```
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
```

FILEID**OTSCVTTIL

```
  000000    TTTTTTTTTT    SSSSSSSS    CCCCCCCC  VV      VV  TTTTTTTTTT  TTTTTTTTTT    IIIIII    LL
  000000    TTTTTTTTTT    SSSSSSSS    CCCCCCCC  VV      VV  TTTTTTTTTT  TTTTTTTTTT    IIIIII    LL
00      00      TT      SS          CC          VV      VV      TT          TT          II      LL
00      00      TT      SS          CC          VV      VV      TT          TT          II      LL
00      00      TT      SS          CC          VV      VV      TT          TT          II      LL
00      00      TT      SS          CC          VV      VV      TT          TT          II      LL
00      00      TT        SSSSSS    CC          VV      VV      TT          TT          II      LL
00      00      TT        SSSSSS    CC          VV      VV      TT          TT          II      LL
00      00      TT              SS  CC          VV      VV      TT          TT          II      LL
00      00      TT              SS  CC          VV      VV      TT          TT          II      LL
00      00      TT              SS  CC            VV  VV        TT          TT          II      LL          ....
00      00      TT              SS  CC            VV  VV        TT          TT          II      LL          ....
  000000        TT      SSSSSSSS      CCCCCCCC      VV          TT          TT      IIIIII    LLLLLLLLLL  ....
  000000        TT      SSSSSSSS      CCCCCCCC      VV          TT          TT      IIIIII    LLLLLLLLLL  ....

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

Table of contents

```
0000     1          .TITLE  OTS$CVTTIL      ; Convert text (integer) to longword
0000     2          .IDENT  /1-009/         ; File: OTSCVTTIL.MAR  Edit: SBL1009
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28 ;
0000    29 ;
0000    30 ; FACILITY: Language independent support library
0000    31 ;++
0000    32 ; ABSTRACT:
0000    33 ;
0000    34 ; OTS$CVT_TI_L converts a text representation of a decimal value to
0000    35 ; an internal binary form.  It replaces FOR$CNV_IN_I.
0000    36 ;
0000    37 ; OTS$CVT_TU_L converts a text representation of an unsigned decimal value
0000    38 ; to internal binary form.
0000    39 ;--
0000    40 ;
0000    41 ; VERSION: 1
0000    42 ;
0000    43 ; HISTORY:
0000    44 ;
0000    45 ; AUTHOR:
0000    46 ;       Steven B. Lionel, 21-Feb-1979: Version 1
0000    47 ;
0000    48 ; MODIFIED BY:
0000    49 ;
0000    50 ;
0000    51 ;
```

```
0000     53          .SBTTL  HISTORY          ; Detailed Current Edit History
0000     54
0000     55
0000     56 ; Edit History for Version 1 of OTS$CVT_TI_L
0000     57 ;
0000     58 ; 1-001 - Adapted from FOR$CNV_IN_I version 1-009.  SBL 21-Feb-79
0000     59 ; 1-002 - Added V_SKIPTABS.  SBL 11-JUL-1979
0000     60 ; 1-003 - Fix bug in overflow test.  SBL 12-July-1979
0000     61 ; 1-004 - Make V_SKIPTABS bit 4 to conform with floating.  SBL 30-Aug-1979
0000     62 ; 1-005 - Fix bug in tab skipping.  Add standard module headers.
0000     63 ;         SBL 11-Sept-1979
0000     64 ; 1-006 - Do correct thing if value_size is incorrect.  SBL 25-Feb-1980
0000     65 ; 1-007 - REALLY do correct thing if value_size is incorrect.  Previous code
0000     66 ;         went into infinite loop.  SBL 5-Jan-1981
0000     67 ; 1-008 - Give error if character value greater than 127 found.
0000     68 ;         SPR 11-52485  SBL 29-Dec-1982
0000     69 ; 1-009 - Add OTS$CVT_TU_L.  SBL 27-Apr-1983
```

