```
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  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   SSSSSSSSSSS     RRR         RRR      TTT        LLLLLLLLLLLLLLL
BBBBBBBBBBBB      AAA         AAA   SSSSSSSSSSS     RRR         RRR      TTT        LLLLLLLLLLLLLLL
BBBBBBBBBBBB      AAA         AAA   SSSSSSSSSSS     RRR         RRR      TTT        LLLLLLLLLLLLLLL
```

```
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
```

```
;     1          0001  0 MODULE BAS$SARITH (                       ! BASIC String Arithmetic
;     2          0002  0                 IDENT = '1-024'           ! File. BASSARITH.B32 Edit: MDL1024
;     3          0003  0                 ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !**********************************************************************
;     7          0007  1 !*                                                                    *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                          *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.           *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                           *
;    11          0011  1 !*                                                                    *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
;    17          0017  1 !*  TRANSFERRED.                                                       *
;    18          0018  1 !*                                                                    *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
;    21          0021  1 !*  CORPORATION.                                                       *
;    22          0022  1 !*                                                                    *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.            *
;    25          0025  1 !*                                                                    *
;    26          0026  1 !*                                                                    *
;    27          0027  1 !**********************************************************************
;    28          0028  1 !
;    29          0029  1
;    30          0030  1 !++
;    31          0031  1 ! FACILITY:  BASIC String Arithmetic
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:
;    34          0034  1 !
;    35          0035  1 !       This module contains the BASIC string arithmetic functions.
;    36          0036  1 !       They operate by calling the STR$ string arithmetic entry
;    37          0037  1 !       points.  These routines are coded for reliability and
;    38          0038  1 !       maintainability, not for speed.
;    39          0039  1 !
;    40          0040  1 ! ENVIRONMENT:  VAX-11 User Mode
;    41          0041  1 !
;    42          0042  1 ! AUTHOR: John Sauter, CREATION DATE: 02-MAR-1979
;    43          0043  1 !
;    44          0044  1 ! MODIFIED BY:
;    45          0045  1 !
;    46          0046  1 ! 1-001 - Original.  JBS 05-MAR-1979
;    47          0047  1 ! 1-002 - Don't produce a leading zero if the result is between 0
;    48          0048  1 !           and 1.  JBS 07-MAR-1979
;    49          0049  1 ! 1-003 - Ask STR$RECIP to work to enough accuracy to be sure that the
;    50          0050  1 !           result of the divide will be sufficiently accurate.
;    51          0051  1 !           JBS 07-MAR-1979
;    52          0052  1 ! 1-004 - Correct the space allocation in PARSE_OUT, so we have enough
;    53          0053  1 !           to hold the entire number.  JBS 22-MAR-1979
;    54          0054  1 ! 1-005 - Correct the sign of the result of COMP%.  JBS 22-MAR-1979
;    55          0055  1 ! 1-006 - Improve comments based on the code review.  JBS 26-MAR-1979
;    56          0056  1 ! 1-007 - Correct a typo introduced in edit 006.  JBS 28-MAR-1979
;    57          0057  1 ! 1-008 - If we get an error, free local strings.  JBS 07-MAY-1979
```

```
;    58          0058  1 ! 1-009 - Call the STR$ facility with input scalars by reference.
;    59          0059  1 !           JBS 15-MAY-1979
;    60          0060  1 ! 1-010 - Change OTS$S and LIB$S to STR$.  JBS 21-MAY-1979
;    61          0061  1 ! 1-011 - Change call to STR$COPY.  JBS 16-JUL-1979
;    62          0062  1 ! 1-012 - In BAS$QUO, search for an exact quotient.  JBS 23-JUL-1979
;    63          0063  1 ! 1-013 - When freeing local strings, watch out for descriptors
;    64          0064  1 !           which have not yet been initialized.  JBS 31-JUL-1979
;    65          0065  1 ! 1-014 - When scanning the input string, allocate at least 8 bytes of
;    66          0066  1 !           space.  QAR N11-02925.  JBS 17-OCT-1979
;    67          0067  1 ! 1-015 - Try to speed up BAS$QUO by shortening the quotient search.
;    68          0068  1 !           Leave in monitoring code under conditional compilation in case
;    69          0069  1 !           it is needed again.  JBS 22-MAY-1980
;    70          0070  1 ! 1-016 - 3.75/3 to one decimal place gives 1.2 instead of 1.3.  To fix this,
;    71          0071  1 !           round before hill climbing.  JBS 13-JUN-1980
;    72          0072  1 ! 1-017 - Added code to routines BAS$PLACE and BAS$PROD to take care of
;    73          0073  1 !           the incompatibilities between VAX and the PDP-11 in handling of
;    74          0074  1 !           certain ROUND/TRUNCATION parameters (specifically rounding values
;    75          0075  1 !           in the range of -1 to -4 and truncation values in the range of
;    76          0076  1 !           9996 to 9999).  LEB  24-AUG-81
;    77          0077  1 ! 1-018 - Changed routine BAS$QUO to now call routine STR$DIVIDE instead of
;    78          0078  1 !           the calls to STR$MUL, STR$RECIP, and STR$ROUND.  LEB  8-OCT-81
;    79          0079  1 ! 1-019 - Added code to BAS$QUO to handle compatibility problems with
;    80          0080  1 !           regard to getting rid of unnecessary leading or trailing
;    81          0081  1 !           zeroes.  LEB 23-DEC-81.
;    82          0082  1 ! 1-020 - Changed code in BAS$PLACE and BAS$PROD to handle the ROUND
;    83          0083  1 !           TRUNCATION parameters - they now correctly handle values
;    84          0084  1 !           in the range of -4999 to -1 and in the range of 5001 to 9999.
;    85          0085  1 !           LEB 4-FEB-82.
;    86          0086  1 ! 1-021 - Added line to BAS$QUO to store updated length field into the
;    87          0087  1 !           descriptor.  LEB 2-MAR-82.
;    88          0088  1 ! 1-022 - Added code to PARSE_IN to allow for an input of -0, to fix
;    89          0089  1 !           a compatibility mode bug.  LEB  29-JUN-1982.
;    90          0090  1 ! 1-023 - additional change to PARSE_IN to accept +0, 0. as well.
;    91          0091  1 !           MDL 10-Jan-1983
;    92          0092  1 ! 1-024 - restore DSC$W_LENGTH before calling STR$FREE1_DX in STR$DIVIDE
;    93          0093  1 !           to avoid faking out the STR$ code and cause an access violation.
;    94          0094  1 !           MDL 15-Mar-1983
;    95          0095  1 !--
;    96          0096  1
;    97          0097  1 !<BLF/PAGE>
```

```
;   99     0098  1 !
;  100     0099  1 ! SWITCHES:
;  101     0100  1 !
;  102     0101  1
;  103     0102  1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;  104     0103  1
;  105     0104  1 !
;  106     0105  1 ! LINKAGES:
;  107     0106  1 !
;  108     0107  1 !       NONE
;  109     0108  1 !
;  110     0109  1 ! TABLE OF CONTENTS:
;  111     0110  1 !
;  112     0111  1
;  113     0112  1 FORWARD ROUTINE
;  114     0113  1     BAS$COMP,                               ! Compare strings
;  115     0114  1     BAS$DIF : NOVALUE,                      ! Difference between strings
;  116     0115  1     BAS$PLACE : NOVALUE,                    ! Shorten a string
;  117     0116  1     BAS$PROD : NOVALUE,                     ! Multiply two strings
;  118     0117  1     BAS$QUO : NOVALUE,                      ! Divide two strings
;  119     0118  1     BAS$SUM : NOVALUE,                      ! Sum of two strings
;  120     0119  1     PARSE_IN : NOVALUE,                     ! Convert to internal form
;  121     0120  1     PARSE_OUT : NOVALUE,                    ! Convert to external form
;  122     0121  1     FREE_STRINGS;                           ! Handler to free strings
;  123     0122  1
;  124     0123  1 !
;  125     0124  1 ! INCLUDE FILES:
;  126     0125  1 !
;  127     0126  1
;  128     0127  1 REQUIRE 'RTLIN:RTLPSECT';                   ! Macros for defining psects
;  129     0222  1
;  130     0223  1 LIBRARY 'RTLSTARLE';                        ! System definitions
;  131     0224  1
;  132     0225  1 !
;  133     0226  1 ! MACROS:
;  134     0227  1 !
;  135     0228  1 !       NONE
;  136     0229  1 !
;  137     0230  1 ! EQUATED SYMBOLS:
;  138     0231  1 !
;  139     0232  1
;  140     0233  1 LITERAL
;  141     0234  1 !+
;  142     0235  1 ! Several of the string arithmetic routines encode the precision and
;  143     0236  1 ! a truncation flag in a single parameter.  This is done as follows:
;  144     0237  1 ! (P% is the precision parameter)
;  145     0238  1 !
;  146     0239  1 !    1. If P% is less than or equal to 5000, rounding occurs; if
;  147     0240  1 !       more, truncation.
;  148     0241  1 !
;  149     0242  1 !    2. If P% is less than or equal to zero, rounding occurs on the
;  150     0243  1 !       left side of the decimal point.
;  151     0244  1 !
;  152     0245  1 !    3. If P% is between 0 and 5000, rounding occurs on the right
;  153     0246  1 !       side of the decimal point.
;  154     0247  1 !
;  155     0248  1 !    4. If P% is between 5000 and 10,000, truncation occurs on the
```

```
;  156          0249  1 !       left side of the decimal point.
;  157          0250  1 !
;  158          0251  1 !    5. If P% is greater than 10,000, truncation occurs on the right
;  159          0252  1 !       side of the decimal point.
;  160          0253  1 !
;  161          0254  1 ! To help the reader recognize where these semantics are implemented, the
;  162          0255  1 ! following symbol is used for 5000 (and 2* the symbol for 10,000) in the
;  163          0256  1 ! above contexts.
;  164          0257  1 !-
;  165          0258  1     BAS$K_PREC_VAL = 5000,
;  166          0259  1 !+
;  167          0260  1 ! There is a limit imposed by the PDP-11 implementation of BASIC-PLUS-2
;  168          0261  1 ! on the number of digits which can be processed.  This limit appears
;  169          0262  1 ! in two places: the precision of QUO$ is limited to 1E-55 (or 1E-56 if
;  170          0263  1 ! truncating) and number of digits which can be input is limited to 60.
;  171          0264  1 ! To help the reader recognize where these limits are implemented, the
;  172          0265  1 ! following constants are defined:
;  173          0266  1 !-
;  174          0267  1     BAS$K_PREC_LIM1 = 55,                   ! Max digits in QUO$ when rounding
;  175          0268  1     BAS$K_PREC_LIM2 = 56,                   ! Max digits in QUO$ when truncating
;  176          0269  1     BAS$K_PREC_LIM3 = 60;                   ! Max digits when scanning input
;  177          0270  1
;  178          0271  1 !
;  179          0272  1 !
;  180          0273  1 ! PSECTS:
;  181          0274  1 !
;  182          0275  1 DECLARE_PSECTS (BAS);                       ! Declare psects for BAS$ facility
;  183          0276  1 !
;  184          0277  1 ! OWN STORAGE:
;  185          0278  1 !
