
                0133     ;                                     [, flags.rbu.v
                0133     ;                                     [, bit_offset.rl.v]]]])
                0133     ;
                0133     ; INPUT PARAMETERS:
                0133     ;
00000004        0133             value = 4                ; Input value to be converted to text
0000000C        0133             int_digits = 12          ; Minimum number of digits to be produced.
                0133                                      ; If actual number of significant digits
                0133                                      ; is smaller, leading zeroes will be
                0133                                      ; produced.  If int_digits is zero
                0133                                      ; and value is zero, a blank field will
                0133                                      ; result.  The default is 1.
00000010        0133             value_size = 16          ; The size of value in bytes.  The
                0133                                      ; default is 4 if this argument is not present.
                0133                                      ; If flags bit V_SIZE_IN_BITS is set, value_size
                0133                                      ; is the number of bits in the value.
00000014        0133             flags = 20               ; Caller supplied flags.  Defined bits are:
00000002        0133               V_SIZE_IN_BITS = 2     ; "value_size" is in units of bits
00000018        0133             bit_offset = 24          ; Offset of value in bits.  Default is zero
                0133
                0133     ;
                0133     ; IMPLICIT INPUTS:
                0133     ;
                0133     ;        NONE
                0133     ;
                0133     ; OUTPUT PARAMETERS:
                0133     ;
00000008        0133             out_string = 8           ; Output string by descriptor.
                0133     ;
                0133     ; IMPLICIT OUTPUTS:
                0133     ;
                0133     ;        NONE
                0133     ;
                0133     ; COMPLETION CODES:
                0133     ;
                0133     ;        SS$_NORMAL      - Successful completion.
                0133     ;        OTS$_OUTCONERR  - Output conversion error.  The converted value
                0133     ;                          does not fit in the field provided.  The field
                0133     ;                          is filled with asterisks.  This error is also given
                0133     ;                          if value_size is not positive.
                0133     ;
                0133     ; SIDE EFFECTS:
                0133     ;
                0133     ;        NONE
                0133     ;
```

```
                                  0133   418 ;--
                                  0133   419
                           00FC   0133   420          .ENTRY OTS$CVT_L_TB , ^M<R2,R3,R4,R5,R6,R7>
                                  0135   421
              54    04 AC   D0    0135   422          MOVL    value(AP), R4            ; Address of value
                      0038  30    0139   423          BSBW    INITIALIZE               ; Set up default values
                                  013C   424
                 56    51   D0    013C   425          MOVL    R1, R6                   ; last set char address
       57    64    01    55   EF  013F   426 10$:     EXTZV   R5, #1, (R4), R7         ; extract 1 bits
                       0B   13    0144   427          BEQL    20$                      ; skip insert if zero
                     006B   30    0146   428          BSBW    ZERO_FILL                ; fill with zeroes
           76   FEB2 CF47   90    0149   429          MOVB    LETTERS[R7], -(R6)       ; move character
                       52   D7    014F   430          DECL    R2                       ; decrement digits count
                       51   D7    0151   431 20$:     DECL    R1                       ; decrement character pointer
                       55   D6    0153   432          INCL    R5                       ; increment bit position
                       53   D7    0155   433          DECL    R3                       ; decrement bit count
                       E6   14    0157   434          BGTR    10$                      ; loop back if more bits
                                  0159   435
                                  0159   436 EXIT_B:
                       52   D5    0159   437          TSTL    R2                       ; more zeroes to fill?
                       09   15    015B   438          BLEQ    30$                      ; no
           51    56    52   C3    015D   439          SUBL3   R2, R6, R1               ; insert R2 zeroes
                       50   D6    0161   440          INCL    R0                       ; we aren't writing a char here
                     004E   30    0163   441          BSBW    ZERO_FILL                ; fill with zeroes
                       50   D5    0166   442 30$:     TSTL    R0                       ; Blank fill?
                       06   15    0168   443          BLEQ    50$                      ; No
                 76    20   90    016A   444 40$:     MOVB    #^A/ /, -(R6)            ; Move a blank
                    FA 50   F5    016D   445          SOBGTR  R0, 40$                  ; Loop till done
                 50    01   D0    0170   446 50$:     MOVL    #1, R0                   ; SS$_NORMAL
                            04    0173   447          RET                              ; exit
```

