```
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSSS  RRRRRRRRRRR      TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSSS  RRRRRRRRRRR      TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSSS  RRRRRRRRRRR      TTTTTTTTTTTTTTT  LLL
BBB         BBB  AAA         AAA  SSS              RRR         RRR        TTT        LLL
BBB         BBB  AAA         AAA  SSS              RRR         RRR        TTT        LLL
BBB         BBB  AAA         AAA  SSS              RRR         RRR        TTT        LLL
BBB         BBB  AAA         AAA  SSS              RRR         RRR        TTT        LLL
BBB         BBB  AAA         AAA  SSS              RRR         RRR        TTT        LLL
BBB         BBB  AAA         AAA  SSS              RRR         RRR        TTT        LLL
BBBBBBBBBBBB     AAA         AAA     SSSSSSSSS     RRRRRRRRRRR            TTT        LLL
BBBBBBBBBBBB     AAA         AAA     SSSSSSSSS     RRRRRRRRRRR            TTT        LLL
BBBBBBBBBBBB     AAA         AAA     SSSSSSSSS     RRRRRRRRRRR            TTT        LLL
BBB         BBB  AAAAAAAAAAAAAAA              SSS  RRR   RRR              TTT        LLL
BBB         BBB  AAAAAAAAAAAAAAA              SSS  RRR   RRR              TTT        LLL
BBB         BBB  AAAAAAAAAAAAAAA              SSS  RRR   RRR              TTT        LLL
BBB         BBB  AAA         AAA              SSS  RRR      RRR           TTT        LLL
BBB         BBB  AAA         AAA              SSS  RRR      RRR           TTT        LLL
BBB         BBB  AAA         AAA              SSS  RRR      RRR           TTT        LLL
BBBBBBBBBBBB     AAA         AAA  SSSSSSSSSSSS     RRR         RRR        TTT        LLLLLLLLLLLLLLL
BBBBBBBBBBBB     AAA         AAA  SSSSSSSSSSSS     RRR         RRR        TTT        LLLLLLLLLLLLLLL
BBBBBBBBBBBB     AAA         AAA  SSSSSSSSSSSS     RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

FILEID**BASSARITH

```
BBBBBBBB     AAAAAA      SSSSSSSS   SSSSSSSS   AAAAAA    RRRRRRRR      IIIIII    TTTTTTTTTT HH      HH
BBBBBBBB     AAAAAA      SSSSSSSS   SSSSSSSS   AAAAAA    RRRRRRRR      IIIIII    TTTTTTTTTT HH      HH
BB      BB AA      AA SS         SS         AA      AA RR      RR     II         TT     HH      HH
BB      BB AA      AA SS         SS         AA      AA RR      RR     II         TT     HH      HH
BB      BB AA      AA SS         SS         AA      AA RR      RR     II         TT     HH      HH
BB      BB AA      AA SS         SS         AA      AA RR      RR     II         TT     HH      HH
BBBBBBBB   AA      AA   SSSSSS     SSSSSS   AA      AA RRRRRRRR       II         TT     HHHHHHHHHH
BBBBBBBB   AA      AA   SSSSSS     SSSSSS   AA      AA RRRRRRRR       II         TT     HHHHHHHHHH
BB      BB AAAAAAAAAA         SS         SS AAAAAAAAAA RR  RR         II         TT     HH      HH
BB      BB AAAAAAAAAA         SS         SS AAAAAAAAAA RR  RR         II         TT     HH      HH
BB      BB AA      AA         SS         SS AA      AA RR    RR       II         TT     HH      HH     ....
BB      BB AA      AA         SS         SS AA      AA RR    RR       II         TT     HH      HH     ....
BBBBBBBB   AA      AA SSSSSSSS   SSSSSSSS   AA      AA RR      RR   IIIIII       TT     HH      HH     ....
BBBBBBBB   AA      AA SSSSSSSS   SSSSSSSS   AA      AA RR      RR   IIIIII       TT     HH      HH     ....

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 MODULE BAS$SARITH (                                   ! BASIC String Arithmetic
0               IDENT = '1-024'         ! File: BASSARITH.B32 Edit: MDL1024
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:  BASIC String Arithmetic
1 !
1 ! ABSTRACT:
1 !
1 !       This module contains the BASIC string arithmetic functions.
1 !       They operate by calling the STR$ string arithmetic entry
1 !       points.  These routines are coded for reliability and
1 !       maintainability, not for speed.
1 !
1 ! ENVIRONMENT:  VAX-11 User Mode
1 !
1 ! AUTHOR: John Sauter, CREATION DATE: 02-MAR-1979
1 !
1 ! MODIFIED BY:
1 !
1 ! 1-001 - Original.  JBS 05-MAR-1979
1 ! 1-002 - Don't produce a leading zero if the result is between 0
1 !         and 1.  JBS 07-MAR-1979
1 ! 1-003 - Ask STR$RECIP to work to enough accuracy to be sure that the
1 !         result of the divide will be sufficiently accurate.
1 !         JBS 07-MAR-1979
1 ! 1-004 - Correct the space allocation in PARSE_OUT, so we have enough
1 !         to hold the entire number.  JBS 22-MAR-1979
1 ! 1-005 - Correct the sign of the result of COMP%.  JBS 22-MAR-1979
1 ! 1-006 - Improve comments based on the code review.  JBS 26-MAR-1979
1 ! 1-007 - Correct a typo introduced in edit 006.  JBS 28-MAR-1979
1 ! 1-008 - If we get an error, free local strings.  JBS 07-MAY-1979
```

```
1 ! 1-009 - Call the STR$ facility with input scalars by reference.
1 !           JBS 15-MAY-1979
1 ! 1-010 - Change OTS$S and LIB$S to STR$.  JBS 21-MAY-1979
1 ! 1-011 - Change call to STR$COPY.  JBS 16-JUL-1979
1 ! 1-012 - In BAS$QUO, search for an exact quotient.  JBS 23-JUL-1979
1 ! 1-013 - When freeing local strings, watch out for descriptors
1 !           which have not yet been initialized.  JBS 31-JUL-1979
1 ! 1-014 - When scanning the input string, allocate at least 8 bytes of
1 !           space.  QAR N11-02925.  JBS 17-OCT-1979
1 ! 1-015 - Try to speed up BAS$QUO by shortening the quotient search.
1 !           Leave in monitoring code under conditional compilation in case
1 !           it is needed again.  JBS 22-MAY-1980
1 ! 1-016 - 3.75/3 to one decimal place gives 1.2 instead of 1.3.  To fix this,
1 !           round before hill climbing.  JBS 13-JUN-1980
1 ! 1-017 - Added code to routines BAS$PLACE and BAS$PROD to take care of
1 !         the incompatibilities between VAX and the PDP-11 in handling of
1 !         certain ROUND/TRUNCATION parameters (specifically rounding values
1 !         in the range of -1 to -4 and truncation values in the range of
1 !         9996 to 9999).  LEB  24-AUG-81
1 ! 1-018 - Changed routine BAS$QUO to now call routine STR$DIVIDE instead of
1 !         the calls to STR$MUL, STR$RECIP, and STR$ROUND.  LEB  8-OCT-81
1 ! 1-019 - Added code to BAS$QUO to handle compatibility problems with
1 !         regard to getting rid of unnecessary leading or trailing
1 !         zeroes.  LEB 23-DEC-81.
1 ! 1-020 - Changed code in BAS$PLACE and BAS$PROD to handle the ROUND
1 !         TRUNCATION parameters - they now correctly handle values
1 !         in the range of -4999 to -1 and in the range of 5001 to 9999.
1 !         LEB 4-FEB-82.
1 ! 1-021 - Added line to BAS$QUO to store updated length field into the
1 !         descriptor.  LEB 2-MAR-82.
1 ! 1-022 - Added code to PARSE_IN to allow for an input of -0, to fix
1 !         a compatibility mode bug.  LEB  29-JUN-1982.
1 ! 1-023 - additional change to PARSE_IN to accept +0, 0. as well.
1 !         MDL 10-Jan-1983
1 ! 1-024 - restore DSC$W_LENGTH before calling STR$FREE1_DX in STR$DIVIDE
1 !         to avoid faking out the STR$ code and cause an access violation.
1 !         MDL 15-Mar-1983
1 !--
1
1 !<BLF/PAGE>
```

```
!
! SWITCHES:
!

SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);

!
! LINKAGES:
!
!       NONE
!
! TABLE OF CONTENTS:
!

FORWARD ROUTINE
    BAS$COMP,                                   ! Compare strings
    BAS$DIF : NOVALUE,                          ! Difference between strings
    BAS$PLACE : NOVALUE,                        ! Shorten a string
    BAS$PROD : NOVALUE,                         ! Multiply two strings
    BAS$QUO : NOVALUE,                          ! Divide two strings
    BAS$SUM : NOVALUE,                          ! Sum of two strings
    PARSE_IN : NOVALUE,                         ! Convert to internal form
    PARSE_OUT : NOVALUE,                        ! Convert to external form
    FREE_STRINGS;                               ! Handler to free strings

!
! INCLUDE FILES:
!

REQUIRE 'RTLIN:RTLPSECT';                       ! Macros for defining psects

LIBRARY 'RTLSTARLE';                            ! System definitions

!
! MACROS:
!
!       NONE
!
! EQUATED SYMBOLS:
!

LITERAL
!+
! Several of the string arithmetic routines encode the precision and
! a truncation flag in a single parameter.  This is done as follows:
! (P% is the precision parameter)
!
!     1. If P% is less than or equal to 5000, rounding occurs; if
!        more, truncation.
!
!     2. If P% is less than or equal to zero, rounding occurs on the
!        left side of the decimal point.
!
!     3. If P% is between 0 and 5000, rounding occurs on the right
!        side of the decimal point.
!
!     4. If P% is between 5000 and 10,000, truncation occurs on the
```

```
!       left side of the decimal point.
!
!    5. If P% is greater than 10,000, truncation occurs on the right
!       side of the decimal point.
!
! To help the reader recognize where these semantics are implemented, the
! following symbol is used for 5000 (and 2* the symbol for 10,000) in the
! above contexts.
!-
    BAS$K_PREC_VAL = 5000,
!+
! There is a limit imposed by the PDP-11 implementation of BASIC-PLUS-2
! on the number of digits which can be processed.  This limit appears
! in two places: the precision of QUO$ is limited to 1E-55 (or 1E-56 if
! truncating) and number of digits which can be input is limited to 60.
! To help the reader recognize where these limits are implemented, the
! following constants are defined:
!-
    BAS$K_PREC_LIM1 = 55,                          ! Max digits in QUO$ when rounding
    BAS$K_PREC_LIM2 = 56,                          ! Max digits in QUO$ when truncating
    BAS$K_PREC_LIM3 = 60;                          ! Max digits when scanning input

!
!
! PSECTS:
!
DECLARE_PSECTS (BAS);                              ! Declare psects for BAS$ facility
!
! OWN STORAGE:
!
!       NONE
!
! EXTERNAL REFERENCES:
!

EXTERNAL ROUTINE
    BAS$$STOP : NOVALUE,                           ! Signals a fatal BASIC error
    BAS$$SIGNAL : NOVALUE,                         ! Signals a BASIC error
    STR$GET1_DX,                                   ! Allocate a string
    STR$FREE1_DX,                                  ! Deallocate a string
    STR$COPY_R,                                    ! Copy a string by ref
    LIB$MATCH_COND,                                ! Match condition codes
    STR$ADD : NOVALUE,                             ! Add two strings
    STR$MUL : NOVALUE,                             ! Multiply two strings
    STR$RECIP : NOVALUE,                           ! Take the reciprocal of a string
    STR$ROUND : NOVALUE,                           ! Rounds a string
    STR$DIVIDE: NOVALUE;                           ! Divide two strings

!+
! The following are the error codes used in this module:
!-

EXTERNAL LITERAL
    BAS$K_DATFORERR : UNSIGNED (8),                ! Data format error
    BAS$K_ILLNUM : UNSIGNED (8),                   ! Illegal number
    BAS$K_FLOPOIERR : UNSIGNED (8),                ! Floating point error
    BAS$K_DIVBY_ZER : UNSIGNED (8);                ! Division by zero
```

```
BIND
    TABLE_Z = UPLIT BYTE (REP 49 OF (%X'00'),REP 9 OF (%X'01'),REP 198 OF (%X'00')),
    TABLE_NZ = UPLIT BYTE (REP 49 OF (%X'01'),REP 207 OF (%X'00')),
    MASK = UPLIT BYTE (%X'01');

BUILTIN
    SPANC;                                          !\Skip over a specified set of
                                                    !/characters in a string
```

```
1 GLOBAL ROUTINE BAS$COMP (                              ! Compare strings
1         ARG1,                                         ! First operand
1         ARG2                                          ! Second operand
1     ) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Compare two strings.  This is done by subtracting them and
1 !       testing the result for zero.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       ARG1.rt.dx        First number to compare, as +nnn.nnn
1 !       ARG2.rt.dx        Number to compare against, as +nnn.nnn
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       0 = ARGs are equal,
1 !       -1 = ARG1 less than ARG2,
1 !       1 = ARG1 greater than ARG2.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Signals PROLOSSOR (for "impossible" conditions) and the
1 !       signals from PARSE_IN.
1 !
1 !--
1
2     BEGIN
2
2     MAP
2         ARG1 : REF BLOCK [8, BYTE],
2         ARG2 : REF BLOCK [8, BYTE];
2
2     LOCAL
2 !+
2 ! Internal form of ARG1
2 !-
2         A_DESC : BLOCK [8, BYTE] VOLATILE,
2         A_SIGN,
2         A_EXP,
2 !+
2 ! Internal form of ARG2
2 !-
2         B_DESC : BLOCK [8, BYTE] VOLATILE,
2         B_SIGN,
2         B_EXP,
```

```
2 !+
2 ! Temp to hold A-B.
2 !-
2         C_DESC : BLOCK [8, BYTE] VOLATILE,
2         C_SIGN,
2         C_EXP,
2 !+
2 ! Temp to hold result of comparison.
2 !-
2         RESULT;
2
2 !+
2 ! Enable a handler which will free the strings
2 !-
2
2     ENABLE
2         FREE_STRINGS (A_DESC, B_DESC, C_DESC);
2
2 !+
2 ! Convert the two arguments from external form to internal form.
2 ! This is done by removing the non-digits from the string and
2 ! returning the sign and exponent as separate values.
2 ! Errors are signaled.
2 !-
2     A_DESC [DSC$W_LENGTH] = 0;
2     A_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2     A_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2     A_DESC [DSC$A_POINTER] = 0;
2     PARSE_IN (.ARG1, A_DESC, A_SIGN, A_EXP);
2     B_DESC [DSC$W_LENGTH] = 0;
2     B_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2     B_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2     B_DESC [DSC$A_POINTER] = 0;
2     PARSE_IN (.ARG2, B_DESC, B_SIGN, B_EXP);
2 !+
2 ! If the signs differ we can compute the result based on them.
2 ! Otherwise we must subtract the inputs.
2 !-
2     C_DESC [DSC$W_LENGTH] = 0;
2     C_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2     C_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2     C_DESC [DSC$A_POINTER] = 0;
2
3     IF (.A_SIGN NEQ .B_SIGN)
2     THEN
2         C_SIGN = .B_SIGN
2     ELSE
3         BEGIN
3         A_SIGN = 1 - .A_SIGN;                   ! Change the sign of A
3         STR$ADD (A_SIGN, A_EXP, A_DESC, B_SIGN, B_EXP, B_DESC, C_SIGN, C_EXP, C_DESC);
2         END;
2
2 !+
2 ! We are done with A and B, so free them.
2 !-
2     STR$FREE1_DX (A_DESC);
2     STR$FREE1_DX (B_DESC);
```

```
;  337        0429  2  !+
;  338        0430  2  ! Set RESULT based on the sign of the difference between A and B.
;  339        0431  2  !-
;  340        0432  2     RESULT = (IF (.C_SIGN) THEN 1 ELSE -1);
;  341        0433  2  !+
;  342        0434  2  ! If the result is a one-character string whose digit is zero,
;  343        0435  2  ! the arguments were equal.  Otherwise they were not.
;  344        0436  2  !-
;  345        0437  2
;  346        0438  3     IF (.C_DESC [DSC$W_LENGTH] EQLU 1)
;  347        0439  2     THEN
;  348        0440  2
;  349        0441  2         IF (CH$RCHAR (.C_DESC [DSC$A_POINTER]) EQL %C'0') THEN RESULT = 0;
;  350        0442  2
;  351        0443  2  !+
;  352        0444  2  ! We can now free C
;  353        0445  2  !-
;  354        0446  2     STR$FREE1_DX (C_DESC);
;  355        0447  2     RETURN (.RESULT);
;  356        0448  1     END;                                            ! end of BAS$COMP

