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

```
  SSSSSSSS  TTTTTTTTTT  RRRRRRRR      CCCCCCCC    000000    NN      NN    CCCCCCCC    AAAAAA    TTTTTTTTTT
  SSSSSSSS  TTTTTTTTTT  RRRRRRRR      CCCCCCCC    000000    NN      NN    CCCCCCCC    AAAAAA    TTTTTTTTTT
SS              TT      RR      RR  CC          00      00  NN      NN  CC          AA      AA      TT
SS              TT      RR      RR  CC          00      00  NN      NN  CC          AA      AA      TT
SS              TT      RR      RR  CC          00      00  NNNN    NN  CC          AA      AA      TT
SS              TT      RR      RR  CC          00      00  NNNN    NN  CC          AA      AA      TT
  SSSSSS        TT      RRRRRRRR    CC          00      00  NN  NN  NN  CC          AA      AA      TT
  SSSSSS        TT      RRRRRRRR    CC          00      00  NN  NN  NN  CC          AA      AA      TT
        SS      TT      RR  RR      CC          00      00  NN    NNNN  CC          AAAAAAAAAA      TT
        SS      TT      RR  RR      CC          00      00  NN    NNNN  CC          AAAAAAAAAA      TT
        SS      TT      RR    RR    CC          00      00  NN      NN  CC          AA      AA      TT
        SS      TT      RR    RR    CC          00      00  NN      NN  CC          AA      AA      TT
SSSSSSSS        TT      RR      RR    CCCCCCCC    000000    NN      NN    CCCCCCCC  AA      AA      TT
SSSSSSSS        TT      RR      RR    CCCCCCCC    000000    NN      NN    CCCCCCCC  AA      AA      TT


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

```
;     1          0001  0 MODULE STR$CONCAT (                      ! Concatenate several strings
;     2          0002  0
;     3          0003  0                 IDENT = '1-017' ! File: STRCONCAT.B32   Edit: DG1017
;     4          0004  0
;     5          0005  0                   ) =
;     6          0006  1 BEGIN
;     7          0007  1 !
;     8          0008  1 !****************************************************************************
;     9          0009  1 !*                                                                          *
;    10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;    11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;    12          0012  1 !*  ALL RIGHTS RESERVED.                                                     *
;    13          0013  1 !*                                                                          *
;    14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    19          0019  1 !*  TRANSFERRED.                                                             *
;    20          0020  1 !*                                                                          *
;    21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    23          0023  1 !*  CORPORATION.                                                             *
;    24          0024  1 !*                                                                          *
;    25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;    27          0027  1 !*                                                                          *
;    28          0028  1 !*                                                                          *
;    29          0029  1 !****************************************************************************
;    30          0030  1 !
;    31          0031  1
;    32          0032  1 !++
;    33          0033  1 ! FACILITY: String support library
;    34          0034  1 !
;    35          0035  1 ! ABSTRACT:
;    36          0036  1 !
;    37          0037  1 !       This module takes up to 254 input strings and concatenates
;    38          0038  1 !       them into a result string.  The strings can be of any supported
;    39          0039  1 !       class and data type.
;    40          0040  1 !
;    41          0041  1 ! ENVIRONMENT: VAX-11 User mode
;    42          0042  1 !
;    43          0043  1 ! AUTHOR: R. Will, CREATION DATE: 12-Feb-79
;    44          0044  1 !
;    45          0045  1 ! MODIFIED BY:
;    46          0046  1 !
;    47          0047  1 ! R. Will, 12-Feb-79 : VERSION 01
;    48          0048  1 !
;    49          0049  1 ! 1-001 - Original.
;    50          0050  1 ! 1-002 - Add multiple input strings (up to 254) to the CALL
;    51          0051  1 !           entry point.  JBS 19-MAR-1979
;    52          0052  1 ! 1-003 - Change facility name to STR.  JBS 19-MAR-1979
;    53          0053  1 ! 1-004 - Make several corrections based on the code review.
;    54          0054  1 !         JBS 09-APR-1979
;    55          0055  1 ! 1-005 - Don't allow a concatenation to get longer than 65535 bytes,
;    56          0056  1 !         the limit of string lengths in the VAX architecture.
;    57          0057  1 !         JBS 09-APR-1979
```

```
;    58          0058  1  ;  1-006 - Use the new STR error codes.  JBS 16-MAY-1979
;    59          0059  1  ;  1-007 - Don't return truncate status unless the result length is less
;    60          0060  1  ;          than the sum of the lengths of the sources.  JBS 02-JUL-1979
;    61          0061  1  ;  1-008 - Correct some typos in comments.  JBS 30-JUL-1979
;    62          0062  1  ;  1-009 - Remove BAS$CONCAT, it gets its own module, since it must
;    63          0063  1  ;          signal.  JBS 18-OCT-1979
;    64          0064  1  ;  1-010 - Add code for string interlock.  JBS 01-NOV-1979
;    65          0065  1  ;  1-011 - Convert to using the string macros to doing interlocks.
;    66          0066  1  ;          JBS 06-NOV-1979
;    67          0067  1  ;  1-012 - String speedup, called routines don't signal.  RW  10-Jan-1980
;    68          0068  1  ;  1-013 - Extend to recognize additional classes of descriptors by
;    69          0069  1  ;          using $STR$GET_LEN_ADDR to extract length and address from
;    70          0070  1  ;          descriptors.  Remove string interlocking code.
;    71          0071  1  ;          RKR 15-APR-1981
;    72          0072  1  ;  1-014 - Speed up code.  RKR 7-OCT-1981.
;    73          0073  1  ;  1-015 - Use $STR$SIGNAL_FATAL instead of $STR$CHECK_STATUS.
;    74          0074  1  ;          RKR 18-NOV-1981.
;    75          0075  1  ;  1-016 - Add support for class SO string descriptor.  DG 3-Oct-1983
;    76          0076  1  ;  1-017 - Change class SO string descriptor to SB.  DG 27-Feb-1984
;    77          0077  1  ;--
;    78          0078  1
;    79          0079  1  !<BLF/PAGE>
```

```
;    81          0080  1 !
;    82          0081  1 ! SWITCHES:
;    83          0082  1 !
;    84          0083  1
;    85          0084  1 SWITCHES ADDRESSING_MODE
;    86          0085  1                 (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;    87          0086  1
;    88          0087  1 !
;    89          0088  1 ! LINKAGES:
;    90          0089  1 !
;    91          0090  1
;    92          0091  1 REQUIRE 'RTLIN:STRLNK';         ! Use require file with string linkages
;    93          0276  1
;    94          0277  1 !
;    95          0278  1 ! TABLE OF CONTENTS:
;    96          0279  1 !
;    97          0280  1
;    98          0281  1 FORWARD ROUTINE
;    99          0282  1     STR$CONCAT;                 ! Concatenate two or more strings
;   100          0283  1
;   101          0284  1 !
;   102          0285  1 ! INCLUDE FILES:
;   103          0286  1 !
;   104          0287  1
;   105          0288  1 REQUIRE 'RTLIN:RTLPSECT';       ! Declare PSECTS code
;   106          0383  1
;   107          0384  1 REQUIRE 'RTLIN:STRMACROS';      ! use string macros to write code
;   108          1300  1
;   109          1301  1 LIBRARY 'RTLSTARLE';            ! STARLET library for macros and symbols
;   110          1302  1
;   111          1303  1 !
;   112          1304  1 ! MACROS: NONE
;   113          1305  1 !
;   114          1306  1 !
;   115          1307  1 ! EQUATED SYMBOLS:
;   116          1308  1 !
;   117          1309  1 !       NONE
;   118          1310  1 !
;   119          1311  1 ! PSECT DECLARATIONS
;   120          1312  1 !
;   121          1313  1
;   122          1314  1 DECLARE_PSECTS (STR);           ! Declare psects for STR$ facility
;   123          1315  1
;   124          1316  1 !
;   125          1317  1 ! OWN STORAGE:
;   126          1318  1 !
;   127          1319  1 !       NONE
;   128          1320  1 !
;   129          1321  1 ! EXTERNAL REFERENCES:
;   130          1322  1 !
;   131          1323  1
;   132          1324  1 EXTERNAL ROUTINE
;   133          1325  1     LIB$STOP;                                  ! Signal fatal errors
;   134          1326  1
;   135          1327  1 !+
;   136          1328  1 ! The following are the error messages used in this module:
;   137          1329  1 !-
```

```
;  138          1330  1
;  139          1331  1 EXTERNAL LITERAL
;  140          1332  1     STR$_NORMAL,          ! Success
;  141          1333  1     STR$_STRIS_INT,       ! String is interlocked
;  142          1334  1     STR$_ILLSTRCLA,       ! Illegal string class
;  143          1335  1     STR$_TRU,             ! Truncation
;  144          1336  1     STR$_FATINTERR,       ! Fatal internal error
;  145          1337  1     STR$_STRTOOLON,       ! String too long
;  146          1338  1     STR$_WRONUMARG;       ! Wrong number of arguments
;  147          1339  1
```

```
;  149          1340  1 GLOBAL ROUTINE STR$CONCAT (                  ! Concatenate strings
;  150          1341  1
;  151          1342  1         DEST_DESC                   ! pointer to destination descriptor
;  152          1343  1
;  153          1344  1                                 ) =
;  154          1345  1
;  155          1346  1 !++
;  156          1347  1 ! FUNCTIONAL DESCRIPTION
;  157          1348  1 !
;  158          1349  1 !       This routine takes up to 254 source strings of any supported
;  159          1350  1 !       DTYPE and CLASS, concatenates them, and assigns that value to
;  160          1351  1 !       the destination string.
;  161          1352  1 !
;  162          1353  1 ! FORMAL PARAMETERS:
;  163          1354  1 !
;  164          1355  1 !       DEST_DESC.wt.dx         Pointer to destination descriptor
;  165          1356  1 !       [INPUT].rt.dx           Pointer to input string descriptor.
;  166          1357  1 !                               There can be up to 254 of these.
;  167          1358  1 !
;  168          1359  1 ! IMPLICIT INPUTS:
;  169          1360  1 !
;  170          1361  1 !       NONE
;  171          1362  1 !
;  172          1363  1 ! IMPLICIT OUTPUTS:
;  173          1364  1 !
;  174          1365  1 !       NONE
;  175          1366  1 !
;  176          1367  1 ! COMPLETION CODES:
;  177          1368  1 !
;  178          1369  1 !       SS$_NORMAL      All of the characters in the input strings were
;  179          1370  1 !                       copied into the destination string.
;  180          1371  1 !       STR$_TRU        One or more input characters were not copied.
;  181          1372  1 !                       This can only happen when the destination is a
;  182          1373  1 !                       string having fixed-length semantics.
;  183          1374  1 !
;  184          1375  1 ! SIDE EFFECTS:
;  185          1376  1 !
;  186          1377  1 !       May allocate storage for the destination.
;  187          1378  1 !       This routine signals if allocation fails (STR$_INSVIRMEM)
;  188          1379  1 !       or a descriptor is bad (STR$_ILLSTRCLA).  An attempt to create a
;  189          1380  1 !       dynamic string longer than 65535 bytes signals STR$_STRTOOLON,
;  190          1381  1 !       STR$_FATINTERR if the debug switch is set in
;  191          1382  1 !       STRMACROS and there is some internal corruption, and
;  192          1383  1 !       STR$_WRONUMARG if there are not at least 2 arguments to this
;  193          1384  1 !       routine.
;  194          1385  1 !
;  195          1386  1 !--
;  196          1387  1
;  197          1388  2     BEGIN
;  198          1389  2
;  199          1390  2     BUILTIN
;  200          1391  2         ACTUALPARAMETER,
;  201          1392  2         ACTUALCOUNT;
;  202          1393  2
;  203          1394  2     MAP
;  204          1395  2         DEST_DESC : REF $STR$DESCRIPTOR;
;  205          1396  2
```