```
                        0174          .SBTTL  Local subroutines
                        0174
                        0174    ;+
                        0174    ; INITIALIZE - Perform common initialization
                        0174    ;
                        0174    ;       1. R0 gets string length.
                        0174    ;       2. R1 gets address of 1 byte past end of out_string.
                        0174    ;       3. R2 gets int_digits value.
                        0174    ;       4. R3 gets value size in bits.
                        0174    ;       5. R5 gets starting bit position
                        0174    ;-
                        0174
                        0174    INITIALIZE:
         50   08 BC  7D 0174            MOVQ    @out_string(AP), R0     ; Get string descriptor
            50   50  3C 0178            MOVZWL  R0, R0                  ; R0 gets string length
            51   50  C0 017B            ADDL2   R0, R1                  ; R1 has 1 past last byte
            52   01  D0 017E            MOVL    #1, R2                  ; default digits in int
            53   20  D0 0181            MOVL    #32, R3                 ; default size in bits
                 55  D4 0184            CLRL    R5                      ; default bit offset
            57   08  D0 0186            MOVL    #8, R7                  ; Default multiplier size-to-bits
      04    02   6C  8F 0189            CASEB   (AP), #2, #4            ; Select on argument count
                 0026'  018D    1$:     .WORD   20$-1$                  ; 2 arguments
                 0022'  018F            .WORD   30$-1$                  ; 3 arguments
                 0016'  0191            .WORD   40$-1$                  ; 4 arguments
                 000E'  0193            .WORD   50$-1$                  ; 5 arguments
                 000A'  0195            .WORD   60$-1$                  ; 6 arguments
                        0197            ; fall through                  ; Assume >6 arguments
         55   18 AC  D0 0197    60$:    MOVL    bit_offset(AP), R5      ; Get bit offset
    03 14 AC      02 E1 019B    50$:    BBC     #V_SIZE_IN_BITS, flags(AP), 40$ ; Bit not set?
            57   01  D0 01A0            MOVL    #1, R7                  ; value_size is in bits
   53    57   10 AC  C5 01A3    40$:    MULL3   value_size(AP), R7, R3  ; Get size in bits
            1D       1D 01A8            BVS     ERROR                   ; Error if overflow
            03       12 01AA            BNEQ    30$                     ; Ok if not zero
         53      20  D0 01AC            MOVL    #32, R3                 ; Assume 32 bits
         52   0C AC  D0 01AF    30$:    MOVL    int_digits(AP), R2      ; Get int_digits argument
                     05 01B3    20$:    RSB                             ; End of initialization
                        01B4
                        01B4
                        01B4    ;+
                        01B4    ; ZERO_FILL - Fill in skipped zeroes
                        01B4    ;
                        01B4    ;       ZERO_FILL is called whenever a main routine wishes to output
                        01B4    ;       a non-zero digit.  First, it checks to see if there is room
                        01B4    ;       for one more character.  If not, it branches to ERROR.
                        01B4    ;       It then fills with zeroes the space between the last non-zero
                        01B4    ;       digit and the current location, if any.  It also updates the
                        01B4    ;       character pointers and counts appropriately.
                        01B4    ;-
                        01B4
                        01B4    ZERO_FILL:
            50       D7 01B4            DECL    R0                      ; Reduce char count
            0F       19 01B6            BLSS    ERROR                   ; If negative, out of room
         56   51     D1 01B8            CMPL    R1, R6                  ; Any difference?
              01     19 01BB            BLSS    10$                     ; Yes
                     05 01BD            RSB                             ; No, exit
         76   30     90 01BE    10$:    MOVB    #^A/0/, -(R6)           ; Move a zero
              52     D7 01C1            DECL    R2                      ; Decrement digits count
```

