D 8

```
LLL                 IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                 IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                 IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III        BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                    III        BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                    III        BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                    III        BBB         BBB  RRR   RRR              TTT        LLL
LLL                    III        BBB         BBB  RRR   RRR              TTT        LLL
LLL                    III        BBB         BBB  RRR   RRR              TTT        LLL
LLL                    III        BBB         BBB  RRR      RRR           TTT        LLL
LLL                    III        BBB         BBB  RRR      RRR           TTT        LLL
LLL                    III        BBB         BBB  RRR      RRR           TTT        LLL
LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL     IIIIIIIII     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

H 10

```
**FILE**ID**STRARITH

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

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

```
0 MODULE STR$ARITH (                            !
0                 IDENT = '1-019'               ! File: STRARITH.B32 EDIT:STAN1019
0                 ) =
1 BEGIN
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! FACILITY:  STRING Arithmetic
1 !
1 ! ABSTRACT:
1 !
1 !       This module is a large-precision arithmetic package based on
1 !       decimal strings.
1 !
1 ! ENVIRONMENT:  VAX-11 User Mode
1 !
1 ! AUTHOR: John Sauter, CREATION DATE: 01-MAR-1979
1 !
1 ! MODIFIED BY:
1 !
1 ! 1-001 - Original.  JBS 05-MAR-1979
1 ! 1-002 - Fix reciprocal of numbers between 0 and 1.  JBS 07-MAR-1979
1 ! 1-003 - Treat minus 0 as zero.  JBS 22-MAR-1979
1 ! 1-004 - Improve comments based on the code review.  JBS 26-MAR-1979
1 ! 1-005 - Free local strings in case of an error.  JBS 07-MAY-1979
1 ! 1-006 - Make the entry points take scalars by reference, in honor
1 !          of the recognition of STR as a facility.  JBS 15-MAY-1979
1 ! 1-007 - Change OTS$S and LIB$S to STR$.  JBS 21-MAY-1979
1 ! 1-008 - Restore some code deleted by mistake in edit 007.
1 !          JBS 22-MAY-1979
1 ! 1-009 - Change calls to STR$COPY.  JBS 16-JUL-1979
1 ! 1-010 - Correct a typo in a comment.  JBS 30-JUL-1979
1 ! 1-011 - When freeing strings after an error, watch out for
1 !          descriptors not yet initialized.  JBS 31-JUL-1979
```

```
1 ! 1-012 - Added a new entry point - STR$DIVIDE.  Also added related
1 !         routines UPDATE_COUNTS,CVT_STR_PACKED and CVT_PACKED_STR.
1 !         Also changed the existing string arithmetic routines, so
1 !         that they all call LIB$ANALYZE_SDESC, to verify that the
1 !         input descriptors are valid.  LB 25-AUG-1981
1 ! 1-013 - Added updated code for STR$DIVIDE as well as ancillary
1 !         routines UPDATE_COUNTS,CVT_STR_PACKED, and CVT_PACKED_STR.
1 !         LB 16-NOV-81
1 ! 1-014 - Moved code to do the conversions to and from packed decimal
1 !         into module LIBPKARIT.  Changed code in STR$DIVIDE to use
1 !         left justification of the input strings instead of right
1 !         justification.  RNH  11-DEC-81.
1 ! 1-015 - Added code in all string arithmetic routines so that they correctly
1 !         handle all string classes.  This required the addition of a new
1 !         internal entry point CHK_STR_TYPE.   LB 15-DEC-81.
1 ! 1-016 - Added code in STR$DIVIDE to zeroize a section of 8 bytes to avoid
1 !         random data being picked up by the associated packed arithmetic
1 !         routines that it calls.  LB 1-APR-82.
1 ! 1-017 - Added code to ensure that a result of zero would always be returned
1 !         as a positive value.  LB 11-APR-82.
1 ! 1-018 - fixed call to LIB$FREE_VM at end of STR$DIVIDE to pass START_BUF
1 !         by reference rather than by value.  Initialize variable STORAGE
1 !         on entry into STR$DIVIDE.  Use its address rather than its contents
1 !         when assigning QSTRBUF in special zero-quotient case.   Allow for
1 !         rounding to possibly propogate another digit into the quotient in
1 !         STR$DIVIDE.  Adjust calculation of BYTES_VM is STR$DIVIDE slightly
1 !         to fix an access violation problem.
1 !         MDL 11-Mar-1983
1 ! 1-019 - Enlargened amount of VM that STR$DIVIDE gets to prevent an
1 !         access violation in certain cases. STAN 18-Jun-1984.
1 !--
1
1 !<BLF/PAGE>
```

```
;     92        0091    1  !+
;     93        0092    1  ! SWITCHES:
;     94        0093    1  !-
;     95        0094    1
;     96        0095    1  SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;     97        0096    1
;     98        0097    1  !+
;     99        0098    1  ! LINKAGES:
;    100        0099    1  !-
;    101        0100    1
;    102        0101    1  LINKAGE
;    103        0102    1          JSB1 = JSB (REGISTER=6, REGISTER=7) : NOPRESERVE (2,3,4,5),
;    104        0103    1          JSB2 = JSB (REGISTER=6, REGISTER=7, REGISTER=8) : NOPRESERVE (2,3,4,5),
;    105        0104    1          JSB3 = JSB (REGISTER=6, REGISTER=7, REGISTER=8, REGISTER=9)
;    106        0105    1                  : NOPRESERVE (2,3,4,5),
;    107        0106    1          JSB4 = JSB (REGISTER=6, REGISTER=7, REGISTER=8, REGISTER=9, REGISTER=10)
;    108        0107    1                  : NOPRESERVE (2,3,4,5);
;    109        0108    1
;    110        0109    1
;    111        0110    1  !+
;    112        0111    1  ! TABLE OF CONTENTS:
;    113        0112    1  !-
;    114        0113    1
;    115        0114    1  FORWARD ROUTINE
;    116        0115    1      STR$ADD : NOVALUE,                          ! Add two strings
;    117        0116    1      STR$MUL : NOVALUE,                          ! Multiply two strings
;    118        0117    1      STR$RECIP : NOVALUE,                        ! Take the reciprocal of a string
;    119        0118    1      STR$ROUND : NOVALUE,                        ! Round a string
;    120        0119    1      STR$DIVIDE: NOVALUE,                        ! Divide two strings
;    121        0120    1      CHK_STR_TYPE:NOVALUE,                       ! Check the string type
;    122        0121    1      FREE_STRINGS;                               ! Free local strings
;    123        0122    1
;    124        0123    1  !+
;    125        0124    1  ! INCLUDE FILES:
;    126        0125    1  !-
;    127        0126    1
;    128        0127    1  REQUIRE 'RTLIN:RTLPSECT';                       ! Macros for defining psects
;    129        0222    1  LIBRARY 'RTLSTARLE';                            ! System definitions
;    130        0223    1
;    131        0224    1  !+
;    132        0225    1  ! MACROS:
;    133        0226    1  !
;    134        0227    1  !       NONE
;    135        0228    1  !-
;    136        0229    1
;    137        0230    1  !+
;    138        0231    1  ! PSECTS:
;    139        0232    1  !-
;    140        0233    1
;    141        0234    1  DECLARE_PSECTS (STR);                           ! Declare psects for STR$ facility
;    142        0235    1
;    143        0236    1  !+
;    144        0237    1  ! OWN STORAGE:
;    145        0238    1  !
;    146        0239    1  !       NONE
;    147        0240    1  !-
;  149        0241  1 !+
;  150        0242  1 ! EXTERNAL REFERENCES:
;  151        0243  1 !-
;  152        0244  1
;  153        0245  1 EXTERNAL ROUTINE
;  154        0246  1         LIB$STOP,                          ! Signal fatal error
;  155        0247  1         STR$GET1_DX,                       ! Allocate a string
;  156        0248  1         STR$FREE1_DX,                      ! Deallocate a string
;  157        0249  1         STR$COPY_R,                        ! Copy a string by reference
;  158        0250  1         STR$COPY_DX,                       ! Copy a string by descriptor
;  159        0251  1         LIB$GET_VM,                        ! Allocate virtual memory
;  160        0252  1         LIB$FREE_VM,                       ! Deallocate virtual memory
;  161        0253  1         LIB$SCOPY_R_DX,                    ! Copy a string by reference
;  162        0254  1         LIB$$ROUND_R7:JSB1 NOVALUE,        ! Rounds quotient to correct length
;  163        0255  1         LIB$$CALC_D_R7:JSB1,               ! Calculates normalization factor
;  164        0256  1         LIB$$CALC_Q_R9:JSB3,               ! Calculates one quotient digit
;  165        0257  1         LIB$$SUB_PACK_R8:JSB2,             ! Subtracts two decimal arrays
;  166        0258  1         LIB$$MUL_PACK_R10:JSB4 NOVALUE,    ! Multiplies a packed array by a single
;  167        0259  1                                            !     entry
;  168        0260  1         LIB$$ADJUST_Q_R9:JSB3 NOVALUE,     ! Adjusts intermediate results of divi-
;  169        0261  1                                            !     sion algorithm if initial quess at
;  170        0262  1                                            !     a quotient digit is wrong
;  171        0263  1         LIB$$CVT_STR_PACK_R9:JSB3 NOVALUE,
;  172        0264  1                                            ! Converts a string of decimal digits
;  173        0265  1                                            !     to an array of packed decimal
;  174        0266  1                                            !     values
;  175        0267  1         LIB$$CVT_PACK_STR_R8:JSB2 NOVALUE,
;  176        0268  1                                            ! Converts an array of packed decimal
;  177        0269  1                                            !     values to a string
;  178        0270  1         LIB$ANALYZE_SDESC,                 ! Extract length and addr of a given
;  179        0271  1                                            !     descriptor and validate inputs
;  180        0272  1         LIB$MATCH_COND,                    ! Match condition codes
;  181        0273  1         STR$DUPL_CHAR;                     ! Used to pad result with leading zeroes
;  182        0274  1
;  183        0275  1 BIND
;  184        0276  1         ZERO = UPLIT BYTE (REP 7 OF (%X'00'),%X'0C'), ! Packed zero
;  185        0277  1         TEN = UPLIT BYTE (REP 6 OF (%X'00'),%X'01',%X'0C'), ! Packed ten
;  186        0278  1         SPANC_TABLE = UPLIT BYTE (REP 48 OF (%X'00'), REP 10 OF (%X'01'),
;  187        0279  1                                   REP 198 OF (%X'00')),
;  188        0280  1         MASK = UPLIT BYTE (REP 1 OF (%X'01'));
;  189        0281  1
;  190        0282  1 BUILTIN
;  191        0283  1         CMPP,                              ! Compare packed decimal data
;  192        0284  1         MOVP,                              ! Move packed decimal data
;  193        0285  1         SPANC;                             ! Skip over a set of characters in a character string
;  194        0286  1
;  195        0287  1 !+
;  196        0288  1 ! The following are the error codes produced by this module.
;  197        0289  1 !-
;  198        0290  1
;  199        0291  1 EXTERNAL LITERAL
;  200        0292  1     LIB$_INVARG,                                   ! Invalid argument
;  201        0293  1     STR$_DIVBY_ZER,                                ! Divide by zero.
;  202        0294  1     STR$_WRONUMARG;                                ! Wrong number of arguments
;  203        0295  1
;  204        0296  1
```

```
1 GLOBAL ROUTINE STR$ADD (                                          ! Add two strings
1         ASIGN,                                                    ! Sign of operand A
1         AEXP,                                                     ! Decimal exponent of operand A
1         ADIGITS,                                                  ! Digits of operand A
1         BSIGN,                                                    ! Sign of operand B
1         BEXP,                                                     ! Decimal exponent of operand B
1         BDIGITS,                                                  ! Digits of operand B
1         CSIGN,                                                    ! Sign of operand C
1         CEXP,                                                     ! Decimal exponent of operand C
1         CDIGITS                                                   ! Digits of operand C
1     ) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Add two decimal numbers.  C := A + B
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       ASIGN.rv.r      0 = operand A is positive, 1 = negative
1 !       AEXP.rl.r       Power of 10 by which to multiply the operand A
1 !                       digits to get the absolute value of operand A.
1 !                       E.g., AEXP = 1, ADIGITS = 123 gives 1230.
1 !       ADIGITS.rnu.d   Descriptor for the digits of operand A
1 !       BSIGN.rv.r      0 = operand B is positive, 1 = negative
1 !       BEXP.rl.r       Power of 10 by which to multiply the operand B
1 !                       digits to get the absolute value of operand B.
1 !                       E.g., BEXP = -1, BDIGITS = 123 gives 12.3.
1 !       BDIGITS.rnu.d   Descriptor for the digits of operand B
1 !       CSIGN.wl.r      0 = operand C is positive, 1 = negative
1 !       CEXP.wl.r       Power of 10 by which to multiply the operand C
1 !                       digits to get the absolute value of operand C.
1 !                       E.g., CEXP = 0, CDIGITS = 123 gives 123.
1 !       CDIGITS.wnu.d   Descriptor for the digits of operand C
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       May allocate space for the CDIGITS string.
1 !       Signals if storage is exceeded.
1 !--
1
2     BEGIN
2
2     MAP
```

```
;  263      0354  2           ADIGITS : REF BLOCK [8, BYTE],
;  264      0355  2           BDIGITS : REF BLOCK [8, BYTE],
;  265      0356  2           CDIGITS : REF BLOCK [8, BYTE];
;  266      0357  2
;  267      0358  2       LOCAL
;  268      0359  2   !+
;  269      0360  2   ! Internal form of A
;  270      0361  2   !-
;  271      0362  2           A_DESC : BLOCK [8, BYTE] VOLATILE,
;  272      0363  2           ABUF : REF VECTOR [65535, BYTE],
;  273      0364  2           A_LEN,
;  274      0365  2           A_SIGN,
;  275      0366  2   !+
;  276      0367  2   ! Internal form of B
;  277      0368  2   !-
;  278      0369  2           B_DESC : BLOCK [8, BYTE] VOLATILE,
;  279      0370  2           BBUF : REF VECTOR [65535, BYTE],
;  280      0371  2           B_LEN,
;  281      0372  2           B_SIGN,
;  282      0373  2   !+
;  283      0374  2   ! Local copy of result.
;  284      0375  2   !-
;  285      0376  2           RSIGN,
;  286      0377  2           REXP,
;  287      0378  2           R_DESC : BLOCK [8, BYTE] VOLATILE,
;  288      0379  2           RBUF : REF VECTOR [65535, BYTE],        ! Addresses result
;  289      0380  2           R_LEN,                                  ! Length of result
;  290      0381  2           RESULT_DIGITS,                          ! Number of digits in result
;  291      0382  2
;  292      0383  2   !+
;  293      0384  2   ! The following locals are needed for calls to LIB$ANALYZE_SDESC.
;  294      0385  2   !-
;  295      0386  2           CBUF,
;  296      0387  2           C_LEN,
;  297      0388  2           STATUS;
;  298      0389  2
;  299      0390  2
;  300      0391  2       BUILTIN
;  301      0392  2           ACTUALCOUNT;
;  302      0393  2
;  303      0394  2   !+
;  304      0395  2   ! Enable a handler to free the local strings in case of an error.
;  305      0396  2   !-
;  306      0397  2
;  307      0398  2       ENABLE
;  308      0399  2           FREE_STRINGS (A_DESC, B_DESC, R_DESC);
;  309      0400  2
;  310      0401  2   !+
;  311      0402  2   ! Check for the proper number of arguments.
;  312      0403  2   !-
;  313      0404  2
;  314      0405  3       IF (ACTUALCOUNT () LSS 9)
;  315      0406  2       THEN
;  316      0407  3           BEGIN
;  317      0408  3
;  318      0409  3           LOCAL
;  319      0410  3               ROUT_NAME_DESC : BLOCK [8, BYTE];
;   320      0411  3            ROUT_NAME_DESC [DSC$W_LENGTH] = 7;
;   321      0412  3            ROUT_NAME_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;   322      0413  3            ROUT_NAME_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
;   323      0414  3            ROUT_NAME_DESC [DSC$A_POINTER] = UPLIT (%ASCII'STR$ADD');
;   324      0415  3            LIB$STOP (STR$_WRONUMARG, 2, ACTUALCOUNT (), ROUT_NAME_DESC);
;   325      0416  3            END;
;   326      0417  2
;   327      0418  2
;   328      0419  2  !+
;   329      0420  2  ! Copy the A and B operands, taking the tens complement of the negative
;   330      0421  2  ! ones.
;   331      0422  2  !-
;   332      0423  2      A_DESC [DSC$W_LENGTH] = 0;
;   333      0424  2      A_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;   334      0425  2      A_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;   335      0426  2      A_DESC [DSC$A_POINTER] = 0;
;   336      0427  2  !+
;   337      0428  2  ! Compute the length of operand A.  Only the leading digits count.
;   338      0429  2  ! (Somday use SCAN or SPAN for this.)
;   339      0430  2  ! First call LIB$ANALYZE_SDESC to ensure that the input descriptor
;   340      0431  2  ! is valid.  If it is, then ABUF will contain the address of the
;   341      0432  2  ! first byte of the string, and A_LEN will contain its length.
;   342      0433  2  !-
;   343      0434  2
;   344      0435  2      STATUS = LIB$ANALYZE_SDESC (.ADIGITS,A_LEN,ABUF);
;   345      0436  2      IF .STATUS NEQ SS$_NORMAL
;   346      0437  2         THEN
;   347      0438  2            LIB$STOP (LIB$_INVARG);
;   348      0439  2
;   349      0440  2  !+
;   350      0441  2  ! Check here for the CDIGITS descriptor before getting too involved
;   351      0442  2  ! in the routine.
;   352      0443  2  !-
;   353      0444  2
;   354      0445  2      STATUS = LIB$ANALYZE_SDESC (.CDIGITS,C_LEN,CBUF);
;   355      0446  2      IF .STATUS NEQ SS$_NORMAL
;   356      0447  2         THEN
;   357      0448  2            LIB$STOP (LIB$_INVARG);
;   358      0449  2      A_LEN = 0;
;   359      0450  2      A_SIGN = ..ASIGN;
;   360      0451  3      BEGIN
;   361      0452  3
;   362      0453  3      LOCAL
;   363      0454  3            SCAN_DONE;
;   364      0455  3
;   365      0456  3      SCAN_DONE = 0;
;   366      0457  3
;   367      0458  3      DO
;   368      0459  4            BEGIN
;   369      0460  4
;   370      0461  5            IF (.A_LEN EQLU .ADIGITS [DSC$W_LENGTH])
;   371      0462  4            THEN
;   372      0463  4                  SCAN_DONE = 1
;   373      0464  4            ELSE
;   374      0465  4
;   375      0466  5                  IF ((.ABUF [.A_LEN] GEQ %C'0') AND (.ABUF [.A_LEN] LEQ %C'9'))
;   376      0467  4                  THEN
```

```
4                   A_LEN = .A_LEN + 1
4               ELSE
4                   SCAN_DONE = 1;
4
4           END
3       UNTIL (.SCAN_DONE);
3
2       END;
2       A_LEN = .A_LEN + 1;                             ! Extra digit for sign
2       STR$GET1_DX (A_LEN, A_DESC);
2       ABUF = .A_DESC [DSC$A_POINTER];
2       ABUF [0] = %C'0';
2       CH$MOVE (.A_LEN - 1, .ADIGITS [DSC$A_POINTER], ABUF [1]);
2
3       IF (.A_SIGN)
2       THEN
3           BEGIN
3   !+
3   ! Take the tens complement of the A operand.  This is done by
3   ! subtracting each digit from 9, and adding 1 to the result.  The final
3   ! add can cause carries.
3   !-
3
3           DECR COUNTER FROM .A_LEN - 1 TO 0 DO
3               ABUF [.COUNTER] = (9 - (.ABUF [.COUNTER] - %C'0')) + %C'0';
3
4           BEGIN
4
4           LOCAL
4               CARRY_DONE,
4               CARRY_COUNTER;
4
4           CARRY_DONE = 0;
4           CARRY_COUNTER = .A_LEN - 1;
4
5           IF (.CARRY_COUNTER GEQ 0)
4           THEN
4
4               DO
5                   BEGIN
5                   ABUF [.CARRY_COUNTER] = .ABUF [.CARRY_COUNTER] + 1;
5
6                   IF (.ABUF [.CARRY_COUNTER] LEQ %C'9')
5                   THEN
5                       CARRY_DONE = 1
5                   ELSE
6                       BEGIN
6                       ABUF [.CARRY_COUNTER] = .ABUF [.CARRY_COUNTER] - 10;
6                       CARRY_COUNTER = .CARRY_COUNTER - 1;
5                       END;
5
5                   END
4               UNTIL ((.CARRY_DONE) OR (.CARRY_COUNTER LSS 0));
4
4           IF ( NOT .CARRY_DONE) THEN A_SIGN = 0;
4
3           END;
```

```
;  434     0525  2         END;
;  435     0526  2
;  436     0527  2     B_DESC [DSC$W_LENGTH] = 0;
;  437     0528  2     B_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;  438     0529  2     B_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  439     0530  2     B_DESC [DSC$A_POINTER] = 0;
;  440     0531  2 !+
;  441     0532  2 ! Compute the length of operand B.  Only the leading digits count.
;  442     0533  2 ! First call LIB$ANALYZE_SDESC to ensure that the input descriptor
;  443     0534  2 ! is valid.  If it is, then BBUF will contain the address of the
;  444     0535  2 ! first byte of the string, and B_LEN will contain its length.
;  445     0536  2 !-
;  446     0537  2
;  447     0538  2     STATUS = LIB$ANALYZE_SDESC (.BDIGITS,B_LEN,BBUF);
;  448     0539  2     IF .STATUS NEQ SS$_NORMAL
;  449     0540  2       THEN
;  450     0541  2         LIB$STOP (LIB$_INVARG);
;  451     0542  2     B_LEN = 0;
;  452     0543  2     B_SIGN = ..BSIGN;
;  453     0544  3     BEGIN
;  454     0545  3
;  455     0546  3     LOCAL
;  456     0547  3         SCAN_DONE;
;  457     0548  3
;  458     0549  3     SCAN_DONE = 0;
;  459     0550  3
;  460     0551  3     DO
;  461     0552  4         BEGIN
;  462     0553  4
;  463     0554  5         IF (.B_LEN EQLU .BDIGITS [DSC$W_LENGTH])
;  464     0555  4         THEN
;  465     0556  4             SCAN_DONE = 1
;  466     0557  4         ELSE
;  467     0558  4
;  468     0559  5             IF ((.BBUF [.B_LEN] GEQ %C'0') AND (.BBUF [.B_LEN] LEQ %C'9'))
;  469     0560  4             THEN
;  470     0561  4                 B_LEN = .B_LEN + 1
;  471     0562  4             ELSE
;  472     0563  4                 SCAN_DONE = 1;
;  473     0564  4
;  474     0565  4         END
;  475     0566  3     UNTIL (.SCAN_DONE);
;  476     0567  3
;  477     0568  2     END;
;  478     0569  2     B_LEN = .B_LEN + 1;                                     ! Extra digit for sign
;  479     0570  2     STR$GET1_DX (B_LEN, B_DESC);
;  480     0571  2     BBUF = .B_DESC [DSC$A_POINTER];
;  481     0572  2     BBUF [0] = %C'0';
;  482     0573  2     CH$MOVE (.B_LEN - 1, .BDIGITS [DSC$A_POINTER], BBUF [1]);
;  483     0574  2
;  484     0575  3     IF (.B_SIGN)
;  485     0576  2     THEN
;  486     0577  3         BEGIN
;  487     0578  3 !+
;  488     0579  3 ! Take the tens complement of the B operand.  This is done by
;  489     0580  3 ! subtracting each digit from 9, and adding 1 to the result.  The final
;  490     0581  3 ! add can cause carries.
```

```
STR
1-0

; R

;
```

```
;  491       0582  3 !-
;  492       0583  3
;  493       0584  3         DECR COUNTER FROM .B_LEN - 1 TO 0 DO
;  494       0585  3             BBUF [.COUNTER] = (9 - (.BBUF [.COUNTER] - %C'0')) + %C'0';
;  495       0586  3
;  496       0587  4         BEGIN
;  497       0588  4
;  498       0589  4         LOCAL
;  499       0590  4             CARRY_DONE,
;  500       0591  4             CARRY_COUNTER;
;  501       0592  4
;  502       0593  4         CARRY_DONE = 0;
;  503       0594  4         CARRY_COUNTER = .B_LEN - 1;
;  504       0595  4
;  505       0596  5         IF (.CARRY_COUNTER GEQ 0)
;  506       0597  4         THEN
;  507       0598  4
;  508       0599  4             DO
;  509       0600  5                 BEGIN
;  510       0601  5                 BBUF [.CARRY_COUNTER] = .BBUF [.CARRY_COUNTER] + 1;
;  511       0602  5
;  512       0603  6                 IF (.BBUF [.CARRY_COUNTER] LEQ %C'9')
;  513       0604  5                 THEN
;  514       0605  5                     CARRY_DONE = 1
;  515       0606  5                 ELSE
;  516       0607  6                     BEGIN
;  517       0608  6                     BBUF [.CARRY_COUNTER] = .BBUF [.CARRY_COUNTER] - 10;
;  518       0609  6                     CARRY_COUNTER = .CARRY_COUNTER - 1;
;  519       0610  5                     END;
;  520       0611  5
;  521       0612  5                 END
;  522       0613  4             UNTIL ((.CARRY_DONE) OR (.CARRY_COUNTER LSS 0));
;  523       0614  4
;  524       0615  4         IF ( NOT .CARRY_DONE) THEN B_SIGN = 0;
;  525       0616  4
;  526       0617  3         END;
;  527       0618  2         END;
;  528       0619  2
;  529       0620  2 !+
;  530       0621  2 ! Compute a tenative result exponent based on the smallest exponent
;  531       0622  2 ! in either A or B.
;  532       0623  2 !-
;  533       0624  2     REXP = MIN (..AEXP, ..BEXP);
;  534       0625  2 !+
;  535       0626  2 ! Allocate enough space to hold the maximum possible number of result
;  536       0627  2 ! digits.  This is done by spanning the powers of ten involved in the
;  537       0628  2 ! two input operands, and adding 1 for carry.
;  538       0629  2 !-
;  539       0630  2     RESULT_DIGITS = (MAX (..AEXP + .A_LEN, ..BEXP + .B_LEN)) + 1 - .REXP;
;  540       0631  2     R_DESC [DSC$W_LENGTH] = 0;
;  541       0632  2     R_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
;  542       0633  2     R_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;  543       0634  2     R_DESC [DSC$A_POINTER] = 0;
;  544       0635  2     STR$GET1_DX (RESULT_DIGITS, R_DESC);
;  545       0636  2     RBUF = .R_DESC [DSC$A_POINTER];
;  546       0637  2     R_LEN = .R_DESC [DSC$W_LENGTH];
;  547       0638  2 !+
2 ! Copy the A operand into the result string, offsetting it properly
2 ! based on the exponents.
2 !-
2     CH$FILL (%C'0', .R_LEN, .R_DESC [DSC$A_POINTER]);
2     CH$MOVE (.A_LEN, .A_DESC [DSC$A_POINTER],      !
2         .R_DESC [DSC$A_POINTER] + .R_LEN - (..AEXP - .REXP) - .A_LEN);
2 !+
2 ! If the A operand was negative we owe high-order nines.
2 !-
2
2     IF (.A_SIGN) THEN CH$FILL (%C'9', (.R_LEN - .A_LEN) - (..AEXP - .REXP), .R_DESC [DSC$A_POINTER]);
2
2 !+
2 ! Now add in the B operand.
2 !-
2
2     DECR COUNTER FROM (.R_LEN - 1 - (..BEXP - .REXP)) TO (.R_LEN - 1 - (..BEXP - .REXP) - (.B_LEN - 1)) DO
3         BEGIN
3
3         LOCAL
3             B_INDEX,
3             SUM;
3
3         B_INDEX = .COUNTER - (.R_LEN - 1 - (..BEXP - .REXP) - (.B_LEN - 1));
3         SUM = .RBUF [.COUNTER] + .BBUF [.B_INDEX] - %C'0';
3
4         IF (.SUM GTR %C'9')
3         THEN
4             BEGIN
4 !+
4 ! We must propagate a carry to the higher digits of RBUF
4 !-
4
4             LOCAL
4                 CARRY_DONE,
4                 CARRY_COUNTER;
4
4             RBUF [.COUNTER] = .SUM - 10;
4             CARRY_DONE = 0;
4             CARRY_COUNTER = .COUNTER - 1;
4
5             IF (.CARRY_COUNTER GEQ 0)
4             THEN
4
4                 DO
5                     BEGIN
5                     RBUF [.CARRY_COUNTER] = .RBUF [.CARRY_COUNTER] + 1;
5
6                     IF (.RBUF [.CARRY_COUNTER] LEQ %C'9')
5                     THEN
5                         CARRY_DONE = 1
5                     ELSE
6                         BEGIN
6                         RBUF [.CARRY_COUNTER] = .RBUF [.CARRY_COUNTER] - 10;
6                         CARRY_COUNTER = .CARRY_COUNTER - 1;
5                         END;
5
```

```
5                       END
4                   UNTIL ((.CARRY_DONE) OR (.CARRY_COUNTER LSS 0));
4
4               END
3           ELSE
3               RBUF [.COUNTER] = .SUM;
3
2           END;
2
2   !+
2   ! End of the DECR loop.
2   !-
2   !+
2   ! If the B operand is negative, we owe high-order nines.
2   !-
2
3       IF (.B_SIGN)
2       THEN
3           BEGIN
3
3           DECR COUNTER FROM ((.R_LEN - 1 - (..BEXP - .REXP) - (.B_LEN - 1)) - 1) TO 0 DO
4               BEGIN
4
4               LOCAL
4                   SUM;
4
4               SUM = .RBUF [.COUNTER] + 9;
4
5               IF (.SUM GTR %C'9')
4               THEN
5                   BEGIN
5   !+
5   ! We must propagate a carry to the higher digits of RBUF
5   !-
5
5                   LOCAL
5                       CARRY_DONE,
5                       CARRY_COUNTER;
5
5                   RBUF [.COUNTER] = .SUM - 10;
5                   CARRY_DONE = 0;
5                   CARRY_COUNTER = .COUNTER - 1;
5
6                   IF (.CARRY_COUNTER GEQ 0)
5                   THEN
5
5                       DO
6                           BEGIN
6                           RBUF [.CARRY_COUNTER] = .RBUF [.CARRY_COUNTER] + 1;
6
7                           IF (.RBUF [.CARRY_COUNTER] LEQ %C'9')
6                           THEN
6                               CARRY_DONE = 1
6                           ELSE
7                               BEGIN
7                               RBUF [.CARRY_COUNTER] = .RBUF [.CARRY_COUNTER] - 10;
7                               CARRY_COUNTER = .CARRY_COUNTER - 1;
```

```
;   662          0753  6                              END;
;   663          0754  6
;   664          0755  6                          END
;   665          0756  5                      UNTIL ((.CARRY_DONE) OR (.CARRY_COUNTER LSS 0));
;   666          0757  5
;   667          0758  5                  END
;   668          0759  4              ELSE
;   669          0760  4                  RBUF [.COUNTER] = .SUM;
;   670          0761  4
;   671          0762  3              END;
;   672          0763  3
;   673          0764  2          END;
;   674          0765  2
;   675          0766  2  !+
;   676          0767  2  ! Compute the sign of the result and recomplement it if negative.
;   677          0768  2  !-
;   678          0769  2
;   679          0770  3      IF (.RBUF [0] GEQ %C'5')
;   680          0771  2      THEN
;   681          0772  3          BEGIN
;   682          0773  3          RSIGN = 1;
;   683          0774  3
;   684          0775  3          DECR COUNTER FROM .R_LEN - 1 TO 0 DO
;   685          0776  3              RBUF [.COUNTER] = (9 - (.RBUF [.COUNTER] - %C'0')) + %C'0';
;   686          0777  3
;   687          0778  4          BEGIN
;   688          0779  4
;   689          0780  4          LOCAL
;   690          0781  4              CARRY_DONE,
;   691          0782  4              CARRY_COUNTER;
;   692          0783  4
;   693          0784  4          CARRY_DONE = 0;
;   694          0785  4          CARRY_COUNTER = .R_LEN - 1;
;   695          0786  4
;   696          0787  5          IF (.CARRY_COUNTER GEQ 0)
;   697          0788  4          THEN
;   698          0789  4
;   699          0790  4              DO
;   700          0791  5                  BEGIN
;   701          0792  5                  RBUF [.CARRY_COUNTER] = .RBUF [.CARRY_COUNTER] + 1;
;   702          0793  5
;   703          0794  6                  IF (.RBUF [.CARRY_COUNTER] LEQ %C'9')
;   704          0795  5                  THEN
;   705          0796  5                      CARRY_DONE = 1
;   706          0797  5                  ELSE
;   707          0798  6                      BEGIN
;   708          0799  6                      RBUF [.CARRY_COUNTER] = .RBUF [.CARRY_COUNTER] - 10;
;   709          0800  6                      CARRY_COUNTER = .CARRY_COUNTER - 1;
;   710          0801  5                      END;
;   711          0802  5
;   712          0803  5                  END
;   713          0804  4              UNTIL ((.CARRY_DONE) OR (.CARRY_COUNTER LSS 0));
;   714          0805  4
;   715          0806  3          END;
;   716          0807  3          END
;   717          0808  2      ELSE
;   718          0809  2          RSIGN = 0;
;  2  !+
;  2  ! Discard low-order zeros, adjusting the exponent.
;  2  !-
;  3      BEGIN
;  3
;  3      LOCAL
;  3          SCAN_DONE,
;  3          SCAN_COUNTER;
;  3
;  3      SCAN_DONE = 0;
;  3      SCAN_COUNTER = .RESULT_DIGITS - 1;
;  3
;  3      DO
;  4          BEGIN
;  4
;  5          IF (.SCAN_COUNTER LSS 0)
;  4          THEN
;  4              SCAN_DONE = 1
;  4          ELSE
;  4
;  4              IF (.RBUF [.SCAN_COUNTER] EQL %C'0') THEN SCAN_COUNTER = .SCAN_COUNTER - 1 ELSE SCAN_DONE = 1;
;  4
;  4          END
;  3      UNTIL (.SCAN_DONE);
;  3
;  3      REXP = .REXP + ((.RESULT_DIGITS - 1) - .SCAN_COUNTER);
;  3      RESULT_DIGITS = .SCAN_COUNTER + 1;
;  2      END;
;  2  !+
;  2  ! Remove high-order zeros.
;  2  !-
;  3      BEGIN
;  3
;  3      LOCAL
;  3          SCAN_DONE,
;  3          SCAN_COUNTER;
;  3
;  3      SCAN_COUNTER = 0;
;  3      SCAN_DONE = 0;
;  3
;  3      DO
;  4          BEGIN
;  4
;  5          IF (.SCAN_COUNTER GEQ .RESULT_DIGITS)
;  4          THEN
;  4              SCAN_DONE = 1
;  4          ELSE
;  4
;  4              IF (.RBUF [.SCAN_COUNTER] EQL %C'0') THEN SCAN_COUNTER = .SCAN_COUNTER + 1 ELSE SCAN_DONE = 1;
;  4
;  4          END
;  3      UNTIL (.SCAN_DONE);
;  3
;  4      IF (.SCAN_COUNTER GTR 0)
;  3      THEN
;  3
```

```
3           INCR COUNTER FROM 0 TO .RESULT_DIGITS - .SCAN_COUNTER - 1 DO
3               RBUF [.COUNTER] = .RBUF [.COUNTER + .SCAN_COUNTER];
3
3       RESULT_DIGITS = .RESULT_DIGITS - .SCAN_COUNTER;
2       END;
2   !+
2   ! Return the results to the caller in the C operand.
2   ! If there are no digits left, return a single zero digit.
2   !-
2
3       IF (.RESULT_DIGITS EQL 0)
2       THEN
3           BEGIN
3           .CSIGN = 0;
3           .CEXP = 0;
3           STR$COPY_R (.CDIGITS, %REF (1), %REF (%ASCII'0'));
3           CHK_STR_TYPE (.CDIGITS[DSC$A_POINTER],%REF (1),.CDIGITS);
3           END
3
3   !+
3   ! Call CHK_STR_TYPE to determine if we need to pad the number with
3   ! leading zeroes depending on the string type.
3   !-
3
2       ELSE
3           BEGIN
3           .CSIGN = .RSIGN;
3           .CEXP = .REXP;
3           CHK_STR_TYPE (.R_DESC[DSC$A_POINTER],RESULT_DIGITS,.CDIGITS);
2           END;
2
2   !     ELSE
2   !         BEGIN
2   !         .CSIGN = .RSIGN;
2   !         .CEXP = .REXP;
2   !         STR$COPY_R (.CDIGITS, RESULT_DIGITS, .R_DESC [DSC$A_POINTER]);
2   !         END;
2
2
2   !+
2   ! Free our strings.
2   !-
2       STR$FREE1_DX (R_DESC);
2       STR$FREE1_DX (A_DESC);
2       STR$FREE1_DX (B_DESC);
1       END;                                            ! end of STR$ADD

                                                          .TITLE  STR$ARITH
                                                          .IDENT  \1-019\

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

                                   00# 00000 P.AAA:      .BYTE   0[7]                                ;
                                   0C  00007              .BYTE   12                                  ;
                                   00# 00008 P.AAB:      .BYTE   0[6]                                ;
                               0C  01  0000E              .BYTE   1, 12                               ;
```

```
                                    00# 00010 P.AAC:  .BYTE    0[48]                                              ;
                                    01# 00040         .BYTE    1[10]                                              ;
                                    00# 0004A         .BYTE    0[198]                                             ;
                                    01  00110 P.AAD:  .BYTE    1                                                  ;
                                        00111         .BLKB    3
  00  44  44  41  24  52  54  53  00114 P.AAE:  .ASCII   \STR$ADD\<0>                                       ;

                                              ZERO=                  P.AAA
                                              TEN=                   P.AAB
                                              SPANC_TABLE=           P.AAC
                                              MASK=                  P.AAD
                                                      .EXTRN   LIB$STOP, STR$GET1_DX
                                                      .EXTRN   STR$FREE1_DX, STR$COPY_R
                                                      .EXTRN   STR$COPY_DX, LIB$GET_VM
                                                      .EXTRN   LIB$FREE_VM, LIB$SCOPY_R_DX
                                                      .EXTRN   LIB$$ROUND_R7, LIB$$CALC_D_R7
                                                      .EXTRN   LIB$$CALC_Q_R9, LIB$$SUB_PACK_R8
                                                      .EXTRN   LIB$$MUL_PACK_R10
                                                      .EXTRN   LIB$$ADJUST_Q_R9
                                                      .EXTRN   LIB$$CVT_STR_PACK_R9
                                                      .EXTRN   LIB$$CVT_PACK_STR_R8
                                                      .EXTRN   LIB$ANALYZE_SDESC
                                                      .EXTRN   LIB$MATCH_COND, STR$DUPL_CHAR
                                                      .EXTRN   LIB$_INVARG, STR$_DIVBY_ZER
                                                      .EXTRN   STR$_WRONUMARG

                                  0FFC 00000          .ENTRY   STR$ADD, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-  ; 0297
                                                               R11                                          ;
          5E           BC  AE  9E 00002               MOVAB    -68(SP), SP                                  ;
                       2C  AE  7C 00006               CLRQ     R_DESC                                       ; 0351
                       34  AE  7C 00009               CLRQ     B_DESC                                       ;
                       3C  AE  7C 0000C               CLRQ     A_DESC                                       ;
          6D         0401  CF  DE 0000F               MOVAL    59$, (FP)                                    ;
          09               6C  91 00014               CMPB     (AP), #9                                     ; 0405
                           22  1E 00017               BGEQU    1$                                           ;
     24   AE  010E0007     8F  D0 00019               MOVL     #17694727, ROUT_NAME_DESC                    ; 0412
     28   AE           D4  AF  9E 00021               MOVAB    P.AAE, ROUT_NAME_DESC+4                      ; 0415
                       24  AE  9F 00026               PUSHAB   ROUT_NAME_DESC                               ; 0416
          7E               6C  9A 00029               MOVZBL   (AP), -(SP)                                  ;
                           02  DD 0002C               PUSHL    #2                                           ;
              00000000G    8F  DD 0002E               PUSHL    #STR$_WRONUMARG                              ;
00000000G  00              04  FB 00034               CALLS    #4, LIB$STOP                                 ;
                       3C  AE  B4 0003B 1$:           CLRW     A_DESC                                       ; 0423
     3E   AE               0F  90 0003E               MOVB     #15, A_DESC+2                                ; 0424
     3F   AE               02  90 00042               MOVB     #2, A_DESC+3                                 ; 0425
                       40  AE  D4 00046               CLRL     A_DESC+4                                     ; 0426
                       08  AE  9F 00049               PUSHAB   ABUF                                         ; 0435
                       18  AE  9F 0004C               PUSHAB   A_LEN                                        ;
          52           0C  AC  D0 0004F               MOVL     ADIGITS, R2                                  ;
                           52  DD 00053               PUSHL    R2                                           ;
00000000G  00              03  FB 00055               CALLS    #3, LIB$ANALYZE_SDESC                        ;
           58              50  D0 0005C               MOVL     R0, STATUS                                   ;
           01              58  D1 0005F               CMPL     STATUS, #1                                   ; 0436
                           0D  13 00062               BEQL     2$                                           ;
              00000000G    8F  DD 00064               PUSHL    #LIB$_INVARG                                 ; 0438
00000000G  00              01  FB 0006A               CALLS    #1, LIB$STOP                                 ;
                       0C  AE  9F 00071 2$:           PUSHAB   CBUF                                         ; 0445
```

```
                                      14  AE  9F 00074          PUSHAB  C_LEN                                        :
                                      24  AC  DD 00077          PUSHL   CDIGITS                                      :
                         00000000G  00    03  FB 0007A          CALLS   #3, LIB$ANALYZE_SDESC                        :
                                    58    50  D0 00081          MOVL    R0, STATUS                                   :
                                    01    58  D1 00084          CMPL    STATUS, #1                                   : 0446
                                          0D  13 00087          BEQL    3$                                           :
                              00000000G   8F  DD 00089          PUSHL   #LIB$_INVARG                                 : 0448
                         00000000G  00    01  FB 0008F          CALLS   #1, LIB$STOP                                 :
                                      14  AE  D4 00096 3$:      CLRL    A_LEN                                        : 0449
                              04    AE    04  BC  D0 00099          MOVL    @ASIGN, A_SIGN                               : 0450
                                          51  D4 0009E          CLRL    SCAN_DONE                                    : 0456
  14    AE           62             10    00  ED 000A0 4$:      CMPZV   #0, #16, (R2), A_LEN                         : 0461
                                          15  13 000A6          BEQL    5$                                           :
                     50       08    AE    14  AE  C1 000A8      ADDL3   A_LEN, ABUF, R0                              : 0466
                                    30    60  91 000AE          CMPB    (R0), #48                                    :
                                          0A  1F 000B1          BLSSU   5$                                           :
                                    39    60  91 000B3          CMPB    (R0), #57                                    :
                                          05  1A 000B6          BGTRU   5$                                           :
                                      14  AE  D6 000B8          INCL    A_LEN                                        : 0468
                                          03  11 000BB          BRB     6$                                           :
                                    51    01  D0 000BD 5$:      MOVL    #1, SCAN_DONE                                : 0470
                                    DD    51  E9 000C0 6$:      BLBC    SCAN_DONE, 4$                                : 0473
                                      14  AE  D6 000C3          INCL    A_LEN                                        : 0476
                                      3C  AE  9F 000C6          PUSHAB  A_DESC                                       : 0477
                                      18  AE  9F 000C9          PUSHAB  A_LEN                                        :
                         00000000G  00    02  FB 000CC          CALLS   #2, STR$GET1_DX                              :
                                08  AE    40  AE  D0 000D3      MOVL    A_DESC+4, ABUF                               : 0478
                                    56    08  AE  D0 000D8      MOVL    ABUF, R6                                     : 0479
                                    66    30  90 000DC          MOVB    #48, (R6)                                    :
                     57       14    AE    01  C3 000DF          SUBL3   #1, A_LEN, R7                                : 0480
              01     A6       04    B2    57  28 000E4          MOVC3   R7, @4(R2), 1(R6)                            :
                                    38    04  AE  E9 000EA      BLBC    A_SIGN, 13$                                  : 0482
                                    50    01  A7  9E 000EE      MOVAB   1(R7), COUNTER                               : 0491
                                          07  11 000F2          BRB     8$                                           :
                   6046       69    8F  6046  83 000F4 7$:      SUBB3   (COUNTER)[R6], #105, (COUNTER)[R6]           : 0492
                                    F6    50  F4 000FB 8$:      SOBGEQ  COUNTER, 7$                                  :
                                          51  D4 000FE          CLRL    CARRY_DONE                                   : 0500
                                    50    57  D0 00100          MOVL    R7, CARRY_COUNTER                            : 0501
                                          1B  19 00103          BLSS    12$                                          : 0503
                                        6046  96 00105 9$:      INCB    (CARRY_COUNTER)[R6]                          : 0508
                                    39  6046  91 00108          CMPB    (CARRY_COUNTER)[R6], #57                     : 0510
                                          05  1A 0010C          BGTRU   10$                                          :
                                    51    01  D0 0010E          MOVL    #1, CARRY_DONE                               : 0512
                                          06  11 00111          BRB     11$                                          :
                                  6046    0A  82 00113 10$:     SUBB2   #10, (CARRY_COUNTER)[R6]                     : 0515
                                          50  D7 00117          DECL    CARRY_COUNTER                                : 0516
                                    0A    51  E8 00119 11$:     BLBS    CARRY_DONE, 13$                              : 0520
                                          50  D5 0011C          TSTL    CARRY_COUNTER                                :
                                          E5  18 0011E          BGEQ    9$                                           :
                                    03    51  E8 00120 12$:     BLBS    CARRY_DONE, 13$                              : 0522
                                      04  AE  D4 00123          CLRL    A_SIGN                                       :
                                      34  AE  B4 00126 13$:     CLRW    B_DESC                                       : 0527
                              36    AE    0F  90 00129          MOVB    #15, B_DESC+2                                : 0528
                              37    AE    02  90 0012D          MOVB    #2, B_DESC+3                                 : 0529
                                      38  AE  D4 00131          CLRL    B_DESC+4                                     : 0530
                                      18  AE  9F 00134          PUSHAB  BBUF                                         : 0538
                                      20  AE  9F 00137          PUSHAB  B_LEN                                        :
```

