```
  CCCCCCCCCCCC     OOOOOOOOO    BBBBBBBBBBBB    RRRRRRRRRRRR    TTTTTTTTTTTTTTT   LLL
  CCCCCCCCCCCC     OOOOOOOOO    BBBBBBBBBBBB    RRRRRRRRRRRR    TTTTTTTTTTTTTTT   LLL
  CCCCCCCCCCCC     OOOOOOOOO    BBBBBBBBBBBB    RRRRRRRRRRRR    TTTTTTTTTTTTTTT   LLL
CCC              OOO       OOO  BBB       BBB   RRR       RRR         TTT         LLL
CCC              OOO       OOO  BBB       BBB   RRR       RRR         TTT         LLL
CCC              OOO       OOO  BBB       BBB   RRR       RRR         TTT         LLL
CCC              OOO       OOO  BBB       BBB   RRR       RRR         TTT         LLL
CCC              OOO       OOO  BBB       BBB   RRR       RRR         TTT         LLL
CCC              OOO       OOO  BBBBBBBBBBBB    RRRRRRRRRRRR          TTT         LLL
CCC              OOO       OOO  BBBBBBBBBBBB    RRRRRRRRRRRR          TTT         LLL
CCC              OOO       OOO  BBBBBBBBBBBB    RRRRRRRRRRRR          TTT         LLL
CCC              OOO       OOO  BBB       BBB   RRR   RRR             TTT         LLL
CCC              OOO       OOO  BBB       BBB   RRR   RRR             TTT         LLL
CCC              OOO       OOO  BBB       BBB   RRR   RRR             TTT         LLL
CCC              OOO       OOO  BBB       BBB   RRR      RRR          TTT         LLL
CCC              OOO       OOO  BBB       BBB   RRR      RRR          TTT         LLL
CCC              OOO       OOO  BBB       BBB   RRR      RRR          TTT         LLL
  CCCCCCCCCCCC     OOOOOOOOO    BBBBBBBBBBBB    RRR       RRR         TTT         LLLLLLLLLLLLLLL
  CCCCCCCCCCCC     OOOOOOOOO    BBBBBBBBBBBB    RRR       RRR         TTT         LLLLLLLLLLLLLLL
  CCCCCCCCCCCC     OOOOOOOOO    BBBBBBBBBBBB    RRR       RRR         TTT         LLLLLLLLLLLLLLL
```