```
                     0000     7          .SBTTL  DECLARATIONS
                     0000    72 ;
                     0000    73 ; INCLUDE FILES:
                     0000    74 ;
                     0000    75
                     0000    76 ;
                     0000    77 ; EXTERNAL SYMBOLS:
                     0000    78 ;
                     0000    79          .DSABL  GBL
                     0000    80          .EXTRN  OTS$_INPCONERR
                     0000    81
                     0000    82 ;
                     0000    83 ; MACROS:
                     0000    84 ;
                     0000    85
                     0000    86 ;
                     0000    87 ; PSECT DECLARATIONS:
                     0000    88 ;
                     0000    89
                00000000     90          .PSECT  _OTS$CODE       PIC, SHR, LONG, EXE, NOWRT
                     0000    91
                     0000    92
                     0000    93 ;
                     0000    94 ; EQUATED SYMBOLS:
                     0000    95 ;
                     0000    96
           0000007C  0000    97          REGMASK         = ^M<R2, R3, R4, R5, R6>
           0000001F  0000    98          V_NEGATIVE      = 31            ; 31th bit position of flag register
           0000001E  0000    99          V_UNSIGNED      = 30            ; indicates OTS$CVT_TU_L
                     0000   100
                     0000   101 ;
                     0000   102 ; OWN STORAGE:
                     0000   103 ;
                     0000   104
```

```
                              0000   106          .SBTTL  OTS$CVT_TU_L    ; convert text (unsigned) to longword
                              0000   107
                              0000   108 ;++
                              0000   109 ;
                              0000   110 ; FUNCTIONAL DESCRIPTION:
                              0000   111 ;
                              0000   112 ;       OTS$CVT_TU_L converts an ASCII string containing a text
                              0000   113 ;       representation of an unsigned decimal number to internal binary form.
                              0000   114 ;
                              0000   115 ;
                              0000   116 ;       The text representation converted is:
                              0000   117 ;               <0 or more blanks>
                              0000   118 ;               <0 or more ASCII digits from "0" through "9">
                              0000   119 ;               <end of string>
                              0000   120 ;
                              0000   121 ;       Notes:
                              0000   122 ;         1.  If caller flag V_SKIPBLANKS is clear, then spaces are
                              0000   123 ;             equivalent to "0".  If set, spaces are ignored.
                              0000   124 ;         2.  If caller flag V_SKIPTABS is clear, then tab characters
                              0000   125 ;             are illegal.  If set, tabs are ignored.
                              0000   126 ;
                              0000   127 ; CALLING SEQUENCE:
                              0000   128 ;
                              0000   129 ;       status.wlc.v = OTS$CVT_TU_L (in_str.rt.dx, value.wx.r
                              0000   130 ;                       [, value_size.rl.v [, caller_flags.rlu.v]])
                              0000   131 ;
                              0000   132 ; INPUT PARAMETERS:
                              0000   133 ;
                     00000004 0000   134          in_str  = 4                    ; Input string by descriptor
                     0000000C 0000   135          value_size = 12                ; Size of value in bytes
                              0000   136                                         ; Must be 1, 2 or 4.
                     00000010 0000   137          caller_flags = 16              ; Caller flags by value
                     00000000 0000   138          V_SKIPBLANKS    = 0            ; If set, blanks are ignored.
                              0000   139                                         ; Else they are treated as
                              0000   140                                         ; zeroes.
                     00000004 0000   141          V_SKIPTABS      = 4            ; If set, tabs are ignored.
                              0000   142                                         ; Else they are invalid.
                              0000   143 ;
                              0000   144 ; IMPLICIT INPUTS:
                              0000   145 ;
                              0000   146 ;       NONE
                              0000   147 ;
                              0000   148 ; OUTPUT PARAMETERS:
                              0000   149 ;
                     00000008 0000   150          value   = 8                    ; Output value by reference
                              0000   151 ;
                              0000   152 ; IMPLICIT OUTPUTS:
                              0000   153 ;
                              0000   154 ;       NONE
                              0000   155 ;
                              0000   156 ; COMPLETION CODES:
                              0000   157 ;
                              0000   158 ;       SS$_NORMAL      - Successful completion
                              0000   159 ;       OTS$_INPCONERR  - There was an invalid character in the input
                              0000   160 ;                         string. the value overflowed the range allowed,
                              0000   161 ;                         or value_size was invalid.  The result "value" is
                              0000   162 ;                         set to zero, unless value_size is invalid, in which
```