```
                      EF    11  01C3    506             BRB     ZERO_FILL                       ; Loop till done
                                01C5    507
                                01C5    508 ;+
                                01C5    509 ; ERROR - Return output conversion error
                                01C5    510 ;           Not used by L format.
                                01C5    511 ;-
                                01C5    512
                                01C5    513 ERROR_CALL:                                     ; Called by I format
                              003C  01C5    514             .WORD   ^M<R2,R3,R4,R5>
                                01C7    515
                                01C7    516 ERROR:
                  50    08 BC   7D  01C7    517             MOVQ    @out_string(AP), R0     ; Get string descriptor
       61   50    2A    6E   00   2C  01CB    518             MOVC5   #0, (SP), #^A/*/, R0, (R1)      ; Fill with *
            50   00000000'8F   D0  01D1    519             MOVL    #OTS$_OUTCONERR, R0     ; Output conversion error
                               04  01D8    520             RET
```

```
                 01D9         .SBTTL  OTS$CVT_L_TI - Long to text, I format
                 01D9   ;++
                 01D9   ; FUNCTIONAL DESCRIPTION:
                 01D9   ;
                 01D9   ;       This routine converts its input value to a text representation,
                 01D9   ;       using base 10 (decimal).
                 01D9   ;
                 01D9   ;       OTS$CVT_L_TI supports FORTRAN Iw and Iw.m output conversion.
                 01D9   ;
                 01D9   ;       A separate entry point FOR$CNV_OUT_I is provided for compatibility
                 01D9   ;       with previous releases.  Note that the input value for
                 01D9   ;       OTS$CVT_L_TI is by reference while that for FOR$CNV_OUT_I is
                 01D9   ;       by value.
                 01D9   ;
                 01D9   ; CALLING SEQUENCE:
                 01D9   ;
                 01D9   ;       status.wlc.v = OTS$CVT_L_TI (value.rx.r, out_string.wt.ds
                 01D9   ;                                   [, int_digits.rl.v
                 01D9   ;                                   [, value_size.rl.v
                 01D9   ;                                   [, caller_flags.rbu.v]]])
                 01D9   ;
                 01D9   ;       status.wlc.v = FOR$CNV_OUT_I (value.rl.v, out_string.wt.ds)
                 01D9   ;
                 01D9   ;
                 01D9   ; INPUT PARAMETERS:
                 01D9   ;
00000004         01D9           value = 4                   ; Input value to be converted to text
0000000C         01D9           int_digits = 12             ; Minimum number of digits to be produced.
                 01D9                                       ; If actual number of significant digits
                 01D9                                       ; is smaller, leading zeroes will be
                 01D9                                       ; produced.  If int_digits is zero
                 01D9                                       ; and value is zero, a blank field will
                 01D9                                       ; result.  The default is 1.
00000010         01D9           value_size = 16             ; The size of value in bytes.  If
                 01D9                                       ; present, value_size must be either
                 01D9                                       ; 1, 2 or 4.  If value_size is 1 or 2, the
                 01D9                                       ; value is sign extended to a longword
                 01D9                                       ; before conversion.  The default is
                 01D9                                       ; 4 if this argument is not present.
00000014         01D9           caller_flags = 20           ; Flags supplied by caller:
00000000         01D9             V_FORCEPLUS = 0           ; If set, a plus sign will be forced
                 01D9                                       ; for positive values.
                 01D9   ;
                 01D9   ; IMPLICIT INPUTS:
                 01D9   ;
                 01D9   ;       NONE
                 01D9   ;
                 01D9   ; OUTPUT PARAMETERS:
                 01D9   ;
00000008         01D9           out_string = 8              ; Output string by descriptor.
                 01D9   ;
                 01D9   ; IMPLICIT OUTPUTS:
                 01D9   ;
                 01D9   ;       NONE
                 01D9   ;
                 01D9   ; COMPLETION CODES:
                 01D9   ;
```