;  186          0279  1 !       NONE
;  187          0280  1 !
;  188          0281  1 ! EXTERNAL REFERENCES:
;  189          0282  1 !
;  190          0283  1
;  191          0284  1 EXTERNAL ROUTINE
;  192          0285  1     BAS$$STOP : NOVALUE,                    ! Signals a fatal BASIC error
;  193          0286  1     BAS$$SIGNAL : NOVALUE,                  ! Signals a BASIC error
;  194          0287  1     STR$GET1_DX,                            ! Allocate a string
;  195          0288  1     STR$FREE1_DX,                           ! Deallocate a string
;  196          0289  1     STR$COPY_R,                             ! Copy a string by ref
;  197          0290  1     LIB$MATCH_COND,                         ! Match condition codes
;  198          0291  1     STR$ADD : NOVALUE,                      ! Add two strings
;  199          0292  1     STR$MUL : NOVALUE,                      ! Multiply two strings
;  200          0293  1     STR$RECIP : NOVALUE,                    ! Take the reciprocal of a string
;  201          0294  1     STR$ROUND : NOVALUE,                    ! Rounds a string
;  202          0295  1     STR$DIVIDE: NOVALUE;                    ! Divide two strings
;  203          0296  1
;  204          0297  1 !+
;  205          0298  1 ! The following are the error codes used in this module:
;  206          0299  1 !-
;  207          0300  1
;  208          0301  1 EXTERNAL LITERAL
;  209          0302  1     BAS$K_DATFORERR : UNSIGNED (8),         ! Data format error
;  210          0303  1     BAS$K_ILLNUM : UNSIGNED (8),            ! Illegal number
;  211          0304  1     BAS$K_FLOPOIERR : UNSIGNED (8),         ! Floating point error
;  212          0305  1     BAS$K_DIVBY_ZER : UNSIGNED (8);         ! Division by zero
```

```
;  213          0306  1
;  214          0307  1 BIND
;  215          0308  1     TABLE_Z = UPLIT BYTE (REP 49 OF (%X'00'),REP 9 OF (%X'01'),REP 198 OF (%X'00')),
;  216          0309  1     TABLE_NZ = UPLIT BYTE (REP 49 OF (%X'01'),REP 207 OF (%X'00')),
;  217          0310  1     MASK = UPLIT BYTE (%X'01');
;  218          0311  1
;  219          0312  1 BUILTIN
;  220          0313  1     SPANC;                                  !\Skip over a specified set of
;  221          0314  1                                             !/characters in a string
```

```
;  223     0315  1 GLOBAL ROUTINE BAS$COMP (                      ! Compare strings
;  224     0316  1         ARG1,                                   ! First operand
;  225     0317  1         ARG2                                    ! Second operand
;  226     0318  1     ) =
;  227     0319  1
;  228     0320  1 !++
;  229     0321  1 ! FUNCTIONAL DESCRIPTION:
;  230     0322  1 !
;  231     0323  1 !       Compare two strings.  This is done by subtracting them and
;  232     0324  1 !       testing the result for zero.
;  233     0325  1 !
;  234     0326  1 ! FORMAL PARAMETERS:
;  235     0327  1 !
;  236     0328  1 !       ARG1.rt.dx      First number to compare, as +nnn.nnn
;  237     0329  1 !       ARG2.rt.dx      Number to compare against, as +nnn.nnn
;  238     0330  1 !
;  239     0331  1 ! IMPLICIT INPUTS:
;  240     0332  1 !
;  241     0333  1 !       NONE
;  242     0334  1 !
;  243     0335  1 ! IMPLICIT OUTPUTS:
;  244     0336  1 !
;  245     0337  1 !       NONE
;  246     0338  1 !
;  247     0339  1 ! ROUTINE VALUE:
;  248     0340  1 ! COMPLETION CODES:
;  249     0341  1 !
;  250     0342  1 !       0 = ARGs are equal,
;  251     0343  1 !       -1 = ARG1 less than ARG2,
;  252     0344  1 !       1 = ARG1 greater than ARG2.
;  253     0345  1 !
;  254     0346  1 ! SIDE EFFECTS:
;  255     0347  1 !
;  256     0348  1 !       Signals PROLOSSOR (for "impossible" conditions) and the
;  257     0349  1 !       signals from PARSE_IN.
;  258     0350  1 !
;  259     0351  1 !--
;  260     0352  1
;  261     0353  2     BEGIN
;  262     0354  2
;  263     0355  2     MAP
;  264     0356  2         ARG1 : REF BLOCK [8, BYTE],
;  265     0357  2         ARG2 : REF BLOCK [8, BYTE];
;  266     0358  2
;  267     0359  2     LOCAL
;  268     0360  2 !+
;  269     0361  2 ! Internal form of ARG1
;  270     0362  2 !-
;  271     0363  2         A_DESC : BLOCK [8, BYTE] VOLATILE,
;  272     0364  2         A_SIGN,
;  273     0365  2         A_EXP,
;  274     0366  2 !+
;  275     0367  2 ! Internal form of ARG2
;  276     0368  2 !-
;  277     0369  2         B_DESC : BLOCK [8, BYTE] VOLATILE,
;  278     0370  2         B_SIGN,
;  279     0371  2         B_EXP,
```

```
;  280     0372  2 !+
;  281     0373  2 ! Temp to hold A-B.
;  282     0374  2 !-
;  283     0375  2         C_DESC : BLOCK [8, BYTE] VOLATILE,
;  284     0376  2         C_SIGN,
;  285     0377  2         C_EXP,
;  286     0378  2 !+
;  287     0379  2 ! Temp to hold result of comparison.
;  288     0380  2 !-
;  289     0381  2         RESULT;
;  290     0382  2
;  291     0383  2 !+
;  292     0384  2 ! Enable a handler which will free the strings
;  293     0385  2 !-
;  294     0386  2
;  295     0387  2     ENABLE
;  296     0388  2         FREE_STRINGS (A_DESC, B_DESC, C_DESC);
;  297     0389  2
;  298     0390  2 !+
;  299     0391  2 ! Convert the two arguments from external form to internal form.
;  300     0392  2 ! This is done by removing the non-digits from the string and
;  301     0393  2 ! returning the sign and exponent as separate values.
;  302     0394  2 ! Errors are signaled.
;  303     0395  2 !-
;  304     0396  2     A_DESC [DSC$W_LENGTH] = 0;
;  305     0397  2     A_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;  306     0398  2     A_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  307     0399  2     A_DESC [DSC$A_POINTER] = 0;
;  308     0400  2     PARSE_IN (.ARG1, A_DESC, A_SIGN, A_EXP);
;  309     0401  2     B_DESC [DSC$W_LENGTH] = 0;
;  310     0402  2     B_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;  311     0403  2     B_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  312     0404  2     B_DESC [DSC$A_POINTER] = 0;
;  313     0405  2     PARSE_IN (.ARG2, B_DESC, B_SIGN, B_EXP);
;  314     0406  2 !+
;  315     0407  2 ! If the signs differ we can compute the result based on them.
;  316     0408  2 ! Otherwise we must subtract the inputs.
;  317     0409  2 !-
;  318     0410  2     C_DESC [DSC$W_LENGTH] = 0;
;  319     0411  2     C_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;  320     0412  2     C_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  321     0413  2     C_DESC [DSC$A_POINTER] = 0;
;  322     0414  2
;  323     0415  3     IF (.A_SIGN NEQ .B_SIGN)
;  324     0416  2     THEN
;  325     0417  2         C_SIGN = .B_SIGN
;  326     0418  2     ELSE
;  327     0419  3         BEGIN
;  328     0420  3         A_SIGN = 1 - .A_SIGN;                   ! Change the sign of A
;  329     0421  3         STR$ADD (A_SIGN, A_EXP, A_DESC, B_SIGN, B_EXP, B_DESC, C_SIGN, C_EXP, C_DESC);
;  330     0422  2         END;
;  331     0423  2
;  332     0424  2 !+
;  333     0425  2 ! We are done with A and B, so free them.
;  334     0426  2 !-
;  335     0427  2     STR$FREE1_DX (A_DESC);
;  336     0428  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
```

```
;  359     0450  1 GLOBAL ROUTINE BAS$DIF (                          ! Subtract strings
;  360     0451  1         DIF_DESC,                                  ! Difference
;  361     0452  1         OP1_DESC,                                  ! First input
;  362     0453  1         OP2_DESC                                   ! Second input
;  363     0454  1     ) : NOVALUE =
;  364     0455  1
;  365     0456  1 !++
;  366     0457  1 ! FUNCTIONAL DESCRIPTION:
;  367     0458  1 !
;  368     0459  1 !       Subtract two strings.  DIF := OP1 - OP2
;  369     0460  1 !
;  370     0461  1 ! FORMAL PARAMETERS:
;  371     0462  1 !
;  372     0463  1 !       DIF_DESC.wt.dx  The difference, OP1 - OP2
;  373     0464  1 !       OP1_DESC.rt.dx  Operand OP1
;  374     0465  1 !       OP2_DESC.rt.dx  Operand OP2
;  375     0466  1 !
;  376     0467  1 ! IMPLICIT INPUTS:
;  377     0468  1 !
;  378     0469  1 !       NONE
;  379     0470  1 !
;  380     0471  1 ! IMPLICIT OUTPUTS:
;  381     0472  1 !
;  382     0473  1 !       NONE
;  383     0474  1 !
;  384     0475  1 ! ROUTINE VALUE:
;  385     0476  1 ! COMPLETION CODES:
;  386     0477  1 !
;  387     0478  1 !       NONE
;  388     0479  1 !
;  389     0480  1 ! SIDE EFFECTS:
;  390     0481  1 !
;  391     0482  1 !       Signals PROLOSSOR for certain "impossible" errors, and the signals
;  392     0483  1 !       from PARSE_IN and PARSE_OUT.
;  393     0484  1 !
;  394     0485  1 !--
;  395     0486  1
;  396     0487  2     BEGIN
;  397     0488  2
;  398     0489  2     MAP
;  399     0490  2         OP1_DESC : REF BLOCK [8, BYTE],
;  400     0491  2         OP2_DESC : REF BLOCK [8, BYTE],
;  401     0492  2         DIF_DESC : REF BLOCK [8, BYTE];
;  402     0493  2
;  403     0494  2     LOCAL
;  404     0495  2 !+
;  405     0496  2 ! Internal form of OP1.
;  406     0497  2 !-
;  407     0498  2         A_DESC : BLOCK [8, BYTE] VOLATILE,
;  408     0499  2         A_SIGN,
;  409     0500  2         A_EXP,
;  410     0501  2 !+
;  411     0502  2 ! Internal form of OP2.
;  412     0503  2 !-
;  413     0504  2         B_DESC : BLOCK [8, BYTE] VOLATILE,
;  414     0505  2         B_SIGN,
;  415     0506  2         B_EXP,
```

```
;  416     0507  2 !+
;  417     0508  2 ! Internal form of the difference, OP1 - OP2.
;  418     0509  2 !-
;  419     0510  2         C_DESC : BLOCK [8, BYTE] VOLATILE,
;  420     0511  2         C_SIGN,
;  421     0512  2         C_EXP;
;  422     0513  2
;  423     0514  2 !+
;  424     0515  2 ! Enable a handler to free the strings.