                                                        .TITLE  BAS$SARITH
                                                        .IDENT  \1-024\

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

                                  00# 00000 P.AAA:      .BYTE   0[49]                                        ;
                                  01# 00031             .BYTE   1[9]                                         ;
                                  00# 0003A             .BYTE   0[198]                                       ;
                                  01# 00100 P.AAB:      .BYTE   1[49]                                        ;
                                  00# 00131             .BYTE   0[207]                                       ;
                                  01  00200 P.AAC:      .BYTE   1                                            ;

                                            TABLE_Z=            P.AAA
                                            TABLE_NZ=           P.AAB
                                            MASK=               P.AAC
                                                        .EXTRN  BAS$$STOP, BAS$$SIGNAL
                                                        .EXTRN  STR$GET1_DX, STR$FREE1_DX
                                                        .EXTRN  STR$COPY_R, LIB$MATCH_COND
                                                        .EXTRN  STR$ADD, STR$MUL
                                                        .EXTRN  STR$RECIP, STR$ROUND
                                                        .EXTRN  STR$DIVIDE, BAS$K_DATFORERR
                                                        .EXTRN  BAS$K_ILLNUM, BAS$K_FLOPOIERR
                                                        .EXTRN  BAS$K_DIVBY_ZER

                              000C 00000                .ENTRY  BAS$COMP, Save R2,R3                 ; 0315
             53 00000000G   00   9E 00002               MOVAB   STR$FREE1_DX, R3                     ;
             5E          30   C2 00009                  SUBL2   #48, SP                              ;
                     18   AE   7C 0000C                 CLRQ    C_DESC                               ; 0353
                     20   AE   7C 0000F                 CLRQ    B_DESC                               ;
                     28   AE   7C 00012                 CLRQ    A_DESC                               ;
             6D     00B3   CF   DE 00015                MOVAL   6$, (FP)                             ;
                     28   AE   B4 0001A                 CLRW    A_DESC                               ; 0396
         2A   AE          0F   90 0001D                 MOVB    #15, A_DESC+2                        ; 0397
         2B   AE          02   90 00021                 MOVB    #2, A_DESC+3                         ; 0398
                     2C   AE   D4 00025                 CLRL    A_DESC+4                             ; 0399
```

```
                                10  AE  9F 00028        PUSHAB  A_EXP                          ; 0400
                                18  AE  9F 0002B        PUSHAB  A_SIGN                         ;
                                30  AE  9F 0002E        PUSHAB  A_DESC                         ;
                                04  AC  DD 00031        PUSHL   ARG1                           ;
              0000V  CF             04  FB 00034        CALLS   #4, PARSE_IN                   ;
                                20  AE  B4 00039        CLRW    B_DESC                         ; 0401
                 22  AE             0F  90 0003C        MOVB    #15, B_DESC+2                  ; 0402
                 23  AE             02  90 00040        MOVB    #2, B_DESC+3                   ; 0403
                                24  AE  D4 00044        CLRL    B_DESC+4                       ; 0404
                                08  AE  9F 00047        PUSHAB  B_EXP                          ; 0405
                                10  AE  9F 0004A        PUSHAB  B_SIGN                         ;
                                28  AE  9F 0004D        PUSHAB  B_DESC                         ;
                                08  AC  DD 00050        PUSHL   ARG2                           ;
              0000V  CF             04  FB 00053        CALLS   #4, PARSE_IN                   ;
                                18  AE  B4 00058        CLRW    C_DESC                         ; 0410
                 1A  AE             0F  90 0005B        MOVB    #15, C_DESC+2                  ; 0411
                 1B  AE             02  90 0005F        MOVB    #2, C_DESC+3                   ; 0412
                                1C  AE  D4 00063        CLRL    C_DESC+4                       ; 0413
                 0C  AE         14  AE  D1 00066        CMPL    A_SIGN, B_SIGN                 ; 0415
                                    07  13 0006B        BEQL    1$                             ;
                 04  AE         0C  AE  D0 0006D        MOVL    B_SIGN, C_SIGN                 ; 0417
                                    28  11 00072        BRB     2$                             ;
 14    AE            01         14  AE  C3 00074 1$:    SUBL3   A_SIGN, #1, A_SIGN             ; 0420
                                18  AE  9F 0007A        PUSHAB  C_DESC                         ; 0421
                                04  AE  9F 0007D        PUSHAB  C_EXP                          ;
                                0C  AE  9F 00080        PUSHAB  C_SIGN                         ;
                                2C  AE  9F 00083        PUSHAB  B_DESC                         ;
                                18  AE  9F 00086        PUSHAB  B_EXP                          ;
                                20  AE  9F 00089        PUSHAB  B_SIGN                         ;
                                40  AE  9F 0008C        PUSHAB  A_DESC                         ;
                                2C  AE  9F 0008F        PUSHAB  A_EXP                          ;
                                34  AE  9F 00092        PUSHAB  A_SIGN                         ;
          00000000G  00             09  FB 00095        CALLS   #9, STR$ADD                    ;
                                28  AE  9F 0009C 2$:    PUSHAB  A_DESC                         ; 0427
                     63             01  FB 0009F        CALLS   #1, STR$FREE1_DX               ;
                                20  AE  9F 000A2        PUSHAB  B_DESC                         ; 0428
                     63             01  FB 000A5        CALLS   #1, STR$FREE1_DX               ;
                     05         04  AE  E9 000A8        BLBC    C_SIGN, 3$                     ; 0432
                     52             01  D0 000AC        MOVL    #1, RESULT                     ;
                                    03  11 000AF        BRB     4$                             ;
                     52             01  CE 000B1 3$:    MNEGL   #1, RESULT                     ;
                     01         18  AE  B1 000B4 4$:    CMPW    C_DESC, #1                     ; 0438
                                    08  12 000B8        BNEQ    5$                             ;
                     30         1C  BE  91 000BA        CMPB    @C_DESC+4, #48                 ; 0441
                                    02  12 000BE        BNEQ    5$                             ;
                                    52  D4 000C0        CLRL    RESULT                         ;
                                18  AE  9F 000C2 5$:    PUSHAB  C_DESC                         ; 0446
                     63             01  FB 000C5        CALLS   #1, STR$FREE1_DX               ;
                     50             52  D0 000C8        MOVL    RESULT, R0                     ; 0447
                                    04     000CB        RET                                    ; 0448
                                  0000     000CC 6$:    .WORD   Save nothing                   ; 0353
                     50         08  AC  D0 000CE        MOVL    8(AP), R0                      ;
                     50         04  A0  D0 000D2        MOVL    4(R0), R0                      ;
                                E8  A0  9F 000D6        PUSHAB  C_DESC                         ;
                                F0  A0  9F 000D9        PUSHAB  B_DESC                         ;
                                F8  A0  9F 000DC        PUSHAB  A_DESC                         ;
                                    03  DD 000DF        PUSHL   #3                             ;
```

```
                              5E  DD 000E1          PUSHL   SP                          ;
         7E          04       AC  7D 000E3          MOVQ    4(AP), -(SP)                ;
  0000V  CF                   03  FB 000E7          CALLS   #3, FREE_STRINGS            ;
                              04  000EC             RET                                 ;

; Routine Size:  237 bytes,    Routine Base:  _BAS$CODE + 0201

;  357          0449  1
```

```
GLOBAL ROUTINE BAS$DIF (                        ! Subtract strings
        DIF_DESC,                               ! Difference
        OP1_DESC,                               ! First input
        OP2_DESC                                ! Second input
    ) : NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Subtract two strings.  DIF := OP1 - OP2
!
! FORMAL PARAMETERS:
!
!       DIF_DESC.wt.dx  The difference, OP1 - OP2
!       OP1_DESC.rt.dx  Operand OP1
!       OP2_DESC.rt.dx  Operand OP2
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       NONE
!
! SIDE EFFECTS:
!
!       Signals PROLOSSOR for certain "impossible" errors, and the signals
!       from PARSE_IN and PARSE_OUT.
!
!--

    BEGIN

    MAP
        OP1_DESC : REF BLOCK [8, BYTE],
        OP2_DESC : REF BLOCK [8, BYTE],
        DIF_DESC : REF BLOCK [8, BYTE];

    LOCAL
!+
! Internal form of OP1.
!-
        A_DESC : BLOCK [8, BYTE] VOLATILE,
        A_SIGN,
        A_EXP,
!+
! Internal form of OP2.
!-
        B_DESC : BLOCK [8, BYTE] VOLATILE,
        B_SIGN,
        B_EXP,
```

```
2 !+
2 ! Internal form of the difference, OP1 - OP2.
2 !-
2         C_DESC : BLOCK [8, BYTE] VOLATILE,
2         C_SIGN,
2         C_EXP;
2
2 !+
2 ! Enable a handler to free the strings.
2 !-
2
2     ENABLE
2         FREE_STRINGS (A_DESC, B_DESC, C_DESC);
2
2 !+
2 ! Convert the two input arguments from external form to internal form.
2 ! This is done by removing the non-digits from the string and
2 ! returning the sign and exponent as separate values.
2 ! Signal any errors.
2 !-
2     A_DESC [DSC$W_LENGTH] = 0;
2     A_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2     A_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2     A_DESC [DSC$A_POINTER] = 0;
2     PARSE_IN (.OP1_DESC, A_DESC, A_SIGN, A_EXP);
2     B_DESC [DSC$W_LENGTH] = 0;
2     B_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2     B_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2     B_DESC [DSC$A_POINTER] = 0;
2     PARSE_IN (.OP2_DESC, B_DESC, B_SIGN, B_EXP);
2 !+
2 ! Subtract the numbers using the large-precision string arithmetic
2 ! package.
2 !-
2     C_DESC [DSC$W_LENGTH] = 0;
2     C_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2     C_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2     C_DESC [DSC$A_POINTER] = 0;
2     B_SIGN = 1 - .B_SIGN;                          ! Change the sign of B.
2     STR$ADD (A_SIGN, A_EXP, A_DESC, B_SIGN, B_EXP, B_DESC, C_SIGN, C_EXP, C_DESC);
2 !+
2 ! We are done with A and B, so free them.
2 !-
2     STR$FREE1_DX (A_DESC);
2     STR$FREE1_DX (B_DESC);
2 !+
2 ! Convert the difference to external form for the caller.
2 !-
2     PARSE_OUT (C_DESC, .C_SIGN, .C_EXP, .DIF_DESC);
2 !+
2 ! We can now free C
2 !-
2     STR$FREE1_DX (C_DESC);
2     RETURN;
1     END;                                           ! end of BAS$DIF
```

```
                          0004 00000        .ENTRY  BAS$DIF, Save R2                   ; 0450
            52 00000000G    00  9E 00002        MOVAB   STR$FREE1_DX, R2                   ;
            5E                  30  C2 00009        SUBL2   #48, SP                            ;
                        18  AE  7C 0000C        CLRQ    C_DESC                             ; 0487
                        20  AE  7C 0000F        CLRQ    B_DESC                             ;
                        28  AE  7C 00012        CLRQ    A_DESC                             ;
            6D        0099  CF  DE 00015        MOVAL   1$, (FP)                           ;
                        28  AE  B4 0001A        CLRW    A_DESC                             ; 0527
        2A  AE              0F  90 0001D        MOVB    #15, A_DESC+2                      ; 0528
        2B  AE              02  90 00021        MOVB    #2, A_DESC+3                       ; 0529
                        2C  AE  D4 00025        CLRL    A_DESC+4                           ; 0530
                        10  AE  9F 00028        PUSHAB  A_EXP                              ; 0531
                        18  AE  9F 0002B        PUSHAB  A_SIGN                             ;
                        30  AE  9F 0002E        PUSHAB  A_DESC                             ;
                        08  AC  DD 00031        PUSHL   OP1_DESC                           ;
     0000V  CF              04  FB 00034        CALLS   #4, PARSE_IN                       ;
                        20  AE  B4 00039        CLRW    B_DESC                             ; 0532
        22  AE              0F  90 0003C        MOVB    #15, B_DESC+2                      ; 0533
        23  AE              02  90 00040        MOVB    #2, B_DESC+3                       ; 0534
                        24  AE  D4 00044        CLRL    B_DESC+4                           ; 0535
                        08  AE  9F 00047        PUSHAB  B_EXP                              ; 0536
                        10  AE  9F 0004A        PUSHAB  B_SIGN                             ;
                        28  AE  9F 0004D        PUSHAB  B_DESC                             ;
                        0C  AC  DD 00050        PUSHL   OP2_DESC                           ;
     0000V  CF              04  FB 00053        CALLS   #4, PARSE_IN                       ;
                        18  AE  B4 00058        CLRW    C_DESC                             ; 0541
        1A  AE              0F  90 0005B        MOVB    #15, C_DESC+2                      ; 0542
        1B  AE              02  90 0005F        MOVB    #2, C_DESC+3                       ; 0543
                        1C  AE  D4 00063        CLRL    C_DESC+4                           ; 0544
0C   AE             01  0C  AE  C3 00066        SUBL3   B_SIGN, #1, B_SIGN                 ; 0545
                        18  AE  9F 0006C        PUSHAB  C_DESC                             ; 0546
                        04  AE  9F 0006F        PUSHAB  C_EXP                              ;
                        0C  AE  9F 00072        PUSHAB  C_SIGN                             ;
                        2C  AE  9F 00075        PUSHAB  B_DESC                             ;
                        18  AE  9F 00078        PUSHAB  B_EXP                              ;
                        20  AE  9F 0007B        PUSHAB  B_SIGN                             ;
                        40  AE  9F 0007E        PUSHAB  A_DESC                             ;
                        2C  AE  9F 00081        PUSHAB  A_EXP                              ;
                        34  AE  9F 00084        PUSHAB  A_SIGN                             ;
  00000000G  00             09  FB 00087        CALLS   #9, STR$ADD                        ;
                        28  AE  9F 0008E        PUSHAB  A_DESC                             ; 0550
             62             01  FB 00091        CALLS   #1, STR$FREE1_DX                   ;
                        20  AE  9F 00094        PUSHAB  B_DESC                             ; 0551
             62             01  FB 00097        CALLS   #1, STR$FREE1_DX                   ;
                        04  AC  DD 0009A        PUSHL   DIF_DESC                           ; 0555
                        04  AE  DD 0009D        PUSHL   C_EXP                              ;
                        0C  AE  DD 000A0        PUSHL   C_SIGN                             ;
                        24  AE  9F 000A3        PUSHAB  C_DESC                             ;
     0000V  CF              04  FB 000A6        CALLS   #4, PARSE_OUT                      ;
                        18  AE  9F 000AB        PUSHAB  C_DESC                             ; 0559
             62             01  FB 000AE        CALLS   #1, STR$FREE1_DX                   ;
                                04 000B1        RET                                        ; 0561
                              0000 000B2 1$:    .WORD   Save nothing                       ; 0487
             50         08  AC  D0 000B4        MOVL    8(AP), R0                          ;
```

```
              50        04   A0  D0 000B8          MOVL    4(R0), R0                    ;
                        E8   A0  9F 000BC          PUSHAB  C_DESC                       ;
                        F0   A0  9F 000BF          PUSHAB  B_DESC                       ;
                        F8   A0  9F 000C2          PUSHAB  A_DESC                       ;
                             03  DD 000C5          PUSHL   #3                           ;
                             5E  DD 000C7          PUSHL   SP                           ;
              7E        04   AC  7D 000C9          MOVQ    4(AP), -(SP)                 ;
         0000V  CF           03  FB 000CD          CALLS   #3, FREE_STRINGS             ;
                                 04 000D2          RET                                  ;