```
;  206          1397  2     LITERAL
;  207          1398  2         MAX_SIZE        = 65535,        ! largest string we can handle
;  208          1399  2         FIRST_INPUT_ARG = 2;            ! Argument number of the first
;  209          1400  2                                         ! input
;  210          1401  2                                         ! string
;  211          1402  2
;  212          1403  2     LOCAL
;  213          1404  2         OUT_LEN,                    ! original length of destination string
;  214          1405  2         OUT_ADDR,                   ! address of 1st byte of original
;  215          1406  2                                     ! destination string
;  216          1407  2         RETURN_STATUS,              ! status from alloc and dealloc
;  217          1408  2         OVERLAP_FLAG,               ! =1 if input strings overlap dest
;  218          1409  2         TOTAL_LENGTH,               ! Sum of bytes in sources
;  219          1410  2         RESULT_LENGTH,              ! Number of bytes in destination
;  220          1411  2         RESULT_CLASS;               ! Descriptor class of destination
```

```
;  222     1412  2 !+
;  223     1413  2 ! This routine contains a great deal of repetitious code.  This is done
;  224     1414  2 ! deliberately so that each class of destination string is handled
;  225     1415  2 ! as efficiently as possible with a minimal amount of invocations of
;  226     1416  2 ! common code.  As an overall guide to the following pages of code,
;  227     1417  2 ! note the overall structure of the code, as indicated below.
;  228     1418  2 !
;  229     1419  2 !       -----
;  230     1420  2 !       |
;  231     1421  2 !       | Loop to count up the total lengths of all the input strings
;  232     1422  2 !       | and to detect whether any of the inputs overlap with the
;  233     1423  2 !       | output area.  If overlaps exist, we must do concatenation
;  234     1424  2 !       | into a temporary area, then move temporary area to true
;  235     1425  2 !       | destination area.  If no overlap, copying directly into
;  236     1426  2 !       | destination area will occur.
;  237     1427  2 !       |
;  238     1428  2 !       -----
;  239     1429  2 !
;  240     1430  2 !       ----- CASE on class of output descriptor
;  241     1431  2 !       |
;  242     1432  2 !       | -- Classes S, Z, A, NCA, SD and SB.  These classes have
;  243     1433  2 !       |    fixed-length string semantics and are copied with
;  244     1434  2 !       |    trailing padding if necessary.  Those that don't fit
;  245     1435  2 !       |    return STR$_TRU
;  246     1436  2 !       |
;  247     1437  2 !       |       -----
;  248     1438  2 !       |       |
;  249     1439  2 !       |       | Code for fixed-length semantic strings, where one or
;  250     1440  2 !       |       | more sources overlap destination string.
;  251     1441  2 !       |       |
;  252     1442  2 !       |       | or
;  253     1443  2 !       |       |
;  254     1444  2 !       |       | Code for fixed-length semantic strings, where there
;  255     1445  2 !       |       | is no overlap.
;  256     1446  2 !       |       |
;  257     1447  2 !       |       -----
;  258     1448  2 !       |
;  259     1449  2 !       | -- Class D.  This class of descriptor has
;  260     1450  2 !       |    dynamic-length string semantics and is copied with
;  261     1451  2 !       |    no trailing padding.  Those that don't fit within 65K
;  262     1452  2 !       |    signal STR$_TOOLON.
;  263     1453  2 !       |
;  264     1454  2 !       |       -----
;  265     1455  2 !       |       |
;  266     1456  2 !       |       | Code for dynamic-length semantic strings, where one or
;  267     1457  2 !       |       | more sources overlap destination string.
;  268     1458  2 !       |       |
;  269     1459  2 !       |       | or
;  270     1460  2 !       |       |
;  271     1461  2 !       |       | Code for dynamic-length semantic strings, where there
;  272     1462  2 !       |       | is no overlap.
;  273     1463  2 !       |       |
;  274     1464  2 !       |       -----
;  275     1465  2 !       |
;  276     1466  2 !       | -- Class VS.  This class of descriptor has
;  277     1467  2 !       |    varying-length string semantics and is copied with
;  278     1468  2 !       |    no trailing padding.  Those that don't fit within
```

```
;  279          1469  2 !        !    DSC$W_MAXSTRLEN return STR$_TRU.
;  280          1470  2 !        !
;  281          1471  2 !        !       -----
;  282          1472  2 !        !       !
;  283          1473  2 !        !       ! Code for varying-length semantic strings, where one or
;  284          1474  2 !        !       ! more sources overlap destination string.
;  285          1475  2 !        !       !
;  286          1476  2 !        !       ! or
;  287          1477  2 !        !       !
;  288          1478  2 !        !       ! Code for varying-length semantic strings, where there
;  289          1479  2 !        !       ! is no overlap.
;  290          1480  2 !        !       !
;  291          1481  2 !        !       -----
;  292          1482  2 !      -----
```

```
;  294     1483  2 !+
;  295     1484  2 ! Check for a proper number of arguments and preset return status.
;  296     1485  2 !-
;  297     1486  2
;  298     1487  2     IF (ACTUALCOUNT () LSS FIRST_INPUT_ARG)
;  299     1488  2     THEN
;  300     1489  2         BEGIN
;  301     1490  2         !+
;  302     1491  2         ! Build a local fixed-length descriptor pointing to name of this
;  303     1492  2         ! routine and use it to signal STR$_WRONUMARG.
;  304     1493  2         !-
;  305     1494  2         LOCAL
;  306     1495  2             ROUT_NAME_DESC : $STR$DESCRIPTOR;
;  307     1496  2
;  308     1497  2         ROUT_NAME_DESC [DSC$W_LENGTH] = 10 ;
;  309     1498  2         ROUT_NAME_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  310     1499  2         ROUT_NAME_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
;  311     1500  2         ROUT_NAME_DESC [DSC$A_POINTER] = UPLIT (%ASCII'STR$CONCAT');
;  312     1501  2         LIB$STOP (STR$_WRONUMARG, 2, ACTUALCOUNT (), ROUT_NAME_DESC);
;  313     1502  2         END;
;  314     1503  2
;  315     1504  2     RETURN_STATUS = 1 ;          ! Assume success to follow
;  316     1505  2
;  317     1506  2 !+
;  318     1507  2 ! Extract length and address of destination string.
;  319     1508  2 !-
;  320     1509  2     $STR$GET_LEN_ADDR (DEST_DESC, OUT_LEN, OUT_ADDR ) ;
;  321     1510  2
;  322     1511  2 !+
;  323     1512  2 ! Check each source argument for overlapping the destination.
;  324     1513  2 ! Note that the code below will sometimes decide we have overlap when
;  325     1514  2 ! we do not: if the destination string is fixed-length and shorter
;  326     1515  2 ! than the sum of the sources, we will reach beyond the end of the
;  327     1516  2 ! destination string, and may run into a source string.  The consequent
;  328     1517  2 ! decrease in speed (because of using a temporary descriptor needlessly)
;  329     1518  2 ! is more than made up for by the improved speed of the scanning loop
;  330     1519  2 ! below.
;  331     1520  2 !-
;  332     1521  2     OVERLAP_FLAG = 0;
;  333     1522  2     TOTAL_LENGTH = 0;
;  334     1523  2
;  335     1524  2     !+
;  336     1525  2     ! Now step through all the input descriptors
;  337     1526  2     !-
;  338     1527  2     INCR ARG_NO FROM FIRST_INPUT_ARG TO ACTUALCOUNT () DO
;  339     1528  3         BEGIN
;  340     1529  3
;  341     1530  3         LOCAL
;  342     1531  3             IN_LEN,                      ! length of Nth input string
;  343     1532  3             IN_ADDR,                     ! addr of 1st byte of Nth string
;  344     1533  3             SRC_DESC : REF $STR$DESCRIPTOR;     ! addr of Nth input
;  345     1534  3                                                 ! string descriptor
;  346     1535  3
;  347     1536  3         SRC_DESC = ACTUALPARAMETER (.ARG_NO);   ! get Nth descr address
;  348     1537  3
;  349     1538  3         !+
;  350     1539  3         ! Extract length and address of this input string.
```

```
;  351          1540  3          !-
;  352          1541  3          $STR$GET_LEN_ADDR (SRC_DESC, IN_LEN, IN_ADDR) ;
;  353          1542  3
;  354          1543  3          TOTAL_LENGTH = .TOTAL_LENGTH + .IN_LEN;
;  355          1544  3
;  356          1545  3          !+
;  357          1546  3          ! If this string overlaps destination then set OVERLAP.
;  358          1547  3          ! In either case, continue looping through all sources so that
;  359          1548  3          ! we know total length involved.
;  360          1549  3          !-
;  361          1550  3          IF ($STR$OVERLAP ( .IN_ADDR, .IN_LEN, .OUT_ADDR, .TOTAL_LENGTH))
;  362          1551  3          THEN
;  363          1552  3              OVERLAP_FLAG = 1;
;  364          1553  3
;  365          1554  2          END;                ! of total length of sources computation and
;  366          1555  2                              ! overlap detection
;  367          1556  2      !+
;  368          1557  2      ! The remainder of the algorithm is different for each class of output
;  369          1558  2      ! string descriptor.
;  370          1559  2      !-
;  371          1560  2          RESULT_CLASS = .DEST_DESC [DSC$B_CLASS];    ! Class of output desc
;  372          1561  2
;  373          1562  2          CASE .RESULT_CLASS FROM DSC$K_CLASS_Z TO DSC$K_CLASS_SB OF
;  374          1563  2              SET
;  375          1564  2
```

```
;  377          1565  2          [DSC$K_CLASS_Z,                 ! Unspecific class (assume S)
;  378          1566  2           DSC$K_CLASS_S,                 ! Fixed length string
;  379          1567  2           DSC$K_CLASS_A,                 ! Array
;  380          1568  2           DSC$K_CLASS_NCA,               ! Non-contiguous array
;  381          1569  2           DSC$K_CLASS_SD,                ! Scaled decimal
;  382          1570  2           DSC$K_CLASS_SB] :              ! String with bounds
;  383          1571  2      !+
;  384          1572  2      ! The destination string has fixed-length semantics.  Copy only as
;  385          1573  2      ! much of the sources into it as its length allows.  If its storage
;  386          1574  2      ! overlaps any of the source strings, do the concatenation into a
;  387          1575  2      ! temporary string and then copy back to the destination string.
;  388          1576  2      ! If sum of source lengths less than destination length, pad with
;  389          1577  2      ! fill character.
;  390          1578  2      !-
;  391          1579  2
;  392          1580  3              BEGIN                       ! Class_S, _Z, _A, _NCA, _SD, _SB
;  393          1581  4              IF (.OVERLAP_FLAG)
;  394          1582  3              THEN
;  395          1583  4                  BEGIN
;  396          1584  4
;  397          1585  4                  LOCAL
;  398          1586  4                      CHR_PTR,            ! Variable pointer into output
;  399          1587  4                      TEMP_DESC : $STR$DESCRIPTOR;
;  400          1588  4
;  401          1589  4                  RETURN_STATUS =
;  402        P 1590  4                      $STR$ALLOC_TMP (MIN (MAX_SIZE, .TOTAL_LENGTH),
;  403          1591  4                                      TEMP_DESC);
;  404          1592  4
;  405          1593  4                  !+
;  406          1594  4                  ! If allocate didn't work, don't continue the
;  407          1595  4                  ! concatenate
;  408          1596  4                  !-
;  409          1597  4
;  410          1598  4                  IF .RETURN_STATUS
;  411          1599  4                  THEN
;  412          1600  5                      BEGIN
;  413          1601  5                      CHR_PTR = .TEMP_DESC [DSC$A_POINTER]; ! init to
;  414          1602  5                                                            ! start of
;  415          1603  5                                                            ! temp output
;  416          1604  5
;  417          1605  5                      INCR ARG_NO FROM FIRST_INPUT_ARG TO ACTUALCOUNT() DO
;  418          1606  6                          BEGIN   ! copying loop
;  419          1607  6
;  420          1608  6                          LOCAL
;  421          1609  6                              IN_LEN,     ! length of Nth input string
;  422          1610  6                              IN_ADDR,    ! address of 1st byte of Nth
;  423          1611  6                                          ! input string
;  424          1612  6
;  425          1613  6                              SRC_DESC : REF $STR$DESCRIPTOR;
;  426          1614  6
;  427          1615  6                          !+
;  428          1616  6                          ! Get Nth input descriptor address
;  429          1617  6                          !-
;  430          1618  6                          SRC_DESC = ACTUALPARAMETER (.ARG_NO);
;  431          1619  6
;  432          1620  6                          !+
;  433          1621  6                          ! Extract length and address of this input
```