STR
1-0

; R

```
                                      52        18   AC  D0 0013A       MOVL    BDIGITS, R2                         ;
                                                     52  DD 0013E       PUSHL   R2                                  ;
                          00000000G   00             03  FB 00140       CALLS   #3, LIB$ANALYZE_SDESC               ;
                                      58             50  D0 00147       MOVL    R0, STATUS                          ;
                                      01             58  D1 0014A       CMPL    STATUS, #1                          ; 0539
                                                     0D  13 0014D       BEQL    14$                                 ;
                                         00000000G   8F  DD 0014F       PUSHL   #LIB$_INVARG                        ; 0541
                          00000000G   00             01  FB 00155       CALLS   #1, LIB$STOP                        ;
                                                1C   AE  D4 0015C 14$:  CLRL    B_LEN                               ; 0542
                                      6E        10   BC  D0 0015F       MOVL    @BSIGN, B_SIGN                      ; 0543
                                                     51  D4 00163       CLRL    SCAN_DONE                           ; 0549
   1C   AE             62             10             00  ED 00165 15$:  CMPZV   #0, #16, (R2), B_LEN                ; 0554
                                                     15  13 0016B       BEQL    16$                                 ;
                       50        18   AE        1C   AE  C1 0016D       ADDL3   B_LEN, BBUF, R0                     ; 0559
                                      30             60  91 00173       CMPB    (R0), #48                           ;
                                                     0A  1F 00176       BLSSU   16$                                 ;
                                      39             60  91 00178       CMPB    (R0), #57                           ;
                                                     05  1A 0017B       BGTRU   16$                                 ;
                                                1C   AE  D6 0017D       INCL    B_LEN                               ; 0561
                                                     03  11 00180       BRB     17$                                 ;
                                      51             01  D0 00182 16$:  MOVL    #1, SCAN_DONE                       ; 0563
                                      DD             51  E9 00185 17$:  BLBC    SCAN_DONE, 15$                      ; 0566
                                                1C   AE  D6 00188       INCL    B_LEN                               ; 0569
                                                34   AE  9F 0018B       PUSHAB  B_DESC                              ; 0570
                                                20   AE  9F 0018E       PUSHAB  B_LEN                               ;
                          00000000G   00             02  FB 00191       CALLS   #2, STR$GET1_DX                     ;
                                 18   AE        38   AE  D0 00198       MOVL    B_DESC+4, BBUF                      ; 0571
                                      59        18   AE  D0 0019D       MOVL    BBUF, R9                            ; 0572
                                      69             30  90 001A1       MOVB    #48, (R9)                           ;
                       5B        1C   AE             01  C3 001A4       SUBL3   #1, B_LEN, R11                      ; 0573
                  01   A9        04   B2             5B  28 001A9       MOVC3   R11, @4(R2), 1(R9)                  ;
                                      37             6E  E9 001AF       BLBC    B_SIGN, 24$                         ; 0575
                                      50        01   AB  9E 001B2       MOVAB   1(R11), COUNTER                     ; 0584
                                                     07  11 001B6       BRB     19$                                 ;
                     6049        69   8F           6049  83 001B8 18$:  SUBB3   (COUNTER)[R9], #105, (COUNTER)[R9]  ; 0585
                                      F6             50  F4 001BF 19$:  SOBGEQ  COUNTER, 18$                        ;
                                                     51  D4 001C2       CLRL    CARRY_DONE                          ; 0593
                                      50             5B  D0 001C4       MOVL    R11, CARRY_COUNTER                  ; 0594
                                                     1B  19 001C7       BLSS    23$                                 ; 0596
                                                   6049  96 001C9 20$:  INCB    (CARRY_COUNTER)[R9]                 ; 0601
                                      39           6049  91 001CC       CMPB    (CARRY_COUNTER)[R9], #57            ; 0603
                                                     05  1A 001D0       BGTRU   21$                                 ;
                                      51             01  D0 001D2       MOVL    #1, CARRY_DONE                      ; 0605
                                                     06  11 001D5       BRB     22$                                 ;
                                    6049             0A  82 001D7 21$:  SUBB2   #10, (CARRY_COUNTER)[R9]            ; 0608
                                                     50  D7 001DB       DECL    CARRY_COUNTER                       ; 0609
                                      09             51  E8 001DD 22$:  BLBS    CARRY_DONE, 24$                     ; 0613
                                                     50  D5 001E0       TSTL    CARRY_COUNTER                       ;
                                                     E5  18 001E2       BGEQ    20$                                 ;
                                      02             51  E8 001E4 23$:  BLBS    CARRY_DONE, 24$                     ; 0615
                                                     6E  D4 001E7       CLRL    B_SIGN                              ;
                                      50        08   BC  D0 001E9 24$:  MOVL    @AEXP, R0                           ; 0624
                                 14   BC             50  D1 001ED       CMPL    R0, @BEXP                           ;
                                                     04  15 001F1       BLEQ    25$                                 ;
                                      50        14   BC  D0 001F3       MOVL    @BEXP, R0                           ;
                                      57             50  D0 001F7 25$:  MOVL    R0, REXP                            ;
                       50        08   BC        14   AE  C1 001FA       ADDL3   A_LEN, @AEXP, R0                    ; 0630
                  51        14    BC      1C    AE  C1 00200       ADDL3   B_LEN, @BEXP, R1                              ;
                                  51            50  D1 00206       CMPL    R0, R1                                        ;
                                                03  18 00209       BGEQ    26$                                           ;
                                  50            51  D0 0020B       MOVL    R1, R0                                        ;
                                  50            57  C2 0020E 26$:  SUBL2   REXP, R0                                      ;
                            20    AE      01    A0  9E 00211       MOVAB   1(R0), RESULT_DIGITS                          ;
                                          2C    AE  B4 00216       CLRW    R_DESC                                        ; 0631
                            2E    AE            0F  90 00219       MOVB    #15, R_DESC+2                                 ; 0632
                            2F    AE            02  90 0021D       MOVB    #2, R_DESC+3                                  ; 0633
                                          30    AE  D4 00221       CLRL    R_DESC+4                                      ; 0634
                                          2C    AE  9F 00224       PUSHAB  R_DESC                                        ; 0635
                                          24    AE  9F 00227       PUSHAB  RESULT_DIGITS                                 ;
                     00000000G    00            02  FB 0022A       CALLS   #2, STR$GET1_DX                               ;
                                  58      30    AE  D0 00231       MOVL    R_DESC+4, RBUF                                ; 0636
                                  56      2C    AE  3C 00235       MOVZWL  R_DESC, R_LEN                                 ; 0637
      56          30              6E            00  2C 00239       MOVC5   #0, (SP), #48, R_LEN, @R_DESC+4               ; 0642
                                          30    BE     0023E                                                             ;
                  50              56      30    AE  C1 00240       ADDL3   R_DESC+4, R_LEN, R0                           ; 0644
                  5A              57      08    BC  C3 00245       SUBL3   @AEXP, REXP, R10                              ;
                                  50            5A  C0 0024A       ADDL2   R10, R0                                       ;
                                  50      14    AE  C2 0024D       SUBL2   A_LEN, R0                                     ;
                  60        40    BE      14    AE  28 00251       MOVC3   A_LEN, @A_DESC+4, (R0)                        ;
                                  0F      04    AE  E9 00257       BLBC    A_SIGN, 27$                                   ; 0649
                  50              56      14    AE  C3 0025B       SUBL3   A_LEN, R_LEN, R0                              ;
                                  50            5A  C0 00260       ADDL2   R10, R0                                       ;
      50          39              6E            00  2C 00263       MOVC5   #0, (SP), #57, R0, @R_DESC+4                  ;
                                          30    BE     00268                                                             ;
                  50              57      14    BC  C3 0026A 27$:  SUBL3   @BEXP, REXP, R0                               ; 0655
                  51              56            50  C1 0026F       ADDL3   R0, R_LEN, R1                                 ;
                  50              51            5B  C3 00273       SUBL3   R11, R1, R0                                   ;
                                  52      FF    A0  9E 00277       MOVAB   -1(R0), R2                                    ;
                                                45  11 0027B       BRB     33$                                           ; 0663
                  50              51            52  C3 0027D 28$:  SUBL3   R2, COUNTER, B_INDEX                          ; 0662
                                  53          6148  9A 00281       MOVZBL  (COUNTER)[RBUF], R3                           ; 0663
                                  50          6049  9A 00285       MOVZBL  (B_INDEX)[R9], R0                             ;
                                  50            53  C0 00289       ADDL2   R3, R0                                        ;
                                  50            30  C2 0028C       SUBL2   #48, SUM                                      ;
                                  39            50  D1 0028F       CMPL    SUM, #57                                      ; 0665
                                                2A  15 00292       BLEQ    32$                                           ;
                6148              50            0A  83 00294       SUBB3   #10, SUM, (COUNTER)[RBUF]                     ; 0676
                                                53  D4 00299       CLRL    CARRY_DONE                                    ; 0677
                                  50      FF    A1  9E 0029B       MOVAB   -1(R1), CARRY_COUNTER                         ; 0678
                                                21  19 0029F       BLSS    33$                                           ; 0680
                                              6048  96 002A1 29$:  INCB    (CARRY_COUNTER)[RBUF]                         ; 0685
                                  39          6048  91 002A4       CMPB    (CARRY_COUNTER)[RBUF], #57                    ; 0687
                                                05  1A 002A8       BGTRU   30$                                           ;
                                  53            01  D0 002AA       MOVL    #1, CARRY_DONE                                ; 0689
                                                06  11 002AD       BRB     31$                                           ;
                                6048            0A  82 002AF 30$:  SUBB2   #10, (CARRY_COUNTER)[RBUF]                    ; 0692
                                                50  D7 002B3       DECL    CARRY_COUNTER                                 ; 0693
                                  0A            53  E8 002B5 31$:  BLBS    CARRY_DONE, 33$                               ; 0697
                                                50  D5 002B8       TSTL    CARRY_COUNTER                                 ;
                                                E5  18 002BA       BGEQ    29$                                           ;
                                                04  11 002BC       BRB     33$                                           ; 0665
                                6148            50  90 002BE 32$:  MOVB    SUM, (COUNTER)[RBUF]                          ; 0701
                                                51  D7 002C2 33$:  DECL    COUNTER                                       ; 0655
                                  52            51  D1 002C4       CMPL    COUNTER, R2                                   ;
```

```
                                      B4  18 002C7        BGEQ    28$                                     ;
                      42              6E  E9 002C9        BLBC    B_SIGN, 40$                             ; 0712
                      51              52  D0 002CC        MOVL    R2, COUNTER                             ; 0716
                                      3A  11 002CF        BRB     39$                                     ;
                      50            6148  9A 002D1 34$:   MOVZBL  (COUNTER)[RBUF], SUM                    ; 0722
                      50              09  C0 002D5        ADDL2   #9, SUM                                 ;
                      39              50  D1 002D8        CMPL    SUM, #57                                ; 0724
                                      2A  15 002DB        BLEQ    38$                                     ;
    6148              50              0A  83 002DD        SUBB3   #10, SUM, (COUNTER)[RBUF]               ; 0735
                                      52  D4 002E2        CLRL    CARRY_DONE                              ; 0736
                      50      FF      A1  9E 002E4        MOVAB   -1(R1), CARRY_COUNTER                   ; 0737
                                      21  19 002E8        BLSS    39$                                     ; 0739
                                    6048  96 002EA 35$:   INCB    (CARRY_COUNTER)[RBUF]                   ; 0744
                      39            6048  91 002ED        CMPB    (CARRY_COUNTER)[RBUF], #57              ; 0746
                                      05  1A 002F1        BGTRU   36$                                     ;
                      52              01  D0 002F3        MOVL    #1, CARRY_DONE                          ; 0748
                                      06  11 002F6        BRB     37$                                     ;
                    6048              0A  82 002F8 36$:   SUBB2   #10, (CARRY_COUNTER)[RBUF]              ; 0751
                                      50  D7 002FC        DECL    CARRY_COUNTER                           ; 0752
                      0A              52  E8 002FE 37$:   BLBS    CARRY_DONE, 39$                         ; 0756
                                      50  D5 00301        TSTL    CARRY_COUNTER                           ;
                                      E5  18 00303        BGEQ    35$                                     ;
                                      04  11 00305        BRB     39$                                     ; 0724
                    6148              50  90 00307 38$:   MOVB    SUM, (COUNTER)[RBUF]                    ; 0760
                      C3              51  F4 0030B 39$:   SOBGEQ  COUNTER, 34$                            ; 0716
                      35              68  91 0030E 40$:   CMPB    (RBUF), #53                             ; 0770
                                      37  1F 00311        BLSSU   46$                                     ;
                      54              01  D0 00313        MOVL    #1, RSIGN                               ; 0773
                      50              56  D0 00316        MOVL    R_LEN, COUNTER                          ; 0775
                                      07  11 00319        BRB     42$                                     ;
    6048      69      8F            6048  83 0031B 41$:   SUBB3   (COUNTER)[RBUF], #105, (COUNTER)[RBUF]  ; 0776
                      F6              50  F4 00322 42$:   SOBGEQ  COUNTER, 41$                            ;
                                      51  D4 00325        CLRL    CARRY_DONE                              ; 0784
                      50      FF      A6  9E 00327        MOVAB   -1(R6), CARRY_COUNTER                   ; 0785
                                      1F  19 0032B        BLSS    47$                                     ; 0787
                                    6048  96 0032D 43$:   INCB    (CARRY_COUNTER)[RBUF]                   ; 0792
                      39            6048  91 00330        CMPB    (CARRY_COUNTER)[RBUF], #57              ; 0794
                                      05  1A 00334        BGTRU   44$                                     ;
                      51              01  D0 00336        MOVL    #1, CARRY_DONE                          ; 0796
                                      06  11 00339        BRB     45$                                     ;
                    6048              0A  82 0033B 44$:   SUBB2   #10, (CARRY_COUNTER)[RBUF]              ; 0799
                                      50  D7 0033F        DECL    CARRY_COUNTER                           ; 0800
                      08              51  E8 00341 45$:   BLBS    CARRY_DONE, 47$                         ; 0804
                                      50  D5 00344        TSTL    CARRY_COUNTER                           ;
                                      E5  18 00346        BGEQ    43$                                     ;
                                      02  11 00348        BRB     47$                                     ; 0770
                                      54  D4 0034A 46$:   CLRL    RSIGN                                   ; 0809
                                      52  D4 0034C 47$:   CLRL    SCAN_DONE                               ; 0820
      51      20      AE              01  C3 0034E        SUBL3   #1, RESULT_DIGITS, R1                   ; 0821
                      50              51  D0 00353        MOVL    R1, SCAN_COUNTER                        ;
                                      50  D5 00356 48$:   TSTL    SCAN_COUNTER                            ; 0826
                                      0A  19 00358        BLSS    49$                                     ;
                      30            6048  91 0035A        CMPB    (SCAN_COUNTER)[RBUF], #48               ; 0831
                                      04  12 0035E        BNEQ    49$                                     ;
                                      50  D7 00360        DECL    SCAN_COUNTER                            ;
                                      03  11 00362        BRB     50$                                     ;
                      52              01  D0 00364 49$:   MOVL    #1, SCAN_DONE                           ;
```

```
                     EC            52 E9 00367 50$:   BLBC    SCAN_DONE, 48$                            ; 0834
                     51            50 C2 0036A        SUBL2   SCAN_COUNTER, R1                          ; 0836
                     57            51 C0 0036D        ADDL2   R1, REXP                                  ;
               20    AE      01    A0 9E 00370        MOVAB   1(R0), RESULT_DIGITS                      ; 0837
                     50            50 7C 00375        CLRQ    SCAN_DONE                                 ; 0849
               20    AE            51 D1 00377 51$:   CMPL    SCAN_COUNTER, RESULT_DIGITS               ; 0854
                                   0A 18 0037B        BGEQ    52$                                       ;
                     30          6148 91 0037D        CMPB    (SCAN_COUNTER)[RBUF], #48                 ; 0859
                                   04 12 00381        BNEQ    52$                                       ;
                                   51 D6 00383        INCL    SCAN_COUNTER                              ;
                                   03 11 00385        BRB     53$                                       ;
                     50            01 D0 00387 52$:   MOVL    #1, SCAN_DONE                             ;
                     EA            50 E9 0038A 53$:   BLBC    SCAN_DONE, 51$                            ; 0862
                                   51 D5 0038D        TSTL    SCAN_COUNTER                              ; 0864
                                   17 15 0038F        BLEQ    56$                                       ;
  53           20    AE            51 C3 00391        SUBL3   SCAN_COUNTER, RESULT_DIGITS, R3           ; 0867
                     50            01 CE 00396        MNEGL   #1, COUNTER                               ;
                                   09 11 00399        BRB     55$                                       ;
  52                 50            51 C1 0039B 54$:   ADDL3   SCAN_COUNTER, COUNTER, R2                 ; 0868
                   6048          6248 90 0039F        MOVB    (R2)[RBUF], (COUNTER)[RBUF]               ;
  F3                 50            53 F2 003A4 55$:   AOBLSS  R3, COUNTER, 54$                          ;
               20    AE            51 C2 003A8 56$:   SUBL2   SCAN_COUNTER, RESULT_DIGITS               ; 0870
                                   31 12 003AC        BNEQ    57$                                       ; 0877
                             1C    BC D4 003AE        CLRL    @CSIGN                                    ; 0880
                             20    BC D4 003B1        CLRL    @CEXP                                     ; 0881
               04    AE            30 D0 003B4        MOVL    #48, 4(SP)                                ; 0882
                             04    AE 9F 003B8        PUSHAB  4(SP)                                     ;
               04    AE            01 D0 003BB        MOVL    #1, 4(SP)                                 ;
                             04    AE 9F 003BF        PUSHAB  4(SP)                                     ;
                             24    AC DD 003C2        PUSHL   CDIGITS                                   ;
        00000000G    00            03 FB 003C5        CALLS   #3, STR$COPY_R                            ;
                             24    AC DD 003CC        PUSHL   CDIGITS                                   ; 0883
               08    AE            01 D0 003CF        MOVL    #1, 8(SP)                                 ;
                             08    AE 9F 003D3        PUSHAB  8(SP)                                     ;
  52           24    AC            04 C1 003D6        ADDL3   #4, CDIGITS, R2                           ;
                                   62 DD 003DB        PUSHL   (R2)                                      ;
                                   11 11 003DD        BRB     58$                                       ;
               1C    BC            54 D0 003DF 57$:   MOVL    RSIGN, @CSIGN                             ; 0893
               20    BC            57 D0 003E3        MOVL    REXP, @CEXP                               ; 0894
                             24    AC DD 003E7        PUSHL   CDIGITS                                   ; 0895
                             24    AE 9F 003EA        PUSHAB  RESULT_DIGITS                             ;
                             38    AE DD 003ED        PUSHL   R_DESC+4                                  ;
            0000V    CF            03 FB 003F0 58$:   CALLS   #3, CHK_STR_TYPE                          ;
                             2C    AE 9F 003F5        PUSHAB  R_DESC                                    ; 0909
        00000000G    00            01 FB 003F8        CALLS   #1, STR$FREE1_DX                          ;
                             3C    AE 9F 003FF        PUSHAB  A_DESC                                    ; 0910
        00000000G    00            01 FB 00402        CALLS   #1, STR$FREE1_DX                          ;
                             34    AE 9F 00409        PUSHAB  B_DESC                                    ; 0911
        00000000G    00            01 FB 0040C        CALLS   #1, STR$FREE1_DX                          ;
                                      04 00413        RET                                               ; 0912
                                    0000 00414 59$:   .WORD   Save nothing                              ; 0351
                     50      08    AC D0 00416        MOVL    8(AP), R0                                 ;
                     50      04    A0 D0 0041A        MOVL    4(R0), R0                                 ;
                             E8    A0 9F 0041E        PUSHAB  R_DESC                                    ;
                             F0    A0 9F 00421        PUSHAB  B_DESC                                    ;
                             F8    A0 9F 00424        PUSHAB  A_DESC                                    ;
                                   03 DD 00427        PUSHL   #3                                        ;
                                   5E  DD 00429          PUSHL   SP                                    ;
                    7E         04  AC  7D 0042B          MOVQ    4(AP), -(SP)                          ;
              0000V CF             03  FB 0042F          CALLS   #3, FREE_STRINGS                      ;
                                       04 00434          RET                                           ;

; Routine Size:  1077 bytes,    Routine Base:  _STR$CODE + 011C


;  822          0913  1
```

```
1 GLOBAL ROUTINE STR$MUL (                              ! Multiply two strings
1           ASIGN,                                      ! Sign of operand A
1           AEXP,                                       ! Decimal exponent of operand A
1           ADIGITS,                                    ! Digits of operand A
1           BSIGN,                                      ! Sign of operand B
1           BEXP,                                       ! Decimal exponent of operand B
1           BDIGITS,                                    ! Digits of operand B
1           CSIGN,                                      ! Sign of operand C
1           CEXP,                                       ! Decimal exponent of operand C
1           CDIGITS                                     ! Digits of operand C
1     ) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Multiply two decimal numbers.  C := A * B
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       ASIGN.rv.r      0 = operand A is positive, 1 = negative
1 !       AEXP.rl.r       Power of 10 by which to multiply the operand A
1 !                       digits to get the absolute value of operand A.
1 !                       E.g., AEXP = 1, ADIGITS = 123 gives 1230.
1 !       ADIGITS.rnu.d   Descriptor for the digits of operand A
1 !       BSIGN.rv.r      0 = operand B is positive, 1 = negative
1 !       BEXP.rl.r       Power of 10 by which to multiply the operand B
1 !                       digits to get the absolute value of operand B.
1 !                       E.g., BEXP = -1, BDIGITS = 123 gives 12.3.
1 !       BDIGITS.rnu.d   Descriptor for the digits of operand B
1 !       CSIGN.wl.r      0 = operand C is positive, 1 = negative
1 !       CEXP.wl.r       Power of 10 by which to multiply the operand C
1 !                       digits to get the absolute value of operand C.
1 !                       E.g., CEXP = 0, CDIGITS = 123 gives 123.
1 !       CDIGITS.wnu.d   Descriptor for the digits of operand C
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       May allocate space for the CDIGITS string.
1 !       Signals if storage is exhausted.
1 !--
1
2     BEGIN
2
2     MAP
```

```
;  881          0971  2             ADIGITS : REF BLOCK [8, BYTE],
;  882          0972  2             BDIGITS : REF BLOCK [8, BYTE],
;  883          0973  2             CDIGITS : REF BLOCK [8, BYTE];
;  884          0974  2
;  885          0975  2         LOCAL
;  886          0976  2 !+
;  887          0977  2 ! Internal form of A.
;  888          0978  2 !-
;  889          0979  2             A_DESC : BLOCK [8, BYTE] VOLATILE,
;  890          0980  2             ABUF : REF VECTOR [65535, BYTE],
;  891          0981  2             A_LEN,
;  892          0982  2             A_SIGN,
;  893          0983  2 !+
;  894          0984  2 ! Internal form of B.
;  895          0985  2 !-
;  896          0986  2             B_DESC : BLOCK [8, BYTE] VOLATILE,
;  897          0987  2             BBUF : REF VECTOR [65535, BYTE],
;  898          0988  2             B_LEN,
;  899          0989  2             B_SIGN,
;  900          0990  2 !+
;  901          0991  2 ! Local copy of result.
;  902          0992  2 !-
;  903          0993  2             RSIGN,
;  904          0994  2             REXP,
;  905          0995  2             R_DESC : BLOCK [8, BYTE] VOLATILE,
;  906          0996  2             RBUF : REF VECTOR [65535, BYTE],
;  907          0997  2             R_LEN,
;  908          0998  2
;  909          0999  2 !+
;  910          1000  2 ! The following are locals for the call to LIB$ANALYZE_SDESC.
;  911          1001  2 !-
;  912          1002  2             CBUF,
;  913          1003  2             C_LEN,
;  914          1004  2             STATUS;
;  915          1005  2
;  916          1006  2         BUILTIN
;  917          1007  2             ACTUALCOUNT;
;  918          1008  2
;  919          1009  2 !+
;  920          1010  2 ! Enable a handler to free the local strings in case of an error.
;  921          1011  2 !-
;  922          1012  2
;  923          1013  2         ENABLE
;  924          1014  2             FREE_STRINGS (A_DESC, B_DESC, R_DESC);
;  925          1015  2
;  926          1016  2 !+
;  927          1017  2 ! Check the number of arguments.
;  928          1018  2 !-
;  929          1019  2
;  930          1020  3         IF (ACTUALCOUNT () LSS 9)
;  931          1021  2         THEN
;  932          1022  3             BEGIN
;  933          1023  3
;  934          1024  3             LOCAL
;  935          1025  3                 ROUT_NAME_DESC : BLOCK [8, BYTE];
;  936          1026  3
;  937          1027  3             ROUT_NAME_DESC [DSC$W_LENGTH] = 7;
```

```
3          ROUT_NAME_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
3          ROUT_NAME_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
3          ROUT_NAME_DESC [DSC$A_POINTER] = UPLIT (%ASCII'STR$MUL');
3          LIB$STOP (STR$_WRONUMARG, 2, ACTUALCOUNT (), ROUT_NAME_DESC);
2          END;
2
2  !+
2  ! Copy the A and B operands.
2  !-
2      A_DESC [DSC$W_LENGTH] = 0;
2      A_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2      A_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2      A_DESC [DSC$A_POINTER] = 0;
2  !+
2  ! Compute the length of operand A.  Only the leading digits count.
2  ! First call LIB$ANALYZE_SDESC to ensure that the input descriptor
2  ! is valid.  If it is, then ABUF will contain the address of the
2  ! first byte of the string, and A_LEN will contain its length.
2  !-
2
2      STATUS = LIB$ANALYZE_SDESC (.ADIGITS,A_LEN,ABUF);
2      IF .STATUS NEQ SS$_NORMAL
2        THEN
2          LIB$STOP (LIB$_INVARG);
2
2  !+
2  ! Check here also for the CDIGITS descriptor before we get too
2  ! involved in the routine.
2  !-
2
2      STATUS = LIB$ANALYZE_SDESC (.CDIGITS,C_LEN,CBUF);
2      IF .STATUS NEQ SS$_NORMAL
2        THEN
2          LIB$STOP (LIB$_INVARG);
2
2      A_LEN = 0;
2      A_SIGN = ..ASIGN;
3      BEGIN
3
3      LOCAL
3          SCAN_DONE;
3
3      SCAN_DONE = 0;
3
3      DO
4          BEGIN
4
5          IF (.A_LEN EQLU .ADIGITS [DSC$W_LENGTH])
4          THEN
4              SCAN_DONE = 1
4          ELSE
4
5              IF ((.ABUF [.A_LEN] GEQ %C'0') AND (.ABUF [.A_LEN] LEQ %C'9'))
4              THEN
4                  A_LEN = .A_LEN + 1
4              ELSE
4                  SCAN_DONE = 1;
```

```
      END
  UNTIL (.SCAN_DONE);

  END;
  STR$GET1_DX (A_LEN, A_DESC);
  ABUF = .A_DESC [DSC$A_POINTER];
  CH$MOVE (.A_LEN, .ADIGITS [DSC$A_POINTER], ABUF [0]);
  B_DESC [DSC$W_LENGTH] = 0;
  B_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
  B_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
  B_DESC [DSC$A_POINTER] = 0;
!+
! Compute the length of operand B.  Only the leading digits count.
! First call LIB$ANALYZE_SDESC to ensure that the input descriptor
! is valid.  If it is, then BBUF will contain the address of the
! first byte of the string, and B_LEN will contain its length.
!-

  STATUS = LIB$ANALYZE_SDESC (.BDIGITS,B_LEN,BBUF);
  IF .STATUS NEQ SS$_NORMAL
    THEN
      LIB$STOP (LIB$_INVARG);
  B_LEN = 0;
  B_SIGN = ..BSIGN;
  BEGIN

  LOCAL
      SCAN_DONE;

  SCAN_DONE = 0;

  DO
      BEGIN

      IF (.B_LEN EQLU .BDIGITS [DSC$W_LENGTH])
      THEN
          SCAN_DONE = 1
      ELSE

          IF ((.BBUF [.B_LEN] GEQ %C'0') AND (.BBUF [.B_LEN] LEQ %C'9'))
          THEN
              B_LEN = .B_LEN + 1
          ELSE
              SCAN_DONE = 1;

      END
  UNTIL (.SCAN_DONE);

  END;
  STR$GET1_DX (B_LEN, B_DESC);
  BBUF = .B_DESC [DSC$A_POINTER];
  CH$MOVE (.B_LEN, .BDIGITS [DSC$A_POINTER], BBUF [0]);
!+
! Set the accumulator to zero.
!-
  R_DESC [DSC$W_LENGTH] = 0;
```

```
; 1052          1142  2       R_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
; 1053          1143  2       R_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
; 1054          1144  2       R_DESC [DSC$A_POINTER] = 0;
; 1055          1145  2       STR$GET1_DX (%REF (1), R_DESC);
; 1056          1146  2       RBUF = .R_DESC [DSC$A_POINTER];
; 1057          1147  2       R_LEN = .R_DESC [DSC$W_LENGTH];
; 1058          1148  2       RBUF [0] = %C'0';
; 1059          1149  2       RSIGN = 0;
; 1060          1150  2       REXP = 0;
; 1061          1151  2   !+
; 1062          1152  2   ! Go through each digit of B, adding appropriately shifted A to
; 1063          1153  2   ! R the indicated number of times.  This is like the old mechanical
; 1064          1154  2   ! adding machines.
; 1065          1155  2   !-
; 1066          1156  2
; 1067          1157  2       INCR POS FROM 0 TO .B_LEN - 1 DO
; 1068          1158  3           BEGIN
; 1069          1159  3
; 1070          1160  3           LOCAL
; 1071          1161  3               DIGIT;
; 1072          1162  3
; 1073          1163  3           DIGIT = .BBUF [(.B_LEN - 1) - .POS];
; 1074          1164  3
; 1075          1165  3           DECR COUNTER FROM .DIGIT TO %C'1' DO
; 1076          1166  3               STR$ADD (%REF (0), POS, A_DESC, RSIGN, REXP, R_DESC, RSIGN, REXP, R_DESC);
; 1077          1167  3
; 1078          1168  2           END;
; 1079          1169  2
; 1080          1170  2   !+
; 1081          1171  2   ! Compute the exponent and sign of the result.
; 1082          1172  2   !-
; 1083          1173  2       REXP = .REXP + (..AEXP + ..BEXP);
; 1084          1174  2       RSIGN = (IF (.A_SIGN EQL .B_SIGN) THEN 0 ELSE 1);
; 1085          1175  2   !+
; 1086          1176  2   ! Return the result to the caller.  Because it is the output of STR$ADD
; 1087          1177  2   ! it is already in normal form.
; 1088          1178  2   !-
; 1089          1179  2       .CSIGN = .RSIGN;
; 1090          1180  2       .CEXP = .REXP;
; 1091          1181  2
; 1092          1182  2   !+
; 1093          1183  2   ! Call CHK_STR_TYPE to determine if we need to pad the number with
