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

FILEID**STRAPPEND

```
  SSSSSSSS  TTTTTTTTTT  RRRRRRRR      AAAAAA    PPPPPPPP    PPPPPPPP    EEEEEEEEEE  NN      NN  DDDDDDDD
  SSSSSSSS  TTTTTTTTTT  RRRRRRRR      AAAAAA    PPPPPPPP    PPPPPPPP    EEEEEEEEEE  NN      NN  DDDDDDDD
SS              TT      RR      RR   AA    AA   PP      PP  PP      PP  EE          NN      NN  DD      DD
SS              TT      RR      RR   AA    AA   PP      PP  PP      PP  EE          NN      NN  DD      DD
SS              TT      RR      RR   AA    AA   PP      PP  PP      PP  EE          NNNN    NN  DD      DD
SS              TT      RR      RR   AA    AA   PP      PP  PP      PP  EE          NNNN    NN  DD      DD
  SSSSSS        TT      RRRRRRRR     AA    AA   PPPPPPPP    PPPPPPPP    EEEEEEEE    NN  NN  NN  DD      DD
  SSSSSS        TT      RRRRRRRR     AA    AA   PPPPPPPP    PPPPPPPP    EEEEEEEE    NN  NN  NN  DD      DD
        SS      TT      RR  RR       AAAAAAAAAA   PP          PP          EE          NN    NNNN  DD      DD
        SS      TT      RR  RR       AAAAAAAAAA   PP          PP          EE          NN    NNNN  DD      DD
        SS      TT      RR    RR     AA    AA   PP          PP          EE          NN      NN  DD      DD
        SS      TT      RR    RR     AA    AA   PP          PP          EE          NN      NN  DD      DD
SSSSSSSS        TT      RR      RR   AA    AA   PP          PP          EEEEEEEEEE  NN      NN  DDDDDDDD
SSSSSSSS        TT      RR      RR   AA    AA   PP          PP          EEEEEEEEEE  NN      NN  DDDDDDDD

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$APPEND (      ! Append a string to the end of the destination
;     2          0002  0
;     3          0003  0                     IDENT = '1-007' ! File: STRAPPEND.B32   Edit: DG1007
;     4          0004  0
;     5          0005  0                     ) =
;     6          0006  1 BEGIN
;     7          0007  1
;     8          0008  1 !
;     9          0009  1 !*****************************************************************************
;    10          0010  1 !*                                                                           *
;    11          0011  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    12          0012  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    13          0013  1 !*  ALL RIGHTS RESERVED.                                                      *
;    14          0014  1 !*                                                                           *
;    15          0015  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    16          0016  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    17          0017  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    18          0018  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    19          0019  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    20          0020  1 !*  TRANSFERRED.                                                              *
;    21          0021  1 !*                                                                           *
;    22          0022  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    23          0023  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    24          0024  1 !*  CORPORATION.                                                              *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    27          0027  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    28          0028  1 !*                                                                           *
;    29          0029  1 !*                                                                           *
;    30          0030  1 !*****************************************************************************
;    31          0031  1 !
;    32          0032  1
;    33          0033  1 !++
;    34          0034  1 ! FACILITY: String support library
;    35          0035  1 !
;    36          0036  1 ! ABSTRACT:
;    37          0037  1 !       This routine appends the input string onto the end of the
;    38          0038  1 !       the destination string.  It will handle strings of any
;    39          0039  1 !       supported dtype or class.
;    40          0040  1 !
;    41          0041  1 ! ENVIRONMENT: User mode, AST level or not or mixed
;    42          0042  1 !
;    43          0043  1 ! AUTHOR: R. Will, CREATION DATE: 12-Nov-79
;    44          0044  1 !
;    45          0045  1 ! MODIFIED BY:
;    46          0046  1 !
;    47          0047  1 ! R. Will, 25-Oct-79 : VERSION 01
;    48          0048  1 ! 1-001 - Original
;    49          0049  1 ! 1-002 - String speedup, status from alloc and dealoc macros.
;    50          0050  1 !         RW  11-Jan-1980
;    51          0051  1 ! 1-003 - Enhance to be able to deal with a larger class of string
;    52          0052  1 !         descriptors.  Uses $STR$GET_LEN_ADDR to extract length and
;    53          0053  1 !         address of data out of a variety of source descriptors.