```
;  434     1622  6                    ! string.  There is no need to check status on
;  435     1623  6                    ! these calls.  If there was anything
;  436     1624  6                    ! wrong with the input descriptors, we would
;  437     1625  6                    ! have signaled our way out of the loop where
;  438     1626  6                    ! we added up the total lengths of the inputs.
;  439     1627  6                    !-
;  440     1628  6                    $STR$GET_LEN_ADDR (SRC_DESC, IN_LEN, IN_ADDR) ;
;  441     1629  6
;  442     1630  6                    CHR_PTR = CH$MOVE (.IN_LEN, .IN_ADDR, .CHR_PTR);
;  443     1631  5                    END;    ! copying loop
;  444     1632  5
;  445     1633  5                !+
;  446     1634  5                ! Now copy from the temporary descriptor to the real
;  447     1635  5                ! destination.  The destination may be shorter than
;  448     1636  5                ! TOTAL_LENGTH, in which case fewer characters will
;  449     1637  5                ! be copied than were concatenated, or it may be
;  450     1638  5                ! longer, in which case the destination will be
;  451     1639  5                ! padded with blanks.
;  452     1640  5                !-
;  453     1641  5                CH$COPY ( MIN (MAX_SIZE, .TOTAL_LENGTH),
;  454     1642  5                          .TEMP_DESC [DSC$A_POINTER],
;  455     1643  5                          STR$K_FILL_CHAR,
;  456     1644  5                          .OUT_LEN,
;  457     1645  5                          .OUT_ADDR);
;  458     1646  5
;  459     1647  5                RETURN_STATUS = $STR$DEALOC_TMP (TEMP_DESC);
;  460     1648  4                END;        ! of concatenation and copy via temp
;  461     1649  4
;  462     1650  4                    !+
;  463     1651  4                    ! Record actual size of constructed output string
;  464     1652  4                    ! for later evaluation of what status to return.
;  465     1653  4                    !-
;  466     1654  4                    RESULT_LENGTH = .OUT_LEN ;
;  467     1655  4                END             ! of overlap subcase
;  468     1656  4
;  469     1657  3            ELSE
;  470     1658  4                BEGIN
;  471     1659  4                !+
;  472     1660  4                ! This is the case of a fixed-length destination which
;  473     1661  4                ! does not overlap any of the sources.  We can copy
;  474     1662  4                ! directly into the destination space.
;  475     1663  4                !-
;  476     1664  4
;  477     1665  4                LOCAL
;  478     1666  4                    CHR_PTR,
;  479     1667  4                    CHARS_MOVED,
;  480     1668  4                    ARG_NO;
;  481     1669  4
;  482     1670  4                CHR_PTR = .OUT_ADDR;    ! init to 1st byte of dest
;  483     1671  4                CHARS_MOVED = 0;
;  484     1672  4                ARG_NO = FIRST_INPUT_ARG;
;  485     1673  4
;  486     1674  4                WHILE (.CHARS_MOVED NEQ .OUT_LEN) DO
;  487     1675  5                    BEGIN
;  488     1676  5                    !+
;  489     1677  5                    ! There is room for more characters in the
;  490     1678  5                    ! destination string.  Copy as much of the next
```

```
;  491     1679  5                ! input string as will fit.
;  492     1680  5                !-
;  493     1681  5
;  494     1682  5                LOCAL
;  495     1683  5                    IN_LEN,             ! length of Nth input string
;  496     1684  5                    IN_ADDR,            ! address of 1st byte of Nth
;  497     1685  5                                        ! input string
;  498     1686  5                    CHARS_LEFT;
;  499     1687  5
;  500     1688  5                CHARS_LEFT = .OUT_LEN - .CHARS_MOVED;
;  501     1689  5
;  502     1690  6                IF (.ARG_NO GTR ACTUALCOUNT ())
;  503     1691  5                THEN
;  504     1692  6                    BEGIN
;  505     1693  6                    !+
;  506     1694  6                    ! We have exhausted the parameters, fill the
;  507     1695  6                    ! remainder of the destination string with
;  508     1696  6                    ! blanks.
;  509     1697  6                    !-
;  510     1698  6                    CH$FILL (STR$K_FILL_CHAR, .CHARS_LEFT, .CHR_PTR);
;  511     1699  6                    CHARS_MOVED = .CHARS_MOVED + .CHARS_LEFT;
;  512     1700  6                    END
;  513     1701  6
;  514     1702  5                ELSE
;  515     1703  5
;  516     1704  6                    BEGIN           ! copy of one string
;  517     1705  6                    !+
;  518     1706  6                    ! We have another input string.  Copy it into
;  519     1707  6                    ! the destination string, or as much of it as
;  520     1708  6                    ! will fit.
;  521     1709  6                    !-
;  522     1710  6
;  523     1711  6                    LOCAL
;  524     1712  6                        SRC_DESC : REF $STR$DESCRIPTOR;
;  525     1713  6
;  526     1714  6                    SRC_DESC = ACTUALPARAMETER (.ARG_NO);
;  527     1715  6                    !+
;  528     1716  6                    ! Extract length and address of this input
;  529     1717  6                    ! string.  There is no need to check status on
;  530     1718  6                    ! these calls.  If there was anything
;  531     1719  6                    ! wrong with the input descriptors, we would
;  532     1720  6                    ! have signaled our way out of the loop where
;  533     1721  6                    ! we added up the total lengths of the inputs.
;  534     1722  6                    !-
;  535     1723  6                    $STR$GET_LEN_ADDR (SRC_DESC, IN_LEN, IN_ADDR) ;
;  536     1724  6
;  537     1725  6                    CHR_PTR = CH$MOVE ( MIN (.IN_LEN, .CHARS_LEFT),
;  538     1726  6                                        .IN_ADDR, .CHR_PTR);
;  539     1727  6
;  540     1728  6                    CHARS_MOVED = .CHARS_MOVED +
;  541     1729  6                                  MIN (.IN_LEN, .CHARS_LEFT);
;  542     1730  6
;  543     1731  6                    ARG_NO = .ARG_NO + 1;
;  544     1732  5                    END;            ! copy of one string
;  545     1733  5
;  546     1734  4                END;                ! of WHILE loop
;  547     1735  4
```

```
;  548          1736  4                    !+
;  549          1737  4                    ! Record the actual length of the output string
;  550          1738  4                    ! for later evaluation of the status to be returned.
;  551          1739  4                    !-
;  552          1740  4                    RESULT_LENGTH = .CHARS_MOVED ;
;  553          1741  4
;  554          1742  3                ELSE;                       ! of non-overlapped
;  555          1743  3                                            ! concatenation operation
;  556          1744  3
;  557          1745  2            END;                            ! of Class_S, _Z, _A, _NCA, _SD, _SB
```

```
;  559          1746  2          [DSC$K_CLASS_D] :
;  560          1747  2      !+
;  561          1748  2      ! If we must reallocate the destination string (because the old string
;  562          1749  2      ! was not as long as the sum of the lengths of the source strings)
;  563          1750  2      ! or if the source strings overlap the destination string (which means
;  564          1751  2      ! that we are concatenating a substring of the result string, and
;  565          1752  2      ! therefore must not store into the destination string until we finish
;  566          1753  2      ! fetching all of the source strings) then we must use a temporary
;  567          1754  2      ! descriptor to hold the concatenation.  This is important for the
;  568          1755  2      ! reallocation case so that an AST will see, when looking at
;  569          1756  2      ! any particular character position of the string, either the old
;  570          1757  2      ! character or the new one.  The AST will never see, for example,
;  571          1758  2      ! an empty string into which we have not yet copied the first input
;  572          1759  2      ! string.
;  573          1760  2      !-
;  574          1761  3                  BEGIN
;  575          1762  3
;  576          1763  4                  IF (.OVERLAP_FLAG
;  577          1764  4                      OR
;  578        P 1765  4                      $STR$NEED_ALLOC ( MIN (MAX_SIZE, .TOTAL_LENGTH),
;  579          1766  5                                        $STR$DYN_AL_LEN (DEST_DESC))
;  580          1767  4                      OR
;  581          1768  4                      (.TOTAL_LENGTH GTR MAX_SIZE))
;  582          1769  3                  THEN
;  583          1770  4                      BEGIN
;  584          1771  4
;  585          1772  4                      LOCAL
;  586          1773  4                          TEMP_DESC : $STR$DESCRIPTOR,
;  587          1774  4                          CHR_PTR,
;  588          1775  4                          CHARS_MOVED,
;  589          1776  4                          CHARS_LEFT;
;  590          1777  4
;  591          1778  4                      !+
;  592          1779  4                      ! Construct a dynamic string descriptor and try to
;  593          1780  4                      ! allocate some space to it.
;  594          1781  4                      !-
;  595          1782  4                      TEMP_DESC [DSC$W_LENGTH] = 0;
;  596          1783  4                      TEMP_DESC [DSC$B_DTYPE] = DEST_DESC [DSC$B_DTYPE] ;
;  597          1784  4                      TEMP_DESC [DSC$B_CLASS] = DSC$K_CLASS_D ;
;  598          1785  4                      TEMP_DESC [DSC$A_POINTER] = 0:
;  599        P 1786  4                      RETURN_STATUS = $STR$ALLOCATE (
;  600        P 1787  4                                                      MIN (MAX_SIZE, .TOTAL_LENGTH),
;  601          1788  4                                                      TEMP_DESC);
;  602          1789  4
;  603          1790  4                      !+
;  604          1791  4                      ! If the allocate did not succeed then don't proceed
;  605          1792  4                      ! with concatenate.
;  606          1793  4                      !-
;  607          1794  4
;  608          1795  4                      IF .RETURN_STATUS
;  609          1796  4                      THEN
;  610          1797  5                          BEGIN
;  611          1798  5                          !+
;  612          1799  5                          ! Init pointer to output area to first byte
;  613          1800  5                          ! allocated to temp descriptor.
;  614          1801  5                          !-
;  615          1802  5                          CHR_PTR = .TEMP_DESC [DSC$A_POINTER] ;
```