```
                          01D9  ;        SS$_NORMAL      - Successful completion.
                          01D9  ;        OTS$_OUTCONERR  - Output conversion error.  Either the converted
                          01D9  ;                          value did not fit in the field provided or the
                          01D9  ;                          byte count was not 1, 2 or 4.  The field is filled
                          01D9  ;                          with asterisks.
                          01D9  ;
                          01D9  ; SIDE EFFECTS:
                          01D9  ;
                          01D9  ;        NONE
                          01D9  ;
                          01D9  ;--
                          01D9
                          01D9  ;+
                          01D9  ; Definition of flag bits.
                          01D9  ;-
                          01D9
              00000008    01D9           V_NEGATIVE = 8                  ; Bit to set in R5 if negative
              00000100    01D9           M_NEGATIVE = 1@V_NEGATIVE       ; Mask for V_NEGATIVE
                          01D9
                  007C    01D9           .ENTRY FOR$CNV_OUT_I, ^M<R2,R3,R4,R5,R6>
                          01DB
           54    01   D0  01DB           MOVL    #1, R4                  ; Number of integer digits
                 55   D4  01DE           CLRL    R5                      ; No flags
      50    04   AC   D0  01E0           MOVL    value(AP), R0           ; Value is in argument list
                 47   11  01E4           BRB     COMMON_I                ; Go to common routine
                          01E6
                  007C    01E6           .ENTRY  OTS$CVT_L_TI, ^M<R2,R3,R4,R5,R6>
                          01E8
                 55   D4  01E8           CLRL    R5                      ; Assume no flags
   03     02     6C   8F  01EA           CASEB   (AP), #2, #3            ; Select on argument count
                 000A'    01EE  1$:      .WORD   20$-1$                  ; 2 arguments
                 0035'    01F0           .WORD   44$-1$                  ; 3 arguments
                 0017'    01F2           .WORD   40$-1$                  ; 4 arguments
                 0013'    01F4           .WORD   50$-1$                  ; 5 arguments
                 09   11  01F6           BRB     50$                     ; assume >5 arguments
                          01F8  20$:
           54    01   D0  01F8           MOVL    #1, R4                  ; Get integer digits
      50    04   BC   D0  01FB           MOVL    @value(AP), R0          ; longword value
                 2C   11  01FF           BRB     COMMON_I
      55    14   AC   90  0201  50$:     MOVB    caller_flags(AP), R5    ; Get flags
04    00    10   AC   CF  0205  40$:     CASEL   value_size(AP), #0, #4  ; Select on value size
                 0019'    020A  2$:      .WORD   44$-2$                  ; 0 - assume 4 bytes
                 000D'    020C           .WORD   41$-2$                  ; 1 byte
                 0013'    020E           .WORD   42$-2$                  ; 2 bytes
                 007D'    0210           .WORD   ERROR_I-2$              ; 3 bytes, error
                 0019'    0212           .WORD   44$-2$                  ; 4 bytes
            0070      31  0214           BRW     ERROR_I                 ; other, error
      50    04   BC   98  0217  41$:     CVTBL   @value(AP), R0          ; Convert byte
                 0A   11  021B           BRB     30$                     ; Continue
      50    04   BC   32  021D  42$:     CVTWL   @value(AP), R0          ; Convert word
                 04   11  0221           BRB     30$                     ; Continue
      50    04   BC   D0  0223  44$:     MOVL    @value(AP), R0          ; Convert longword
      54    0C   AC   D0  0227  30$:     MOVL    int_digits(AP), R4      ; Get integer digits
                 50   D5  022B           TSTL    R0                      ; Set condition codes for test
                          022D
                          022D  COMMON_I:
           08         18  022D           BGEQ    COMMON_IU               ; Skip if value positive
           55  0100 8F   A8  022F   636          BISW2   #M_NEGATIVE, R5          ; Indicate negative
               50    50  CE  0234   637          MNEGL   R0, R0                   ; Get absolute value
                             0237   638 COMMON_IU:
                     51  D4  0237   639          CLRL    R1                       ; Clear low-order part of value
             52    08 BC 7D  0239   640          MOVQ    @out_string(AP), R2      ; Get descriptor in R2-R3
               52    52  3C  023D   641          MOVZWL  R2, R2                   ; Get length in R2
               53    52  C0  0240   642          ADDL2   R2, R3                   ; Get address of last+1 character
                     0F  11  0243   643          BRB     15$                      ; Store digits
                             0245   644
                             0245   645 ;+
                             0245   646 ; Store all significant digits.
                             0245   647 ;-
      56   50    50  0A  7B  0245   648 10$:     EDIV    #10, R0, R0, R6          ; Get quotient in R0-R1, remainder in R6
                     52  D7  024A   649          DECL    R2                       ; Decrement length
                     39  19  024C   650          BLSS    ERROR_I                  ; Error if no more chars left
           73   56   30  81  024E   651          ADDB3   #^A/07, R6, -(R3)        ; Store next digit
                     54  D7  0252   652          DECL    R4                       ; Decrement zero-fill count
                     50  D5  0254   653 15$:     TSTL    R0                       ; Are we done now?
                     ED  12  0256   654          BNEQ    10$                      ; Loop if not
                     07  11  0258   655          BRB     25$                      ; Fill in leading zeroes
                             025A   656
                             025A   657 ;+
                             025A   658 ; Fill in any leading zeroes needed.  R4 has remaining zero count.
                             025A   659 ;-
                     52  D7  025A   660 20$:     DECL    R2                       ; 1 less character
                     29  19  025C   661          BLSS    ERROR_I                  ; Have we run out?
                73   30  90  025E   662          MOVB    #^A/07, -(R3)            ; Move a zero
                  F6 54  F4  0261   663 25$:     SOBGEQ  R4, 20$                  ; Loop till done
                             0264   664
                             0264   665 ;+
                             0264   666 ; Store sign, if needed.
                             0264   667 ;-
             09 55   08  E0  0264   668          BBS     #V_NEGATIVE, R5, 30$     ; Skip if value negative
             14 55   00  E1  0268   669          BBC     #V_FORCEPLUS, R5, 50$    ; Test for forced plus
                50   2B  90  026C   670          MOVB    #^A/+/, R0               ; Use plus sign
                     03  11  026F   671          BRB     35$                      ; Rejoin common code
                50   2D  90  0271   672 30$:     MOVB    #^A/-/, R0               ; Use minus sign
                     52  D7  0274   673 35$:     DECL    R2                       ; 1 less character
                     0F  19  0276   674          BLSS    ERROR_I                  ; Have we run out?
                73   50  90  0278   675          MOVB    R0, -(R3)                ; Move sign, "-" or "+"
                     03  11  027B   676          BRB     50$                      ; Blank fill, if needed.
                             027D   677
                             027D   678 ;+
                             027D   679 ; Blank fill remainder, if needed.  R2 has remaining blank count.
                             027D   680 ;-
                73   20  90  027D   681 40$:     MOVB    #^A/ /, -(R3)            ; Move a blank
                  FA 52  F4  0280   682 50$:     SOBGEQ  R2, 40$                  ; Loop till done
                50   01  D0  0283   683          MOVL    #1, R0                   ; Success
                         04  0286   684          RET
                             0287   685
                             0287   686 ERROR_I:
           FF39 CF   6C  FA  0287   687          CALLG   (AP), W^ERROR_CALL       ; Fill with asterisks
                         04  028C   688          RET                              ; with error status in R0
                  028D        .SBTTL  OTS$CVT_L_TU - Long to text, unsigned decimal format
                  028D  ;++
                  028D  ; FUNCTIONAL DESCRIPTION:
                  028D  ;
                  028D  ;       This routine converts its input value to a text representation,
                  028D  ;       using unsigned base 10 (decimal).
                  028D  ;
                  028D  ; CALLING SEQUENCE:
                  028D  ;
                  028D  ;       status.wlc.v = OTS$CVT_L_TU  (value.rx.r, out_string.wt.ds
                  028D  ;                                   [, int_digits.rl.v
                  028D  ;                                   [, value_size.rl.v
                  028D  ;                                   [, caller_flags.rbu.v]]])
                  028D  ;
                  028D  ;
                  028D  ; INPUT PARAMETERS:
                  028D  ;
         00000004 028D          value = 4               ; Input value to be converted to text
         0000000C 028D          int_digits = 12         ; Minimum number of digits to be produced.
                  028D                                  ; If actual number of significant digits
                  028D                                  ; is smaller, leading zeroes will be
                  028D                                  ; produced.  If int_digits is zero
                  028D                                  ; and value is zero, a blank field will
                  028D                                  ; result.  The default is 1.
         00000010 028D          value_size = 16         ; The size of value in bytes.  If
                  028D                                  ; present, value_size must be either
                  028D                                  ; 1, 2 or 4.  If value_size is 1 or 2, the
                  028D                                  ; value is sign extended to a longword
                  028D                                  ; before conversion.  The default is
                  028D                                  ; 4 if this argument is not present.
         00000014 028D          caller_flags = 20       ; Flags supplied by caller:
                  028D
                  028D  ;
                  028D  ; IMPLICIT INPUTS:
                  028D  ;
                  028D  ;       NONE
                  028D  ;
                  028D  ; OUTPUT PARAMETERS:
                  028D  ;
         00000008 028D          out_string = 8          ; Output string by descriptor.
                  028D  ;
                  028D  ; IMPLICIT OUTPUTS:
                  028D  ;
                  028D  ;       NONE
                  028D  ;
                  028D  ; COMPLETION CODES:
                  028D  ;
                  028D  ;       SS$_NORMAL      - Successful completion.
                  028D  ;       OTS$_OUTCONERR  - Output conversion error.  Either the converted
                  028D  ;                         value did not fit in the field provided or the
                  028D  ;                         byte count was not 1, 2 or 4.  The field is filled
                  028D  ;                         with asterisks.
                  028D  ;
                  028D  ; SIDE EFFECTS:
                  028D  ;
                  028D  ;       NONE
                  028D  ;
```