;    54          0054  1 !         CASE on class of output descriptor expanded to allow writing
;    55          0055  1 !         of both CLASS_D and CLASS_VS.
;    56          0056  1 !         RKR 10-APR-1981
;    57          0057  1 ! 1-004 - Speed up code. RKR 7-OCT-1981.
```

```
;    58          0058  1 ! 1-005 - Fix for SPR 11-55716.  Addressing problem when appending a
;    59          0059  1 !           truncated source string to a CLASS_VS descriptor because the
;    60          0060  1 !           total length would exceed MAXSTRLEN.  RKR 18-APR-1983.
;    61          0061  1 ! 1-006 - Add support for class SO string descriptors.  DG 3-Oct-1983.
;    62          0062  1 ! 1-007 - Change class SO string descriptors to SB.  DG 27-Feb-1984.
;    63          0063  1 !--
;    64          0064  1 !<BLF/PAGE>
```

```
;    66          0065  1 !
;    67          0066  1 ! SWITCHES:
;    68          0067  1 !
;    69          0068  1
;    70          0069  1 SWITCHES ADDRESSING_MODE
;    71          0070  1                 (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;    72          0071  1
;    73          0072  1 !
;    74          0073  1 ! LINKAGES:
;    75          0074  1 !
;    76          0075  1
;    77          0076  1 REQUIRE 'RTLIN:STRLNK';         ! Use require file with string linkages
;    78          0261  1
;    79          0262  1 !
;    80          0263  1 ! TABLE OF CONTENTS:
;    81          0264  1 !
;    82          0265  1
;    83          0266  1 FORWARD ROUTINE
;    84          0267  1     STR$APPEND;                 ! append the input string to the end
;    85          0268  1                                 ! of the destination string
;    86          0269  1
;    87          0270  1 !
;    88          0271  1 ! INCLUDE FILES:
;    89          0272  1 !
;    90          0273  1
;    91          0274  1 REQUIRE 'RTLIN:RTLPSECT';       ! Declare PSECTS code
;    92          0369  1 REQUIRE 'RTLIN:STRMACROS';      ! use string macros to write code
;    93          1285  1 LIBRARY 'RTLSTARLE';            ! STARLET library for macros and symbo s
;    94          1286  1
;    95          1287  1 !
;    96          1288  1 ! MACROS:  NONE
;    97          1289  1 !
;    98          1290  1
;    99          1291  1 !
;   100          1292  1 ! EQUATED SYMBOLS
;   101          1293  1 !
;   102          1294  1 LITERAL
;   103          1295  1         MAX_SIZE = 65535;       ! largest string we can handle
;   104          1296  1
;   105          1297  1
;   106          1298  1 !
;   107          1299  1 ! PSECT DECLARATIONS
;   108          1300  1 !
;   109          1301  1
;   110          1302  1 DECLARE_PSECTS (STR);
;   111          1303  1
;   112          1304  1 !
;   113          1305  1 ! OWN STORAGE: NONE
;   114          1306  1 !
;   115          1307  1
;   116          1308  1 !
;   117          1309  1 ! EXTERNAL REFERENCES:
;   118          1310  1 !
;   119          1311  1 EXTERNAL LITERAL
;   120          1312  1     STR$_ILLSTRCLA,                 ! signal illegal class error
;   121          1313  1     STR$_STRTOOLON,                 ! signal string too long
;   122          1314  1     STR$_TRU,                       ! status -- truncation (warning)
```

```
;  123          1315  1      STR$_NORMAL ;                ! successful append
;  124          1316  1
;  125          1317  1 EXTERNAL ROUTINE
;  126          1318  1      LIB$STOP:                    ! signal errors
```

```
;  128     1319  1 GLOBAL ROUTINE STR$APPEND (     ! append a string to the end of another
;  129     1320  1
;  130     1321  1                                  DEST_DESC,   ! pointer to destination descr.
;  131     1322  1                                  SRC_DESC     ! pointer to source descriptor
;  132     1323  1
;  133     1324  1                                 ) : =         !
;  134     1325  1
;  135     1326  1 !++
;  136     1327  1 ! FUNCTIONAL DESCRIPTION:
;  137     1328  1 !
;  138     1329  1 !       This routine takes a source string of any supported dtype and
;  139     1330  1 ! class, and appends that string to the end of the destination string,
;  140     1331  1 ! which may be of any supported class or dtype, except that it is
;  141     1332  1 ! impossible to add something to the end of a string having fixed-length
;  142     1333  1 ! semantics -- an error will always be signalled in that case
;  143     1334  1 !