; 1094          1184  2   ! leading zeroes depending on the string type.
; 1095          1185  2   !-
; 1096          1186  2
; 1097          1187  2       R_LEN = .R_DESC[DSC$W_LENGTH];
; 1098          1188  2       CHK_STR_TYPE (.R_DESC[DSC$A_POINTER],R_LEN,.CDIGITS);
; 1099          1189  2
; 1100          1190  2
; 1101          1191  2   !   STR$COPY_DX (.CDIGITS, R_DESC);
; 1102          1192  2   !+
; 1103          1193  2   ! Free our strings.
; 1104          1194  2   !-
; 1105          1195  2       STR$FREE1_DX (R_DESC);
; 1106          1196  2       STR$FREE1_DX (A_DESC);
; 1107          1197  2       STR$FREE1_DX (B_DESC);
; 1108          1198  1       END;                                    ! end of STR$MUL
```

```
                                          00551          .BLKB    3
          00  4C  55  4D  24  52  54  53  00554 P.AAF:   .ASCII   \STR$MUL\<0>                                 ;

                                    0FFC  00000          .ENTRY   STR$MUL, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,- ; 0914
                                                                  R11                                          ;
                   5B 00000000G  00   9E  00002          MOVAB    STR$GET1_DX, R11                             ;
                   5A 00000000G  8F   D0  00009          MOVL     #LIB$_INVARG, R10                            ;
                   59 00000000G  00   9E  00010          MOVAB    LIB$ANALYZE_SDESC, R9                        ;
                   58 00000000G  00   9E  00017          MOVAB    LIB$STOP, R8                                 ;
                   5E        B4  AE   9E  0001E          MOVAB    -76(SP), SP                                  ;
                             34  AE   7C  00022          CLRQ     R_DESC                                       ; 0968
                             3C  AE   7C  00025          CLRQ     B_DESC                                       ;
                             44  AE   7C  00028          CLRQ     A_DESC                                       ;
                   6D      01D9  CF   DE  0002B          MOVAL    17$, (FP)                                    ;
                   09            6C   91  00030          CMPB     (AP), #9                                     ; 1020
                                 1E   1E  00033          BGEQU    1$                                           ;
              2C   AE  010E0007  8F   D0  00035          MOVL     #17694727, ROUT_NAME_DESC                    ; 1027
              30   AE        B8  AF   9E  0003D          MOVAB    P.AAF, ROUT_NAME_DESC+4                      ; 1030
                             2C  AE   9F  00042          PUSHAB   ROUT_NAME_DESC                               ; 1031
                   7E            6C   9A  00045          MOVZBL   (AP), -(SP)                                  ;
                                 02   DD  00048          PUSHL    #2                                           ;
                      00000000G  8F   DD  0004A          PUSHL    #STR$_WRONUMARG                              ;
                   68            04   FB  00050          CALLS    #4, LIB$STOP                                 ;
                             44  AE   B4  00053 1$:      CLRW     A_DESC                                       ; 1037
              46   AE            0F   90  00056          MOVB     #15, A_DESC+2                                ; 1038
              47   AE            02   90  0005A          MOVB     #2, A_DESC+3                                 ; 1039
                             48  AE   D4  0005E          CLRL     A_DESC+4                                     ; 1040
                             04  AE   9F  00061          PUSHAB   ABUF                                         ; 1048
                             14  AE   9F  00064          PUSHAB   A_LEN                                        ;
                   52        0C  AC   D0  00067          MOVL     ADIGITS, R2                                  ;
                                 52   DD  0006B          PUSHL    R2                                           ;
                   69            03   FB  0006D          CALLS    #3, LIB$ANALYZE_SDESC                        ;
                   56            50   DC  00070          MOVL     R0, STATUS                                   ;
                   01            56   D1  00073          CMPL     STATUS, #1                                   ; 1049
                                 05   13  00076          BEQL     2$                                           ;
                                 5A   DD  00078          PUSHL    R10                                          ; 1051
                   68            01   FB  0007A          CALLS    #1, LIB$STOP                                 ;
                             08  AE   9F  0007D 2$:      PUSHAB   CBUF                                         ; 1058
                             10  AE   9F  00080          PUSHAB   C_LEN                                        ;
                             24  AC   DD  00083          PUSHL    CDIGITS                                      ;
                   69            03   FB  00086          CALLS    #3, LIB$ANALYZE_SDESC                        ;
                   56            50   D0  00089          MOVL     R0, STATUS                                   ;
                   01            56   D1  0008C          CMPL     STATUS, #1                                   ; 1059
                                 05   13  0008F          BEQL     3$                                           ;
                                 5A   DD  00091          PUSHL    R10                                          ; 1061
                   68            01   FB  00093          CALLS    #1, LIB$STOP                                 ;
                             10  AE   D4  00096 3$:      CLRL     A_LEN                                        ; 1063
                   57        04  BC   D0  00099          MOVL     @ASIGN, A_SIGN                               ; 1064
                                 51   D4  0009D          CLRL     SCAN_DONE                                    ; 1070
  10   AE         62         10  00   ED  0009F 4$:      CMPZV    #0, #16, (R2), A_LEN                         ; 1075
                                 15   13  000A5          BEQL     5$                                           ;
                  50     04  AE  10  AE   C1  000A7      ADDL3    A_LEN, ABUF, R0                              ; 1080
```

```
                                          30             60   91  000AD          CMPB     (R0), #48                          ;
                                                         0A   1F  000B0          BLSSU    5$                                 ;
                                          39             60   91  000B2          CMPB     (R0), #57                          ;
                                                         05   1A  000B5          BGTRU    5$                                 ;
                                                    10   AE   D6  000B7          INCL     A_LEN                              ; 1082
                                                         03   11  000BA          BRB      6$                                 ;
                                          51             01   D0  000BC 5$:      MOVL     #1, SCAN_DONE                      ; 1084
                                          DD             51   E9  000BF 6$:      BLBC     SCAN_DONE, 4$                      ; 1087
                                                    44   AE   9F  000C2          PUSHAB   A_DESC                             ; 1090
                                                    14   AE   9F  000C5          PUSHAB   A_LEN                              ;
                                          6B             02   FB  000C8          CALLS    #2, STR$GET1_DX                    ;
                                    04    AE        48   AE   D0  000CB          MOVL     A_DESC+4, ABUF                     ; 1091
                    04    BE        04    B2        10   AE   28  000D0          MOVC3    A_LEN, @4(R2), @ABUF               ; 1092
                                                    3C   AE   B4  000D7          CLRW     B_DESC                             ; 1093
                                    3E    AE             0F   90  000DA          MOVB     #15, B_DESC+2                      ; 1094
                                    3F    AE             02   90  000DE          MOVB     #2, B_DESC+3                       ; 1095
                                                    40   AE   D4  000E2          CLRL     B_DESC+4                           ; 1096
                                                    14   AE   9F  000E5          PUSHAB   BBUF                               ; 1104
                                                    1C   AE   9F  000E8          PUSHAB   B_LEN                              ;
                                          52        18   AC   D0  000EB          MOVL     BDIGITS, R2                        ;
                                                         52   DD  000EF          PUSHL    R2                                 ;
                                          69             03   FB  000F1          CALLS    #3, LIB$ANALYZE_SDESC              ;
                                          56             50   D0  000F4          MOVL     R0, STATUS                         ;
                                          01             56   D1  000F7          CMPL     STATUS, #1                         ; 1105
                                                         05   13  000FA          BEQL     7$                                 ;
                                                         5A   DD  000FC          PUSHL    R10                                ; 1107
                                          68             01   FB  000FE          CALLS    #1, LIB$STOP                       ;
                                                    18   AE   D4  00101 7$:      CLRL     B_LEN                              ; 1108
                                          56        10   BC   D0  00104          MOVL     @BSIGN, B_SIGN                     ; 1109
                                                         51   D4  00108          CLRL     SCAN_DONE                          ; 1115
    18    AE                62                  10       00   ED  0010A 8$:      CMPZV    #0, #16, (R2), B_LEN               ; 1120
                                                         15   13  00110          BEQL     9$                                 ;
                          50        14    AE        18   AE   C1  00112          ADDL3    B_LEN, BBUF, R0                    ; 1125
                                          30             60   91  00118          CMPB     (R0), #48                          ;
                                                         0A   1F  0011B          BLSSU    9$                                 ;
                                          39             60   91  0011D          CMPB     (R0), #57                          ;
                                                         05   1A  00120          BGTRU    9$                                 ;
                                                    18   AE   D6  00122          INCL     B_LEN                              ; 1127
                                                         03   11  00125          BRB      10$                                ;
                                          51             01   D0  00127 9$:      MOVL     #1, SCAN_DONE                      ; 1129
                                          DD             51   E9  0012A 10$:     BLBC     SCAN_DONE, 8$                      ; 1132
                                                    3C   AE   9F  0012D          PUSHAB   B_DESC                             ; 1135
                                                    1C   AE   9F  00130          PUSHAB   B_LEN                              ;
                                          6B             02   FB  00133          CALLS    #2, STR$GET1_DX                    ;
                                    14    AE        40   AE   D0  00136          MOVL     B_DESC+4, BBUF                     ; 1136
                    14    BE        04    B2        18   AE   28  0013B          MOVC3    B_LEN, @4(R2), @BBUF               ; 1137
                                                    34   AE   B4  00142          CLRW     R_DESC                             ; 1141
                                    36    AE             0F   90  00145          MOVB     #15, R_DESC+2                      ; 1142
                                    37    AE             02   90  00149          MOVB     #2, R_DESC+3                       ; 1143
                                                    38   AE   D4  0014D          CLRL     R_DESC+4                           ; 1144
                                                    34   AE   9F  00150          PUSHAB   R_DESC                             ; 1145
                                    04    AE             01   D0  00153          MOVL     #1, 4(SP)                          ;
                                                    04   AE   9F  00157          PUSHAB   4(SP)                              ;
                                          6B             02   FB  0015A          CALLS    #2, STR$GET1_DX                    ;
                                          50        38   AE   D0  0015D          MOVL     R_DESC+4, RBUF                     ; 1146
                                    28    AE        34   AE   3C  00161          MOVZWL   R_DESC, R_LEN                      ; 1147
                                          60             30   90  00166          MOVB     #48, (RBUF)                        ; 1148
                          1C   AE  7C 00169          CLRQ     REXP                          ; 1150
              24   AE          01  CE 0016C          MNEGL    #1, POS                       ; 1157
                               3A  11 00170          BRB      14$                           ;
   50         18   AE     24   AE  C3 00172  11$:    SUBL3    POS, B_LEN, R0                ; 1163
                   50     14   AE  C0 00178          ADDL2    BBUF, R0                      ;
                   52     FF   A0  9A 0017C          MOVZBL   -1(R0), DIGIT                 ;
                               25  11 00180          BRB      13$                           ; 1165
                          34   AE  9F 00182  12$:    PUSHAB   R_DESC                        ; 1166
                          20   AE  9F 00185          PUSHAB   REXP                          ;
                          28   AE  9F 00188          PUSHAB   RSIGN                         ;
                          40   AE  9F 0018B          PUSHAB   R_DESC                        ;
                          2C   AE  9F 0018E          PUSHAB   REXP                          ;
                          34   AE  9F 00191          PUSHAB   RSIGN                         ;
                          5C   AE  9F 00194          PUSHAB   A_DESC                        ;
                          40   AE  9F 00197          PUSHAB   POS                           ;
                          20   AE  D4 0019A          CLRL     32(SP)                        ;
                          20   AE  9F 0019D          PUSHAB   32(SP)                        ;
            FA1B   CF          09  FB 001A0          CALLS    #9, STR$ADD                   ;
                               52  D7 001A5          DECL     COUNTER                       ;
                   31          52  D1 001A7  13$:    CMPL     COUNTER, #49                  ;
                               D6  18 001AA          BGEQ     12$                           ;
   C0         24   AE     18   AE  F2 001AC  14$:    AOBLSS   B_LEN, POS, 11$               ; 1157
   50         08   BC     14   BC  C1 001B2          ADDL3    @BEXP, @AEXP, R0              ; 1173
              1C   AE          50  C0 001B8          ADDL2    R0, REXP                      ;
                   56          57  D1 001BC          CMPL     A_SIGN, B_SIGN                ; 1174
                               04  12 001BF          BNEQ     15$                           ;
                               50  D4 001C1          CLRL     R0                            ;
                               03  11 001C3          BRB      16$                           ;
                   50          01  D0 001C5  15$:    MOVL     #1, R0                        ;
              20   AE          50  D0 001C8  16$:    MOVL     R0, RSIGN                     ;
              1C   BC     20   AE  D0 001CC          MOVL     RSIGN, @CSIGN                 ; 1179
              20   BC     1C   AE  D0 001D1          MOVL     REXP, @CEXP                   ; 1180
              28   AE     34   AE  3C 001D6          MOVZWL   R_DESC, R_LEN                 ; 1187
                          24   AC  DD 001DB          PUSHL    CDIGITS                       ; 1188
                          2C   AE  9F 001DE          PUSHAB   R_LEN                         ;
                          40   AE  DD 001E1          PUSHL    R_DESC+4                      ;
           0000V   CF          03  FB 001E4          CALLS    #3, CHK_STR_TYPE              ;
                          34   AE  9F 001E9          PUSHAB   R_DESC                        ; 1195
       00000000G   00          01  FB 001EC          CALLS    #T, STR$FREE1_DX              ;
                          44   AE  9F 001F3          PUSHAB   A_DESC                        ; 1196
       00000000G   00          01  FB 001F6          CALLS    #T, STR$FREE1_DX              ;
                          3C   AE  9F 001FD          PUSHAB   B_DESC                        ; 1197
       00000000G   00          01  FB 00200          CALLS    #T, STR$FREE1_DX              ;
                                   04 00207          RET                                    ; 1198
                                 0000 00208  17$:    .WORD    Save nothing                  ; 0968
                   50     08   AC  D0 0020A          MOVL     8(AP), R0                     ;
                   50     04   A0  D0 0020E          MOVL     4(R0), R0                     ;
                          E8   A0  9F 00212          PUSHAB   R_DESC                        ;
                          F0   A0  9F 00215          PUSHAB   B_DESC                        ;
                          F8   A0  9F 00218          PUSHAB   A_DESC                        ;
                               03  DD 0021B          PUSHL    #3                            ;
                               5E  DD 0021D          PUSHL    SP                            ;
                   7E     04   AC  7D 0021F          MOVQ     4(AP), -(SP)                  ;
           0000V   CF          03  FB 00223          CALLS    #3, FREE_STRINGS              ;
                                   04 00228          RET                                    ;

; Routine Size:  553 bytes,    Routine Base:  _STR$CODE + 055C
; 1109          1199  1
```

```
; F

; 1
```

```
1 GLOBAL ROUTINE STR$RECIP (                      ! Take the reciprocal of a string
1               ASIGN,                           ! Sign of operand A
1               AEXP,                            ! Decimal exponent of operand A
1               ADIGITS,                         ! Digits of operand A
1               BSIGN,                           ! Sign of operand B
1               BEXP,                            ! Decimal exponent of operand B
1               BDIGITS,                         ! Digits of operand B
1               CSIGN,                           ! Sign of operand C
1               CEXP,                            ! Decimal exponent of operand C
1               CDIGITS                          ! Digits of operand C
1       ) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Take the reciprocal of A, to precision B.  C := 1 / A
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       ASIGN.rv.l      0 = operand A is positive, 1 = negative
1 !       AEXP.rl.l       Power of 10 by which to multiply the operand A
1 !                       digits to get the absolute value of operand A.
1 !                       E.g., AEXP = 1, ADIGITS = 123 gives 1230.
1 !       ADIGITS.rnu.d   Descriptor for the digits of operand A
1 !       BSIGN.rv.l      0 = operand B is positive, 1 = negative
1 !       BEXP.rl.r       Power of 10 by which to multiply the operand B
1 !                       digits to get the absolute value of operand B.
1 !                       E.g., BEXP = -1, BDIGITS = 123 gives 12.3.
1 !       BDIGITS.rnu.d   Descriptor for the digits of operand B
1 !       CSIGN.wl.r      0 = operand C is positive, 1 = negative
1 !       CEXP.wl.r       Power of 10 by which to multiply the operand C
1 !                       digits to get the absolute value of operand C.
1 !                       E.g., CEXP = 0, CDIGITS = 123 gives 123.
1 !       CDIGITS.wnu.d   Descriptor for the digits of operand C
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       May allocate space for the CDIGITS string.
1 !       Signals if memory is exausted.
1 !       Signals Division by zero if operand A is zero.
1 !
1 !--
1
2     BEGIN
```

```
; 1168          1257  2
; 1169          1258  2       MAP
; 1170          1259  2           ADIGITS : REF BLOCK [8, BYTE],
; 1171          1260  2           BDIGITS : REF BLOCK [8, BYTE],
; 1172          1261  2           CDIGITS : REF BLOCK [8, BYTE];
; 1173          1262  2
; 1174          1263  2       LOCAL
; 1175          1264  2   !+
; 1176          1265  2   ! Internal form of A.
; 1177          1266  2   !-
; 1178          1267  2           A_DESC : BLOCK [8, BYTE] VOLATILE,
; 1179          1268  2           A_BUF : REF VECTOR [65535, BYTE],
; 1180          1269  2           A_LEN,
; 1181          1270  2           A_SIGN,
; 1182          1271  2   !+
; 1183          1272  2   ! Internal form of B.
; 1184          1273  2   !-
; 1185          1274  2           B_DESC : BLOCK [8, BYTE] VOLATILE,
; 1186          1275  2           B_BUF : REF VECTOR [65535, BYTE],
; 1187          1276  2           B_LEN,
; 1188          1277  2           B_SIGN,
; 1189          1278  2   !+
; 1190          1279  2   ! The following are various auxiliary variables required to do the division
; 1191          1280  2   ! and check for its completion.
; 1192          1281  2   !-
; 1193          1282  2           X_SIGN,
; 1194          1283  2           X_EXP,
; 1195          1284  2           X_DESC : BLOCK [8, BYTE] VOLATILE,
; 1196          1285  2           X_BUF : REF VECTOR [65535, BYTE],
; 1197          1286  2           X2_SIGN,
; 1198          1287  2           X2_EXP,
; 1199          1288  2           X2_DESC : BLOCK [8, BYTE] VOLATILE,
; 1200          1289  2           X2_BUF : REF VECTOR [65535, BYTE],
; 1201          1290  2           Q_SIGN,
; 1202          1291  2           Q_EXP,
; 1203          1292  2           Q_DESC : BLOCK [8, BYTE] VOLATILE,
; 1204          1293  2           Q_BUF : REF VECTOR [65535, BYTE],
; 1205          1294  2           Q_LEN,                                  ! Added for call to CHK_STR_TYPE
; 1206          1295  2           XA_SIGN,
; 1207          1296  2           XA_EXP,
; 1208          1297  2           XA_DESC : BLOCK [8, BYTE] VOLATILE,
; 1209          1298  2           XA_BUF : REF VECTOR [65535, BYTE],
; 1210          1299  2           DELTA_SIGN,
; 1211          1300  2           DELTA_EXP,
; 1212          1301  2           DELTA_DESC : BLOCK [8, BYTE] VOLATILE,
; 1213          1302  2           DELTA_BUF : REF VECTOR [65535, BYTE],
; 1214          1303  2           ONE_DESC : BLOCK [8, BYTE],
; 1215          1304  2           ONE_BUF : VECTOR [1, BYTE],
; 1216          1305  2           ITER_DONE,                              ! 1 = the division process is done, exit its loop
; 1217          1306  2           POS,                                    ! Power of ten by which we are dividing (shifting right)
; 1218          1307  2
; 1219          1308  2   !+
; 1220          1309  2   ! The following are locals needed for calls to LIB$ANALYZE_SDESC.
; 1221          1310  2   !-
; 1222          1311  2           CBUF,
; 1223          1312  2           C_LEN,
; 1224          1313  2           STATUS;
```

```
2
2       BUILTIN
2           ACTUALCOUNT;
2
2   !+
2   ! Enable a handler to free the local strings in case of an error.
2   !-
2
2       ENABLE
2           FREE_STRINGS (A_DESC, B_DESC, X_DESC, X2_DESC, Q_DESC, XA_DESC, DELTA_DESC);
2
2   !+
2   ! Check for the proper number of arguments.
2   !-
2
3       IF (ACTUALCOUNT () LSS 9)
3       THEN
3           BEGIN
3
3           LOCAL
3               ROUT_NAME_DESC : BLOCK [8, BYTE];
3
3           ROUT_NAME_DESC [DSC$W_LENGTH] = 9;
3           ROUT_NAME_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
3           ROUT_NAME_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
3           ROUT_NAME_DESC [DSC$A_POINTER] = UPLIT (%ASCII'STR$RECIP');
3           LIB$STOP (STR$_WRONUMARG, 2, ACTUALCOUNT (), ROUT_NAME_DESC);
2           END;
2
2   !+
2   ! Copy the A and B operands.
2   !-
2       A_DESC [DSC$W_LENGTH] = 0;
2       A_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2       A_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2       A_DESC [DSC$A_POINTER] = 0;
2   !+
2   ! Compute the length of operand A.  Only the leading digits count.
2   ! First call LIB$ANALYZE_SDESC to ensure that the input descriptor
2   ! is valid.  If it is, then A_BUF will contain the address of the
2   ! first byte of the string, and A_LEN will contain its length.
2   !-
2
2       STATUS = LIB$ANALYZE_SDESC (.ADIGITS,A_LEN,A_BUF);
2       IF .STATUS NEQ SS$_NORMAL
2         THEN
2           LIB$STOP (LIB$_INVARG);
2
2   !+
2   ! Also check here for the CDIGITS descriptor before getting too
2   ! involved in the routine.
2   !-
2
2       STATUS = LIB$ANALYZE_SDESC (.CDIGITS,C_LEN,CBUF);
2       IF .STATUS NEQ SS$_NORMAL
2         THEN
2           LIB$STOP (LIB$_INVARG);
```

```
2
2       A_LEN = 0;
2       A_SIGN = ..ASIGN;
3       BEGIN
3
3       LOCAL
3           SCAN_DONE;
3
3       SCAN_DONE = 0;
3
3       DO
4           BEGIN
4
5           IF (.A_LEN EQLU .ADIGITS [DSC$W_LENGTH])
4           THEN
4               SCAN_DONE = 1
4           ELSE
4
5               IF ((.A_BUF [.A_LEN] GEQ %C'0') AND (.A_BUF [.A_LEN] LEQ %C'9'))
4               THEN
4                   A_LEN = .A_LEN + 1
4               ELSE
4                   SCAN_DONE = 1;
4
4           END
3       UNTIL (.SCAN_DONE);
3
2       END;
2       STR$GET1_DX (A_LEN, A_DESC);
2       A_BUF = .A_DESC [DSC$A_POINTER];
2       CH$MOVE (.A_LEN, .ADIGITS [DSC$A_POINTER], A_BUF [0]);
2   !+
2   ! If operand A is zero, fail.
2   !-
2
2       IF CH$EQL (1, CH$PTR (UPLIT ('0')), .A_LEN, A_BUF [0], %C'0') THEN LIB$STOP (STR$_DIVBY_ZER);
2
2       B_DESC [DSC$W_LENGTH] = 0;
2       B_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2       B_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2       B_DESC [DSC$A_POINTER] = 0;
2   !+
2   ! Compute the length of operand B.  Only the leading digits count.
2   ! First call LIB$ANALYZE_SDESC to ensure that the input descriptor
2   ! is valid.  If it is, then B_BUF will contain the address of the
2   ! first byte of the string, and B_LEN will contain its length.
2   !-
2
2       STATUS = LIB$ANALYZE_SDESC (.BDIGITS,B_LEN,B_BUF);
2       IF .STATUS NEQ SS$_NORMAL
2         THEN
2           LIB$STOP (LIB$_INVARG);
2       B_LEN = 0;
2       B_SIGN = ..BSIGN;
3       BEGIN
3
3       LOCAL
```

```
3              SCAN_DONE;
3
3          SCAN_DONE = 0;
3
3          DO
4              BEGIN
4
5              IF (.B_LEN EQLU .BDIGITS [DSC$W_LENGTH])
4              THEN
4                  SCAN_DONE = 1
4              ELSE
4
5                  IF ((.B_BUF [.B_LEN] GEQ %C'0') AND (.B_BUF [.B_LEN] LEQ %C'9'))
4                  THEN
4                      B_LEN = .B_LEN + 1
4                  ELSE
4                      SCAN_DONE = 1;
4
4              END
3          UNTIL (.SCAN_DONE);
3
2          END;
2          STR$GET1_DX (B_LEN, B_DESC);
2          B_BUF = .B_DESC [DSC$A_POINTER];
2          CH$MOVE (.B_LEN, .BDIGITS [DSC$A_POINTER], B_BUF [0]);
2      !+
2      ! Initialize the auxiliary variables.
2      !-
2          X_DESC [DSC$W_LENGTH] = 0;
2          X_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2          X_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2          X_DESC [DSC$A_POINTER] = 0;
2          STR$GET1_DX (%REF (1), X_DESC);
2          X_BUF = .X_DESC [DSC$A_POINTER];
2          X_BUF [0] = %C'1';
2          X_SIGN = 0;
2          X_EXP = 0;
2      !
2          X2_DESC [DSC$W_LENGTH] = 0;
2          X2_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2          X2_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2          X2_DESC [DSC$A_POINTER] = 0;
2          STR$GET1_DX (%REF (1), X2_DESC);
2          X2_BUF = .X2_DESC [DSC$A_POINTER];
2          X2_BUF [0] = %C'0';
2          X2_SIGN = 0;
2          X2_EXP = 0;
2      !
2          Q_DESC [DSC$W_LENGTH] = 0;
2          Q_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2          Q_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2          Q_DESC [DSC$A_POINTER] = 0;
2          STR$GET1_DX (%REF (1), Q_DESC);
2          Q_BUF = .Q_DESC [DSC$A_POINTER];
2          Q_BUF [0] = %C'0';
2          Q_SIGN = 0;
2          Q_EXP = 0;
```

```
2  !
2       XA_DESC [DSC$W_LENGTH] = 0;
2       XA_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2       XA_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2       XA_DESC [DSC$A_POINTER] = 0;
2       STR$GET1_DX (%REF (1), XA_DESC);
2       XA_BUF = .XA_DESC [DSC$A_POINTER];
2       XA_BUF [0] = %C'0';
2       XA_SIGN = 0;
2       XA_EXP = 0;
2  !
2       DELTA_DESC [DSC$W_LENGTH] = 0;
2       DELTA_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2       DELTA_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2       DELTA_DESC [DSC$A_POINTER] = 0;
2       STR$GET1_DX (%REF (1), DELTA_DESC);
2       DELTA_BUF = .DELTA_DESC [DSC$A_POINTER];
2       DELTA_BUF [0] = %C'0';
2       DELTA_SIGN = 0;
2       DELTA_EXP = 0;
2  !
2       ONE_DESC [DSC$W_LENGTH] = 1;
2       ONE_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2       ONE_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
2       ONE_DESC [DSC$A_POINTER] = ONE_BUF;
2       ONE_BUF [0] = %C'1';
2  !+
2  ! Decide on the best position to start forming the quotient.  Unless
2  ! the divisor is 1, the first subtract will cause X to go negative
2  ! and force us to back off.
2  !-
2       POS = -.X_EXP;
2  !+
2  ! Iterate until we are close to the quotient.
2  ! If B = 0, this will take a long time.
2  !-
2       ITER_DONE = 0;
2
2       DO
3           BEGIN
3           STR$ADD (X_SIGN, X_EXP, X_DESC,          !
3               %REF (1), %REF (..AEXP + .POS), A_DESC,      !
3               X_SIGN, X_EXP, X_DESC);
3  !+
3  ! If we have gone negative, back off.  Otherwise increase the quotient.
3  !-
3
4           IF (.X_SIGN)
3           THEN
4               BEGIN
4               STR$ADD (X_SIGN, X_EXP, X_DESC,      !
4                   %REF (0), %REF (..AEXP + .POS), A_DESC,        !
4                   X_SIGN, X_EXP, X_DESC);
4  !+
4  ! Go down to the next lower digit
4  !-
4               POS = .POS - 1;
```

```
4 !+
4 ! Now see if we are close enough to the reciprocal.
4 !-
4            STR$MUL (Q_SIGN, Q_EXP, Q_DESC,      !
4                %REF (0), .AEXP, A_DESC,         !
4                XA_SIGN, XA_EXP, XA_DESC);
4            STR$ADD (XA_SIGN, XA_EXP, XA_DESC,  !
4                %REF (1), %REF (0), ONE_DESC,    !
4                DELTA_SIGN, DELTA_EXP, DELTA_DESC);
4            DELTA_SIGN = 0;
4            STR$ADD (DELTA_SIGN, DELTA_EXP, DELTA_DESC,          !
4                %REF (1), .BEXP, B_DESC,         !
4                X2_SIGN, X2_EXP, X2_DESC);
4
5            IF (.X2_SIGN)
4            THEN
4                ITER_DONE = 1
4            ELSE
4
5                IF (.DELTA_DESC [DSC$W_LENGTH] EQLU 1)
4                THEN
5                    BEGIN
5
5                    LOCAL
5                        DELTA_BUF : REF VECTOR [65535, BYTE];
5
5                    DELTA_BUF = .DELTA_DESC [DSC$A_POINTER];
5
5                    IF (.DELTA_BUF [0] EQL %C'0') THEN ITER_DONE = 1;
5
4                    END;
4
4            END
3        ELSE
4            BEGIN
4            STR$ADD (Q_SIGN, Q_EXP, Q_DESC,      !
4                %REF (0), POS, ONE_DESC,         !
4                Q_SIGN, Q_EXP, Q_DESC);
3            END;
3
3        END
2    UNTIL (.ITER_DONE);
2
2 !+
2 ! The reciprocal now lives in Q.  Return it to the caller with the
2 ! original sign of A, which was not used above.
2 !-
2    .CSIGN = .A_SIGN;
2    .CEXP = .Q_EXP;
2
2 !+
2 ! Call CHK_STR_TYPE to determine if we need to pad the number with
2 ! leading zeroes depending on the string type.
2 !-
2
2    QLEN = .Q_DESC[DSC$W_LENGTH];
2    CHK_STR_TYPE (.Q_DESC[DSC$A_POINTER],QLEN,.CDIGITS);
```

```
; 1510        1599  2
; 1511        1600  2
; 1512        1601  2  !     STR$COPY_DX (.CDIGITS, Q_DESC);
; 1513        1602  2  !+
; 1514        1603  2  ! Free our strings.
; 1515        1604  2  !-
; 1516        1605  2        STR$FREE1_DX (X_DESC);
; 1517        1606  2        STR$FREE1_DX (X2_DESC);
; 1518        1607  2        STR$FREE1_DX (Q_DESC);
; 1519        1608  2        STR$FREE1_DX (XA_DESC);
; 1520        1609  2        STR$FREE1_DX (DELTA_DESC);
; 1521        1610  1        END;                                              ! end of STR$RECIP


                                                    00785          .BLKB   3
      00  00  00  50  49  43  45  52  24  52  54  53  00788 P.AAG:   .ASCII  \STR$RECIP\<0><0><0>                               ;
                                      00  00  00  30  00794 P.AAH:   .ASCII  \0\<0><0><0>                                        ;


                                            0FFC 00000          .ENTRY  STR$RECIP, Save R2,R3,R4,R5,R6,R7,R8,R9,-      ; 1200
                                                                        R10,R11                                          ;
                         5B       F97E   CF   9E 00002          MOVAB   STR$ADD, R11                                     ;
                         5A  00000000G   00   9E 00007          MOVAB   STR$FREE1_DX, R10                                ;
                         59  00000000G   00   9E 0000E          MOVAB   LIB$STOP, R9                                     ;
                         58  00000000G   00   9E 00015          MOVAB   STR$GET1_DX, R8                                  ;
                         5E       FF64   CE   9E 0001C          MOVAB   -156(SP), SP                                     ;
                                    64   AE   7C 00021          CLRQ    DELTA_DESC                                       ; 1256
                                    6C   AE   7C 00024          CLRQ    XA_DESC                                          ;
                                    74   AE   7C 00027          CLRQ    Q_DESC                                           ;
                                    7C   AE   7C 0002A          CLRQ    X2_DESC                                          ;
                                    E8   AD   7C 0002D          CLRQ    X_DESC                                           ;
                                    F0   AD   7C 00030          CLRQ    B_DESC                                           ;
                                    F8   AD   7C 00033          CLRQ    A_DESC                                           ;
                         6D       0353   CF   DE 00036          MOVAL   18$, (FP)                                        ;
                         09              6C   91 0003B          CMPB    (AP), #9                                         ; 1329
                                         1E   1E 0003E          BGEQU   1$                                               ;
              54    AE   010E0009        8F   D0 00040          MOVL    #17694729, ROUT_NAME_DESC                        ; 1336
              58    AE          A5       AF   9E 00048          MOVAB   P.AAG, ROUT_NAME_DESC+4                          ; 1339
                                54       AE   9F 0004D          PUSHAB  ROUT_NAME_DESC                                   ; 1340
                         7E              6C   9A 00050          MOVZBL  (AP), -(SP)                                      ;
                                         02   DD 00053          PUSHL   #2                                               ;