; Routine Size:  211 bytes,    Routine Base:  _BAS$CODE + 02EE

;  471          0562  1
```

```
1 GLOBAL ROUTINE BAS$PLACE (                               ! Round or truncate a string
1         RESULT_DESC,                                     ! Result
1         OP1_DESC,                                        ! Number to be rounded/truncated
1         PRECISION                                        ! Digits of result
1     ) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Round or truncate a string to a particular position after
1 !       (or before) the decimal point.  RESULT = round (OP1)
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       RESULT_DESC.wt.dx The result of rounding/truncating OP1
1 !       OP1_DESC.rt.dx  Operand OP1
1 !       PRECISION.rl.v  Number of digits to retain and, encoded,
1 !                       the round/truncate flag.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Signals PROLOSSOR for certain "impossible" errors, and the signals
1 !       from PARSE_IN and PARSE_OUT.
1 !
1 !--
1
2     BEGIN
2
2     MAP
2         OP1_DESC : REF BLOCK [8, BYTE],
2         RESULT_DESC : REF BLOCK [8, BYTE];
2
2     LOCAL
2 !+
2 ! Internal form of OP1.
2 !-
2         A_DESC : BLOCK [8, BYTE] VOLATILE,
2         A_SIGN,
2         A_EXP,
2         TRUNC_FLAG,                                      ! 1 = truncate (rather than round)
2         ROUND_POS,                                       ! Where to round
2 !+
2 ! The following is a descriptor for the constant 1.  It is multiplied by
2 ! a power of 10 for rounding or truncating.
```

```
2 !-
2         ONE_DESC : BLOCK [8, BYTE],
2         ONE_BUF : VECTOR [1, BYTE];
2
2 !+
2 ! Enable a handler to free A_DESC in case of error.
2 !-
2
2     ENABLE
2         FREE_STRINGS (A_DESC);
2
2 !+
2 ! Convert the input argument from external form to internal form.
2 ! This is done by removing the non-digits from the string and
2 ! returning the sign and exponent as separate values.
2 ! Errors are signaled.
2 !-
2     A_DESC [DSC$W_LENGTH] = 0;
2     A_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2     A_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2     A_DESC [DSC$A_POINTER] = 0;
2     PARSE_IN (.OPT_DESC, A_DESC, A_SIGN, A_EXP);
2 !+
2 ! Round or truncate the result to a specified decimal place.
2 !-
2
3     IF (.PRECISION LEQ BAS$K_PREC_VAL)
3     THEN
3         BEGIN
3         TRUNC_FLAG = 0;
3         ROUND_POS = -.PRECISION;
3         END
2     ELSE
3         BEGIN
3         TRUNC_FLAG = 1;
3         ROUND_POS = -(.PRECISION - (BAS$K_PREC_VAL*2));
2         END;
2
2 !+
2 ! In order to do the BASIC rounding/truncation based on decimal place
2 ! using the string package's rounding/truncation based on number of
2 ! significant digits, we must be sure that the position we are rounding
2 ! to is part of the significance.  Therefore we add 1E<ROUND_POS>,
2 ! round or truncate to discard all lower digits, and then subtract
2 ! 1E<ROUND_POS> to get back to where we belong.
2 !-
2     ONE_DESC [DSC$W_LENGTH] = 1;
2     ONE_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2     ONE_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
2     ONE_DESC [DSC$A_POINTER] = ONE_BUF [0];
2     ONE_BUF [0] = %C'1';
2     STR$ADD (A_SIGN, A_EXP, A_DESC,                    !
2         A_SIGN, ROUND_POS, ONE_DESC,                   !
2         A_SIGN, A_EXP, A_DESC);
2     STR$ROUND (%REF (.A_EXP + .A_DESC [DSC$W_LENGTH] - .ROUND_POS), TRUNC_FLAG,        !
2         A_SIGN, A_EXP, A_DESC,                         !
2         A_SIGN, A_EXP, A_DESC);
```

```
;  587      0677  2      STR$ADD (A_SIGN, A_EXP, A_DESC,                          !
;  588      0678  2          %REF ((IF (.A_SIGN) THEN 0 ELSE 1)), ROUND_POS, ONE_DESC,        !
;  589      0679  2          A_SIGN, A_EXP, A_DESC);
;  590      0680  2
;  591      0681  2  !+
;  592      0682  2  ! Check if the precision value is in the range of -1 to -4 (indicating
;  593      0683  2  ! rounding) or if the value is in the range of 9996 to 9999 (indicating
;  594      0684  2  ! truncation).  In either case, there is an incompatibility to be fixed
;  595      0685  2  ! here, so that VAX BASIC returns the same value as BASIC+2.
;  596      0686  2  !-
;  597      0687  2
;  598      0688  2      SELECTONE .PRECISION OF
;  599      0689  2        SET
;  600      0690  2          [-4999 TO -1]:                          ! Rounding parameter
;  601      0691  3            BEGIN
;  602      0692  3            IF .A_EXP NEQ 0
;  603      0693  3              THEN
;  604      0694  3                A_EXP = .A_EXP - ABS(.PRECISION);
;  605      0695  2            END;
;  606      0696  2          [5001 TO 9999]:                         ! Truncation parameter
;  607      0697  3            BEGIN
;  608      0698  3              LOCAL
;  609      0699  3                TEMP;
;  610      0700  3            TEMP = 10000 - .PRECISION;
;  611      0701  3            IF .A_EXP NEQ 0
;  612      0702  3              THEN
;  613      0703  3                A_EXP = .A_EXP - ABS(.TEMP);
;  614      0704  2            END;
;  615      0705  2        TES;
;  616      0706  2
;  617      0707  2  !+
;  618      0708  2  ! Convert the product to external form for the caller.
;  619      0709  2  !-
;  620      0710  2      PARSE_OUT (A_DESC, .A_SIGN, .A_EXP, .RESULT_DESC);
;  621      0711  2  !+
;  622      0712  2  ! We can now free A
;  623      0713  2  !-
;  624      0714  2      STR$FREE1_DX (A_DESC);
;  625      0715  2      RETURN;
;  626      0716  1      END;                                        ! end of BAS$PLACES


                              000C 00000          .ENTRY  BAS$PLACE, Save R2,R3          ; 0563
           53 00000000G    00  9E 00002          MOVAB   STR$ADD, R3                    ;
           5E              28  C2 00009          SUBL2   #40, SP                        ;
                  20   AE  7C 0000C          CLRQ    A_DESC                         ; 0602
           6D      0129   CF  DE 0000F          MOVAL   95, (FP)                       ;
                  20   AE  B4 00014          CLRW    A_DESC                         ; 0637
        22    AE          0F  90 00017          MOVB    #15, A_DESC+2                  ; 0638
        23    AE          02  90 0001B          MOVB    #2, A_DESC+3                   ; 0639
                  24   AE  D4 0001F          CLRL    A_DESC+4                       ; 0640
                  10   AE  9F 00022          PUSHAB  A_EXP                          ; 0641
                  18   AE  9F 00025          PUSHAB  A_SIGN                         ;
                  28   AE  9F 00028          PUSHAB  A_DESC                         ;
```

```
                                   08  AC  DD 0002B         PUSHL   OP1_DESC                          ;
              0000V  CF                04  FB 0002E         CALLS   #4, PARSE_IN                      ;
                     52            0C  AC  D0 00033         MOVL    PRECISION, R2                     ; 0646
           00001388  8F                52  D1 00037         CMPL    R2, #5000                         ;
                                       09  14 0003E         BGTR    1$                                ;
                                   08  AE  D4 00040         CLRL    TRUNC_FLAG                        ; 0649
                 0C  AE                52  CE 00043         MNEGL   R2, ROUND_POS                     ; 0650
                                       0D  11 00047         BRB     2$                                ; 0646
                 08  AE                01  D0 00049 1$:     MOVL    #1, TRUNC_FLAG                    ; 0654
                     50          D8F0  C2  9E 0004D         MOVAB   -10000(R2), R0                    ; 0655
                 0C  AE                50  CE 00052         MNEGL   R0, ROUND_POS                     ;
                 18  AE  010F0001      8F  D0 00056 2$:     MOVL    #17760257, ONE_DESC               ; 0666
                 1C  AE            04  AE  9E 0005E         MOVAB   ONE_BUF, ONE_DESC+4               ; 0669
                 04  AE                31  90 00063         MOVB    #49, ONE_BUF                      ; 0670
                                   20  AE  9F 00067         PUSHAB  A_DESC                            ; 0671
                                   14  AE  9F 0006A         PUSHAB  A_EXP                             ;
                                   1C  AE  9F 0006D         PUSHAB  A_SIGN                            ;
                                   24  AE  9F 00070         PUSHAB  ONE_DESC                          ;
                                   1C  AE  9F 00073         PUSHAB  ROUND_POS                         ;
                                   28  AE  9F 00076         PUSHAB  A_SIGN                            ;
                                   38  AE  9F 00079         PUSHAB  A_DESC                            ;
                                   2C  AE  9F 0007C         PUSHAB  A_EXP                             ;
                                   34  AE  9F 0007F         PUSHAB  A_SIGN                            ;
                     63                09  FB 00082         CALLS   #9, STR$ADD                       ;
                                   20  AE  9F 00085         PUSHAB  A_DESC                            ; 0674
                                   14  AE  9F 00088         PUSHAB  A_EXP                             ;
                                   1C  AE  9F 0008B         PUSHAB  A_SIGN                            ;
                                   2C  AE  9F 0008E         PUSHAB  A_DESC                            ;
                                   20  AE  9F 00091         PUSHAB  A_EXP                             ;
                                   28  AE  9F 00094         PUSHAB  A_SIGN                            ;
                                   20  AE  9F 00097         PUSHAB  TRUNC_FLAG                        ;
                     50            3C  AE  3C 0009A         MOVZWL  A_DESC, R0                        ;
                     50            2C  AE  C0 0009E         ADDL2   A_EXP, R0                         ;
      1C  AE         50            28  AE  C3 000A2         SUBL3   ROUND_POS, R0, 28(SP)             ;
                                   1C  AE  9F 000A8         PUSHAB  28(SP)                            ;
          00000000G  00                08  FB 000AB         CALLS   #8, STR$ROUND                     ;
                                   20  AE  9F 000B2         PUSHAB  A_DESC                            ; 0677
                                   14  AE  9F 000B5         PUSHAB  A_EXP                             ;
                                   1C  AE  9F 000B8         PUSHAB  A_SIGN                            ;
                                   24  AE  9F 000BB         PUSHAB  ONE_DESC                          ;
                                   1C  AE  9F 000BE         PUSHAB  ROUND_POS                         ;
                     05            28  AE  E9 000C1         BLBC    A_SIGN, 3$                        ; 0678
                                   14  AE  D4 000C5         CLRL    20(SP)                            ;
                                       04  11 000C8         BRB     4$                                ;
                 14  AE                01  D0 000CA 3$:     MOVL    #1, 20(SP)                        ;
                                   14  AE  9F 000CE 4$:     PUSHAB  20(SP)                            ;
                                   38  AE  9F 000D1         PUSHAB  A_DESC                            ; 0677
                                   2C  AE  9F 000D4         PUSHAB  A_EXP                             ;
                                   34  AE  9F 000D7         PUSHAB  A_SIGN                            ;
                     63                09  FB 000DA         CALLS   #9, STR$ADD                       ;
           FFFFEC79  8F                52  D1 000DD         CMPL    R2, #-4999                        ; 0690
                                       10  19 000E4         BLSS    5$                                ;
                                       52  D5 000E6         TSTL    R2                                ;
                                       0C  18 000E8         BGEQ    5$                                ;
                                   10  AE  D5 000EA         TSTL    A_EXP                             ; 0692
                                       31  13 000ED         BEQL    8$                                ;
                     50                52  D0 000EF         MOVL    R2, R0                            ; 0694
```

```
                                  25  19 000F2          BLSS    6$                          :
                                  26  11 000F4          BRB     7$                          :
       00001389  8F               52  D1 000F6 5$:      CMPL    R2, #5001                   : 0696
                                  21  19 000FD          BLSS    8$                          :
       0000270F  8F               52  D1 000FF          CMPL    R2, #9999                   :
                                  18  14 00106          BGTR    8$                          :
    50 00002710  8F               52  C3 00108          SUBL3   R2, #10000, TEMP            : 0700
                          10      AE  D5 00110          TSTL    A_EXP                       : 0701
                                  0B  13 00113          BEQL    8$                          :
                                  50  D5 00115          TSTL    R0                          : 0703
                                  03  18 00117          BGEQ    7$                          :
                 50               50  CE 00119 6$:      MNEGL   R0, R0                      :
             10  AE               50  C2 0011C 7$:      SUBL2   R0, A_EXP                   :
                          04      AC  DD 00120 8$:      PUSHL   RESULT_DESC                 : 0710
                          14      AE  DD 00123          PUSHL   A_EXP                       :
                          1C      AE  DD 00126          PUSHL   A_SIGN                      :
                          2C      AE  9F 00129          PUSHAB  A_DESC                      :
          0000V  CF               04  FB 0012C          CALLS   #4, PARSE_OUT               :
                          20      AE  9F 00131          PUSHAB  A_DESC                      : 0714
      00000000G  00               01  FB 00134          CALLS   #1, STR$FREE1_DX            :
                                      04 0013B          RET                                 : 0716
                                    0000 0013C 9$:      .WORD   Save nothing                : 0602
                 50       08      AC  D0 0013E          MOVL    8(AP), R0                   :
                 50       04      A0  D0 00142          MOVL    4(R0), R0                   :
                          F8      A0  9F 00146          PUSHAB  A_DESC                      :
                                  01  DD 00149          PUSHL   #1                          :
                                  5E  DD 0014B          PUSHL   SP                          :
                 7E       04      AC  7D 0014D          MOVQ    4(AP), -(SP)                :
          0000V  CF               03  FB 00151          CALLS   #3, FREE_STRINGS            :
                                      04 00156          RET                                 :

; Routine Size:  343 bytes,    Routine Base:  _BAS$CODE + 03C1

;  627          0717  1
```

```
1 GLOBAL ROUTINE BAS$PROD (                               ! Multiply strings
1           PROD_DESC,                                    ! Product
1           OP1_DESC,                                     ! First input
1           OP2_DESC,                                     ! Second input
1           PRECISION                                     ! Digits of result
1      ) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !         Multiply two strings.  PROD := OP1 * OP2
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !         PROD_DESC.wt.dx The product of OP1 and OP2
1 !         OP1_DESC.rt.dx  Operand OP1
1 !         OP2_DESC.rt.dx  Operand OP2
1 !         PRECISION.rl.v  Number of digits to retain and, encoded,
1 !                         the round/truncate flag.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !         NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !         NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !         NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !         Signals PROLOSSOR for certain "impossible" errors, and the signals
1 !         from PARSE_IN and PARSE_OUT.
1 !
1 !--
1
2     BEGIN
2
2     MAP
2         OP1_DESC : REF BLOCK [8, BYTE],
2         OP2_DESC : REF BLOCK [8, BYTE],
2         PROD_DESC : REF BLOCK [8, BYTE];
2
2     LOCAL
2 !+
2 ! Internal form of OP1.
2 !-
2         A_DESC : BLOCK [8, BYTE] VOLATILE,
2         A_SIGN,
2         A_EXP,
2 !+
2 ! Internal form of OP2.
2 !-
```

```
        B_DESC : BLOCK [8, BYTE] VOLATILE,
        B_SIGN,
        B_EXP,
!+
! Internal form of the product, A*B.
!-
        C_DESC : BLOCK [8, BYTE] VOLATILE,
        C_SIGN,
        C_EXP,
        TRUNC_FLAG,                             ! 1 = truncate, 0 = round.
        ROUND_POS,                              ! Where to round the product
!+
! This is the constant 1, which is muliplied by a power of 10 for rounding
! purposes.
!-
        ONE_DESC : BLOCK [8, BYTE],
        ONE_BUF : VECTOR [1, BYTE];

!+
! Enable a handler to free the local strings.
!-

    ENABLE
        FREE_STRINGS (A_DESC, B_DESC, C_DESC);

!+
! Convert the two input arguments from external form to internal form.
! This is done by removing the non-digits from the string and
! returning the sign and exponent as separate values.
! Errors are signaled.
!-
    A_DESC [DSC$W_LENGTH] = 0;
    A_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
    A_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
    A_DESC [DSC$A_POINTER] = 0;
    PARSE_IN (.OP1_DESC, A_DESC, A_SIGN, A_EXP);
    B_DESC [DSC$W_LENGTH] = 0;
    B_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
    B_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
    B_DESC [DSC$A_POINTER] = 0;
    PARSE_IN (.OP2_DESC, B_DESC, B_SIGN, B_EXP);
!+
! Multiply the numbers using the large-precision string arithmetic
! package.
!-
    C_DESC [DSC$W_LENGTH] = 0;
    C_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
    C_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
    C_DESC [DSC$A_POINTER] = 0;
    STR$MUL (A_SIGN, A_EXP, A_DESC, B_SIGN, B_EXP, B_DESC, C_SIGN, C_EXP, C_DESC);
!+
! We are done with A and B, so free them.
!-
    STR$FREE1_DX (A_DESC);
    STR$FREE1_DX (B_DESC);
!+
! Round or truncate the result to a specified decimal place.
```