;  144     1335  1 ! FORMAL PARAMETERS:
;  145     1336  1 !
;  146     1337  1 !       DEST_DESC.wt.dx         pointer to destination descriptor
;  147     1338  1 !       SRC_DESC.rt.dx          pointer to source descriptor
;  148     1339  1 !
;  149     1340  1 ! IMPLICIT INPUTS:
;  150     1341  1 !
;  151     1342  1 !       NONE
;  152     1343  1 !
;  153     1344  1 ! IMPLICIT OUTPUTS:
;  154     1345  1 !
;  155     1346  1 !       NONE
;  156     1347  1 !
;  157     1348  1 ! COMPLETION CODES:
;  158     1349  1 !
;  159     1350  1 !       STR$_NORMAL             Success
;  160     1351  1 !       STR$_TRU                Truncation -- a warning
;  161     1352  1 !
;  162     1353  1 ! SIDE EFFECTS:
;  163     1354  1 !
;  164     1355  1 !       STR$_ILLSTRCLA may be signalled if the destination string is
;  165     1356  1 !                       of fixed-length semantics
;  166     1357  1 !                       or undefined
;  167     1358  1 !       STR$_STRTOOLON if combined lengths of source and destination
;  168     1359  1 !                       exceed 65K in a dynamic string destination.
;  169     1360  1 !       Dynamic string space may be allocated or deallocated
;  170     1361  1 !
;  171     1362  1 !--
;  172     1363  1
;  173     1364  2     BEGIN
;  174     1365  2
;  175     1366  2     LOCAL
;  176     1367  2         SRC_LEN,                    ! length of source string
;  177     1368  2         SRC_ADDR,                   ! addr of 1st byte of source
;  178     1369  2         DEST_LEN,                   ! length of destination string
;  179     1370  2         DEST_ADDR,                  ! addr of 1st byte of dest.
;  180     1371  2         TOT_LEN,                    ! sum of src and dst length
;  181     1372  2         RETURN_STATUS;              ! status from macros
;  182     1373  2
;  183     1374  2     MAP DEST_DESC : REF $STR$DESCRIPTOR;
;  184     1375  2     MAP SRC_DESC : REF $STR$DESCRIPTOR;
```

```
;  185          1376  2
;  186          1377  2      RETURN_STATUS = 1 ;          ! Assume success
;  187          1378  2
;  188          1379  2  !+
;  189          1380  2  ! find lengths and starting addresses of strings we're dealing with,
;  190          1381  2  ! and their combined lengths.
;  191          1382  2  !-
;  192          1383  2
;  193          1384  2      $STR$GET_LEN_ADDR (SRC_DESC, SRC_LEN,  SRC_ADDR) ;
;  194          1385  2
;  195          1386  2      $STR$GET_LEN_ADDR (DEST_DESC, DEST_LEN, DEST_ADDR) ;
;  196          1387  2
;  197          1388  2      TOT_LEN = .SRC_LEN + .DEST_LEN ;
;  198          1389  2
;  199          1390  2  !+
;  200          1391  2  ! if the combined lengths are greater that 65K, the result is not
;  201          1392  2  ! going to fit in any class of descriptor we support -- return
;  202          1393  2  ! STR$_STRTOOLON.
;  203          1394  2  !-
;  204          1395  2
;  205          1396  2      IF .TOT_LEN GTR MAX_SIZE THEN LIB$STOP (STR$_STRTOOLON) ;
;  206          1397  2
;  207          1398  2
;  208          1399  2  !+
;  209          1400  2  ! Algorithm for writing output differs based on the class of the
;  210          1401  2  ! destination descriptor.  The operation of appending is defined only
;  211          1402  2  ! for class CLASS_D and CLASS_VS destination strings.