```
                     0000   163 ;                           case "value" is unpredictable.
                     00C0   164 ;
                     0000   165 ; SIDE EFFECTS:
                     0000   166 ;
                     0000   167 ;       NONE
                     0000   168 ;
                     0000   169 ;--
                     0000   170
                 007C 0000   171         .ENTRY  OTS$CVT_TU_L, REGMASK
              56   D4 0002   172         CLRL    R6                      ; Clear flags mask
           04 56   1E E3 0004   173         BBCS    #V_UNSIGNED, R6, COMMON_ENTRY ; Set UNSIGNED and join common code
                     0008   174
```

```
                     0008   176          .SBTTL  OTS$CVT_TI_L    ; convert text (integer) to longword
                     0008   177
                     0008   178 ;++
                     0008   179 ;
                     0008   180 ; FUNCTIONAL DESCRIPTION:
                     0008   181 ;
                     0008   182 ;       OTS$CVT_TI_L converts an ASCII string containing a text
                     0008   183 ;       representation of a decimal number to internal binary form.
                     0008   184 ;
                     0008   185 ;       This routine supports FORTRAN I input format conversion as well
                     0008   186 ;       as similar types for other languages.
                     0008   187 ;
                     0008   188 ;       The text representation converted is:
                     0008   189 ;               <0 or more blanks>
                     0008   190 ;               <"+", "-" or nothing>
                     0008   191 ;               <0 or more ASCII digits from "0" through "9">
                     0008   192 ;               <end of string>
                     0008   193 ;
                     0008   194 ;       Notes:
                     0008   195 ;         1.  If caller flag V_SKIPBLANKS is clear, then spaces are
                     0008   196 ;             equivalent to "0".  If set, spaces are ignored.
                     0008   197 ;         2.  If caller flag V_SKIPTABS is clear, then tab characters
                     0008   198 ;             are illegal.  If set, tabs are ignored.
                     0008   199 ;
                     0008   200 ; CALLING SEQUENCE:
                     0008   201 ;
                     0008   202 ;       status.wlc.v = OTS$CVT_TI_L (in_str.rt.dx, value.wx.r
                     0008   203 ;                       [, value_size.rl.v [, caller_flags.rlu.v]])
                     0008   204 ;
                     0008   205 ; INPUT PARAMETERS:
                     0008   206 ;
00000004             0008   207          in_str  = 4                    ; Input string by descriptor
0000000C             0008   208          value_size = 12                ; Size of value in bytes
                     0008   209                                         ; Must be 1, 2 or 4.
00000010             0008   210          caller_flags = 16              ; Caller flags by value
00000000             0008   211          V_SKIPBLANKS    = 0            ; If set, blanks are ignored.
                     0008   212                                         ; Else they are treated as
                     0008   213                                         ; zeroes.
00000004             0008   214          V_SKIPTABS      = 4            ; If set, tabs are ignored.
                     0008   215                                         ; Else they are invalid.
                     0008   216 ;
                     0008   217 ; IMPLICIT INPUTS:
                     0008   218 ;
                     0008   219 ;       NONE
                     0008   220 ;
                     0008   221 ; OUTPUT PARAMETERS:
                     0008   222 ;
00000008             0008   223          value   = 8                    ; Output value by reference
                     0008   224 ;
                     0008   225 ; IMPLICIT OUTPUTS:
                     0008   226 ;
                     0008   227 ;       NONE
                     0008   228 ;
                     0008   229 ; COMPLETION CODES:
                     0008   230 ;
                     0008   231 ;       SS$_NORMAL      - Successful completion
                     0008   232 ;       OTS$_INPCONERR  - There was an invalid character in the input
```