```
;  616     1803  5                CHARS_MOVED = 0;
;  617     1804  5                CHARS_LEFT = MIN (MAX_SIZE, .TOTAL_LENGTH);
;  618     1805  5
;  619     1806  5                INCR ARG_NO FROM FIRST_INPUT_ARG TO ACTUALCOUNT() DO
;  620     1807  6                    BEGIN
;  621     1808  6
;  622     1809  6                    LOCAL
;  623     1810  6                        IN_LEN,      ! length of Nth input string
;  624     1811  6                        IN_ADDR,     ! addr of 1st byte of Nth input
;  625     1812  6                                     ! string
;  626     1813  6                        SRC_DESC : REF $STR$DESCRIPTOR;
;  627     1814  6
;  628     1815  6                    SRC_DESC = ACTUALPARAMETER (.ARG_NO);
;  629     1816  6
;  630     1817  6                    !+
;  631     1818  6                    ! Extract length and address of this input
;  632     1819  6                    ! string.  There is no need to check status on
;  633     1820  6                    ! these calls.  If there was anything
;  634     1821  6                    ! wrong with the input descriptors, we would
;  635     1822  6                    ! have signaled our way out of the loop where
;  636     1823  6                    ! we added up the total lengths of the inputs.
;  637     1824  6                    !-
;  638     1825  6                    $STR$GET_LEN_ADDR (SRC_DESC, IN_LEN, IN_ADDR) ;
;  639     1826  6
;  640     1827  7                    IF (.CHARS_LEFT GTR 0)
;  641     1828  6                    THEN
;  642     1829  7                        BEGIN
;  643     1830  7
;  644     1831  7                        LOCAL
;  645     1832  7                            LEN;
;  646     1833  7
;  647     1834  7                        LEN = MIN (.IN_LEN, .CHARS_LEFT);
;  648     1835  7                        CHR_PTR = CH$MOVE (
;  649     1836  7                                          .LEN, .IN_ADDR, .CHR_PTR);
;  650     1837  7
;  651     1838  7                        CHARS_MOVED = .CHARS_MOVED + LEN;
;  652     1839  7                        CHARS_LEFT = .CHARS_LEFT - .LEN;
;  653     1840  6                        END;
;  654     1841  6
;  655     1842  5                    END;             ! concatenate into temp
;  656     1843  5
;  657     1844  5                !+
;  658     1845  5                ! Now exchange our temporary descriptor with the
;  659     1846  5                ! original destination descriptor, thus changing it
;  660     1847  5                ! from pointing to its old string to pointing to
;  661     1848  5                ! the concatenation.
;  662     1849  5                !-
;  663     1850  5                $STR$EXCH_DESCS (TEMP_DESC, DEST_DESC);
;  664     1851  5
;  665     1852  5                !+
;  666     1853  5                ! Now free the space which was described by the
;  667     1854  5                ! destination descriptor on entry to this routine,
;  668     1855  5                ! since the caller no longer has access to it.
;  669     1856  5                !-
;  670     1857  5                RETURN_STATUS = $STR$DEALLOCATE (TEMP_DESC);
;  671     1858  5
;  672     1859  4                END;                 ! concatenate into temp and
```

```
;  673     1860  4                                    ! exchange of temp and dest
;  674     1861  4
;  675     1862  4                        END                         ! of overlapped subcase
;  676     1863  4
;  677     1864  3                    ELSE
;  678     1865  3
;  679     1866  4                        BEGIN
;  680     1867  4                        !+
;  681     1868  4                        ! There is no overlap and the destination does not need
;  682     1869  4                        ! to be reallocated.  We can use the more efficient
;  683     1870  4                        ! algorithm of concatenating directly into the
;  684     1871  4                        ! destination string.
;  685     1872  4                        !-
;  686     1873  4
;  687     1874  4                        LOCAL
;  688     1875  4                            CHR_PTR;
;  689     1876  4
;  690     1877  4                        CHR_PTR = .DEST_DESC [DSC$A_POINTER];
;  691     1878  4
;  692     1879  4                        INCR ARG_NO FROM FIRST_INPUT_ARG TO ACTUALCOUNT () DO
;  693     1880  5                            BEGIN
;  694     1881  5
;  695     1882  5                            LOCAL
;  696     1883  5                                IN_LEN,         ! length of Nth input string
;  697     1884  5                                IN_ADDR,        ! addr of 1st byte of Nth input
;  698     1885  5                                                ! string
;  699     1886  5                                SRC_DESC : REF $STR$DESCRIPTOR;
;  700     1887  5
;  701     1888  5                            SRC_DESC = ACTUALPARAMETER (.ARG_NO);
;  702     1889  5
;  703     1890  5                            !+
;  704     1891  5                            ! Extract length and address of this input
;  705     1892  5                            ! string.  There is no need to check status on
;  706     1893  5                            ! these calls.  If there was anything
;  707     1894  5                            ! wrong with the input descriptors, we would
;  708     1895  5                            ! have signaled our way out of the loop where
;  709     1896  5                            ! we added up the total lengths of the inputs.
;  710     1897  5                            !-
;  711     1898  5                            $STR$GET_LEN_ADDR (SRC_DESC, IN_LEN, IN_ADDR) ;
;  712     1899  5
;  713     1900  5                            CHR_PTR = CH$MOVE ( .IN_LEN, .IN_ADDR, .CHR_PTR);
;  714     1901  5
;  715     1902  4                            END;        ! copy directly into destination
;  716     1903  4
;  717     1904  4                        !+
;  718     1905  4                        ! The destination descriptor may (if it is a "short
;  719     1906  4                        ! string") have been longer than the sum of the source
;  720     1907  4                        ! lengths.  If so, shorten it.
;  721     1908  4                        !-
;  722     1909  4                        DEST_DESC [DSC$W_LENGTH]= MIN (MAX_SIZE, .TOTAL_LENGTH);
;  723     1910  4
;  724     1911  3                        END;                ! of non-overlapped subcase
;  725     1912  3
;  726     1913  3                        !+
;  727     1914  3                        ! Record length of output string constructed for later
;  728     1915  3                        ! evaluation of what status to return.
;  729     1916  3                        !-
```

```
;  730          1917  3              RESULT_LENGTH = .DEST_DESC [DSC$W_LENGTH] ;
;  731          1918  3
;  732          1919  2          END;                ! of Class_D
```

```
;  734          1920  2          [DSC$K_CLASS_VS]:
;  735          1921  2      !+
;  736          1922  2      ! The destination string has varying-length string semantics.   Copy
;  737          1923  2      ! only as much of the sources into it as its DSC$W_MAXSTRLEN length
;  738          1924  2      ! allows.  If its storage overlaps any of the source strings, do the
;  739          1925  2      ! concatenation into a temporary string and then copy back to the
;  740          1926  2      ! destination string.
;  741          1927  2      ! If sum of source lengths is less than or equal to DSC$W_MAXSTRLEN,
;  742          1928  2      ! only its CURLEN field needs to be rewritten.  If sum of sources is
;  743          1929  2      ! greater than DSC$W_MAXSTRLEN field, STR$_TRU is returned.
;  744          1930  2      !-
;  745          1931  2
;  746          1932  3              BEGIN                           ! Class_VS
;  747          1933  3
;  748          1934  3              !+
;  749          1935  3              ! Real length of a Class_VS destination is contained in
;  750          1936  3              ! the MAXSTRLEN field.  Readjust our record of what can
;  751          1937  3              ! be written into.
;  752          1938  3              !-
;  753          1939  3              OUT_LEN = .DEST_DESC [DSC$W_MAXSTRLEN] ;
;  754          1940  3
;  755          1941  4              IF (.OVERLAP_FLAG)
;  756          1942  3              THEN
;  757          1943  4                  BEGIN
;  758          1944  4
;  759          1945  4                  LOCAL
;  760          1946  4                      CHR_PTR,            ! Variable pointer into output
;  761          1947  4                      TEMP_DESC : $STR$DESCRIPTOR;
;  762          1948  4
;  763          1949  4                  RETURN_STATUS =
;  764        P 1950  4                      $STR$ALLOC_TMP (MIN (MAX_SIZE, .TOTAL_LENGTH),
;  765          1951  4                                      TEMP_DESC);
;  766          1952  4
;  767          1953  4                  !+
;  768          1954  4                  ! If allocate didn't work, don't continue the
;  769          1955  4                  ! concatenate
;  770          1956  4                  !-
;  771          1957  4
;  772          1958  4                  IF .RETURN_STATUS
;  773          1959  4                  THEN
;  774          1960  5                      BEGIN
;  775          1961  5                      CHR_PTR = .TEMP_DESC [DSC$A_POINTER]; ! init to
;  776          1962  5                                                            ! start of
;  777          1963  5                                                            ! temp output
;  778          1964  5
;  779          1965  5                      INCR ARG_NO FROM FIRST_INPUT_ARG TO ACTUALCOUNT() DO
;  780          1966  6                          BEGIN   ! INCR copying loop
;  781          1967  6
;  782          1968  6                          LOCAL
;  783          1969  6                              IN_LEN,     ! length of Nth input string
;  784          1970  6                              IN_ADDR,    ! address of 1st byte of Nth
;  785          1971  6                                          ! input string
;  786          1972  6
;  787          1973  6                              SRC_DESC : REF $STR$DESCRIPTOR;
;  788          1974  6
;  789          1975  6                          !+
;  790          1976  6                          ! Get Nth input descriptor address
```

```
;  791          1977  6                    !-
;  792          1978  6                    SRC_DESC = ACTUALPARAMETER (.ARG_NO);
;  793          1979  6
;  794          1980  6                    !+
;  795          1981  6                    ! Extract length and address of this input
;  796          1982  6                    ! string.  There is no need to check status on
;  797          1983  6                    ! these calls.  If there was anything
;  798          1984  6                    ! wrong with the input descriptors, we would
;  799          1985  6                    ! have signaled our way out of the loop where
;  800          1986  6                    ! we added up the total lengths of the inputs.
;  801          1987  6                    !-
;  802          1988  6                    $STR$GET_LEN_ADDR (SRC_DESC, IN_LEN, IN_ADDR) ;
;  803          1989  6
;  804          1990  6                    CHR_PTR = CH$MOVE (.IN_LEN, .IN_ADDR, .CHR_PTR);
;  805          1991  5                    END;    ! INCR copying loop
;  806          1992  5
;  807          1993  5                !+
;  808          1994  5                ! Now copy from the temporary descriptor to the real
;  809          1995  5                ! destination.  The destination may be shorter than
;  810          1996  5                ! TOTAL_LENGTH, in which case fewer characters will
;  811          1997  5                ! be copied than were concatenated.
;  812          1998  5                !-
;  813          1999  5                CH$MOVE ( MIN (.DEST_DESC [DSC$W_MAXSTRLEN],
;  814          2000  5                               .TOTAL_LENGTH),
;  815          2001  5                          .TEMP_DESC [DSC$A_POINTER],
;  816          2002  5                          .OUT_ADDR);
;  817          2003  5
;  818          2004  5                RETURN_STATUS = $STR$DEALOC_TMP (TEMP_DESC);
;  819          2005  5
;  820          2006  4                END;        ! of concatenation and copy via temp
;  821          2007  4
;  822          2008  4                    !+
;  823          2009  4                    ! Record actual size of output string written for
;  824          2010  4                    ! later evaluation of what status to return.
;  825          2011  4                    !-
;  826          2012  4                    RESULT_LENGTH = MIN ( .DEST_DESC [DSC$W_MAXSTRLEN],
;  827          2013  4                                          .TOTAL_LENGTH) ;
;  828          2014  4
;  829          2015  4                END             ! of overlap subcase
;  830          2016  4
;  831          2017  3            ELSE
;  832          2018  3
;  833          2019  4                BEGIN
;  834          2020  4                !+
;  835          2021  4                ! This is the case of a varying_length string
;  836          2022  4                ! destination which does not overlap any of the sources.
;  837          2023  4                ! We can copy directly into the destination space.
;  838          2024  4                !-
;  839          2025  4
;  840          2026  4                LOCAL
;  841          2027  4                    CHR_PTR,
;  842          2028  4                    CHARS_MOVED,
;  843          2029  4                    ARG_NO;
;  844          2030  4
;  845          2031  4                CHR_PTR = .OUT_ADDR;    ! init to 1st byte of dest
;  846          2032  4                CHARS_MOVED = 0;
;  847          2033  4                ARG_NO = FIRST_INPUT_ARG;
```