;  212          1403  2  !-
;  213          1404  2
;  214          1405  2      CASE .DEST_DESC [DSC$B_CLASS]
;  215          1406  2      FROM DSC$K_CLASS_Z TO DSC$K_CLASS_SB OF
;  216          1407  2          SET
```

```
;  218          1408  2 !+
;  219          1409  2 ! dynamic destination strings
;  220          1410  2 ! ************************
;  221          1411  2 !-
;  222          1412  2
;  223          1413  2         [DSC$K_CLASS_D]:
;  224          1414  3             BEGIN
;  225          1415  3             IF
;  226        L 1416  3             %IF %BLISS (BLISS16) OR %BLISS (BLISS36)
;  227        U 1417  3             %THEN                                   ! except on VAX
;  228        U 1418  3             $STR$OVERLAP ( .SRC_ADDR, .SRC_LEN      ! If src overlaps
;  229        U 1419  3                 CH$PLUS (.DEST_ADDR, .DEST_LEN),    ! with where it will be
;  230        U 1420  3                         .SRC_LEN)                   ! written
;  231        U 1421  3             OR
;  232          1422  3             %FI                                     ! or if dest. not large
;  233        P 1423  4             ($STR$NEED_ALLOC (                      !  enough for append
;  234          1424  4                 .TOT_LEN, ($STR$DYN_AL_LEN (DEST_DESC) ) ) )
;  235          1425  3             THEN                                    ! then allocate a temp
;  236          1426  3                                                     ! and use it for
;  237          1427  4                 BEGIN                               ! building output string
;  238          1428  4
;  239          1429  4                 LOCAL TEMP_DESC : $STR$DESCRIPTOR;
;  240          1430  4
;  241          1431  5                 IF (RETURN_STATUS = $STR$ALLOCATE (.TOT_LEN, TEMP_DESC))
;  242          1432  4                 THEN
;  243          1433  5                     BEGIN
;  244          1434  5                     CH$MOVE (.DEST_LEN, .DEST_ADDR, ! move dest to temp
;  245          1435  5                              .TEMP_DESC [DSC$A_POINTER]);
;  246          1436  5
;  247          1437  5                     CH$MOVE (.SRC_LEN,  ! move src to end of temp
;  248          1438  5                        .SRC_ADDR,
;  249          1439  5                        CH$PLUS (.TEMP_DESC [DSC$A_POINTER], .DEST_LEN));
;  250          1440  5
;  251          1441  5                     $STR$EXCH_DESCS (TEMP_DESC, DEST_DESC);
;  252          1442  5                                      ! switch temp and dest
;  253          1443  5
;  254          1444  5                     RETURN_STATUS =      ! deallocate temp
;  255          1445  5                         $STR$DEALLOCATE (TEMP_DESC);
;  256          1446  4                     END;
;  257          1447  4                 END     ! of append via temp descriptor
;  258          1448  3             ELSE
;  259          1449  4                 BEGIN   ! movement directly into output string
;  260          1450  4                 CH$MOVE ( .SRC_LEN, .SRC_ADDR,
;  261          1451  4                           CH$PLUS (.DEST_ADDR, .DEST_LEN));
;  262          1452  4
;  263          1453  4                 DEST_DESC [DSC$W_LENGTH] = .TOT_LEN ;  ! adjust size of
;  264          1454  4                                                        ! result string
;  265          1455  3                 END;    ! movement directly into output string
;  266          1456  2         END;            ! of Class_D
```

```
;  268          1457  2 !+
;  269          1458  2 ! Varying string descriptor
;  270          1459  2 ! ************************
;  271          1460  2 !-
;  272          1461  2
;  273          1462  2         [DSC$K_CLASS_VS]:
;  274          1463  3             BEGIN
;  275          1464  3                 IF .TOT_LEN GTRU .DEST_DESC [DSC$W_MAXSTRLEN]
;  276          1465  3                 THEN            ! resultant string won't fit within
;  277          1466  4                     BEGIN       ! MAXSTRLEN of allocated string.
;  278          1467  4
;  279          1468  4                         !+
;  280          1469  4                         ! Copy as much as is possible.  This length is
;  281          1470  4                         ! given by MAXSTRLEN - CURLEN.