```
                                0008   233 ;                           string, the value overflowed the range allowed,
                                0008   234 ;                           or value_size was invalid.  The result "value" is
                                0008   235 ;                           set to zero, unless value_size is invalid, in which
                                0008   236 ;                           case "value" is unpredictable.
                                0008   237 ;
                                0008   238 ; SIDE EFFECTS:
                                0008   239 ;
                                0008   240 ;       NONE
                                0008   241 ;
                                0008   242 ;--
                                0008   243
                                0008   244 FOR$CNV_IN_I::                                ; For compatibility
                         007C   0008   245         .ENTRY  OTS$CVT_TI_L, REGMASK
                                000A   246
                      56   D4   000A   247         CLRL    R6                            ; clear flags
                                000C   248 COMMON_ENTRY:
                50    04 BC   7D   000C   249         MOVQ    @in_str(AP), R0               ; R0 = width of the input string
                                0010   250                                               ; R1 = address of the input string
                      54   7C   0010   251         CLRQ    R4                            ; R4/R5 = ACC = 0
                   04   6C   91   0012   252         CMPB    (AP), #<caller_flags/4>       ; Optional argument present?
                      04   19   0015   253         BLSS    5$                            ; No
                56    10 AC   90   0017   254         MOVB    caller_flags(AP), R6          ; Yes, move it
                                001B   255
                                001B   256
                                001B   257 ;+
                                001B   258 ; Find the first non-blank character.  Process the sign, if present,
                                001B   259 ; and if we are not doing unsigned conversion.
                                001B   260 ;-
                                001B   261
          61    50    20   3B   001B   262 5$:     SKPC    #^A/ /, R0, (R1)              ; skip blanks
                                001F   263                                               ; R0 = #CHAR_REMAINING
                                001F   264                                               ; R1 = POINTER_TO_INPUT
                                001F   265                                               ; Z bit is set if R0 = 0
                      61   13   001F   266         BEQL    DONE                          ; branch to DONE if no non-blank
             0B       56   04   E1   0021   267         BBC     #V_SKIPTABS, R6, 7$           ; Ignoring tabs?
                09    61   91   0025   268         CMPB    (R1), #^X09                   ; Yes, is it a tab?
                      06   12   0028   269         BNEQ    7$                            ; If not, continue
                      51   D6   002A   270         INCL    R1                            ; Bump pointer
                      50   D7   002C   271         DECL    R0                            ; Decrement counter
                      EB   11   002E   272         BRB     5$                            ; Look for more.
             12       56   1E   E0   0030   273 7$:     BBS     #V_UNSIGNED, R6, DIGIT_LOOP ; If unsigned, skip sign test
                2D    61   91   0034   274         CMPB    (R1), #^A/-/                  ; is the current char a "-" sign?
                      04   12   0037   275         BNEQ    10$                           ; no, branch to 10$
             05       56   1F   E3   0039   276         BBCS    #V_NEGATIVE, R6, DECIMAL
                                003D   277                                               ; set negative flag and continue
                                003D   278
                2B    61   91   003D   279 10$:    CMPB    (R1), #^A/+/                  ; is current char a "+" sign?
                      04   12   0040   280         BNEQ    DIGIT_LOOP                    ; no, branch to check if it is a digit
                                0042   281
                                0042   282 ;+
                                0042   283 ; skip over "-" or "+" sign
                                0042   284 ;-
                                0042   285
                                0042   286 DECIMAL:
                      50   D7   0042   287         DECL    R0                            ; R0 = #CHAR_REMAINING
                      51   D6   0044   288         INCL    R1                            ; R1 = POINTER_TO_INPUT
```