FILEID**COBACCECV

```
  CCCCCCCC    OOOOOO    BBBBBBBB      AAAAAA      CCCCCCCC    CCCCCCCC  EEEEEEEEEE    CCCCCCCC  VV      VV
  CCCCCCCC    OOOOOO    BBBBBBBB      AAAAAA      CCCCCCCC    CCCCCCCC  EEEEEEEEEE    CCCCCCCC  VV      VV
CC          OO      OO  BB      BB  AA      AA  CC          CC          EE          CC          VV      VV
CC          OO      OO  BB      BB  AA      AA  CC          CC          EE          CC          VV      VV
CC          OO      OO  BB      BB  AA      AA  CC          CC          EE          CC          VV      VV
CC          OO      OO  BB      BB  AA      AA  CC          CC          EE          CC          VV      VV
CC          OO      OO  BBBBBBBB    AA      AA  CC          CC          EEEEEEEE    CC          VV      VV
CC          OO      OO  BBBBBBBB    AA      AA  CC          CC          EEEEEEEE    CC          VV      VV
CC          OO      OO  BB      BB  AAAAAAAAAA  CC          CC          EE          CC          VV      VV
CC          OO      OO  BB      BB  AAAAAAAAAA  CC          CC          EE          CC          VV      VV
CC          OO      OO  BB      BB  AA      AA  CC          CC          EE          CC            VV  VV    ::::
CC          OO      OO  BB      BB  AA      AA  CC          CC          EE          CC            VV  VV    ::::
  CCCCCCCC    OOOOOO    BBBBBBBB    AA      AA    CCCCCCCC    CCCCCCCC  EEEEEEEEEE    CCCCCCCC      VV      ::::
  CCCCCCCC    OOOOOO    BBBBBBBB    AA      AA    CCCCCCCC    CCCCCCCC  EEEEEEEEEE    CCCCCCCC      VV      ::::

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
```

```
0 %TITLE 'COB$ACCECV - ACCEPT Conversion routines'
0 MODULE COB$ACCECV (
0                       IDENT = '1-001'         ! File: COBACCECV.B32 EDIT:LGB1001
0                       ) =
1 BEGIN
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! FACILITY:  COBOL SUPPORT
1 !
1 ! ABSTRACT:
1 !
1 !       Supports the COBOL ACCEPT statement.
1 !
1 ! ENVIRONMENT:  VAX-11 User Mode
1 !
1 ! AUTHOR: Linda Baillie, CREATION DATE: 7-FEB-84
1 !
1 ! MODIFIED BY:
1 !
1 ! 1-001 - Original.  LGB 7-FEB-84
1 !--
```

```
!
! PROLOGUE FILE
!
REQUIRE 'RTLIN:COBPROLOG' ;                         ! Switches, Psects, Include
                                                    ! files
!
! LINKAGES:
!
LINKAGE
        JSB_678 = JSB
                (REGISTER = 6, REGISTER = 7, REGISTER = 8):
                NOPRESERVE (2, 3, 4, 5, 6, 7, 8)
                NOTUSED (9, 10, 11),

        JSB_6789 = JSB
                (REGISTER = 6, REGISTER = 7, REGISTER = 8, REGISTER = 9):
                NOPRESERVE (2, 3, 4, 5, 6, 7, 8, 9)
                NOTUSED (10, 11);
!
! TABLE OF CONTENTS:
!
FORWARD ROUTINE
        COB$$ACC_CONVERT,                           ! Conversion routine
        COB$$NUMERIC_CONV,                          ! Convert to numeric text strings
        COB$$COMP_CONV,                             ! Convert to Word, Longword,
                                                    ! Quadword and Packed strings
        COB$$FLOAT_CONV,                            ! Convert to Floating and Double
                                                    ! Floating Point strings
        COB$$STRIP_BLANKS_SIGN,                     ! Strip blanks and sign from
                                                    ! input numeric string
        COB$$ZERO_FILL          :  NOVALUE,         ! Initialize STRING_DEST with
                                                    ! zeroes
        COB$$VERIFY_FL_RANGE,                       ! Check that input for Floating
                                                    ! Point data items is within range
        COB$$SCAN_INPUT ;                           ! Scan input data

!
! EQUATED SYMBOLS
!
LITERAL
        V_DEC_PT = 64 ;                             ! Bit flag for 'DECIMAL POINT
                                                    ! IS COMMA'
!
! EXTERNAL REFERENCES:
!
EXTERNAL ROUTINE

        COB$CVTIL_R8: JSB_678,                      ! Convert CIT to long
        COB$CVTIP_R9: JSB_6789,                     ! Convert CIT to packed
        COB$CVTIQ_R8: JSB_678,                      ! Convert CIT to quad
        COB$CVTIW_R8: JSB_678,                      ! Convert CIT to word
        COB$CVTTI_R8: JSB_678,                      ! Convert text to CIT
        LIB$STOP : NOVALUE,                         ! Signals fatal error
        STR$GET1_DX,                                ! Allocate a string
        STR$DUPL_CHAR,                              ! Duplicate character n times
        STR$FREET_DX,                               ! Deallocate a string
        STR$COPY_R,                                 ! Copy a string by ref
```

```
1            COB$$FREE_STRINGS,                         ! Free local strings
1            OTS$CVT_T_F,                               ! Convert Text to Floating point
1            OTS$CVT_T_D ;                              ! Convert Text to Double Fl point
1
1 EXTERNAL LITERAL
1            COB$_INVARG ;                              ! Invalid Argument(s)
```

```
%SBTTL 'COB$$ACC_CONVERT - Conversion'
GLOBAL ROUTINE COB$$ACC_CONVERT (STRING_DEST   :  REF $STR$DESCRIPTOR,
                                                  ! Final destination for input chars
                         FLAGS,                   ! Enhancement flag
                         DEFAULT      :  REF $STR$DESCRIPTOR,
                                                  ! Input default value
                         PUT_HERE     :  REF BLOCK [8, BYTE],
                                                  ! Contains input characters
                         CHARS_READ,              ! # of input characters
                         YES_DEFAULT,             ! =1 if DEFAULT was used
                         YES_SIGN                 ! =1 if sign should be included
                        ) =
!++
! FUNCTIONAL DESCRIPTION:
!
!       Convert TEXT input string to specified VAX COBOL data type.
!       This routine selects the appropriate routine to convert the specified
!       data type.
!
! FORMAL PARAMETERS:
!
!
!       STRING_DEST.mt.ds   Address of descriptor to receive the read input.
!
!       FLAGS.rlu.v     Screen enhancement flag;
!
!       DEFAULT.rt.dx   Default source moved to destination descriptor
!                       (STRING_DEST) in the event of null input.
!
!       PUT_HERE.rt.dx  Buffer to hold input characters.
!
!       CHARS_READ.rlu.v   Number of characters accepted as input.
!
!       YES_DEFAULT.rlu.v  Flag = 1 if DEFAULT used because of null input.
!
!       YES_SIGN.rlu.v  Flag = 1 if sign should be included in COMP or COMP3
!                       data type.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
!
!       1 - Conversion Success
!       0 - Conversion Failure
!
! SIDE EFFECTS:
!
!       Signals COB$_INVARG if the syntax of the number is wrong,
!--

    BEGIN
```

```
        LOCAL
            ERROR          :  INITIAL (0),                  ! = 1, Conversion error
            PUT_HERE_BUF   :  REF VECTOR [1100, BYTE],      ! Temp for special case
                                                            ! Conversion check
            CONV_OK        :  INITIAL (0) ;                 ! Return Status

            !+
            !  This validity check does not apply to TEXT, FLOATING or
            !  DOUBLE FLOATING POINT data types.
            !-
            IF ( .STRING_DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_T AND
                 .STRING_DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_F AND
                 .STRING_DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_D )
            THEN
                BEGIN                                       ! Begin special case
                !+
                !  Looking for two special case conversion errors not caught
                !  in other conversion routines.
                !  Check for invalid value  '.-9' and '0000-1234'
                !  ('9-.' found in COB$$NUMERIC_CONV).
                !-

                IF .YES_DEFAULT                             ! Use local buffer for
                THEN                                        ! convenience
                    PUT_HERE_BUF = .DEFAULT [DSC$A_POINTER]
                ELSE
                    PUT_HERE_BUF = .PUT_HERE [DSC$A_POINTER] ;

                INCR X FROM 0 TO .CHARS_READ - 1 DO
                    BEGIN
                    !+
                    !  Work through characters one at a time.
                    !-
                    SELECTONE .PUT_HERE_BUF [.X] OF
                        SET

                        [%C'0' TO %C'9'] :
                            0 ;                             ! Legal
                        [%C' '] :
                            0 ;                             ! Legal

                        [%C'-', %C'+'] :
                            IF .X NEQ 0 AND .X NEQ .CHARS_READ - 1
                            THEN
                                !+
                                ! Looking for 0000-1234 case - a sign in middle
                                ! of digits.  Sign should be first or last
                                ! character.  Note: bbbb-1234 is legal.
                                !-
                                IF (( .PUT_HERE_BUF [.X-1] GEQ %C'0'  AND
                                      .PUT_HERE_BUF [.X-1] LEQ %C'9' ) AND
                                    ( .PUT_HERE_BUF [.X+1] GEQ %C'0'  AND
                                      .PUT_HERE_BUF [.X+1] LEQ %C'9' ))
                                THEN
                                    BEGIN
                                    ERROR = 1 ;
                                    EXITLOOP ;
```

```
4                                   END ;
4
4                       [%C'.', %C','] :
5                           BEGIN
5                           IF .X NEQ .CHARS_READ - 1
5                           THEN
5                               !+
5                               ! Looking for .-9 case - a sign separating a
5                               ! decimal point and digit.  -.9 and .9- are
5                               ! acceptable.  (9-. caught by COB$$NUMERIC_CONV)
5                               !-
5                               IF .PUT_HERE_BUF [.X+1] EQL %C'-'  OR
5                                  .PUT_HERE_BUF [.X+1] EQL %C' '
5                               THEN
6                                   BEGIN
6                                   ERROR = 1 ;
6                                   EXITLOOP ;
5                                   END ;
4                           END ;
4
4                       [ OTHERWISE ]:
4                           0;                          ! Let conversion routines
4                                                       ! handle other errors
4                       TES ;
3                   END ;
2               END ;                                   ! End special case
2
2               IF .ERROR
2               THEN
2                   CONV_OK = 0                         ! Return failure status
2               ELSE
2               !+
2               ! Continue with rest of conversion check.  Call appropriate
2               ! routine (determined by data type).
2               !-
2               CASE .STRING_DEST [DSC$B_DTYPE] FROM DSC$K_DTYPE_WU TO
2                                                   DSC$K_DTYPE_P OF
2                   SET
2
2                   [DSC$K_DTYPE_NU,                    ! Numeric string
2                    DSC$K_DTYPE_NL, DSC$K_DTYPE_NR,
2                    DSC$K_DTYPE_NLO, DSC$K_DTYPE_NRO] :
2
3                       BEGIN
4                       CONV_OK = COB$$NUMERIC_CONV (( IF .YES_DEFAULT
4                                                   THEN .DEFAULT
3                                                   ELSE .PUT_HERE ),
3                               .STRING_DEST, .CHARS_READ, .FLAGS ) ;
2                       END ;
2
2                   [DSC$K_DTYPE_W, DSC$K_DTYPE_WU,     ! Word
2                    DSC$K_DTYPE_L, DSC$K_DTYPE_LU,     ! Longword
2                    DSC$K_DTYPE_Q, DSC$K_DTYPE_QU,     ! Quadword
2                    DSC$K_DTYPE_P] :                   ! Packed Decimal
2
3                       BEGIN
3                       CONV_OK = COB$$COMP_CONV ( .STRING_DEST,
```

```
                                        .FLAGS, .DEFAULT, .PUT_HERE,
                                        .CHARS_READ, .YES_DEFAULT,
                                        .YES_SIGN  ) ;
                END ;

            [DSC$K_DTYPE_F, DSC$K_DTYPE_D] :      ! Floating and Double
                                                  ! Floating Point
                BEGIN
                !+
                !   Since DEFAULT is Read-only, copy it to PUT_HERE if
                !   it was used, just in case it is necessary to WRITE
                !   a DOT to override a COMMA in routine COB$$FLOAT_CONV.
                !-

                IF .YES_DEFAULT
                THEN
                    CH$MOVE ( .CHARS_READ, .DEFAULT [DSC$A_POINTER],
                                      .PUT_HERE [DSC$A_POINTER] ) ;

                CONV_OK = COB$$FLOAT_CONV ( .STRING_DEST, .FLAGS,
                                        .PUT_HERE, .CHARS_READ ) ;
                END ;

            [DSC$K_DTYPE_T] :                       ! Text

                !+
                !   Copy ACCEPTed data to STRING_DEST.  If more chars
                !   are ACCEPTed than STRING_DEST[DSC$W_LENGTH] can
                !   handle, accept only the leftmost characters and
                !   ignore the extra characters.
                !   Use STR$COPY because it BLANK fills.
                !-

                BEGIN
                    LOCAL
                        COPY_NUM ;

                IF .CHARS_READ LSS .STRING_DEST[DSC$W_LENGTH]
                THEN
                    COPY_NUM = .CHARS_READ
                ELSE
                    COPY_NUM = .STRING_DEST[DSC$W_LENGTH] ;

                STR$COPY_R ( .STRING_DEST, COPY_NUM,
                            (IF .YES_DEFAULT
                            THEN .DEFAULT [DSC$A_POINTER]
                            ELSE .PUT_HERE [DSC$A_POINTER] )) ;
                CONV_OK = 1 ;                       ! no need to conv TEXT
                END ;

            [INRANGE, OUTRANGE] :

                LIB$STOP ( COB$_INVARG ) ;

            TES ;

        RETURN .CONV_OK ;
```

```
; 339       1853  1     END ;                                        ! End of COB$$ACC_CONVERT

                                                  .TITLE  COB$ACCECV COB$ACCECV - ACCEPT Conversion routi
;                                                                        nes
                                                  .IDENT  \1-001\

                                                  .EXTRN  COB$CVTIL_R8, COB$CVTIP_R9
                                                  .EXTRN  COB$CVTIQ_R8, COB$CVTIW_R8
                                                  .EXTRN  COB$CVTTI_R8, LIB$STOP
                                                  .EXTRN  STR$GET1_DX, STR$DUPL_CHAR
                                                  .EXTRN  STR$FREE1_DX, STR$COPY_R
                                                  .EXTRN  COB$$FREE_STRINGS
                                                  .EXTRN  OTS$CVT_T_F, OTS$CVT_T_D
                                                  .EXTRN  COB$_INVARG

                                                  .PSECT  _COB$CODE,NOWRT, SHR, PIC,2

                              00FC 00000          .ENTRY  COB$$ACC_CONVERT, Save R2,R3,R4,R5,R6,R7   ; 1626
              5E          04    C2 00002          SUBL2   #4, SP                                     ;
                          53    D4 00005          CLRL    ERROR                                      ; 1681
                          57    D4 00007          CLRL    CONV_OK                                    ;
              56    04    AC    D0 00009          MOVL    STRING_DEST, R6                            ; 1692
              0E    02    A6    91 0000D          CMPB    2(R6), #14                                 ;
                          0A    13 00011          BEQL    1$                                         ;
              0A    02    A6    91 00013          CMPB    2(R6), #10                                 ; 1693
                          04    13 00017          BEQL    1$                                         ;
              0B    02    A6    91 00019          CMPB    2(R6), #11                                 ; 1694
                          03    12 0001D  1$:     BNEQ    2$                                         ;
                        008C    31 0001F          BRW     12$                                        ;
              06    18    AC    E9 00022  2$:     BLBC    YES_DEFAULT, 3$                            ; 1704
              50    0C    AC    D0 00026          MOVL    DEFAULT, R0                                ; 1706
                          04    11 0002A          BRB     4$                                         ;
              50    10    AC    D0 0002C  3$:     MOVL    PUT_HERE, R0                               ; 1708
              50    04    A0    D0 00030  4$:     MOVL    4(R0), PUT_HERE_BUF                        ;
              51          01    CE 00034          MNEGL   #1, X                                      ; 1715
                          70    11 00037          BRB     11$                                        ;
              52        6140    9A 00039  5$:     MOVZBL  (X)[PUT_HERE_BUF], R2                      ;
              30          52    91 0003D          CMPB    R2, #48                                    ; 1718
                          05    1F 00040          BLSSU   6$                                         ;
              39          52    91 00042          CMPB    R2, #57                                    ;
                          62    1B 00045          BLEQU   11$                                        ;
              20          52    91 00047  6$:     CMPB    R2, #32                                    ; 1720
                          5D    13 0004A          BEQL    11$                                        ;
              2B          52    91 0004C          CMPB    R2, #43                                    ; 1723
                          05    13 0004F          BEQL    7$                                         ;
              2D          52    91 00051          CMPB    R2, #45                                    ;
                          2C    12 00054          BNEQ    8$                                         ;
                          51    D5 00056  7$:     TSTL    X                                          ; 1724
                          4F    13 00058          BEQL    11$                                        ;
     52       14    AC    01    C3 0005A          SUBL3   #1, CHARS_READ, R2                         ;
              52          51    D1 0005F          CMPL    X, R2                                      ;
                          45    13 00062          BEQL    11$                                        ;
              30     FF A140    91 00064          CMPB    -1(X)[PUT_HERE_BUF], #48                   ; 1731
                          3E    1F 00069          BLSSU   11$                                        ;
              39     FF A140    91 0006B          CMPB    -1(X)[PUT_HERE_BUF], #57                   ; 1732
                          37    1A 00070          BGTRU   11$                                        ;
```

```
                              30        01 A140   91 00072          CMPB    1(X)[PUT_HERE_BUF], #48              ; 1733
                                             30   1F 00077          BLSSU   11$                                  ;
                              39        01 A140   91 00079          CMPB    1(X)[PUT_HERE_BUF], #57              ; 1734
                                             29   1A 0007E          BGTRU   11$                                  ;
                                             22   11 00080          BRB     10$                                  ; 1737
                              2C             52   91 00082 8$:      CMPB    R2, #44                              ; 1741
                                             05   13 00085          BEQL    9$                                   ;
                              2E             52   91 00087          CMPB    R2, #46                              ;
                                             1D   12 0008A          BNEQ    11$                                  ;
                  52      14  AC             01   C3 0008C 9$:      SUBL3   #1, CHARS_READ, R2                   ; 1743
                              52             51   D1 00091          CMPL    X, R2                                ;
                                             13   13 00094          BEQL    11$                                  ;
                              2D        01 A140   91 00096          CMPB    1(X)[PUT_HERE_BUF], #45              ; 1750
                                             07   13 0009B          BEQL    10$                                  ;
                              20        01 A140   91 0009D          CMPB    1(X)[PUT_HERE_BUF], #32              ; 1751
                                             05   12 000A2          BNEQ    11$                                  ;
                              53             01   D0 000A4 10$:     MOVL    #1, ERROR                            ; 1754
                                             05   11 000A7          BRB     12$                                  ; 1753
                  8B          51        14   AC   F2 000A9 11$:     AOBLSS  CHARS_READ, X, 5$                    ; 1710
                              04             53   E9 000AE 12$:     BLBC    ERROR, 13$                           ; 1766
                                             57   D4 000B1          CLRL    CONV_OK                              ; 1768
                                             38   11 000B3          BRB     16$                                  ;
                  12          03        02   A6   8F 000B5 13$:     CASEB   2(R6), #3, #18                       ; 1774
     0026         0050        0050           0050    000BA 14$:     .WORD   20$-14$,-                            ;
     0065         0050        0050           0050    000C2                  20$-14$,-                            ;
     008B         0026        0026           0065    000CA                  20$-14$,-                            ;
     0035         0035        0035           0035    000D2                  15$-14$,-                            ;
                  0050        0026           0035    000DA                  20$-14$,-                            ;
                                                                            20$-14$,-                            ;
                                                                            20$-14$,-                            ;
                                                                            21$-14$,-                            ;
                                                                            21$-14$,-                            ;
                                                                            15$-14$,-                            ;
                                                                            15$-14$,-                            ;
                                                                            25$-14$,-                            ;
                                                                            17$-14$,-                            ;
                                                                            17$-14$,-                            ;
                                                                            17$-14$,-                            ;
                                                                            17$-14$,-                            ;
                                                                            17$-14$,-                            ;
                                                                            15$-14$,-                            ;
                                                                            20$-14$                              ;
                                 00000000G   8F   DD 000E0 15$:     PUSHL   #COB$_INVARG                         ; 1848
                      00000000G  00          01   FB 000E6          CALLS   #1, LIB$STOP                         ;
                                             54   11 000ED 16$:     BRB     24$                                  ;
                                    08       AC   DD 000EF 17$:     PUSHL   FLAGS                                ; 1786
                                    14       AC   DD 000F2          PUSHL   CHARS_READ                           ;