;  282          1471  4                         !-
;  283          1472  4                         CH$MOVE (
;  284          1473  4                                 .DEST_DESC [DSC$W_MAXSTRLEN] -
;  285          1474  4                                 .(.DEST_DESC [DSC$A_POINTER])<0,16>,
;  286          1475  4                                 .SRC_ADDR,
;  287          1476  4                                 .DEST_DESC [DSC$A_POINTER] +
;  288          1477  4                                 .DEST_LEN + 2 ) ;
;  289          1478  4
;  290          1479  4                         !+
;  291          1480  4                         ! Adjust CURLEN field to reflect the new length
;  292          1481  4                         !-
;  293          1482  4                         (.DEST_DESC [DSC$A_POINTER])<0,16> =
;  294          1483  4                                         . DEST_DESC [DSC$W_MAXSTRLEN];
;  295          1484  4
;  296          1485  4                         RETURN_STATUS = STR$_TRU;
;  297          1486  4
;  298          1487  4                     END ! won't fit within MAXSTRLEN
;  299          1488  3                 ELSE
;  300          1489  4                     BEGIN       ! will fit within MAXSTRLEN
;  301          1490  4                         !+
;  302          1491  4                         ! Move source data directly into varying
;  303          1492  4                         ! string, starting at byte just beyond the last
;  304          1493  4                         ! byte that is currently there.
;  305          1494  4                         !-
;  306          1495  4                         CH$MOVE (.SRC_LEN, .SRC_ADDR,
;  307          1496  5                                  (.DEST_DESC [DSC$A_POINTER] +
;  308          1497  4                                   .DEST_LEN + 2) );
;  309          1498  4
;  310          1499  4                         !+
;  311          1500  4                         ! Adjust CURLEN field to reflect the new length
;  312          1501  4                         !-
;  313          1502  4                         (.DEST_DESC [DSC$A_POINTER])<0,16> =.TOT_LEN ;
;  314          1503  3                     END;
;  315          1504  2             END;        ! of Class_VS
;  316          1505  2
```

```
2 !+
2 ! otherwise we have an undefined class of descriptor, or a descriptor
2 ! which has fixed-length string semantics.  In either case, its an
2 ! error to try to append to it.
2 !-
2         [INRANGE,OUTRANGE]:
2             RETURN_STATUS = STR$_ILLSTRCLA;
2         TES;
2
2     $STR$SIGNAL_FATAL (RETURN_STATUS);          ! signal if fatal error
2     RETURN .RETURN_STATUS;                      ! no non-fatal errors
2                                                 ! possible
1     END;                                        ! End of STR$APPEND

                                        .TITLE  STR$APPEND
                                        .IDENT  \1-007\

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

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

                         0FFC 00000     .ENTRY  STR$APPEND, Save R2,R3,R4,R5,R6,R7,R8,R9,-  ; 1319
                                                R10,R11                                      ;
           5E            14 C2 00002    SUBL2   #20, SP                                      ;
           5B            01 D0 00005    MOVL    #1, RETURN_STATUS                            ; 1377
           50      08    AC D0 00008    MOVL    SRC_DESC, R0                                 ; 1384
           02      03    A0 91 0000C    CMPB    3(R0), #2                                    ;
                         09 1A 00010    BGTRU   1$                                           ;
           5A            60 3C 00012    MOVZWL  (R0), SRC_LEN                                ;
           59      04    A0 D0 00015    MOVL    4(R0), SRC_ADDR                              ;
                         0C 11 00019    BRB     2$                                           ;
              00000000G  00 16 0001B 1$: JSB    STR$ANALYZE_SDESC_R1                         ;
           5A            50 D0 00021    MOVL    R0, R10                                      ;
           59            51 D0 00024    MOVL    R1, R9                                       ;
           57      04    AC D0 00027 2$: MOVL   DEST_DESC, R7                                ; 1386
           02      03    A7 91 0002B    CMPB    3(R7), #2                                    ;
                         09 1A 0002F    BGTRU   3$                                           ;
           58            67 3C 00031    MOVZWL  (R7), DEST_LEN                               ;
           52      04    A7 D0 00034    MOVL    4(R7), DEST_ADDR                             ;
                         0F 11 00038    BRB     4$                                           ;
           50            57 D0 0003A 3$: MOVL   R7, R0                                       ;
              00000000G  00 16 0003D    JSB     STR$ANALYZE_SDESC_R1                         ;
           58            50 D0 00043    MOVL    R0, R8                                       ;
           52            51 D0 00046    MOVL    R1, R2                                       ;
     56    5A            58 C1 00049 4$: ADDL3  DEST_LEN, SRC_LEN, TOT_LEN                   ; 1388
 0000FFFF  8F            56 D1 0004D    CMPL    TOT_LEN, #65535                              ; 1396
                         0D 15 00054    BLEQ    5$                                           ;
              00000000G  8F DD 00056    PUSHL   #STR$_STRTOOLON                              ;
 00000000G 00            01 FB 0005C    CALLS   #1, LIB$STOP                                 ;