```
!-

    IF (.PRECISION LEQ BAS$K_PREC_VAL)
    THEN
        BEGIN
        TRUNC_FLAG = 0;
        ROUND_POS = -.PRECISION;
        END
    ELSE
        BEGIN
        TRUNC_FLAG = 1;
        ROUND_POS = -(.PRECISION - (2*BAS$K_PREC_VAL));
        END;

!+
! In order to do the BASIC rounding/truncation based on decimal place
! using the string package's rounding/truncation based on number of
! significant digits, we must be sure that the position we are rounding
! to is part of the significance.  Therefore we add 1E<ROUND_POS>,
! round or truncate to discard all lower digits, and then subtract
! 1E<ROUND_POS> to get back to where we belong.
!-
    ONE_DESC [DSC$W_LENGTH] = 1;
    ONE_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
    ONE_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
    ONE_DESC [DSC$A_POINTER] = ONE_BUF [0];
    ONE_BUF [0] = %C'1';
    STR$ADD (C_SIGN, C_EXP, C_DESC,                 !
        C_SIGN, ROUND_POS, ONE_DESC,                !
        C_SIGN, C_EXP, C_DESC);
    STR$ROUND (%REF (.C_EXP + .C_DESC [DSC$W_LENGTH] - .ROUND_POS), TRUNC_FLAG,       !
        C_SIGN, C_EXP, C_DESC,                      !
        C_SIGN, C_EXP, C_DESC);
    STR$ADD (C_SIGN, C_EXP, C_DESC,                 !
        %REF ((IF (.C_SIGN) THEN 0 ELSE 1)), ROUND_POS, ONE_DESC,        !
        C_SIGN, C_EXP, C_DESC);

!+
! Check if the precision value is in the range of -1 to -4 (indicating
! rounding) or if the value is in the range of 9996 to 9999 (indicating
! truncation).  In either case, there is an incompatibility to be fixed
! here, so that VAX BASIC returns the same value as BASIC+2.
!-

    SELECTONE .PRECISION OF
      SET
        [-4999 TO -1]:                              ! Rounding parameter
          BEGIN
          IF .C_EXP NEQ 0
            THEN
              C_EXP = .C_EXP - ABS(.PRECISION);
          END;
        [5001 TO 9999]:                             ! Truncation parameter
          BEGIN
            LOCAL
              TEMP;
          TEMP = 10000 - .PRECISION;
```

```
;  800        0889  3          IF .C_EXP NEQ 0
;  801        0890  3            THEN
;  802        0891  3              C_EXP = .C_EXP - ABS(.TEMP);
;  803        0892  2          END;
;  804        0893  2       TES;
;  805        0894  2
;  806        0895  2  !+
;  807        0896  2  ! Convert the product to external form for the caller.
;  808        0897  2  !-
;  809        0898  2     PARSE_OUT (C_DESC, .C_SIGN, .C_EXP, .PROD_DESC);
;  810        0899  2  !+
;  811        0900  2  ! We can now free C
;  812        0901  2  !-
;  813        0902  2     STR$FREE1_DX (C_DESC);
;  814        0903  2     RETURN;
;  815        0904  1     END;                                    ! end of BAS$PROD



                                  001C 00000        .ENTRY  BAS$PROD, Save R2,R3,R4                     ; 0718
            54 00000000G    00    9E 00002        MOVAB   STR$ADD, R4                                 ;
            53 00000000G    00    9E 00009        MOVAB   STR$FREE1_DX, R3                            ;
            5E          B8  AE    9E 00010        MOVAB   -72(SP), SP                                 ;
                        30  AE    7C 00014        CLRQ    C_DESC                                      ; 0758
                        38  AE    7C 00017        CLRQ    B_DESC                                      ;
                        40  AE    7C 0001A        CLRQ    A_DESC                                      ;
            6D        0180  CF    DE 0001D        MOVAL   9$, (FP)                                    ;
                        40  AE    B4 00022        CLRW    A_DESC                                      ; 0806
        42  AE                0F  90 00025        MOVB    #15, A_DESC+2                               ; 0807
        43  AE                02  90 00029        MOVB    #2, A_DESC+3                                ; 0808
                        44  AE    D4 0002D        CLRL    A_DESC+4                                    ; 0809
                        0C  AE    9F 00030        PUSHAB  A_EXP                                       ; 0810
                        14  AE    9F 00033        PUSHAB  A_SIGN                                      ;
                        48  AE    9F 00036        PUSHAB  A_DESC                                      ;
                        08  AC    DD 00039        PUSHL   OP1_DESC                                    ;
     0000V  CF                04  FB 0003C        CALLS   #4, PARSE_IN                                ;
                        38  AE    B4 00041        CLRW    B_DESC                                      ; 0811
        3A  AE                0F  90 00044        MOVB    #15, B_DESC+2                               ; 0812
        3B  AE                02  90 00048        MOVB    #2, B_DESC+3                                ; 0813
                        3C  AE    D4 0004C        CLRL    B_DESC+4                                    ; 0814
                        04  AE    9F 0004F        PUSHAB  B_EXP                                       ; 0815
                        0C  AE    9F 00052        PUSHAB  B_SIGN                                      ;
                        40  AE    9F 00055        PUSHAB  B_DESC                                      ;
                        0C  AC    DD 00058        PUSHL   OP2_DESC                                    ;
     0000V  CF                04  FB 0005B        CALLS   #4, PARSE_IN                                ;
                        30  AE    B4 00060        CLRW    C_DESC                                      ; 0820
        32  AE                0F  90 00063        MOVB    #15, C_DESC+2                               ; 0821
        33  AE                02  90 00067        MOVB    #2, C_DESC+3                                ; 0822
                        34  AE    D4 0006B        CLRL    C_DESC+4                                    ; 0823
                        30  AE    9F 0006E        PUSHAB  C_DESC                                      ; 0824
                        24  AE    9F 00071        PUSHAB  C_EXP                                       ;
                        2C  AE    9F 00074        PUSHAB  C_SIGN                                      ;
                        44  AE    9F 00077        PUSHAB  B_DESC                                      ;
                        14  AE    9F 0007A        PUSHAB  B_EXP                                       ;
                        1C  AE    9F 0007D        PUSHAB  B_SIGN                                      ;
```

```
                                58    AE  9F 00080       PUSHAB  A_DESC                       :
                                28    AE  9F 00083       PUSHAB  A_EXP                        :
                                30    AE  9F 00086       PUSHAB  A_SIGN                       :
              00000000G  00           09  FB 00089       CALLS   #9, STR$MUL                  :
                                40    AE  9F 00090       PUSHAB  A_DESC                       : 0828
                         63           01  FB 00093       CALLS   #1, STR$FREE1_DX             :
                                38    AE  9F 00096       PUSHAB  B_DESC                       : 0829
                         63           01  FB 00099       CALLS   #1, STR$FREE1_DX             :
                         52     10    AC  D0 0009C       MOVL    PRECISION, R2                : 0834
              00001388   8F           52  D1 000A0       CMPL    R2, #5000                    :
                                      09  14 000A7       BGTR    1$                           :
                                18    AE  D4 000A9       CLRL    TRUNC_FLAG                   : 0837
                     1C  AE           52  CE 000AC       MNEGL   R2, ROUND_POS                : 0838
                                      0D  11 000B0       BRB     2$                           : 0834
                     18  AE           01  D0 000B2 1$:   MOVL    #1, TRUNC_FLAG               : 0842
                         50   D8F0    C2  9E 000B6       MOVAB   -10000(R2), R0               : 0843
                     1C  AE           50  CE 000BB       MNEGL   R0, ROUND_POS                :
                     28  AE 010F0001  8F  D0 000BF 2$:   MOVL    #17760257, ONE_DESC          : 0854
                     2C  AE     14    AE  9E 000C7       MOVAB   ONE_BUF, ONE_DESC+4          : 0857
                     14  AE           31  90 000CC       MOVB    #49, ONE_BUF                 : 0858
                                30    AE  9F 000D0       PUSHAB  C_DESC                       : 0859
                                24    AE  9F 000D3       PUSHAB  C_EXP                        :
                                2C    AE  9F 000D6       PUSHAB  C_SIGN                       :
                                34    AE  9F 000D9       PUSHAB  ONE_DESC                     :
                                2C    AE  9F 000DC       PUSHAB  ROUND_POS                    :
                                38    AE  9F 000DF       PUSHAB  C_SIGN                       :
                                48    AE  9F 000E2       PUSHAB  C_DESC                       :
                                3C    AE  9F 000E5       PUSHAB  C_EXP                        :
                                44    AE  9F 000E8       PUSHAB  C_SIGN                       :
                         64           09  FB 000EB       CALLS   #9, STR$ADD                  :
                                30    AE  9F 000EE       PUSHAB  C_DESC                       : 0862
                                24    AE  9F 000F1       PUSHAB  C_EXP                        :
                                2C    AE  9F 000F4       PUSHAB  C_SIGN                       :
                                3C    AE  9F 000F7       PUSHAB  C_DESC                       :
                                30    AE  9F 000FA       PUSHAB  C_EXP                        :
                                38    AE  9F 000FD       PUSHAB  C_SIGN                       :
                                30    AE  9F 00100       PUSHAB  TRUNC_FLAG                   :
                         50     4C    AE  3C 00103       MOVZWL  C_DESC, R0                   :
                         50     3C    AE  C0 00107       ADDL2   C_EXP, R0                    :
       1C    AE          50     38    AE  C3 0010B       SUBL3   ROUND_POS, R0, 28(SP)        :
                                1C    AE  9F 00111       PUSHAB  28(SP)                       :
              00000000G  00           08  FB 00114       CALLS   #8, STR$ROUND                :
                                30    AE  9F 0011B       PUSHAB  C_DESC                       : 0865
                                24    AE  9F 0011E       PUSHAB  C_EXP                        :
                                2C    AE  9F 00121       PUSHAB  C_SIGN                       :
                                34    AE  9F 00124       PUSHAB  ONE_DESC                     :
                                2C    AE  9F 00127       PUSHAB  ROUND_POS                    :
                         05     38    AE  E9 0012A       BLBC    C_SIGN, 3$                   : 0866
                                14    AE  D4 0012E       CLRL    20(SP)                       :
                                      04  11 00131       BRB     4$                           :
                     14  AE           01  D0 00133 3$:   MOVL    #1, 20(SP)                   :
                                14    AE  9F 00137 4$:   PUSHAB  20(SP)                       :
                                48    AE  9F 0013A       PUSHAB  C_DESC                       : 0865
                                3C    AE  9F 0013D       PUSHAB  C_EXP                        :
                                44    AE  9F 00140       PUSHAB  C_SIGN                       :
                         64           09  FB 00143       CALLS   #9, STR$ADD                  :
              FFFFEC79   8F           52  D1 00146       CMPL    R2, #-4999                   : 0878
```

```
                                10  19 0014D          BLSS    5$                          ;
                                52  D5 0014F          TSTL    R2                          ;
                                0C  18 00151          BGEQ    5$                          ;
                         20     AE  D5 00153          TSTL    C_EXP                       ; 0880
                  50            31  13 00156          BEQL    8$                          ;
                                52  D0 00158          MOVL    R2, R0                      ; 0882
                                25  19 0015B          BLSS    6$                          ;
                                26  11 0015D          BRB     7$                          ;
      00001389    8F            52  D1 0015F 5$:      CMPL    R2, #5001                   ; 0884
                                21  19 00166          BLSS    8$                          ;
      0000270F    8F            52  D1 00168          CMPL    R2, #9999                   ;
                                18  14 0016F          BGTR    8$                          ;
   50 00002710    8F            52  C3 00171          SUBL3   R2, #10000, TEMP            ; 0888
                         20     AE  D5 00179          TSTL    C_EXP                       ; 0889
                                0B  13 0017C          BEQL    8$                          ;
                                50  D5 0017E          TSTL    R0                          ; 0891
                                03  18 00180          BGEQ    7$                          ;
                  50            50  CE 00182 6$:      MNEGL   R0, R0                      ;
            20    AE            50  C2 00185 7$:      SUBL2   R0, C_EXP                   ;
                         04     AC  DD 00189 8$:      PUSHL   PROD_DESC                   ; 0898
                         24     AE  DD 0018C          PUSHL   C_EXP                       ;
                         2C     AE  DD 0018F          PUSHL   C_SIGN                      ;
                         3C     AE  9F 00192          PUSHAB  C_DESC                      ;
         0000V    CF            04  FB 00195          CALLS   #4, PARSE_OUT               ;
                         30     AE  9F 0019A          PUSHAB  C_DESC                      ; 0902
                  63            01  FB 0019D          CALLS   #1, STR$FREE1_DX            ;
                                    04 001A0          RET                                 ; 0904
                                  0000 001A1 9$:      .WORD   Save nothing                ; 0758
                  50     08     AC  D0 001A3          MOVL    8(AP), R0                   ;
                  50     04     A0  D0 001A7          MOVL    4(R0), R0                   ;
                         E8     A0  9F 001AB          PUSHAB  C_DESC                      ;
                         F0     A0  9F 001AE          PUSHAB  B_DESC                      ;
                         F8     A0  9F 001B1          PUSHAB  A_DESC                      ;
                                03  DD 001B4          PUSHL   #3                          ;
                                5E  DD 001B6          PUSHL   SP                          ;
                  7E     04     AC  7D 001B8          MOVQ    4(AP), -(SP)                ;
         0000V    CF            03  FB 001BC          CALLS   #3, FREE_STRINGS            ;
                                    04 001C1          RET                                 ;

; Routine Size:  450 bytes,    Routine Base:  _BAS$CODE + 0518


;  816          0905  1
```

```
1 GLOBAL ROUTINE BAS$QUO (                               ! Divide strings
1           QUO_DESC,                                    ! Quotient
1           OP1_DESC,                                    ! First input
1           OP2_DESC,                                    ! Second input
1           PRECISION                                    ! Digits of result
1      ) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !         Divide two strings.  QUO := OP1 / OP2
1 !         No more than 55 digits of precision are permitted.
1 !         (56 if truncating.)
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !         QUO_DESC.wt.dx  The quotient of OP1 and OP2
1 !         OP1_DESC.rt.dx  Operand OP1
1 !         OP2_DESC.rt.dx  Operand OP2
1 !         PRECISION.rl.v  Number of digits to retain and, encoded,
1 !                         the round/truncate flag.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !         NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !         NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !         NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !         Signals BAS$_FLOPOIERR if the number of places is greater than
1 !         55 or 56.  Signals BAS$_DIVBY_ZER if the divisor is zero.
1 !         Also, PARSE_IN and PARSE_OUT signal.
1 !--
1
2     BEGIN
2
2     MAP
2         OP1_DESC : REF BLOCK [8, BYTE],
2         OP2_DESC : REF BLOCK [8, BYTE],
2         QUO_DESC : REF BLOCK [8, BYTE];
2
2     LOCAL
2 !+
2 ! Internal form of OP1.
2 !-
2         A_DESC : BLOCK [8, BYTE] VOLATILE,
2         A_SIGN,
2         A_EXP,
2 !+
```