;  425     0516  2 !-
;  426     0517  2
;  427     0518  2     ENABLE
;  428     0519  2         FREE_STRINGS (A_DESC, B_DESC, C_DESC);
;  429     0520  2
;  430     0521  2 !+
;  431     0522  2 ! Convert the two input arguments from external form to internal form.
;  432     0523  2 ! This is done by removing the non-digits from the string and
;  433     0524  2 ! returning the sign and exponent as separate values.
;  434     0525  2 ! Signal any errors.
;  435     0526  2 !-
;  436     0527  2     A_DESC [DSC$W_LENGTH] = 0;
;  437     0528  2     A_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;  438     0529  2     A_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  439     0530  2     A_DESC [DSC$A_POINTER] = 0;
;  440     0531  2     PARSE_IN (.OP1_DESC, A_DESC, A_SIGN, A_EXP);
;  441     0532  2     B_DESC [DSC$W_LENGTH] = 0;
;  442     0533  2     B_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;  443     0534  2     B_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  444     0535  2     B_DESC [DSC$A_POINTER] = 0;
;  445     0536  2     PARSE_IN (.OP2_DESC, B_DESC, B_SIGN, B_EXP);
;  446     0537  2 !+
;  447     0538  2 ! Subtract the numbers using the large-precision string arithmetic
;  448     0539  2 ! package.
;  449     0540  2 !-
;  450     0541  2     C_DESC [DSC$W_LENGTH] = 0;
;  451     0542  2     C_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;  452     0543  2     C_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  453     0544  2     C_DESC [DSC$A_POINTER] = 0;
;  454     0545  2     B_SIGN = 1 - .B_SIGN;                          ! Change the sign of B.
;  455     0546  2     STR$ADD (A_SIGN, A_EXP, A_DESC, B_SIGN, B_EXP, B_DESC, C_SIGN, C_EXP, C_DESC);
;  456     0547  2 !+
;  457     0548  2 ! We are done with A and B, so free them.
;  458     0549  2 !-
;  459     0550  2     STR$FREE1_DX (A_DESC);
;  460     0551  2     STR$FREE1_DX (B_DESC);
;  461     0552  2 !+
;  462     0553  2 ! Convert the difference to external form for the caller.
;  463     0554  2 !-
;  464     0555  2     PARSE_OUT (C_DESC, .C_SIGN, .C_EXP, .DIF_DESC);
;  465     0556  2 !+
;  466     0557  2 ! We can now free C
;  467     0558  2 !-
;  468     0559  2     STR$FREE1_DX (C_DESC);
;  469     0560  2     RETURN;
;  470     0561  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
```

```
;  473     0563  1 GLOBAL ROUTINE BAS$PLACE (                    ! Round or truncate a string
;  474     0564  1         RESULT_DESC,                           ! Result
;  475     0565  1         OP1_DESC,                              ! Number to be rounded/truncated
;  476     0566  1         PRECISION                              ! Digits of result
;  477     0567  1     ) : NOVALUE =
;  478     0568  1
;  479     0569  1 !++
;  480     0570  1 ! FUNCTIONAL DESCRIPTION:
;  481     0571  1 !
;  482     0572  1 !       Round or truncate a string to a particular position after
;  483     0573  1 !       (or before) the decimal point.  RESULT = round (OP1)
;  484     0574  1 !
;  485     0575  1 ! FORMAL PARAMETERS:
;  486     0576  1 !
;  487     0577  1 !       RESULT_DESC.wt.dx The result of rounding/truncating OP1
;  488     0578  1 !       OP1_DESC.rt.dx  Operand OP1
;  489     0579  1 !       PRECISION.rl.v  Number of digits to retain and, encoded,
;  490     0580  1 !                       the round/truncate flag.
;  491     0581  1 !
;  492     0582  1 ! IMPLICIT INPUTS:
;  493     0583  1 !
;  494     0584  1 !       NONE
;  495     0585  1 !
;  496     0586  1 ! IMPLICIT OUTPUTS:
;  497     0587  1 !
;  498     0588  1 !       NONE
;  499     0589  1 !
;  500     0590  1 ! ROUTINE VALUE:
;  501     0591  1 ! COMPLETION CODES:
;  502     0592  1 !
;  503     0593  1 !       NONE
;  504     0594  1 !
;  505     0595  1 ! SIDE EFFECTS:
;  506     0596  1 !
;  507     0597  1 !       Signals PROLOSSOR for certain "impossible" errors, and the signals
;  508     0598  1 !       from PARSE_IN and PARSE_OUT.
;  509     0599  1 !
;  510     0600  1 !--
;  511     0601  1
;  512     0602  2     BEGIN
;  513     0603  2
;  514     0604  2     MAP
;  515     0605  2         OP1_DESC : REF BLOCK [8, BYTE],
;  516     0606  2         RESULT_DESC : REF BLOCK [8, BYTE];
;  517     0607  2
;  518     0608  2     LOCAL
;  519     0609  2 !+
;  520     0610  2 ! Internal form of OP1.
;  521     0611  2 !-
;  522     0612  2         A_DESC : BLOCK [8, BYTE] VOLATILE,
;  523     0613  2         A_SIGN,
;  524     0614  2         A_EXP,
;  525     0615  2         TRUNC_FLAG,                            ! 1 = truncate (rather than round)
;  526     0616  2         ROUND_POS,                             ! Where to round
;  527     0617  2 !+
;  528     0618  2 ! The following is a descriptor for the constant 1.  It is multiplied by
;  529     0619  2 ! a power of 10 for rounding or truncating.
```

```
;  530          0620  2 !-
;  531          0621  2         ONE_DESC : BLOCK [8, BYTE],
;  532          0622  2         ONE_BUF : VECTOR [1, BYTE];
;  53           0623  2
;  534          0624  2 !+
;  535          0625  2 ! Enable a handler to free A_DESC in case of error.
;  536          0626  2 !-
;  537          0627  2
;  538          0628  2     ENABLE
;  539          0629  2         FREE_STRINGS (A_DESC);
;  540          0630  2
;  541          0631  2 !+
;  542          0632  2 ! Convert the input argument from external form to internal form.
;  543          0633  2 ! This is done by removing the non-digits from the string and
;  544          0634  2 ! returning the sign and exponent as separate values.
;  545          0635  2 ! Errors are signaled.
;  546          0636  2 !-
;  547          0637  2     A_DESC [DSC$W_LENGTH] = 0;
;  548          0638  2     A_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;  549          0639  2     A_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  550          0640  2     A_DESC [DSC$A_POINTER] = 0;
;  551          0641  2     PARSE_IN (.OPT_DESC, A_DESC, A_SIGN, A_EXP);
;  552          0642  2 !+
;  553          0643  2 ! Round or truncate the result to a specified decimal place.
;  554          0644  2 !-
;  555          0645  2
;  556          0646  3     IF (.PRECISION LEQ BAS$K_PREC_VAL)
;  557          0647  2     THEN
;  558          0648  3         BEGIN
;  559          0649  3         TRUNC_FLAG = 0;
;  560          0650  3         ROUND_POS = -.PRECISION;
;  561          0651  3         END
;  562          0652  2     ELSE
;  563          0653  3         BEGIN
;  564          0654  3         TRUNC_FLAG = 1;
;  565          0655  3         ROUND_POS = -(.PRECISION - (BAS$K_PREC_VAL*2));
;  566          0656  2         END;
;  567          0657  2
;  568          0658  2 !+
;  569          0659  2 ! In order to do the BASIC rounding/truncation based on decimal place
;  570          0660  2 ! using the string package's rounding/truncation based on number of
;  571          0661  2 ! significant digits, we must be sure that the position we are rounding
;  572          0662  2 ! to is part of the significance.  Therefore we add 1E<ROUND_POS>,
;  573          0663  2 ! round or truncate to discard all lower digits, and then subtract
;  574          0664  2 ! 1E<ROUND_POS> to get back to where we belong.
;  575          0665  2 !-
;  576          0666  2     ONE_DESC [DSC$W_LENGTH] = 1;
;  577          0667  2     ONE_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;  578          0668  2     ONE_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
;  579          0669  2     ONE_DESC [DSC$A_POINTER] = ONE_BUF [0];
;  580          0670  2     ONE_BUF [0] = %C'1';
;  581          0671  2     STR$ADD (A_SIGN, A_EXP, A_DESC,                    !
;  582          0672  2         A_SIGN, ROUND_POS, ONE_DESC,                   !
;  583          0673  2         A_SIGN, A_EXP, A_DESC);
;  584          0674  2     STR$ROUND (%REF (.A_EXP + .A_DESC [DSC$W_LENGTH] - .ROUND_POS), TRUNC_FLAG,        !
;  585          0675  2         A_SIGN, A_EXP, A_DESC,                         !
;  586          0676  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   9$, (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
```

```
;  629          0718  1 GLOBAL ROUTINE BAS$PROD (                     ! Multiply strings
;  630          0719  1         PROD_DESC,                            ! Product
;  631          0720  1         OP1_DESC,                             ! First input
;  632          0721  1         OP2_DESC,                             ! Second input
;  633          0722  1         PRECISION                             ! Digits of result
;  634          0723  1     ) : NOVALUE =
;  635          0724  1
;  636          0725  1 !++
;  637          0726  1 ! FUNCTIONAL DESCRIPTION:
;  638          0727  1 !
;  639          0728  1 !       Multiply two strings.  PROD := OP1 * OP2
;  640          0729  1 !
;  641          0730  1 ! FORMAL PARAMETERS:
;  642          0731  1 !
;  643          0732  1 !       PROD_DESC.wt.dx The product of OP1 and OP2
;  644          0733  1 !       OP1_DESC.rt.dx  Operand OP1
;  645          0734  1 !       OP2_DESC.rt.dx  Operand OP2
;  646          0735  1 !       PRECISION.rl.v  Number of digits to retain and, encoded,
;  647          0736  1 !                       the round/truncate flag.
;  648          0737  1 !
;  649          0738  1 ! IMPLICIT INPUTS:
;  650          0739  1 !
;  651          0740  1 !       NONE
;  652          0741  1 !
;  653          0742  1 ! IMPLICIT OUTPUTS:
;  654          0743  1 !
;  655          0744  1 !       NONE
;  656          0745  1 !
;  657          0746  1 ! ROUTINE VALUE:
;  658          0747  1 ! COMPLETION CODES:
;  659          0748  1 !
;  660          0749  1 !       NONE
;  661          0750  1 !
;  662          0751  1 ! SIDE EFFECTS:
;  663          0752  1 !
;  664          0753  1 !       Signals PROLOSSOR for certain "impossible" errors, and the signals
;  665          0754  1 !       from PARSE_IN and PARSE_OUT.
;  666          0755  1 !
;  667          0756  1 !--
;  668          0757  1
;  669          0758  2     BEGIN
;  670          0759  2
;  671          0760  2     MAP
;  672          0761  2         OP1_DESC : REF BLOCK [8, BYTE],
;  673          0762  2         OP2_DESC : REF BLOCK [8, BYTE],
;  674          0763  2         PROD_DESC : REF BLOCK [8, BYTE];
;  675          0764  2
;  676          0765  2     LOCAL
;  677          0766  2 !+
;  678          0767  2 ! Internal form of OP1.