                            00000000G    8F   DD 00055          PUSHL   #STR$_WRONUMARG                                  ;
                         69              04   FB 0005B          CALLS   #4, LIB$STOP                                     ;
                                    F8   AD   B4 0005E 1$:      CLRW    A_DESC                                           ; 1346
              FA    AD                   0F   90 00061          MOVB    #15, A_DESC+2                                    ; 1347
              FB    AD                   02   90 00065          MOVB    #2, A_DESC+3                                     ; 1348
                                    FC   AD   D4 00069          CLRL    A_DESC+4                                         ; 1349
                                    08   AE   9F 0006C          PUSHAB  A_BUF                                            ; 1357
                                    18   AE   9F 0006F          PUSHAB  A_LEN                                            ;
                         52         0C   AC   D0 00072          MOVL    ADIGITS, R2                                      ;
                                         52   DD 00076          PUSHL   R2                                               ;
          00000000G      00              03   FB 00078          CALLS   #3, LIB$ANALYZE_SDESC                            ;
                         56              50   D0 0007F          MOVL    R0, STATUS                                       ;
                         01              56   D1 00082          CMPL    STATUS, #1                                       ; 1358
                                         09   13 00085          BEQL    2$                                               ;
```

```
                      00000000G               8F  DD 00087        PUSHL   #LIB$_INVARG                          ; 1360
                               69             01  FB 0008D        CALLS   #1, LIB$STOP                          ;
                                         0C   AE  9F 00090 2$:    PUSHAB  CBUF                                  ; 1367
                                         14   AE  9F 00093        PUSHAB  C_LEN                                 ;
                                         24   AC  DD 00096        PUSHL   CDIGITS                               ;
                      00000000G  00           03  FB 00099        CALLS   #3, LIB$ANALYZE_SDESC                 ;
                                 56           50  D0 000A0        MOVL    R0, STATUS                            ;
                                 01           56  D1 000A3        CMPL    STATUS, #1                            ; 1368
                                              09  13 000A6        BEQL    3$                                    ;
                      00000000G               8F  DD 000A8        PUSHL   #LIB$_INVARG                          ; 1370
                                 69           01  FB 000AE        CALLS   #1, LIB$STOP                          ;
                                         14   AE  D4 000B1 3$:    CLRL    A_LEN                                 ; 1372
                                 57      04   BC  D0 000B4        MOVL    @ASIGN, A_SIGN                        ; 1373
                                              51  D4 000B8        CLRL    SCAN_DONE                             ; 1379
14   AE           62                     10   00  ED 000BA 4$:    CMPZV   #0, #16, (R2), A_LEN                  ; 1384
                                              15  13 000C0        BEQL    5$                                    ;
                  50         08  AE      14   AE  C1 000C2        ADDL3   A_LEN, A_BUF, R0                      ; 1389
                                 30           60  91 000C8        CMPB    (R0), #48                             ;
                                              0A  1F 000CB        BLSSU   5$                                    ;
                                 39           60  91 000CD        CMPB    (R0), #57                             ;
                                              05  1A 000D0        BGTRU   5$                                    ;
                                         14   AE  D6 000D2        INCL    A_LEN                                 ; 1391
                                              03  11 000D5        BRB     6$                                    ;
                                 51           01  D0 000D7 5$:    MOVL    #1, SCAN_DONE                         ; 1393
                                 DD           51  E9 000DA 6$:    BLBC    SCAN_DONE, 4$                         ; 1396
                                         F8   AD  9F 000DD        PUSHAB  A_DESC                                ; 1399
                                         18   AE  9F 000E0        PUSHAB  A_LEN                                 ;
                                 68           02  FB 000E3        CALLS   #2, STR$GET1_DX                       ;
                             08  AE      FC   AD  D0 000E6        MOVL    A_DESC+4, A_BUF                       ; 1400
              08  BE         04  B2      14   AE  28 000EB        MOVC3   A_LEN, @4(R2), @A_BUF                 ; 1401
14   AE           30       FF05  CF           01  2D 000F2        CMPC5   #1, P.AAH, #48, A_LEN, @A_BUF         ; 1406
                                         08   BE     000FA                                                      ;
                                              09  12 000FC        BNEQ    7$                                    ;
                      00000000G               8F  DD 000FE        PUSHL   #STR$_DIVBY_ZER                       ;
                                 69           01  FB 00104        CALLS   #1, LIB$STOP                          ;
                                         F0   AD  B4 00107 7$:    CLRW    B_DESC                                ; 1408
                             F2  AD           0F  90 0010A        MOVB    #15, B_DESC+2                         ; 1409
                             F3  AD           02  90 0010E        MOVB    #2, B_DESC+3                          ; 1410
                                         F4   AD  D4 00112        CLRL    B_DESC+4                              ; 1411
                                         18   AE  9F 00115        PUSHAB  B_BUF                                 ; 1419
                                         20   AE  9F 00118        PUSHAB  B_LEN                                 ;
                                 52      18   AC  D0 0011B        MOVL    BDIGITS, R2                           ;
                                              52  DD 0011F        PUSHL   R2                                    ;
                      00000000G  00           03  FB 00121        CALLS   #3, LIB$ANALYZE_SDESC                 ;
                                 56           50  D0 00128        MOVL    R0, STATUS                            ;
                                 01           56  D1 0012B        CMPL    STATUS, #1                            ; 1420
                                              09  13 0012E        BEQL    8$                                    ;
                      00000000G               8F  DD 00130        PUSHL   #LIB$_INVARG                          ; 1422
                                 69           01  FB 00136        CALLS   #1, LIB$STOP                          ;
                                         1C   AE  D4 00139 8$:    CLRL    B_LEN                                 ; 1423
                                 50      10   BC  D0 0013C        MOVL    @BSIGN, B_SIGN                        ; 1424
                                              51  D4 00140        CLRL    SCAN_DONE                             ; 1430
1C   AE           62                     10   00  ED 00142 9$:    CMPZV   #0, #16, (R2), B_LEN                  ; 1435
                                              15  13 00148        BEQL    10$                                   ;
                  50         18  AE      1C   AE  C1 0014A        ADDL3   B_LEN, B_BUF, R0                      ; 1440
                                 30           60  91 00150        CMPB    (R0), #48                             ;
                                              0A  1F 00153        BLSSU   10$                                   ;
```

```
                   39         60  91 00155         CMPB    (R0), #57                          ;
                              05  1A 00158         BGTRU   10$                                ;
                         1C   AE  D6 0015A         INCL    B_LEN                              ; 1442
                              03  11 0015D         BRB     11$                                ;
                   51         01  D0 0015F  10$:   MOVL    #1, SCAN_DONE                      ; 1444
                   DD         51  E9 00162  11$:   BLBC    SCAN_DONE, 9$                      ; 1447
                         F0   AD  9F 00165         PUSHAB  B_DESC                             ; 1450
                         20   AE  9F 00168         PUSHAB  B_LEN                              ;
                   68         02  FB 0016B         CALLS   #2, STR$GET1_DX                    ;
                   18    AE   F4  AD  D0 0016E     MOVL    B_DESC+4, B_BUF                    ; 1451
  18    BE         04    B2   1C  AE  28 00173     MOVC3   B_LEN, @4(R2), @B_BUF              ; 1452
                         E8   AD  B4 0017A         CLRW    X_DESC                             ; 1456
                   EA    AD   0F  90 0017D         MOVB    #15, X_DESC+2                      ; 1457
                   EB    AD   02  90 00181         MOVB    #2, X_DESC+3                       ; 1458
                         EC   AD  D4 00185         CLRL    X_DESC+4                           ; 1459
                         E8   AD  9F 00188         PUSHAB  X_DESC                             ; 1460
                   08    AE   01  D0 0018B         MOVL    #1, 8(SP)                          ;
                         08   AE  9F 0018F         PUSHAB  8(SP)                              ;
                   68         02  FB 00192         CALLS   #2, STR$GET1_DX                    ;
                   50    EC   AD  D0 00195         MOVL    X_DESC+4, X_BUF                    ; 1461
                   60         31  90 00199         MOVB    #49, (X_BUF)                       ; 1462
                         24   AE  7C 0019C         CLRQ    X_EXP                              ; 1464
                         7C   AE  B4 0019F         CLRW    X2_DESC                            ; 1466
                   7E    AE   0F  90 001A2         MOVB    #15, X2_DESC+2                     ; 1467
                   7F    AE   02  90 001A6         MOVB    #2, X2_DESC+3                      ; 1468
                         E4   AD  D4 001AA         CLRL    X2_DESC+4                          ; 1469
                         7C   AE  9F 001AD         PUSHAB  X2_DESC                            ; 1470
                   08    AE   01  D0 001B0         MOVL    #1, 8(SP)                          ;
                         08   AE  9F 001B4         PUSHAB  8(SP)                              ;
                   68         02  FB 001B7         CALLS   #2, STR$GET1_DX                    ;
                   50    E4   AD  D0 001BA         MOVL    X2_DESC+4, X2_BUF                  ; 1471
                   60         30  90 001BE         MOVB    #48, (X2_BUF)                      ; 1472
                         34   AE  7C 001C1         CLRQ    X2_EXP                             ; 1474
                         74   AE  B4 001C4         CLRW    Q_DESC                             ; 1476
                   76    AE   0F  90 001C7         MOVB    #15, Q_DESC+2                      ; 1477
                   77    AE   02  90 001CB         MOVB    #2, Q_DESC+3                       ; 1478
                         78   AE  D4 001CF         CLRL    Q_DESC+4                           ; 1479
                         74   AE  9F 001D2         PUSHAB  Q_DESC                             ; 1480
                   08    AE   01  D0 001D5         MOVL    #1, 8(SP)                          ;
                         08   AE  9F 001D9         PUSHAB  8(SP)                              ;
                   68         02  FB 001DC         CALLS   #2, STR$GET1_DX                    ;
                   50    78   AE  D0 001DF         MOVL    Q_DESC+4, Q_BUF                    ; 1481
                   60         30  90 001E3         MOVB    #48, (Q_BUF)                       ; 1482
                         48   AE  7C 001E6         CLRQ    Q_EXP                              ; 1484
                         6C   AE  B4 001E9         CLRW    XA_DESC                            ; 1486
                   6E    AE   0F  90 001EC         MOVB    #15, XA_DESC+2                     ; 1487
                   6F    AE   02  90 001F0         MOVB    #2, XA_DESC+3                      ; 1488
                         70   AE  D4 001F4         CLRL    XA_DESC+4                          ; 1489
                         6C   AE  9F 001F7         PUSHAB  XA_DESC                            ; 1490
                   08    AE   01  D0 001FA         MOVL    #1, 8(SP)                          ;
                         08   AE  9F 001FE         PUSHAB  8(SP)                              ;
                   68         02  FB 00201         CALLS   #2, STR$GET1_DX                    ;
                   50    70   AE  D0 00204         MOVL    XA_DESC+4, XA_BUF                  ; 1491
                   60         30  90 00208         MOVB    #48, (XA_BUF)                      ; 1492
                         2C   AE  7C 0020B         CLRQ    XA_EXP                             ; 1494
                         64   AE  B4 0020E         CLRW    DELTA_DESC                         ; 1496
                   66    AE   0F  90 00211         MOVB    #15, DELTA_DESC+2                  ; 1497
```

```
                67    AE                02  90 00215           MOVB    #2, DELTA_DESC+3                        ; 1498
                                  68    AE  D4 00219           CLRL    DELTA_DESC+4                            ; 1499
                                  64    AE  9F 0021C           PUSHAB  DELTA_DESC                              ; 1500
                08    AE                01  D0 0021F           MOVL    #1, 8(SP)                               ;
                                  08    AE  9F 00223           PUSHAB  8(SP)                                   ;
                      68                02  FB 00226           CALLS   #2, STR$GET1_DX                         ;
                      50          68    AE  D0 00229           MOVL    DELTA_DESC+4, DELTA_BUF                 ; 1501
                      60                30  90 0022D           MOVB    #48, (DELTA_BUF)                        ; 1502
                                  3C    AE  7C 00230           CLRQ    DELTA_EXP                               ; 1504
                5C    AE    010F0001    8F  D0 00233           MOVL    #17760257, ONE_DESC                     ; 1506
                60    AE          20    AE  9E 0023B           MOVAB   ONE_BUF, ONE_DESC+4                     ; 1509
                20    AE                31  90 00240           MOVB    #49, ONE_BUF                            ; 1510
                44    AE          24    AE  CE 00244           MNEGL   X_EXP, POS                              ; 1516
                                        53  D4 00249           CLRL    ITER_DONE                               ; 1521
                                  E8    AD  9F 0024B 12$:      PUSHAB  X_DESC                                  ; 1525
                                  28    AE  9F 0024E           PUSHAB  X_EXP                                   ;
                                  30    AE  9F 00251           PUSHAB  X_SIGN                                  ;
                                  F8    AD  9F 00254           PUSHAB  A_DESC                                  ;
          52    08    BC          54    AE  C1 00257           ADDL3   POS, @AEXP, R2                          ; 1526
                14    AE                52  D0 0025D           MOVL    R2, 20(SP)                              ;
                                  14    AE  9F 00261           PUSHAB  20(SP)                                  ;
                14    AE                01  D0 00264           MOVL    #1, 20(SP)                              ;
                                  14    AE  9F 00268           PUSHAB  20(SP)                                  ;
                                  E8    AD  9F 0026B           PUSHAB  X_DESC                                  ; 1525
                                  40    AE  9F 0026E           PUSHAB  X_EXP                                   ;
                                  48    AE  9F 00271           PUSHAB  X_SIGN                                  ;
                      6B                09  FB 00274           CALLS   #9, STR$ADD                             ;
                      03          28    AE  E8 00277           BLBS    X_SIGN, 13$                             ; 1532
                                      00AD  31 0027B           BRW     15$                                     ;
                                  E8    AD  9F 0027E 13$:      PUSHAB  X_DESC                                  ; 1535
                                  28    AE  9F 00281           PUSHAB  X_EXP                                   ;
                                  30    AE  9F 00284           PUSHAB  X_SIGN                                  ;
                                  F8    AD  9F 00287           PUSHAB  A_DESC                                  ;
                14    AE                52  D0 0028A           MOVL    R2, 20(SP)                              ; 1536
                                  14    AE  9F 0028E           PUSHAB  20(SP)                                  ;
                                  14    AE  D4 00291           CLRL    20(SP)                                  ;
                                  14    AE  9F 00294           PUSHAB  20(SP)                                  ;
                                  E8    AD  9F 00297           PUSHAB  X_DESC                                  ; 1535
                                  40    AE  9F 0029A           PUSHAB  X_EXP                                   ;
                                  48    AE  9F 0029D           PUSHAB  X_SIGN                                  ;
                      6B                09  FB 002A0           CALLS   #9, STR$ADD                             ;
                                  44    AE  D7 002A3           DECL    POS                                     ; 1541
                                  6C    AE  9F 002A6           PUSHAB  XA_DESC                                 ; 1545
                                  30    AE  9F 002A9           PUSHAB  XA_EXP                                  ;
                                  38    AE  9F 002AC           PUSHAB  XA_SIGN                                 ;
                                  F8    AD  9F 002AF           PUSHAB  A_DESC                                  ;
                                  08    AC  DD 002B2           PUSHL   AEXP                                    ; 1546
                                  18    AE  D4 002B5           CLRL    24(SP)                                  ;
                                  18    AE  9F 002B8           PUSHAB  24(SP)                                  ;
                                  D8    AD  9F 002BB           PUSHAB  Q_DESC                                  ; 1545
                                  64    AE  9F 002BE           PUSHAB  Q_EXP                                   ;
                                  6C    AE  9F 002C1           PUSHAB  Q_SIGN                                  ;
              0440    CB                09  FB 002C4           CALLS   #9, STR$MUL                             ;
                                  64    AE  9F 002C9           PUSHAB  DELTA_DESC                              ; 1548
                                  40    AE  9F 002CC           PUSHAB  DELTA_EXP                               ;
                                  48    AE  9F 002CF           PUSHAB  DELTA_SIGN                              ;
                                  68    AE  9F 002D2           PUSHAB  ONE_DESC                                ;
```

```
                 14    AE   D4 002D5           CLRL    20(SP)                          ; 1549
                 14    AE   9F 002D8           PUSHAB  20(SP)                          ;
   14     AE           01   D0 002DB           MOVL    #1, 20(SP)                      ;
                 14    AE   9F 002DF           PUSHAB  20(SP)                          ;
                 D0    AD   9F 002E2           PUSHAB  XA_DESC                         ; 1548
                 48    AE   9F 002E5           PUSHAB  XA_EXP                          ;
                 50    AE   9F 002E8           PUSHAB  XA_SIGN                         ;
          6B           09   FB 002EB           CALLS   #9, STR$ADD                     ;
                 40    AE   D4 002EE           CLRL    DELTA_SIGN                      ; 1551
                 7C    AE   9F 002F1           PUSHAB  X2_DESC                         ; 1552
                 38    AE   9F 002F4           PUSHAB  X2_EXP                          ;
                 40    AE   9F 002F7           PUSHAB  X2_SIGN                         ;
                 F0    AD   9F 002FA           PUSHAB  B_DESC                          ;
                 14    AC   DD 002FD           PUSHL   BEXP                            ; 1553
   18     AE           01   D0 00300           MOVL    #1, 24(SP)                      ;
                 18    AE   9F 00304           PUSHAB  24(SP)                          ;
                 7C    AE   9F 00307           PUSHAB  DELTA_DESC                      ; 1552
                 58    AE   9F 0030A           PUSHAB  DELTA_EXP                       ;
                 60    AE   9F 0030D           PUSHAB  DELTA_SIGN                      ;
          6B           09   FB 00310           CALLS   #9, STR$ADD                     ;
          0F     38    AE   E8 00313           BLBS    X2_SIGN, 14$                    ; 1556
          01     64    AE   B1 00317           CMPW    DELTA_DESC, #1                  ; 1561
                       2F   12 0031B           BNEQ    16$                             ;
          50     68    AE   D0 0031D           MOVL    DELTA_DESC+4, DELTA_BUF         ; 1568
          30           60   91 00321           CMPB    (DELTA_BUF), #48                ; 1570
                       26   12 00324           BNEQ    16$                             ;
          53           01   D0 00326 14$:      MOVL    #1, ITER_DONE                   ;
                       21   11 00329           BRB     16$                             ; 1532
                 74    AE   9F 0032B 15$:      PUSHAB  Q_DESC                          ; 1577
                 4C    AE   9F 0032E           PUSHAB  Q_EXP                           ;
                 54    AE   9F 00331           PUSHAB  Q_SIGN                          ;
                 68    AE   9F 00334           PUSHAB  ONE_DESC                        ;
                 54    AE   9F 00337           PUSHAB  POS                             ;
                 18    AE   D4 0033A           CLRL    24(SP)                          ; 1578
                 18    AE   9F 0033D           PUSHAB  24(SP)                          ;
                 D8    AD   9F 00340           PUSHAB  Q_DESC                          ; 1577
                 64    AE   9F 00343           PUSHAB  Q_EXP                           ;
                 6C    AE   9F 00346           PUSHAB  Q_SIGN                          ;
          6B           09   FB 00349           CALLS   #9, STR$ADD                     ;
          03           53   E8 0034C 16$:      BLBS    ITER_DONE, 17$                  ; 1583
                     FEF9   31 0034F           BRW     12$                             ;
   1C     BC           57   D0 00352 17$:      MOVL    A_SIGN, @CSIGN                  ; 1589
   20     BC     48    AE   D0 00356           MOVL    Q_EXP, @CEXP                    ; 1590
   50     AE     74    AE   3C 0035B           MOVZWL  Q_DESC, QLEN                    ; 1597
                 24    AC   DD 00360           PUSHL   CDIGITS                         ; 1598
                 54    AE   9F 00363           PUSHAB  QLEN                            ;
                 DC    AD   DD 00366           PUSHL   Q_DESC+4                        ;
0000V     CF           03   FB 00369           CALLS   #3, CHK_STR_TYPE                ;
                 E8    AD   9F 0036E           PUSHAB  X_DESC                          ; 1605
          6A           01   FB 00371           CALLS   #1, STR$FREE1_DX                ;
                 7C    AE   9F 00374           PUSHAB  X2_DESC                         ; 1606
          6A           01   FB 00377           CALLS   #1, STR$FREE1_DX                ;
                 74    AE   9F 0037A           PUSHAB  Q_DESC                          ; 1607
          6A           01   FB 0037D           CALLS   #1, STR$FREE1_DX                ;
                 6C    AE   9F 00380           PUSHAB  XA_DESC                         ; 1608
          6A           01   FB 00383           CALLS   #1, STR$FREE1_DX                ;
                 64    AE   9F 00386           PUSHAB  DELTA_DESC                      ; 1609
```

```
                   6A           01  FB 00389          CALLS   #1, STR$FREE1_DX                   ;
                                    04 0038C          RET                                        ; 1610
                                  0000 0038D 18$:     .WORD   Save nothing                       ; 1256
                   50           08  AC  D0 0038F      MOVL    8(AP), R0                          ;
                   50           04  A0  D0 00393      MOVL    4(R0), R0                          ;
                                C8  A0  9F 00397      PUSHAB  DELTA_DESC                         ;
                                D0  A0  9F 0039A      PUSHAB  XA_DESC                            ;
                                D8  A0  9F 0039D      PUSHAB  Q_DESC                             ;
                                E0  A0  9F 003A0      PUSHAB  X2_DESC                            ;
                                E8  A0  9F 003A3      PUSHAB  X_DESC                             ;
                                F0  A0  9F 003A6      PUSHAB  B_DESC                             ;
                                F8  A0  9F 003A9      PUSHAB  A_DESC                             ;
                                    07  DD 003AC      PUSHL   #7                                 ;
                                    5E  DD 003AE      PUSHL   SP                                 ;
                   7E           04  AC  7D 003B0      MOVQ    4(AP), -(SP)                       ;
             0000V CF               03  FB 003B4      CALLS   #3, FREE_STRINGS                   ;
                                    04 003B9          RET                                        ;

; Routine Size:  954 bytes,    Routine Base:  _STR$CODE + 0798

; 1522          1611  1
```

```
1 GLOBAL ROUTINE STR$ROUND (                              ! Round a number
1         PLACES,                                         ! Max decimal places in the result
1         TRUNC,                                          ! Truncate to that many places
1         ASIGN,                                          ! Sign of operand A
1         AEXP,                                           ! Decimal exponent of operand A
1         ADIGITS,                                        ! Digits of operand A
1         BSIGN,                                          ! Sign of operand B
1         BEXP,                                           ! Decimal exponent of operand B
1         BDIGITS                                         ! Digits of operand B
1     ) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Round or truncate a number to a specified number of significant
1 !       digits.  B := ROUND (A)
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       PLACES.rl.r     Max decimal digits to retain in the result
1 !       TRUNC.rv.r      0 = round, 1 = truncate to that many places
1 !       ASIGN.rv.r      0 = operand A is positive, 1 = negative
1 !       AEXP.rl.r       Power of 10 by which to multiply the operand A
1 !                       digits to get the absolute value of operand A.
1 !                       E.g., AEXP = 1, ADIGITS = 123 gives 1230.
1 !       ADIGITS.rnu.d   Descriptor for the digits of operand A
1 !       BSIGN.wl.r      0 = operand B is positive, 1 = negative
1 !       BEXP.wl.r       Power of 10 by which to multiply the operand B
1 !                       digits to get the absolute value of operand B.
1 !                       E.g., BEXP = -1, BDIGITS = 123 gives 12.3.
1 !       BDIGITS.wnu.d   Descriptor for the digits of operand B
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       May allocate space for the BDIGITS string.
1 !       Signals if memory is exhausted.
1 !
1 !--
1
2     BEGIN
2
2     MAP
2         ADIGITS : REF BLOCK [8, BYTE],
2         BDIGITS : REF BLOCK [8, BYTE];
```

```

    LOCAL
!+
! Internal form of A.
!-
        ABUF : REF VECTOR [65535, BYTE],
        A_SIGN,
!+
! Internal form of B.
!-
        REXP,
        R_DESC : BLOCK [8, BYTE] VOLATILE,
        RBUF : REF VECTOR [65535, BYTE],
        R_LEN,                                  ! Length of the result
        RESULT_DIGITS,                          ! Number of digits in the result

!+
! The following locals are needed for calls to LIB$ANALYZE_SDESC.
!-
        A_LEN,
        B_LEN,
        BBUF,
        STATUS;

    BUILTIN
        ACTUALCOUNT;

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

    ENABLE
        FREE_STRINGS (R_DESC);

!+
! Check for the proper number of arguments.
!-

    IF (ACTUALCOUNT () LSS 8)
    THEN
        BEGIN

        LOCAL
            ROUT_NAME_DESC : BLOCK [8, BYTE];

        ROUT_NAME_DESC [DSC$W_LENGTH] = 9;
        ROUT_NAME_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
        ROUT_NAME_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
        ROUT_NAME_DESC [DSC$A_POINTER] = UPLIT (%ASCII'STR$ROUND');
        LIB$STOP (STR$_WRONUMARG, 2, ACTUALCOUNT (), ROUT_NAME_DESC);
        END;

!+
! Copy the given number to local storage before we begin work on it.
!-
    A_SIGN = ..ASIGN;
    REXP = ..AEXP;
```

```
2       R_DESC [DSC$W_LENGTH] = 0;
2       R_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
2       R_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
2       R_DESC [DSC$A_POINTER] = 0;
2   !+
2   ! Compute the length of operand A.  Only the leading digits count.
2   ! First call LIB$ANALYZE_SDESC to ensure that the input descriptor
2   ! is valid.  If it is, then ABUF will contain the address of the
2   ! first byte of the string, and A_LEN will contain its length.
2   !-
2
2       STATUS = LIB$ANALYZE_SDESC (.ADIGITS,A_LEN,ABUF);
2       IF .STATUS NEQ SS$_NORMAL
2         THEN
2           LIB$STOP (LIB$_INVARG);
2
2   !+
2   ! Also check the BDIGITS descriptor before getting too involved
2   ! in this routine.
2   !-
2
2       STATUS = LIB$ANALYZE_SDESC (.BDIGITS,B_LEN,BBUF);
2       IF .STATUS NEQ SS$_NORMAL
2         THEN
2           LIB$STOP (LIB$_INVARG);
2
2       R_LEN = 0;
3       BEGIN
3
3       LOCAL
3           SCAN_DONE;
3
3       SCAN_DONE = 0;
3
3       DO
4           BEGIN
4
5           IF (.R_LEN EQLU .ADIGITS [DSC$W_LENGTH])
4           THEN
4               SCAN_DONE = 1
4           ELSE
4
5               IF ((.ABUF [.R_LEN] GEQ %C'0') AND (.ABUF [.R_LEN] LEQ %C'9'))
4               THEN
4                   R_LEN = .R_LEN + 1
4               ELSE
4                   SCAN_DONE = 1;
4
4           END
3       UNTIL (.SCAN_DONE);
3
2       END;
2       STR$GET1_DX (R_LEN, R_DESC);
2       RBUF = .R_DESC [DSC$A_POINTER];
2       CH$MOVE (.R_LEN, .ADIGITS [DSC$A_POINTER], RBUF [0]);
2   !+
2   ! Round or truncate the number if it has more than the desired number
```

```
2 ! of significant digits.
2 !-
2     RESULT_DIGITS = .R_LEN;
2
3     IF (.RESULT_DIGITS GTR ..PLACES)
2     THEN
3          BEGIN
3
4          IF ( NOT ..TRUNC)
3          THEN
4               BEGIN
4 !+
4 ! Check the highest-order digit we will discard.  If it is five or
4 ! larger, round up.  Note that the number is in sign-magnitude form
4 ! at this point, so rounding "up" is always away from zero.
4 !-
4
5               IF (.RBUF [..PLACES] GEQ %C'5')
4               THEN
5                    BEGIN
5
5                    LOCAL
5                         CARRY_COUNTER,
5                         CARRY_DONE;
5
5                    CARRY_DONE = 0;
5                    CARRY_COUNTER = ..PLACES - 1;
5
6                    IF (.CARRY_COUNTER GEQ 0)
5                    THEN
5
5                         DO
6                              BEGIN
6                              RBUF [.CARRY_COUNTER] = .RBUF [.CARRY_COUNTER] + 1;
6
7                              IF (.RBUF [.CARRY_COUNTER] LEQ %C'9')
6                              THEN
6                                   CARRY_DONE = 1
6                              ELSE
7                                   BEGIN
7                                   RBUF [.CARRY_COUNTER] = .RBUF [.CARRY_COUNTER] - 10;
7                                   CARRY_COUNTER = .CARRY_COUNTER - 1;
6                                   END;
6
6                              END
5                         UNTIL ((.CARRY_DONE) OR (.CARRY_COUNTER LSS 0));
5
6                    IF ( NOT .CARRY_DONE)
5                    THEN
6                         BEGIN
6 !+
6 ! The carry has forced a right shift (all 9's rounded up).
6 ! We are guaranteed enough space since we must be dropping at least
6 ! one digit.  Because of this shift we must now be dropping at least
6 ! two digits.
6 !-
6
```

```
6                        INCR COUNTER FROM 0 TO ..PLACES - 2 DO
6                            RBUF [.COUNTER + 1] = .RBUF [.COUNTER];
6
6                        RBUF [0] = %C'1';
6                        REXP = .REXP + 1;
5                        END;
5
4                   END;
4
3               END;
3
3       REXP = .REXP + (.RESULT_DIGITS - ..PLACES);
3       RESULT_DIGITS = ..PLACES;
3       END;
2
2  !+
2  ! Return the results to the caller in the B operand.
2  ! If there are no digits left, return a single zero digit.
2  !-
2
3    IF (.RESULT_DIGITS EQL 0)
2    THEN
3       BEGIN
3       .BSIGN = 0;
3       .BEXP = 0;
3       STR$COPY_R (.BDIGITS, %REF (1), %REF (%ASCII'0'));
3       CHK_STR_TYPE (.BDIGITS[DSC$A_POINTER],%REF (1),.BDIGITS);
3       END
3
2    ELSE
2
2  !+
2  ! Call CHK_STR_TYPE to determine if we need to pad the number with
2  ! leading zeroes depending on the string type.
2  !-
2
3     BEGIN
3     .BSIGN = .A_SIGN;
3     .BEXP = .REXP;
3     CHK_STR_TYPE (.R_DESC[DSC$A_POINTER],RESULT_DIGITS,.BDIGITS);
2     END;
2
2  !    BEGIN
2  !    .BSIGN = .A_SIGN;
2  !    .BEXP = .REXP;
2  !    STR$COPY_R (.BDIGITS, RESULT_DIGITS, .R_DESC [DSC$A_POINTER]);
2  !    END;
2
2  !+
2  ! Free our string.
2  !-
2    STR$FREE1_DX (R_DESC);
2    RETURN;
1    END;                                       ! end of STR$ROUND
```

```
                                               00B52          .BLKB    2
  00  00  00  44  4E  55  4F  52  24  52  54  53  00B54 P.AAI:  .ASCII   \STR$ROUND\<0><0><0>                   ;


                                          0FFC 00000          .ENTRY   STR$ROUND, Save R2,R3,R4,R5,R6,R7,R8,R9,-  ; 1612
                                                                       R10,R11                                   ;
                     5B 00000000G   00   9E 00002          MOVAB    LIB$ANALYZE_SDESC, R11                    ;
                     5A 00000000G   00   9E 00009          MOVAB    LIB$STOP, R10                             ;
                     5E             30   C2 00010          SUBL2    #48, SP                                   ;
                              28    AE   7C 00013          CLRQ     R_DESC                                    ; 1664
                     6D     016C    CF   DE 00016          MOVAL    17$, (FP)                                 ;
                     08             6C   91 0001B          CMPB     (AP), #8                                  ; 1707
                                    1E   1E 0001E          BGEQU    1$                                        ;