```
   2 ! Internal form of OP2.
   2 !-
   2          B_DESC : BLOCK [8, BYTE] VOLATILE,
   2          B_SIGN,
   2          B_EXP,
   2 !+
   2 ! Internal form of quotient.
   2 !-
   2          C_DESC : BLOCK [8, BYTE] VOLATILE,
   2          C_SIGN,
   2          C_EXP,
   2
   2          RND_TRUNC,                                  ! 0 = truncate, 1 = round
   2          PREC,                                       ! Number of digits to the
   2                                                      !   right of the decimal point
   2          END_ADDR,                                   ! End of string address
   2          NEW_ADDR,                                   !\Used to store resultant
   2                                                      !/address from SPANC
   2          BUF,                                        ! Pointer to string of digits
   2          SAV_ADDR,                                   ! Temporary storage address
   2          LENGTH,                                     ! Used in SPANC
   2          C_LENGTH,                                   ! Used to store length of string
   2          SAV_C_LEN,                                  ! storage for actual len. of C
   2          SAV_C_PTR,                                  ! storage for actual addr of C
   2          SAV_BUF,                                    ! Storage for buffer address
   2          SAV_EXP,                                    ! Storage for exp value
   2          SAV_LENGTH,                                 ! Storage for length value
   2          DIFF;                                       ! Used in SPANC to calc offsets
   2 !+
   2 ! The following FORTRAN subroutines can be used to print the progress of
   2 ! this routine, if the calls to them are enabled.
   2 !-
C  2      %(
C  2          SUBROUTINE MONITOR_I (ICODE, IVAL)
C  2          IMPLICIT INTEGER (A-Z)
C  2          TYPE 900, ICODE, IVAL
C  2 900      FORMAT (I3,I6)
C  2          RETURN
C  2          END
C  2          SUBROUTINE MONITOR_T (ICODE, IVAL)
C  2          IMPLICIT INTEGER (A-Z)
C  2          CHARACTER*(*) IVAL
C  2          TYPE 901, ICODE, IVAL
C  2 901      FORMAT (I3, 3X, A)
C  2          RETURN
C  2          END
   2 )%
   2 !+
   2 ! In the printout from the above code,
   2 !
   2 !        1 = eps_exp
   2 !        2 = delta_exp
   2 !        3 = round_pos
   2 !        4 = c_exp
   2 !        5 = trial_exp
   2 !        6 = trial_digits
   2 !        7 = c digits
```

```
  2 !-
  2
  2     COMPILETIME
  2         PERFORM_MONITORING = 0;                         ! Set =1 to get calls to monitors
  2
L 2 %IF PERFORM_MONITORING
U 2 %THEN
U 2
U 2     EXTERNAL ROUTINE
U 2         MONITOR_I,                                      ! Print an integer
U 2         MONITOR_T;                                      ! Print text
U 2
  2 %FI
  2
  2 !+
  2 ! Enable a handler to free the local strings in case of error.
  2 !-
  2
  2     ENABLE
  2         FREE_STRINGS (A_DESC, B_DESC, C_DESC);
  2
  2 !+
  2 ! Convert the two input arguments from external form to internal form.
  2 ! This is done by removing the non-digits from the string and
  2 ! returning the sign and exponent as separate values.
  2 ! Errors are signaled.
  2 !-
  2     A_DESC [DSC$W_LENGTH] = 0;
  2     A_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
  2     A_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
  2     A_DESC [DSC$A_POINTER] = 0;
  2     PARSE_IN (.OP1_DESC, A_DESC, A_SIGN, A_EXP);
  2
  2     B_DESC [DSC$W_LENGTH] = 0;
  2     B_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
  2     B_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
  2     B_DESC [DSC$A_POINTER] = 0;
  2     PARSE_IN (.OP2_DESC, B_DESC, B_SIGN, B_EXP);
  2
  2 !+
  2 ! Set up descriptor for internal form of C
  2 !-
  2
  2     C_DESC [DSC$W_LENGTH] = 0;
  2     C_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
  2     C_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
  2     C_DESC [DSC$A_POINTER] = 0;
  2
  2 !+
  2 ! Set up parameters for call to STR$DIVIDE
  2 !-
  2
  3     IF (.PRECISION LEQ BAS$K_PREC_VAL)
  2     THEN
  3         BEGIN
  3         RND_TRUNC = 1;
  3         PREC = .PRECISION;
```

```
        END
    ELSE
        BEGIN
        RND_TRUNC = 0;
        PREC = .PRECISION - 2*BAS$K_PREC_VAL;
        END;

!+
! For compatability with the PDP-11, don't allow more than
! 55 or 56 digits to the right of the decimal point.
!-

    IF .C_DESC[DSC$W_LENGTH] GTR (BAS$K_PREC_LIM2 - .RND_TRUNC)
    THEN
        BAS$$SIGNAL (BAS$K_FLOPOIERR);


!+
! The next section of code is to interface with the new routine STR$DIVIDE.
!-

    STR$DIVIDE (A_SIGN,A_EXP,A_DESC,
                B_SIGN,B_EXP,B_DESC,
                PREC,RND_TRUNC,
                C_SIGN,C_EXP,C_DESC);           ! Invoke STR$DIVIDE
%IF PERFORM_MONITORING
%THEN
    MONITOR_I (%REF (3), %REF (.ROUND_POS));
%FI

!+
! The following code is to ensure compatibility with the PDP-11.
! Need to strip off any leading zeroes or any trailing zeroes.
!-

!+
! save descriptor for C because the following code modifies it and it must
! be restored before calling STR$FREE1_DX.
!-
    SAV_C_LEN = .C_DESC [DSC$W_LENGTH];
    SAV_C_PTR = .C_DESC [DSC$A_POINTER];

!+
! Initial set-up.  Set-up pointers to various sections within
! the resultant string.
!-

    BUF = .C_DESC[DSC$A_POINTER];                       ! Fetch addr of buffer
    C_LENGTH = .C_DESC[DSC$W_LENGTH];                   ! Fetch length of string
    END_ADDR = .BUF + .C_LENGTH;                        ! Ptr to end of buffer
    SAV_EXP = .C_EXP;                                   ! Save the exp value
    SAV_LENGTH = .C_LENGTH;                             ! Save the length of C
    SAV_BUF = .BUF;                                     ! Save ptr to beg of buffer

!+
! First check if the exponent value if less than the length of
! the string.  If it is, then we know that we have digits to the
```

```
2   ! left of the decimal point.
2   !-
2
2   IF (.C_EXP) LSS 0
2     THEN
3       BEGIN
3       IF ABS(.C_EXP) LSS .C_LENGTH
3         THEN
4           BEGIN
4           LENGTH = ABS (.C_EXP);
4           BUF = .BUF + (.C_LENGTH + .C_EXP);
4           END
3         ELSE
3           LENGTH = .C_LENGTH;
3
3   !+
3   ! Check for first non-zero digit to right of decimal point.
3   !-
3
3       NEW_ADDR = SPANC (LENGTH,.BUF,TABLE_NZ,MASK);
3       IF .NEW_ADDR EQL 0                              !\Indicates all zeroes to
3                                                       !/right of decimal point
3         THEN
4           BEGIN
4           IF ABS(.C_EXP) GEQ .C_LENGTH
4             THEN
5               BEGIN                                   ! Here, length = exponent value
5               C_LENGTH = 1;                           ! Just set length to one
5               C_EXP = 0;                              ! Set exp of result to zero
5               END
4             ELSE
5               BEGIN                                   ! Here, exponent < length
5               C_LENGTH = .C_LENGTH - ABS(.C_EXP);     ! Decr C_LENGTH by the exp value
5               C_EXP = 0;                              ! Set exp of result to zero
4               END;
4           C_DESC[DSC$W_LENGTH] = .C_LENGTH;           ! Store updated length
4           END
3         ELSE
3           !+
3           ! At this point, we have the address of the first non-zero
3           ! position within the resultant string.  Now we need to search
3           ! for a zero digit.  If none, then we know that all the digits
3           ! to the right of the decimal point are non-zero, and we can
3           ! leave the exponent value and length value as is.
3           !-
4           BEGIN
4           !+
4           ! Start searching for a zero digit, starting at the address
4           ! returned from the previous SPANC.
4           !-
4
4           IF .NEW_ADDR NEQ .END_ADDR                              ! If not at end of string
4             THEN
5               BEGIN
5               WHILE 1 DO
6                 BEGIN
6                 SAV_ADDR = .NEW_ADDR;                             ! Save the current addr
```

```
6                       LENGTH = .END_ADDR - .NEW_ADDR;                 ! Calculate new length
6                       NEW_ADDR = SPANC (LENGTH,.NEW_ADDR,TABLE_Z,MASK);
6                       IF .NEW_ADDR NEQ 0                              !\Indicates a zero
6                                                                       !/digit has been found
6                         THEN
7                           BEGIN
7                           !+
7                           ! Now search for a non-zero digit.
7                           !-
7                           SAV_ADDR = .NEW_ADDR;                       ! Save current address
7                           LENGTH = .END_ADDR - .NEW_ADDR;             ! Calculate length of string to be search
7                           NEW_ADDR = SPANC (LENGTH,.NEW_ADDR,TABLE_NZ,MASK);
7                           IF .NEW_ADDR EQL 0                          ! No match found
7                             THEN
8                               BEGIN
8                               DIFF = .END_ADDR - .SAV_ADDR;           !\Calculate length of
8                                                                       !/string that is zero
8                               C_LENGTH = .C_LENGTH - .DIFF;           ! Decr length by that amount
8                               C_EXP = - (ABS(.C_EXP) - .DIFF);        ! Decr exponent value by that amount
8                               C_DESC[DSC$W_LENGTH] = .C_LENGTH;       ! Store updated length
8                               EXITLOOP                                ! Escape from loop
7                               END;
7                             END
6                         ELSE
6                             EXITLOOP
5                       END;
4                   END;
3               END;
2             END;
3           IF (.SAV_EXP LEQ 0) AND (ABS(.SAV_EXP) LSS .SAV_LENGTH)
2             THEN
3               BEGIN
3                 LOCAL
3                   LEFT_DIGITS;
3
3               !+
3               ! Here we know that C_EXP is less than C_LENGTH.
3               ! Now we are ready to strip off leading zeroes that
3               ! may exist to the left side of the decimal point.
3               !-
3
3               LEFT_DIGITS = .C_LENGTH - ABS(.C_EXP);                  !\Calc # of digits to
3                                                                       !/left of decimal point
3               NEW_ADDR = SPANC (LEFT_DIGITS,.SAV_BUF,TABLE_NZ,MASK);  !\Search for
3                                                                       !/non-zero
3               IF .NEW_ADDR EQL 0                                      ! All zeroes to left of D.P.
3                 THEN
4                   BEGIN
4                   C_LENGTH = .C_LENGTH - .LEFT_DIGITS;                ! Decr C_LENGTH
4                   IF .C_LENGTH EQL 0
4                     THEN
4                       C_LENGTH = 1                                    ! Ensure length of 1
4                     ELSE
4                       C_DESC[DSC$A_POINTER] = .SAV_BUF + .LEFT_DIGITS; !Update buffer address
4                   END
3                 ELSE
4                   BEGIN
```

```
4                DIFF = .NEW_ADDR - .SAV_BUF;             ! Calculate offset
4                C_LENGTH = .C_LENGTH - .DIFF;            ! Update length
4                C_DESC[DSC$A_POINTER] = .SAV_BUF + .DIFF;! Update buffer address
3                END;
3            C_DESC[DSC$W_LENGTH] = .C_LENGTH;
3            END
2          ELSE
2            IF .SAV_EXP GTR 0
2              THEN
2          !+
2          ! Here we know that the exponent value is positive.
2          !-
3            BEGIN
3            NEW_ADDR = SPANC (C_LENGTH,.SAV_BUF,TABLE_NZ,MASK);
3            IF .NEW_ADDR EQL 0                          !\Indicates all zeroes
3                                                        !/to left of decimal pt
3              THEN
4                BEGIN
4                C_LENGTH = 1;                           ! C_LENGTH = 1
4                C_EXP = 0;                              ! Set exponent to zero
4                END
3              ELSE
4                BEGIN
4                DIFF = .NEW_ADDR - .SAV_BUF;
4                C_LENGTH = .C_LENGTH - .DIFF;
4                C_DESC[DSC$A_POINTER] = .SAV_BUF + .DIFF;
3                END;
3            C_DESC[DSC$W_LENGTH] = .C_LENGTH;
2            END;
2
2
2    !+
2    ! Convert the quotient to external form for the caller.
2    !-
2        PARSE_OUT (C_DESC, .C_SIGN, .C_EXP, .QUO_DESC);
2
2    !+
2    ! restore actual length & address of C so we don't fake out STR$FREE1 and cause
2    ! an access violation
2    !-
2        C_DESC [DSC$W_LENGTH] = .SAV_C_LEN;
2        C_DESC [DSC$A_POINTER] = .SAV_C_PTR;
2
2    !+
2    ! We can now free A, B and C
2    !-
2        STR$FREE1_DX (A_DESC);
2        STR$FREE1_DX (B_DESC);
2        STR$FREE1_DX (C_DESC);
2        RETURN;
1        END;                                    ! end of BAS$QUO


                              0FFC 00000        .ENTRY  BAS$QUO, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,- ; 0906
```

```
                                                         R11                           ;
                 5E      BC   AE   9E 00002      MOVAB   -68(SP), SP                   ;
                         2C   AE   7C 00006      CLRQ    C_DESC                        ; 0948
                         34   AE   7C 00009      CLRQ    B_DESC                        ;
                         3C   AE   7C 0000C      CLRQ    A_DESC                        ;
                 6D    0215   CF   DE 0000F      MOVAL   27$, (FP)                     ;
                         3C   AE   B4 00014      CLRW    A_DESC                        ; 1047
             3E   AE          0F   90 00017      MOVB    #15, A_DESC+2                 ; 1048
             3F   AE          02   90 0001B      MOVB    #2, A_DESC+3                  ; 1049
                         40   AE   D4 0001F      CLRL    A_DESC+4                      ; 1050
                         24   AE   9F 00022      PUSHAB  A_EXP                         ; 1051
                         2C   AE   9F 00025      PUSHAB  A_SIGN                        ;
                         44   AE   9F 00028      PUSHAB  A_DESC                        ;
                         08   AC   DD 0002B      PUSHL   OP1_DESC                      ;
           0000V  CF          04   FB 0002E      CALLS   #4, PARSE_IN                  ;
                         34   AE   B4 00033      CLRW    B_DESC                        ; 1053
             36   AE          0F   90 00036      MOVB    #15, B_DESC+2                 ; 1054
             37   AE          02   90 0003A      MOVB    #2, B_DESC+3                  ; 1055
                         38   AE   D4 0003E      CLRL    B_DESC+4                      ; 1056
                         1C   AE   9F 00041      PUSHAB  B_EXP                         ; 1057
                         24   AE   9F 00044      PUSHAB  B_SIGN                        ;
                         3C   AE   9F 00047      PUSHAB  B_DESC                        ;
                         0C   AC   DD 0004A      PUSHL   OP2_DESC                      ;
           0000V  CF          04   FB 0004D      CALLS   #4, PARSE_IN                  ;
                         2C   AE   B4 00052      CLRW    C_DESC                        ; 1063
             2E   AE          0F   90 00055      MOVB    #15, C_DESC+2                 ; 1064
             2F   AE          02   90 00059      MOVB    #2, C_DESC+3                  ; 1065
                         30   AE   D4 0005D      CLRL    C_DESC+4                      ; 1066
       00001388   8F     10   AC   D1 00060      CMPL    PRECISION, #5000              ; 1072
                              0B   14 00068      BGTR    1$                            ;
             14   AE          01   D0 0006A      MOVL    #1, RND_TRUNC                 ; 1075
             18   AE     10   AC   D0 0006E      MOVL    PRECISION, PREC               ; 1076
                              0D   11 00073      BRB     2$                            ; 1072
                         14   AE   D4 00075 1$:  CLRL    RND_TRUNC                     ; 1080
     18   AE     10   AC 00002710  8F   C3 00078 SUBL3   #10000, PRECISION, PREC       ; 1081
          50     14   AE     38   C3 00082 2$:  SUBL3   #56, RND_TRUNC, R0            ; 1089
                         50   50   CE 00087      MNEGL   R0, R0                        ;
 50      2C   AE         10   00   ED 0008A      CMPZV   #0, #16, C_DESC, R0           ;
                              0B   15 00090      BLEQ    3$                            ;
                 7E     00G   8F   9A 00092      MOVZBL  #BAS$K_FLOPOIERR, -(SP)       ; 1091
       00000000G  00          01   FB 00096      CALLS   #1, BAS$$SIGNAL               ;
                         2C   AE   9F 0009D 3$:  PUSHAB  C_DESC                        ; 1098
                         10   AE   9F 000A0      PUSHAB  C_EXP                         ;
                         18   AE   9F 000A3      PUSHAB  C_SIGN                        ;
                         20   AE   9F 000A6      PUSHAB  RND_TRUNC                     ;
                         28   AE   9F 000A9      PUSHAB  PREC                          ;
                         48   AE   9F 000AC      PUSHAB  B_DESC                        ;
                         34   AE   9F 000AF      PUSHAB  B_EXP                         ;
                         3C   AE   9F 000B2      PUSHAB  B_SIGN                        ;
                         5C   AE   9F 000B5      PUSHAB  A_DESC                        ;
                         48   AE   9F 000B8      PUSHAB  A_EXP                         ;
                         50   AE   9F 000BB      PUSHAB  A_SIGN                        ;
       00000000G  00          0B   FB 000BE      CALLS   #11, STR$DIVIDE               ;
             08   AE     2C   AE   3C 000C5      MOVZWL  C_DESC, SAV_C_LEN             ; 1116
             04   AE     30   AE   D0 000CA      MOVL    C_DESC+4, SAV_C_PTR           ; 1117
                 52      30   AE   D0 000CF      MOVL    C_DESC+4, BUF                 ; 1124
                 54      2C   AE   3C 000D3      MOVZWL  C_DESC, C_LENGTH              ; 1125
```