                                             56   DD 000F5          PUSHL   R6                                   ;
                              05    18       AC   E9 000F7          BLBC    YES_DEFAULT, 18$                     ; 1783
                                    0C       AC   DD 000FB          PUSHL   DEFAULT                              ; 1784
                                             03   11 000FE          BRB     19$                                  ;
                                    10       AC   DD 00100 18$:     PUSHL   PUT_HERE                             ; 1785
                          0000V   CF         04   FB 00103 19$:     CALLS   #4, COB$$NUMERIC_CONV                ; 1783
                                             36   11 00108          BRB     23$                                  ;
                              7E    18       AC   7D 0010A 20$:     MOVQ    YES_DEFAULT, -(SP)                   ; 1797
                              7E    10       AC   7D 0010E          MOVQ    PUT_HERE, -(SP)                      ; 1796
                              7E    08       AC   7D 00112          MOVQ    FLAGS, -(SP)                         ;
```

```
                                         56  DD 00116         PUSHL   R6                                           ; 1795
                         0000V  CF       07  FB 00118         CALLS   #7, COB$$COMP_CONV                           ;
                                         21  11 0011D         BRB     23$                                          ;
                                0F   18  AC  E9 0011F  21$:   BLBC    YES_DEFAULT, 22$                             ; 1810
                                51   0C  AC  D0 00123         MOVL    DEFAULT, R1                                  ; 1812
                                50   10  AC  D0 00127         MOVL    PUT_HERE, R0                                 ; 1813
             04    B0      04   B1   14  AC  28 0012B         MOVC3   CHARS_READ, a4(R1), a4(R0)                   ;
                                7E   10  AC  7D 00132  22$:   MOVQ    PUT_HERE, -(SP)                              ; 1816
                                     08  AC  DD 00136         PUSHL   FLAGS                                        ; 1815
                                         56  DD 00139         PUSHL   R6                                           ;
                         0000V  CF       04  FB 0013B         CALLS   #4, COB$$FLOAT_CONV                          ;
                                57       50  D0 00140  23$:   MOVL    R0, CONV_OK                                  ;
                                         31  11 00143  24$:   BRB     30$                                          ; 1774
     14   AC               66   10       00  ED 00145  25$:   CMPZV   #0, #16, (R6), CHARS_READ                    ; 1833
                                         06  15 0014B         BLEQ    26$                                          ;
                                6E   14  AC  D0 0014D         MOVL    CHARS_READ, COPY_NUM                         ; 1835
                                         03  11 00151         BRB     27$                                          ;
                                6E       66  3C 00153  26$:   MOVZWL  (R6), COPY_NUM                               ; 1837
                                06   18  AC  E9 00156  27$:   BLBC    YES_DEFAULT, 28$                             ; 1840
                                50   0C  AC  D0 0015A         MOVL    DEFAULT, R0                                  ; 1841
                                         04  11 0015E         BRB     29$                                          ;
                                50   10  AC  D0 00160  28$:   MOVL    PUT_HERE, R0                                 ; 1842
                                     04  A0  DD 00164  29$:   PUSHL   4(R0)                                        ;
                                     04  AE  9F 00167         PUSHAB  COPY_NUM                                     ; 1839
                                         56  DD 0016A         PUSHL   R6                                           ;
                      00000000G  00      03  FB 0016C         CALLS   #3, STR$COPY_R                               ;
                                57       01  D0 00173         MOVL    #1, CONV_OK                                  ; 1843
                                50       57  D0 00176  30$:   MOVL    CONV_OK, R0                                  ; 1852
                                             04 00179         RET                                                  ; 1853

; Routine Size:  378 bytes,    Routine Base:  _COB$CODE + 0000


;  340          1854  1
```

```
%SBTTL 'COB$$NUMERIC_CONV - Convert to numeric string'
ROUTINE COB$$NUMERIC_CONV (                       ! Scan a number and divide it up
        ARG_DESC        :  REF BLOCK [8, BYTE],   ! The number to scan
        STRING_DEST     :  REF BLOCK [12,BYTE],   ! Final resting place of input
        CHARS_READ,                               ! Number of characters read
        FLAGS ) =                                 ! Needed for DECIMAL POINT IS COMMA

!++
! FUNCTIONAL DESCRIPTION:
!
!       Convert a TEXT input string to the appropriate VAX COBOL Text
!       Numeric data type.  Pull off decimal point, pull off sign then
!       place it in the correct position for internal representation.
!       Do nothing about errors in this routine, return control to COB$ACC_SCR
!       via COB$$ACC_CONVERT.
!
! FORMAL PARAMETERS:
!
!       ARG_DESC.rt.dx        The number to parse
!
!       STRING_DEST.mt.ds     Address of descriptor to receive the read input.
!
!       CHARS_READ.rlu.v      Number of characters accepted as input.
!
!       FLAGS.rlu.v           Screen enhancement flag;
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
!
!       1 = Conversion Success
!       0 = Conversion Failure
!
! SIDE EFFECTS:
!
!       Signals COB$_INVARG if the syntax of the number is wrong,
!--

    BEGIN

    LOCAL
        SIGN_VAL : BYTE,                          ! Holds + or - sign
        BUF_DESC : BLOCK [8, BYTE] VOLATILE,      ! Temporary buffer
        SIGN_SEEN   :  INITIAL (0),               ! 1 = we have seen a + or -
        DIGIT_SEEN  :  INITIAL (0),               ! 1 = we have seen at least one digit
        DOT_SEEN    :  INITIAL (0),               ! 1 = we have seen a decimal point
        BLANKS_SEEN :  INITIAL (0),               ! 1 = we have seen trailing blanks
        ZERO_SEEN   :  INITIAL (0),               ! 1 = zero seen
        PUTTER      :  INITIAL (0),               ! Counts position in the output buffer
        BUF : REF VECTOR [1100, BYTE],            ! Addresses result
        ARG : REF VECTOR [1100, BYTE],            ! Addresses source
```

```
2               ARG_LEN,                                ! Length of the source
2               NUM_DIGITS  :  INITIAL (0),             ! Number of digits in ARG_DESC
2               LEFT_DEC    :  INITIAL (0),             ! Number of digits to left of dec pt
2               RIGHT_DEC   :  INITIAL (0),             ! Number of digits to right of dec pt
2               LEADING_ZEROES: INITIAL (0),            ! Counter of leading zeroes
2               OK_LEFT ;                               ! Correct number of digits allowed
2                                                       ! to left of decimal point
2
2           BIND
2               ZERO = UPLIT ('0');
2
2       !+
2       ! Enable a handler to free the local string in case of an error.
2       !-
2           ENABLE
2               COB$$FREE_STRINGS (BUF_DESC);
2
2       !+
2       ! Allocate enough space to hold the digits.  It is convenient to
2       ! allocate before scanning, so we may allocate a little too much,
2       ! but the space will be freed before we return.
2       !-
2           BUF_DESC [DSC$W_LENGTH] = 0;
2           BUF_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2           BUF_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2           BUF_DESC [DSC$A_POINTER] = 0;
2           ARG_LEN = .ARG_DESC [DSC$W_LENGTH];
2           STR$GET1_DX (%REF (.ARG_LEN + 20), BUF_DESC);
2       !+
2       !  Set pointers.
2       !-
2           BUF = .BUF_DESC [DSC$A_POINTER];
2           ARG = .ARG_DESC [DSC$A_POINTER];
2           SIGN_VAL = %C'+';
2
2       !+
2       !  Scan the input number, put result in BUF.
2       !-
2
3           IF NOT ( COB$$SCAN_INPUT ( .ARG_DESC, .CHARS_READ, .FLAGS, BUF_DESC,
3                     LEFT_DEC, NUM_DIGITS, SIGN_VAL, PUTTER, LEADING_ZEROES, SIGN_SEEN,
3                               DIGIT_SEEN, DOT_SEEN, ZERO_SEEN, BLANKS_SEEN ) )
2           THEN
2               RETURN 0 ;
2
2       !+
2       !  Now ensure that a number of the form 1.0000 has the trailing
2       !  zeroes stripped off.
2       !-
2           IF .DOT_SEEN
2           THEN
3               BEGIN
3                 LOCAL
3                   X;
3
3               X = .PUTTER - 1;
3               IF .X NEQ 0
```

```
3               THEN
4                   BEGIN
4                   WHILE .BUF[.X] EQL %C'0' DO
5                       BEGIN                                   ! Throw out trailing
5                       NUM_DIGITS = .NUM_DIGITS - 1;           ! zeroes after dec pt.
5                       IF .X EQL 0
5                       THEN
6                           BEGIN                               ! Get here if all digits
6                           DIGIT_SEEN = 0;                     ! after decimal point
6                           DOT_SEEN = 0;                       ! are zeroes, and zeroes
6                           EXITLOOP;                           ! are the only digits in
5                           END;                                ! the string (eg. .0000)
5                       X = .X - 1;
4                       END;
3                   END;
2               END;
2
2       !+
2       ! NOTE: Call to COB$$ZERO_FILL was originally done here, however because
2       !       VAX RPG wants an illegal string returned in its original state,
2       !       it was necessary to be more selective about where and when to
2       !       call COB$$ZERO_FILL.
2       !-
2
2       !+
2       ! If there are no digits, or only leading zeros, take the number to
2       ! be zero.  Don't be too gullible, however.
2       !-
3           IF ( NOT .DIGIT_SEEN)
2           THEN
3               BEGIN
3
4               IF (.SIGN_SEEN OR .DOT_SEEN OR .BLANKS_SEEN) AND (.ZERO_SEEN EQL 0)
3               THEN RETURN 0 ;
3
3               !+
3               ! Fill STRING_DEST with zeroes
3               !-
3
3               COB$$ZERO_FILL ( .STRING_DEST ) ;
3               RETURN 1 ;
3
3               END
3
3       !+
3       ! Validate size of entered data, left and right of decimal point.
3       ! If everything is OK, copy the input string to STRING_DEST with
3       ! the sign set up correctly.
3       ! Return if DEFAULT is being converted (there will never be a decimal point
3       ! in the DEFAULT parameter)
3       !-
2           ELSE
3               BEGIN
3
3               LOCAL
3                   DEST_PTR ,                                  ! Pointer where result will go in destination
3                   DIGITS_IN_STRING ,                          ! Number of digits in destination string
```

```
                DEST_LENGTH ,                                   ! Destination length
                LENGTH_DIFF ;                                   ! Difference between number of digits
                                                                ! to the left of the decimal point
                                                                ! in the number typed in and the dest

        LITERAL
                POSOP = 16,                                     ! Number to add to make overpunched +
                NEGOP = 25,                                     ! Number to add to make overpunched -
                POSZEROP = 123,                                 ! +0 overpunched
                NEGZEROP = 125;                                 ! -0 overpunched


        IF NOT (.DOT_SEEN)                                      ! No dec pt. therefore
        THEN LEFT_DEC = .NUM_DIGITS ;                           ! all digits are left_dec
        RIGHT_DEC = .NUM_DIGITS - .LEFT_DEC ;

        DEST_LENGTH = .STRING_DEST [DSC$W_LENGTH];

        SELECTONE .STRING_DEST [DSC$B_CLASS] OF
            SET

            [ DSC$K_CLASS_S ] :

                BEGIN

                !+
                ! If a decimal point was typed in, all the digits after it
                ! MUST be zeroes.
                !-

                IF .RIGHT_DEC GTR 0
                THEN
                    INCR I FROM (.PUTTER - .RIGHT_DEC) TO .PUTTER DO
                        IF .BUF[.I] NEQ %C'0'
                        THEN RETURN 0;

                !+
                ! If the number of digits typed in is less than the number of
                ! digits in the destination string, then a pointer must be
                ! set up here so that the typed in digits get moved to the
                ! correct place in the destination.
                !-

                DIGITS_IN_STRING =
                        (IF (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_NL) OR
                            (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_NR)
                         THEN .DEST_LENGTH - 1
                         ELSE .DEST_LENGTH );
                IF .LEFT_DEC GTR .DIGITS_IN_STRING
                THEN RETURN 0;                                  ! Data entered too bit

                STR$DUPL_CHAR (.STRING_DEST, DEST_LENGTH, ZERO);! Zero the destination
                LENGTH_DIFF = .DIGITS_IN_STRING - .LEFT_DEC;
                IF .LENGTH_DIFF GTR 0
                THEN
                    DEST_PTR = .STRING_DEST [DSC$A_POINTER] + .LENGTH_DIFF
                ELSE
```

```
                DEST_PTR = .STRING_DEST [DSC$A_POINTER];

            END ;

        [ DSC$K_CLASS_SD ] :

            BEGIN
            !+
            ! All code for "P" data types are in lowercase.
            !-

            LOCAL
                ok_right,
                LENGTH_DIFF2;                           ! Difference between number of digits
                                                        ! to the right of the decimal point
                                                        ! in the typed in number and the dest

            dest_length = .string_dest[dsc$b_digits];

            !+
            ! This is checking for the P Picture of 99PP.
            ! If the scale is positive and the number of digits in the
            ! number equal the scale factor, then simply copy the digits
            ! in BUF to the destination descriptor.
            !-

            if .string_dest[dsc$b_scale] gtr 0
            then
                begin

                local
                    tot_digits,
                    diff;

                if ((.right_dec gtr 0) or (.num_digits gtr (.string_dest[dsc$b_digits] + .string_dest[ds
                then                                    ! number too large
                    return 0;                           ! re-prompt - error

                if .num_digits leq .string_dest[dsc$b_scale]
                then

                !+
                ! Zero out the destination field using the digits as the proper
                ! number of zero fill characters, rather than using the length
                ! as found in the descriptor, since class SD is a special case.
                !-
                    begin
                    str$dupl_char (.string_dest, dest_length, zero);
                    return 1;                           ! answer is zero
                    end ;

                if .leading_zeroes neq 0
                then
                    begin

                    diff = (.string_dest[dsc$b_digits]+.string_dest[dsc$b_scale]) - .num_digits;
                    dest_ptr = .string_dest[dsc$a_pointer]+.diff;
```

```
6                               end
5                           else
6                               begin
6
6                               diff = .num_digits - .string_dest[dsc$b_scale];
6                               if .diff eql .string_dest[dsc$b_digits]
6                               then
6                                   dest_ptr = .string_dest[dsc$a_pointer]
6                               else
7                                   begin
7
7                                   tot_digits = (.string_dest[dsc$b_digits] + .string_dest[dsc$b_scale]) - .num_dig
7                                   dest_ptr = .string_dest[dsc$a_pointer]+.tot_digits;
7
6                                   end;
6
6                               num_digits = .diff;
6
5                               end;
5
5                           end
4                       else
5                           begin
5
5                           OK_LEFT = .STRING_DEST [DSC$B_DIGITS] + .STRING_DEST [DSC$B_SCALE] ;
5                           if .ok_left lss 0
5                           then
5                               !+
5                               ! Here we have a P Picture field of type PP99.
5                               ! We know this when OK_LEFT is less than zero.
5                               ! It requires some special casing.
5                               !-
6                               begin
6
6                               local
6                                   diff,
6                                   diff2,
6                                   ptr,
6                                   buf_ptr;
6
6                               if .left_dec gtr 0                  ! error no '.' entered
6                               then                                ! ring bell and reprompt
6                                   return 0;
6
6                               ok_left = 0;
6                               ok_right = abs(.string_dest[dsc$b_scale]);
6                               if .right_dec gtr .ok_right
6                               then
6                                   return 0;
6
6                               !+
6                               ! This handles case where the number of digits
6                               ! entered is less than the absolute value of the
6                               ! scale factor, meaning that the number returned
6                               ! would have to be zero.  The first part of the
6                               ! if statement takes care of the case where the
```

```
; 6                             ! number of digits entered equals the number of
; 6                             ! digits expected taking into account if the absolute
; 6                             ! value of the scale factor is equal to the number
; 6                             ! of digits entered to the right of the decimal point
; 6                             ! thereby giving us a result of zero again.
; 6                             !-
; 6
; 7                             diff = (abs(.string_dest[dsc$b_scale]) -
; 6                                     .string_dest[dsc$b_digits]);    ! Number of Placeholders in picture
; 7                             if ((.right_dec eql .string_dest[dsc$b_digits]) and
; 6                             (.right_dec eql .diff)) OR
; 7                             (.right_dec leq .diff)
; 6                             then
; 7                                 begin
; 7                                 str$dupl_char (.string_dest, dest_length, zero);
; 7                                 return 1;                       ! done - answer is zero
; 6                                 end ;
; 6
; 6                             buf_ptr = .buf_desc [dsc$a_pointer] + .diff;
; 6                             diff = .num_digits - .diff;      ! Number of digits minus placeholders
; 6                             ch$move (.diff, .buf_ptr, .buf_desc[dsc$a_pointer]);
; 6                             diff2 = .string_dest[dsc$b_digits] - .diff;      ! Number of digits to zero fill
; 6                             ptr = .buf_desc[dsc$a_pointer] + .diff;
; 6                             incr i from 0 to .diff2 - 1 do
; 6                                 ch$move (1, zero, .ptr + .i);
; 6
; 6                             dest_ptr = .string_dest[dsc$a_pointer];
; 6                             num_digits = .string_dest[dsc$b_digits] ;        ! should only reflect number of digi
; 6
; 6                             end
; 5                         else
; 6                             begin
; 6
; 6                             LENGTH_DIFF = .OK_LEFT - .LEFT_DEC;
; 6                             LENGTH_DIFF2 = (.STRING_DEST [DSC$B_DIGITS] - .OK_LEFT) - .RIGHT_DEC;
; 7                             IF ( .LENGTH_DIFF LSS 0) OR ( .LENGTH_DIFF2 LSS 0)
; 6                             THEN
; 6                                 RETURN 0 ;                      ! Data entered too big
; 6
; 6                             !+
; 6                             ! If the number of digits to the left of the decimal
; 6                             ! point of the number typed in is less than what
; 6                             ! should be in the destination string, then a pointer
; 6                             ! must be set up here so that the typed in digits get
; 6                             ! moved to the correct place in the destination.
; 6                             !-
; 6                             IF .LENGTH_DIFF GTR 0
; 6                             THEN
; 6                                 DEST_PTR = .STRING_DEST [DSC$A_POINTER] + .LENGTH_DIFF
; 6                             ELSE
; 6                                 DEST_PTR = .STRING_DEST [DSC$A_POINTER];
; 6
; 6                             !+
; 6                             ! If the number of digits to the right of the decimal
; 6                             ! point of the number typed in is less than what
; 6                             ! should be in the destination string, then the typed
; 6                             ! in number must be padded with trailing zeroes so that
```

```
6                                   ! sign placement can be done correctly.
6                                   !-
6                                   IF .LENGTH_DIFF2 GTR 0
6                                   THEN
7                                       BEGIN
7
7                                       LOCAL
7                                           PTR;                        ! Pointer into input buffer
7                                                                       !  past the digits that were typed in
7
7                                       PTR = .BUF_DESC [DSC$A_POINTER] + .NUM_DIGITS - 1;
7                                       NUM_DIGITS = .NUM_DIGITS + .LENGTH_DIFF2;
7
7                                       DO
8                                           BEGIN
8
8                                           LOCAL
8                                               PTR2;                   ! Loop pointer
8
8                                           PTR2 = .PTR + .LENGTH_DIFF2;
8                                           CH$MOVE (1, ZERO, .PTR2);
8                                           LENGTH_DIFF2 = .LENGTH_DIFF2 - 1;
8
8                                           END
7                                       UNTIL .LENGTH_DIFF2 EQL 0;
7
6                                       END;
6
5                                   end;
4                               end;
3                       END ;
3
3                   [ OTHERWISE ] :
3
3                       LIB$STOP ( COB$_INVARG ) ;
3
3                   TES ;
3
3               !+
3               ! Fill STRING_DEST with zeroes.
3               ! If everything is OK, copy the input string to STRING_DEST with
3               ! the sign set up correctly.
3               !-
3
3               COB$$ZERO_FILL ( .STRING_DEST ) ;
3
3               CASE .STRING_DEST [DSC$B_DTYPE] FROM DSC$K_DTYPE_NU TO DSC$K_DTYPE_NRO
3                   OF
3                   SET
3
3                   [DSC$K_DTYPE_NU]:                                   ! Numeric unsigned
3                       !+
3                       ! Simply ignore a sign if it was part of the input
3                       ! string.
3                       !-
3                       CH$MOVE (.NUM_DIGITS, .BUF_DESC [DSC$A_POINTER], .DEST_PTR);
3
```

```
        [DSC$K_DTYPE_NL]:                                     ! Numeric left separate

            BEGIN

            CH$MOVE (1, SIGN_VAL, .STRING_DEST [DSC$A_POINTER]);
            CH$MOVE (.NUM_DIGITS, .BUF_DESC [DSC$A_POINTER], .DEST_PTR + 1);

            END;

        [DSC$K_DTYPE_NR]:                                     ! Numeric right separate

            BEGIN

            BUF [.NUM_DIGITS] = .SIGN_VAL;
            CH$MOVE (.NUM_DIGITS + 1, .BUF_DESC [DSC$A_POINTER], .DEST_PTR);