                20   AE 010E0009    8F   D0 00020          MOVL     #17694729, ROUT_NAME_DESC                 ; 1714
                24   AE       C9    AF   9E 00028          MOVAB    P.AAI, ROUT_NAME_DESC+4                   ; 1717
                              20    AE   9F 0002D          PUSHAB   ROUT_NAME_DESC                            ; 1718
                     7E             6C   9A 00030          MOVZBL   (AP), -(SP)                               ;
                                    02   DD 00033          PUSHL    #2                                        ;
                        00000000G   8F   DD 00035          PUSHL    #STR$_WRONUMARG                           ;
                     6A             04   FB 0003B          CALLS    #4, LIB$STOP                              ;
                     59       0C    BC   D0 0003E 1$:      MOVL     @ASIGN, A_SIGN                            ; 1724
                     58       10    BC   D0 00042          MOVL     @AEXP, REXP                               ; 1725
                              28    AE   B4 00046          CLRW     R_DESC                                    ; 1726
                2A   AE             0F   90 00049          MOVB     #15, R_DESC+2                             ; 1727
                2B   AE             02   90 0004D          MOVB     #2, R_DESC+3                              ; 1728
                              2C    AE   D4 00051          CLRL     R_DESC+4                                  ; 1729
                              08    AE   9F 00054          PUSHAB   ABUF                                      ; 1737
                              10    AE   9F 00057          PUSHAB   A_LEN                                     ;
                     52       14    AC   D0 0005A          MOVL     ADIGITS, R2                               ;
                                    52   DD 0005E          PUSHL    R2                                        ;
                     6B             03   FB 00060          CALLS    #3, LIB$ANALYZE_SDESC                     ;
                     53             50   D0 00063          MOVL     R0, STATUS                                ;
                     01             53   D1 00066          CMPL     STATUS, #1                                ; 1738
                                    09   13 00069          BEQL     2$                                        ;
                        00000000G   8F   DD 0006B          PUSHL    #LIB$_INVARG                              ; 1740
                     6A             01   FB 00071          CALLS    #1, LIB$STOP                              ;
                              10    AE   9F 00074 2$:      PUSHAB   BBUF                                      ; 1747
                              18    AE   9F 00077          PUSHAB   B_LEN                                     ;
                     57       20    AC   D0 0007A          MOVL     BDIGITS, R7                               ;
                                    57   DD 0007E          PUSHL    R7                                        ;
                     6B             03   FB 00080          CALLS    #3, LIB$ANALYZE_SDESC                     ;
                     53             50   D0 00083          MOVL     R0, STATUS                                ;
                     01             53   D1 00086          CMPL     STATUS, #1                                ; 1748
                                    09   13 00089          BEQL     3$                                        ;
                        00000000G   8F   DD 0008B          PUSHL    #LIB$_INVARG                              ; 1750
                     6A             01   FB 00091          CALLS    #1, LIB$STOP                              ;
                              18    AE   D4 00094 3$:      CLRL     R_LEN                                     ; 1752
                                    51   D4 00097          CLRL     SCAN_DONE                                 ; 1758
    18   AE         62        10    00   ED 00099 4$:      CMPZV    #0, #16, (R2), R_LEN                      ; 1763
                                    15   13 0009F          BEQL     5$                                        ;
               50        08  AE   18    AE   C1 000A1          ADDL3    R_LEN, ABUF, R0                           ; 1768
                             30     60   91 000A7          CMPB     (R0), #48                                 ;
                                    0A   1F 000AA          BLSSU    5$                                        ;
                     39             60   91 000AC          CMPB     (R0), #57                                 ;
                                    05   1A 000AF          BGTRU    5$                                        ;
                              18    AE   D6 000B1          INCL     R_LEN                                     ; 1770
```

```
                          03 11 000B4        BRB     6$                                       ;
                51        01 D0 000B6 5$:    MOVL    #1, SCAN_DONE                            ; 1772
                DD        51 E9 000B9 6$:    BLBC    SCAN_DONE, 4$                            ; 1775
                     28   AE 9F 000BC        PUSHAB  R_DESC                                   ; 1778
                     1C   AE 9F 000BF        PUSHAB  R_LEN                                    ;
    00000000G   00        02 FB 000C2        CALLS   #2, STR$GET1_DX                          ;
                56   2C   AE D0 000C9        MOVL    R_DESC+4, RBUF                           ; 1779
66         04   B2   18   AE 28 000CD        MOVC3   R_LEN, @4(R2), (RBUF)                    ; 1780
           1C   AE   18   AE D0 000D3        MOVL    R_LEN, RESULT_DIGITS                     ; 1785
                51   04   BC D0 000D8        MOVL    @PLACES, R1                              ; 1787
                51   1C   AE D1 000DC        CMPL    RESULT_DIGITS, R1                        ;
                          54 15 000E0        BLEQ    14$                                      ;
                44   08   BC E8 000E2        BLBS    @TRUNC, 13$                              ; 1791
                35      6146 91 000E6        CMPB    (R1)[RBUF], #53                          ; 1800
                          3E 1F 000EA        BLSSU   13$                                      ;
                          52 D4 000EC        CLRL    CARRY_DONE                               ; 1808
                50   FF   A1 9E 000EE        MOVAB   -1(R1), CARRY_COUNTER                    ; 1809
                          1B 19 000F2        BLSS    10$                                      ; 1811
                        6046 96 000F4 7$:    INCB    (CARRY_COUNTER)[RBUF]                    ; 1816
                39      6046 91 000F7        CMPB    (CARRY_COUNTER)[RBUF], #57               ; 1818
                          05 1A 000FB        BGTRU   8$                                       ;
                52        01 D0 000FD        MOVL    #1, CARRY_DONE                           ; 1820
                          06 11 00100        BRB     9$                                       ;
              6046        0A 82 00102 8$:    SUBB2   #10, (CARRY_COUNTER)[RBUF]               ; 1823
                          50 D7 00106        DECL    CARRY_COUNTER                            ; 1824
                1F        52 E8 00108 9$:    BLBS    CARRY_DONE, 13$                          ; 1828
                          50 D5 0010B        TSTL    CARRY_COUNTER                            ;
                          E5 18 0010D        BGEQ    7$                                       ;
                18        52 E8 0010F 10$:   BLBS    CARRY_DONE, 13$                          ; 1830
                50   FE   A1 9E 00112        MOVAB   -2(R1), R0                               ; 1840
                52        01 CE 00116        MNEGL   #1, COUNTER                              ;
                          06 11 00119        BRB     12$                                      ;
           01 A246      6246 90 0011B 11$:   MOVB    (COUNTER)[RBUF], 1(COUNTER)[RBUF]        ; 1841
F6              52        50 F3 00121 12$:   AOBLEQ  R0, COUNTER, 11$                         ;
                66        31 90 00125        MOVB    #49, (RBUF)                              ; 1843
                          58 D6 00128        INCL    REXP                                     ; 1844
50         1C   AE        51 C3 0012A 13$:   SUBL3   R1, RESULT_DIGITS, R0                    ; 1851
                58        50 C0 0012F        ADDL2   R0, REXP                                 ;
           1C   AE        51 D0 00132        MOVL    R1, RESULT_DIGITS                        ; 1852
                     1C   AE D5 00136 14$:   TSTL    RESULT_DIGITS                            ; 1860
                          2B 12 00139        BNEQ    15$                                      ;
                     18   BC D4 0013B        CLRL    @BSIGN                                   ; 1863
                     1C   BC D4 0013E        CLRL    @BEXP                                    ; 1864
           04   AE        30 D0 00141        MOVL    #48, 4(SP)                               ; 1865
                     04   AE 9F 00145        PUSHAB  4(SP)                                    ;
           04   AE        01 D0 00148        MOVL    #1, 4(SP)                                ;
                     04   AE 9F 0014C        PUSHAB  4(SP)                                    ;
                          57 DD 0014F        PUSHL   R7                                       ;
    00000000G   00        03 FB 00151        CALLS   #3, STR$COPY_R                           ;
                          57 DD 00158        PUSHL   R7                                       ; 1866
           08   AE        01 D0 0015A        MOVL    #1, 8(SP)                                ;
                     08   AE 9F 0015E        PUSHAB  8(SP)                                    ;
                     04   A7 DD 00161        PUSHL   4(R7)                                    ;
                          10 11 00164        BRB     16$                                      ;
           18   BC        59 D0 00166 15$:   MOVL    A_SIGN, @BSIGN                           ; 1877
           1C   BC        58 D0 0016A        MOVL    REXP, @BEXP                              ; 1878
                          57 DD 0016E        PUSHL   R7                                       ; 1879
                          20   AE   9F 00170          PUSHAB  RESULT_DIGITS                     ;
                          34   AE   DD 00173          PUSHL   R_DESC+4                          ;
          0000V  CF            03   FB 00176 16$:     CALLS   #3, CHK_STR_TYPE                  ;
                          28   AE   9F 0017B          PUSHAB  R_DESC                            ; 1891
      00000000G  00            01   FB 0017E          CALLS   #T, STR$FREE1_DX                  ;
                                    04 00185          RET                                       ; 1893
                                  0000 00186 17$:     .WORD   Save nothing                      ; 1664
                 50       08   AC   D0 00188          MOVL    8(AP), R0                         ;
                 50       04   A0   D0 0018C          MOVL    4(R0), R0                         ;
                          F8   A0   9F 00190          PUSHAB  R_DESC                            ;
                               01   DD 00193          PUSHL   #T                                ;
                               5E   DD 00195          PUSHL   SP                                ;
                 7E       04   AC   7D 00197          MOVQ    4(AP), -(SP)                      ;
          0000V  CF            03   FB 0019B          CALLS   #3, FREE_STRINGS                  ;
                                    04 001A0          RET                                       ;

; Routine Size:  417 bytes,    Routine Base:  _STR$CODE + 0B60


; 1806          1894  1
```

```
1 GLOBAL ROUTINE STR$DIVIDE (
1                     ASIGN,                  ! Sign of operand A
1                     AEXP,                   ! Decimal exponent of operand A
1                     ADIGITS,                ! Digits of operand A
1                     BSIGN,                  ! Sign of operand B
1                     BEXP,                   ! Decimal exponent of operand B
1                     BDIGITS,                ! Digits of operand B
1                     TOT_DIGITS,             !\Number of digits to the right of the
1                                             !/decimal point to carry out the divide
1                     RND_TRUNC,              ! Round/Truncate parameter
1                     CSIGN,                  ! To contain sign of operand C
1                     CEXP,                   ! To contain decimal exponent of operand C
1                     CDIGITS ):NOVALUE =     ! To contain digits of operand C
```

```
1  !++
1  !
1  ! FUNCTIONAL DESCRIPTION:
1  !
1  !  This routine finds the quotient of two decimal strings i.e. C = A / B.
1  !  The algorithm implemented here has been provided by KNUTH.  It is
1  !  his famous Algorithm D (division of non-negative integers (which has
1  !  been modified to handle negative integers)) found in Volume 2 of
1  !  that extraordinary series (Vol. 2 is entitled Seminumerical Algorithms).
1  !  An explanation of the algorithm appears further on in the program.
1  !
1  ! CALLING SEQUENCE:
1  !
1  !       STR$DIVIDE (ASIGN.rv.r,AEXP.rl.r,ADIGITS.rnu.dx,
1  !                   BSIGN.rv.r,BEXP.rl.r,BDIGITS.rnu.dx,
1  !                   TOT_DIGITS.rl.r,RND_TRUNC.rv.r,
1  !                   CSIGN.wv.r,CEXP.wl.r,CDIGITS.wnu.dx)
1  !
1  ! FORMAL PARAMETERS:
1  !
1  !       ASIGN.rv.r       Sign of operand A (0=positive, 1=negative)
1  !       AEXP.rl.r        Power of 10 by which to multiply the operand A digits
1  !                        to get the absolute value of operand A.
1  !                        Ex:  AEXP=1,ADIGITS=123 gives 1230.
1  !       ADIGITS.rnu.dx   Descriptor for the digits of operand A.
1  !       BSIGN.rv.r       Sign of operand B (0=positive, 1=negative)
1  !       BEXP.rl.r        Power of 10 by which to multiply the operand B digits
1  !                        to get the absolute value of operand B.
1  !                        Ex:  BEXP=-1,BDIGITS=123 gives 12.3
1  !       BDIGITS.rnu.dx   Descriptor for the digits of operand B.
1  !       TOT_DIGITS.rl.r  Number of digits to the right of the decimal point
1  !                        to carry out the division
1  !       RND_TRUNC.rv.r   Round/Truncate parameter (0 = truncate, 1 = round)
1  !       CSIGN.wv.r       Sign of operand C (0=positive, 1=negative)
1  !       CEXP.wl.r        Power of 10 by which to multiply the operand C digits
1  !                        to get the absolute value of operand C.
1  !                        Ex:  CEXP=0,CDIGITS=123 gives 123.
1  !       CDIGITS.wnu.dx   Descriptor for the digits of operand C.
1  !
1  ! IMPLICIT INPUTS:
1  !
1  !       -NONE
1  !
1  ! IMPLICIT OUTPUTS:
1  !
1  !       -CSIGN
1  !       -CEXP
1  !       -CDIGITS
1  !
1  ! ROUTINE VALUE:
1  !
1  !       -NONE
1  !
1  ! COMPLETION CODES
1  !
1  !       -NONE
1  !
```

```
; 1879          1965  1 ! MACROS:
; 1880          1966  1 !
; 1881          1967  1 !       -NONE
; 1882          1968  1 !
; 1883          1969  1 ! SIDE EFFECTS:
; 1884          1970  1 !
; 1885          1971  1 !       Signals if storage is exceeded.
; 1886          1972  1 !
; 1887          1973  1 !-
```

```
2 BEGIN
2
2      MAP
2          ADIGITS:REF BLOCK [8,BYTE],
2          BDIGITS:REF BLOCK [8,BYTE],
2          CDIGITS:REF BLOCK [8,BYTE];
2
2      STACKLOCAL
2
2          A_LENGTH:WORD,               ! Number of digits in A string
2          A_ADDR,                      ! Address of A string
2          B_LENGTH:WORD,               ! Number of digits in B string
2          B_ADDR,                      ! Address of B string
2          C_LENGTH:WORD,               ! Number of digits in result string
2          START_BUF,                   ! Pointer to 1st byte of allocated memory
2          A_LEN,                       ! Number of digits needed in A to compute result
2          A_CHUNKS,                    ! Number of 15 digit chunks needed in A
2          ABUF,                        ! Pointer to packed decimal chunks of A
2          B_CHUNKS,                    ! Number of 15 digit chunks in B
2          BBUF,                        ! Pointer to packed decimal chunks of B
2          DRBUF,                       ! Pointer to D / pointer to rounding chunk
2          Q_LENGTH,                    ! Number of digit required in the quotient
2          Q_CHUNKS,                    ! Number of 15 digit chunks required in quotient
2          QBUF,                        ! Pointer to 15 digit chunks of the quotient
2          QSTRBUF,                     ! Pointer to string of quotient digits
2          QBBUF,                       ! Pointer to buffer containing q*B
2          STATUS,                      ! Longword for returning status
2          FLAG,                        ! B_CHUNKS = 1 ==> FLAG = 1, FLAG = 0 otherwise
2          TEMP,                        ! Longword to hold intermediate results
2          LEADING_ZEROS,               ! Number of leading zeros in the quotient string
2          STORAGE: VECTOR[4,BYTE] INITIAL(%C'0'),
2                                       ! Dummy storage area
2          BYTES_VM;                    ! Total bytes of storage allocated
```

```
2 !***********************************************************************
2 !                                                                      *
2 !                         THE ALGORITHM                                *
2 !                                                                      *
2 ! GIVENS:  n = length of the divisor                                   *
2 !          m = length of dividend - n                                  *
2 !          radix = 10 (decimal)                                        *
2 !                                                                      *
2 ! STEP 1.  Normalize.  Set D = FLOOR (radix/(v1+1)) where v1 is  the   *
2 !          first digit of the divisor which must not be zero.  Where   *
2 !          U0 U1...Um+n represent the chunks of  15  digits  of  the   *
2 !          dividend and V1 V2...Vn represent the chunks of 15 digits   *
2 !          of the divisor.                                             *
2 !          Multiply  A  by  D  thus  giving the sequence of 15 digit   *
2 !          chunks U0 U1 U2...Um+n. (Note the introduction of the new   *
2 !          chunk.)  Multiply B by d to obtain a sequence  of  chunks   *
2 !          V1 V2...Vn. (Note no new chunk occurs                       *
2 !                                                                      *
2 ! STEP 2.  Set J = 0.  This is the value we will loop on.  For this    *
2 !          routine we will loop "LOOP" number of times.  Steps  2-7    *
2 !          will provide the basis for the division of Uj Uj+1...Uj+n   *
2 !          by V1 V2...Vn, to get a single quotient digit - Qj.         *
2 !                                                                      *
2 ! STEP 3.  Calculate the first digit of the quotient:                  *
2 !          If Uj = V1 then set q = radix-1.  Otherwise, set q =        *
2 !          FLOOR(Uj*radix + Uj+1)/V1).  Now test if V2*q >             *
2 !          (((Uj*radix + Uj+1 - q*V1)*radix)+Uj+2).  If so, then       *
2 !          decrease q by 1 and repeat this test.  When finish q is     *
2 !          either equal to the qoutient digit or one greater.          *
2 !                                                                      *
2 ! STEP 4.  Multiply and subtract.  Replace Uj Uj+1...Uj+n by           *
2 !          Uj Uj+1...Uj+n - (q * V1 V2...Vn).                          *
2 !          This step consists of a simple multiplication by a one-place *
2 !          number, combined with a subtraction.  The digits            *
2 !          Uj Uj+1...Uj+n should be kept positive; if the result of this*
2 !          step is negative, Uj Uj+1...Uj+n should be left as the true *
2 !          value plus radix raised to the n+1, i.e. as the radix'      *
2 !          complement of the true value, and a "borrow" to the left    *
2 !          should be remembered.                                       *
2 !                                                                      *
2 ! STEP 5.  Set Q[.J] = q.  This is a digit of the quotient.  If the    *
2 !          result of STEP 4 was negative, go to STEP 6; otherwise go to *
2 !          STEP 7.                                                     *
2 !                                                                      *
2 ! STEP 6.  Decrease Q[.J] by 1. Add 0V1 V2...Vn to Uj Uj+1...Uj+n.     *
2 !                                                                      *
2 ! STEP 7.  Loop on J.  If J <= "LOOP" then go back to STEP 3.          *
2 !                                                                      *
2 !***********************************************************************
```

```
2 !+
2 ! Validate input descriptors.
2 !-
2       STATUS = LIB$ANALYZE_SDESC (.ADIGITS, A_LENGTH, A_ADDR);
2       IF .STATUS NEQ SS$_NORMAL
2         THEN
2           LIB$STOP (LIB$_INVARG);                   ! Unsuccessful status
2
2       STATUS = LIB$ANALYZE_SDESC (.BDIGITS, B_LENGTH, B_ADDR);
2       IF .STATUS NEQ SS$_NORMAL
2         THEN
2           LIB$STOP (LIB$_INVARG);                   ! Unsuccessful status
2
2       STATUS = LIB$ANALYZE_SDESC (.CDIGITS, C_LENGTH, TEMP);
2       IF .STATUS NEQ SS$_NORMAL
2         THEN
2           LIB$STOP (LIB$_INVARG);
2 !+
2 ! Validate input strings - If SPANC returns a zero value all characters are
2 ! digits.  If it returns a non-zero value, then TEMP is the address of the
2 ! first non-digit.
2 !-
2       TEMP = SPANC (A_LENGTH, .A_ADDR, SPANC_TABLE, MASK);
2       IF .TEMP NEQ 0                              ! No match found
2         THEN
2           A_LENGTH = .TEMP - .A_ADDR;
2
2       TEMP = SPANC (B_LENGTH, .B_ADDR, SPANC_TABLE, MASK);
2       IF .TEMP NEQ 0                              ! No match found
2         THEN
2           B_LENGTH = .TEMP - .B_ADDR;
```

```
; 2005      2087   2  !+
; 2006      2088   2  !          ***   BEGIN THE DIVISION ALGORITHM   ***
; 2007      2089   2  !-
; 2008      2090   2
; 2009      2091   2  !+
; 2010      2092   2  ! Calculate the resultant sign and exponent.
; 2011      2093   2  !-
; 2012      2094   2          .CSIGN = ..ASIGN XOR ..BSIGN;
; 2013      2095   2          .CEXP = -..TOT_DIGITS;
; 2014      2096   2
; 2015      2097   2  !+
; 2016      2098   2  ! Strip off leading zeros for A and B and compute their length.
; 2017      2099   2  ! CH$FIND_NOT_CH returns a null pointer if the desired match on character
; 2018      2100   2  ! is not found.  To determine if the pointer is null or not,
; 2019      2101   2  ! one must invoke CH$FAIL which returns a value of one if the pointer
; 2020      2102   2  ! is null, and a zero if it is not null.
; 2021      2103   2  !-
; 2022      2104   2         TEMP = CH$FIND_NOT_CH (.A_LENGTH, .A_ADDR, %C'0');
; 2023      2105   2         STATUS = CH$FAIL (.TEMP);
; 2024      2106   2         IF .STATUS EQL 0
; 2025      2107   2           THEN
; 2026      2108   3             BEGIN
; 2027      2109   3             A_LENGTH = .A_LENGTH - (.TEMP - .A_ADDR);
; 2028      2110   3             A_ADDR = .TEMP;
; 2029      2111   3             END
; 2030      2112   2           ELSE
; 2031      2113   2             .CSIGN = 0;
; 2032      2114   2
; 2033      2115   2         TEMP = CH$FIND_NOT_CH (.B_LENGTH, .B_ADDR, %C'0');
; 2034      2116   2         STATUS = CH$FAIL (.TEMP);
; 2035      2117   2         IF .STATUS EQL 1
; 2036      2118   2           THEN
; 2037      2119   2             LIB$STOP (STR$_DIVBY_ZER);
; 2038      2120   2         B_LENGTH = .B_LENGTH - (.TEMP - .B_ADDR);
; 2039      2121   2         B_ADDR = .TEMP;
; 2040      2122   2
; 2041      2123   2  !+
; 2042      2124   2  ! Calculate maximum number of result digits required
; 2043      2125   2  !+
; 2044      2126   3         Q_LENGTH = (.A_LENGTH + ..AEXP) - (.B_LENGTH + ..BEXP)
; 2045      2127   2                    + ..TOT_DIGITS + ..RND_TRUNC;
; 2046      2128   2
; 2047      2129   2         IF .Q_LENGTH LSS 0
; 2048      2130   2           THEN
; 2049      2131   2  !+
; 2050      2132   2  ! Special case for zero quotient
; 2051      2133   2  !-
; 2052      2134   3             BEGIN
; 2053      2135   3             LEADING_ZEROS = 0;
; 2054      2136   3             BYTES_VM = MAXU(.C_LENGTH, 1);
; 2055      2137   3             STATUS = LIB$GET_VM (BYTES_VM, START_BUF);
; 2056      2138   3             QSTRBUF = STORAGE;
; 2057      2139   3             END
; 2058      2140   2           ELSE
; 2059      2141   3             BEGIN
; 2060      2142   3  !+
; 2061      2143   3  ! Determine the number of digits required in A to obtain the proper number
; 2062          2144  3  ! of digits in the result
; 2063          2145  3  !-
; 2064          2146  3            A_LEN = .B_LENGTH + .Q_LENGTH;
; 2065          2147  3  !+
; 2066          2148  3  ! Determine the number of 15 digit CHUNKS needed to hold B, the required
; 2067          2149  3  ! digits of A and the result digits
; 2068          2150  3  !-
; 2069          2151  3            A_CHUNKS = (.A_LEN + 14)/15;
; 2070          2152  3            B_CHUNKS = (.B_LENGTH + 14)/15;
; 2071          2153  3            Q_CHUNKS = (.Q_LENGTH + 29)/15;
; 2072          2154  3  !+
; 2073          2155  3  ! For the algorithm we must have A_CHUNKS >= B_CHUNKS + Q_CHUNKS.
; 2074          2156  3  !-
; 2075          2157  3            A_CHUNKS = MAXU(.A_CHUNKS, .B_CHUNKS + .Q_CHUNKS);
; 2076          2158  3  !+
; 2077          2159  3  ! Determine total storage needed as the maximum of {the storage needed for
; 2078          2160  3  ! the computation of the quotient in packed decimal, the storage needed to
; 2079          2161  3  ! convert the quotient to a string, the length of the result string}
; 2080          2162  3  !-
; 2081          2163  3            BYTES_VM = 8*(.B_CHUNKS*2 + .A_CHUNKS + 3); ! # of bytes to perform
; 2082          2164  3                                                        !   division algorithm in
; 2083          2165  3                                                        !   packed decimal.
; 2084          2166  3            TEMP = (.Q_CHUNKS + 1) * 15;                ! # of bytes needed to hold
; 2085          2167  3                                                        !   quotient string
; 2086          2168  3            BYTES_VM = MAXU (.BYTES_VM, .TEMP + .C_LENGTH);
; 2087          2169  3                                                        ! Need .TEMP+.C_LENGTH
; 2088          2170  3                                                        ! here to ensure string
; 2089          2171  3                                                        ! is long enough for case of
; 2090          2172  3                                                        ! zero padding of fixed
; 2091          2173  3                                                        ! length strings.
; 2092          2174  3  !+
; 2093          2175  3  ! Allocate working storage and set up pointers into it.
; 2094          2176  3  !-
; 2095          2177  3            STATUS = LIB$GET_VM (BYTES_VM, START_BUF);
; 2096          2178  3            DRBUF = .START_BUF;
; 2097          2179  3            QBUF = .DRBUF;
; 2098          2180  3            ABUF = .QBUF + 8;
; 2099          2181  3            BBUF = .ABUF + (.A_CHUNKS + 1)*8;
; 2100          2182  3            QBBUF = .BBUF + .B_CHUNKS*8;
; 2101          2183  3            QSTRBUF = .START_BUF + .BYTES_VM - .Q_CHUNKS*15;
; 2102          2184  3  !+
; 2103          2185  3  ! Convert A and B strings to packed decimal.
; 2104          2186  3  !-
; 2105          2187  3            LIB$$CVT_STR_PACK_R9 (.A_ADDR, .A_LENGTH, .A_CHUNKS, .ABUF + 8);
; 2106          2188  3            MOVP (%REF(15),ZERO,.ABUF);
; 2107          2189  3            LIB$$CVT_STR_PACK_R9 (.B_ADDR, .B_LENGTH, .B_CHUNKS, .BBUF);
```

```
3 !+
3 ! Step 1 - Normalize A and B.  NOTE: If B_CHUNKS = 1 this step is not necessary
3 ! and the computation of q can be simplified.  A flag is used to indicate the
3 ! proper method of evaluating q.  FLAG = 1 if .B_CHUNKS = 1 and 0 otherwise.
3 !-
3               IF .B_CHUNKS NEQ 1
3                 THEN
4                   BEGIN
4                   FLAG = 0;
4                   STATUS = LIB$$CALC_D_R7 (.BBUF, .DRBUF);
4                   IF .STATUS NEQ 1                ! STATUS = 1 <==> D = 1
4                     THEN
5                       BEGIN
5                       LIB$$MUL_PACK_R10 (.DRBUF, .ABUF + 8, .A_CHUNKS,
5                                          .A_CHUNKS + 1, .ABUF + 8);
5                       LIB$$MUL_PACK_R10 (.DRBUF, .BBUF, .B_CHUNKS,
5                                          .B_CHUNKS, .BBUF);
5                       END
4                     ELSE
4                       MOVP (%REF(15), ZERO, .ABUF);
4                   END
3                 ELSE
3                   FLAG = 1;
```

```
3 !+
3 ! Ready to start the actual divide algorithm.
3 !-
3             INCR J FROM 0 TO (.Q_CHUNKS*8 - 8) BY 8 DO
4               BEGIN
4 !+
4 ! Step 3 - Calculate digit of quotient.
4 !-
4               STATUS = LIB$$CALC_Q_R9 (.BBUF, .ABUF + .J, .FLAG, .QBUF + .J);
4               IF .STATUS NEQ 1
4                 THEN
4                   LIB$STOP (LIB$_INVARG);
4 !+
4 ! Step 4 - Multiply and subtract.  Replace the digits of ABUF by ABUF - Q*BBUF
4 !-
4               LIB$$MUL_PACK_R10 (.QBUF + .J, .BBUF, .B_CHUNKS,
4                                  .B_CHUNKS+1, .QBBUF +8);
4               STATUS = LIB$$SUB_PACK_R8 (.B_CHUNKS, .ABUF + .J, .QBBUF);
4 !+
4 ! Step 6 - Adjust q if the resuic of step 4 was negative
4 !-
4               IF .STATUS EQL 1                    ! If remainder is negative
4                 THEN
4                   LIB$$ADJUST_Q_R9 (.B_CHUNKS, .ABUF + .J + 8, .BBUF, .QBUF + .J);
3             END;
3 !+
3 ! Check if rounding is required and round result if necessary
3 !-
3             IF ..RND_TRUNC EQL 1
3               THEN
4                 BEGIN
4                 TEMP = (.Q_CHUNKS-1)*15 - .Q_LENGTH;
4                 Q_LENGTH = .Q_LENGTH - 1;
4                 DRBUF = .QBUF + (.Q_CHUNKS - 1)*8;
4                 LIB$$ROUND_R7 (.DRBUF, .TEMP);
3                 END;
3 !+
3 ! Check if 1st chunk of the quotient is zero.  If it is, A < B, the number of
3 ! leading zeros is 15.  Otherwise, see if its less than 10.  if it is, then
3 ! the number of leading zeros is 14 and the number of digits in the quotient
3 ! should be increased by 1.  Otherwise, the number of leading zeros is 13 and
3 ! the number of digits in the quotient should be increased by 2.
3 !-
3             STATUS = CMPP (%REF(15), .QBUF, %REF(15), ZERO);
3             IF .STATUS EQL 0
3               THEN
3                 LEADING_ZEROS = 15
3               ELSE
4                 BEGIN
4                 STATUS = CMPP (%REF(15), .QBUF, %REF(15), TEN);
4                 IF .STATUS LSS 0
4                   THEN
5                     BEGIN
5                     Q_LENGTH = .Q_LENGTH + 1;
5                     LEADING_ZEROS = 14;
5                     END
4                   ELSE
```

```
5                               BEGIN
5                               Q_LENGTH = .Q_LENGTH + 2;
5                               LEADING_ZEROS = 13;
4                               END;
3                           END;
3
3  !+
3  ! Convert the packed number back into its original numeric form.
3  !-
3                       LIB$$CVT_PACK_STR_R8 (.QBUF, .Q_CHUNKS, .QSTRBUF);
3
2               END;
2  !+
2  ! Check descriptor class to see if the string needs to be padded with leading
2  ! zeros before copying the quotient string to the result string.
2  !-
2               IF (.CDIGITS[DSC$B_CLASS] NEQ DSC$K_CLASS_D) AND
2                  (.CDIGITS[DSC$B_CLASS] NEQ DSC$K_CLASS_VS) AND
3                  (.C_LENGTH GTR .Q_LENGTH)
2                 THEN
3                   BEGIN
3                     TEMP = .C_LENGTH - .Q_LENGTH - .LEADING_ZEROS;
3                     Q_LENGTH = .C_LENGTH;
3                     QSTRBUF = .QSTRBUF - .TEMP;
3                     IF .TEMP GEQ 0
3                       THEN
3                         CH$FILL (%C'0', .TEMP, .QSTRBUF);
3                     END
2                 ELSE
2                   QSTRBUF = .QSTRBUF + .LEADING_ZEROS;
2
2  !+
2  ! Check the type of descriptor our resultant descriptor is.
2  !-
2
2  !            QSTRBUF = .QSTRBUF + .LEADING_ZEROS;
2  !            CHK_STR_TYPE (QSTRBUF,Q_LENGTH,.CDIGITS);
2
2  !+
2  ! Copy quotient string to result string and deallocate virtual memory.
2  !-
2
2               IF .Q_LENGTH LEQ 0
2                 THEN
2                   STATUS = LIB$SCOPY_R_DX (%REF(1),%REF (%ASCII'0'),.CDIGITS)
2                 ELSE
2                   STATUS = LIB$SCOPY_R_DX (Q_LENGTH, .QSTRBUF, .CDIGITS);
2               STATUS = LIB$FREE_VM (BYTES_VM, START_BUF);
1  END;



                                OFFC 00000          .ENTRY  STR$DIVIDE, Save R2,R3,R4,R5,R6,R7,R8,R9,-   ; 1895
                                                            R10,R11                                      ;
          5E       94   AE   9E 00002               MOVAB   -108(SP), SP                                 ;
```

```
                               14  AE             30 D0 00006      MOVL    #48, STORAGE                              ; 1974
                                               64 AE 9F 0000A      PUSHAB  A_ADDR                                    ; 2059
                                               6E AE 9F 0000D      PUSHAB  A_LENGTH                                  ;
                                               0C AC DD 00010      PUSHL   ADIGITS                                   ;
                        00000000G  00             03 FB 00013      CALLS   #3, LIB$ANALYZE_SDESC                     ;
                               24  AE             50 D0 0001A      MOVL    R0, STATUS                                ;
                                   01         24  AE D1 0001E      CMPL    STATUS, #1                                ; 2060
                                                  0D 13 00022      BEQL    1$                                        ;
                                        00000000G 8F DD 00024      PUSHL   #LIB$_INVARG                              ; 2062
                        00000000G  00             01 FB 0002A      CALLS   #1, LIB$STOP                              ;
                                               5C AE 9F 00031 1$:  PUSHAB  B_ADDR                                    ; 2064
                                               66 AE 9F 00034      PUSHAB  B_LENGTH                                  ;
                                               18 AC DD 00037      PUSHL   BDIGITS                                   ;
                        00000000G  00             03 FB 0003A      CALLS   #3, LIB$ANALYZE_SDESC                     ;
                               24  AE             50 D0 00041      MOVL    R0, STATUS                                ;
                                   01         24  AE D1 00045      CMPL    STATUS, #1                                ; 2065
                                                  0D 13 00049      BEQL    2$                                        ;
                                        00000000G 8F DD 0004B      PUSHL   #LIB$_INVARG                              ; 2067
                        00000000G  00             01 FB 00051      CALLS   #1, LIB$STOP                              ;
                                               1C AE 9F 00058 2$:  PUSHAB  TEMP                                      ; 2069
                                               5E AE 9F 0005B      PUSHAB  C_LENGTH                                  ;
                                               2C AC DD 0005E      PUSHL   CDIGITS                                   ;
                        00000000G  00             03 FB 00061      CALLS   #3, LIB$ANALYZE_SDESC                     ;
                               24  AE             50 D0 00068      MOVL    R0, STATUS                                ;
                                   01         24  AE D1 0006C      CMPL    STATUS, #1                                ; 2070