;  679          0768  2 !-
;  680          0769  2         A_DESC : BLOCK [8, BYTE] VOLATILE,
;  681          0770  2         A_SIGN,
;  682          0771  2         A_EXP,
;  683          0772  2 !+
;  684          0773  2 ! Internal form of OP2.
;  685          0774  2 !-
```

```
;  686          0775  2          B_DESC : BLOCK [8, BYTE] VOLATILE,
;  687          0776  2          B_SIGN,
;  688          0777  2          B_EXP,
;  689          0778  2      !+
;  690          0779  2      ! Internal form of the product, A*B.
;  691          0780  2      !-
;  692          0781  2          C_DESC : BLOCK [8, BYTE] VOLATILE,
;  693          0782  2          C_SIGN,
;  694          0783  2          C_EXP,
;  695          0784  2          TRUNC_FLAG,                         ! 1 = truncate, 0 = round.
;  696          0785  2          ROUND_POS,                          ! Where to round the product
;  697          0786  2      !+
;  698          0787  2      ! This is the constant 1, which is muliplied by a power of 10 for rounding
;  699          0788  2      ! purposes.
;  700          0789  2      !-
;  701          0790  2          ONE_DESC : BLOCK [8, BYTE],
;  702          0791  2          ONE_BUF : VECTOR [1, BYTE];
;  703          0792  2
;  704          0793  2      !+
;  705          0794  2      ! Enable a handler to free the local strings.
;  706          0795  2      !-
;  707          0796  2
;  708          0797  2          ENABLE
;  709          0798  2              FREE_STRINGS (A_DESC, B_DESC, C_DESC);
;  710          0799  2
;  711          0800  2      !+
;  712          0801  2      ! Convert the two input arguments from external form to internal form.
;  713          0802  2      ! This is done by removing the non-digits from the string and
;  714          0803  2      ! returning the sign and exponent as separate values.
;  715          0804  2      ! Errors are signaled.
;  716          0805  2      !-
;  717          0806  2          A_DESC [DSC$W_LENGTH] = 0;
;  718          0807  2          A_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;  719          0808  2          A_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  720          0809  2          A_DESC [DSC$A_POINTER] = 0;
;  721          0810  2          PARSE_IN (.OP1_DESC, A_DESC, A_SIGN, A_EXP);
;  722          0811  2          B_DESC [DSC$W_LENGTH] = 0;
;  723          0812  2          B_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;  724          0813  2          B_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  725          0814  2          B_DESC [DSC$A_POINTER] = 0;
;  726          0815  2          PARSE_IN (.OP2_DESC, B_DESC, B_SIGN, B_EXP);
;  727          0816  2      !+
;  728          0817  2      ! Multiply the numbers using the large-precision string arithmetic
;  729          0818  2      ! package.
;  730          0819  2      !-
;  731          0820  2          C_DESC [DSC$W_LENGTH] = 0;
;  732          0821  2          C_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;  733          0822  2          C_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  734          0823  2          C_DESC [DSC$A_POINTER] = 0;
;  735          0824  2          STR$MUL (A_SIGN, A_EXP, A_DESC, B_SIGN, B_EXP, B_DESC, C_SIGN, C_EXP, C_DESC);
;  736          0825  2      !+
;  737          0826  2      ! We are done with A and B, so free them.
;  738          0827  2      !-
;  739          0828  2          STR$FREE1_DX (A_DESC);
;  740          0829  2          STR$FREE1_DX (B_DESC);
;  741          0830  2      !+
;  742          0831  2      ! Round or truncate the result to a specified decimal place.
```

```
;  743          0832  2 !-
;  744          0833  2
;  745          0834  3     IF (.PRECISION LEQ BAS$K_PREC_VAL)
;  746          0835  2     THEN
;  747          0836  3         BEGIN
;  748          0837  3         TRUNC_FLAG = 0;
;  749          0838  3         ROUND_POS = -.PRECISION;
;  750          0839  3         END
;  751          0840  2     ELSE
;  752          0841  3         BEGIN
;  753          0842  3         TRUNC_FLAG = 1;
;  754          0843  3         ROUND_POS = -(.PRECISION - (2*BAS$K_PREC_VAL));
;  755          0844  3         END;
;  756          0845  2
;  757          0846  2 !+
;  758          0847  2 ! In order to do the BASIC rounding/truncation based on decimal place
;  759          0848  2 ! using the string package's rounding/truncation based on number of
;  760          0849  2 ! significant digits, we must be sure that the position we are rounding
;  761          0850  2 ! to is part of the significance.  Therefore we add 1E<ROUND_POS>,
;  762          0851  2 ! round or truncate to discard all lower digits, and then subtract
;  763          0852  2 ! 1E<ROUND_POS> to get back to where we belong.
;  764          0853  2 !-
;  765          0854  2     ONE_DESC [DSC$W_LENGTH] = 1;
;  766          0855  2     ONE_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;  767          0856  2     ONE_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
;  768          0857  2     ONE_DESC [DSC$A_POINTER] = ONE_BUF [0];
;  769          0858  2     ONE_BUF [0] = %C'1';
;  770          0859  2     STR$ADD (C_SIGN, C_EXP, C_DESC,                 !
;  771          0860  2         C_SIGN, ROUND_POS, ONE_DESC,                !
;  772          0861  2         C_SIGN, C_EXP, C_DESC);
;  773          0862  2     STR$ROUND (%REF (.C_EXP + .C_DESC [DSC$W_LENGTH] - .ROUND_POS), TRUNC_FLAG,         !
;  774          0863  2         C_SIGN, C_EXP, C_DESC,                      !
;  775          0864  2         C_SIGN, C_EXP, C_DESC);
;  776          0865  2     STR$ADD (C_SIGN, C_EXP, C_DESC,                 !
;  777          0866  2         %REF ((IF (.C_SIGN) THEN 0 ELSE 1)), ROUND_POS, ONE_DESC,       !
;  778          0867  2         C_SIGN, C_EXP, C_DESC);
;  779          0868  2
;  780          0869  2 !+
;  781          0870  2 ! Check if the precision value is in the range of -1 to -4 (indicating
;  782          0871  2 ! rounding) or if the value is in the range of 9996 to 9999 (indicating
;  783          0872  2 ! truncation).  In either case, there is an incompatibility to be fixed
;  784          0873  2 ! here, so that VAX BASIC returns the same value as BASIC+2.
;  785          0874  2 !-
;  786          0875  2
;  787          0876  2     SELECTONE .PRECISION OF
;  788          0877  2       SET
;  789          0878  2         [-4999 TO -1]:                              ! Rounding parameter
;  790          0879  3           BEGIN
;  791          0880  3           IF .C_EXP NEQ 0
;  792          0881  3             THEN
;  793          0882  3               C_EXP = .C_EXP - ABS(.PRECISION);
;  794          0883  2           END;
;  795          0884  2         [5001 TO 9999]:                             ! Truncation parameter
;  796          0885  3           BEGIN
;  797          0886  3             LOCAL
;  798          0887  3               TEMP;
;  799          0888  3           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
                                    31  13 00156          BEQL    8$                                      ;
                                50      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
```

```
;  818          0906  1 GLOBAL ROUTINE BAS$QUO (                      ! Divide strings
;  819          0907  1         QUO_DESC,                              ! Quotient
;  820          0908  1         OP1_DESC,                              ! First input
;  821          0909  1         OP2_DESC,                              ! Second input
;  822          0910  1         PRECISION                              ! Digits of result
;  823          0911  1     ) : NOVALUE =
;  824          0912  1
;  825          0913  1 !++
;  826          0914  1 ! FUNCTIONAL DESCRIPTION:
;  827          0915  1 !
;  828          0916  1 !       Divide two strings.  QUO := OP1 / OP2
;  829          0917  1 !       No more than 55 digits of precision are permitted.
;  830          0918  1 !       (56 if truncating.)
;  831          0919  1 !
;  832          0920  1 ! FORMAL PARAMETERS:
;  833          0921  1 !
;  834          0922  1 !       QUO_DESC.wt.dx  The quotient of OP1 and OP2
;  835          0923  1 !       OP1_DESC.rt.dx  Operand OP1
;  836          0924  1 !       OP2_DESC.rt.dx  Operand OP2
;  837          0925  1 !       PRECISION.rl.v  Number of digits to retain and, encoded,
;  838          0926  1 !                       the round/truncate flag.
;  839          0927  1 !
;  840          0928  1 ! IMPLICIT INPUTS:
;  841          0929  1 !
;  842          0930  1 !       NONE
;  843          0931  1 !
;  844          0932  1 ! IMPLICIT OUTPUTS:
;  845          0933  1 !
;  846          0934  1 !       NONE
;  847          0935  1 !
;  848          0936  1 ! ROUTINE VALUE:
;  849          0937  1 ! COMPLETION CODES:
;  850          0938  1 !
;  851          0939  1 !       NONE
;  852          0940  1 !
;  853          0941  1 ! SIDE EFFECTS:
;  854          0942  1 !
;  855          0943  1 !       Signals BAS$_FLOPOIERR if the number of places is greater than
;  856          0944  1 !       55 or 56.  Signals BAS$_DIVBY_ZER if the divisor is zero.
;  857          0945  1 !       Also, PARSE_IN and PARSE_OUT signal.
;  858          0946  1 !--
;  859          0947  1
;  860          0948  2     BEGIN
;  861          0949  2
;  862          0950  2     MAP
;  863          0951  2         OP1_DESC : REF BLOCK [8, BYTE],
;  864          0952  2         OP2_DESC : REF BLOCK [8, BYTE],
;  865          0953  2         QUO_DESC : REF BLOCK [8, BYTE];
;  866          0954  2
;  867          0955  2     LOCAL
;  868          0956  2 !+
;  869          0957  2 ! Internal form of OP1.
;  870          0958  2 !-
;  871          0959  2         A_DESC : BLOCK [8, BYTE] VOLATILE,
;  872          0960  2         A_SIGN,
;  873          0961  2         A_EXP,
;  874          0962  2 !+
```

```
;  875          0963  2 ! Internal form of OP2.
;  876          0964  2 !-
;  877          0965  2         B_DESC : BLOCK [8, BYTE] VOLATILE,
;  878          0966  2         B_SIGN,
;  879          0967  2         B_EXP,
;  880          0968  2 !+
;  881          0969  2 ! Internal form of quotient.