            END;

        [DSC$K_DTYPE_NLO]:                                    ! Numeric left overpunched

            BEGIN
            LOCAL
                FIRST_TWO    : VECTOR [2, BYTE],              ! To be compared with
                                                              ! first two bytes of STRING_DEST
                FIRST_DIGIT  :  BYTE ;                        ! First digit and overpunch sign

            FIRST_TWO [0] = %X'7B' ;                          ! Positive overpunched 0
            FIRST_TWO [1] = %X'30' ;                          ! Regular 0

            !+
            !  First byte of initial state STRING_DEST is always 7B,
            !  therefore have to look at first two bytes.
            !-

            CH$MOVE (.NUM_DIGITS, .BUF_DESC [DSC$A_POINTER], .DEST_PTR) ;
            IF CH$EQL (2, .STRING_DEST [DSC$A_POINTER], 2, FIRST_TWO )

            THEN
                FIRST_DIGIT =
                    ( IF .SIGN_VAL EQL %C'+'
                      THEN POSZEROP
                      ELSE NEGZEROP )
            ELSE
                !+
                !  Special treatment needed when .NUM_DIGITS is less
                !  than total # STRING_DEST can hold.
                !  For example :  S99V99 - input 3.00 ->
                !  First two bytes of STRING_DEST now hold '7B' and '33',
                !  but want POSZEROP not POSOP as POSOP will result in wrong
                !  number 33.00.
                !-
                IF .NUM_DIGITS LSS .STRING_DEST [DSC$W_LENGTH]
                THEN
                    FIRST_DIGIT =
                        ( IF .SIGN_VAL EQL %C'+'
                              THEN POSZEROP
                              ELSE NEGZEROP )
```

```
                        ELSE
                            FIRST_DIGIT =
                                !+
                                !  Add whatever necessary to a regular hex number
                                !  to put it in overpunch format.  Have to special
                                !  case zero.
                                !-
                                ( IF .SIGN_VAL EQL %C'+'
                                  THEN
                                      IF .BUF [0] EQL %C'0'
                                      THEN POSZEROP
                                      ELSE .BUF [0] + POSOP
                                  ELSE
                                      IF .BUF [0] EQL %C'0'
                                      THEN NEGZEROP
                                      ELSE .BUF [0] + NEGOP );

                    CH$MOVE (1, FIRST_DIGIT, .STRING_DEST [DSC$A_POINTER] ) ;

                    END;

                [DSC$K_DTYPE_NRO]:                          ! Numeric right overpunched

                    BEGIN
                    !+
                    !  Add whatever necessary to a regular hex number
                    !  to put it in overpunch format.  Have to special
                    !  case zero.
                    !-

                    IF .BUF [.NUM_DIGITS - 1] EQL %C'0'
                    THEN
                        BUF [.NUM_DIGITS - 1] =
                            ( IF .SIGN_VAL EQL %C'+'
                              THEN POSZEROP
                              ELSE NEGZEROP )
                    ELSE
                        BUF [.NUM_DIGITS - 1] =
                            ( IF .SIGN_VAL EQL %C'+'
                              THEN .BUF [.NUM_DIGITS - 1] + POSOP
                              ELSE .BUF [.NUM_DIGITS - 1] + NEGOP );

                    CH$MOVE (.NUM_DIGITS, .BUF_DESC [DSC$A_POINTER], .DEST_PTR);

                    END;

                TES;

            END ;

    !+
    ! Free local string
    !-
        STR$FREE1_DX (BUF_DESC);
        RETURN 1 ;
        END;                                        ! end of COB$$NUMERIC_CONV
```

```
                                        0017A           .BLKB   2
                        00  00  00  30  0017C P.AAA:    .ASCII  \0\<0><0><0>                                    ;

                                              ZERO=             P.AAA

                                  0FFC 00000 COB$$NUMERIC_CONV:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11            ; 1856
                   5B     F7   AF   9E 00002            MOVAB   ZERO, R11                                       ;
                   5E          3C   C2 00006            SUBL2   #60, SP                                         ;
                          10   AE   7C 00009            CLRQ    DIGIT_SEEN                                      ; 1899
                          04   AE   D4 0000C            CLRL    BLANKS_SEEN                                     ;
                          08   AE   7C 0000F            CLRQ    ZERO_SEEN                                       ;
                          24   AE   7C 00012            CLRQ    NUM_DIGITS                                      ;
                               52   D4 00015            CLRL    RIGHT_DEC                                       ;
                          18   AE   7C 00017            CLRQ    LEADING_ZEROES                                  ;
                          34   AE   7C 0001A            CLRQ    BUF_DESC                                        ;
                   6D   0327   CF   DE 0001D            MOVAL   63$, (FP)                                       ;
                          34   AE   B4 00022            CLRW    BUF_DESC                                        ; 1934
              36   AE          0F   90 00025            MOVB    #15, BUF_DESC+2                                 ; 1935
              37   AE          02   90 00029            MOVB    #2, BUF_DESC+3                                  ; 1936
                          38   AE   D4 0002D            CLRL    BUF_DESC+4                                      ; 1937
                   53     04   AC   D0 00030            MOVL    ARG_DESC, R3                                    ; 1938
                   50          63   3C 00034            MOVZWL  (R3), ARG_LEN                                   ;
                          34   AE   9F 00037            PUSHAB  BUF_DESC                                        ; 1939
              04   AE     14   A0   9E 0003A            MOVAB   20(R0), 4(SP)                                   ;
                          04   AE   9F 0003F            PUSHAB  4(SP)                                           ;
       00000000G   00          02   FB 00042            CALLS   #2, STR$GET1_DX                                 ;
                   59     38   AE   D0 00049            MOVL    BUF_DESC+4, BUF                                 ; 1943
                   50     04   A3   D0 0004D            MOVL    4(R3), ARG                                      ; 1944
              20   AE          2B   90 00051            MOVB    #43, SIGN_VAL                                   ; 1945
                          04   AE   9F 00055            PUSHAB  BLANKS_SEEN                                     ; 1951
                          0C   AE   9F 00058            PUSHAB  ZERO_SEEN                                       ;
                          14   AE   9F 0005B            PUSHAB  DOT_SEEN                                        ;
                          1C   AE   9F 0005E            PUSHAB  DIGIT_SEEN                                      ;
                          24   AE   9F 00061            PUSHAB  SIGN_SEEN                                       ;
                          2C   AE   9F 00064            PUSHAB  LEADING_ZEROES                                  ;
                          34   AE   9F 00067            PUSHAB  PUTTER                                          ;
                          3C   AE   9F 0006A            PUSHAB  SIGN_VAL                                        ;
                          44   AE   9F 0006D            PUSHAB  NUM_DIGITS                                      ;
                          4C   AE   9F 00070            PUSHAB  LEFT_DEC                                        ;
                          5C   AE   9F 00073            PUSHAB  BUF_DESC                                        ;
                   7E     0C   AC   7D 00076            MOVQ    CHARS_READ, -(SP)                               ;
                               53   DD 0007A            PUSHL   R3                                              ;
           0000V   CF          0E   FB 0007C            CALLS   #14, COB$$SCAN_INPUT                            ;
                   03          50   E8 00081            BLBS    R0, 2$                                          ;
                        02BE   31 00084 1$:             BRW     62$                                             ;
                   1D     0C   AE   E9 00087 2$:        BLBC    DOT_SEEN, 5$                                    ; 1961
    50        1C   AE          01   C3 0008B            SUBL3   #1, PUTTER, X                                   ; 1967
                               16   13 00090            BEQL    5$                                              ; 1968
                   30        6049   91 00092 3$:        CMPB    (X)[BUF], #48                                   ; 1971
                               10   12 00096            BNEQ    5$                                              ;
                          24   AE   D7 00098            DECL    NUM_DIGITS                                      ; 1973
                               50   D5 0009B            TSTL    X                                               ; 1974
                               05   12 0009D            BNEQ    4$                                              ;
```

```
                          0C     AE   7C 0009F        CLRQ    DOT_SEEN                                 ; 1978
                                 04   11 000A2        BRB     5$                                       ; 1976
                                 50   D7 000A4 4$:    DECL    X                                        ; 1981
                                 EA   11 000A6        BRB     3$                                       ; 1971
                  56      08     AC   D0 000A8 5$:    MOVL    STRING_DEST, R6                          ; 2008
                  1B      10     AE   E8 000AC        BLBS    DIGIT_SEEN, 8$                           ; 1997
                  08      14     AE   E8 000B0        BLBS    SIGN_SEEN, 6$                            ; 2001
                  04      0C     AE   E8 000B4        BLBS    DOT_SEEN, 6$                             ;
                  05      04     AE   E9 000B8        BLBC    BLANKS_SEEN, 7$                          ;
                          08     AE   D5 000BC 6$:    TSTL    ZERO_SEEN                                ;
                                 C3   13 000BF        BEQL    1$                                       ;
                                 56   DD 000C1 7$:    PUSHL   R6                                       ; 2008
       0000V      CF             01   FB 000C3        CALLS   #1, COB$$ZERO_FILL                       ;
                               0276   31 000C8        BRW     61$                                      ; 2009
                  05      0C     AE   E8 000CB 8$:    BLBS    DOT_SEEN, 9$                             ; 2038
           28     AE      24     AE   D0 000CF        MOVL    NUM_DIGITS, LEFT_DEC                     ; 2039
                  55      24     AE   D0 000D4 9$:    MOVL    NUM_DIGITS, R5                           ; 2040
                  57      28     AE   D0 000D8        MOVL    LEFT_DEC, R7                             ;
    52            55             57   C3 000DC        SUBL3   R7, R5, RIGHT_DEC                        ;
           2C     AE             66   3C 000E0        MOVZWL  (R6), DEST_LENGTH                        ; 2042
                  50      03     A6   9A 000E4        MOVZBL  3(R6), R0                                ; 2044
                  01             50   91 000E8        CMPB    R0, #1                                   ; 2047
                                 54   12 000EB        BNEQ    17$                                      ;
                                 52   D5 000ED        TSTL    RIGHT_DEC                                ; 2056
                                 14   15 000EF        BLEQ    12$                                      ;
    50     1C     AE             52   C3 000F1        SUBL3   RIGHT_DEC, PUTTER, R0                    ; 2058
                                 50   D7 000F6        DECL    I                                        ; 2059
                                 06   11 000F8        BRB     11$                                      ;
                  30           6049   91 000FA 10$:   CMPB    (I)[BUF], #48                            ;
                                 84   12 000FE        BNEQ    1$                                       ;
    F5            50      1C     AE   F3 00100 11$:   AOBLEQ  PUTTER, I, 10$                           ;
                  10      02     A6   91 00105 12$:   CMPB    2(R6), #16                               ; 2070
                                 06   13 00109        BEQL    13$                                      ;
                  12      02     A6   91 0010B        CMPB    2(R6), #18                               ; 2071
                                 07   12 0010F        BNEQ    14$                                      ;
    53     2C     AE             01   C3 00111 13$:   SUBL3   #1, DEST_LENGTH, DIGITS_IN_STRING        ; 2072
                                 04   11 00116        BRB     15$                                      ;
                  53      2C     AE   D0 00118 14$:   MOVL    DEST_LENGTH, DIGITS_IN_STRING            ; 2073
                  53             57   D1 0011C 15$:   CMPL    R7, DIGITS_IN_STRING                     ; 2074
                                 45   14 0011F        BGTR    19$                                      ;
                                 5B   DD 00121        PUSHL   R11                                      ; 2077
                          30     AE   9F 00123        PUSHAB  DEST_LENGTH                              ;
                                 56   DD 00126        PUSHL   R6                                       ;
      00000000G   00             03   FB 00128        CALLS   #3, STR$DUPL_CHAR                        ;
                  53             57   C2 0012F        SUBL2   R7, LENGTH_DIFF                          ; 2078
                                 07   15 00132        BLEQ    16$                                      ; 2079
    58            53      04     A6   C1 00134        ADDL3   4(R6), LENGTH_DIFF, DEST_PTR             ; 2081
                                 77   11 00139        BRB     23$                                      ;
                  58      04     A6   D0 0013B 16$:   MOVL    4(R6), DEST_PTR                          ; 2083
                                 71   11 0013F        BRB     23$                                      ; 2044
                  09             50   91 00141 17$:   CMPB    R0, #9                                   ; 2087
                                 03   13 00144        BEQL    18$                                      ;
                               011B   31 00146        BRW     39$                                      ;
                  54      09     A6   9A 00149 18$:   MOVZBL  9(R6), R4                                ; 2100
           2C     AE             54   D0 0014D        MOVL    R4, DEST_LENGTH                          ;
                  5A      04     A6   9E 00151        MOVAB   4(R6), R10                               ; 2139
                  58      08     A6   98 00155        CVTBL   8(R6), R8                                ; 2109
```

```
                                          59  15 00159        BLEQ    24$                                        ;
                                          52  D5 0015B        TSTL    RIGHT_DEC                                  ; 2117
                                          6F  14 0015D        BGTR    26$                                        ;
                  50         54           58  C1 0015F        ADDL3   R8, R4, R0                                 ;
                             50           55  D1 00163        CMPL    R5, R0                                     ;
                                          66  14 00166 19$:   BGTR    26$                                        ;
                             58           55  D1 00168        CMPL    R5, R8                                     ; 2121
                                          79  15 0016B        BLEQ    30$                                        ;
                                    18    AE  D5 0016D        TSTL    LEADING_ZEROES                             ; 2134
                                          15  13 00170        BEQL    20$                                        ;
                             51     09    A6  9A 00172        MOVZBL  9(R6), R1                                  ; 2138
                             50     08    A6  98 00176        CVTBL   8(R6), R0                                  ;
                             51           50  C0 0017A        ADDL2   R0, R1                                     ;
                  50         51           55  C3 0017D        SUBL3   R5, R1, DIFF                               ;
                  58         6A           50  C1 00181        ADDL3   DIFF, (R10), DEST_PTR                      ; 2139
                                          2B  11 00185        BRB     23$                                        ; 2134
                  50         08           A6  98 00187 20$:   CVTBL   8(R6), DIFF                                ; 2145
                  50         55           50  C3 0018B        SUBL3   DIFF, R5, DIFF                             ;
      50     09   A6         08           00  ED 0018F        CMPZV   #0, #8, 9(R6), DIFF                        ; 2146
                                          05  12 00195        BNEQ    21$                                        ;
                             58           6A  D0 00197        MOVL    (R10), DEST_PTR                            ; 2148
                                          12  11 0019A        BRB     22$                                        ;
                             51     09    A6  9A 0019C 21$:   MOVZBL  9(R6), R1                                  ; 2152
                             53     08    A6  98 001A0        CVTBL   8(R6), R3                                  ;
                             51           53  C0 001A4        ADDL2   R3, R1                                     ;
                             51           55  C2 001A7        SUBL2   R5, TOT_DIGITS                             ;
                  58         6A           51  C1 001AA        ADDL3   TOT_DIGITS, (R10), DEST_PTR                ; 2153
                        24   AE           50  D0 001AE 22$:   MOVL    DIFF, NUM_DIGITS                           ; 2157
                                          72  11 001B2 23$:   BRB     34$                                        ; 2109
                  50         54           58  C1 001B4 24$:   ADDL3   R8, R4, OK_LEFT                            ; 2165
                                          6E  18 001B8        BGEQ    35$                                        ; 2166
                                          57  D5 001BA        TSTL    R7                                         ; 2181
                                          10  14 001BC        BGTR    26$                                        ;
                                          50  D4 001BE        CLRL    OK_LEFT                                    ; 2185
                             50           58  D0 001C0        MOVL    R8, R0                                     ; 2186
                                          03  18 001C3        BGEQ    25$                                        ;
                             50           50  CE 001C5        MNEGL   R0, R0                                     ;
                             51           50  D0 001C8 25$:   MOVL    R0, OK_RIGHT                               ;
                             51           52  D1 001CB        CMPL    RIGHT_DEC, OK_RIGHT                        ; 2187
                                          03  15 001CE 26$:   BLEQ    28$                                        ;
                                        0172  31 001D0 27$:   BRW     62$                                        ;
                  57         50           54  C3 001D3 28$:   SUBL3   R4, R0, DIFF                               ; 2205
                             54           52  D1 001D7        CMPL    RIGHT_DEC, R4                              ; 2206
                                          05  12 001DA        BNEQ    29$                                        ;
                             57           52  D1 001DC        CMPL    RIGHT_DEC, DIFF                            ; 2207
                                          05  13 001DF        BEQL    30$                                        ;
                             57           52  D1 001E1 29$:   CMPL    RIGHT_DEC, DIFF                            ; 2208
                                          11  14 001E4        BGTR    31$                                        ;
                                          5B  DD 001E6 30$:   PUSHL   R11                                        ; 2211
                                    30    AE  9F 001E8        PUSHAB  DEST_LENGTH                                ;
                                          56  DD 001EB        PUSHL   R6                                         ;
                       00000000G     00   03  FB 001ED        CALLS   #3, STR$DUPL_CHAR                          ;
                                        014A  31 001F4        BRW     61$                                        ; 2212
                  50         57     38    AE  C1 001F7 31$:   ADDL3   BUF_DESC+4, DIFF, BUF_PTR                  ; 2215
                  57         55           57  C3 001FC        SUBL3   DIFF, R5, DIFF                             ; 2216
             38   BE         60           57  28 00200        MOVC3   DIFF, (BUF_PTR), @BUF_DESC+4               ; 2217
                             52     09    A6  9A 00205        MOVZBL  9(R6), DIFF2                               ; 2218
```

```
                                52        57 C2 00209        SUBL2   DIFF, DIFF2                              ;
               50               57   38   AE C1 0020C        ADDL3   BUF_DESC+4, DIFF, PTR                    ; 2219
                                51        01 CE 00211        MNEGL   #1, I                                    ; 2221
                                          04 11 00214        BRB     33$                                      ;
                              6140        6B 90 00216 32$:   MOVB    ZERO, (I)[PTR]                           ;
               F8               51        52 F2 0021A 33$:   AOBLSS  DIFF2, I, 32$                            ;
                                58        6A D0 0021E        MOVL    (R10), DEST_PTR                          ; 2223
                          24    AE   09   A6 9A 00221        MOVZBL  9(R6), NUM_DIGITS                        ; 2224
                                          49 11 00226 34$:   BRB     40$                                      ; 2166
               53               50        57 C3 00228 35$:   SUBL3   R7, OK_LEFT, LENGTH_DIFF                 ; 2230
                                54        50 C2 0022C        SUBL2   OK_LEFT, R4                              ; 2231
               51               54        52 C3 0022F        SUBL3   RIGHT_DEC, R4, LENGTH_DIFF2              ;
                                          53 D5 00233        TSTL    LENGTH_DIFF                              ; 2232
                                          99 19 00235        BLSS    27$                                      ;
                                          51 D5 00237        TSTL    LENGTH_DIFF2                             ;
                                          95 19 00239        BLSS    27$                                      ;
                                          53 D5 0023B        TSTL    LENGTH_DIFF                              ; 2243
                                          06 15 0023D        BLEQ    36$                                      ;
               58               6A        53 C1 0023F        ADDL3   LENGTH_DIFF, (R10), DEST_PTR             ; 2245
                                          03 11 00243        BRB     37$                                      ;
                                58        6A D0 00245 36$:   MOVL    (R10), DEST_PTR                          ; 2247
                                          51 D5 00248 37$:   TSTL    LENGTH_DIFF2                             ; 2256