                                                  0D 13 00070      BEQL    3$                                        ;
                                        00000000G 8F DD 00072      PUSHL   #LIB$_INVARG                              ; 2072
                        00000000G  00             01 FB 00078      CALLS   #1, LIB$STOP                              ;
F385    CF    F288  CF         64  BE         6A  AE 2B 0007F 3$:  SPANC   A_LENGTH, @A_ADDR, SPANC_TABLE, MASK      ; 2078
                                                  02 12 0008A      BNEQ    4$                                        ;
                                                  51 D4 0008C      CLRL    R1                                        ;
                               1C  AE             51 D0 0008E 4$:  MOVL    R1, TEMP                                  ;
                                                  07 13 00092      BEQL    5$                                        ; 2079
                6A  AE         1C  AE         64  AE A3 00094      SUBW3   A_ADDR, TEMP, A_LENGTH                    ; 2081
F369    CF    F26C  CF         5C  BE         62  AE 2B 0009B 5$:  SPANC   B_LENGTH, @B_ADDR, SPANC_TABLE, MASK      ; 2083
                                                  02 12 000A6      BNEQ    6$                                        ;
                                                  51 D4 000A8      CLRL    R1                                        ;
                               1C  AE             51 D0 000AA 6$:  MOVL    R1, TEMP                                  ;
                                                  07 13 000AE      BEQL    7$                                        ; 2084
                62  AE         1C  AE         5C  AE A3 000B0      SUBW3   B_ADDR, TEMP, B_LENGTH                    ; 2086
                24  BC         04  BC         10  BC CD 000B7 7$:  XORL3   @BSIGN, @ASIGN, @CSIGN                    ; 2094
                                   53         1C  BC D0 000BE      MOVL    @TOT_DIGITS, R3                           ; 2095
                               28  BC             53 CE 000C2      MNEGL   R3, @CEXP                                 ;
                64  BE         6A  AE             30 3B 000C6      SKPC    #48, A_LENGTH, @A_ADDR                    ; 2104
                                                  02 12 000CC      BNEQ    8$                                        ;
                                                  51 D4 000CE      CLRL    R1                                        ;
                               1C  AE             51 D0 000D0 8$:  MOVL    R1, TEMP                                  ;
                                                  50 D4 000D4      CLRL    R0                                        ; 2105
                                                  51 D5 000D6      TSTL    R1                                        ;
                                                  02 12 000D8      BNEQ    9$                                        ;
                                                  50 D6 000DA      INCL    R0                                        ;
                               24  AE             50 D0 000DC 9$:  MOVL    R0, STATUS                                ;
                                                  0F 12 000E0      BNEQ    10$                                       ; 2106
                    50         64  AE             51 C3 000E2      SUBL3   R1, A_ADDR, R0                            ; 2109
                               6A  AE             50 A0 000E7      ADDW2   R0, A_LENGTH                              ;
                               64  AE             51 D0 000EB      MOVL    R1, A_ADDR                                ; 2110
                                                  03 11 000EF      BRB     11$                                       ; 2106
```

```
                                      24    BC D4 000F1 10$:  CLRL    @CSIGN                      ; 2113
  5C    BE          62    AE                30 3B 000F4 11$:  SKPC    #48, B_LENGTH, @B_ADDR      ; 2115
                                            02 12 000FA       BNEQ    12$                         ;
                                            51 D4 000FC       CLRL    R1                          ;
                    1C    AE                51 D0 000FE 12$:  MOVL    R1, TEMP                    ;
                          52          1C    AE D0 00102       MOVL    TEMP, R2                    ; 2116
                                            50 D4 00106       CLRL    R0                          ;
                                            52 D5 00108       TSTL    R2                          ;
                                            02 12 0010A       BNEQ    13$                         ;
                                            50 D6 0010C       INCL    R0                          ;
                    24    AE                50 D0 0010E 13$:  MOVL    R0, STATUS                  ;
                          01          24    AE D1 00112       CMPL    STATUS, #1                  ; 2117
                                            0D 12 00116       BNEQ    14$                         ;
                                 00000000G  8F DD 00118       PUSHL   #STR$_DIVBY_ZER             ; 2119
             00000000G    00                01 FB 0011E       CALLS   #1, LIB$STOP                ;
          50        5C    AE                52 C3 00125 14$:  SUBL3   R2, B_ADDR, R0              ; 2120
                    62    AE                50 A0 0012A       ADDW2   R0, B_LENGTH                ;
                    5C    AE                52 D0 0012E       MOVL    R2, B_ADDR                  ; 2121
                          50          6A    AE 3C 00132       MOVZWL  A_LENGTH, R0                ; 2126
                          50          08    BC C0 00136       ADDL2   @AEXP, R0                   ;
                          51          62    AE 3C 0013A       MOVZWL  B_LENGTH, R1                ;
                          51          14    BC C0 0013E       ADDL2   @BEXP, R1                   ;
                          50                51 C2 00142       SUBL2   R1, R0                      ;
                          50                53 C0 00145       ADDL2   R3, R0                      ; 2127
  38    AE                50          20    BC C1 00148       ADDL3   @RND_TRUNC, R0, Q_LENGTH    ;
                    08    AE          5A    AE 3C 0014E       MOVZWL  C_LENGTH, 8(SP)             ; 2136
                                      38    AE D5 00153       TSTL    Q_LENGTH                    ; 2129
                                            29 18 00156       BGEQ    16$                         ;
                                      18    AE D4 00158       CLRL    LEADING_ZEROS               ; 2135
                          50          08    AE D0 0015B       MOVL    8(SP), R0                   ; 2136
                                            03 12 0015F       BNEQ    15$                         ;
                          50                01 D0 00161       MOVL    #1, R0                      ;
                    10    AE                50 D0 00164 15$:  MOVL    R0, BYTES_VM                ;
                                      54    AE 9F 00168       PUSHAB  START_BUF                   ; 2137
                                      14    AE 9F 0016B       PUSHAB  BYTES_VM                    ;
             00000000G    00                02 FB 0016E       CALLS   #2, LIB$GET_VM              ;
                    24    AE                50 D0 00175       MOVL    R0, STATUS                  ;
                    2C    AE          14    AE 9E 00179       MOVAB   STORAGE, QSTRBUF            ; 2138
                                          026F 31 0017E       BRW     29$                         ; 2129
                          50          62    AE 3C 00181 16$:  MOVZWL  B_LENGTH, R0                ; 2146
                    50    AE          38  BE40 9E 00185       MOVAB   @Q_LENGTH[R0], A_LEN        ;
          50        50    AE                0E C1 0018B       ADDL3   #14, A_LEN, R0              ; 2151
  4C    AE                50                0F C7 00190       DIVL3   #15, R0, A_CHUNKS           ;
                          50          62    AE 3C 00195       MOVZWL  B_LENGTH, R0                ; 2152
                          50                0E C0 00199       ADDL2   #14, R0                     ;
  44    AE                50                0F C7 0019C       DIVL3   #15, R0, B_CHUNKS           ;
        50          38    AE                1D C1 001A1       ADDL3   #29, Q_LENGTH, R0           ; 2153
  34    AE                50                0F C7 001A6       DIVL3   #15, R0, Q_CHUNKS           ;
        58          44    AE          34    AE C1 001AB       ADDL3   Q_CHUNKS, B_CHUNKS, R8      ; 2157
                          50          4C    AE D0 001B1       MOVL    A_CHUNKS, R0                ;
                          58                50 D1 001B5       CMPL    R0, R8                      ;
                                            03 1E 001B8       BGEQU   17$                         ;
                          50                58 D0 001BA       MOVL    R8, R0                      ;
                    4C    AE                50 D0 001BD 17$:  MOVL    R0, A_CHUNKS                ;
                          51          44    AE D0 001C1       MOVL    B_CHUNKS, R1                ; 2163
                          50          4C  BE41 3E 001C5       MOVAW   @A_CHUNKS[R1], R0           ;
  10    AE                50                03 78 001CA       ASHL    #3, R0, BYTES_VM            ;
```

```
                  10   AE               18   C0 001CF         ADDL2   #24, BYTES_VM                 ;
        57        34   AE               0F   C5 001D3         MULL3   #15, Q_CHUNKS, R7             ; 2166
                  1C   AE       0F      A7   9E 001D8         MOVAB   15(R7), TEMP                  ;
        51        1C   AE       08      AE   C1 001DD         ADDL3   8(SP), TEMP, R1               ; 2168
                       50       10      AE   D0 001E3         MOVL    BYTES_VM, R0                  ;
                       51               50   D1 001E7         CMPL    R0, R1                        ;
                                        03   1E 001EA         BGEQU   18$                           ;
                       50               51   D0 001EC         MOVL    R1, R0                        ;
                  10   AE               50   D0 001EF 18$:    MOVL    R0, BYTES_VM                  ;
                                54      AE   9F 001F3         PUSHAB  START_BUF                     ; 2177
                                14      AE   9F 001F6         PUSHAB  BYTES_VM                      ;
           00000000G   00               02   FB 001F9         CALLS   #2, LIB$GET_VM                ;
                  24   AE               50   D0 00200         MOVL    R0, STATUS                    ;
                  3C   AE       54      AE   D0 00204         MOVL    START_BUF, DRBUF              ; 2178
                  30   AE       3C      AE   D0 00209         MOVL    DRBUF, QBUF                   ; 2179
   48   AE        30   AE               08   C1 0020E         ADDL3   #8, QBUF, ABUF                ; 2180
                       5B       48      AE   D0 00214         MOVL    ABUF, R11                     ; 2181
                       50       4C      AE   D0 00218         MOVL    A_CHUNKS, R0                  ;
                  40   AE       08 AB40      7E 0021C         MOVAQ   8(R11)[R0], BBUF              ;
                       50       44      AE   D0 00222         MOVL    B_CHUNKS, R0                  ; 2182
                  28   AE       40 BE40      7E 00226         MOVAQ   @BBUF[R0], QBBUF              ;
        50        54   AE       10      AE   C1 0022C         ADDL3   BYTES_VM, START_BUF, R0       ; 2183
   2C   AE             50               57   C3 00232         SUBL3   R7, R0, QSTRBUF               ;
                       59       08      AB   9E 00237         MOVAB   8(R11), R9                    ; 2187
                       58       4C      AE   D0 0023B         MOVL    A_CHUNKS, R8                  ;
                       57       6A      AE   3C 0023F         MOVZWL  A_LENGTH, R7                  ;
                       56       64      AE   D0 00243         MOVL    A_ADDR, R6                    ;
                         00000000G      00   16 00247         JSB     LIB$$CVT_STR_PACK_R9          ;
        6B      F0AD   CF               0F   34 0024D         MOVP    #15, ZERO, (R11)              ; 2188
                       59       40      AE   D0 00253         MOVL    BBUF, R9                      ; 2189
                       58       44      AE   D0 00257         MOVL    B_CHUNKS, R8                  ;
                       57       62      AE   3C 0025B         MOVZWL  B_LENGTH, R7                  ;
                       56       5C      AE   D0 0025F         MOVL    B_ADDR, R6                    ;
                         00000000G      00   16 00263         JSB     LIB$$CVT_STR_PACK_R9          ;
                       01       44      AE   D1 00269         CMPL    B_CHUNKS, #1                  ; 2195
                                        59   13 0026D         BEQL    20$                           ;
                                20      AE   D4 0026F         CLRL    FLAG                          ; 2198
                       57       3C      AE   D0 00272         MOVL    DRBUF, R7                     ; 2199
                       56       40      AE   D0 00276         MOVL    BBUF, R6                      ;
                         00000000G      00   16 0027A         JSB     LIB$$CALC_D_R7                ;
                  24   AE               50   D0 00280         MOVL    R0, STATUS                    ;
                       01       24      AE   D1 00284         CMPL    STATUS, #1                    ; 2200
                                        36   13 00288         BEQL    19$                           ;
                       5A       08      AB   9E 0028A         MOVAB   8(R11), R10                   ; 2204
        50        4C   AE               01   C1 0028E         ADDL3   #1, A_CHUNKS, R0              ;
                       59               60   9E 00293         MOVAB   (R0), R9                      ;
                       57       08      AB   9E 00296         MOVAB   8(R11), R7                    ; 2203
                       58       4C      AE   D0 0029A         MOVL    A_CHUNKS, R8                  ;
                       56       3C      AE   D0 0029E         MOVL    DRBUF, R6                     ;
                         00000000G      00   16 002A2         JSB     LIB$$MUL_PACK_R10             ;
                       5A       40      AE   D0 002A8         MOVL    BBUF, R10                     ; 2205
                       59       44      AE   D0 002AC         MOVL    B_CHUNKS, R9                  ;
                       57       40      AE   7D 002B0         MOVQ    BBUF, R7                      ;
                       56       3C      AE   D0 002B4         MOVL    DRBUF, R6                     ;
                         00000000G      00   16 002B8         JSB     LIB$$MUL_PACK_R10             ;
                                        0C   11 002BE         BRB     21$                           ; 2200
        6B      F03A   CF               0F   34 002C0 19$:    MOVP    #15, ZERO, (R11)              ; 2209
```

```
                                                            04 11 002C6          BRB     21$                                 ; 2195
                                      20    AE              01 D0 002C8 20$:     MOVL    #1, FLAG                            ; 2212
                      04    AE        34    AE              03 78 002CC 21$:     ASHL    #3, Q_CHUNKS, 4(SP)                 ; 2216
                                      04    AE              08 C2 002D2          SUBL2   #8, 4(SP)                           ;
                            5A        28    AE              08 C1 002D6          ADDL3   #8, QBBUF, R10                      ; 2229
                      0C    AE        44    AE              01 C1 002DB          ADDL3   #1, B_CHUNKS, 12(SP)                ;
                                            6E              08 CE 002E1          MNEGL   #8, J                               ;
                                                          0081 31 002E4          BRW     24$                                 ;
                                            50        30    AE D0 002E7 22$:     MOVL    QBUF, R0                            ; 2221
                            59              50              6E C1 002EB          ADDL3   J, R0, R9                           ;
                            57              5B              6E C1 002EF          ADDL3   J, R11, R7                          ;
                                            58        20    AE D0 002F3          MOVL    FLAG, R8                            ;
                                            56        40    AE D0 002F7          MOVL    BBUF, R6                            ;
                                                 00000000G  00 16 002FB          JSB     LIB$$CALC_Q_R9                      ;
                                      24    AE              50 D0 00301          MOVL    R0, STATUS                          ;
                                            01        24    AE D1 00305          CMPL    STATUS, #1                          ; 2222
                                                            0D 13 00309          BEQL    23$                                 ;
                                                 00000000G  8F DD 0030B          PUSHL   #LIB$_INVARG                        ; 2224
                               00000000G    00              01 FB 00311          CALLS   #1, LIB$STOP                        ;
                                            50        30    AE D0 00318 23$:     MOVL    QBUF, R0                            ; 2228
                            56              50              6E C1 0031C          ADDL3   J, R0, R6                           ;
                                            59        0C    AE D0 00320          MOVL    12(SP), R9                          ;
                                            57        40    AE 7D 00324          MOVQ    BBUF, R7                            ;
                                                 00000000G  00 16 00328          JSB     LIB$$MUL_PACK_R10                   ;
                            57              5B              6E C1 0032E          ADDL3   J, R11, R7                          ; 2230
                                            58        28    AE D0 00332          MOVL    QBBUF, R8                           ;
                                            56        44    AE D0 00336          MOVL    B_CHUNKS, R6                        ;
                                                 00000000G  00 16 0033A          JSB     LIB$$SUB_PACK_R8                    ;
                                      24    AE              50 D0 00340          MOVL    R0, STATUS                          ;
                                            01        24    AE D1 00344          CMPL    STATUS, #1                          ; 2234
                                                            1E 12 00348          BNEQ    24$                                 ;
                                            50        30    AE D0 0034A          MOVL    QBUF, R0                            ; 2236
                            59              50              6E C1 0034E          ADDL3   J, R0, R9                           ;
                            50              6E              08 C1 00352          ADDL3   #8, J, R0                           ;
                            57              5B              50 C1 00356          ADDL3   R0, R11, R7                         ;
                                            58        40    AE D0 0035A          MOVL    BBUF, R8                            ;
                                            56        44    AE D0 0035E          MOVL    B_CHUNKS, R6                        ;
                                                 00000000G  00 16 00362          JSB     LIB$$ADJUST_Q_R9                    ;
            FF78            6E              08        04    AE F1 00368 24$:     ACBL    4(SP), #8, J, 22$                   ; 2216
                                            01        20    BC D1 0036F          CMPL    @RND_TRUNC, #1                      ; 2241
                                                            2C 12 00373          BNEQ    25$                                 ;
                            57        34    AE              0F C5 00375          MULL3   #15, Q_CHUNKS, R7                   ; 2244
                                            57        38    AE C2 0037A          SUBL2   Q_LENGTH, R7                        ;
                                      1C    AE        F1    A7 9E 0037E          MOVAB   -15(R7), TEMP                       ;
                                                      38    AE D7 00383          DECL    Q_LENGTH                            ; 2245
                            50        34    AE              03 78 00386          ASHL    #3, Q_CHUNKS, R0                    ; 2246
                                            50        30    AE C0 0038B          ADDL2   QBUF, R0                            ;
                                      3C    AE              70 7E 0038F          MOVAQ   -(R0), DRBUF                        ;
                                            57        1C    AE D0 00393          MOVL    TEMP, R7                            ; 2247
                                            56        3C    AE D0 00397          MOVL    DRBUF, R6                           ;
                                                 00000000G  00 16 0039B          JSB     LIB$$ROUND_R7                       ;
                    EF57    CF        30    BE              0F 35 003A1 25$:     CMPP3   #15, @QBUF, ZERO                    ; 2256
                                                            54 DC 003A8          MOVPSL  R4                                  ;
              54            54              02              02 EF 003AA          EXTZV   #2, #2, R4, R4                      ;
                      24    AE              01              54 C3 003AF          SUBL3   R4, #1, STATUS                      ;
                                                            06 12 003B4          BNEQ    26$                                 ; 2257
                                      18    AE              0F D0 003B6          MOVL    #15, LEADING_ZEROS                  ; 2259
```

```
                                                            26 11 003BA         BRB      28$                                      ;
                EF44    CF          30    BE                0F 35 003BC   26$:  CMPP3    #15, @QBUF, TEN                          ; 2262
                                                            54 DC 003C3         MOVPSL   R4                                       ;
          54            54                02                02 EF 003C5         EXTZV    #2, #2, R4, R4                           ;
                  24    AE                01                54 C3 003CA         SUBL3    R4, #1, STATUS                           ;
                                                            09 18 003CF         BGEQ     27$                                      ; 2263
                                                      38    AE D6 003D1         INCL     Q_LENGTH                                 ; 2266
                                    18    AE                0E D0 003D4         MOVL     #14, LEADING_ZEROS                       ; 2267
                                                            08 11 003D8         BRB      28$                                      ; 2263
                                    38    AE                02 C0 003DA   27$:  ADDL2    #2, Q_LENGTH                             ; 2271
                                    18    AE                0D D0 003DE         MOVL     #13, LEADING_ZEROS                       ; 2272
                                          58          2C    AE D0 003E2   28$:  MOVL     QSTRBUF, R8                              ; 2279
                                          56          30    AE 7D 003E6         MOVQ     QBUF, R6                                 ;
                                               00000000G    00 16 003EA         JSB      LIB$$CVT_PACK_STR_R8                     ;
                        50          2C    AC                03 C1 003F0   29$:  ADDL3    #3, CDIGITS, R0                          ; 2286
                                          02                60 91 003F5         CMPB     (R0), #2                                 ;
                                                            36 13 003F8         BEQL     30$                                      ;
                        50          2C    AC                03 C1 003FA         ADDL3    #3, CDIGITS, R0                          ; 2287
                                          0B                60 91 003FF         CMPB     (R0), #11                                ;
                                                            2C 13 00402         BEQL     30$                                      ;
                                    38    AE          08    AE D1 00404         CMPL     8(SP), Q_LENGTH                          ; 2288
                                                            25 15 00409         BLEQ     30$                                      ;
                        50          08    AE          38    AE C3 0040B         SUBL3    Q_LENGTH, 8(SP), R0                      ; 2291
                  1C    AE                50          18    AE C3 00411         SUBL3    LEADING_ZEROS, R0, TEMP                  ;
                                    38    AE          08    AE D0 00417         MOVL     8(SP), Q_LENGTH                          ; 2292
                                    2C    AE          1C    AE C2 0041C         SUBL2    TEMP, QSTRBUF                            ; 2293
                                                      1C    AE D5 00421         TSTL     TEMP                                     ; 2294
                                                            0F 19 00424         BLSS     31$                                      ;
  1C    AE              30                6E                00 2C 00426         MOVC5    #0, (SP), #48, TEMP, @QSTRBUF            ; 2296
                                                      2C    BE    0042C                                                           ;
                                                            05 11 0042E         BRB      31$                                      ; 2286
                                    2C    AE          18    AE C0 00430   30$:  ADDL2    LEADING_ZEROS, QSTRBUF                   ; 2299
                                                      38    AE D5 00435   31$:  TSTL     Q_LENGTH                                 ; 2312
                                                            13 14 00438         BGTR     32$                                      ;
                                                      2C    AC DD 0043A         PUSHL    CDIGITS                                  ; 2314
                                    10    AE                30 D0 0043D         MOVL     #48, 16(SP)                              ;
                                                      10    AE 9F 00441         PUSHAB   16(SP)                                   ;
                                    10    AE                01 D0 00444         MOVL     #1, 16(SP)                               ;
                                                      10    AE 9F 00448         PUSHAB   16(SP)                                   ;
                                                            09 11 0044B         BRB      33$                                      ;
                                                      2C    AC DD 0044D   32$:  PUSHL    CDIGITS                                  ; 2316
                                                      30    AE DD 00450         PUSHL    QSTRBUF                                  ;
                                                      40    AE 9F 00453         PUSHAB   Q_LENGTH                                 ;
                             00000000G    00                03 FB 00456   33$:  CALLS    #3, LIB$SCOPY_R_DX                       ;
                                    24    AE                50 D0 0045D         MOVL     R0, STATUS                               ;
                                                      54    AE 9F 00461         PUSHAB   START_BUF                                ; 2317
                                                      14    AE 9F 00464         PUSHAB   BYTES_VM                                 ;
                             00000000G    00                02 FB 00467         CALLS    #2, LIB$FREE_VM                          ;
                                    24    AE                50 D0 0046E         MOVL     R0, STATUS                               ;
                                                               04 00472         RET                                               ; 2318

; Routine Size:  1139 bytes,    Routine Base:  _STR$CODE + 0D01


; 2239         2319  1 ROUTINE CHK_STR_TYPE (SRC_BUF,SRC_LEN,DST_DESC): NOVALUE =
; 2240         2320  1
; 2241         2321  1
```

```
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine checks the destination string type and copies
1 !       the appropriate number of digits from the source buffer
1 !       into it as dictated by the rules of the destination string.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       CHK_STR_TYPE (src_buf.rnu.r,src_len.rl.r,dst_desc.wnu.dx)
1 !
1 ! FORMAL PARAMETERS:
1 !       SRC_BUF.rnu.r               Addr of the source buffer which contains
1 !                                   the digits of the result.
1 !       SRC_LEN.rl.r                Length of the source string
1 !       DST_DESC.wnu.dx             Addr of the destination string (where to store
1 !                                   the resultant string).
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       -NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       DST_DESC                    Store the resultant string in DST_DESC
1 !
1 ! ROUTINE VALUE:
1 !
1 !       -NONE
1 !
1 ! COMPLETION CODES:
1 !
1 !       -NONE
1 !
1 ! MACROS:
1 !
1 !       -NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       -NONE
1 !--
```

```
2 BEGIN
2
2     MAP
2       DST_DESC:REF BLOCK [8,BYTE];
2
2     LOCAL
2       TMP_BUF:REF VECTOR[65535,BYTE],
2       RESULT_DIGITS,
2       RLEN;
2
2         TMP_BUF = .SRC_BUF;
2         RESULT_DIGITS = ..SRC_LEN;
2 !+
2 ! Check the class of strings we are dealing with.  For
2 ! dynamic and varying strings, return the calculated length;
2 ! for all other classes of strings, return the number of
2 ! characters specified in the destination descriptor.
2 !-
2
2         IF (.DST_DESC[DSC$B_CLASS] EQL DSC$K_CLASS_D) OR
3          (.DST_DESC[DSC$B_CLASS] EQL DSC$K_CLASS_VS)
2           THEN
3             BEGIN
3             IF .DST_DESC[DSC$B_CLASS] EQL DSC$K_CLASS_VS ! Varying string only
3               THEN
4                 BEGIN
4                 RLEN = .DST_DESC[DSC$W_MAXSTRLEN];      ! Fetch max string size
4                 IF .RLEN LSS .RESULT_DIGITS             !\If max < actual then
4                   THEN                                  !/return max # of chars
4 !                   BEGIN
4 !                   CH$MOVE (.RLEN,TMP_BUF,.DST_DESC[DSC$A_POINTER] + 2);
4 !                   (.DST_DESC[DSC$A_POINTER])<0,16> = .RLEN;
4 !                   END
4
4                     (.DST_DESC[DSC$A_POINTER])<0,16> = .RLEN
4                   ELSE
4 !                   BEGIN                                 !\Just retn # of
4 !                                                         !/calculated characters
4 !                   CH$MOVE (.RESULT_DIGITS,TMP_BUF,.DST_DESC[DSC$A_POINTER] + 2);
4 !                   (.DST_DESC[DSC$A_POINTER])<0,16> = .RESULT_DIGITS;
4 !                   END;
4
4                     (.DST_DESC[DSC$A_POINTER])<0,16> = .RESULT_DIGITS;
4                 STR$COPY_R (.DST_DESC,RLEN,.TMP_BUF);
4                 END
3               ELSE
3
3               !+
3               ! Here we know the string is dynamic.
3               ! Return actual number of characters as calculated in algorithm.
3               !-
3
3                 STR$COPY_R (.DST_DESC,RESULT_DIGITS,.TMP_BUF);
3               END
3
3             !+
3             ! Here we know we are dealing with static strings.
```

```
; 2343        2422  3             !-
; 2344        2423  3
; 2345        2424  2             ELSE
; 2346        2425  3               BEGIN
; 2347        2426  3               RLEN = .DST_DESC[DSC$W_LENGTH];              !\Fetch length passed
; 2348        2427  3                                                            !/in output descriptor
; 2349        2428  3               IF .RLEN GTR .RESULT_DIGITS                  ! Given length>actual?
; 2350        2429  3                 THEN
; 2351        2430  4                   BEGIN                                    ! Yes.
; 2352        2431  4                   !+
; 2353        2432  4                   ! Duplicate the zero character for the length
; 2354        2433  4                   ! of the string.  Then copy the calculated
; 2355        2434  4                   ! numeric string into the appropriate offset into
; 2356        2435  4                   ! the destination descriptor.
; 2357        2436  4                   !-
; 2358        2437  4                   STR$DUPL_CHAR (.DST_DESC,RLEN,%REF(%ASCII'0'));
; 2359        2438  4                   CH$MOVE (.RESULT_DIGITS,.TMP_BUF,.DST_DESC[DSC$A_POINTER] +
; 2360        2439  4                           .RLEN - .RESULT_DIGITS);
; 2361        2440  4                   END
; 2362        2441  4                 !+
; 2363        2442  4                 ! Still dealing with static strings here.
; 2364        2443  4                 !-
; 2365        2444  3                 ELSE
; 2366        2445  4                   BEGIN
; 2367        2446  4
; 2368        2447  4                   !+
; 2369        2448  4                   ! For case where RLEN is less than or equal to the
; 2370        2449  4                   ! actual length of the result, just copy RLEN digits
; 2371        2450  4                   ! into the output descriptor.
; 2372        2451  4                   !-
; 2373        2452  4
; 2374        2453  4                   STR$COPY_R (.DST_DESC,RLEN,.TMP_BUF);
; 2375        2454  3                   END;
; 2376        2455  2               END;
; 2377        2456  1 END;



                                003C 00000 CHK_STR_TYPE:
                                                   .WORD   Save R2,R3,R4,R5                          ; 2319
        5E          0C   C2 00002                  SUBL2   #12, SP                                   ;
        53    04    AC   D0 00005                  MOVL    SRC_BUF, TMP_BUF                          ; 2375
  04    AE    08    BC   D0 00009                  MOVL    @SRC_LEN, RESULT_DIGITS                   ; 2376
        52    0C    AC   D0 0000E                  MOVL    DST_DESC, R2                              ; 2384
        02    03    A2   91 00012                  CMPB    3(R2), #2                                 ;
              06         13 00016                  BEQL    1$                                        ;
        0B    03    A2   91 00018                  CMPB    3(R2), #11                                ; 2385
              26         12 0001C                  BNEQ    4$                                        ;
        0B    03    A2   91 0001E 1$:              CMPB    3(R2), #11                                ; 2388
              19         12 00022                  BNEQ    3$                                        ;
  08    AE          62   3C 00024                  MOVZWL  (R2), RLEN                                ; 2391
  04    AE    08    AE   D1 00028                  CMPL    RLEN, RESULT_DIGITS                       ; 2392
              07         18 0002D                  BGEQ    2$                                        ;
  04    B2    08    AE   B0 0002F                  MOVW    RLEN, @4(R2)                              ; 2399
              3A         11 00034                  BRB     5$                                        ;
```