;  882          0970  2 !-
;  883          0971  2         C_DESC : BLOCK [8, BYTE] VOLATILE,
;  884          0972  2         C_SIGN,
;  885          0973  2         C_EXP,
;  886          0974  2
;  887          0975  2         RND_TRUNC,                      ! 0 = truncate, 1 = round
;  888          0976  2         PREC,                           ! Number of digits to the
;  889          0977  2                                         !   right of the decimal point
;  890          0978  2         END_ADDR,                       ! End of string address
;  891          0979  2         NEW_ADDR,                       !\Used to store resultant
;  892          0980  2                                         !/address from SPANC
;  893          0981  2         BUF,                            ! Pointer to string of digits
;  894          0982  2         SAV_ADDR,                       ! Temporary storage address
;  895          0983  2         LENGTH,                         ! Used in SPANC
;  896          0984  2         C_LENGTH,                       ! Used to store length of string
;  897          0985  2         SAV_C_LEN,                      ! storage for actual len. of C
;  898          0986  2         SAV_C_PTR,                      ! storage for actual addr of C
;  899          0987  2         SAV_BUF,                        ! Storage for buffer address
;  900          0988  2         SAV_EXP,                        ! Storage for exp value
;  901          0989  2         SAV_LENGTH,                     ! Storage for length value
;  902          0990  2         DIFF;                           ! Used in SPANC to calc offsets
;  903          0991  2 !+
;  904          0992  2 ! The following FORTRAN subroutines can be used to print the progress of
;  905          0993  2 ! this routine, if the calls to them are enabled.
;  906          0994  2 !-
;  907        C 0995  2     %(
;  908        C 0996  2         SUBROUTINE MONITOR_I (ICODE, IVAL)
;  909        C 0997  2         IMPLICIT INTEGER (A-Z)
;  910        C 0998  2         TYPE 900, ICODE, IVAL
;  911        C 0999  2 900     FORMAT (I3,I6)
;  912        C 1000  2         RETURN
;  913        C 1001  2         END
;  914        C 1002  2         SUBROUTINE MONITOR_T (ICODE, IVAL)
;  915        C 1003  2         IMPLICIT INTEGER (A-Z)
;  916        C 1004  2         CHARACTER*(*) IVAL
;  917        C 1005  2         TYPE 901, ICODE, IVAL
;  918        C 1006  2 901     FORMAT (I3, 3X, A)
;  919        C 1007  2         RETURN
;  920        C 1008  2         END
;  921          1009  2 )%
;  922          1010  2 !+
;  923          1011  2 ! In the printout from the above code,
;  924          1012  2 !
;  925          1013  2 !         1 = eps_exp
;  926          1014  2 !         2 = delta_exp
;  927          1015  2 !         3 = round_pos
;  928          1016  2 !         4 = c_exp
;  929          1017  2 !         5 = trial_exp
;  930          1018  2 !         6 = trial_digits
;  931          1019  2 !         7 = c_digits
```

```
;  932          1020  2 !-
;  933          1021  2
;  934          1022  2     COMPILETIME
;  935          1023  2         PERFORM_MONITORING = 0;                  ! Set =1 to get calls to monitors
;  936          1024  2
;  937        L 1025  2 %IF PERFORM_MONITORING
;  938        U 1026  2 %THEN
;  939        U 1027  2
;  940        U 1028  2     EXTERNAL ROUTINE
;  941        U 1029  2         MONITOR_I:                              ! Print an integer
;  942          1030  2         MONITOR_T;                              ! Print text
;  943        U 1031  2
;  944          1032  2 %FI
;  94           1033  2
;  946          1034  2 !+
;  947          1035  2 ! Enable a handler to free the local strings in case of error.
;  948          1036  2 !-
;  949          1037  2
;  950          1038  2     ENABLE
;  951          1039  2         FREE_STRINGS (A_DESC, B_DESC, C_DESC);
;  952          1040  2
;  953          1041  2 !+
;  954          1042  2 ! Convert the two input arguments from external form to internal form.
;  955          1043  2 ! This is done by removing the non-digits from the string and
;  956          1044  2 ! returning the sign and exponent as separate values.
;  957          1045  2 ! Errors are signaled.
;  958          1046  2 !-
;  959          1047  2     A_DESC [DSC$W_LENGTH] = 0;
;  960          1048  2     A_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;  961          1049  2     A_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  962          1050  2     A_DESC [DSC$A_POINTER] = 0;
;  963          1051  2     PARSE_IN (.OP1_DESC, A_DESC, A_SIGN, A_EXP);
;  964          1052  2
;  965          1053  2     B_DESC [DSC$W_LENGTH] = 0;
;  966          1054  2     B_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;  967          1055  2     B_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  968          1056  2     B_DESC [DSC$A_POINTER] = 0;
;  969          1057  2     PARSE_IN (.OP2_DESC, B_DESC, B_SIGN, B_EXP);
;  970          1058  2
;  971          1059  2 !+
;  972          1060  2 ! Set up descriptor for internal form of C
;  973          1061  2 !-
;  974          1062  2
;  975          1063  2     C_DESC [DSC$W_LENGTH] = 0;
;  976          1064  2     C_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;  977          1065  2     C_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  978          1066  2     C_DESC [DSC$A_POINTER] = 0;
;  979          1067  2
;  980          1068  2 !+
;  981          1069  2 ! Set up parameters for call to STR$DIVIDE
;  982          1070  2 !-
;  983          1071  2
;  984          1072  3     IF (.PRECISION LEQ BAS$K_PREC_VAL)
;  985          1073  2     THEN
;  986          1074  3         BEGIN
;  987          1075  3         RND_TRUNC = 1;
;  988          1076  3         PREC = .PRECISION;
```

```
;  989          1077  3          END
;  990          1078  2      ELSE
;  991          1079  3          BEGIN
;  992          1080  3          RND_TRUNC = 0;
;  993          1081  3          PREC = .PRECISION - 2*BAS$K_PREC_VAL;
;  994          1082  2          END;
;  995          1083  2
;  996          1084  2  !+
;  997          1085  2  ! For compatability with the PDP-11, don't allow more than
;  998          1086  2  ! 55 or 56 digits to the right of the decimal point.
;  999          1087  2  !-
; 1000          1088  2
; 1001          1089  3      IF .C_DESC[DSC$W_LENGTH] GTR (BAS$K_PREC_LIM2 - .RND_TRUNC)
; 1002          1090  2      THEN
; 1003          1091  2          BAS$$SIGNAL (BAS$K_FLOPOIERR);
; 1004          1092  2
; 1005          1093  2
; 1006          1094  2  !+
; 1007          1095  2  ! The next section of code is to interface with the new routine STR$DIVIDE.
; 1008          1096  2  !-
; 1009          1097  2
; 1010          1098  2      STR$DIVIDE (A_SIGN,A_EXP,A_DESC,
; 1011          1099  2                  B_SIGN,B_EXP,B_DESC,
; 1012          1100  2                  PREC,RND_TRUNC,
; 1013          1101  2                  C_SIGN,C_EXP,C_DESC);          ! Invoke STR$DIVIDE
; 1014        L 1102  2  %IF PERFORM_MONITORING
; 1015        U 1103  2  %THEN
; 1016        U 1104  2      MONITOR_I (%REF (3), %REF (.ROUND_POS));
; 1017          1105  2  %FI
; 1018          1106  2
; 1019          1107  2  !+
; 1020          1108  2  ! The following code is to ensure compatibility with the PDP-11.
; 1021          1109  2  ! Need to strip off any leading zeroes or any trailing zeroes.
; 1022          1110  2  !-
; 1023          1111  2
; 1024          1112  2  !+
; 1025          1113  2  ! save descriptor for C because the following code modifies it and it must
; 1026          1114  2  ! be restored before calling STR$FREE1_DX.
; 1027          1115  2  !-
; 1028          1116  2      SAV_C_LEN = .C_DESC [DSC$W_LENGTH];
; 1029          1117  2      SAV_C_PTR = .C_DESC [DSC$A_POINTER];
; 1030          1118  2
; 1031          1119  2  !+
; 1032          1120  2  ! Initial set-up.  Set-up pointers to various sections within
; 1033          1121  2  ! the resultant string.
; 1034          1122  2  !-
; 1035          1123  2
; 1036          1124  2      BUF = .C_DESC[DSC$A_POINTER];                  ! Fetch addr of buffer
; 1037          1125  2      C_LENGTH = .C_DESC[DSC$W_LENGTH];              ! Fetch length of string
; 1038          1126  2      END_ADDR = .BUF + .C_LENGTH;                   ! Ptr to end of buffer
; 1039          1127  2      SAV_EXP = .C_EXP;                              ! Save the exp value
; 1040          1128  2      SAV_LENGTH = .C_LENGTH;                        ! Save the length of C
; 1041          1129  2      SAV_BUF = .BUF;                                ! Save ptr to beg of buffer
; 1042          1130  2
; 1043          1131  2  !+
; 1044          1132  2  ! First check if the exponent value if less than the length of
; 1045          1133  2  ! the string.  If it is, then we know that we have digits to the
```

```
;  1046     1134  2 ! left of the decimal point.
;  1047     1135  2 !-
;  1048     1136  2
;  1049     1137  2 IF (.C_EXP) LSS 0
;  1050     1138  2   THEN
;  1051     1139  3     BEGIN
;  1052     1140  3     IF ABS(.C_EXP) LSS .C_LENGTH
;  1053     1141  3       THEN
;  1054     1142  4         BEGIN
;  1055     1143  4         LENGTH = ABS (.C_EXP);
;  1056     1144  4         BUF = .BUF + (.C_LENGTH + .C_EXP);
;  1057     1145  4         END
;  1058     1146  3       ELSE
;  1059     1147  3         LENGTH = .C_LENGTH;
;  1060     1148  3
;  1061     1149  3 !+
;  1062     1150  3 ! Check for first non-zero digit to right of decimal point.
;  1063     1151  3 !-
;  1064     1152  3
;  1065     1153  3     NEW_ADDR = SPANC (LENGTH,.BUF,TABLE_NZ,MASK);
;  1066     1154  3     IF .NEW_ADDR EQL 0                              !\Indicates all zeroes to
;  1067     1155  3                                                     !/right of decimal point
;  1068     1156  3       THEN
;  1069     1157  4         BEGIN
;  1070     1158  4         IF ABS(.C_EXP) GEQ .C_LENGTH
;  1071     1159  4           THEN
;  1072     1160  5             BEGIN                                   ! Here, length = exponent value
;  1073     1161  5             C_LENGTH = 1;                           ! Just set length to one
;  1074     1162  5             C_EXP = 0;                              ! Set exp of result to zero
;  1075     1163  5             END
;  1076     1164  4           ELSE
;  1077     1165  5             BEGIN                                   ! Here, exponent < length
;  1078     1166  5             C_LENGTH = .C_LENGTH - ABS(.C_EXP);     ! Decr C_LENGTH by the exp value
;  1079     1167  5             C_EXP = 0;                              ! Set exp of result to zero
;  1080     1168  4             END;
;  1081     1169  4         C_DESC[DSC$W_LENGTH] = .C_LENGTH;           ! Store updated length
;  1082     1170  4         END
;  1083     1171  3       ELSE
;  1084     1172  3         !+
;  1085     1173  3         ! At this point, we have the address of the first non-zero
;  1086     1174  3         ! position within the resultant string.  Now we need to search
;  1087     1175  3         ! for a zero digit.  If none, then we know that all the digits
;  1088     1176  3         ! to the right of the decimal point are non-zero, and we can
;  1089     1177  3         ! leave the exponent value and length value as is.
;  1090     1178  3         !-
;  1091     1179  4         BEGIN
;  1092     1180  4         !+
;  1093     1181  4         ! Start searching for a zero digit, starting at the address
;  1094     1182  4         ! returned from the previous SPANC.