```
;  848          2034  4
;  849          2035  4          WHILE (.CHARS_MOVED NEQ .OUT_LEN) DO
;  850          2036  5              BEGIN
;  851          2037  5              !+
;  852          2038  5              ! There is room for more characters in the
;  853          2039  5              ! destination string.  Copy as much of the next
;  854          2040  5              ! input string as will fit.
;  855          2041  5              !-
;  856          2042  5
;  857          2043  5              LOCAL
;  858          2044  5                  IN_LEN,          ! length of Nth input string
;  859          2045  5                  IN_ADDR,         ! address of 1st byte of Nth
;  860          2046  5                                   ! input string
;  861          2047  5                  CHARS_LEFT;
;  862          2048  5
;  863          2049  5              CHARS_LEFT = .OUT_LEN - .CHARS_MOVED;
;  864          2050  5
;  865          2051  6              IF (.ARG_NO GTR ACTUALCOUNT ())
;  866          2052  5              THEN
;  867          2053  6                  BEGIN
;  868          2054  6                  !+
;  869          2055  6                  ! We have exhausted the parameters, fill the
;  870          2056  6                  ! remainder of the destination string with
;  871          2057  6                  ! blanks.
;  872          2058  6                  !-
;  873          2059  6                  EXITLOOP ;
;  874          2060  6                  END
;  875          2061  6
;  876          2062  5              ELSE
;  877          2063  5
;  878          2064  6                  BEGIN           ! copy of one more string
;  879          2065  6                  !+
;  880          2066  6                  ! We have another input string.  Copy it into
;  881          2067  6                  ! the destination string, or as much of it as
;  882          2068  6                  ! will fit.
;  883          2069  6                  !-
;  884          2070  6
;  885          2071  6                  LOCAL
;  886          2072  6                      SRC_DESC : REF $STR$DESCRIPTOR;
;  887          2073  6
;  888          2074  6                  SRC_DESC = ACTUALPARAMETER (.ARG_NO);
;  889          2075  6                  !+
;  890          2076  6                  ! Extract length and address of this input
;  891          2077  6                  ! string.  There is no need to check status on
;  892          2078  6                  ! these calls.  If there was anything
;  893          2079  6                  ! wrong with the input descriptors, we would
;  894          2080  6                  ! have signaled our way out of the loop where
;  895          2081  6                  ! we added up the total lengths of the inputs.
;  896          2082  6                  !-
;  897          2083  6                  $STR$GET_LEN_ADDR (SRC_DESC, IN_LEN, IN_ADDR) ;
;  898          2084  6
;  899          2085  6                  CHR_PTR = CH$MOVE ( MIN (.IN_LEN, .CHARS_LEFT),
;  900          2086  6                                      .IN_ADDR, .CHR_PTR);
;  901          2087  6
;  902          2088  6                  CHARS_MOVED = .CHARS_MOVED +
;  903          2089  6                                MIN (.IN_LEN, .CHARS_LEFT);
;  904          2090  6
```

```
;   905          2091  6                    ARG_NO = .ARG_NO + 1;
;   906          2092  5                    END;                    ! copy of one more string
;   907          2093  5
;   908          2094  4                END;                        ! of WHILE loop
;   909          2095  4
;   910          2096  4                !+
;   911          2097  4                ! Record actual length of output string written for
;   912          2098  4                ! later evaluation of status to be returned.
;   913          2099  4                !-
;   914          2100  4                RESULT_LENGTH = .CHARS_MOVED ;
;   915          2101  4
;   916          2102  3            END;                            ! of non-overlapped
;   917          2103  3                                            ! concatenation operation
;   918          2104  3        !+
;   919          2105  3        ! Adjust CURLEN field to reflect the new size of the
;   920          2106  3        ! varying string.
;   921          2107  3        !-
;   922          2108  3        (.DEST_DESC [DSC$A_POINTER])<0,16> = .RESULT_LENGTH ;
;   923          2109  3
;   924          2110  2        END;                                ! of Class_VS
```

```
;  926     2111  2          [INRANGE, OUTRANGE] :
;  927     2112  2      !+
;  928     2113  2      ! The class of the destination string is unknown.  Will cause an error
;  929     2114  2      ! to be signaled.
;  930     2115  2      !-
;  931     2116  2              RETURN_STATUS = STR$_ILLSTRCLA;
;  932     2117  2
;  933     2118  2          TES;
;  934     2119  2
;  935     2120  2      !+
;  936     2121  2      ! If any of the allocations or deallocations previously failed, or
;  937     2122  2      ! illegal string class was found then signal the error.
;  938     2123  2      !-
;  939     2124  2          $STR$SIGNAL_FATAL (RETURN_STATUS);  ! Signal fatal error
;  940     2125  2
;  941     2126  2          IF .RESULT_CLASS EQL DSC$K_CLASS_D
;  942     2127  2          THEN
;  943     2128  3              BEGIN           ! special processing for dynamic semantics
;  944     2129  3
;  945     2130  3              IF (.RESULT_LENGTH NEQ .TOTAL_LENGTH) THEN
;  946     2131  3                  LIB$STOP (STR$_STRTOOLON);
;  947     2132  3
;  948     2133  3              RETURN (STR$_NORMAL);           ! used because bliss compiler
;  949     2134  3                                              !  doesn't understand routines
;  950     2135  3                                              !  that don't return
;  951     2136  3
;  952     2137  3              END             ! special processing for dynamic semantics
;  953     2138  3
;  954     2139  2          ELSE                ! special processing for fixed and varying
;  955     2140  2                              ! string semantics
;  956     2141  4              RETURN (IF (.RESULT_LENGTH GEQ .TOTAL_LENGTH)
;  957     2142  3                  THEN
;  958     2143  3                      SS$_NORMAL
;  959     2144  3                  ELSE
;  960     2145  2                      STR$_TRU  );
;  961     2146  2
;  962     2147  1          END;                                ! End of STR$CONCAT


                                                        .TITLE  STR$CONCAT
                                                        .IDENT  \1-017\

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

00  00  54  41  43  4E  4F  43  24  52  54  53  00000 P.AAA:  .ASCII  \STR$CONCAT\<0><0>                    ;

                                                        .EXTRN  LIB$STOP, STR$_NORMAL
                                                        .EXTRN  STR$_STRIS_INT, STR$_ILLSTRCLA
                                                        .EXTRN  STR$_TRU, STR$_FATINTERR
                                                        .EXTRN  STR$_STRTOOLON, STR$_WRONUMARG
                                                        .EXTRN  STR$ANALYZE_SDESC_R1
                                                        .EXTRN  STR$$INIT, STR$$V_INIT
                                                        .EXTRN  STR$$ALOC_SHORT
                                                        .EXTRN  STR$$Q_SHORT_Q, LIB$GET_VM
                                                        .EXTRN  STR$_INSVIRMEM, LIB$FREE_VM
                                                        .EXTRN  STR$$MOVQ_R1
```

```
                                0FFC 00000          .ENTRY  STR$CONCAT, Save R2,R3,R4,R5,R6,R7,R8,R9,-      ; 1340
                                                            R10,R11
                          5E      28  C2 00002        SUBL2   #40, SP
                          02      6C  91 00005        CMPB    (AP), #2                                        ; 1487
                                  22  1E 00008        BGEQU   1$
                    20    AE 010E000A   8F  D0 0000A        MOVL    #17694730, ROUT_NAME_DESC                 ; 1497
                    24    AE        DF  AF  9E 00012        MOVAB   P.AAA, ROUT_NAME_DESC+4                   ; 1500
                                    20  AE  9F 00017        PUSHAB  ROUT_NAME_DESC                            ; 1501
                          7E            6C  9A 0001A        MOVZBL  (AP), -(SP)
                                        02  DD 0001D        PUSHL   #2
                                00000000G   8F  DD 0001F        PUSHL   #STR$_WRONUMARG
                00000000G       00          04  FB 00025        CALLS   #4, LIB$STOP
                          08    AE            01  D0 0002C 1$:  MOVL    #1, RETURN_STATUS                   ; 1504
                          57            04    AC  D0 00030        MOVL    DEST_DESC, R7                       ; 1509
                          02            03    A7  91 00034        CMPB    3(R7), #2
                                              0A  1A 00038        BGTRU   2$
                          04    AE            67  3C 0003A        MOVZWL  (R7), OUT_LEN
                          6E            04    A7  D0 0003E        MOVL    4(R7), OUT_ADDR
                                              10  11 00042        BRB     3$
                          50                  57  D0 00044 2$:  MOVL    R7, R0
                                00000000G     00  16 00047        JSB     STR$ANALYZE_SDESC_R1
                          04    AE            50  D0 0004D        MOVL    R0, 4(SP)
                          6E                  51  D0 00051        MOVL    R1, (SP)
                                              56  D4 00054 3$:  CLRL    TOTAL_LENGTH                        ; 1522
                          55                  6C  9A 00056        MOVZBL  (AP), R5                            ; 1527
                          52                  01  7D 00059        MOVQ    #1, ARG_NO                          ; 1550
                                              39  11 0005C        BRB     9$
                          50                6C42  D0 0005E 4$:  MOVL    (AP)[ARG_NO], SRC_DESC              ; 1536
                          02            03    A0  91 00062        CMPB    3(SRC_DESC), #2                     ; 1541
                                              09  1A 00066        BGTRU   5$
                          54                  60  3C 00068        MOVZWL  (SRC_DESC), IN_LEN
                          51            04    A0  D0 0006B        MOVL    4(SRC_DESC), IN_ADDR
                                              09  11 0006F        BRB     6$
                                00000000G     00  16 00071 5$:  JSB     STR$ANALYZE_SDESC_R1
                          54                  50  D0 00077        MOVL    R0, R4
                          56                  54  C0 0007A 6$:  ADDL2   IN_LEN, TOTAL_LENGTH                ; 1543
                          6E                  51  D1 0007D        CMPL    IN_ADDR, OUT_ADDR                   ; 1550
                                              09  1E 00080        BGEQU   7$
                    50                    51  54  C1 00082        ADDL3   IN_LEN, IN_ADDR, R0
                          50                  6E  D1 00086        CMPL    OUT_ADDR, R0
                                              07  11 00089        BRB     8$
                    50                    6E  56  C1 0008B 7$:  ADDL3   TOTAL_LENGTH, OUT_ADDR, R0
                          50                  51  D1 0008F        CMPL    IN_ADDR, R0
                                              03  18 00092 8$:  BGEQ    9$
                          53                  01  D0 00094        MOVL    #1, OVERLAP_FLAG                    ; 1552
                    C3    52                  55  F3 00097 9$:  AOBLEQ  R5, ARG_NO, 4$                     ; 1527
                                10    AE      03  A7  9A 0009B        MOVZBL  3(R7), RESULT_CLASS                 ; 1560
                    0F          00    10    AE  CF  00A0        CASEL   RESULT_CLASS, #0, #15                   ; 1562
0020      01ED            002B            002B        00A5 10$:  .WORD   12$-10$,-
0020      0020            0020            002B        00AD               12$-10$,-
0458      002B            002B            0020        00B5               43$-10$,-
002B      0020            0020            0020        00BD               11$-10$,-
                                                                         12$-10$,-
                                                                         11$-10$,-
                                                                         11$-10$,-
                                                                         11$-10$,-
                                                                         11$-10$,-
```