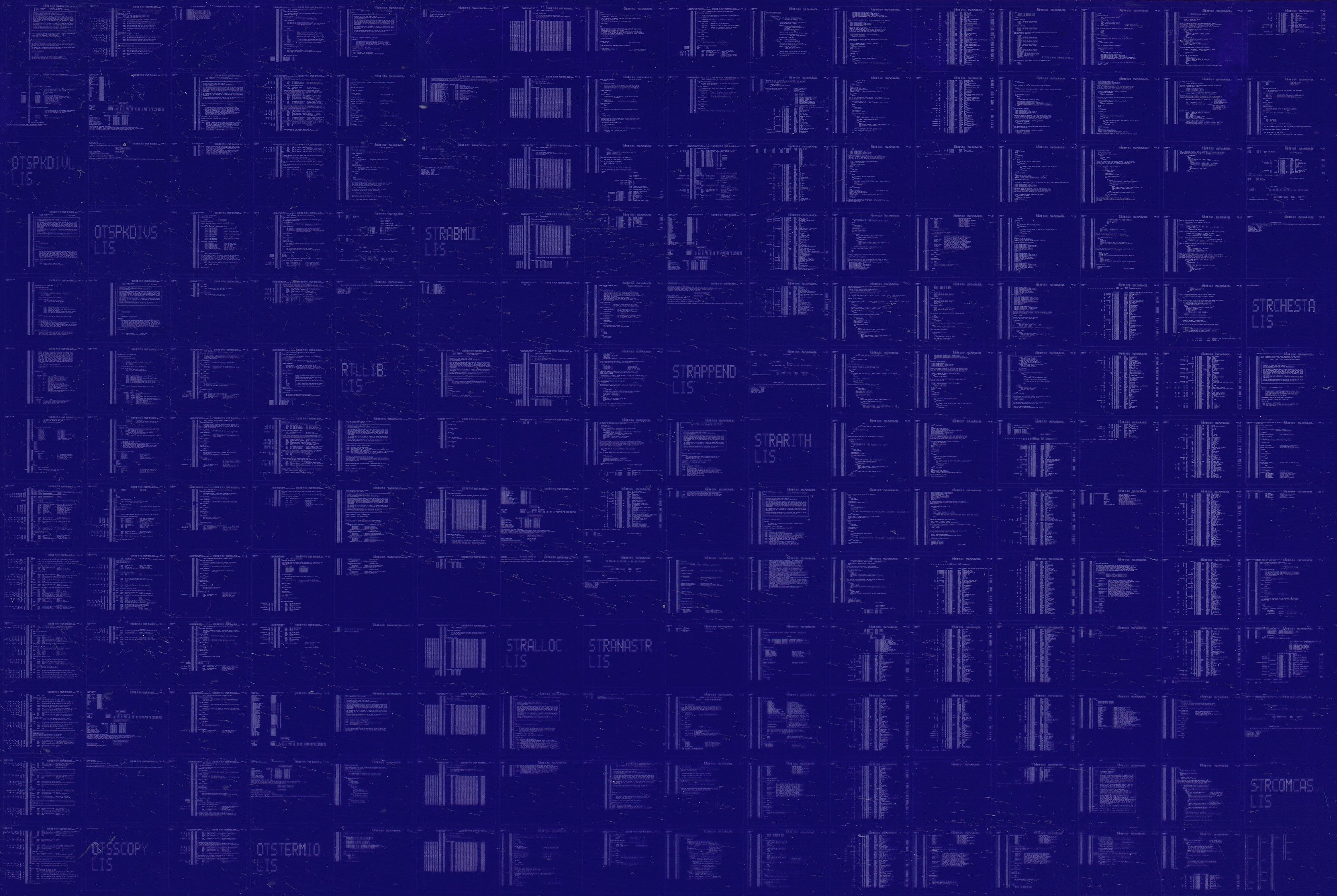
```
                 04    B2       04    AE   B0 00036 2$:    MOVW     RESULT_DIGITS, @4(R2)                   ; 2407
                                      33   11 0003B        BRB      5$                                      ; 2408
                                      53   DD 0003D 3$:    PUSHL    TMP_BUF                                 ; 2417
                                08    AE   9F 0003F        PUSHAB   RESULT_DIGITS                           ;
                                      31   11 00042        BRB      6$                                      ;
                 08    AE             62   3C 00044 4$:    MOVZWL   (R2), RLEN                              ; 2426
                 04    AE       08    AE   D1 00048        CMPL     RLEN, RESULT_DIGITS                     ; 2428
                                      21   15 0004D        BLEQ     5$                                      ;
                       6E             30   D0 0004F        MOVL     #48, (SP)                               ; 2437
                                      5E   DD 00052        PUSHL    SP                                      ;
                                0C    AE   9F 00054        PUSHAB   RLEN                                    ;
                                      52   DD 00057        PUSHL    R2                                      ;
          00000000G    00             03   FB 00059        CALLS    #3, STR$DUPL_CHAR                       ;
    50           04    A2       08    AE   C1 00060        ADDL3    RLEN, 4(R2), R0                         ; 2439
                       50       04    AE   C2 00066        SUBL2    RESULT_DIGITS, R0                       ;
    60                 63       04    AE   28 0006A        MOVC3    RESULT_DIGITS, (TMP_BUF), (R0)          ;
                                           04 0006F        RET                                              ; 2428
                                      53   DD 00070 5$:    PUSHL    TMP_BUF                                 ; 2453
                                0C    AE   9F 00072        PUSHAB   RLEN                                    ;
                                      52   DD 00075 6$:    PUSHL    R2                                      ;
          00000000G    00             03   FB 00077        CALLS    #3, STR$COPY_R                          ;
                                           04 0007E        RET                                              ; 2456

; Routine Size:  127 bytes,    Routine Base:  _STR$CODE + 1174
```