;  1095     1183  4         !-
;  1096     1184  4
;  1097     1185  4         IF .NEW_ADDR NEQ .END_ADDR                          ! If not at end of string
;  1098     1186  4           THEN
;  1099     1187  5             BEGIN
;  1100     1188  5             WHILE 1 DO
;  1101     1189  6               BEGIN
;  1102     1190  6               SAV_ADDR = .NEW_ADDR;                       ! Save the current addr
```

```
;  1103          1191  6          LENGTH = .END_ADDR - .NEW_ADDR;                 ! Calculate new length
;  1104          1192  6          NEW_ADDR = SPANC (LENGTH,.NEW_ADDR,TABLE_Z,MASK);
;  1105          1193  6          IF .NEW_ADDR NEQ 0                              !\Indicates a zero
;  1106          1194  6                                                          !/digit has been found
;  1107          1195  6            THEN
;  1108          1196  7              BEGIN
;  1109          1197  7              !+
;  1110          1198  7              ! Now search for a non-zero digit.
;  1111          1199  7              !-
;  1112          1200  7              SAV_ADDR = .NEW_ADDR;                       ! Save current address
;  1113          1201  7              LENGTH = .END_ADDR - .NEW_ADDR;             ! Calculate length of string to be search
;  1114          1202  7              NEW_ADDR = SPANC (LENGTH,.NEW_ADDR,TABLE_NZ,MASK);
;  1115          1203  7              IF .NEW_ADDR EQL 0                          ! No match found
;  1116          1204  7                THEN
;  1117          1205  8                  BEGIN
;  1118          1206  8                  DIFF = .END_ADDR - .SAV_ADDR;           !\Calculate length of
;  1119          1207  8                                                          !/string that is zero
;  1120          1208  8                  C_LENGTH = .C_LENGTH - .DIFF;           ! Decr length by that amount
;  1121          1209  8                  C_EXP = - (ABS(.C_EXP) - .DIFF);        ! Decr exponent value by that amount
;  1122          1210  8                  C_DESC[DSC$W_LENGTH] = .C_LENGTH;       ! Store updated length
;  1123          1211  8                  EXITLOOP                                ! Escape from loop
;  1124          1212  7                  END;
;  1125          1213  7                END
;  1126          1214  6              ELSE
;  1127          1215  6                EXITLOOP
;  1128          1216  5            END;
;  1129          1217  4          END;
;  1130          1218  3        END;
;  1131          1219  2      END;
;  1132          1220  3      IF (.SAV_EXP LEQ 0) AND (ABS(.SAV_EXP) LSS .SAV_LENGTH)
;  1133          1221  2       THEN
;  1134          1222  3         BEGIN
;  1135          1223  3           LOCAL
;  1136          1224  3             LEFT_DIGITS;
;  1137          1225  3
;  1138          1226  3           !+
;  1139          1227  3           ! Here we know that C_EXP is less than C_LENGTH.
;  1140          1228  3           ! Now we are ready to strip off leading zeroes that
;  1141          1229  3           ! may exist to the left side of the decimal point.
;  1142          1230  3           !-
;  1143          1231  3
;  1144          1232  3           LEFT_DIGITS = .C_LENGTH - ABS(.C_EXP);          !\Calc # of digits to
;  1145          1233  3                                                           !/left of decimal point
;  1146          1234  3           NEW_ADDR = SPANC (LEFT_DIGITS,.SAV_BUF,TABLE_NZ,MASK); !\Search for
;  1147          1235  3                                                           !/non-zero
;  1148          1236  3           IF .NEW_ADDR EQL 0                              ! All zeroes to left of D.P.
;  1149          1237  3             THEN
;  1150          1238  4               BEGIN
;  1151          1239  4               C_LENGTH = .C_LENGTH - .LEFT_DIGITS;      ! Decr C_LENGTH
;  1152          1240  4               IF .C_LENGTH EQL 0
;  1153          1241  4                 THEN
;  1154          1242  4                   C_LENGTH = 1                          ! Ensure length of 1
;  1155          1243  4                 ELSE
;  1156          1244  4                   C_DESC[DSC$A_POINTER] = .SAV_BUF + .LEFT_DIGITS; !Update buffer address
;  1157          1245  4               END
;  1158          1246  3             ELSE
;  1159          1247  4               BEGIN
```

```
;1160     1248  4                DIFF = .NEW_ADDR - .SAV_BUF;                    ! Calculate offset
;1161     1249  4                C_LENGTH = .C_LENGTH - .DIFF;                   ! Update length
;1162     1250  4                C_DESC[DSC$A_POINTER] = .SAV_BUF + .DIFF;! Update buffer address
;1163     1251  3                END;
;1164     1252  3              C_DESC[DSC$W_LENGTH] = .C_LENGTH;
;1165     1253  3              END
;1166     1254  2            ELSE
;1167     1255  2              IF .SAV_EXP GTR 0
;1168     1256  2                THEN
;1169     1257  2            !+
;1170     1258  2            ! Here we know that the exponent value is positive.
;1171     1259  2            !-
;1172     1260  3              BEGIN
;1173     1261  3              NEW_ADDR = SPANC (C_LENGTH,.SAV_BUF,TABLE_NZ,MASK);
;1174     1262  3              IF .NEW_ADDR EQL 0                                  !\Indicates all zeroes
;1175     1263  3                                                                  !/to left of decimal pt
;1176     1264  3                THEN
;1177     1265  4                  BEGIN
;1178     1266  4                  C_LENGTH = 1;                                 ! C_LENGTH = 1
;1179     1267  4                  C_EXP = 0;                                    ! Set exponent to zero
;1180     1268  4                  END
;1181     1269  3                ELSE
;1182     1270  4                  BEGIN
;1183     1271  4                  DIFF = .NEW_ADDR - .SAV_BUF;
;1184     1272  4                  C_LENGTH = .C_LENGTH - .DIFF;
;1185     1273  4                  C_DESC[DSC$A_POINTER] = .SAV_BUF + .DIFF;
;1186     1274  3                  END;
;1187     1275  3              C_DESC[DSC$W_LENGTH] = .C_LENGTH;
;1188     1276  2              END;
;1189     1277  2
;1190     1278  2
;1191     1279  2 !+
;1192     1280  2 ! Convert the quotient to external form for the caller.
;1193     1281  2 !-
;1194     1282  2     PARSE_OUT (C_DESC, .C_SIGN, .C_EXP, .QUO_DESC);
;1195     1283  2
;1196     1284  2 !+
;1197     1285  2 ! restore actual length & address of C so we don't fake out STR$FREE1 and cause
;1198     1286  2 ! an access violation
;1199     1287  2 !-
;1200     1288  2     C_DESC [DSC$W_LENGTH] = .SAV_C_LEN;
;1201     1289  2     C_DESC [DSC$A_POINTER] = .SAV_C_PTR;
;1202     1290  2
;1203     1291  2 !+
;1204     1292  2 ! We can now free A, B and C
;1205     1293  2 !-
;1206     1294  2     STR$FREE1_DX (A_DESC);
;1207     1295  2     STR$FREE1_DX (B_DESC);
;1208     1296  2     STR$FREE1_DX (C_DESC);
;1209     1297  2     RETURN;
;1210     1298  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  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$:   ADDL3   LEFT_DIGITS, SAV_BUF, C_DESC+4              : 1244
                                            2A 11 001BE        BRB     25$                                         : [illegible]
                                            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$:   PUSHL   QOO_DESC                                    : 1282
                                       10   AE DD 001F1        PUSHL   C_EXP                                       :
                                       18   AE DD 001F4        PUSHL   C_SIGN                                      :
                                       38   AE 9F 001F7        PUSHAB  C_DESC                                      :
                         0000V    CF        04 FB 001FA        CALLS   #4, PARSE_OUT                               :
                            2C    AE   08   AE B0 001FF        MOVW    SAV_C_LEN, C_DESC                           : 1288
                            30    AE   04   AE D0 00204        MOVL    SAV_C_PTR, C_DESC+4                         : 1289
                                       3C   AE 9F 00209        PUSHAB  A_DESC                                      : 1294
                     00000000G    00        01 FB 0020C        CALLS   #1, STR$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                                                 : [illegible]
                                             0000 00228 27$:   .WORD   Save nothing                                : [illegible]
                                  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
```

```
;  1213          1300  1 GLOBAL ROUTINE BAS$SUM (                          ! Add strings
;  1214          1301  1          SUM_DESC,                                ! Sum
;  1215          1302  1          OP1_DESC,                                ! First input
;  1216          1303  1          OP2_DESC                                 ! Second input
;  1217          1304  1      ) : NOVALUE =
;  1218          1305  1
;  1219          1306  1 !++
;  1220          1307  1 ! FUNCTIONAL DESCRIPTION:
;  1221          1308  1 !
;  1222          1309  1 !       Add two strings.  SUM := OP1 + OP2
;  1223          1310  1 !
;  1224          1311  1 ! FORMAL PARAMETERS:
;  1225          1312  1 !
;  1226          1313  1 !       SUM_DESC.wt.dx  The sum of OP1 and OP2
;  1227          1314  1 !       OP1_DESC.rt.dx  Operand OP1
;  1228          1315  1 !       OP2_DESC.rt.dx  Operand OP2
;  1229          1316  1 !
;  1230          1317  1 ! IMPLICIT INPUTS:
;  1231          1318  1 !
;  1232          1319  1 !       NONE
;  1233          1320  1 !
;  1234          1321  1 ! IMPLICIT OUTPUTS:
;  1235          1322  1 !
;  1236          1323  1 !       NONE
;  1237          1324  1 !
;  1238          1325  1 ! ROUTINE VALUE:
;  1239          1326  1 ! COMPLETION CODES:
;  1240          1327  1 !
;  1241          1328  1 !       NONE
;  1242          1329  1 !
;  1243          1330  1 ! SIDE EFFECTS:
;  1244          1331  1 !
;  1245          1332  1 !       Signals PROSOSSOR for certain "impossible" errors, and the
;  1246          1333  1 !       signals from PARSE_IN and PARSE_OUT.
;  1247          1334  1 !
;  1248          1335  1 !--                                                                                    ;
;  1249          1336  1
;  1250          1337  2     BEGIN
;  1251          1338  2
;  1252          1339  2     MAP
;  1253          1340  2         OP1_DESC : REF BLOCK [8, BYTE],
;  1254          1341  2         OP2_DESC : REF BLOCK [8, BYTE],
;  1255          1342  2         SUM_DESC : REF BLOCK [8, BYTE];
;  1256          1343  2
;  1257          1344  2     LOCAL
;  1258          1345  2 !+
;  1259          1346  2 ! Internal form of OP1.
;  1260          1347  2 !-
;  1261          1348  2         A_DESC : BLOCK [8, BYTE] VOLATILE,
;  1262          1349  2         A_SIGN,
;  1263          1350  2         A_EXP,
;  1264          1351  2 !+
;  1265          1352  2 ! Internal form of OP2.
;  1266          1353  2 !-
;  1267          1354  2         B_DESC : BLOCK [8, BYTE] VOLATILE,
;  1268          1355  2         B_SIGN,
;  1269          1356  2         B_EXP,
```

```
;1270     1357  2 !+
;1271     1358  2 ! Internal form of the sum, OP1 + OP2.