```
                                                                                                12$-10$,-
                                                                                                12$-10$,-
                                                                                                85$-10$,-
                                                                                                11$-10$,-
                                                                                                11$-10$,-
                                                                                                11$-10$,-
                                                                                                12$-10$
        08    AE 00000000G   8F   D0 000C5 11$:     MOVL    #STR$_ILLSTRCLA, RETURN_STATUS                  2116
                       05E7   31   000CD             BRW     116$
              03              53   E8 000D0 12$:     BLBS    OVERLAP_FLAG, 13$                               1581
                       015B   31   000D3             BRW     35$
              07 00000000G    00   E8 000D6 13$:     BLBS    STR$$V_INIT, 14$                                1591
00000000G     00              00   FB 000DD         CALLS   #0, STR$$INIT
              50 00000000G    8F   D0 000E4 14$:     MOVL    #STR$_NORMAL, RETURN_STATUS
              52              56   D0 000EB         MOVL    TOTAL_LENGTH, R2
0000FFFF      8F              52   D1 000EE         CMPL    R2, #65535
                              05   15 000F5         BLEQ    15$
              52      FFFF    8F   3C 000F7         MOVZWL  #65535, R2
000000F0      8F              52   D1 000FC 15$:     CMPL    R2, #240
                              61   1A 00103         BGTRU   23$
                              52   D5 00105         TSTL    R2
                              04   12 00107         BNEQ    16$
                              53   D4 00109         CLRL    TEMP
                              3B   11 0010B         BRB     21$
              51        FF    A2   9E 0010D 16$:     MOVAB   -1(R2), R1
              51              07   8A 00111         BICB2   #7, R1
              54 00000000G0041   9E 00114         MOVAB   STR$$Q_SHORT_Q[R1], REMQUE_ADDR
              53        00    B4   0F 0011C 17$:     REMQUE  @0(REMQUE_ADDR), TEMP
                              05   1D 00120         BVS     18$
              52              01   D0 00122         MOVL    #1, ALLOC_DONE
                              19   11 00125         BRB     20$
                              52   D4 00127 18$:     CLRL    ALLOC_DONE
                              56   DD 00129         PUSHL   TOTAL_LENGTH
0000FFFF      8F              6E   D1 0012B         CMPL    (SP), #65535
                              05   15 00132         BLEQ    19$
              6E      FFFF    8F   3C 00134         MOVZWL  #65535, (SP)
00000000G     00              01   FB 00139 19$:     CALLS   #1, STR$$ALOC_SHORT
              05              52   E8 00140 20$:     BLBS    ALLOC_DONE, 21$
              41              50   E9 00143         BLBC    RETURN_STATUS, 25$
                              D4   11 00146         BRB     17$
              3C              50   E9 00148 21$:     BLBC    RETURN_STATUS, 25$
        24    AE              53   D0 0014B         MOVL    TEMP, TEMP_DESC+4
              51              56   D0 0014F         MOVL    TOTAL_LENGTH, R1
0000FFFF      8F              51   D1 00152         CMPL    R1, #65535
                              05   15 00159         BLEQ    22$
              51      FFFF    8F   3C 0015B         MOVZWL  #65535, R1
        20    AE              51   B0 00160 22$:     MOVW    R1, TEMP_DESC
                              21   11 00164         BRB     25$
                        24    AE   9F 00166 23$:     PUSHAB  TEMP_DESC+4
        10    AE              52   D0 00169         MOVL    R2, T6(SP)
                        10    AE   9F 0016D         PUSHAB  16(SP)
00000000G     00              02   FB 00170         CALLS   #2, LIB$GET_VM
              09              50   E8 00177         BLBS    RETURN_STATUS, 24$
              50 00000000G    8F   D0 0017A         MOVL    #STR$_INSVIRMEM, RETURN_STATUS
                              04   11 00181         BRB     25$
        20    AE              52   B0 00183 24$:     MOVW    R2, TEMP_DESC
        08    AE              50   D0 00187 25$:     MOVL    RETURN_STATUS, RETURN_STATUS
```

```
                                03        08    AE   E8 0018B          BLBS    RETURN_STATUS, 26$                                  ; 1598
                                        0099    31   0018F          BRW     34$
                                59        24    AE   D0 00192 26$:    MOVL    TEMP_DESC+4, R9                                     ; 1601
                                53              59   D0 00196          MOVL    R9, CHR_PTR
                                5A              6C   9A 00199          MOVZBL  (AP), R10                                           ; 1605
                                58              01   D0 0019C          MOVL    #1, ARG_NO
                                                20   11 0019F          BRB     30$
                                50            6C48   D0 001A1 27$:    MOVL    (AP)[ARG_NO], SRC_DESC                              ; 1618
                                02        03    A0   91 001A5          CMPB    3(SRC_DESC), #2                                     ; 1628
                                                09   1A 001A9          BGTRU   28$
                                52              60   3C 001AB          MOVZWL  (SRC_DESC), IN_LEN
                                51        04    A0   D0 001AE          MOVL    4(SRC_DESC), IN_ADDR
                                                09   11 001B2          BRB     29$
                             00000000G          00   16 001B4 28$:    JSB     STR$ANALYZE_SDESC_R1
                                52              50   D0 001BA          MOVL    R0, R2
                          63                    61       52 28 001BD 29$:    MOVC3   IN_LEN, (IN_ADDR), (CHR_PTR)                ; 1630
                          DC                    58       5A F3 001C1 30$:    AOBLEQ  R10, ARG_NO, 27$                            ; 1605
                                51              56   D0 001C5          MOVL    TOTAL_LENGTH, R1                                    ; 1641
                             0000FFFF           8F              51   D1 001C8          CMPL    R1, #65535
                                                05   15 001CF          BLEQ    31$
                                51        FFFF    8F   3C 001D1          MOVZWL  #65535, R1
                04    AE                  20        69       51 2C 001D6 31$:    MOVC5   R1, (R9), #32, OUT_LEN, @OUT_ADDR           ; 1645
                                              00    BE       001DC
                                50 00000000G    8F   D0 001DE          MOVL    #STR$_NORMAL, RETURN_STATUS                         ; 1647
                                                59   D5 001E5          TSTL    R9
                                                3E   13 001E7          BEQL    33$
                                      00F0      8F        20    AE   B1 001E9          CMPW    TEMP_DESC, #240
                                                1A   1A 001EF          BGTRU   32$
                                51              59   D0 001F1          MOVL    R9, STRING_BLOCK
                                51        FE    A1   3C 001F4          MOVZWL  -2(STRING_BLOCK), ALLOC_LENGTH
                                51              51   D7 001F8          DECL    R1
                                51              07   8A 001FA          BICB2   #7, R1
                                51 00000000G0041   9E 001FD          MOVAB   STR$$Q_SHORT_Q[R1], INSQUE_ADDR
                             00    B1                 69   0E 00205          INSQUE  (R9), @0(INSQUE_ADDR)
                                                1C   11 00209          BRB     33$
                                           24    AE   9F 0020B 32$:    PUSHAB  TEMP_DESC+4
                                10    AE        24    AE   3C 0020E          MOVZWL  TEMP_DESC, 16(SP)
                                           10    AE   9F 00213          PUSHAB  16(SP)
                          00000000G 00              02   FB 00216          CALLS   #2, LIB$FREE_VM
                                07              50   E8 0021D          BLBS    RETURN_STATUS, 33$
                                50 00000000G    8F   D0 00220          MOVL    #STR$_FATINTERR, RETURN_STATUS
                                08    AE              50   D0 00227 33$:    MOVL    RETURN_STATUS, RETURN_STATUS
                                5A        04    AE   D0 0022B 34$:    MOVL    OUT_LEN, RESULT_LENGTH                              ; 1654
                                                5E   11 0022F          BRB     42$                                                 ; 1581
                                0C    AE              6E   D0 00231 35$:    MOVL    OUT_ADDR, CHR_PTR                           ; 1670
                                                5B   D4 00235          CLRL    CHARS_MOVED                                         ; 1671
                                58              02   D0 00237          MOVL    #2, ARG_NO                                          ; 1672
                                04    AE              5B   D1 0023A 36$:    CMPL    CHARS_MOVED, OUT_LEN                        ; 1674
                                                4C   13 0023E          BEQL    41$
                          5A    04    AE              5B   C3 00240          SUBL3   CHARS_MOVED, OUT_LEN, CHARS_LEFT            ; 1688
                58        6C                    08       00 ED 00245          CMPZV   #0, #8, (AP), ARG_NO                        ; 1690
                                                0C   18 0024A          BGEQ    37$
                5A        20                    6E       00 2C 0024C          MOVC5   #0, (SP), #32, CHARS_LEFT, @CHR_PTR         ; 1698
                                           0C    BE       00251
                                5B              5A   C0 00253          ADDL2   CHARS_LEFT, CHARS_MOVED                             ; 1699
                                                E2   11 00256          BRB     36$                                                 ; 1690
                                50            6C48   D0 00258 37$:    MOVL    (AP)[ARG_NO], SRC_DESC                              ; 1714
```

```
                                02        03    A0  91 0025C          CMPB    3(SRC_DESC), #2                          ; 1723
                                                09  1A 00260          BGTRU   38$
                                59              60  3C 00262          MOVZWL  (SRC_DESC), IN_LEN
                                51        04    A0  D0 00265          MOVL    4(SRC_DESC), IN_ADDR
                                                09  11 00269          BRB     39$
                                    00000000G   00  16 0026B 38$:      JSB     STR$ANALYZE_SDESC_R1
                                59              50  D0 00271          MOVL    R0, R9
                                5A              59  D1 00274 39$:     CMPL    R9, CHARS_LEFT                           ; 1725
                                                03  15 00277          BLEQ    40$
                                59              5A  D0 00279          MOVL    CHARS_LEFT, R9
                        0C  BE  61              59  28 0027C 40$:     MOVC3   R9, (IN_ADDR), @CHR_PTR                  ; 1726
                            0C  AE              53  D0 00281          MOVL    R3, CHR_PTR
                                5B              59  C0 00285          ADDL2   R9, CHARS_MOVED                          ; 1729
                                                58  D6 00288          INCL    ARG_NO                                   ; 1731
                                                AE  11 0028A          BRB     36$                                      ; 1674
                                5A              5B  D0 0028C 41$:     MOVL    CHARS_MOVED, RESULT_LENGTH               ; 1740
                                            0425  31 0028F 42$:     BRW     116$                                     ; 1562
                                5B              56  D0 00292 43$:     MOVL    TOTAL_LENGTH, R11                        ; 1788
                        0000FFFF    8F          5B  D1 00295          CMPL    R11, #65535
                                                05  15 0029C          BLEQ    44$
                                5B      FFFF    8F  3C 0029E          MOVZWL  #65535, R11
                                5C              53  E8 002A3 44$:     BLBS    OVERLAP_FLAG, 52$                        ; 1763
                                51              56  D0 002A6          MOVL    TOTAL_LENGTH, R1                         ; 1766
                        0000FFFF    8F          51  D1 002A9          CMPL    R1, #65535
                                                05  15 002B0          BLEQ    45$
                                51      FFFF    8F  3C 002B2          MOVZWL  #65535, R1
                                52        04    A7  D0 002B7 45$:     MOVL    4(R7), R2
                                                53  D4 002BB          CLRL    R3
                                                52  D5 002BD          TSTL    R2
                                                06  12 002BF          BNEQ    46$
                                                53  D6 002C1          INCL    R3
                                                50  D4 002C3          CLRL    R0
                                                13  11 002C5          BRB     48$
                            00F0    8F          67  B1 002C7 46$:     CMPW    (R7), #240
                                                05  1B 002CC          BLEQU   47$
                                50              67  3C 002CE          MOVZWL  (R7), R0
                                                07  11 002D1          BRB     48$
                                50              52  D0 002D3 47$:     MOVL    R2, STRING_BLOCK
                                50        FE    A0  3C 002D6          MOVZWL  -2(STRING_BLOCK), R0
                        000000F0    8F          50  D1 002DA 48$:     CMPL    R0, #240
                                                21  1F 002E1          BLSSU   53$
                                04              53  E9 002E3          BLBC    R3, 49$
                                                50  D4 002E6          CLRL    R0
                                                13  11 002E8          BRB     51$
                            00F0    8F          67  B1 002EA 49$:     CMPW    (R7), #240
                                                05  1B 002EF          BLEQU   50$
                                50              67  3C 002F1          MOVZWL  (R7), R0
                                                07  11 002F4          BRB     51$
                                50              52  D0 002F6 50$:     MOVL    R2, STRING_BLOCK
                                50        FE    A0  3C 002F9          MOVZWL  -2(STRING_BLOCK), R0
                                50              51  D1 002FD 51$:     CMPL    R1, R0
                                                21  13 00300          BEQL    57$
                                                2B  11 00302 52$:     BRB     58$
                                04              53  E9 00304 53$:     BLBC    R3, 54$
                                                50  D4 00307          CLRL    R0
                                                13  11 00309          BRB     56$
                            00F0    8F          67  B1 0030B 54$:     CMPW    (R7), #240
```