```
                    5A       52        54  C1 000D7         ADDL3   C_LENGTH, BUF, END_ADDR                      ; 1126
                             55    0C  AE  D0 000DB         MOVL    C_EXP, SAV_EXP                               ; 1127
                             6E        54  D0 000DF         MOVL    C_LENGTH, SAV_LENGTH                         ; 1128
                             59        52  D0 000E2         MOVL    BUF, SAV_BUF                                 ; 1129
                                   0C  AE  D5 000E5         TSTL    C_EXP                                        ; 1137
                                       03  19 000E8         BLSS    4$                                           ;
                                     0092  31 000EA         BRW     17$                                          ;
                             57    0C  AE  D0 000ED 4$:     MOVL    C_EXP, R7                                    ; 1140
                                       03  18 000F1         BGEQ    5$                                           ;
                             57        57  CE 000F3         MNEGL   R7, R7                                       ;
                             54        57  D1 000F6 5$:     CMPL    R7, C_LENGTH                                 ;
                                       0D  18 000F9         BGEQ    6$                                           ;
                             58        57  D0 000FB         MOVL    R7, LENGTH                                   ; 1143
                    50       54    0C  AE  C1 000FE         ADDL3   C_EXP, C_LENGTH, R0                          ; 1144
                             52        50  C0 00103         ADDL2   R0, BUF                                      ;
                                       03  11 00106         BRB     7$                                           ; 1140
                             58        54  D0 00108 6$:     MOVL    C_LENGTH, LENGTH                             ; 1147
  FA12   CF   F915  CF       62        58  2B 0010B 7$:     SPANC   LENGTH, (BUF), TABLE_NZ, MASK                ; 1153
                                       02  12 00114         BNEQ    8$                                           ;
                                       51  D4 00116         CLRL    R1                                           ;
                             56        51  D0 00118 8$:     MOVL    R1, NEW_ADDR                                 ;
                                       12  12 0011B         BNEQ    11$                                          ; 1154
                             54        57  D1 0011D         CMPL    R7, C_LENGTH                                 ; 1158
                                       05  19 00120         BLSS    9$                                           ;
                             54        01  D0 00122         MOVL    #1, C_LENGTH                                 ; 1161
                                       03  11 00125         BRB     10$                                          ; 1158
                             54        57  C2 00127 9$:     SUBL2   R7, C_LENGTH                                 ; 1166
                                   0C  AE  D4 0012A 10$:    CLRL    C_EXP                                        ; 1162
                                       4C  11 0012D         BRB     16$                                          ; 1169
                             5A        56  D1 0012F 11$:    CMPL    NEW_ADDR, END_ADDR                           ; 1185
                                       4B  13 00132         BEQL    17$                                          ;
                             5B        56  D0 00134 12$:    MOVL    NEW_ADDR, SAV_ADDR                           ; 1190
                    58       5A        56  C3 00137         SUBL3   NEW_ADDR, END_ADDR, LENGTH                   ; 1191
  F9E2   CF   F7E5  CF       66        58  2B 0013B         SPANC   LENGTH, (NEW_ADDR), TABLE_Z, MASK            ; 1192
                                       02  12 00144         BNEQ    13$                                          ;
                                       51  D4 00146         CLRL    R1                                           ;
                             56        51  D0 00148 13$:    MOVL    R1, NEW_ADDR                                 ;
                                       32  13 0014B         BEQL    17$                                          ; 1193
                             5B        56  D0 0014D         MOVL    NEW_ADDR, SAV_ADDR                           ; 1200
                    58       5A        56  C3 00150         SUBL3   NEW_ADDR, END_ADDR, LENGTH                   ; 1201
  F9C9   CF   F8CC  CF       66        58  2B 00154         SPANC   LENGTH, (NEW_ADDR), TABLE_NZ, MASK           ; 1202
                                       02  12 0015D         BNEQ    14$                                          ;
                                       51  D4 0015F         CLRL    R1                                           ;
                             56        51  D0 00161 14$:    MOVL    R1, NEW_ADDR                                 ;
                                       CE  12 00164         BNEQ    12$                                          ; 1203
                    57       5A        5B  C3 00166         SUBL3   SAV_ADDR, END_ADDR, DIFF                     ; 1206
                             54        57  C2 0016A         SUBL2   DIFF, C_LENGTH                               ; 1208
                             50    0C  AE  D0 0016D         MOVL    C_EXP, R0                                    ; 1209
                                       03  18 00171         BGEQ    15$                                          ;
                             50        50  CE 00173         MNEGL   R0, R0                                       ;
               0C   AE       57        50  C3 00176 15$:    SUBL3   R0, DIFF, C_EXP                              ;
                        2C   AE        54  B0 0017B 16$:    MOVW    C_LENGTH, C_DESC                             ; 1210
                                       55  D5 0017F 17$:    TSTL    SAV_EXP                                      ; 1220
                                       3D  14 00181         BGTR    22$                                          ;
                             50        55  D0 00183         MOVL    SAV_EXP, R0                                  ;
                                       03  18 00186         BGEQ    18$                                          ;
                             50        50  CE 00188         MNEGL   R0, R0                                       ;
```

```
                          6E          50  D1 0018B 18$:    CMPL    R0, SAV_LENGTH                                   ;
                                      30  18 0018E         BGEQ    22$                                              ;
                          50    0C    AE  D0 00190         MOVL    C_EXP, R0                                        ; 1232
                                      03  18 00194         BGEQ    19$                                              ;
                          50          50  CE 00196         MNEGL   R0, R0                                           ;
                    55    54          50  C3 00199 19$:    SUBL3   R0, C_LENGTH, LEFT_DIGITS                        ;
  F980    CF  F883  CF    69          55  2B 0019D         SPANC   LEFT_DIGITS, (SAV_BUF), TABLE_NZ, MASK           ; 1234
                                      02  12 001A6         BNEQ    20$                                              ;
                                      51  D4 001A8         CLRL    R1                                               ;
                          56          51  D0 001AA 20$:    MOVL    R1, NEW_ADDR                                     ;
                                      2F  12 001AD         BNEQ    24$                                              ; 1236
                          54          55  C2 001AF         SUBL2   LEFT_DIGITS, C_LENGTH                            ; 1239
                                      05  12 001B2         BNEQ    21$                                              ; 1240
                          54          01  D0 001B4         MOVL    #1, C_LENGTH                                     ; 1242
                                      31  11 001B7         BRB     25$                                              ;
              30    AE    59          55  C1 001B9 21$:    ADDL3   LEFT_DIGITS, SAV_BUF, C_DESC+4                   ; 1244
                                      2A  11 001BE         BRB     25$                                              ; 1236
                                      55  D5 001C0 22$:    TSTL    SAV_EXP                                          ; 1255
                                      2A  15 001C2         BLEQ    26$                                              ;
  F959    CF  F85C  CF    69          54  2B 001C4         SPANC   C_LENGTH, (SAV_BUF), TABLE_NZ, MASK              ; 1261
                                      02  12 001CD         BNEQ    23$                                              ;
                                      51  D4 001CF         CLRL    R1                                               ;
                          56          51  D0 001D1 23$:    MOVL    R1, NEW_ADDR                                     ;
                                      08  12 001D4         BNEQ    24$                                              ; 1262
                          54          01  D0 001D6         MOVL    #1, C_LENGTH                                     ; 1266
                                0C    AE  D4 001D9         CLRL    C_EXP                                            ; 1267
                                      0C  11 001DC         BRB     25$                                              ; 1262
                    57    56          59  C3 001DE 24$:    SUBL3   SAV_BUF, NEW_ADDR, DIFF                          ; 1271
                          54          57  C2 001E2         SUBL2   DIFF, C_LENGTH                                   ; 1272
              30    AE    59          57  C1 001E5         ADDL3   DIFF, SAV_BUF, C_DESC+4                          ; 1273
                    2C    AE          54  B0 001EA 25$:    MOVW    C_LENGTH, C_DESC                                 ; 1275
                                04    AC  DD 001EE 26$:    PUSHL   QUO_DESC                                         ; 1282
                                10    AE  DD 001F1         PUSHL   C_EXP                                            ;
                                18    AE  DD 001F4         PUSHL   C_SIGN                                           ;
                                38    AE  9F 001F7         PUSHAB  C_DESC                                           ;
                 0000V    CF          04  FB 001FA         CALLS   #4, PARSE_OUT                                    ;
                    2C    AE    08    AE  B0 001FF         MOVW    SAV_C_LEN, C_DESC                                ; 1288
                    30    AE    04    AE  D0 00204         MOVL    SAV_C_PTR, C_DESC+4                              ; 1289
                                3C    AE  9F 00209         PUSHAB  A_DESC                                           ; 1294
             00000000G    00          01  FB 0020C         CALLS   #1, STR$FREE1_DX                                 ;
                                34    AE  9F 00213         PUSHAB  B_DESC                                           ; 1295
             00000000G    00          01  FB 00216         CALLS   #1, STR$FREE1_DX                                 ;
                                2C    AE  9F 0021D         PUSHAB  C_DESC                                           ; 1296
             00000000G    00          01  FB 00220         CALLS   #1, STR$FREE1_DX                                 ;
                                          04 00227         RET                                                      ; 1298
                                        0000 00228 27$:    .WORD   Save nothing                                     ; 0948
                          50    08    AC  D0 0022A         MOVL    8(AP), R0                                        ;
                          50    04    A0  D0 0022E         MOVL    4(R0), R0                                        ;
                                E8    A0  9F 00232         PUSHAB  C_DESC                                           ;
                                F0    A0  9F 00235         PUSHAB  B_DESC                                           ;
                                F8    A0  9F 00238         PUSHAB  A_DESC                                           ;
                                      03  DD 0023B         PUSHL   #3                                               ;
                                      5E  DD 0023D         PUSHL   SP                                               ;
                          7E    04    AC  7D 0023F         MOVQ    4(AP), -(SP)                                     ;
                 0000V    CF          03  FB 00243         CALLS   #3, FREE_STRINGS                                 ;
                                          04 00248         RET                                                      ;
```

```
; Routine Size:  585 bytes,    Routine Base:  _BAS$CODE + 06DA

; 1211          1299  1
```

```
GLOBAL ROUTINE BAS$SUM (                           ! Add strings
        SUM_DESC,                                  ! Sum
        OP1_DESC,                                  ! First input
        OP2_DESC                                   ! Second input
   ) : NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Add two strings.  SUM := OP1 + OP2
!
! FORMAL PARAMETERS:
!
!       SUM_DESC.wt.dx  The sum of OP1 and OP2
!       OP1_DESC.rt.dx  Operand OP1
!       OP2_DESC.rt.dx  Operand OP2
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       NONE
!
! SIDE EFFECTS:
!
!       Signals PROSOSSOR for certain "impossible" errors, and the
!       signals from PARSE_IN and PARSE_OUT.
!
!--

    BEGIN

    MAP
        OP1_DESC : REF BLOCK [8, BYTE],
        OP2_DESC : REF BLOCK [8, BYTE],
        SUM_DESC : REF BLOCK [8, BYTE];

    LOCAL
!+
! Internal form of OP1.
!-
        A_DESC : BLOCK [8, BYTE] VOLATILE,
        A_SIGN,
        A_EXP,
!+
! Internal form of OP2.
!-
        B_DESC : BLOCK [8, BYTE] VOLATILE,
        B_SIGN,
        B_EXP,
```

```
2 !+
2 ! Internal form of the sum, OP1 + OP2.
2 !-
2         C_DESC : BLOCK [8, BYTE] VOLATILE,
2         C_SIGN,
2         C_EXP;
2
2 !+
2 ! Enable a handler to free the local strings in case of error.
2 !-
2
2     ENABLE
2         FREE_STRINGS (A_DESC, B_DESC, C_DESC);
2
2 !+
2 ! Convert the two input arguments from external form to internal form.
2 ! This is done by removing the non-digits from the string and
2 ! returning the sign and exponent as separate values.
2 ! Errors are signaled.
2 !-
2     A_DESC [DSC$W_LENGTH] = 0;
2     A_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2     A_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2     A_DESC [DSC$A_POINTER] = 0;
2     PARSE_IN (.OP1_DESC, A_DESC, A_SIGN, A_EXP);
2     B_DESC [DSC$W_LENGTH] = 0;
2     B_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2     B_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2     B_DESC [DSC$A_POINTER] = 0;
2     PARSE_IN (.OP2_DESC, B_DESC, B_SIGN, B_EXP);
2 !+
2 ! Add the numbers using the large-precision string arithmetic package.
2 !-
2     C_DESC [DSC$W_LENGTH] = 0;
2     C_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2     C_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2     C_DESC [DSC$A_POINTER] = 0;
2     STR$ADD (A_SIGN, A_EXP, A_DESC, B_SIGN, B_EXP, B_DESC, C_SIGN, C_EXP, C_DESC);
2 !+
2 ! We are done with A and B, so free them.
2 !-
2     STR$FREE1_DX (A_DESC);
2     STR$FREE1_DX (B_DESC);
2 !+
2 ! Convert the sum to external form for the caller.
2 !-
2     PARSE_OUT (C_DESC, .C_SIGN, .C_EXP, .SUM_DESC);
2 !+
2 ! We can now free C
2 !-
2     STR$FREE1_DX (C_DESC);
2     RETURN;
1     END;                                        ! end of BAS$SUM
```

```
                              0004 00000        .ENTRY  BAS$SUM, Save R2                      ; 1300
        52 00000000G     00    9E 00002        MOVAB   STR$FREE1_DX, R2                      ;
        5E               30    C2 00009        SUBL2   #48, SP                               ;
                   18    AE    7C 0000C        CLRQ    C_DESC                                ; 1337
                   20    AE    7C 0000F        CLRQ    B_DESC                                ;
                   28    AE    7C 00012        CLRQ    A_DESC                                ;
        6D       0093    CF    DE 00015        MOVAL   1$, (FP)                              ;
                   28    AE    B4 0001A        CLRW    A_DESC                                ; 1377
    2A    AE             0F    90 0001D        MOVB    #15, A_DESC+2                         ; 1378
    2B    AE             02    90 00021        MOVB    #2, A_DESC+3                          ; 1379
                   2C    AE    D4 00025        CLRL    A_DESC+4                              ; 1380
                   10    AE    9F 00028        PUSHAB  A_EXP                                 ; 1381
                   18    AE    9F 0002B        PUSHAB  A_SIGN                                ;
                   30    AE    9F 0002E        PUSHAB  A_DESC                                ;
                   08    AC    DD 00031        PUSHL   OP1_DESC                              ;
 0000V    CF             04    FB 00034        CALLS   #4, PARSE_IN                          ;
                   20    AE    B4 00039        CLRW    B_DESC                                ; 1382
    22    AE             0F    90 0003C        MOVB    #15, B_DESC+2                         ; 1383
    23    AE             02    90 00040        MOVB    #2, B_DESC+3                          ; 1384
                   24    AE    D4 00044        CLRL    B_DESC+4                              ; 1385
                   08    AE    9F 00047        PUSHAB  B_EXP                                 ; 1386
                   10    AE    9F 0004A        PUSHAB  B_SIGN                                ;
                   28    AE    9F 0004D        PUSHAB  B_DESC                                ;
                   0C    AC    DD 00050        PUSHL   OP2_DESC                              ;
 0000V    CF             04    FB 00053        CALLS   #4, PARSE_IN                          ;
                   18    AE    B4 00058        CLRW    C_DESC                                ; 1390
    1A    AE             0F    90 0005B        MOVB    #15, C_DESC+2                         ; 1391
    1B    AE             02    90 0005F        MOVB    #2, C_DESC+3                          ; 1392
                   1C    AE    D4 00063        CLRL    C_DESC+4                              ; 1393
                   18    AE    9F 00066        PUSHAB  C_DESC                                ; 1394
                   04    AE    9F 00069        PUSHAB  C_EXP                                 ;
                   0C    AE    9F 0006C        PUSHAB  C_SIGN                                ;
                   2C    AE    9F 0006F        PUSHAB  B_DESC                                ;
                   18    AE    9F 00072        PUSHAB  B_EXP                                 ;
                   20    AE    9F 00075        PUSHAB  B_SIGN                                ;
                   40    AE    9F 00078        PUSHAB  A_DESC                                ;
                   2C    AE    9F 0007B        PUSHAB  A_EXP                                 ;
                   34    AE    9F 0007E        PUSHAB  A_SIGN                                ;
00000000G  00            09    FB 00081        CALLS   #9, STR$ADD                           ;
                   28    AE    9F 00088        PUSHAB  A_DESC                                ; 1398
          62             01    FB 0008B        CALLS   #1, STR$FREE1_DX                      ;
                   20    AE    9F 0008E        PUSHAB  B_DESC                                ; 1399
          62             01    FB 00091        CALLS   #1, STR$FREE1_DX                      ;
                   04    AC    DD 00094        PUSHL   SUM_DESC                              ; 1403
                   04    AE    DD 00097        PUSHL   C_EXP                                 ;
                   0C    AE    DD 0009A        PUSHL   C_SIGN                                ;
                   24    AE    9F 0009D        PUSHAB  C_DESC                                ;
 0000V    CF             04    FB 000A0        CALLS   #4, PARSE_OUT                         ;
                   18    AE    9F 000A5        PUSHAB  C_DESC                                ; 1407
          62             01    FB 000A8        CALLS   #1, STR$FREE1_DX                      ;
                               04 000AB        RET                                           ; 1409
                             0000 000AC 1$:    .WORD   Save nothing                          ; 1337
          50       08    AC    D0 000AE        MOVL    8(AP), R0                             ;
          50       04    A0    D0 000B2        MOVL    4(R0), R0                             ;
                   E8    A0    9F 000B6        PUSHAB  C_DESC                                ;
                   F0    A0    9F 000B9        PUSHAB  B_DESC                                ;
```