```
                            028D    747 ;--
                            028D    748
                            028D    749 ;+
                            028D    750 ; Definition of flag bits.
                            028D    751 ;-
                            028D    752
                            028D    753
                     007C   028D    754          .ENTRY  OTS$CVT_L_TU, ^M<R2,R3,R4,R5,R6>
                            028F    755
                  55   D4   028F    756          CLRL    R5                         ; Assume no flags
        03   02   6C   8F   0291    757          CASEB   (AP), #2, #3               ; Select on argument count
                   000A'    0295    758 1$:      .WORD   20$-1$                     ; 2 arguments
                   0032'    0297    759          .WORD   44$-1$                     ; 3 arguments
                   0014'    0299    760          .WORD   40$-1$                     ; 4 arguments
                   0014'    029B    761          .WORD   40$-1$                     ; 5 arguments
                  0A   11   029D    762          BRB     40$                        ; assume >5 arguments
                            029F    763 20$:
             54   01   D0   029F    764          MOVL    #1, R4                     ; Get integer digits
        50   04   BC   D0   02A2    765          MOVL    @value(AP), R0             ; longword value
                FF8E   31   02A6    766          BRW     COMMON_IU
   04   00   10   AC   CF   02A9    767 40$:     CASEL   value_size(AP), #0, #4     ; Select on value size
                   0019'    02AE    768 2$:      .WORD   44$-2$                     ; 0 - assume 4 bytes
                   000D'    02B0    769          .WORD   41$-2$                     ; 1 byte
                   0013'    02B2    770          .WORD   42$-2$                     ; 2 bytes
                   FFD9     02B4    771          .WORD   ERROR_I-2$                 ; 3 bytes, error
                   0019'    02B6    772          .WORD   44$-2$                     ; 4 bytes
                FFCC   31   02B8    773          BRW     ERROR_I                    ; other, error
        50   04   BC   9A   02BB    774 41$:     MOVZBL  @value(AP), R0             ; Convert byte
                  0A   11   02BF    775          BRB     30$                        ; Continue
        50   04   BC   3C   02C1    776 42$:     MOVZWL  @value(AP), R0             ; Convert word
                  04   11   02C5    777          BRB     30$                        ; Continue
        50   04   BC   D0   02C7    778 44$:     MOVL    @value(AP), R0             ; Convert longword
        54   0C   AC   D0   02CB    779 30$:     MOVL    int_digits(AP), R4         ; Get integer digits
                FF65   31   02CF    780          BRW     COMMON_IU                  ; Join common code
```