```
                                    05  1B 00310          BLEQU   55$
                               50   67  3C 00312          MOVZWL  (R7), R0
                                    07  11 00315          BRB     56$
                               50   52  D0 00317 55$:     MOVL    R2, STRING_BLOCK
                               50   FE  A0 3C 0031A       MOVZWL  -2(STRING_BLOCK), R0
                               50   51  D1 0031E 56$:     CMPL    R1, R0
                                    0C  1A 00321          BGTRU   58$
                    0000FFFF   8F   56  D1 00323 57$:     CMPL    TOTAL_LENGTH, #65535                    1768
                                    03  14 0032A          BGTR    58$
                                0194    31 0032C          BRW     79$
                                20  AE  B4 0032F 58$:     CLRW    TEMP_DESC                               1782
                22  AE          57  02  81 00332          ADDB3   #2, R7, TEMP_DESC+2                     1783
                        23  AE      02  90 00337          MOVB    #2, TEMP_DESC+3                         1784
                                24  AE  D4 0033B          CLRL    TEMP_DESC+4                             1785
                            07 00000000G    00  E8 0033E          BLBS    STR$$V_INIT, 59$                1788
                    00000000G   00      00  FB 00345          CALLS   #0, STR$$INIT
                               50 00000000G 8F  D0 0034C 59$:     MOVL    #STR$_NORMAL, RETURN_STATUS
                    000000F0   8F   5B  D1 00353          CMPL    R11, #240
                                    61  1A 0035A          BGTRU   67$
                                    5B  D5 0035C          TSTL    R11
                                    04  12 0035E          BNEQ    60$
                                    53  D4 00360          CLRL    TEMP
                                    3B  11 00362          BRB     65$
                               51   FF  AB  9E 00364 60$:     MOVAB   -1(R11), R1
                               51       07  8A 00368          BICB2   #7, R1
                               54 00000000G0041    9E 0036B          MOVAB   STR$$Q_SHORT_Q[R1], REMQUE_ADDR
                               53   00  B4  0F 00373 61$:     REMQUE  @0(REMQUE_ADDR), TEMP
                                    05  1D 00377          BVS     62$
                               52       01  D0 00379          MOVL    #1, ALLOC_DONE
                                    19  11 0037C          BRB     64$
                                    52  D4 0037E 62$:     CLRL    ALLOC_DONE
                                    56  DD 00380          PUSHL   TOTAL_LENGTH
                    C000FFFF   8F   6E  D1 00382          CMPL    (SP), #65535
                                    05  15 00389          BLEQ    63$
                                6E      FFFF    8F  3C 0038B          MOVZWL  #65535, (SP)
                    00000000G   00      01  FB 00390 63$:     CALLS   #1, STR$$ALOC_SHORT
                                05      52  E8 00397 64$:     BLBS    ALLOC_DONE, 65$
                                41      50  E9 0039A          BLBC    RETURN_STATUS, 69$
                                    D4  11 0039D          BRB     61$
                                3C      50  E9 0039F 65$:     BLBC    RETURN_STATUS, 69$
                            24  AE      53  D0 003A2          MOVL    TEMP, TEMP_DESC+4
                                51      56  D0 003A6          MOVL    TOTAL_LENGTH, R1
                    0000FFFF   8F       5   D1 003A9          CMPL    R1, #65535
                                    05  15 003B0          BLEQ    66$
                                51      FFFF    8F  3C 003B2          MOVZWL  #65535, R1
                            20  AE          51  B0 003B7 66$:     MOVW    R1, TEMP_DESC
                                    21  11 003BB          BRB     69$
                                    24  AE  9F 003BD 67$:     PUSHAB  TEMP_DESC+4
                            10  AE          5B  D0 003C0          MOVL    R11, 16(SP)
                                    10  AE  9F 003C4          PUSHAB  16(SP)
                    00000000G   00      02  FB 003C7          CALLS   #2, LIB$GET_VM
                                09      50  E8 003CE          BLBS    RETURN_STATUS, 68$
                                50 00000000G    8F  D0 003D1          MOVL    #STR$_INSVIRMEM, RETURN_STATUS
                                    04  11 003D8          BRB     69$
                            20  AE          5B  B0 003DA 68$:     MOVW    R11, TEMP_DESC
                            08  AE          50  D0 003DE 69$:     MOVL    RETURN_STATUS, RETURN_STATUS
                                03      08  AE  E8 003E2          BLBS    RETURN_STATUS, 70$              1795
```

```
                          010D  31 003E6          BRW     84$
                    53      24  AE  D0 003E9 70$:  MOVL    TEMP_DESC+4, CHR_PTR                    ; 1802
                                5B  D4 003ED       CLRL    CHARS_MOVED                             ; 1803
                    51          56  D0 003EF       MOVL    TOTAL_LENGTH, R1                        ; 1804
              0000FFFF  8F          51  D1 003F2       CMPL    R1, #65535
                                    05  15 003F9       BLEQ    71$
                    51      FFFF  8F  3C 003FB       MOVZWL  #65535, R1
                    58          51  D0 00400 71$:    MOVL    R1, CHARS_LEFT
                    57          6C  9A 00403       MOVZBL  (AP), R7                                ; 1806
                    59          01  D0 00406       MOVL    #1, ARG_NO
                                3D  11 00409       BRB     76$
                    50        6C49  D0 0040B 72$:  MOVL    (AP)[ARG_NO], SRC_DESC                  ; 1815
                    02      03  A0  91 0040F       CMPB    3(SRC_DESC), #2                         ; 1825
                                09  1A 00413       BGTRU   73$
                    52          60  3C 00415       MOVZWL  (SRC_DESC), IN_LEN
                    51      04  A0  D0 00418       MOVL    4(SRC_DESC), IN_ADDR
                                09  11 0041C       BRB     74$
                         00000000G  00  16 0041E 73$:  JSB     STR$ANALYZE_SDESC_R1
                    52          50  D0 00424       MOVL    R0, R2
                                58  D5 00427 74$:  TSTL    CHARS_LEFT                              ; 1827
                                1D  15 00429       BLEQ    76$
                    50          52  D0 0042B       MOVL    IN_LEN, R0                              ; 1834
                    58          50  D1 0042E       CMPL    R0, CHARS_LEFT
                                03  15 00431       BLEQ    75$
                    50          58  D0 00433       MOVL    CHARS_LEFT, R0
                 14 AE          50  D0 00436 75$:  MOVL    R0, LEN
              63    61      14  AC  28 0043A       MOVC3   LEN, (IN_ADDR), (CHR_PTR)               ; 1836
                    5B      14 AE4B  9E 0043F       MOVAB   LEN[CHARS_MOVED], CHARS_MOVED           ; 1838
                    58      14  AE  C2 00444       SUBL2   LEN, CHARS_LEFT                         ; 1839
              BF    59          57  F3 00448 76$:  AOBLEQ  R7, ARG_NO, 72$                         ; 1806
                    51      04  AC  D0 0044C       MOVL    DEST_DESC, R1                           ; 1850
                 18 AE          61  B0 00450       MOVW    (R1), $STR$TEMP_DESC
                 1C AE      04  A1  D0 00454       MOVL    4(R1), $STR$TEMP_DESC+4
                 22 AE      02  A1  B0 00459       MOVW    2(R1), TEMP_DESC+2
                    50      20  AE  9E 0045E       MOVAB   TEMP_DESC, R0
                         00000000G  00  16 00462       JSB     STR$$MOVQ_R1
                 20 AE      18  AE  B0 00468       MOVW    $STR$TEMP_DESC, TEMP_DESC
                 24 AE      1C  AE  D0 0046D       MOVL    $STR$TEMP_DESC+4, TEMP_DESC+4
                    50 00000000G  8F  D0 00472       MOVL    #STR$_NORMAL, RETURN_STATUS             ; 1857
                    52      24  AE  D0 00479       MOVL    TEMP_DESC+4, R2
                                3E  13 0047D       BEQL    78$
                00F0  8F        20  AE  B1 0047F       CMPW    TEMP_DESC, #240
                                1A  1A 00485       BGTRU   77$
                    51          52  D0 00487       MOVL    R2, STRING_BLOCK
                    51      FE  A1  3C 0048A       MOVZWL  -2(STRING_BLOCK), ALLOC_LENGTH
                                51  D7 0048E       DECL    R1
                    51          07  8A 00490       BICB2   #7, R1
                    51 00000000G0041  9E 00493       MOVAB   STR$$Q_SHORT_Q[R1], INSQUE_ADDR
                 00 B1          62  0E 0049B       INSQUE  (R2), @0(INSQUE_ADDR)
                                1C  11 0049F       BRB     78$
                        24  AE  9F 004A1 77$:  PUSHAB  TEMP_DESC+4
                 10 AE      24  AE  3C 004A4       MOVZWL  TEMP_DESC, 16(SP)
                        10  AE  9F 004A9       PUSHAB  16(SP)
          00000000G  00          02  FB 004AC       CALLS   #2, LIB$FREE_VM
                    07          50  E8 004B3       BLBS    RETURN_STATUS, 78$
                    50 00000000G  8F  D0 004B6       MOVL    #STR$_FATINTERR, RETURN_STATUS
                 08 AE          50  D0 004BD 78$:  MOVL    RETURN_STATUS, RETURN_STATUS
```

```
                                    33  11 004C1          BRB     84$                                     ; 1763
                              53    04  A7  DO 004C3 79$:  MOVL    4(R7), CHR_PTR                          ; 1877
                              59        6C  9A 004C7      MOVZBL  (AP), R9                                ; 1879
                              58        01  DO 004CA      MOVL    #1, ARG_NO
                                        20  11 004CD      BRB     83$
                              50      6C48  DO 004CF 80$:  MOVL    (AP)[ARG_NO], SRC_DESC                  ; 1888
                              02    03  A0  91 004D3      CMPB    3(SRC_DESC), #2                         ; 1898
                                        09  1A 004D7      BGTRU   81$
                              52        60  3C 004D9      MOVZWL  (SRC_DESC), IN_LEN
                              51    04  A0  DO 004DC      MOVL    4(SRC_DESC), IN_ADDR
                                        09  11 004E0      BRB     82$
                         00000000G      00  16 004E2 81$:  JSB     STR$ANALYZE_SDESC_R1
                              52        50  DO 004E8      MOVL    R0, R2
                        63    61        52  28 004EB 82$:  MOVC3   IN_LEN, (IN_ADDR), (CHR_PTR)            ; 1900
                        DC    58        59  F3 004EF 83$:  AOBLEQ  R9, ARG_NO, 80$                        ; 1879
                              67        5B  BO 004F3      MOVW    R11, (R7)                               ; 1909
                              5A    04  BC  3C 004F6 84$:  MOVZWL  @DEST_DESC, RESULT_LENGTH               ; 1917
                                    01BA  31 004FA      BRW     116$                                    ; 1562
                        04    AE        67  3C 004FD 85$:  MOVZWL  (R7), OUT_LEN                        ; 1939
                              03        53  E8 00501      BLBS    OVERLAP_FLAG, 86$                       ; 1941
                                    015E  31 00504      BRW     109$
                              07 00000000G  00  E8 00507 86$:  BLBS    STR$$V_INIT, 87$                    ; 1951
                 00000000G    00        00  FB 0050E      CALLS   #0, STR$$INIT
                              50 00000000G  8F  DO 00515 87$:  MOVL    #STR$_NORMAL, RETURN_STATUS
                              52        56  DO 0051C      MOVL    TOTAL_LENGTH, R2
                 0000FFFF     8F        52  D1 0051F      CMPL    R2, #65535
                                        05  15 00526      BLEQ    88$
                              52    FFFF  8F  3C 00528      MOVZWL  #65535, R2
                 000000F0     8F        52  D1 0052D 88$:  CMPL    R2, #240
                                        61  1A 00534      BGTRU   96$
                                        52  D5 00536      TSTL    R2
                                        04  12 00538      BNEQ    89$
                                        53  D4 0053A      CLRL    TEMP
                                        3B  11 0053C      BRB     94$
                              51    FF  A2  9E 0053E 89$:  MOVAB   -1(R2), R1
                              51        07  8A 00542      BICB2   #7, R1
                              54 00000000G0041  9E 00545      MOVAB   STR$$Q_SHORT_Q[R1], REMQUE_ADDR
                              53      00  B4  0F 0054D 90$:  REMQUE  @0(REMQUE_ADDR), TEMP
                                        05  1D 00551      BVS     91$
                              52        01  DO 00553      MOVL    #1, ALLOC_DONE
                                        19  11 00556      BRB     93$
                              52        D4  00558 91$:  CLRL    ALLOC_DONE
                                        56  7D 0055A      PUSHL   TOTAL_LENGTH
                 0000FFFF     8F        6E  D1 0055C      CMPL    (SP), #65535
                                        05  15 00563      BLEQ    92$
                              6E    FFFF  8F  3C 00565      MOVZWL  #65535, (SP)
                 00000000G    00        01  FB 0056A 92$:  CALLS   #1, STR$$ALOC_SHORT
                              05        52  E8 00571 93$:  BLBS    ALLOC_DONE, 94$
                              41        50  E9 00574      BLBC    RETURN_STATUS, 98$
                                        D4  11 00577      BRB     90$
                              3C        50  E9 00579 94$:  BLBC    RETURN_STATUS, 98$
                           24   AE        53  DO 0057C      MOVL    TEMP, TEMP_DESC+4
                              51        56  DO 00580      MOVL    TOTAL_LENGTH, R1
                 0000FFFF     8F        51  D1 00583      CMPL    R1, #65535
                                        05  15 0058A      BLEQ    95$
                              51    FFFF  8F  3C 0058C      MOVZWL  #65535, R1
                           20   AE        51  BO 00591 95$:  MOVW    R1, TEMP_DESC
```