```
                 F8  AO  9F 000BC        PUSHAB  A_DESC                 ;
                         03  DD 000BF     PUSHL   #3                     ;
                         5E  DD 000C1     PUSHL   SP                     ;
         7E        04    AC  7D 000C3     MOVQ    4(AP), -(SP)           ;
   0000V CF              03  FB 000C7     CALLS   #3, FREE_STRINGS       ;
                             04 000CC     RET                            ;

; Routine Size:  205 bytes,    Routine Base:  _BAS$CODE + 0923

; 1323       1410  1
```

```
1 ROUTINE PARSE_IN (                                   ! Scan a number and divide it up
1       ARG_DESC,                                      ! The number to scan
1       DIGITS,                                        ! Where to put the digits
1       SIGN,                                          ! Where to put the sign
1       EXPONENT                                       ! Where to put the power of ten
1    ) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Convert a numeric string of the form +nnn.nnn into a digit
1 !       string and a separate sign and decimal exponent.  Don't allow
1 !       more than 60 digits, for compatability with the PDP-11.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       ARG_DESC.rt.dx  The number to parse
1 !       DIGITS.wnu.dx   Where to put the digits found
1 !       SIGN.wl.r       0 = positive, 1 = negative
1 !       EXPONENT.wl.r   Decimal exponent
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Signals BAS$_ILLNUM if there are more than 60 digit,
1 !       BAS$_DATFORERR if the syntax of the number is wrong,
1 !       and also signals if storage is exhausted.
1 !
1 !--
1
2    BEGIN
2
2    MAP
2       ARG_DESC : REF BLOCK [8, BYTE],
2       DIGITS : REF BLOCK [8, BYTE];
2
2    LOCAL
2 !+
2 ! The following three locals hold the internal form of the number.
2 !-
2       SIGN_VAL,
2       BUF_DESC : BLOCK [8, BYTE] VOLATILE,
2       EXPON,
2       SIGN_SEEN,                                  ! 1 = we have scanned a + or -
2       DIGIT_SEEN,                                 ! 1 = we have seen at least one digit
```

```
        DOT_SEEN,                                  ! 1 = we have seen a decimal point
        BLANKS_SEEN,                               ! 1 = we have seen trailing blanks
        PUTTER,                                    ! Counts position in the output buffer
        BUF : REF VECTOR [65535, BYTE],            ! Addresses result
        ARG : REF VECTOR [65535, BYTE],            ! Addresses source
        ARG_LEN;                                   ! Length of the source

!+
! Enable a handler to free the local string in case of an error.
!-

    ENABLE
        FREE_STRINGS (BUF_DESC);

!+
! 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.
! Note that we must fetch the length field of the descriptor only
! once, so that we will not attempt to overrun our local string if
! the source is reallocated longer by an AST.
!-
    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;
    ARG_LEN = .ARG_DESC [DSC$W_LENGTH];
    STR$GET1_DX (%REF (.ARG_LEN + 8), BUF_DESC);
!+
! Now scan the incoming number.
!-
    PUTTER = 0;
    BUF = .BUF_DESC [DSC$A_POINTER];
    ARG = .ARG_DESC [DSC$A_POINTER];
    SIGN_SEEN = 0;
    SIGN_VAL = 0;
    DIGIT_SEEN = 0;
    DOT_SEEN = 0;
    BLANKS_SEEN = 0;
    EXPON = 0;

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

        SELECTONE .ARG [.GETTER] OF
            SET

            [%C'+'] :
                BEGIN                              ! Plus sign

                IF (.SIGN_SEEN OR .DIGIT_SEEN OR .DOT_SEEN OR .BLANKS_SEEN) THEN BAS$$STOP (BAS$K_DATFORERR)

                SIGN_SEEN = 1;
                SIGN_VAL = 0;
                END;

            [%C'-'] :
                BEGIN                              ! Minus sign
```

```
3
3               IF (.SIGN_SEEN OR .DIGIT_SEEN OR .DOT_SEEN OR .BLANKS_SEEN) THEN BAS$$STOP (BAS$K_DATFORERR)
3
3               SIGN_SEEN = 1;
3               SIGN_VAL = 1;
2               END;
2
2           [%C'.'] :
3               BEGIN                           ! Decimal point
3
3               IF (.DOT_SEEN OR .BLANKS_SEEN) THEN BAS$$STOP (BAS$K_DATFORERR);
3
3               DOT_SEEN = 1;
2               END;
2
2           [%C' '] :
3               BEGIN                           ! Blank, better be leading or trailing.
3
3               IF (.SIGN_SEEN OR .DIGIT_SEEN OR .DOT_SEEN) THEN BLANKS_SEEN = 1;
3
2               END;
2
2           [%C'0' TO %C'9'] :
3               BEGIN                           ! Decimal digit
3
3               IF (.BLANKS_SEEN) THEN BAS$$STOP (BAS$K_DATFORERR);
3
3 !             IF (.DIGIT_SEEN OR .DOT_SEEN OR (.SIGN_VAL EQL 1) OR
3 !                 (.ARG [.GETTER] NEQ %C'0'))
3 !             THEN
4                   BEGIN
4 !+
4 ! This is not a leading zero
4 !-
4                   DIGIT_SEEN = 1;
4                   BUF [.PUTTER] = .ARG [.GETTER];
4                   PUTTER = .PUTTER + 1;
4
4                   IF (.DOT_SEEN) THEN EXPON = .EXPON - 1;
4
3                   END;
3
2               END;
2
2           [OTHERWISE] :
2               BAS$$STOP (BAS$K_DATFORERR);
2           TES;
2
2 !+
2 ! This is the end of the INCR loop.
2 !-
2
3       IF ( NOT .DIGIT_SEEN)
2       THEN
3           BEGIN
3 !+
3 ! If there are no digits, or only leading zeros, take the number to
```

```
; 1496       1582 3 ! be zero.  Don't be too gullible, however.
; 1497       1583 3 !-
; 1498       1584 3
; 1499       1585 3         IF (.SIGN_SEEN OR .DOT_SEEN OR .BLANKS_SEEN) THEN BAS$$STOP (BAS$K_DATFORERR);
; 1500       1586 3
; 1501       1587 3         BUF [.PUTTER] = %C'0';
; 1502       1588 3         PUTTER = .PUTTER + 1;
; 1503       1589 2         END;
; 1504       1590 2
; 1505       1591 2 !+
; 1506       1592 2 ! If there are more than 60 digits in the number, reject it for compatabliity
; 1507       1593 2 ! with the PDP-11.
; 1508       1594 2 !-
; 1509       1595 2
; 1510       1596 2     IF (.PUTTER GTR BAS$K_PREC_LIM3) THEN BAS$$STOP (BAS$K_ILLNUM);
; 1511       1597 2
; 1512       1598 2 !+
; 1513       1599 2 ! If we make it to here the number is in proper form.
; 1514       1600 2 ! Return it to the caller.
; 1515       1601 2 !-
; 1516       1602 2     .SIGN = .SIGN_VAL;
; 1517       1603 2     .EXPONENT = .EXPON;
; 1518       1604 2     STR$COPY_R (.DIGITS, PUTTER, .BUF);
; 1519       1605 2 !+
; 1520       1606 2 ! Free our local string
; 1521       1607 2 !-
; 1522       1608 2     STR$FREE1_DX (BUF_DESC);
; 1523       1609 1     END;                                        ! end of PARSE_IN
```

```
                                 0FFC 00000 PARSE_IN:
                                                      .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11       ; 1411
              5E           10  C2 00002              SUBL2   #16, SP                                    :
                   08      AE  7C 00005              CLRQ    BUF_DESC                                   ; 1453
              6D   0148    CF  DE 00008              MOVAL   24$, (FP)                                  :
                   08      AE  B4 0000D              CLRW    BUF_DESC                                   ; 1490
        0A    AE           0F  90 00010              MOVB    #15, BUF_DESC+2                            ; 1491
        0B    AE           02  90 00014              MOVB    #2, BUF_DESC+3                             ; 1492
                   0C      AE  D4 00018              CLRL    BUF_DESC+4                                 ; 1493
              52   04      AC  D0 0001B              MOVL    ARG_DESC, R2                               ; 1494
              5A           62  3C 0001F              MOVZWL  (R2), ARG_LEN                              :
                   08      AE  9F 00022              PUSHAB  BUF_DESC                                   ; 1495
        04    AE   08      AA  9E 00025              MOVAB   8(R10), 4(SP)                              :
                   04      AE  9F 0002A              PUSHAB  4(SP)                                      :
   00000000G  00           02  FB 0002D              CALLS   #2, STR$GET1_DX                            :
              53   0C      AE  D0 00034              MOVL    BUF_DESC+4, BUF                            ; 1500
              55   04      A2  D0 00038              MOVL    4(R2), ARG                                 ; 1501
                           5B  D4 0003C              CLRL    SIGN_VAL                                   ; 1503
                           57  7C 0003E              CLRQ    SIGN_SEEN                                  ; 1502
                           54  D4 00040              CLRL    DOT_SEEN                                   ; 1505
                           56  D4 00042              CLRL    BLANKS_SEEN                                ; 1506
                           6E  7C 00044              CLRQ    EXPON                                      ; 1507
              59           01  CE 00046              MNEGL   #1, GETTER                                 ; 1511
                           77  11 00049              BRB     12$                                        :
```

```
           52        6945 9A 0004B 1$:   MOVZBL  (GETTER)[ARG], R2                  ;
           2B          52 91 0004F       CMPB    R2, #43                            ; 1514
                       1E 12 00052       BNEQ    4$                                 ;
           09          57 E8 00054       BLBS    SIGN_SEEN, 2$                      ; 1517
           06          58 E8 00057       BLBS    DIGIT_SEEN, 2$                     ;
           03          54 E8 0005A       BLBS    DOT_SEEN, 2$                       ;
           0B          56 E9 0005D       BLBC    BLANKS_SEEN, 3$                    ;
           7E   00G    8F 9A 00060 2$:   MOVZBL  #BAS$K_DATFORERR, -(SP)            ;
 00000000G 00          01 FB 00064       CALLS   #1, BAS$$STOP                      ;
           57          01 D0 0006B 3$:   MOVL    #1, SIGN_SEEN                      ; 1519
           5B             D4 0006E       CLRL    SIGN_VAL                           ; 1520
                       7A 11 00070       BRB     15$                                ; 1511
           2D          52 91 00072 4$:   CMPB    R2, #45                            ; 1523
                       1F 12 00075       BNEQ    7$                                 ;
           09          57 E8 00077       BLBS    SIGN_SEEN, 5$                      ; 1526
           06          58 E8 0007A       BLBS    DIGIT_SEEN, 5$                     ;
           03          54 E8 0007D       BLBS    DOT_SEEN, 5$                       ;
           0B          56 E9 00080       BLBC    BLANKS_SEEN, 6$                    ;
           7E   00G    8F 9A 00083 5$:   MOVZBL  #BAS$K_DATFORERR, -(SP)            ;
 00000000G 00          01 FB 00087       CALLS   #1, BAS$$STOP                      ;
           57          01 D0 0008E 6$:   MOVL    #1, SIGN_SEEN                      ; 1528
           5B          01 D0 00091       MOVL    #1, SIGN_VAL                       ; 1529
                       63 11 00094       BRB     17$                                ; 1511
           2E          52 91 00096 7$:   CMPB    R2, #46                            ; 1532
                       16 12 00099       BNEQ    10$                                ;
           03          54 E8 0009B       BLBS    DOT_SEEN, 8$                       ; 1535
           0B          56 E9 0009E       BLBC    BLANKS_SEEN, 9$                    ;
           7E   00G    8F 9A 000A1 8$:   MOVZBL  #BAS$K_DATFORERR, -(SP)            ;
 00000000G 00          01 FB 000A5       CALLS   #1, BAS$$STOP                      ;
           54          01 D0 000AC 9$:   MOVL    #1, DOT_SEEN                       ; 1537
                       48 11 000AF       BRB     17$                                ; 1511
           20          52 91 000B1 10$:  CMPB    R2, #32                            ; 1540
                       0E 12 000B4       BNEQ    13$                                ;
           06          57 E8 000B6       BLBS    SIGN_SEEN, 11$                     ; 1543
           03          58 E8 000B9       BLBS    DIGIT_SEEN, 11$                    ;
           3A          54 E9 000BC       BLBC    DOT_SEEN, 17$                      ;
           56          01 D0 000BF 11$:  MOVL    #1, BLANKS_SEEN                    ;
                       35 11 000C2 12$:  BRB     17$                                ; 1511
           30          52 91 000C4 13$:  CMPB    R2, #48                            ; 1547
                       25 1F 000C7       BLSSU   16$                                ;
           39          52 91 000C9       CMPB    R2, #57                            ;
                       20 1A 000CC       BGTRU   16$                                ;
           0B          56 E9 000CE       BLBC    BLANKS_SEEN, 14$                   ; 1550
           7E   00G    8F 9A 000D1       MOVZBL  #BAS$K_DATFORERR, -(SP)            ;
 00000000G 00          01 FB 000D5       CALLS   #1, BAS$$STOP                      ;
           58          01 D0 000DC 14$:  MOVL    #1, DIGIT_SEEN                     ; 1559
      04 BE43          52 90 000DF       MOVB    R2, @PUTTER[BUF]                   ; 1560
                04     AE D6 000E4       INCL    PUTTER                             ; 1561
           0F          54 E9 000E7       BLBC    DOT_SEEN, 17$                      ; 1563
                       6E D7 000EA       DECL    EXPON                              ;
                       0B 11 000EC 15$:  BRB     17$                                ; 1511
           7E   00G    8F 9A 000EE 16$:  MOVZBL  #BAS$K_DATFORERR, -(SP)            ; 1570
 00000000G 00          01 FB 000F2       CALLS   #1, BAS$$STOP                      ;
    02     59          5A F2 000F9 17$:  AOBLSS  ARG_LEN, GETTER, 18$               ; 1511
                       03 11 000FD       BRB     19$                                ;
                     FF49 31 000FF 18$:  BRW     1$                                 ;
           1C          58 E8 00102 19$:  BLBS    DIGIT_SEEN, 22$                    ; 1577
```