```
                        02D2          .SBTTL  OTS$CVT_L_TL - Long to text, L format
                        02D2    ;++
                        02D2    ; FUNCTIONAL DESCRIPTION:
                        02D2    ;
                        02D2    ;       This routine converts its input value to a text representation,
                        02D2    ;       using FORTRAN L (logical) format.
                        02D2    ;
                        02D2    ;       The output field will consist of (width-1) blanks followed by:
                        02D2    ;               the letter T if the lowest bit is set;
                        02D2    ;               the letter F if the lowest bit is clear.
                        02D2    ;
                        02D2    ;       A separate entry point FOR$CNV_OUT_L is provided for compatibility
                        02D2    ;       with previous releases.  Note that the input value for
                        02D2    ;       OTS$CVT_L_TL is by reference while that for FOR$CNV_OUT_L is
                        02D2    ;       by value.
                        02D2    ;
                        02D2    ; CALLING SEQUENCE:
                        02D2    ;
                        02D2    ;       status.wlc.v = OTS$CVT_L_TL (value.rL.r, out_string.wt.ds)
                        02D2    ;
                        02D2    ;       status.wlc.v = FOR$CNV_OUT_L (value.rl.v, out_string.wt.ds)
                        02D2    ;
                        02D2    ;
                        02D2    ; INPUT PARAMETERS:
                        02D2    ;
           00000004     02D2            value = 4               ; Input value to be converted to text
                        02D2    ;
                        02D2    ; IMPLICIT INPUTS:
                        02D2    ;
                        02D2    ;       NONE
                        02D2    ;
                        02D2    ; OUTPUT PARAMETERS:
                        02D2    ;
           00000008     02D2            out_string = 8          ; Output string by descriptor.
                        02D2    ;
                        02D2    ; IMPLICIT OUTPUTS:
                        02D2    ;
                        02D2    ;       NONE
                        02D2    ;
                        02D2    ; COMPLETION CODES:
                        02D2    ;
                        02D2    ;       SS$_NORMAL      - Successful completion.
                        02D2    ;       OTS$_OUTCONERR  - Output conversion error.  This can only occur
                        02D2    ;                         if the output string is of zero length.
                        02D2    ;
                        02D2    ; SIDE EFFECTS:
                        02D2    ;
                        02D2    ;       NONE
                        02D2    ;
                        02D2    ;--
                        02D2
               0004     02D2            .ENTRY  FOR$CNV_OUT_L, ^M<R2>
                        02D4
 52    04 AC     90     02D4            MOVB    value(AP), R2           ; Get low byte of value
          06     11     02D8            BRB     COMMON_L                ; Go to common routine
                        02DA
               0004     02DA            .ENTRY OTS$CVT_L_TL , ^M<R2>
```