```
                                                21  11 00595                BRB     98$
                                            24  AE  9F 00597 96$:           PUSHAB  TEMP_DESC+4
                                        10  AE      52  D0 0059A            MOVL    R2, T6(SP)
                                            10  AE  9F 0059E                PUSHAB  16(SP)
                                00000000G   00      02  FB 005A1            CALLS   #2, LIB$GET_VM
                                            09      50  E8 005A8            BLBS    RETURN_STATUS, 97$
                                            50 00000000G 8F D0 005AB        MOVL    #STR$_INSVIRMEM, RETURN_STATUS
                                                04  11 005B2                BRB     98$
                                        20  AE          52  B0 005B4 97$:    MOVW    R2, TEMP_DESC
                                        08  AE          50  D0 005B8 98$:    MOVL    RETURN_STATUS, RETURN_STATUS
                                            03      08  AE  E8 005BC        BLBS    RETURN_STATUS, 99$                      1958
                                                0091    31 005C0            BRW     107$
                                            58      24  AE  D0 005C3 99$:    MOVL    TEMP_DESC+4, R8                         1961
                                            53          58  D0 005C7        MOVL    R8, CHR_PTR
                                            57          6C  9A 005CA        MOVZBL  (AP), R7                                1965
                                            59          01  D0 005CD        MOVL    #1, ARG_NO
                                                        20  11 005D0        BRB     103$
                                            50          6C49 D0 005D2 100$:  MOVL    (AP)[ARG_NO], SRC_DESC                  1978
                                            02      03  A0  91 005D6        CMPB    3(SRC_DESC), #2                         1988
                                                        09  1A 005DA        BGTRU   101$
                                            52          60  3C 005DC        MOVZWL  (SRC_DESC), IN_LEN
                                            51      04  A0  D0 005DF        MOVL    4(SRC_DESC), IN_ADDR
                                                        09  11 005E3        BRB     102$
                                    00000000G           00  16 005E5 101$:  JSB     STR$ANALYZE_SDESC_R1
                                            52          50  D0 005EB        MOVL    R0, R2
                                    63      61          52  28 005EE 102$:  MOVC3   IN_LEN, (IN_ADDR), (CHR_PTR)           1990
                                    DC      59          57  F3 005F2 103$:  AOBLEQ  R7, ARG_NO, 100$                      1965
                                            50      04  BC  3C 005F6        MOVZWL  @DEST_DESC, R0                          2000
                                            56          50  D1 005FA        CMPL    R0, TOTAL_LENGTH
                                                        03  15 005FD        BLEQ    104$
                                            50          56  D0 005FF        MOVL    TOTAL_LENGTH, R0
                                00      BE              68  50 28 00602 104$:  MOVC3   R0, (R8), @OUT_ADDR                 2002
                                            50 00000000G 8F D0 00607        MOVL    #STR$_NORMAL, RETURN_STATUS             2004
                                                        58  D5 0060E        TSTL    R8
                                                        3E  13 00610        BEQL    106$
                                    00F0    8F      20  AE  B1 00612        CMPW    TEMP_DESC, #240
                                                        1A  1A 00618        BGTRU   105$
                                            51          58  D0 0061A        MOVL    R8, STRING_BLOCK
                                            51      FE  A1  3C 0061D        MOVZWL  -2(STRING_BLOCK), ALLOC_LENGTH
                                                        51  D7 00621        DECL    R1
                                            51          07  8A 00623        BICB2   #7, R1
                                            51 00000000G0041 9E 00626       MOVAB   STR$$Q_SHORT_Q[R1], INSQUE_ADDR
                                        00  B1              68  0E 0062E    INSQUE  (R8), @0(INSQUE_ADDR)
                                                        1C  11 00632        BRB     106$
                                                24  AE  9F 00634 105$:      PUSHAB  TEMP_DESC+4
                                        10  AE      24  AE  3C 00637        MOVZWL  TEMP_DESC, 16(SP)
                                                10  AE  9F 0063C            PUSHAB  16(SP)
                                00000000G   00          02  FB 0063F        CALLS   #2, LIB$FREE_VM
                                            07          50  E8 00646        BLBS    RETURN_STATUS, 106$
                                            50 00000000G 8F D0 00649        MOVL    #STR$_FATINTERR, RETURN_STATUS
                                        08  AE          50  D0 00650 106$:  MOVL    RETURN_STATUS, RETURN_STATUS
                                            50      04  BC  3C 00654 107$:  MOVZWL  @DEST_DESC, R0                          2013
                                            56          50  D1 00658        CMPL    R0, TOTAL_LENGTH
                                                        03  15 0065B        BLEQ    108$
                                            50          56  D0 0065D        MOVL    TOTAL_LENGTH, R0
                                            5A          50  D0 00660 108$:  MOVL    R0, RESULT_LENGTH                       2012
                                                        4A  11 00663        BRB     115$                                    1941
```

```
                                53              6E  D0 00665 109$:   MOVL    OUT_ADDR, CHR_PTR                       ; 2031
                                57              02  7D 00668        MOVQ    #2, ARG_NO                              ; 2033
                            04  AE              58  D1 0066B 110$:   CMPL    CHARS_MOVED, OUT_LEN                    ; 2035
                                                3B  13 0066F        BEQL    114$                                    ;
                        52      04  AE          58  C3 00671        SUBL3   CHARS_MOVED, OUT_LEN, CHARS_LEFT        ; 2049
                57      6C          08          00  ED 00676        CMPZV   #0, #8, (AP), ARG_NO                    ; 2051
                                                2F  19 0067B        BLSS    114$                                    ;
                                50          6C47  D0 0067D        MOVL    (AP)[ARG_NO], SRC_DESC                  ; 2074
                                02      03      A0  91 00681        CMPB    3(SRC_DESC), #2                         ; 2083
                                                09  1A 00685        BGTRU   111$                                    ;
                                59              60  3C 00687        MOVZWL  (SRC_DESC), IN_LEN                      ;
                                51      04      A0  D0 0068A        MOVL    4(SRC_DESC), IN_ADDR                    ;
                                                09  11 0068E        BRB     112$                                    ;
                            000000000G          00  16 00690 111$:   JSB     STR$ANALYZE_SDESC_R1                    ;
                                59              50  D0 00696        MOVL    R0, R9                                  ;
                                52              59  D1 00699 112$:   CMPL    R9, CHARS_LEFT                          ; 2085
                                                03  15 0069C        BLEQ    113$                                    ;
                                59              52  D0 0069E        MOVL    CHARS_LEFT, R9                          ;
                        63      61              59  28 006A1 113$:   MOVC3   R9, (IN_ADDR), (CHR_PTR)              ; 2086
                                58              59  C0 006A5        ADDL2   R9, CHARS_MOVED                         ; 2089
                                                57  D6 006A8        INCL    ARG_NO                                  ; 2091
                                                BF  11 006AA        BRB     110$                                    ; 2035
                                5A              58  D0 006AC 114$:   MOVL    CHARS_MOVED, RESULT_LENGTH              ; 2100
                                50      04      AC  D0 006AF 115$:   MOVL    DEST_DESC, R0                           ; 2108
                            04  B0              5A  B0 006B3        MOVW    RESULT_LENGTH, @4(R0)                   ;
                                12      08      AE  E8 006B7 116$:   BLBS    RETURN_STATUS, 117$                     ; 2124
            04      08  AE                  03  00  ED 006BB        CMPZV   #0, #3, RETURN_STATUS, #4               ;
                                                0A  12 006C1        BNEQ    117$                                    ;
                                        08      AE  DD 006C3        PUSHL   RETURN_STATUS                           ;
                                00000000G       00  01  FB 006C6        CALLS   #1, LIB$STOP                            ;
                                02      10      AE  D1 006CD 117$:   CMPL    RESULT_CLASS, #2                        ; 2126
                                                1A  12 006D1        BNEQ    119$                                    ;
                                56              5A  D1 006D3        CMPL    RESULT_LENGTH, TOTAL_LENGTH             ; 2130
                                                0D  13 006D6        BEQL    118$                                    ;
                                    00000000G   8F  DD 006D8        PUSHL   #STR$_STRTOOLON                         ; 2131
                        00000000G   00              01  FB 006DE        CALLS   #1, LIB$STOP                            ;
                                50 00000000G    8F  D0 006E5 118$:   MOVL    #STR$_NORMAL, R0                        ; 2141
                                                    04 006EC        RET                                             ;
                                56              5A  D1 006ED 119$:   CMPL    RESULT_LENGTH, TOTAL_LENGTH             ;
                                                04  19 006F0        BLSS    120$                                    ;
                                50              01  D0 006F2        MOVL    #1, R0                                  ;
                                                    04 006F5        RET                                             ;
                                50 00000000G    8F  D0 006F6 120$:   MOVL    #STR$_TRU, R0                           ;
                                                    04 006FD        RET                                             ; 2147

; Routine Size:  1790 bytes,    Routine Base:  _STR$CODE + 000C

;  963          2148  1
;  964          2149  1 END                                     !End of module STR$CONCAT
;  965          2150  1
;  966          2151  0 ELUDOM
```

```
;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  _STR$CODE                      1802  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        17         0       581          00:00.8




;                                   COMMAND QUALIFIERS

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

; Size:         1790 code + 12 data bytes
; Run Time:       00:26.6
; Elapsed Time:   01:45.7
; Lines/CPU Min:   4851
; Lexemes/CPU-Min: 26449
; Memory Used:  478 pages
; Compilation Complete
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

STRDUPLCH
LIS

STRCOMPAR
LIS

STRFINDSB
LIS

STRMATCH
LIS

STRMSG
LIS

STRLENEXT
LIS

STRCOPY
LIS

STRFINDFI
LIS

STRLEFT
LIS

STRMULTI
LIS

STRCOMEQL
LIS

STRCONCAT
LIS

STRMOVQ
LIS

STRGETFRE
LIS