```
               06          57  E8 00105          BLBS    SIGN_SEEN, 20$                        ; 1585
               03          54  E8 00108          BLBS    DOT_SEEN, 20$                         ;
               0B          56  E9 0010B          BLBC    BLANKS_SEEN, 21$                      ;
               7E     00G  8F  9A 0010E  20$:    MOVZBL  #BAS$K_DATFORERR, -(SP)               ;
       00000000G  00       01  FB 00112          CALLS   #1, BAS$$STOP                         ;
           04 BE43         30  90 00119  21$:    MOVB    #48, @PUTTER[BUF]                     ; 1587
                      04   AE  D6 0011E          INCL    PUTTER                                ; 1588
               3C     04   AE  D1 00121  22$:    CMPL    PUTTER, #60                           ; 1596
                           0B  15 00125          BLEQ    23$                                   ;
               7E     00G  8F  9A 00127          MOVZBL  #BAS$K_ILLNUM, -(SP)                  ;
       00000000G  00       01  FB 0012B          CALLS   #1, BAS$$STOP                         ;
              0C  BC       5B  D0 00132  23$:    MOVL    SIGN_VAL, @SIGN                       ; 1602
              10  BC       6E  D0 00136          MOVL    EXPON, @EXPONENT                      ; 1603
                           53  DD 0013A          PUSHL   BUF                                   ; 1604
                      08   AE  9F 0013C          PUSHAB  PUTTER                                ;
                      08   AC  DD 0013F          PUSHL   DIGITS                                ;
       00000000G  00       03  FB 00142          CALLS   #3, STR$COPY_R                        ;
                      08   AE  9F 00149          PUSHAB  BUF_DESC                              ; 1608
       00000000G  00       01  FB 0014C          CALLS   #1, STR$FREE1_DX                      ;
                           04 00153              RET                                           ; 1609
                         0000 00154      24$:    .WORD   Save nothing                          ; 1453
               50     08   AC  D0 00156          MOVL    8(AP), R0                             ;
               50     04   A0  D0 0015A          MOVL    4(R0), R0                             ;
                      F8   A0  9F 0015E          PUSHAB  BUF_DESC                              ;
                           01  DD 00161          PUSHL   #1                                    ;
                           5E  DD 00163          PUSHL   SP                                    ;
               7E     04   AC  7D 00165          MOVQ    4(AP), -(SP)                          ;
           0000V  CF       03  FB 00169          CALLS   #3, FREE_STRINGS                      ;
                           04 0016E              RET                                           ;

; Routine Size:  367 bytes,    Routine Base:  _BAS$CODE + 09F0
```

```
ROUTINE PARSE_OUT (                            ! Put together a number
        DIGITS,                                ! Where to find the digits
        SIGN,                                  ! Where to find the sign
        EXPONENT,                              ! Where to find the power of ten
        ARG_DESC                               ! Where to put the result
    ) : NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Convert an internal number, which has separate sign, exponent
!       and digit string, into an external number, which combines them
!       in the form +nnn.nnn.
!
! FORMAL PARAMETERS:
!
!       DIGITS.rnu.dx   Where to find the digits
!       SIGN.rl.r       0 = positive, 1 = negative
!       EXPONENT.wl.r   Decimal exponent
!       ARG_DESC.wt.dx  Where to put the composite number
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       Signals PROLOSSOR for certain "impossible" errors, and MAXMEMEXC
!       if string memory is exhausted.
!
! SIDE EFFECTS:
!
!       Signals BAS$_ILLNUM if there are more than 65,535 digits,
!       BAS$_MAXMEMEXC if space for the string is unobtainable,
!       and BAS$_PROLOSSOR if we are unable to free the string.
!
!--

    BEGIN

    MAP
        ARG_DESC : REF BLOCK [8, BYTE],
        DIGITS : REF BLOCK [8, BYTE];

    LOCAL
        BUF_DESC : BLOCK [8, BYTE] VOLATILE,   ! Describes result
        PUTTER,                                ! Counts positions in BUF (below)
        BUF : REF VECTOR [65535, BYTE],        ! Addresses result string
        DIGIT_LEN,                             ! Length of input string
        DIGIT_BUF : REF VECTOR [65535, BYTE],  ! Addresses input string
        HIGH_POS,                              ! Max power of ten in result
        LOW_POS;                               ! Min power of ten in result
```

```
2   !+
2   ! Enable a handler to free the local string in case of error.
2   !-
2
2       ENABLE
2           FREE_STRINGS (BUF_DESC);
2
2   !+
2   ! Allocate enough space to hold the digits, a leading sign, and an
2   ! imbedded (or trailing) decimal point.  We must allocate enough
2   ! space to reach the units position in order to be able to place
2   ! the decimal point.
2   !-
2       DIGIT_LEN = .DIGITS [DSC$W_LENGTH];
2       DIGIT_BUF = .DIGITS [DSC$A_POINTER];
2       HIGH_POS = MAX (.DIGIT_LEN + .EXPONENT - 1, -1);
2       LOW_POS = MIN (.EXPONENT, 0);
2       BUF_DESC [DSC$W_LENGTH] = 0;
2       BUF_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
2       BUF_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2       BUF_DESC [DSC$A_POINTER] = 0;
2   !+
2   ! If the resultant number has too many digits to be represented on
2   ! VAX, give an error message.
2   !-
2
2       IF ((.HIGH_POS - .LOW_POS + 3) GTR 65535) THEN BAS$$STOP (BAS$K_ILLNUM);
2
2       STR$GET1_DX (%REF (.HIGH_POS - .LOW_POS + 3), BUF_DESC);
2       BUF = .BUF_DESC [DSC$A_POINTER];
2       PUTTER = 0;
2
3       IF (.SIGN)
2       THEN
3           BEGIN
3           BUF [.PUTTER] = %C'-';
3           PUTTER = .PUTTER + 1;
2           END;
2
2       DECR POS FROM .HIGH_POS TO .LOW_POS DO
3           BEGIN
3
4           IF (.POS EQL -1)
3           THEN
4               BEGIN
4               BUF [.PUTTER] = %C'.';
4               PUTTER = .PUTTER + 1;
3               END;
3
4           IF ((.POS GTR (.EXPONENT + .DIGIT_LEN - 1)) OR (.POS LSS .EXPONENT))
3           THEN
3               BUF [.PUTTER] = %C'0'
3           ELSE
3               BUF [.PUTTER] = .DIGIT_BUF [(.EXPONENT + .DIGIT_LEN - 1) - .POS];
3
3           PUTTER = .PUTTER + 1;
```

```
; 1639         1724  2          END;
; 1640         1725  2
; 1641         1726  2  !+
; 1642         1727  2  ! Now copy the string back to the caller.
; 1643         1728  2  !-
; 1644         1729  2      STR$COPY_R (.ARG_DESC, PUTTER, .BUF);
; 1645         1730  2  !+
; 1646         1731  2  ! Free our local string
; 1647         1732  2  !-
; 1648         1733  2      STR$FREE1_DX (BUF_DESC);
; 1649         1734  2      RETURN;
; 1650         1735  1      END;                                    ! end of PARSE_OUT



                                       007C 00000 PARSE_OUT:
                                                          .WORD    Save R2,R3,R4,R5,R6                    ; 1610
               5E            10  C2 00002                 SUBL2    #16, SP
                       08    AE  7C 00005                 CLRQ     BUF_DESC                               ; 1653
               6D    00DB    CF  DE 00008                 MOVAL    11$, (FP)                              ;
               50      04    AC  D0 0000D                 MOVL     DIGITS, R0                             ; 1681
               53            60  3C 00011                 MOVZWL   (R0), DIGIT_LEN
               55      04    A0  D0 00014                 MOVL     4(R0), DIGIT_BUF                       ; 1682
   50          53      0C    AC  C1 00018                 ADDL3    EXPONENT, DIGIT_LEN, R0                ; 1683
                             50  D7 0001D                 DECL     R0                                     ;
     FFFFFFFF  8F            50  D1 0001F                 CMPL     R0, #-1                                ;
                             03  18 00026                 BGEQ     1$                                     ;
               50            01  CE 00028                 MNEGL    #1, R0                                 ;
               52            50  D0 0002B 1$:             MOVL     R0, HIGH_POS                           ;
               50      0C    AC  D0 0002E                 MOVL     EXPONENT, R0                           ; 1684
                             02  15 00032                 BLEQ     2$                                     ;
                             50  D4 00034                 CLRL     R0                                     ;
               56            50  D0 00036 2$:             MOVL     R0, LOW_POS                            ;
                       08    AE  B4 00039                 CLRW     BUF_DESC                               ; 1685
          0A   AE            0E  90 0003C                 MOVB     #14, BUF_DESC+2                        ; 1686
          0B   AE            02  90 00040                 MOVB     #2, BUF_DESC+3                         ; 1687
                       0C    AE  D4 00044                 CLRL     BUF_DESC+4                             ; 1688
   51          52            56  C3 00047                 SUBL3    LOW_POS, HIGH_POS, R1                  ; 1694
               54      03    A1  9E 0004B                 MOVAB    3(R1), R4                              ;
     0000FFFF  8F            54  D1 0004F                 CMPL     R4, #65535                             ;
                             0B  15 00056                 BLEQ     3$                                     ;
               7E     00G    8F  9A 00058                 MOVZBL   #BAS$K_ILLNUM, -(SP)                   ;
    00000000G  00            01  FB 0005C                 CALLS    #1, BAS$$STOP                          ;
                       08    AE  9F 00063 3$:             PUSHAB   BUF_DESC                               ; 1696
          04   AE            54  D0 00066                 MOVL     R4, 4(SP)                              ;
                       04    AE  9F 0006A                 PUSHAB   4(SP)                                  ;
    00000000G  00            02  FB 0006D                 CALLS    #2, STR$GET1_DX                        ;
               50      0C    AE  D0 00074                 MOVL     BUF_DESC+4, BUF                        ; 1697
                       04    AE  D4 00078                 CLRL     PUTTER                                 ; 1698
               08      08    AC  E9 0007B                 BLBC     SIGN, 4$                               ; 1700
       04 BE40               2D  90 0007F                 MOVB     #45, @PUTTER[BUF]                      ; 1703
                       04    AE  D6 00084                 INCL     PUTTER                                 ; 1704
   51          53      0C    AC  C1 00087 4$:             ADDL3    EXPONENT, DIGIT_LEN, R1                ; 1717
               53      FF    A1  9E 0008C                 MOVAB    -1(R1), R3                             ;
               51            52  D0 00090                 MOVL     HIGH_POS, POS                          ; 1719
```

```
                                  33  11 00093       BRB     10$
        FFFFFFFF  8F              51  D1 00095 5$:   CMPL    POS, #-1                ; 1710
                                  08  12 0009C       BNEQ    6$
              04 BE40             2E  90 0009E       MOVB    #46, @PUTTER[BUF]       ; 1713
                           04     AE  D6 000A3       INCL    PUTTER                  ; 1714
    52            50       04     AE  C1 000A6 6$:   ADDL3   PUTTER, BUF, R2         ; 1719
                  53              51  D1 000AB       CMPL    POS, R3                 ; 1717
                                  06  14 000AE       BGTR    7$
              0C  AC              51  D1 000B0       CMPL    POS, EXPONENT
                                  05  18 000B4       BGEQ    8$
                  62              30  90 000B6 7$:   MOVB    #48, (R2)               ; 1719
                                  08  11 000B9       BRB     9$
    54            53              51  C3 000BB 8$:   SUBL3   POS, R3, R4             ; 1721
                  62            6445  90 000BF       MOVB    (R4)[DIGIT_BUF], (R2)
                           04     AE  D6 000C3 9$:   INCL    PUTTER                  ; 1723
                                  51  D7 000C6       DECL    POS                     ; 1707
                  56              51  D1 000C8 10$:  CMPL    POS, LOW_POS
                                  C8  18 000CB       BGEQ    5$
                                  50  DD 000CD       PUSHL   BUF                     ; 1729
                           08     AE  9F 000CF       PUSHAB  PUTTER
                           10     AC  DD 000D2       PUSHL   ARG_DESC
        00000000G 00              03  FB 000D5       CALLS   #3, STR$COPY_R
                           08     AE  9F 000DC       PUSHAB  BUF_DESC                ; 1733
        00000000G 00              01  FB 000DF       CALLS   #1, STR$FREE1_DX
                                      04 000E6       RET                             ; 1735
                                    0000 000E7 11$:  .WORD   Save nothing            ; 1653
                  50       08     AC  D0 000E9       MOVL    8(AP), R0
                  50       04     A0  D0 000ED       MOVL    4(R0), R0
                           F8     A0  9F 000F1       PUSHAB  BUF_DESC
                                  01  DD 000F4       PUSHL   #1
                                  5E  DD 000F6       PUSHL   SP
                  7E       04     AC  7D 000F8       MOVQ    4(AP), -(SP)
            0000V CF              03  FB 000FC       CALLS   #3, FREE_STRINGS
                                      04 00101       RET

; Routine Size:  258 bytes,    Routine Base:  _BAS$CODE + 0B5F
```

```
ROUTINE FREE_STRINGS (                          ! Free local strings
        SIG,                                    ! Signal vector
        MECH,                                   ! Mechanism vector
        ENBL                                    ! Enable vector
    ) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       If we are unwinding, free the local strings.  They are passed
!       in the enable vector.
!
! FORMAL PARAMETERS:
!
!       SIG.rl.a        A counted vector of parameters to LIB$SIGNAL/STOP
!       MECH.rl.a       A counted vector of info from CHF
!       ENBL.ra.a       A counted vector of ENABLE argument addresses.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       Always SS$_RESIGNAL, which is ignored when unwinding.
!
! SIDE EFFECTS:
!
!       Frees all of the strings passed as enable arguments.
!
!--

    BEGIN

    MAP
        SIG : REF VECTOR,
        MECH : REF VECTOR,
        ENBL : REF VECTOR;

!+
! Only free the strings if this is the UNWIND condition.
!-

    IF ( NOT (LIB$MATCH_COND (SIG [1], %REF (SS$_UNWIND)))) THEN RETURN (SS$_RESIGNAL);

!+
! Go through the enable arguments, freeing them.
!-

    INCR ARG_NO FROM 1 TO .ENBL [0] DO

        IF (..ENBL [.ARG_NO] NEQ 0) THEN STR$FREE1_DX (.ENBL [.ARG_NO]);
```

```
; 1709          1793  2
; 1710          1794  2      RETURN (SS$_RESIGNAL);
; 1711          1795  1      END;                                                ! end of FREE_STRINGS



                                                    0004 00000 FREE_STRINGS:
                                                                          .WORD   Save R2                          ; 1736
                               7E      0920   8F   3C 00002              MOVZWL  #2336, -(SP)                     ; 1784
                                              5E   DD 00007              PUSHL   SP                               ;
                    7E      04    AC          04   C1 00009              ADDL3   #4, SIG, -(SP)                   ;
                    00000000G    00          02   FB 0000E              CALLS   #2, LIB$MATCH_COND               ;
                                 1B          50   E9 00015              BLBC    R0, 3$                           ;
                                              52   D4 00018              CLRL    ARG_NO                           ; 1790
                                              12   11 0001A              BRB     2$                               ;
                                 50      0C BC42   D0 0001C 1$:         MOVL    @ENBL[ARG_NO], R0                ; 1792
                                              60   D5 00021              TSTL    (R0)                             ;
                                              09   13 00023              BEQL    2$                               ;
                                              50   DD 00025              PUSHL   R0                               ;
                    00000000G    00          01   FB 00027              CALLS   #1, STR$FREE1_DX                 ;
                 E9              52      0C   BC   F3 0002E 2$:         AOBLEQ  @ENBL, ARG_NO, 1$                ;
                                 50    0918   8F   3C 00033 3$:         MOVZWL  #2328, R0                        ; 1794
                                                   04 00038              RET                                      ; 1795

; Routine Size:  57 bytes,    Routine Base:  _BAS$CODE + 0C61


; 1712          1796  1 END                                                      ! end of module BAS$SARITH
; 1713          1797  1
; 1714          1798  0 ELUDOM



;                                PSECT SUMMARY
;
;      Name                  Bytes                    Attributes
;
;   _BAS$CODE                       3226  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        10         0         581         00:01.0
```

```
;                                   COMMAND QUALIFIERS

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

; 1715          1799  0
; Size:         2713 code + 513 data bytes
; Run Time:        00:43.7
; Elapsed Time:    01:27.1
; Lines/CPU Min:     2468
; Lexemes/CPU-Min: 12779
; Memory Used:  188 pages
; Compilation Complete
```

0031 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

BASRTDIM
LIS

BASSARITH
LIS

BASSCALE
LIS

BASSIGNAL
LIS

BASRUNINI
LIS

BASSCRATC
LIS

BASRSTSFI
LIS

BASSLEEP
LIS

BASSTOP
LIS

BASSEG
LIS