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

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

    BEGIN

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

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

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

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

    INCR ARG_NO FROM 1 TO .ENBL [0] DO

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

```
; 2436          2514  2
; 2437          2515  2      RETURN (SS$_RESIGNAL);
; 2438          2516  1      END;                                              ! end of FREE_STRINGS



                                           0004 00000 FREE_STRINGS:
                                                              .WORD   Save R2                          ; 2457
                       7E     0920    8F   3C 00002          MOVZWL  #2336, -(SP)                     ; 2505
                                      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                           ; 2511
                                      12   11 0001A          BRB     2$                               ;
                         50      0C BC42   D0 0001C  1$:     MOVL    @ENBL[ARG_NO], R0                ; 2513
                                      60   D5 00021          TSTL    (R0)                             ;
                                      09   13 00023          BEQL    2$                               ;
                                      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                        ; 2515
                                           04 00038          RET                                      ; 2516

; Routine Size:  57 bytes,    Routine Base:  _STR$CODE + 11F3


; 2439          2517  1 END                                                    ! end of module STR$ARITH
; 2440          2518  1
; 2441          2519  0 ELUDOM




;
;                        PSECT SUMMARY
;
;    Name                     Bytes                     Attributes
;
;  _STR$CODE                       4652  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        13         0        581        00:00.7
;                                COMMAND QUALIFIERS

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

; Size:            4324 code + 328 data bytes
; Run Time:           00:48.5
; Elapsed Time:       03:07.9
; Lines/CPU Min:       3113
; Lexemes/CPU-Min: 19754
; Memory Used:   373 pages
; Compilation Complete
```

0213 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



OTSPKDIVL
LIS

OTSPKDIVS
LIS

STRABMUL
LIS

STRCHESTA
LIS

RTLLIB
LIS

STRAPPEND
LIS

STRARITH
LIS

STRALLOC
LIS

STRANASTR
LIS

STRCOMCAS
LIS

OTSSCOPY
LIS

OTSTERMIO
LIS