```
                          02DC    839
        52    04 BC   90  02DC    840            MOVB    @value(AP), R2          ; Get low byte of value
                          02E0    841
                          02E0    842
                          02E0    843 COMMON_L:
        50    08 BC   7D  02E0    844            MOVQ    @out_string(AP), R0     ; Get descriptor
           50    50   3C  02E4    845            MOVZWL  R0, R0                  ; Length
                 1E   15  02E7    846            BLEQ    ERROR_L                 ; Zero length, error
           51    50   C0  02E9    847            ADDL2   R0, R1                  ; 1 byte past end of string
              06 52   E8  02EC    848            BLBS    R2, TRUE                ; TRUE or FALSE?
        71    46 8F   90  02EF    849            MOVB    #^A/F/, -(R1)           ; result is F
                 04   11  02F3    850            BRB     EXIT_L                  ; Finish
        71    54 8F   90  02F5    851 TRUE:      MOVB    #^A/T/, -(R1)           ; result is T
                          02F9    852
                          02F9    853 EXIT_L:
                 50   D7  02F9    854            DECL    R0                      ; 1 less character
                 06   13  02FB    855            BEQL    20$                     ; All done?
           71    20   90  02FD    856 10$:       MOVB    #^A/ /, -(R1)           ; Blank fill
              FA 50   F5  0300    857            SOBGTR  R0, 10$                 ; Loop till done
           50    01   D0  0303    858 20$:       MOVL    #1, R0                  ; Success
                 04       0306    859            RET                             ; Return
                          0307    860
                          0307    861 ERROR_L:
   50   00000000'8F   D0  0307    862            MOVL    #OTS$_OUTCONERR, R0     ; Output conversion error
                      04  030E    863            RET                             ; Return
                          030F    864
                          030F    865
                          030F    866            .END
```