```
                     0046   290 ;+
                     0046   291 ; Loop to collect digits, treat blanks as zeroes, until the string is exhausted
                     0046   292 ; then branch to DONE
                     0046   293 ;-
                     0046   294
                     0046   295 DIGIT_LOOP:
           50   D7   0046   296         DECL    R0                      ; R0 = #CHAR_REMAINING
           38   19   0048   297         BLSS    DONE                    ; branch to DONE if the string is exhausted
                     004A   298
                     004A   299 ;+
                     004A   300 ; Get next character, converting blanks into zeroes unless V_SKIPBLANKS set.
                     004A   301 ;-
                     004A   302
        53   81   9A   004A   303         MOVZBL  (R1)+, R3               ; get current char and adjust POINTER_TO_INP
        20   53   91   004D   304         CMPB    R3, #^A/ /              ; compare char with blank
             0B   13   0050   305         BEQL    10$                     ; possibly ignore or set to 0
     0E 56   04   E1   0052   306         BBC     #V_SKIPTABS, R6, CHECK_DIGIT    ; not ignoring tabs?
        09   53   91   0056   307         CMPB    R3, #^X09               ; Tab?
             EB   13   0059   308         BEQL    DIGIT_LOOP              ; Yes, ignore it
             07   11   005B   309         BRB     CHECK_DIGIT             ; Continue
     E5 56   00   E0   005D   310 10$:    BBS     #V_SKIPBLANKS, R6, DIGIT_LOOP   ; ignore if V_SKIPBLANKS set
        53   30   D0   0061   311         MOVL    #^A/0/, R3              ; convert blank into zero
                     0064   312
                     0064   313 ;+
                     0064   314 ; Check if current char is a legal digit, accumulate it in ACC if yes and
                     0064   315 ; then branch to DIGIT_LOOP if no overflow. Otherwise fall into ERROR.
                     0064   316 ;-
                     0064   317
                     0064   318 CHECK_DIGIT:
             53   30   C2   0064   319         SUBL    #^A/0/, R3              ; R3 = ASCII(current_char) - ASCII('0')
             0E   19   0067   320         BLSS    ERROR                   ; Error if less than "0"
             09   53   D1   0069   321         CMPL    R3, #9                  ; Is it greater than "9"?
             09   14   006C   322         BGTR    ERROR                   ; If so, error
     54   53   54   0A   7A   006E   323         EMUL    #10, R4, R3, R4         ; #10 = radix
                     0073   324                                             ; R4 = LP(ACC), only LP(ACC) will be used in
                     0073   325                                             ; since R5 (=HP(ACC)) must be zero
                     0073   326                                             ; R3 = current digit
                     0073   327                                             ; R4/R5 = ACC = ACC * radix + current_digit
             55   D5   0073   328         TSTL    R5                      ; compare R5 with 0, since a non-zero value
                     0075   329                                             ; in HP(ACC) meand overflow
             CF   13   0075   330         BEQL    DIGIT_LOOP              ; if no overflow branch back to get more
                     0077   331                                             ; character. Otherwise fall into ERROR
```