;1272     1359  2 !-
;1273     1360  2         C_DESC : BLOCK [8, BYTE] VOLATILE,
;1274     1361  2         C_SIGN,
;1275     1362  2         C_EXP;
;1276     1363  2
;1277     1364  2 !+
;1278     1365  2 ! Enable a handler to free the local strings in case of error.
;1279     1366  2 !-
;1280     1367  2
;1281     1368  2     ENABLE
;1282     1369  2         FREE_STRINGS (A_DESC, B_DESC, C_DESC);
;1283     1370  2
;1284     1371  2 !+
;1285     1372  2 ! Convert the two input arguments from external form to internal form.
;1286     1373  2 ! This is done by removing the non-digits from the string and
;1287     1374  2 ! returning the sign and exponent as separate values.
;1288     1375  2 ! Errors are signaled.
;1289     1376  2 !-
;1290     1377  2     A_DESC [DSC$W_LENGTH] = 0;
;1291     1378  2     A_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;1292     1379  2     A_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;1293     1380  2     A_DESC [DSC$A_POINTER] = 0;
;1294     1381  2     PARSE_IN (.OP1_DESC, A_DESC, A_SIGN, A_EXP);
;1295     1382  2     B_DESC [DSC$W_LENGTH] = 0;
;1296     1383  2     B_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;1297     1384  2     B_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;1298     1385  2     B_DESC [DSC$A_POINTER] = 0;
;1299     1386  2     PARSE_IN (.OP2_DESC, B_DESC, B_SIGN, B_EXP);
;1300     1387  2 !+
;1301     1388  2 ! Add the numbers using the large-precision string arithmetic package.
;1302     1389  2 !-
;1303     1390  2     C_DESC [DSC$W_LENGTH] = 0;
;1304     1391  2     C_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;1305     1392  2     C_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;1306     1393  2     C_DESC [DSC$A_POINTER] = 0;
;1307     1394  2     STR$ADD (A_SIGN, A_EXP, A_DESC, B_SIGN, B_EXP, B_DESC, C_SIGN, C_EXP, C_DESC);
;1308     1395  2 !+
;1309     1396  2 ! We are done with A and B, so free them.
;1310     1397  2 !-
;1311     1398  2     STR$FREE1_DX (A_DESC);
;1312     1399  2     STR$FREE1_DX (B_DESC);
;1313     1400  2 !+
;1314     1401  2 ! Convert the sum to external form for the caller.
;1315     1402  2 !-
;1316     1403  2     PARSE_OUT (C_DESC, .C_SIGN, .C_EXP, .SUM_DESC);
;1317     1404  2 !+
;1318     1405  2 ! We can now free C
;1319     1406  2 !-
;1320     1407  2     STR$FREE1_DX (C_DESC);
;1321     1408  2     RETURN;
;1322     1409  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  A0  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
```

```
;  1325          1411  1 ROUTINE PARSE_IN (                          ! Scan a number and divide it up
;  1326          1412  1         ARG_DESC,                           ! The number to scan
;  1327          1413  1         DIGITS,                             ! Where to put the digits
;  1328          1414  1         SIGN,                               ! Where to put the sign
;  1329          1415  1         EXPONENT                            ! Where to put the power of ten
;  1330          1416  1     ) : NOVALUE =
;  1331          1417  1
;  1332          1418  1 !++
;  1333          1419  1 ! FUNCTIONAL DESCRIPTION:
;  1334          1420  1 !
;  1335          1421  1 !       Convert a numeric string of the form +nnn.nnn into a digit
;  1336          1422  1 !       string and a separate sign and decimal exponent.  Don't allow
;  1337          1423  1 !       more than 60 digits, for compatability with the PDP-11.
;  1338          1424  1 !
;  1339          1425  1 ! FORMAL PARAMETERS:
;  1340          1426  1 !
;  1341          1427  1 !       ARG_DESC.rt.dx  The number to parse
;  1342          1428  1 !       DIGITS.wnu.dx   Where to put the digits found
;  1343          1429  1 !       SIGN.wl.r       0 = positive, 1 = negative
;  1344          1430  1 !       EXPONENT.wl.r   Decimal exponent
;  1345          1431  1 !
;  1346          1432  1 ! IMPLICIT INPUTS:
;  1347          1433  1 !
;  1348          1434  1 !       NONE
;  1349          1435  1 !
;  1350          1436  1 ! IMPLICIT OUTPUTS:
;  1351          1437  1 !
;  1352          1438  1 !       NONE
;  1353          1439  1 !
;  1354          1440  1 ! ROUTINE VALUE:
;  1355          1441  1 ! COMPLETION CODES:
;  1356          1442  1 !
;  1357          1443  1 !       NONE
;  1358          1444  1 !
;  1359          1445  1 ! SIDE EFFECTS:
;  1360          1446  1 !
;  1361          1447  1 !       Signals BAS$_ILLNUM if there are more than 60 digit,
;  1362          1448  1 !       BAS$_DATFORERR if the syntax of the number is wrong,
;  1363          1449  1 !       and also signals if storage is exhausted.
;  1364          1450  1 !
;  1365          1451  1 !--
;  1366          1452  1
;  1367          1453  2     BEGIN
;  1368          1454  2
;  1369          1455  2     MAP
;  1370          1456  2         ARG_DESC : REF BLOCK [8, BYTE],
;  1371          1457  2         DIGITS : REF BLOCK [8, BYTE];
;  1372          1458  2
;  1373          1459  2     LOCAL
;  1374          1460  2 !+
;  1375          1461  2 ! The following three locals hold the internal form of the number.
;  1376          1462  2 !-
;  1377          1463  2         SIGN_VAL,
;  1378          1464  2         BUF_DESC : BLOCK [8, BYTE] VOLATILE,
;  1379          1465  2         EXPON,
;  1380          1466  2         SIGN_SEEN,                          ! 1 = we have scanned a + or -
;  1381          1467  2         DIGIT_SEEN,                         ! 1 = we have seen at least one digit
```

```
;  1382          1468  2          DOT_SEEN,                       ! 1 = we have seen a decimal point
;  1383          1469  2          BLANKS_SEEN,                    ! 1 = we have seen trailing blanks
;  1384          1470  2          PUTTER,                         ! Counts position in the output buffer
;  1385          1471  2          BUF : REF VECTOR [65535, BYTE], ! Addresses result
;  1386          1472  2          ARG : REF VECTOR [65535, BYTE], ! Addresses source
;  1387          1473  2          ARG_LEN;                        ! Length of the source
;  1388          1474  2
;  1389          1475  2      !+
;  1390          1476  2      ! Enable a handler to free the local string in case of an error.
;  1391          1477  2      !-
;  1392          1478  2
;  1393          1479  2          ENABLE
;  1394          1480  2              FREE_STRINGS (BUF_DESC);
;  1395          1481  2
;  1396          1482  2      !+
;  1397          1483  2      ! Allocate enough space to hold the digits.  It is convenient to
;  1398          1484  2      ! allocate before scanning, so we may allocate a little too much,
;  1399          1485  2      ! but the space will be freed before we return.
;  1400          1486  2      ! Note that we must fetch the length field of the descriptor only
;  1401          1487  2      ! once, so that we will not attempt to overrun our local string if
;  1402          1488  2      ! the source is reallocated longer by an AST.
;  1403          1489  2      !-
;  1404          1490  2          BUF_DESC [DSC$W_LENGTH] = 0;
;  1405          1491  2          BUF_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;  1406          1492  2          BUF_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  1407          1493  2          BUF_DESC [DSC$A_POINTER] = 0;
;  1408          1494  2          ARG_LEN = .ARG_DESC [DSC$W_LENGTH];
;  1409          1495  2          STR$GET1_DX (%REF (.ARG_LEN + 8), BUF_DESC);
;  1410          1496  2      !+
;  1411          1497  2      ! Now scan the incoming number.
;  1412          1498  2      !-
;  1413          1499  2          PUTTER = 0;
;  1414          1500  2          BUF = .BUF_DESC [DSC$A_POINTER];
;  1415          1501  2          ARG = .ARG_DESC [DSC$A_POINTER];
;  1416          1502  2          SIGN_SEEN = 0;
;  1417          1503  2          SIGN_VAL = 0;
;  1418          1504  2          DIGIT_SEEN = 0;
;  1419          1505  2          DOT_SEEN = 0;
;  1420          1506  2          BLANKS_SEEN = 0;
;  1421          1507  2          EXPON = 0;
;  1422          1508  2
;  1423          1509  2          INCR GETTER FROM 0 TO (.ARG_LEN - 1) DO
;  1424          1510  2
;  1425          1511  2              SELECTONE .ARG [.GETTER] OF
;  1426          1512  2                  SET
;  1427          1513  2
;  1428          1514  2                  [%C'+'] :
;  1429          1515  3                      BEGIN                           ! Plus sign
;  1430          1516  3
;  1431          1517  3                      IF (.SIGN_SEEN OR .DIGIT_SEEN OR .DOT_SEEN OR .BLANKS_SEEN) THEN BAS$$STOP (BAS$K_DATFORERR)
;  1432          1518  3
;  1433          1519  3                      SIGN_SEEN = 1;
;  1434          1520  3                      SIGN_VAL = 0;
;  1435          1521  2                      END;
;  1436          1522  2
;  1437          1523  2                  [%C'-'] :
;  1438          1524  3                      BEGIN                           ! Minus sign
```

```
;  1439          1525  3
;  1440          1526  3                 IF (.SIGN_SEEN OR .DIGIT_SEEN OR .DOT_SEEN OR .BLANKS_SEEN) THEN BAS$$STOP (BAS$K_DATFORERR)
;  1441          1527  3
;  1442          1528  3                 SIGN_SEEN = 1;
;  1443          1529  3                 SIGN_VAL = 1;
;  1444          1530  3                 END;
;  1445          1531  2
;  1446          1532  2             [%C'.'] :
;  1447          1533  2                 BEGIN                           ! Decimal point
;  1448          1534  2
;  1449          1535  2                 IF (.DOT_SEEN OR .BLANKS_SEEN) THEN BAS$$STOP (BAS$K_DATFORERR);
;  1450          1536  2
;  1451          1537  2                 DOT_SEEN = 1;
;  1452          1538  2                 END;
;  1453          1539  2
;  1454          1540  2             [%C' '] :
;  1455          1541  2                 BEGIN                           ! Blank, better be leading or trailing.