                                          25 15 0024A        BLEQ    40$                                      ;
               50               55   38   AE C1 0024C        ADDL3   BUF_DESC+4, R5, R0                       ; 2264
                                          50 D7 00251        DECL    PTR                                      ;
                          24    AE        51 C0 00253        ADDL2   LENGTH_DIFF2, NUM_DIGITS                 ; 2265
               52               50        51 C1 00257 38$:   ADDL3   LENGTH_DIFF2, PTR, PTR2                  ; 2273
                                62        6B 90 0025B        MOVB    ZERO, (PTR2)                             ; 2274
                                          51 D7 0025E        DECL    LENGTH_DIFF2                             ; 2275
                                          F5 12 00260        BNEQ    38$                                      ; 2278
                                          0D 11 00262        BRB     40$                                      ; 2044
                               00000000G  8F DD 00264 39$:   PUSHL   #COB$_INVARG                             ; 2288
                    00000000G   00        01 FB 0026A        CALLS   #1, LIB$STOP                             ;
                                          56 DD 00271 40$:   PUSHL   R6                                       ; 2298
                        0000V   CF        01 FB 00273        CALLS   #1, COB$$ZERO_FILL                       ;
                                57   24   AE D0 00278        MOVL    NUM_DIGITS, R7                           ; 2309
               04               0F   02   A6 8F 0027C        CASEB   2(R6), #15, #4                           ; 2300
   001A      002A             000D      00B1    00281 41$:   .WORD   59$-41$,-                                ;
                                        007E    00289                42$-41$,-                                ;
                                                                     44$-41$,-                                ;
                                                                     43$-41$,-                                ;
                                                                     54$-41$                                  ;
                                        00A4 31 0028B        BRW     59$                                      ; 2309
                          04    B6   20   AE 90 0028E 42$:   MOVB    SIGN_VAL, @4(R6)                         ; 2315
            01 A8         38    BE        57 28 00293        MOVC3   R7, @BUF_DESC+4, 1(DEST_PTR)             ; 2316
                                          62 11 00299        BRB     53$                                      ; 2300
                              6749   20   AE 90 0029B 43$:   MOVB    SIGN_VAL, (R7)[BUF]                      ; 2324
                                50   01   A7 9E 002A0        MOVAB   1(R7), R0                                ; 2325
               68         38    BE        50 28 002A4        MOVC3   R0, @BUF_DESC+4, (DEST_PTR)              ;
                                          52 11 002A9        BRB     53$                                      ; 2300
                          30    AE 307B   8F B0 002AB 44$:   MOVW    #12411, FIRST_TWO                        ; 2337
               68         38    BE        57 28 002B1        MOVC3   R7, @BUF_DESC+4, (DEST_PTR)              ; 2345
                          30    AE   04   B6 B1 002B6        CMPW    @4(R6), FIRST_TWO                        ; 2346
                                          07 13 002BB        BEQL    45$                                      ;
   57          66               10        00 ED 002BD        CMPZV   #0, #16, (R6), R7                        ; 2362
                                          08 15 002C2        BLEQ    46$                                      ;
                                2B   20   AE 91 002C4 45$:   CMPB    SIGN_VAL, #43                            ; 2365
```

```
                              20   12 002C8          BNEQ    50$                                ;
                              0B   11 002CA          BRB     47$                                ;
                  2B      20  AE   91 002CC  46$:    CMPB    SIGN_VAL, #43                      ; 2375
                              13   12 002D0          BNEQ    49$                                ;
                  30          69   91 002D2          CMPB    (BUF), #48                         ; 2377
                              06   12 002D5          BNEQ    48$                                ;
                  50      7B  8F   9A 002D7  47$:    MOVZBL  #123, R0                           ;
                              19   11 002DB          BRB     52$                                ;
                  50          69   9A 002DD  48$:    MOVZBL  (BUF), R0                          ; 2379
                  50          10   C0 002E0          ADDL2   #16, R0                            ;
                              11   11 002E3          BRB     52$                                ; 2377
                  30          69   91 002E5  49$:    CMPB    (BUF), #48                         ; 2381
                              06   12 002E8          BNEQ    51$                                ;
                  50      7D  8F   9A 002EA  50$:    MOVZBL  #125, R0                           ;
                              06   11 002EE          BRB     52$                                ;
                  50          69   9A 002F0  51$:    MOVZBL  (BUF), R0                          ; 2383
                  50          19   C0 002F3          ADDL2   #25, R0                            ;
                  51          50   90 002F6  52$:    MOVB    R0, FIRST_DIGIT                    ; 2375
              04  B6          51   90 002F9          MOVB    FIRST_DIGIT, @4(R6)                ; 2385
                              38   11 002FD  53$:    BRB     60$                                ; 2300
                  50     FF A749   9E 002FF  54$:    MOVAB   -1(R7)[BUF], R0                    ; 2398
                  30          60   91 00304          CMPB    (R0), #48                          ;
                              12   12 00307          BNEQ    56$                                ;
                  2B      20  AE   91 00309          CMPB    SIGN_VAL, #43                      ; 2401
                              06   12 0030D          BNEQ    55$                                ;
                  51      7B  8F   9A 0030F          MOVZBL  #123, R1                           ;
                              1A   11 00313          BRB     58$                                ;
                  51      7D  8F   9A 00315  55$:    MOVZBL  #125, R1                           ;
                              14   11 00319          BRB     58$                                ;
                  2B      20  AE   91 0031B  56$:    CMPB    SIGN_VAL, #43                      ; 2406
                              08   12 0031F          BNEQ    57$                                ;
                  51          60   9A 00321          MOVZBL  (R0), R1                           ; 2407
                  51          10   C0 00324          ADDL2   #16, R1                            ;
                              06   11 00327          BRB     58$                                ;
                  51          60   9A 00329  57$:    MOVZBL  (R0), R1                           ; 2408
                  51          19   C0 0032C          ADDL2   #25, R1                            ;
                  60          51   90 0032F  58$:    MOVB    R1, (R0)                           ; 2406
        68    38  BE          57   28 00332  59$:    MOVC3   R7, @BUF_DESC+4, (DEST_PTR)        ; 2410
                          34  AE   9F 00337  60$:    PUSHAB  BUF_DESC                           ; 2421
         00000000G  00        01   FB 0033A          CALLS   #1, STR$FREE1_DX                   ;
                  50          01   D0 00341  61$:    MOVL    #1, R0                             ; 2422
                              04      00344          RET                                        ;
                              50   D4 00345  62$:    CLRL    R0                                 ; 2423
                              04      00347          RET                                        ;
                            0000      00348  63$:    .WORD   Save nothing                       ; 1899
                  50      08  AC   D0 0034A          MOVL    8(AP), R0                          ;
                  50      04  A0   D0 0034E          MOVL    4(R0), R0                          ;
                          F8  A0   9F 00352          PUSHAB  BUF_DESC                           ;
                              01   DD 00355          PUSHL   #1                                 ;
                              5E   DD 00357          PUSHL   SP                                 ;
                  7E      04  AC   7D 00359          MOVQ    4(AP), -(SP)                       ;
         00000000G  00        03   FB 0035D          CALLS   #3, COB$$FREE_STRINGS              ;
                              04      00364          RET                                        ;

; Routine Size:  869 bytes,    Routine Base:  _COB$CODE + 0180
```

```
%SBTTL 'COB$$FLOAT_CONV - Convert to Floating Point'
ROUTINE COB$$FLOAT_CONV (STRING_DEST   :  REF $STR$DESCRIPTOR,
                                            ! Final destination for input chars
                         FLAGS,             ! Enhancement flag
                         PUT_HERE      :  REF BLOCK [8, BYTE],
                                            ! Contains input characters
                         CHARS_READ         ! # of input characters
                        ) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Convert TEXT input string to Floating or Double Floating Point.
!       Do nothing about errors in this routine, return control to calling
!       routine.
!
! FORMAL PARAMETERS:
!
!       STRING_DEST.mt.ds   Address of descriptor to receive the read input.
!
!       FLAGS.rlu.v      Screen enhancement flag;
!
!       PUT_HERE.rt.dx  Buffer to hold input characters.
!
!       CHARS_READ.rlu.v   Number of characters accepted as input.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
!
!       1 = Conversion Success
!       0 = Conversion Failure
!
! SIDE EFFECTS:
!
!       Signals COB$_INVARG if the syntax of the number is wrong.
!--

    BEGIN
        LOCAL
            CONV_OK  :  INITIAL (0) ;                   ! Conversion flag
                                                        ! =0 error, =1 no error
    LABEL
        FLOAT_PROCESSOR ;

        FLOAT_PROCESSOR:
                BEGIN
                    LOCAL
                        TEMP_PUT_HERE    :  BLOCK [12,BYTE] VOLATILE ;
                                            ! Temporary buffer with length
                                            ! reflecting number of chars read
```

```
3          LITERAL
3              ONLY_E_ALLOWED = 15 ;          ! Bit 0 - blanks are ignored
3                                             ! Bit 1 - only E or e for exp
3                                             ! Bit 2 - underflow is an error
3                                             ! Bit 3 - do not round
3          BIND
3              COMMA = UPLIT (','),
3              DOT = UPLIT ('.');
3
3          !+
3          ! Pick appropriate conversion routine based on
3          ! data type.
3          !-
4          BIND ROUTINE CVTTX = (
4                  IF .STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_F
4                  THEN OTS$CVT_T_F
4
4                  ELSE IF .STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_D
4                  THEN OTS$CVT_T_D
3                  ELSE 0 );
3
3          !+
3          !  Make TEMP_PUT_HERE a fixed length descriptor -
3          !  don't need STR$GET1_DX or STR$FREE1_DX.
3          !-
3          TEMP_PUT_HERE [DSC$W_LENGTH]  = .CHARS_READ ;
3          TEMP_PUT_HERE [DSC$B_DTYPE]   = DSC$K_DTYPE_NL ;
3          TEMP_PUT_HERE [DSC$B_CLASS]   = DSC$K_CLASS_S ;
3          TEMP_PUT_HERE [DSC$A_POINTER] = .PUT_HERE [DSC$A_POINTER] ;
3
3          !+
3          ! If DECIMAL POINT IS COMMA is set and a comma came in,
3          ! must change it to a decimal point before the convert.
3          ! OTS$ routines expect a decimal point.
3          ! Also must not allow a dec pt as input in that case.
3          !-
3          IF (.FLAGS AND V_DEC_PT) NEQ 0
3          THEN
4              BEGIN
4
4              INCR PTR FROM 0 TO (.CHARS_READ - 1) DO
5                  BEGIN
5
5                  IF CH$EQL (1,.TEMP_PUT_HERE [DSC$A_POINTER] + .PTR,
5                             1, DOT)
5                  THEN
6                      BEGIN
6
6                      CONV_OK = 0;              ! Illegal, looking
6                      LEAVE FLOAT_PROCESSOR;    ! for a comma
6
5                      END;
5
5                  IF CH$EQL (1, .TEMP_PUT_HERE [DSC$A_POINTER] + .PTR,
5                             1, COMMA)
5                  THEN
5                      CH$MOVE (1, DOT, .TEMP_PUT_HERE [DSC$A_POINTER] + .PTR);
```

```
4                          END;
4                      END
4                  !+
4                  ! If DECIMAL POINT IS COMMA is NOT set and a comma
4                  ! came in, this is a conversion error
4                  !-
3                  ELSE
4                      BEGIN
4
4                      INCR PTR FROM 0 TO (.CHARS_READ - 1) DO
5                          BEGIN
5
5                          IF CH$EQL (1,.TEMP_PUT_HERE [DSC$A_POINTER] + .PTR,
5                                  1, COMMA)
5                          THEN
6                              BEGIN
6
6                              CONV_OK = 0;            ! Illegal, looking
6                              LEAVE FLOAT_PROCESSOR; ! for a decimal pt
6
5                              END;
4                          END ;
3                      END ;
3
3                  !+
3                  ! Check that input data is not out of range.
3                  !-
4                  BEGIN
4                      LOCAL
4                          MAX ;    ! Maximum significant digits allowed
4
5                  MAX = ( IF .STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_F
5                          THEN 7                      ! Floating
4                          ELSE 16 ) ;                 ! Double Floating
4
4                  CONV_OK = COB$$VERIFY_FL_RANGE ( TEMP_PUT_HERE, .CHARS_READ,
4                                                   .MAX ) ;
4                  IF .CONV_OK
4                  THEN
5                      BEGIN
5                      !+
5                      ! Convert to float or double.
5                      !-
6                      IF NOT ( CVTTX ( TEMP_PUT_HERE,
6                                  .STRING_DEST[DSC$A_POINTER],
6                                  0, 0, ONLY_E_ALLOWED ) )
5                      THEN CONV_OK = 0                ! Conversion error
5                      ELSE
5                          CONV_OK = 1 ;               ! No error
4                      END ;
3                  END ;
2              END ;
2
2      RETURN .CONV_OK ;
1      END ;                                           ! End of COB$$FLOAT_CONV
```

```
                                 004E5         .BLKB    3
            00  00  00  2C  004E8 P.AAB:  .ASCII   \,\<0><0><0>                               ;
            00  00  00  2E  004EC P.AAC:  .ASCII   \.\<0><0><0>                               ;

                                      COMMA=            P.AAB
                                      DOT=              P.AAC

                          007C 00000 COB$$FLOAT_CONV:
                                              .WORD    Save R2,R3,R4,R5,R6                        ; 2425
          56        F7    AF   9E 00002       MOVAB    DOT, R6                                    ;
          5E              0C   C2 00006       SUBL2    #12, SP                                    ;
                          53   D4 00009       CLRL     CONV_OK                                    ; 2468
          52        04    AC   D0 0000B       MOVL     STRING_DEST, R2                            ; 2495
                          54   D4 0000F       CLRL     R4                                         ;
          0A        02    A2   91 00011       CMPB     2(R2), #10                                 ;
                          0B   12 00015       BNEQ     1$                                         ;
                          54   D6 00017       INCL     R4                                         ;
          50 00000000G    00   9E 00019       MOVAB    OTS$CVT_T_F, R0                            ;
                          11   11 00020       BRB      3$                                         ;
          0B        02    A2   91 00022 1$:   CMPB     2(R2), #11                                 ; 2498
                          09   12 00026       BNEQ     2$                                         ;
          50 00000000G    00   9E 00028       MOVAB    OTS$CVT_T_D, R0                            ;
                          02   11 0002F       BRB      3$                                         ;
                          50   D4 00031 2$:   CLRL     R0                                         ;
          55              50   D0 00033 3$:   MOVL     R0, R5                                     ;
          6E        10    AC   B0 00036       MOVW     CHARS_READ, TEMP_PUT_HERE                  ; 2506
      02  AE              10   90 0003A       MOVB     #16, TEMP_PUT_HERE+2                       ; 2507
      03  AE              01   90 0003E       MOVB     #1, TEMP_PUT_HERE+3                        ; 2508
          50        0C    AC   D0 00042       MOVL     PUT_HERE, R0                               ; 2509
      04  AE        04    A0   D0 00046       MOVL     4(R0), TEMP_PUT_HERE+4                     ;
 51   10  AC              01   C3 0004B       SUBL3    #1, CHARS_READ, R1                         ; 2521
 1F   08  AC              06   E1 00050       BBC      #6, FLAGS, 6$                              ; 2517
          50              01   CE 00055       MNEGL    #1, PTR                                    ; 2524
                          14   11 00058       BRB      5$                                         ;
          66        04 BE40    91 0005A 4$:   CMPB     @TEMP_PUT_HERE+4[PTR], DOT                 ;
                          52   13 0005F       BEQL     12$                                        ;
      FC  A6        04 BE40    91 00061       CMPB     @TEMP_PUT_HERE+4[PTR], COMMA               ; 2534
                          05   12 00067       BNEQ     5$                                         ;
      04 BE40             66   90 00069       MOVB     DOT, @TEMP_PUT_HERE+4[PTR]                 ; 2537
 E8       50              51   F3 0006E 5$:   AOBLEQ   R1, PTR, 4$                                ; 2521
                          11   11 00072       BRB      9$                                         ; 2517
          50              01   CE 00074 6$:   MNEGL    #1, PTR                                    ; 2547
                          08   11 00077       BRB      8$                                         ;
      FC  A6        04 BE40    91 00079 7$:   CMPB     @TEMP_PUT_HERE+4[PTR], COMMA               ; 2550
                          32   13 0007F       BEQL     12$                                        ;
 F4       50              51   F3 00081 8$:   AOBLEQ   R1, PTR, 7$                                ; 2547
          05              54   E9 00085 9$:   BLBC     R4, 10$                                    ; 2569
          50              07   D0 00088       MOVL     #7, MAX                                    ;
                          03   11 0008B       BRB      11$                                        ;
          50              10   D0 0008D 10$:  MOVL     #16, MAX                                   ;
                          50   DD 00090 11$:  PUSHL    MAX                                        ; 2574
                    10    AC   DD 00092       PUSHL    CHARS_READ                                 ; 2573
                    08    AE   9F 00095       PUSHAB   TEMP_PUT_HERE                              ;
      0000V  CF           03   FB 00098       CALLS    #3, COB$$VERIFY_FL_RANGE                   ;
          53              50   D0 0009D       MOVL     R0, CONV_OK                                ;
```

```
        17          53  E9 000A0         BLBC    CONV_OK, 14$                ; 2575
                    0F  DD 000A3         PUSHL   #15                         ; 2581
                    7E  7C 000A5         CLRQ    -(SP)                       ;
              04    A2  DD 000A7         PUSHL   4(R2)                       ; 2582
              10    AE  9F 000AA         PUSHAB  TEMP_PUT_HERE               ; 2581
        65          05  FB 000AD         CALLS   #5, (R5)                    ;
        04          50  E8 000B0         BLBS    R0, 13$                     ;
                    53  D4 000B3  12$:   CLRL    CONV_OK                     ; 2584
                    03  11 000B5         BRB     14$                         ;
        53          01  D0 000B7  13$:   MOVL    #1, CONV_OK                 ; 2586
        50          53  D0 000BA  14$:   MOVL    CONV_OK, R0                 ; 2591
                        04 000BD         RET                                 ; 2592

; Routine Size:  190 bytes,    Routine Base:  _COB$CODE + 04F0
```

```
1 %SBTTL 'COB$$COMP_CONV - Convert to COMP and COMP3'
1 ROUTINE COB$$COMP_CONV (STRING_DEST   :  REF $STR$DESCRIPTOR,
1                                           ! Final destination for input chars
1                         FLAGS,            ! Enhancement flag
1                         DEFAULT       :  REF $STR$DESCRIPTOR,
1                         PUT_HERE      :  REF BLOCK [8, BYTE],
1                                           ! Contains input characters
1                         CHARS_READ,       ! # of input characters
1                         YES_DEFAULT,      ! =1 if DEFAULT value is used
1                         YES_SIGN          ! =1 if sign should be included
1                        ) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Convert TEXT input string to appropriate VAX COBOL COMP or COMP3
1 !       data type.
1 !       Do nothing about errors in this routine, return control to calling
1 !       routine.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       STRING_DEST.mt.ds   Address of descriptor to receive the read input.
1 !
1 !       FLAGS.rlu.v         Screen enhancement flag;
1 !
1 !       DEFAULT.rt.dx       Default source moved to destination descriptor
1 !                           (STRING_DEST) in the event of null input.
1 !
1 !       PUT_HERE.rt.dx      Buffer to hold input characters.
1 !
1 !       CHARS_READ.rlu.v    Number of characters accepted as input.
1 !
1 !       YES_DEFAULT.rlu.v   Flag = 1 if DEFAULT used because of null input.
1 !
1 !       YES_SIGN.rlu.v      Flag = 1 if sign should be included in COMP or COMP3
1 !                           data type.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       1 = Conversion Success
1 !       0 = Conversion Failure
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Signals COB$_INVARG if the syntax of the number is wrong,
1 !--
1
2    BEGIN
```

```
        LOCAL
            CONV_OK   :   INITIAL (0) ;                       ! Conversion flag,
                                                              ! =0 error, =1 no error
                      BEGIN

                      LOCAL
                          EXPONENT: INITIAL (0),              ! Exponent for the CIT
                          I_VALUE: VECTOR[12, BYTE],          ! COBOL intermediate temporary
                          SIGN: BYTE,                         ! Sign of the input string
                          SEND_CHARS_READ;                    ! Local to hold .CHARS_READ

                      !+
                      !  COB$$STRIP_BLANKS_SIGN may change the value of
                      !  SEND_CHARS_READ therefore it is important to send
                      !  this local instead of CHARS_READ (which should NEVER
                      !  be altered).
                      !-
                      SEND_CHARS_READ = .CHARS_READ ;

                      !+
                      ! First must strip off leading and trailing blanks and
                      ! the sign because COB$CVTTI won't accept them.
                      !-
                      IF .YES_DEFAULT
                      THEN
                          !+
                          !  This move must be done because STRIP routine
                          !  writes back into the 1st parameter and DEFAULT