```
                                0077   333 ;+
                                0077   334 ; ERROR return
                                0077   335 ;-
                                0077   336
50   00000000'8F    D0          0077   337 ERROR:  MOVL    #OTS$_INPCONERR, R0     ; R0 = error return code
                    54    D4    007E   338          CLRL    R4                      ; zero result
                    21    11    0080   339          BRB     EXIT                    ; exit with zero and error
                                0082   340
                                0082   341 ;+
                                0082   342 ; DONE
                                0082   343 ;-
                                0082   344
                 50   01    D0  0082   345 DONE:   MOVL    #1, R0                  ; return function value of SS$_NORMAL
              12 56   1F    E1  0085   346          BBC     #V_NEGATIVE, R6, 10$    ; branch if "-" wasn't seen
      80000000 8F     54    D1  0089   347          CMPL    R4,#^X80000000          ; is it 2**31?
                      11    13  0090   348          BEQL    EXIT                    ; yes, already correct!
                      54    D5  0092   349          TSTL    R4                      ; test for overflow
                      E1    19  0094   350          BLSS    ERROR                   ; if already negative, overflow
                 54   54    CE  0096   351          MNEGL   R4, R4                  ; answer is -R4
                      08    11  0099   352          BRB     EXIT                    ; Store result
              04 56   1E    E0  009B   353 10$:     BBS     #V_UNSIGNED, R6, EXIT   ; Skip overflow test if unsigned
                      54    D5  009F   354          TSTL    R4                      ; Overflow?
                      D4    19  00A1   355          BLSS    ERROR                   ; If negative, yes
                 03   6C    91  00A3   356 EXIT:   CMPB    (AP), #<value_size/4>   ; Is arg present?
                      22    19  00A6   357          BLSS    40$                     ; If not, assume longword
              52   0C AC    D0  00A8   358          MOVL    value_size(AP), R2      ; Get value size in R2
                      1C    13  00AC   359          BEQL    40$                     ; If zero, assume 4
                 04   52    D1  00AE   360          CMPL    R2, #4                  ; Is it a longword?
                      17    13  00B1   361          BEQL    40$                     ; Yes
                 02   52    D1  00B3   362          CMPL    R2, #2                  ; Word?
                      0A    13  00B6   363          BEQL    20$                     ; Yes
                      15    1A  00B8   364          BGTRU   BADSIZE                 ; Byte if LSS, bad otherwise
              08 BC   54    F6  00BA   365          CVTLB   R4, @value(AP)          ; Convert byte
                      B7    1D  00BE   366          BVS     ERROR                   ; Overflow?
                      0C    11  00C0   367          BRB     50$                     ; No, exit
              08 BC   54    F7  00C2   368 20$:     CVTLW   R4, @value(AP)          ; Convert to word
                      AF    1D  00C6   369          BVS     ERROR                   ; Overflow?
                      04    11  00C8   370          BRB     50$                     ; No, exit
              08 BC   54    D0  00CA   371 40$:     MOVL    R4, @value(AP)          ; Move longword
                      04    00CE   372 50$:     RET
                                00CF   373
                                00CF   374 ;+
                                00CF   375 ; Come here when value_size is incorrect.
                                00CF   376 ;-
                                00CF   377
                                00CF   378 BADSIZE:
50   00000000'8F    D0          00CF   379          MOVL    #OTS$_INPCONERR, R0     ; Input conversion error
                    04    00D6   380          RET
                                00D7   381
                                00D7   382          .END
```

```
BADSIZE             000000CF R      01
CALLER_FLAGS    =   00000010
CHECK_DIGIT         00000064 R      01
COMMON_ENTRY        0000000C R      01
DECIMAL             00000042 R      01
DIGIT_LOOP          00000046 R      01
DONE                00000082 R      01
ERROR               00000077 R      01
EXIT                000000A3 R      01
FOR$CNV_IN_I        00000008 RG     01
IN_STR          =   00000004
OTS$CVT_TI_L        00000008 RG     01
OTS$CVT_TU_L        00000000 RG     01
OTS$_INPCONERR      ********   X    00
REGMASK         =   0000007C
VALUE           =   00000008
VALUE_SIZE      =   0000000C
V_NEGATIVE      =   0000001F
V_SKIPBLANKS    =   00000000
V_SKIPTABS      =   00000004
V_UNSIGNED      =   0000001E
```

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	000000D7 (215.)	01 (1.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.04	00:00:02.60
Command processing	107	00:00:00.32	00:00:02.57
Pass 1	75	00:00:00.52	00:00:02.71
Symbol table sort	0	00:00:00.01	00:00:00.01
Pass 2	74	00:00:00.46	00:00:02.43
Symbol table output	4	00:00:00.02	00:00:00.02
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	294	00:00:01.38	00:00:10.35

```
The working set limit was 1050 pages.
5207 bytes (11 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 21 non-local and 8 local symbols.
382 source lines were read in Pass 1, producing 14 object records in Pass 2.
0 pages of virtual memory were used to define 0 macros.
```

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYSLIB]STARLET.MLB;2	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

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

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