;  1456          1542  2
;  1457          1543  2                 IF (.SIGN_SEEN OR .DIGIT_SEEN OR .DOT_SEEN) THEN BLANKS_SEEN = 1;
;  1458          1544  2
;  1459          1545  2                 END;
;  1460          1546  2
;  1461          1547  2             [%C'0' TO %C'9'] :
;  1462          1548  2                 BEGIN                           ! Decimal digit
;  1463          1549  2
;  1464          1550  2                 IF (.BLANKS_SEEN) THEN BAS$$STOP (BAS$K_DATFORERR);
;  1465          1551  2
;  1466          1552  2 !               IF (.DIGIT_SEEN OR .DOT_SEEN OR (.SIGN_VAL EQL 1) OR
;  1467          1553  2 !                  (.ARG [.GETTER] NEQ %C'0'))
;  1468          1554  2 !               THEN
;  1469          1555  3 !                   BEGIN
;  1470          1556  3 !+
;  1471          1557  3 ! This is not a leading zero
;  1472          1558  3 !-
;  1473          1559  3                     DIGIT_SEEN = 1;
;  1474          1560  3                     BUF [.PUTTER] = .ARG [.GETTER];
;  1475          1561  3                     PUTTER = .PUTTER + 1;
;  1476          1562  3
;  1477          1563  3                     IF (.DOT_SEEN) THEN EXPON = .EXPON - 1;
;  1478          1564  3
;  1479          1565  3                     END;
;  1480          1566  3
;  1481          1567  2                 END;
;  1482          1568  2
;  1483          1569  2             [OTHERWISE] :
;  1484          1570  2                 BAS$$STOP (BAS$K_DATFORERR);
;  1485          1571  2             TES;
;  1486          1572  2
;  1487          1573  2 !+
;  1488          1574  2 ! This is the end of the INCR loop.
;  1489          1575  2 !-
;  1490          1576  2
;  1491          1577  3         IF ( NOT .DIGIT_SEEN)
;  1492          1578  2         THEN
;  1493          1579  3             BEGIN
;  1494          1580  3 !+
;  1495          1581  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


                                OFFC 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$:      MOVZBL  (GETTER)[ARG], R2                       ;
                28          52  91 0004F          CMPB    R2, #43                                 ; 1514
                            1E  12 00052          BNEQ    4$                                      ;
                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
```

```
;  1525          1610  1 ROUTINE PARSE_OUT (                          ! Put together a number
;  1526          1611  1         DIGITS,                              ! Where to find the digits
;  1527          1612  1         SIGN,                                ! Where to find the sign
;  1528          1613  1         EXPONENT,                            ! Where to find the power of ten
;  1529          1614  1         ARG_DESC                             ! Where to put the result
;  1530          1615  1     ) : NOVALUE =
;  1531          1616  1
;  1532          1617  1 !++
;  1533          1618  1 ! FUNCTIONAL DESCRIPTION:
;  1534          1619  1 !
;  1535          1620  1 !       Convert an internal number, which has separate sign, exponent
;  1536          1621  1 !       and digit string, into an external number, which combines them
;  1537          1622  1 !       in the form +nnn.nnn.
;  1538          1623  1 !
;  1539          1624  1 ! FORMAL PARAMETERS:
;  1540          1625  1 !
;  1541          1626  1 !       DIGITS.rnu.dx   Where to find the digits
;  1542          1627  1 !       SIGN.rl.r       0 = positive, 1 = negative
;  1543          1628  1 !       EXPONENT.wl.r   Decimal exponent
;  1544          1629  1 !       ARG_DESC.wt.dx  Where to put the composite number
;  1545          1630  1 !
;  1546          1631  1 ! IMPLICIT INPUTS:
;  1547          1632  1 !
;  1548          1633  1 !       NONE
;  1549          1634  1 !
;  1550          1635  1 ! IMPLICIT OUTPUTS:
;  1551          1636  1 !
;  1552          1637  1 !       NONE
;  1553          1638  1 !
;  1554          1639  1 ! ROUTINE VALUE:
;  1555          1640  1 ! COMPLETION CODES:
;  1556          1641  1 !
;  1557          1642  1 !       Signals PROLOSSOR for certain "impossible" errors, and MAXMEMEXC
;  1558          1643  1 !       if string memory is exhausted.
;  1559          1644  1 !
;  1560          1645  1 ! SIDE EFFECTS:
;  1561          1646  1 !
;  1562          1647  1 !       Signals BAS$_ILLNUM if there are more than 65,535 digits,
;  1563          1648  1 !       BAS$_MAXMEMEXC if space for the string is unobtainable,
;  1564          1649  1 !       and BAS$_PROLOSSOR if we are unable to free the string.
;  1565          1650  1 !
;  1566          1651  1 !--
;  1567          1652  1
;  1568          1653  2     BEGIN
;  1569          1654  2
;  1570          1655  2     MAP
;  1571          1656  2         ARG_DESC : REF BLOCK [8, BYTE],
;  1572          1657  2         DIGITS : REF BLOCK [8, BYTE];
;  1573          1658  2
;  1574          1659  2     LOCAL
;  1575          1660  2         BUF_DESC : BLOCK [8, BYTE] VOLATILE,     ! Describes result
;  1576          1661  2         PUTTER,                                  ! Counts positions in BUF (below)
;  1577          1662  2         BUF : REF VECTOR [65535, BYTE],          ! Addresses result string
;  1578          1663  2         DIGIT_LEN,                               ! Length of input string
;  1579          1664  2         DIGIT_BUF : REF VECTOR [65535, BYTE],    ! Addresses input string
;  1580          1665  2         HIGH_POS,                                ! Max power of ten in result
;  1581          1666  2         LOW_POS;                                 ! Min power of ten in result
```

```
;  1582          1667  2
;  1583          1668  2 !+
;  1584          1669  2 ! Enable a handler to free the local string in case of error.
;  1585          1670  2 !-
;  1586          1671  2
;  1587          1672  2     ENABLE
;  1588          1673  2         FREE_STRINGS (BUF_DESC);
;  1589          1674  2
;  1590          1675  2 !+
;  1591          1676  2 ! Allocate enough space to hold the digits, a leading sign, and an
;  1592          1677  2 ! imbedded (or trailing) decimal point.  We must allocate enough
;  1593          1678  2 ! space to reach the units position in order to be able to place
;  1594          1679  2 ! the decimal point.
;  1595          1680  2 !-
;  1596          1681  2     DIGIT_LEN = .DIGITS [DSC$W_LENGTH];
;  1597          1682  2     DIGIT_BUF = .DIGITS [DSC$A_POINTER];
;  1598          1683  2     HIGH_POS = MAX (.DIGIT_LEN + .EXPONENT - 1, -1);
;  1599          1684  2     LOW_POS = MIN (.EXPONENT, 0);
;  1600          1685  2     BUF_DESC [DSC$W_LENGTH] = 0;
;  1601          1686  2     BUF_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  1602          1687  2     BUF_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  1603          1688  2     BUF_DESC [DSC$A_POINTER] = 0;
;  1604          1689  2 !+
;  1605          1690  2 ! If the resultant number has too many digits to be represented on
;  1606          1691  2 ! VAX, give an error message.
;  1607          1692  2 !-
;  1608          1693  2
;  1609          1694  2     IF ((.HIGH_POS - .LOW_POS + 3) GTR 65535) THEN BAS$$STOP (BAS$K_ILLNUM);
;  1610          1695  2
;  1611          1696  2     STR$GET1_DX (%REF (.HIGH_POS - .LOW_POS + 3), BUF_DESC);
;  1612          1697  2     BUF = .BUF_DESC [DSC$A_POINTER];
;  1613          1698  2     PUTTER = 0;
;  1614          1699  2
;  1615          1700  3     IF (.SIGN)
;  1616          1701  2     THEN
;  1617          1702  3         BEGIN
;  1618          1703  3         BUF [.PUTTER] = %C'-';
;  1619          1704  3         PUTTER = .PUTTER + 1;
;  1620          1705  2         END;
;  1621          1706  2
;  1622          1707  2     DECR POS FROM .HIGH_POS TO .LOW_POS DO
;  1623          1708  3         BEGIN
;  1624          1709  3
;  1625          1710  4         IF (.POS EQL -1)
;  1626          1711  3         THEN
;  1627          1712  4             BEGIN
;  1628          1713  4             BUF [.PUTTER] = %C'.';
;  1629          1714  4             PUTTER = .PUTTER + 1;
;  1630          1715  3             END;
;  1631          1716  3
;  1632          1717  4         IF ((.POS GTR (.EXPONENT + .DIGIT_LEN - 1)) OR (.POS LSS .EXPONENT))
;  1633          1718  3         THEN
;  1634          1719  3             BUF [.PUTTER] = %C'0'
;  1635          1720  3         ELSE
;  1636          1721  3             BUF [.PUTTER] = .DIGIT_BUF [(.EXPONENT + .DIGIT_LEN - 1) - .POS];
;  1637          1722  3
;  1638          1723  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 0000B        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
```

```
;1652     1736  1 ROUTINE FREE_STRINGS (                        ! Free local strings
;1653     1737  1         SIG,                                  ! Signal vector
;1654     1738  1         MECH,                                 ! Mechanism vector
;1655     1739  1         ENBL                                  ! Enable vector
;1656     1740  1     ) =
;1657     1741  1
;1658     1742  1 !++
;1659     1743  1 ! FUNCTIONAL DESCRIPTION:
;1660     1744  1 !
;1661     1745  1 !       If we are unwinding, free the local strings.  They are passed
;1662     1746  1 !       in the enable vector.
;1663     1747  1 !
;1664     1748  1 ! FORMAL PARAMETERS:
;1665     1749  1 !
;1666     1750  1 !       SIG.rl.a        A counted vector of parameters to LIB$SIGNAL/STOP
;1667     1751  1 !       MECH.rl.a       A counted vector of info from CHF
;1668     1752  1 !       ENBL.ra.a       A counted vector of ENABLE argument addresses.
;1669     1753  1 !
;1670     1754  1 ! IMPLICIT INPUTS:
;1671     1755  1 !
;1672     1756  1 !       NONE
;1673     1757  1 !
;1674     1758  1 ! IMPLICIT OUTPUTS:
;1675     1759  1 !
;1676     1760  1 !       NONE
;1677     1761  1 !
;1678     1762  1 ! ROUTINE VALUE:
;1679     1763  1 ! COMPLETION CODES:
;1680     1764  1 !
;1681     1765  1 !       Always SS$_RESIGNAL, which is ignored when unwinding.
;1682     1766  1 !
;1683     1767  1 ! SIDE EFFECTS:
;1684     1768  1 !
;1685     1769  1 !       Frees all of the strings passed as enable arguments.
;1686     1770  1 !
;1687     1771  1 !--
;1688     1772  1
;1689     1773  2     BEGIN
;1690     1774  2
;1691     1775  2     MAP
;1692     1776  2         SIG : REF VECTOR,
;1693     1777  2         MECH : REF VECTOR,
;1694     1778  2         ENBL : REF VECTOR;
;1695     1779  2
;1696     1780  2 !+
;1697     1781  2 ! Only free the strings if this is the UNWIND condition.
;1698     1782  2 !-
;1699     1783  2
;1700     1784  2     IF ( NOT (LIB$MATCH_COND (SIG [1], %REF (SS$_UNWIND)))) THEN RETURN (SS$_RESIGNAL);
;1701     1785  2
;1702     1786  2 !+
;1703     1787  2 ! Go through the enable arguments, freeing them.
;1704     1788  2 !-
;1705     1789  2
;1706     1790  2     INCR ARG_NO FROM 1 TO .ENBL [0] DO
;1707     1791  2
;1708     1792  2         IF (..ENBL [.ARG_NO] NEQ 0) THEN STR$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
```