                          !  is read-only.
                          !-
                          CH$MOVE (.CHARS_READ, .DEFAULT [DSC$A_POINTER],
                                  .PUT_HERE [DSC$A_POINTER]);

                      CONV_OK = COB$$STRIP_BLANKS_SIGN (.PUT_HERE, .STRING_DEST,
                                      SEND_CHARS_READ, EXPONENT, SIGN, .FLAGS);
                      IF .CONV_OK
                      THEN
                          !+
                          ! Convert the stripped input string to CIT.
                          ! Must do convert to CIT, THEN to destination data
                          ! type because these COBOL conversion routines take
                          ! into account the scale factor.
                          !-
                          CONV_OK = COB$CVTTI_R8 (.SEND_CHARS_READ,
                              .PUT_HERE [DSC$A_POINTER], I_VALUE);

                      IF .CONV_OK
                      THEN
                          BEGIN

                          LOCAL
                              SCALE;

                          !+
                          ! Pick the appropriate conversion routine based
                          ! on data type.  Note that routines have the
                          ! same linkage.  At this point we can be sure that
```

```
4                 ! we are dealing with one of these six data types.
4                 !-
5                 BIND ROUTINE CVTIX = (
5                     IF .STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_W
5                     THEN COB$CVTIW_R8
5
5                     ELSE IF .STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_WU
5                     THEN COB$CVTIW_R8
5
5                     ELSE IF .STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_L
5                     THEN COB$CVTIL_R8
5
5                     ELSE IF .STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_LU
5                     THEN COB$CVTIL_R8
5
5                     ELSE IF .STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_Q
5                     THEN COB$CVTIQ_R8
5
5                     ELSE IF .STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_QU
5                     THEN COB$CVTIQ_R8
4                     ELSE 0) : JSB_678 ;
4
4                 !+
4                 ! First must re-insert the sign in the CIT.
4                 ! The sign of the CIT is contained in byte 12
4                 ! (see appendix C of the RTL Ref Manual for a
4                 ! description of the CIT).  'C' means + and 'D'
4                 ! means -.  COB$CVTTI always returns a positive
4                 ! number, so if the input number was really
4                 ! negative, must make the sign byte negative.
4                 ! Check to see if sign should be included.
4                 !-
4                 IF .SIGN EQL %C'-' AND .YES_SIGN
4                 THEN
4                         I_VALUE[11] = .I_VALUE[11] + 1;
4
4                 !+
4                 ! Next must insert the exponent in the CIT.
4                 ! The first word of the CIT contains the exponent.
4                 !-
4                 CH$MOVE (2, EXPONENT, I_VALUE);
4
4                 !+
4                 ! Convert from CIT to destination data type taking
4                 ! into account the scale factor.
4                 !-
5                 SCALE = ( IF .STRING_DEST [DSC$B_CLASS] EQL DSC$K_CLASS_SD
5                           THEN -.STRING_DEST [DSC$B_SCALE]
4                           ELSE 0 );
4
5                 CONV_OK = ( IF CVTIX EQL 0
5                             THEN
5                                 !+
5                                 ! Packed
5                                 !-
5                                 COB$CVTIP_R9 (.SCALE, I_VALUE,
5                                     .STRING_DEST [DSC$W_LENGTH], .STRING_DEST [DSC$A_POINTER])
```

```
5                              ELSE
5                                  !+
5                                  ! Integer types
5                                  !-
4                                  CVTIX (.SCALE, I_VALUE, .STRING_DEST [DSC$A_POINTER]) ) ;
3                          END;
2                  END ;
2
2      RETURN .CONV_OK ;
1      END ;                                        ! End of COB$$COMP_CONV
```

```
                                   0FFC 00000 COB$$COMP_CONV:
                                                   .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11   ; 2594
                    5E        18  C2 00002         SUBL2   #24, SP                                ;
                              5B  D4 00005         CLRL    CONV_OK                                ; 2649
                          04  AE  D4 00007         CLRL    EXPONENT                               ; 2653
               08   AE    14  AC  D0 0000A         MOVL    CHARS_READ, SEND_CHARS_READ            ; 2667
                    0F    18  AC  E9 0000F         BLBC    YES_DEFAULT, 1$                        ; 2673
                    51    0C  AC  D0 00013         MOVL    DEFAULT, R1                            ; 2680
                    50    10  AC  D0 00017         MOVL    PUT_HERE, R0                           ; 2681
 04    B0      04   B1    14  AC  28 0001B         MOVC3   CHARS_READ, @4(R1), @4(R0)             ;
                          08  AC  DD 00022 1$:     PUSHL   FLAGS                                  ; 2684
                          04  AE  9F 00025         PUSHAB  SIGN                                   ; 2683
                          0C  AE  9F 00028         PUSHAB  EXPONENT                               ;
                          14  AE  9F 0002B         PUSHAB  SEND_CHARS_READ                        ;
                    5A    04  AC  D0 0002E         MOVL    STRING_DEST, R10                       ;
                              5A  DD 00032         PUSHL   R10                                    ;
                    52    10  AC  D0 00034         MOVL    PUT_HERE, R2                           ;
                              52  DD 00038         PUSHL   R2                                     ;
            0000V   CF        06  FB 0003A         CALLS   #6, COB$$STRIP_BLANKS_SIGN             ;
                    5B        50  D0 0003F         MOVL    R0, CONV_OK                            ;
                    15        5B  E9 00042         BLBC    CONV_OK, 2$                            ; 2685
                    58    0C  AE  9E 00045         MOVAB   I_VALUE, R8                            ; 2693
                    57    04  A2  D0 00049         MOVL    4(R2), R7                              ;
                    56    08  AE  D0 0004D         MOVL    SEND_CHARS_READ, R6                    ;
                     00000000G    00  16 00051         JSB     COB$CVTTI_R8                           ;
                    5B        50  D0 00057         MOVL    R0, CONV_OK                            ;
                    03        5B  E8 0005A 2$:     BLBS    CONV_OK, 3$                            ; 2696
                            009D  31 0005D         BRW     18$                                    ;
                    50    02  AA  9A 00060 3$:     MOVZBL  2(R10), R0                             ; 2710
                    07        50  91 00064         CMPB    R0, #7                                 ;
                              09  12 00067         BNEQ    4$                                     ;
                 51 00000000G 00  9E 00069         MOVAB   COB$CVTIW_R8, R1                       ;
                              45  11 00070         BRB     12$                                    ;
                    03        50  91 00072 4$:     CMPB    R0, #3                                 ; 2713
                              09  12 00075         BNEQ    5$                                     ;
                 51 00000000G 00  9E 00077         MOVAB   COB$CVTIW_R8, R1                       ;
                              1F  11 0007E         BRB     8$                                     ;
                    08        50  91 00080 5$:     CMPB    R0, #8                                 ; 2716
                              09  12 00083         BNEQ    7$                                     ;
                 51 00000000G 00  9E 00085 6$:     MOVAB   COB$CVTIL_R8, R1                       ;
                              11  11 0008C         BRB     8$                                     ;
                    04        50  91 0008E 7$:     CMPB    R0, #4                                 ; 2719
```

```
                              F2  13 00091          BEQL    6$                       ;
          09                  50  91 00093          CMPB    R0, #9                   ; 2722
                              0C  12 00096          BNEQ    9$                       ;
          51 00000000G        00  9E 00098          MOVAB   COB$CVTIQ_R8, R1         ;
          50                  51  D0 0009F  8$:     MOVL    R1, R0                   ;
                              10  11 000A2          BRB     11$                      ;
          05                  50  91 000A4  9$:     CMPB    R0, #5                   ; 2725
                              09  12 000A7          BNEQ    10$                      ;
          50 00000000G        00  9E 000A9          MOVAB   COB$CVTIQ_R8, R0         ;
                              02  11 000B0          BRB     11$                      ;
                              50  D4 000B2  10$:    CLRL    R0                       ;
          51                  50  D0 000B4  11$:    MOVL    R0, R1                   ; 2713
          2D                  6E  91 000B7  12$:    CMPB    SIGN, #45                ; 2739
                              07  12 000BA          BNEQ    13$                      ;
          03            1C    AC  E9 000BC          BLBC    YES_SIGN, 13$            ;
                        17    AE  96 000C0          INCB    I_VALUE+11               ; 2741
    0C    AE            04    AE  B0 000C3  13$:    MOVW    EXPONENT, I_VALUE        ; 2747
          09            03    AA  91 000C8          CMPB    3(R10), #9               ; 2753
                              09  12 000CC          BNEQ    14$                      ;
          56            08    AA  98 000CE          CVTBL   8(R10), SCALE            ; 2754
          56                  56  CE 000D2          MNEGL   SCALE, SCALE             ;
                              02  11 000D5          BRB     15$                      ;
                              56  D4 000D7  14$:    CLRL    SCALE                    ; 2753
                              51  D5 000D9  15$:    TSTL    R1                       ; 2757
                              13  12 000DB          BNEQ    16$                      ;
          57            0C    AE  9E 000DD          MOVAB   I_VALUE, R7              ; 2762
          59            04    AA  D0 000E1          MOVL    4(R10), R9               ;
          58                  6A  3C 000E5          MOVZWL  (R10), R8                ;
             00000000G        00  16 000E8          JSB     COB$CVTIP_R9             ;
                              0A  11 000EE          BRB     17$                      ;
          57            0C    AE  9E 000F0  16$:    MOVAB   I_VALUE, R7              ; 2768
          58            04    AA  D0 000F4          MOVL    4(R10), R8               ;
                              61  16 000F8          JSB     (R1)                     ;
          5B                  50  D0 000FA  17$:    MOVL    R0, CONV_OK              ;
          50                  5B  D0 000FD  18$:    MOVL    CONV_OK, R0              ; 2772
                                  04 00100          RET                              ; 2773

; Routine Size:  257 bytes,    Routine Base:  _COB$CODE + 05AE
```

```
1 %SBTTL 'COB$$STRIP_BLANKS_SIGN - Pull blanks and sign'
1 ROUTINE COB$$STRIP_BLANKS_SIGN (                        ! Strip blanks and sign from
1                                                         ! input numeric string
1         INPUT_STRING    : REF $STR$DESCRIPTOR,          ! Numeric string to scan
1         STRING_DEST     : REF $STR$DESCRIPTOR,          ! Destination string
1         NUM_DIGITS,                                     ! Number of characters read
1                                                         ! Will contain the number of
1                                                         ! chars after the stripping
1         EXPONENT,                                       ! Will contain the exponent
1         SIGN_VAL        : REF BLOCK [,BYTE],            ! Will contain the sign char
1         FLAGS   ) =                                     ! Needed for COMMA verification
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Strips leading and trailing blanks from the input numeric string.
1 !       Puts the remaining string back and adjusts the number of digits
1 !       accordingly.  Also strips off the sign char and puts it into the
1 !       output parameter SIGN_VAL.
1 !       Strips the decimal point and figures out the exponent.
1 !       Does nothing about errors in this routine, returns control to
1 !       calling routine.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       INPUT_STRING.rt.dx        The numeric string to scan
1 !       STRING_DESC.rt.dx         The destination string
1 !       NUM_DIGITS.ml.r           As input, its the number of chars read
1 !                                 As output, its the number of chars after
1 !                                 the stripping
1 !       EXPONENT.wl.r             The exponent
1 !       SIGN_VAL.wb.r             The sign character
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       0 = failure, 1 = success
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !--
1
2   BEGIN
2
2   LOCAL
2       TEMP_NUM_DIGITS : INITIAL (0),                  ! Tally of stripped number of digits
2       BUF_DESC : BLOCK [8, BYTE] VOLATILE,            ! Temporary buffer
2       SIGN_SEEN : INITIAL (0),                        ! 1 = we have seen a + or -
2       DIGIT_SEEN : INITIAL (0),                       ! 1 = we have seen at least one digit
```

```
        DOT_SEEN : INITIAL (0),                 ! 1 = we have seen a decimal point
        ZERO_SEEN   :  INITIAL (0),             ! 1 = zero seen
        BLANKS_SEEN : INITIAL (0),              ! 1 = we have seen trailing blanks
        PUTTER : INITIAL (0),                   ! Counts position in the output buffer
        LEFT_DEC : INITIAL (0),                 ! Number of digits to left of dec. pt.
        PDATA_FLAG: INITIAL (0),                ! Flag to indicate if a P Picture item
        BUF : REF VECTOR [1100, BYTE],          ! Addresses result
        INP : REF VECTOR [1100, BYTE],          ! Addresses result in input_string
        LEADING_ZEROES:INITIAL (0),             ! Counter of leading zeroes
        ARG : REF VECTOR [1100, BYTE];          ! Addresses source

    LITERAL
        SUCCESS = 1,
        FAILURE = 0;

   BIND
        ZERO = UPLIT ('0');
!+
! Enable a handler to free the local string in case of an error.
!-

    ENABLE
        COB$$FREE_STRINGS (BUF_DESC);

!+
! If there were no digits input, it means that a <CR> was hit.
!-
    IF ..NUM_DIGITS EQL 0
    THEN
        RETURN SUCCESS;

!+
! Allocate enough space to hold the digits.  It is convenient to
! allocate before scanning, so we may allocate a little too much,
! but the space will be freed before we return.
!-
    BUF_DESC [DSC$W_LENGTH] = 0;
    BUF_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
    BUF_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
    BUF_DESC [DSC$A_POINTER] = 0;
    STR$GET1_DX (.NUM_DIGITS, BUF_DESC);

!+
!  Set pointers.
!-

    BUF = .BUF_DESC [DSC$A_POINTER];
    ARG = .INPUT_STRING [DSC$A_POINTER];
    .SIGN_VAL = %C'+';

!+
!  Scan the input number, put result in BUF.
!-

    IF NOT ( COB$$SCAN_INPUT ( .INPUT_STRING, ..NUM_DIGITS, .FLAGS, BUF_DESC,
                      LEFT_DEC, TEMP_NUM_DIGITS, .SIGN_VAL, PUTTER,
                      LEADING_ZEROES, SIGN_SEEN, DIGIT_SEEN, DOT_SEEN,
                      ZERO_SEEN, BLANKS_SEEN ) )
```

```
	THEN
	    RETURN 0 ;

!+
! If there are no digits, or only leading zeros, take the number to
! be zero.  Don't be too gullible, however.
!-
	IF ( NOT .DIGIT_SEEN)
	THEN
	    BEGIN

	    IF (.SIGN_SEEN OR .DOT_SEEN OR .BLANKS_SEEN) AND (.ZERO_SEEN EQL 0)
	    THEN RETURN FAILURE ;

	    BUF [.PUTTER] = %C'0' ;
	    PUTTER = .PUTTER + 1 ;
	    TEMP_NUM_DIGITS = .TEMP_NUM_DIGITS + 1;

	    END

!+
!  Validate size of entered data, left and right of decimal point.
!-
	ELSE
	    BEGIN

	    LOCAL
		DEST_LENGTH ,				! Destination length
		OK_LEFT ,				! Correct number of digits allowed
							! to left of decimal point
		RIGHT_DEC   :  INITIAL (0) ;		! Number of digits to right of dec pt


	    IF NOT (.DOT_SEEN)				! No dec pt. therefore
	    THEN LEFT_DEC = .TEMP_NUM_DIGITS ;		! all digits are left_dec
	    RIGHT_DEC = .TEMP_NUM_DIGITS - .LEFT_DEC ;
	    !+
	    ! Strip trailing zeroes after the decimal point.
	    !-
	    INCR GETTER FROM 1 TO .RIGHT_DEC DO
		IF .BUF [.TEMP_NUM_DIGITS - .GETTER] EQL %C'0'
		THEN
		    RIGHT_DEC = .RIGHT_DEC - 1
		ELSE
		    EXITLOOP;


	    DEST_LENGTH = .STRING_DEST [DSC$W_LENGTH];

	    SELECTONE .STRING_DEST [DSC$B_CLASS] OF
		SET

		[ DSC$K_CLASS_S ] :

		    BEGIN

		    !+
```

```
		! If a decimal point was typed in, all the digits after it
		! MUST be zeroes.
		!-

		IF .RIGHT_DEC GTR 0
		THEN
		    INCR I FROM (.PUTTER - .RIGHT_DEC) TO .PUTTER DO
			IF .BUF[.I] NEQ %C'0'
			THEN RETURN FAILURE;

		IF .LEFT_DEC GTR .DEST_LENGTH
		THEN RETURN FAILURE;				! Data entered too big

		END ;

	    [ DSC$K_CLASS_SD ] :

		BEGIN

		LOCAL
		    LENGTH_DIFF,				! Difference between number of digits
								!  to the left of the decimal point
								!  in the typed in number and the dest
		    LENGTH_DIFF2;				! Difference between number of digits
								!  to the right of the decimal point
								!  in the typed in number and the dest
		!+
		! This is checking for the P Picture of 99PP.
		! If the scale is positive and the number of digits in the
		! number equal the scale factor, then simply copy the digits
		! in BUF to the destination descriptor.
		! NOTE: Code for P Picture left in lowercase.
		!-
		if .string_dest[dsc$b_scale] gtr 0
		then
		    begin

		    local
			tot_digits,
			diff;

		    tot_digits = (.string_dest[dsc$b_digits] + .string_dest[dsc$b_scale]);
		    if ((.right_dec gtr 0) or (.temp_num_digits gtr .tot_digits))
		    then					! number too large
			return 0;				! re-prompt - error

		    if .temp_num_digits leq .string_dest[dsc$b_scale]
		    then
			begin

			str$dupl_char (.input_string,temp_num_digits,zero);
			pdata_flag = 1;				! answer is zero

			end
		    else
			begin
			!+
```

```
6          ! Zero out the destination field using the digits as
6          ! proper number of zero fill characters, rather than
6          ! using the length as found in the descriptor, since
6          ! class SD is a special case.
6          !-
6          str$dupl_char (.input_string,tot_digits,zero);
6          if .leading_zeroes neq 0
6          then
7              begin

7              diff = (.string_dest[dsc$b_digits] + .string_dest[dsc$b_scale]) - .temp_num_digi
7              ch$move (.string_dest[dsc$b_digits],.buf,.input_string[dsc$a_pointer]+.diff);
7              end
6          else
7              begin

7              diff = .temp_num_digits - .string_dest[dsc$b_scale];
7              if .diff eql .string_dest[dsc$b_digits]
7              then
7                  ch$move (.diff,.buf,.input_string[dsc$a_pointer])
7              else
8                  begin

8                  tot_digits = .tot_digits - .temp_num_digits;
8                  ch$move (.diff,.buf,.input_string[dsc$a_pointer]+.tot_digits);

7                  end;

6              end;
6          pdata_flag = 1;
6          .num_digits = .string_dest[dsc$b_digits] + .string_dest[dsc$b_scale];

5          end;

5  end
4  else
5      begin

5      if .string_dest[dsc$b_scale] gtr 0
5      then
5          ok_left = .string_dest[dsc$b_digits]
5      else
5          OK_LEFT = .STRING_DEST [DSC$B_DIGITS] + .STRING_DEST [DSC$B_SCALE] ;
5      if .ok_left lss 0
5      then
6          begin
6          !
6          ! Here we have a P Picture field of type PP99.
6          ! We know this when OK_LEFT is less than zero.
6          ! It requires some special casing.
6          !-
6          local
6              diff,
6              ok_right,
6              buf_ptr;

6          if .left_dec gtr 0              ! error no '.' entered
```

```
6                  then                                        ! ring bell and reprompt
6                      return 0;
6
6                  ok_left = 0;
6                  ok_right = abs(.string_dest[dsc$b_scale]);
6                  if .right_dec gtr .ok_right
6                  then
6                      return 0;
6
6                  !+
6                  ! This handles case where the number of digits entered
6                  ! is less than the absolute value of the scale factor,
6                  ! meaning that the number returned would have to be 0.
6                  ! The first part of the if statement takes care of the
6                  ! case where the number of digits entered equals the
6                  ! number of digits expected taking into account if the
6                  ! absolute value of the scale factor is equal to the
6                  ! number of digits entered to the right of the decimal
6                  ! point thereby giving us a result of zero again.
6                  !-
6                  diff = (abs(.string_dest[dsc$b_scale]) - .string_dest[dsc$b_digits]);
7                  if ((.right_dec eql .string_dest[dsc$b_digits]) and
6                      (.right_dec eql .diff)) OR
7                      (.right_dec leq .diff)
6                  then
7                      begin
7
7                      str$dupl_char (.input_string,temp_num_digits,zero);
7                      pdata_flag = 1;                     ! done - result is zero
7
7                      end
6                  else
7                      begin
7
7                      buf_ptr = .buf_desc[dsc$a_pointer] + .diff;
7                      diff = .right_dec - .diff;          ! move only necessary digits
7                      !+
7                      ! Zero out the destination field using the digits
7                      ! as the proper number of zero fill characters,
7                      ! rather than using the length as found in the
7                      ! descriptor, since class SD is a special case.
7                      !-
7                      str$dupl_char (.input_string,temp_num_digits,zero);
7                      ch$move (.diff,.buf_ptr,(.input_string[dsc$a_pointer]+(.temp_num_digits-.diff)))
7                      pdata_flag = 1;
7
6                      end;
6
6                  end
5              else
5
6                  begin                                   ! ok left is not < zero
6                  OK_LEFT = .STRING_DEST [DSC$B_DIGITS] + .STRING_DEST [DSC$B_SCALE] ;
6                  LENGTH_DIFF = .OK_LEFT - .LEFT_DEC;
6                  LENGTH_DIFF2 = (.STRING_DEST [DSC$B_DIGITS] - .OK_LEFT) - .RIGHT_DEC;
7                  IF ( .LENGTH_DIFF LSS 0)
6                          OR
```

```
7                               ( .LENGTH_DIFF2 LSS 0)
6                                 THEN RETURN FAILURE ;                 ! Data entered too big
5                               end;
4                            end;
3                         END ;
3
3                   [ OTHERWISE ] :
3
3                         LIB$STOP ( COB$_INVARG ) ;
3
3                   TES ;
3
2               END;
2
2   !+
2   ! Figure out the exponent.
2   !-
3           IF NOT (.DOT_SEEN)                          ! No decimal pt, therefore
2           THEN                                        ! all digits are left_dec
2               LEFT_DEC = .TEMP_NUM_DIGITS;
2
2           IF .LEFT_DEC EQL 0
2           THEN
2               !+
2               ! Figure out exponent if all digits are to right of decimal point.
2               !-
2           if .pdata_flag
2             then
3               begin
3               inp = .input_string [dsc$a_pointer];    ! point to re-written data
3               .EXPONENT = 0;
3               INCR GETTER FROM 0 TO (.TEMP_NUM_DIGITS - 1) DO
3                   !+
3                   ! Exponent decreases for every leading zero.
3                   !-
3                   IF .inp [.GETTER] EQL %C'0'
3                   THEN
3                       .EXPONENT = ..EXPONENT - 1
3                   ELSE
3                       EXITLOOP;
3
3               END
2           ELSE
3               BEGIN
3               .EXPONENT = 0;
3               INCR GETTER FROM 0 TO (.TEMP_NUM_DIGITS - 1) DO
3                   !+
3                   ! Exponent decreases for every leading zero.