```
BIT_OFFSET       = 00000018
COMMON_I           0000022D R       01
COMMON_IU          00000237 R       01
COMMON_L           000002E0 R       01
COMMON_O           00000027 R       01
COMMON_Z           000000AB R       01
ERROR              000001C7 R       01
ERROR_CALL         000001C5 R       01
ERROR_I            00000287 R       01
ERROR_L            00000307 R       01
EXIT_B             00000159 R       01
EXIT_L             000002F9 R       01
EXIT_O             00000079 R       01
EXIT_Z             00000118 R       01
FLAGS            = 00000014
FOR$CNV_OUT_I      000001D9 RG      01
FOR$CNV_OUT_L      000002D2 RG      01
FOR$CNV_OUT_O      00000010 RG      01
FOR$CNV_OUT_Z      00000094 RG      01
INITIALIZE         00000174 R       01
INT_DIGITS       = 0000000C
LETTERS            00000000 R       01
M_NEGATIVE       = 00000100
OTS$CVT_L_TB       00000133 RG      01
OTS$CVT_L_TI       000001E6 RG      01
OTS$CVT_L_TL       000002DA RG      01
OTS$CVT_L_TO       0000001E RG      01
OTS$CVT_L_TU       0000028D RG      01
OTS$CVT_L_TZ       000000A2 RG      01
OTS$_OUTCONERR     ********    X    00
OUT_STRING       = 00000008
TRUE               000002F5 R       01
VALUE            = 00000004
VALUE_SIZE       = 00000010
V_FORCEPLUS      = 00000000
V_NEGATIVE       = 00000008
V_SIZE_IN_BITS   = 00000002
ZERO_FILL          000001B4 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                           0000030F  (  783.)  01 (  1.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD   NOWRT NOVEC LONG

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

Phase                      Page faults    CPU Time       Elapsed Time
-----                      -----------    --------       ------------
Initialization                  29        00:00:00.06    00:00:02.11
Command processing             106        00:00:00.31    00:00:02.72
Pass 1                         100        00:00:01.23    00:00:03.77
```

```
Symbol table sort                  0      00:00:00.05     00:00:00.26
Pass 2                           154      00:00:01.00     00:00:03.60
Symbol table output                4      00:00:00.03     00:00:00.65
Psect synopsis output              3      00:00:00.01     00:00:00.01
Cross-reference output             0      00:00:00.00     00:00:00.00
Assembler run totals             398      00:00:02.69     00:00:13.13

The working set limit was 1200 pages.
13930 bytes (28 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 39 non-local and 58 local symbols.
866 source lines were read in Pass 1, producing 38 object records in Pass 2.
0 pages of virtual memory were used to define 0 macros.

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

Macro library name                             Macros defined
------------------                             --------------
_$255$DUA28:[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$:OTSCVTLT/OBJ=OBJ$:OTSCVTLT MSRC$:OTSCVTLT/UPDATE=(ENH$:OTSCVTLT)
```

0211 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

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

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