3                   !-
3                   IF .BUF [.GETTER] EQL %C'0'
3                   THEN
3                       .EXPONENT = ..EXPONENT - 1
3                   ELSE
3                       EXITLOOP;
3
3               END
2           ELSE
```

```
2            !+
2            ! If all digits are to left of decimal point, exponent is equal
2            ! to left_dec.
2            !-
2            .EXPONENT = .LEFT_DEC;
2
2    !+
2    ! Move the stripped string into the buffer and adjust the number-of-digits
2    !-
2
2        if .pdata_flag eql 0
2        then
3            begin
3            CH$MOVE (.TEMP_NUM_DIGITS, .BUF_DESC [DSC$A_POINTER],
3                    .INPUT_STRING [DSC$A_POINTER]);
3            .NUM_DIGITS = .TEMP_NUM_DIGITS;
3            end
2        else
2            if .string_dest[dsc$b_scale] lss 0
2            then
2                .num_digits = .temp_num_digits ;
2
2    !+
2    ! Free our local string
2    !-
2        STR$FREE1_DX (BUF_DESC);
2        RETURN SUCCESS ;
2
1        END;                                        ! end of COB$$STRIP_BLANKS_SIGN


                                     006AF          .BLKB   1
             00  00  00  30          006B0 P.AAD:   .ASCII  \0\<0><0><0>                           ;

                                           ZERO=            P.AAD


                                OFFC 00000 COB$$STRIP_BLANKS_SIGN:
                                                    .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11   ; 2775
         5E              1C  C2 00002               SUBL2   #28, SP                                ;
                   10    AE  D4 00005               CLRL    TEMP_NUM_DIGITS                        ; 2824
                         7E  7C 00008               CLRQ    DIGIT_SEEN                             ;
                         7E  7C 0000A               CLRQ    ZERO_SEEN                              ;
                         7E  D4 0000C               CLRL    BLANKS_SEEN                            ;
                   18    AE  7C 0000E               CLRQ    PUTTER                                 ;
                         7E  D4 00011               CLRL    PDATA_FLAG                             ;
                   18    AE  D4 00013               CLRL    LEADING_ZEROES                         ;
                   2C    AE  7C 00016               CLRQ    BUF_DESC                               ;
         6D      027F    CF  DE 00019               MOVAL   45$, (FP)                              ;
                   0C    BC  D5 0001E               TSTL    @NUM_DIGITS                            ; 2858
                         03  12 00021               BNEQ    1$                                     ;
                       026F  31 00023               BRW     43$                                    ;
                   2C    AE  B4 00026 1$:           CLRW    BUF_DESC                               ; 2867
     2E  AE              0F  90 00029               MOVB    #15, BUF_DESC+2                        ; 2868
     2F  AE              02  90 0002D               MOVB    #2, BUF_DESC+3                         ; 2869
                   30    AE  D4 00031               CLRL    BUF_DESC+4                             ; 2870
```

```
                              2C   AE   9F 00034         PUSHAB  BUF_DESC                          ; 2871
                              0C   AC   DD 00037         PUSHL   NUM_DIGITS                        ;
             00000000G  00              02 FB 0003A      CALLS   #2, STR$GET1_DX                   ;
                        58    30   AE   D0 00041         MOVL    BUF_DESC+4, BUF                   ; 2876
                        59    04   AC   D0 00045         MOVL    INPUT_STRING, R9                  ; 2877
                        5A    04   A9   9E 00049         MOVAB   4(R9), R10                        ;
                        50         6A   D0 0004D         MOVL    (R10), ARG                        ;
                 14     BC         2B   D0 00050         MOVL    #43, @SIGN_VAL                    ; 2878
                              04   AE   9F 00054         PUSHAB  BLANKS_SEEN                       ; 2884
                              0C   AE   9F 00057         PUSHAB  ZERO_SEEN                         ;
                              14   AE   9F 0005A         PUSHAB  DOT_SEEN                          ;
                              1C   AE   9F 0005D         PUSHAB  DIGIT_SEEN                        ;
                              24   AE   9F 00060         PUSHAB  SIGN_SEEN                         ;
                              2C   AE   9F 00063         PUSHAB  LEADING_ZEROES                    ;
                              34   AE   9F 00066         PUSHAB  PUTTER                            ;
                              14   AC   DD 00069         PUSHL   SIGN_VAL                          ; 2885
                              48   AE   9F 0006C         PUSHAB  TEMP_NUM_DIGITS                   ; 2884
                              44   AE   9F 0006F         PUSHAB  LEFT_DEC                          ;
                              54   AE   9F 00072         PUSHAB  BUF_DESC                          ;
                              18   AC   DD 00075         PUSHL   FLAGS                             ;
                              0C   BC   DD 00078         PUSHL   @NUM_DIGITS                       ;
                                   59   DD 0007B         PUSHL   R9                                ;
                 0000V  CF         0E   FB 0007D         CALLS   #14, COB$$SCAN_INPUT              ;
                        7C         50   E9 00082         BLBC    R0, 13$                           ;
                        1F    10   AE   E8 00085         BLBS    DIGIT_SEEN, 5$                    ; 2895
                        08    14   AE   E8 00089         BLBS    SIGN_SEEN, 2$                     ; 2899
                        04    0C   AE   E8 0008D         BLBS    DOT_SEEN, 2$                      ;
                        05    04   AE   E9 00091         BLBC    BLANKS_SEEN, 3$                   ;
                              08   AE   D5 00095   2$:   TSTL    ZERO_SEEN                         ;
                                   67   13 00098         BEQL    13$                               ;
                 1C BE48           30   90 0009A   3$:   MOVB    #48, @PUTTER[BUF]                 ; 2902
                              1C   AE   D6 0009F         INCL    PUTTER                            ; 2903
                              28   AE   D6 000A2         INCL    TEMP_NUM_DIGITS                   ; 2904
                                 017D   31 000A5   4$:   BRW     31$                               ; 2895
                                   56   D4 000A8   5$:   CLRL    RIGHT_DEC                         ; 2912
                        05    0C   AE   E8 000AA         BLBS    DOT_SEEN, 6$                      ; 2921
                 20     AE    28   AE   D0 000AE         MOVL    TEMP_NUM_DIGITS, LEFT_DEC         ; 2922
                        54    28   AE   D0 000B3   6$:   MOVL    TEMP_NUM_DIGITS, R4               ; 2923
        56              54    20   AE   C3 000B7         SUBL3   LEFT_DEC, R4, RIGHT_DEC           ;
                        52         56   D0 000BC         MOVL    RIGHT_DEC, R2                     ; 2927
                                   50   D4 000BF         CLRL    GETTER                            ;
                                   0C   11 000C1         BRB     8$                                ;
        51              54         50   C3 000C3   7$:   SUBL3   GETTER, R4, R1                    ; 2928
                        30       6148   91 000C7         CMPB    (R1)[BUF], #48                    ;
                                   06   12 000CB         BNEQ    9$                                ;
                                   56   D7 000CD         DECL    RIGHT_DEC                         ; 2930
        F0              50         52   F3 000CF   8$:   AOBLEQ  R2, GETTER, 7$                    ; 2928
                        50    08   AC   D0 000D3   9$:   MOVL    STRING_DEST, R0                   ; 2935
                        52         60   3C 000D7         MOVZWL  (R0), DEST_LENGTH                 ;
                        51    03   A0   9A 000DA         MOVZBL  3(R0), R1                         ; 2937
                        01         51   91 000DE         CMPB    R1, #1                            ; 2940
                                   21   12 000E1         BNEQ    14$                               ;
                                   56   D5 000E3         TSTL    RIGHT_DEC                         ; 2949
                                   14   15 000E5         BLEQ    12$                               ;
        51       1C     AE         56   C3 000E7         SUBL3   RIGHT_DEC, PUTTER, R1             ; 2951
                                   51   D7 000EC         DECL    I                                 ;
                                   06   11 000EE         BRB     11$                               ;
```

```
                        30          6148  91 000F0 10$:   CMPB    (I)[BUF], #48                         ; 2952
                                      0B  12 000F4        BNEQ    13$                                   ;
            F5          51     1C     AE  F3 000F6 11$:   AOBLEQ  PUTTER, I, 10$                        ;
                        52     20     AE  D1 000FB 12$:   CMPL    LEFT_DEC, DEST_LENGTH                 ; 2955
                                      A4  15 000FF        BLEQ    4$                                    ;
                                    0195  31 00101 13$:   BRW     44$                                   ; 2956
                        09            51  91 00104 14$:   CMPB    R1, #9                                ; 2960
                                      03  13 00107        BEQL    15$                                   ;
                                    010C  31 00109        BRW     30$                                   ;
                        53     09     A0  9E 0010C 15$:   MOVAB   9(R0), R3                             ; 2986
                        52     08     A0  98 00110        CVTBL   8(R0), R2                             ; 2978
                                      50  D4 00114        CLRL    R0                                    ;
                                      52  D5 00116        TSTL    R2                                    ;
                                      6A  15 00118        BLEQ    20$                                   ;
                                      50  D6 0011A        INCL    R0                                    ;
                        5B            63  9A 0011C        MOVZBL  (R3), R11                             ; 2986
                        5B            52  C0 0011F        ADDL2   R2, R11                               ;
                   24   AE            5B  D0 00122        MOVL    R11, TOT_DIGITS                       ;
                                      56  D5 00126        TSTL    RIGHT_DEC                             ; 2987
                                      D7  14 00128        BGTR    13$                                   ;
                   24   AE            54  D1 0012A        CMPL    R4, TOT_DIGITS                        ;
                                      D1  14 0012E        BGTR    13$                                   ;
                        6E            01  D0 00130        MOVL    #1, PDATA_FLAG                        ; 2996
                        52            54  D1 00133        CMPL    R4, R2                                ; 2991
                                      03  14 00136        BGTR    16$                                   ;
                                    008A  31 00138        BRW     25$                                   ;
                             FEBD     CF  9F 0013B 16$:   PUSHAB  ZERO                                  ; 3007
                               28     AE  9F 0013F        PUSHAB  TOT_DIGITS                            ;
                                      59  DD 00142        PUSHL   R9                                    ;
              00000000G 00            03  FB 00144        CALLS   #3, STR$DUPL_CHAR                     ;
                               18     AE  D5 0014B        TSTL    LEADING_ZEROES                        ; 3008
                                      0F  13 0014E        BEQL    17$                                   ;
            57          5B            54  C3 00150        SUBL3   R4, R11, DIFF                         ; 3012
                        50            63  9A 00154        MOVZBL  (R3), R0                              ; 3013
       00 BA47          68            50  28 00157        MOVC3   R0, (BUF), @0(R10)[DIFF]              ;
                                      1F  11 0015D        BRB     19$                                   ; 3008
            57          54            52  C3 0015F 17$:   SUBL3   R2, R4, DIFF                          ; 3018
   57       63          08            00  ED 00163        CMPZV   #0, #8, (R3), DIFF                    ; 3019
                                      07  12 00168        BNEQ    18$                                   ;
       00   BA          68            57  28 0016A        MOVC3   DIFF, (BUF), @0(R10)                  ; 3021
                                      0D  11 0016F        BRB     19$                                   ;
                   24   AE            54  C2 00171 18$:   SUBL2   R4, TOT_DIGITS                        ; 3025
            50          6A     24     AE  C1 00175        ADDL3   TOT_DIGITS, (R10), R0                 ; 3026
            60          68            57  28 0017A        MOVC3   DIFF, (BUF), (R0)                     ;
                   0C   BC            5B  D0 0017E 19$:   MOVL    R11, @NUM_DIGITS                      ; 3032
                                      77  11 00182        BRB     27$                                   ; 2978
                        08            50  E9 00184 20$:   BLBC    R0, 21$                               ; 3040
                        51            63  9A 00187        MOVZBL  (R3), R1                              ; 3042
                        50            51  D0 0018A        MOVL    R1, OK_LEFT                           ;
                                      07  11 0018D        BRB     22$                                   ;
                        51            63  9A 0018F 21$:   MOVZBL  (R3), R1                              ; 3044
            50          51            52  C1 00192        ADDL3   R2, R1, OK_LEFT                       ;
                                      65  18 00196 22$:   BGEQ    28$                                   ; 3045
                               20     AE  D5 00198        TSTL    LEFT_DEC                              ; 3058
                                      78  14 0019B        BGTR    29$                                   ;
                                      50  D4 0019D        CLRL    OK_LEFT                               ; 3062
                        50            52  D0 0019F        MOVL    R2, R0                                ; 3063
```

```
                                        03  18 001A2          BGEQ    23$                                    :
                            50          50  CE 001A4          MNEGL   R0, R0                                 :
                            53          50  D0 001A7 23$:     MOVL    R0, OK_RIGHT                           :
                            53          56  D1 001AA          CMPL    RIGHT_DEC, OK_RIGHT                    : 3064
                                        66  14 001AD          BGTR    29$                                    :
                 52         50          51  C3 001AF          SUBL3   R1, R0, DIFF                           : 3079
                            6E          01  D0 001B3          MOVL    #1, PDATA_FLAG                         : 3087
                            51          56  D1 001B6          CMPL    RIGHT_DEC, R1                          : 3080
                                        05  12 001B9          BNEQ    24$                                    :
                            52          56  D1 001BB          CMPL    RIGHT_DEC, DIFF                        : 3081
                                        05  13 001BE          BEQL    25$                                    :
                            52          56  D1 001C0 24$:     CMPL    RIGHT_DEC, DIFF                        : 3082
                                        12  14 001C3          BGTR    26$                                    :
                                  FE33  CF  9F 001C5 25$:     PUSHAB  ZERO                                   : 3086
                                    2C  AE  9F 001C9          PUSHAB  TEMP_NUM_DIGITS                        :
                                        59  DD 001CC          PUSHL   R9                                     :
                      00000000G     00  03  FB 001CE          CALLS   #3, STR$DUPL_CHAR                      :
                                        4E  11 001D5          BRB     31$                                    : 3080
                 53         52    30    AE  C1 001D7 26$:     ADDL3   BUF_DESC+4, DIFF, BUF_PTR              : 3093
                 52         56          52  C3 001DC          SUBL3   DIFF, RIGHT_DEC, DIFF                  : 3094
                                  FE18  CF  9F 001E0          PUSHAB  ZERO                                   : 3101
                                    2C  AE  9F 001E4          PUSHAB  TEMP_NUM_DIGITS                        :
                                        59  DD 001E7          PUSHL   R9                                     :
                      00000000G     00  03  FB 001E9          CALLS   #3, STR$DUPL_CHAR                      :
                 50         28      AE  52  C3 001F0          SUBL3   DIFF, TEMP_NUM_DIGITS, R0              : 3102
              00 BA40               63  52  28 001F5          MOVC3   DIFF, (BUF_PTR), @0(R10)[R0]           :
                                        28  11 001FB 27$:     BRB     31$                                    : 3045
                 50         51          52  C1 001FD 28$:     ADDL3   R2, R1, OK_LEFT                        : 3111
                 52         50    20    AE  C3 00201          SUBL3   LEFT_DEC, OK_LEFT, LENGTH_DIFF         : 3112
                            51          50  C2 00206          SUBL2   OK_LEFT, R1                            : 3113
                 50         51          56  C3 00209          SUBL3   RIGHT_DEC, R1, LENGTH_DIFF2            :
                                        52  D5 0020D          TSTL    LENGTH_DIFF                            : 3114
                                        04  19 0020F          BLSS    29$                                    :
                                        50  D5 00211          TSTL    LENGTH_DIFF2                           : 3116
                                        10  18 00213          BGEQ    31$                                    :
                                      0081  31 00215 29$:     BRW     44$                                    : 3117
                             00000000G  8F  DD 00218 30$:     PUSHL   #COB$_INVARG                           : 3124
                      00000000G     00  01  FB 0021E          CALLS   #1, LIB$STOP                           :
                            05    0C    AE  E8 00225 31$:     BLBS    DOT_SEEN, 32$                          : 3133
                      20    AE    28    AE  D0 00229          MOVL    TEMP_NUM_DIGITS, LEFT_DEC              : 3135
                            50    10    AC  D0 0022E 32$:     MOVL    EXPONENT, R0                           : 3146
                                  20    AE  D5 00232          TSTL    LEFT_DEC                               : 3137
                                        35  12 00235          BNEQ    38$                                    :
                 53         28      AE  01  C3 00237          SUBL3   #1, TEMP_NUM_DIGITS, R3                : 3147
                            18          6E  E9 0023C          BLBC    PDATA_FLAG, 35$                        : 3142
                            51          6A  D0 0023F          MOVL    (R10), INP                             : 3145
                                        60  D4 00242          CLRL    (R0)                                   : 3146
                            52          01  CE 00244          MNEGL   #1, GETTER                             : 3151
                                        08  11 00247          BRB     34$                                    :
                            30        6241  91 00249 33$:     CMPB    (GETTER)[INP], #48                     :
                                        21  12 0024D          BNEQ    39$                                    :
                                        60  D7 0024F          DECL    (R0)                                   : 3153
                 F4         52          53  F3 00251 34$:     AOBLEQ  R3, GETTER, 33$                        : 3151
                                        19  11 00255          BRB     39$                                    : 3142
                                        60  D4 00257 35$:     CLRL    (R0)                                   : 3160
                            51          01  CE 00259          MNEGL   #1, GETTER                             : 3161
                                        08  11 0025C          BRB     37$                                    :
```

```
                        30        6148  91 0025E 36$:   CMPB    (GETTER)[BUF], #48                        ; 3165
                                    0C  12 00262        BNEQ    39$                                       ;
                                    60  D7 00264        DECL    (R0)                                      ; 3167
              F4        51          53  F3 00266 37$:   AOBLEQ  R3, GETTER, 36$                           ; 3165
                                    04  11 0026A        BRB     39$                                       ; 3142
                        60    20    AE  D0 0026C 38$:   MOVL    LEFT_DEC, (R0)                            ; 3177
                                    6E  D5 00270 39$:   TSTL    PDATA_FLAG                                ; 3183
                                    09  12 00272        BNEQ    40$                                       ;
         00   BA    30  BE    28    AE  28 00274        MOVC3   TEMP_NUM_DIGITS, @BUF_DESC+4, @0(R10)     ; 3187
                                    09  11 0027B        BRB     41$                                       ; 3188
                        50    08    AC  D0 0027D 40$:   MOVL    STRING_DEST, R0                           ; 3191
                              08    A0  95 00281        TSTB    8(R0)                                     ;
                                    05  18 00284        BGEQ    42$                                       ;
                  0C    BC    28    AE  D0 00286 41$:   MOVL    TEMP_NUM_DIGITS, @NUM_DIGITS              ; 3193
                              2C    AE  9F 0028B 42$:   PUSHAB  BUF_DESC                                  ; 3198
            00000000G   00          01  FB 0028E        CALLS   #1, STR$FREE1_DX                          ;
                        50          01  D0 00295 43$:   MOVL    #1, R0                                    ; 3199
                                        04 00298        RET                                               ;
                                    50  D4 00299 44$:   CLRL    R0                                        ; 3201
                                        04 0029B        RET                                               ;
                                      0000 0029C 45$:   .WORD   Save nothing                              ; 2824
                        50    08    AC  D0 0029E        MOVL    8(AP), R0                                 ;
                        50    04    A0  D0 002A2        MOVL    4(R0), R0                                 ;
                              F8    A0  9F 002A6        PUSHAB  BUF_DESC                                  ;
                                    01  DD 002A9        PUSHL   #1                                        ;
                                    5E  DD 002AB        PUSHL   SP                                        ;
                        7E    04    AC  7D 002AD        MOVQ    4(AP), -(SP)                              ;
            00000000G   00          03  FB 002B1        CALLS   #3, COB$$FREE_STRINGS                     ;
                                        04 002B8        RET                                               ;
```

; Routine Size: 697 bytes,   Routine Base: _COB$CODE + 06B4

```
%SBTTL 'COB$$ZERO_FILL - Initialize destination'
ROUTINE COB$$ZERO_FILL ( STRING_DEST   :  REF $STR$DESCRIPTOR
                                                   ! Destination for input
                       )  :  NOVALUE  =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine will initialize STRING_DEST to zeroes before the input
!       data is copied to it.
!
! FORMAL PARAMETERS:
!
!       STRING_DEST.mt.ds   Address of descriptor to receive the read input.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
!
!       NONE
!
! SIDE EFFECTS:
!
!--

        BEGIN

        LOCAL
            SIGN        : BYTE,
            ZERO        : BYTE,
            DEST_PTR ,                              ! Pointer where result will go in destination
            DEST_LENGTH ;                           ! Destination length

        LITERAL
            ZERO_0   = 48,                          ! Zero
            POS_SIGN = 43,                          ! Plus sign
            POSZEROP = 123 ;                        ! +0 overpunched


        ZERO = ZERO_0 ;
        DEST_PTR    = .STRING_DEST [DSC$A_POINTER] ;
        DEST_LENGTH = .STRING_DEST [DSC$W_LENGTH] ;

        !+
        ! Zero fill, then handle sign correctly.
        !-

        STR$DUPL_CHAR ( .STRING_DEST, DEST_LENGTH, ZERO ) ;

        CASE .STRING_DEST [DSC$B_DTYPE] FROM DSC$K_DTYPE_NU TO DSC$K_DTYPE_NRO
            OF
```

```
                SET

                [DSC$K_DTYPE_NU]:                                   ! Numeric unsigned
                    !+
                    !   Move all zeroes to STRING_DEST
                    !-
                    0 ;                                             ! No further action

                [DSC$K_DTYPE_NL]:                                   ! Numeric left separate
                    !+
                    !   Move sign then all zeroes to STRING_DEST
                    !-

                    BEGIN

                    SIGN = POS_SIGN ;
                    CH$MOVE (1, SIGN, .STRING_DEST [DSC$A_POINTER]);

                    END;

                [DSC$K_DTYPE_NR]:                                   ! Numeric right separate
                    !+
                    !   Move all zeroes followed by sign to STRING_DEST
                    !-

                    BEGIN

                    SIGN = POS_SIGN ;
                    CH$MOVE ( 1, SIGN, (.DEST_PTR + (.DEST_LENGTH - 1)) ) ;

                    END;

                [DSC$K_DTYPE_NLO]:                                  ! Numeric left overpunched
                    !+
                    !   Move all zeroes to STRING_DEST
                    !   First digit has overpunch sign (positive)
                    !-

                    BEGIN

                    SIGN = POSZEROP ;
                    CH$MOVE (1, SIGN, .STRING_DEST[DSC$A_POINTER]);

                    END;

                [DSC$K_DTYPE_NRO]:                                  ! Numeric right overpunched
                    !+
                    !   Move all zeroes to STRING_DEST
                    !   Last digit has overpunch sign (positive)
                    !-

                    BEGIN

                    SIGN = POSZEROP ;
                    CH$MOVE (1, SIGN, (.DEST_PTR + (.DEST_LENGTH - 1)) );

                    END;
```

```
; 1807        3316  2
; 1808        3317  2           TES;
; 1809        3318  2
; 1810        3319  1       END ;                                       ! End of COB$$ZERO_FILL


                                        000C 00000 COB$$ZERO_FILL:
                                                           .WORD    Save R2,R3                          ; 3203
                        5E              08  C2 00002       SUBL2    #8, SP
                        6E              30  90 00005       MOVB     #48, ZERO                           ; 3247
                        52        04    AC  D0 00008       MOVL     STRING_DEST, R2                     ; 3248
                        53        04    A2  D0 0000C       MOVL     4(R2), DEST_PTR
                   04   AE              62  3C 00010       MOVZWL   (R2), DEST_LENGTH                   ; 3249
                                        5E  DD 00014       PUSHL    SP                                  ; 3255
                                  08    AE  9F 00016       PUSHAB   DEST_LENGTH
                                        52  DD 00019       PUSHL    R2
              00000000G 00              03  FB 0001B       CALLS    #3, STR$DUPL_CHAR
             04         0F        02    A2  8F 00022       CASEB    2(R2), #15, #4                      ; 3257
   0010      0015             000B         002B   00027 1$:       .WORD    8$-1$,-
                                           001E   0002F                    2$-1$,-
                                                                           4$-1$,-
                                                                           3$-1$,-
                                                                           6$-1$
                                        04 00031           RET
                        50              2B  90 00032 2$:   MOVB     #43, SIGN                           ; 3274
                                        09  11 00035       BRB      5$                                  ; 3275
                        50              2B  90 00037 3$:   MOVB     #43, SIGN                           ; 3286
                                        0D  11 0003A       BRB      7$                                  ; 3287
                        50        7B    8F  90 0003C 4$:   MOVB     #123, SIGN                          ; 3299
                   04   B2              50  90 00040 5$:   MOVB     SIGN, @4(R2)                        ; 3300
                                        04 00044           RET                                          ; 3257
                        50        7B    8F  90 00045 6$:   MOVB     #123, SIGN                          ; 3312
             51         53        04    AE  C1 00049 7$:   ADDL3    DEST_LENGTH, DEST_PTR, R1           ; 3313
                   FF   A1              50  90 0004E       MOVB     SIGN, -1(R1)
                                        04 00052 8$:       RET                                          ; 3319

; Routine Size:  83 bytes,    Routine Base:  _COB$CODE + 096D
```

```
1 %SBTTL 'COB$$VERIFY_FL_RANGE - Verify Float Pt range'
1 ROUTINE COB$$VERIFY_FL_RANGE (
1                               TEMP_PUT_HERE  :  REF BLOCK [8, BYTE], ! # to scan
1                               CHARS_READ,             ! # of chars in TEMP_PUT_HERE
1                               MAX                     ! # of significant digits allowed
1                          ) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Check range of Floating and Double Floating Point input data.
1 !       Do nothing about errors in this routine, return to calling routine.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       TEMP_PUT_HERE.rt.dx     Input data to be verified.
1 !
1 !       CHARS_READ.rlu.v        Number of input characters.
1 !
1 !       MAX.rlu.v               Number of significant digits allowed in
1 !                               mantissa of E notation representation.
1 !                               7 for Floating Point
1 !                               16 for Double Floating Point
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       1 = SUCCESS
1 !       0 = FAILURE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !--
1
2     BEGIN
2     !+
2     ! This routine counts the significant digits (1-9) and significant zeroes
2     ! of data that is input to either a floating point or a double floating
2     ! point data item.   Some zeroes can be ignored.
2     !
2     !    0000000000012.340000000000000000
2     !    \_____._____/   \______._______/
2     !          .                .
2     !        ignore           ignore
2     !
2     !    000.00000000000000000000000000012345
2     !    \_/ \__________.__________/
2     !     .             .
2     !  ignore       do not ignore
2     !
```

```
2         !       1234000000000000000000000000000000.000
2         !            \__________________________/ \_/
2         !                         |                |
2         !                   do not ignore        ignore
2         !-
2
2
2         LOCAL
2                     PUT_BUF         :   REF VECTOR [1100, BYTE],
2                     COUNT           :   INITIAL (0),          ! Count for TEMP_PUT_HERE
2                     DOT_SEEN        :   INITIAL (0),          ! =1 Decimal point seen
2                     DIGIT_SEEN      :   INITIAL (0),          ! =1 At least 1 digit seen
2                     SIGN_SEEN       :   INITIAL (0),          ! =1 Sign seen
2                     E_SEEN          :   INITIAL (0),          ! =1 E/e of exponent seen
2                     R_SIGNIF        :   INITIAL (0),          ! Signigicant digits to
2                                                               ! right of decimal point
2                     L_SIGNIF        :   INITIAL (0),          ! Significant digits to
2                                                               ! left of decimal point
2                     R_ZERO          :   INITIAL (0),          ! Significant zeroes to
2                                                               ! right of decimal point
2                                         ! Calculated after incr loop .00...00123
2                     L_ZERO          :   INITIAL (0) ;         ! Significant zeroes to
2                                                               ! left of decimal point
2                                         ! Calculated in incr loop 1200...0.0
2
2                             PUT_BUF = .TEMP_PUT_HERE [DSC$A_POINTER] ;
2
2                             INCR X FROM 0 TO .CHARS_READ - 1 DO
3                                 BEGIN                                 ! Begin INCR loop
3                                     !+
3                                     ! Count significant digits and significant zeroes.
3                                     !-
3                                     SELECTONE .PUT_BUF [.X] OF
3                                         SET
3
3                                         [ %C'1' TO %C'9' ] :
3                                             !+
3                                             ! Count significant digits to the left
3                                             ! or right of the decimal point.
3                                             !-
3
4                                             BEGIN
4                                             DIGIT_SEEN = 1 ;
4                                             IF .DOT_SEEN
4                                             THEN
4                                                 R_SIGNIF = .R_SIGNIF + 1
4                                             ELSE
4                                                 L_SIGNIF = .L_SIGNIF + 1 ;
4                                             COUNT = .COUNT + 1 ;
3                                             END ;
3
3                                         [ %C'0' ] :
3                                             !+
3                                             ! Count zeroes after DIGIT_SEEN
3                                             ! and/or after DOT_SEEN.
3                                             !-
```

```
4        BEGIN
4        IF .DIGIT_SEEN
4        THEN
4            IF .DOT_SEEN
4            THEN
4                R_SIGNIF = .R_SIGNIF + 1
4            ELSE
4                L_SIGNIF = .L_SIGNIF + 1
4        ELSE
4            !+
4            !   Count zeroes after decimal point,
4            !   but before significant digits
4            !-
4            IF .DOT_SEEN
4            THEN
4                R_ZERO = .R_ZERO + 1 ;
4        COUNT = .COUNT + 1 ;
3        END ;
3
3    [ %C'-', %C'+' ] :
3        !+
3        ! Only one sign is valid
3        !-
3
3        IF .SIGN_SEEN EQL 0
3        THEN
4            BEGIN
4            SIGN_SEEN = 1 ;
4            COUNT = .COUNT + 1 ;
4            END
3        ELSE
3            RETURN 0 ;
3
3    [ %C'.', %C',' ] :
3
4        BEGIN
4        DOT_SEEN = 1 ;
4        COUNT = .COUNT + 1 ;
3        END ;
3
3    [ %C' '] :
3        !+
3        ! Spaces are allowed
3        !-
3            COUNT = .COUNT + 1 ;
3
3    [ %C'E', %C'e' ] :
3
3        !+
3        ! Don't check range of exponent, leave
3        ! that for OTS$CVT_T_F/D
3        !-
4        BEGIN
4        E_SEEN = 1 ;
4        EXITLOOP ;
3        END ;
3
```

```
                        [ OTHERWISE] :

                            RETURN 0 ;

                        TES ;

                    !+
                    ! If maximum significant digits allowed has already
                    ! been reached, or all is left is the exponent
                    ! - pull out of loop.
                    !-

                    IF (.L_SIGNIF + .R_SIGNIF EQL .MAX)  OR
                                            ( .E_SEEN )
                    THEN EXITLOOP ;

                    END ;                               ! End INCR loop

                        !+
                        ! Make sure all remaining digits (if any)
                        ! are zeroes
                        ! Count zeroes after significant digits, but
                        ! before decimal point.
                        !-

                        IF .COUNT LSS .CHARS_READ
                        THEN
                            !+
                            ! All input characters have not yet been processed.
                            !-
                            BEGIN
                            INCR Y FROM .COUNT TO .CHARS_READ - 1 DO
                                BEGIN

                                SELECTONE .PUT_BUF [.Y] OF
                                    SET

                                    [ %C'E', %C'e' ] :

                                        !+
                                        ! Don't check range of exponent,
                                        ! leave that for OTS$CVT_T_F/D
                                        !-
                                        EXITLOOP ;

                                    [ %C'0' ] :
                                        !+
                                        ! Count zeroes to left
                                        ! of decimal point.
                                        !-
                                          IF .DOT_SEEN EQL 0
                                          THEN
                                              L_ZERO = .L_ZERO + 1 ;

                                    [ %C'.', %C',', %C'-', %C'+', %C' ' ] :
                                        !+
                                        ! Of no consequence here
```

```
4                                   !-
4                                   0 ;
4
4                               [ OTHERWISE ]:
4
4                                   RETURN 0 ;
4                               TES ;
4
3                           END ;
3                       END ;
2
2                   !+
2                   !  Check range.  At this point you are only concerned
2                   !  about the  # of zeroes beteen the decimal point and
2                   !  the significant digits.  Anything greater then 38
2                   !  is out of range.  For example the following are out
2                   !  of range -
2                   !      0.000000000000000000000000000000000000000123
2                   !         because R_SIGNIF + R_ZERO GTR 38
2                   !      1230000000000000000000000000000000000000000
2                   !         because L_SIGNIF + L_ZERO GTR 38
2                   !
2                   ! NOTE :  The following error would have been caught
2                   !         by the loop above -
2                   !      120000000.00000000000000000000000000000034
2                   !                 out of range
2                   !-
2
2                   IF .R_SIGNIF + .R_ZERO GTR %X'38' OR
2                      .L_SIGNIF + .L_ZERO GTR %X'38'
2                   THEN
2                       RETURN 0 ;              ! Out of range
2
2           RETURN 1 ;
1           END ;                               ! End COB$$VERIFY_FL_RANGE


                                    OFFC 00000 COB$$VERIFY_FL_RANGE:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 3321
                              52 D4 00002               CLRL    COUNT                                          ; 3361
                              55 7C 00004               CLRQ    SIGN_SEEN                                      ;
                              54 D4 00006               CLRL    E_SEEN                                         ;
                              58 7C 00008               CLRQ    R_SIGNIF                                       ;
                              57 D4 0000A               CLRL    L_SIGNIF                                       ;
                              5A 7C 0000C               CLRQ    L_ZERO                                         ;
          50          04      AC D0 0000E               MOVL    TEMP_PUT_HERE, R0                              ; 3402
          51          04      A0 D0 00012               MOVL    4(R0), PUT_BUF                                 ;
          53                  01 CE 00016               MNEGL   #1, X                                          ; 3503
                              76 11 00019               BRB     16$                                            ;
          50              6341 9A 0001B 1$:             MOVZBL  (X)[PUT_BUF], R0                               ; 3409
          31                  50 91 0001F               CMPB    R0, #49                                        ; 3412
                              0A 1F 00022               BLSSU   2$                                             ;
          39                  50 91 00024               CMPB    R0, #57                                        ;
                              05 1A 00027               BGTRU   2$                                             ;
```

```
                  56          01  D0 00029          MOVL    #1, DIGIT_SEEN                     ; 3419
                              08  11 0002C          BRB     3$                                 ; 3420
                  30          50  91 0002E 2$:      CMPB    R0, #48                            ; 3428
                              15  12 00031          BNEQ    6$                                 ;
                  0B          56  E9 00033          BLBC    DIGIT_SEEN, 5$                     ; 3435
                  04          59  E9 00036 3$:      BLBC    DOT_SEEN, 4$                       ; 3437
                              58  D6 00039          INCL    R_SIGNIF                           ; 3439
                              32  11 0003B          BRB     11$                                ;
                              57  D6 0003D 4$:      INCL    L_SIGNIF                           ; 3441
                              2E  11 0003F          BRB     11$                                ; 3437
                  2B          59  E9 00041 5$:      BLBC    DOT_SEEN, 11$                      ; 3447
                              5B  D6 00044          INCL    R_ZERO                             ; 3449
                              27  11 00046          BRB     11$                                ; 3450
                  2B          50  91 00048 6$:      CMPB    R0, #43                            ; 3453
                              05  13 0004B          BEQL    7$                                 ;
                  2D          50  91 0004D          CMPB    R0, #45                            ;
                              09  12 00050          BNEQ    8$                                 ;
                              55  D5 00052 7$:      TSTL    SIGN_SEEN                          ; 3458
                              27  12 00054          BNEQ    13$                                ;
                  55          01  D0 00056          MOVL    #1, SIGN_SEEN                      ; 3461
                              14  11 00059          BRB     11$                                ; 3462
                  2C          50  91 0005B 8$:      CMPB    R0, #44                            ; 3467
                              05  13 0005E          BEQL    9$                                 ;
                  2E          50  91 00060          CMPB    R0, #46                            ;
                              05  12 00063          BNEQ    10$                                ;
                  59          01  D0 00065 9$:      MOVL    #1, DOT_SEEN                       ; 3470
                              05  11 00068          BRB     11$                                ; 3471
                  20          50  91 0006A 10$:     CMPB    R0, #32                            ; 3474
                              04  12 0006D          BNEQ    12$                                ;
                              52  D6 0006F 11$:     INCL    COUNT                              ; 3478
                              11  11 00071          BRB     15$                                ;
              45  8F          50  91 00073 12$:     CMPB    R0, #69                            ; 3480
                              06  13 00077          BEQL    14$                                ;
              65  8F          50  91 00079          CMPB    R0, #101                           ;
                              68  12 0007D 13$:     BNEQ    22$                                ;
                  54          01  D0 0007F 14$:     MOVL    #1, E_SEEN                         ; 3487
                              12  11 00082          BRB     17$                                ; 3486
      50          57          58  C1 00084 15$:     ADDL3   R_SIGNIF, L_SIGNIF, R0             ; 3503
              0C  AC          50  D1 00088          CMPL    R0, MAX                            ;
                              08  13 0008C          BEQL    17$                                ;
                  05          54  E8 0008E          BLBS    E_SEEN, 17$                        ; 3504
      85          53      08  AC  F2 00091 16$:     AOBLSS  CHARS_READ, X, 1$                  ; 3404
              08  AC          52  D1 00096 17$:     CMPL    COUNT, CHARS_READ                  ; 3516
                              35  18 0009A          BGEQ    21$                                ;
                              52  D7 0009C          DECL    Y                                  ; 3522
                              2C  11 0009E          BRB     20$                                ;
                  50        6241  9A 000A0 18$:     MOVZBL  (Y)[PUT_BUF], R0                   ; 3525
              45  8F          50  91 000A4          CMPB    R0, #69                            ; 3528
                              27  13 000A8          BEQL    21$                                ;
              65  8F          50  91 000AA          CMPB    R0, #101                           ;
                              21  13 000AE          BEQL    21$                                ;
                  30          50  91 000B0          CMPB    R0, #48                            ; 3536
                              08  12 000B3          BNEQ    19$                                ;
                              59  D5 000B5          TSTL    DOT_SEEN                           ; 3541
                              13  12 000B7          BNEQ    20$                                ;
                              5A  D6 000B9          INCL    L_ZERO                             ; 3543
                              0F  11 000BB          BRB     20$                                ; 3541
```

```
                         20           50 91 000BD 19$:    CMPB    R0, #32                            ; 3545
                                      0A 13 000C0         BEQL    20$                                ;
                         2B           50 91 000C2         CMPB    R0, #43                            ;
                                      20 1F 000C5         BLSSU   22$                                ;
                         2E           50 91 000C7         CMPB    R0, #46                            ;
                                      1B 1A 000CA         BGTRU   22$                                ;
              CF         52      08   AC F2 000CC 20$:    AOBLSS  CHARS_READ, Y, 18$                 ; 3522
              50         5B           5B C1 000D1 21$:    ADDL3   R_ZERO, R_SIGNIF, R0               ; 3576
                         38           50 D1 000D5         CMPL    R0, #56                            ;
                                      0D 14 000D8         BGTR    22$                                ;
              50         57           5A C1 000DA         ADDL3   L_ZERO, L_SIGNIF, R0               ; 3577
                         38           50 D1 000DE         CMPL    R0, #56                            ;
                                      04 14 000E1         BGTR    22$                                ;
                         50           01 D0 000E3         MOVL    #1, R0                             ; 3581
                                         04 000E6         RET                                        ;
                                      50 D4 000E7 22$:    CLRL    R0                                 ; 3582
                                         04 000E9         RET                                        ;

; Routine Size:  234 bytes,    Routine Base:  _COB$CODE + 09C0
```

```
%SBTTL 'COB$$SCAN_INPUT - Scan the input string'
ROUTINE COB$$SCAN_INPUT ( ARG_DESC      : REF $STR$DESCRIPTOR,
                          CHARS_READ,
                          FLAGS,
                          BUF_DESC      : REF $STR$DESCRIPTOR,
                          LEFT_DEC,
                          NUM_DIGITS,
                          SIGN_VAL,
                          PUTTER,
                          LEADING_ZEROES,
                          SIGN_SEEN,
                          DIGIT_SEEN,
                          DOT_SEEN,
                          ZERO_SEEN,
                          BLANKS_SEEN
                        )  : =

!++
! FUNCTIONAL DESCRIPTION:
!
!  Scan the input number, put result in BUF.
!
! FORMAL PARAMETERS:
!
!       ARG_DESC.rt.dx          The numeric string to scan
!       CHARS_READ.rlu.v        Number of input characters read
!       FLAGS.rlu.v             Screen enhancement flag
!       BUF_DESC.rt.dx          Deposit for scanned input
!       LEFT_DEC.ml.r           # of digits to left of decimal point
!       NUM_DIGITS.ml.r         # of digits in ARG_DESC
!       SIGN_VAL.ml.r           Temp to hold sign
!       PUTTER.ml.r             Counts position in buffer BUF_DESC
!       LEADING_ZEROES.ml.r     # of leading zeroes
!       SIGN_SEEN.ml.r          = 1 if '+' or '-' scanned
!       DIGIT_SEEN.ml.r         = 1 if a digit 0-9 was scanned
!       DOT_SEEN.ml.r           = 1 if '.' or ',' scanned
!       ZERO_SEEN.ml.r          = 1 if digit 0 scanned
!       BLANKS_SEEN.ml.r        = 1 if trailing blanks scanned
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       0 = Failure, 1 = Success
!
! SIDE EFFECTS:
!
!       NONE
!--
```

```
	BEGIN
	    LOCAL
		BUF : REF VECTOR [1100, BYTE],		! Addresses result
		ARG : REF VECTOR [1100, BYTE] ;		! Addresses source

	BUF = .BUF_DESC [DSC$A_POINTER];
	ARG = .ARG_DESC [DSC$A_POINTER];

!+
! Scan Input, put result in BUF.
!-

	INCR GETTER FROM 0 TO (.CHARS_READ - 1) DO

	    SELECTONE .ARG [.GETTER] OF
		SET

		[%C'0' TO %C'9'] :

		    BEGIN				! Decimal digit
		    IF ( .ARG [.GETTER] NEQ %C'0' ) OR (..DOT_SEEN EQL 1) OR
		       ((.ARG [.GETTER] EQL %C'0' ) AND ..DIGIT_SEEN EQL 1)
		    THEN
			BEGIN
			!+
			! This is not a leading zero
			!-
			IF ..BLANKS_SEEN		! Ensure no imbedded blanks
			THEN RETURN 0;

			.DIGIT_SEEN = 1;
			BUF [..PUTTER] = .ARG [.GETTER];
			.PUTTER = ..PUTTER + 1;

			.NUM_DIGITS = ..NUM_DIGITS + 1 ;
			END
		    ELSE
			BEGIN
			.LEADING_ZEROES = ..LEADING_ZEROES + 1;
			.ZERO_SEEN = 1 ;		! 00. is valid - dot_seen and
							! zero_seen
			END;
		    END;

		[%C'+', %C'-'] :

		    BEGIN				! Plus or minus sign
		    IF ( ..SIGN_SEEN ) THEN RETURN 0 ;
		    IF .GETTER NEQ .CHARS_READ - 1	! Ensure no imbedded signs
		    THEN
			IF (( ..DIGIT_SEEN ) AND ( .ARG [.GETTER + 1] NEQ %C' ' ))
			THEN RETURN 0 ;

		    .SIGN_SEEN = 1;
		    .SIGN_VAL = .ARG [.GETTER];
		    END;
```

```
        [%C'.'] :

            BEGIN                               ! Decimal point
            IF ( ..DOT_SEEN ) THEN RETURN 0;
            !+
            !   Is decimal point a valid character - look at bit 6 of FLAGS
            !-
            IF ( .FLAGS AND V_DEC_PT ) NEQ 0
            THEN
                RETURN 0                        ! Decimal point is illegal
            ELSE
                BEGIN
                .DOT_SEEN = 1;
                .LEFT_DEC = ..NUM_DIGITS ;      ! Count for validating size of
                END ;                           ! entered data.  NUM_DIGITS
            END;                                ! calculated below.

        [%C','] :

            BEGIN                               ! Decimal point is Comma
            IF ( ..DOT_SEEN ) THEN RETURN 0;
            !+
            !   Is comma a valid character - look at bit 6 of FLAGS
            !-
            IF ( .FLAGS AND V_DEC_PT ) NEQ 0
            THEN
                BEGIN                           ! Comma is an illegal character
                .DOT_SEEN = 1;
                .LEFT_DEC = ..NUM_DIGITS ;      ! Count for validating size of
                END                             ! entered data.  NUM_DIGITS
            ELSE                                ! calculated below.
                RETURN 0 ;
            END;

        [%C' '] :

            BEGIN                               ! Blank, better be leading or trailing.
            IF (..SIGN_SEEN OR ..DIGIT_SEEN OR ..DOT_SEEN)
            THEN .BLANKS_SEEN = 1;
            END;

        [OTHERWISE] :           !  reprompt by passing back a routine value of 0

            RETURN 0 ;
        TES;

    RETURN 1 ;
    END ;                                       ! End COB$$SCAN_INPUT


                        003C 00000 COB$$SCAN_INPUT:
                                          .WORD   Save R2,R3,R4,R5                  ; 3584
     50        10    AC  D0 00002         MOVL    BUF_DESC, R0                      ; 3645
     55        04    A0  D0 00006         MOVL    4(R0), BUF                        ;
```

```
                   50       04   AC  D0 0000A          MOVL    ARG_DESC, R0                          ; 3646
                   52       04   A0  D0 0000E          MOVL    4(R0), ARG                            ;
                   54            01  CE 00012          MNEGL   #1, GETTER                            ; 3654
                                 75  11 00015          BRB     8$                                    ;
                   50          6442  9A 00017 1$:      MOVZBL  (GETTER)[ARG], R0                     ;
                   30            50  91 0001B          CMPB    R0, #48                               ; 3657
                                 41  1F 0001E          BLSSU   5$                                    ;
                   39            50  91 00020          CMPB    R0, #57                               ;
                                 3C  1A 00023          BGTRU   5$                                    ;
                   30            50  91 00025          CMPB    R0, #48                               ; 3660
                                 11  12 00028          BNEQ    2$                                    ;
                   01       30   BC  D1 0002A          CMPL    @DOT_SEEN, #1                         ;
                                 0B  13 0002E          BEQL    2$                                    ;
                   30            50  91 00030          CMPB    R0, #48                               ; 3661
                                 23  12 00033          BNEQ    4$                                    ;
                   01       2C   BC  D1 00035          CMPL    @DIGIT_SEEN, #1                       ;
                                 1D  12 00039          BNEQ    4$                                    ;
                   03       38   BC  E9 0003B 2$:      BLBC    @BLANKS_SEEN, 3$                      ; 3667
                               0098  31 0003F          BRW     17$                                   ;
              2C   BC            01  D0 00042 3$:      MOVL    #1, @DIGIT_SEEN                       ; 3670
                   51       20   AC  D0 00046          MOVL    PUTTER, R1                            ; 3671
                   53            61  D0 0004A          MOVL    (R1), R3                              ;
                 6345            50  90 0004D          MOVB    R0, (R3)[BUF]                         ;
                                 61  D6 00051          INCL    (R1)                                  ; 3672
                            18   BC  D6 00053          INCL    @NUM_DIGITS                           ; 3674
                                 74  11 00056          BRB     14$                                   ; 3660
                            24   BC  D6 00058 4$:      INCL    @LEADING_ZEROES                       ; 3678
              34   BC            01  D0 0005B          MOVL    #1, @ZERO_SEEN                        ; 3679
                                 6B  11 0005F          BRB     14$                                   ; 3654
                   2B            50  91 00061 5$:      CMPB    R0, #43                               ; 3684
                                 05  13 00064          BEQL    6$                                    ;
                   2D            50  91 00066          CMPB    R0, #45                               ;
                                 23  12 00069          BNEQ    9$                                    ;
                   6B       28   BC  E8 0006B 6$:      BLBS    @SIGN_SEEN, 17$                       ; 3687
         51   08   AC            01  C3 0006F          SUBL3   #1, CHARS_READ, R1                    ; 3688
                   51            54  D1 00074          CMPL    GETTER, R1                            ;
                                 0B  13 00077          BEQL    7$                                    ;
                   07       2C   BC  E9 00079          BLBC    @DIGIT_SEEN, 7$                       ; 3690
                   20       01 A442  91 0007D          CMPB    1(GETTER)[ARG], #32                   ;
                                 56  12 00082          BNEQ    17$                                   ;
              28   BC            01  D0 00084 7$:      MOVL    #1, @SIGN_SEEN                        ; 3693
              1C   BC            50  D0 00088          MOVL    R0, @SIGN_VAL                         ; 3694
                                 3E  11 0008C 8$:      BRB     14$                                   ; 3654
                   2E            50  91 0008E 9$:      CMPB    R0, #46                               ; 3697
                                 0B  12 00091          BNEQ    10$                                   ;
                   43       30   BC  E8 00093          BLBS    @DOT_SEEN, 17$                        ; 3700
         3E   0C   AC            06  E0 00097          BBS     #6, FLAGS, 17$                        ; 3704
                                 0E  11 0009C          BRB     11$                                   ; 3709
                   2C            50  91 0009E 10$:     CMPB    R0, #44                               ; 3714
                                 14  12 000A1          BNEQ    12$                                   ;
                   33       30   BC  E8 000A3          BLBS    @DOT_SEEN, 17$                        ; 3717
         2E   0C   AC            06  E1 000A7          BBC     #6, FLAGS, 17$                        ; 3721
              30   BC            01  D0 000AC 11$:     MOVL    #1, @DOT_SEEN                         ; 3724
              14   BC       18   BC  D0 000B0          MOVL    @NUM_DIGITS, @LEFT_DEC                ; 3725
                                 15  11 000B5          BRB     14$                                   ; 3721
                   20            50  91 000B7 12$:     CMPB    R0, #32                               ; 3731
                                 1E  12 000BA          BNEQ    17$                                   ;
```

```
                         08        28   BC  E8 000BC          BLBS    @SIGN_SEEN, 13$                    ; 3734
                         04        2C   BC  E8 000C0          BLBS    @DIGIT_SEEN, 13$                   ;
                         04        30   BC  E9 000C4          BLBC    @DOT_SEEN, 14$                     ;
                    38   BC             01  D0 000C8 13$:     MOVL    #1, @BLANKS_SEEN                   ; 3735
              02         54        08   AC  F2 000CC 14$:     AOBLSS  CHARS_READ, GETTER, 15$            ; 3654
                                        03  11 000D1          BRB     16$                                ;
                                      FF41  31 000D3 15$:     BRW     1$                                 ;
                         50             01  D0 000D6 16$:     MOVL    #1, R0                             ; 3743
                                            04 000D9          RET                                        ;
                                        50  D4 000DA 17$:     CLRL    R0                                 ; 3744
                                            04 000DC          RET                                        ;

; Routine Size:  221 bytes,    Routine Base:  _COB$CODE + 0AAA


; 2238          3745  1
; 2239          3746  1      END                                              ! End of module COB$ACCECV
; 2240          3747  0      ELUDOM




;                                PSECT SUMMARY
;
;       Name                          Bytes                         Attributes
;
;  _COB$CODE                           2951  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)



;                          Library Statistics
;
;                                              -------- Symbols --------       Pages       Processing
;       File                                   Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1            9776        27         0         581        00:00.8
;  _$255$DUA28:[COBRTL.OBJ]SMGLIB.L32;1          469         0         0          38        00:00.2




;                                 COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:COBACCECV/OBJ=OBJ$:COBACCECV MSRC$:COBACCECV/UPDATE=(ENH$:COBACCECV
;       )

; Size:          2929 code + 22 data bytes
; Run Time:         00:42.5
; Elapsed Time:     04:56.0
; Lines/CPU Min:      5286
; Lexemes/CPU-Min: 21421
```

```
; Memory Used:  364 pages
; Compilation Complete
```

0060 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

TYPEMAIN
LIS

COBPROLOG
REQ

COBACCDAT
LIS

COBACCDWK
LIS

UNLOCK
LIS

UTILSUBS
LIS

INTPAR
SDL

COBDEF
REQ

COBACCDAY
LIS

COBACCECV
LIS

COBRTL

COBLNK
REQ

COBRTL
MAP

0061 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

COBCVTRPQ
LIS

COBCVTDQ
LIS

COBCANCEL
LIS

COBCVTQP
LIS

COBACCTIM
LIS

COBCVTPQ
LIS

COBCVTRFQ
LIS

COBCVTQF
LIS

COBCVTRQP
LIS

COBCALL
LIS

COBCVTFQ
LIS

COBACCEPT
LIS

COBCVTRDQ
LIS

COBCVTQD
LIS