```

```
                                 0F           00     03  A7  8F 00063 5$:    CASEB   3(R7), #0, #15                ; 1405
     0020              002A           0020       0020      00068 6$:    .WORD   7$-6$,-
     0020              0020           0020       0020      00070                7$-6$,-
     01A7              0020           0020       0020      00078                8$-6$,-
     0020              0020           0020       0020      00080                7$-6$,-
                                                                                7$-6$,-
                                                                                7$-6$,-
                                                                                7$-6$,-
                                                                                7$-6$,-
                                                                                7$-6$,-
                                                                                7$-6$,-
                                                                                7$-6$,-
                                                                                34$-6$,-
                                                                                7$-6$,-
                                                                                7$-6$,-
                                                                                7$-6$,-
                                                                                7$-6$
                            5B 00000000G  8F  D0 00088 7$:    MOVL    #STR$_ILLSTRCLA, RETURN_STATUS  ; 1512
                                        01B3  31 0008F        BRW     36$
                            51       04  A7  D0 00092 8$:    MOVL    4(R7), R1                     ; 1424
                                         53  D4 00096        CLRL    R3
                                         51  D5 00098        TSTL    R1
                                         06  12 0009A        BNEQ    9$
                                         53  D6 0009C        INCL    R3
                                         50  D4 0009E        CLRL    R0
                                         13  11 000A0        BRB     11$
                     00F0   8F           67  B1 000A2 9$:    CMPW    (R7), #240
                                         05  1B 000A7        BLEQU   10$
                            50           67  3C 000A9        MOVZWL  (R7), R0
                                         07  11 000AC        BRB     11$
                            50           51  D0 000AE 10$:   MOVL    R1, STRING_BLOCK
                            50       FE  A0  3C 000B1        MOVZWL  -2(STRING_BLOCK), R0
                 000000F0   8F           50  D1 000B5 11$:   CMPL    R0, #240
                                         21  1F 000BC        BLSSU   15$
                            04           53  E9 000BE        BLBC    R3, 12$
                                         50  D4 000C1        CLRL    R0
                                         13  11 000C3        BRB     14$
                     00F0   8F           67  B1 000C5 12$:   CMPW    (R7), #240
                                         05  1B 000CA        BLEQU   13$
                            50           67  3C 000CC        MOVZWL  (R7), R0
                                         07  11 000CF        BRB     14$
                            50           51  D0 000D1 13$:   MOVL    R1, STRING_BLOCK
                            50       FE  A0  3C 000D4        MOVZWL  -2(STRING_BLOCK), R0
                            50           56  D1 000D8 14$:   CMPL    TOT_LEN, R0
                                         21  13 000DB        BEQL    19$
                                         22  11 000DD        BRB     20$
                            04           53  E9 000DF 15$:   BLBC    R3, 16$
                                         50  D4 000E2        CLRL    R0
                                         13  11 000E4        BRB     18$
                     00F0   8F           67  B1 000E6 16$:   CMPW    (R7), #240
                                         05  1B 000EB        BLEQU   17$
                            50           67  3C 000ED        MOVZWL  (R7), R0
                                         07  11 000F0        BRB     18$
                            50           51  D0 000F2 17$:   MOVL    R1, STRING_BLOCK
                            50       FE  A0  3C 000F5        MOVZWL  -2(STRING_BLOCK), R0
                            50           56  D1 000F9 18$:   CMPL    TOT_LEN, R0
                                         03  1A 000FC        BGTRU   20$
```

```
                                 0104  31 000FE 19$:    BRW     33$                                                   ;
                 07 00000000G     00   E8 00101 20$:    BLBS    STR$$V_INIT, 21$                                      ; 1431
       00000000G  00              00   FB 00108         CALLS   #0, STR$$INIT                                         ;
                 50 00000000G     8F   D0 0010F 21$:    MOVL    #STR$_NORMAL, RETURN_STATUS                           ;
       000000F0   8F              56   D1 00116         CMPL    TOT_LEN, #240                                         ;
                                  3E   1A 0011D         BGTRU   27$                                                   ;
                                  56   D5 0011F         TSTL    TOT_LEN                                               ;
                                  04   12 00121         BNEQ    22$                                                   ;
                                  54   D4 00123         CLRL    TEMP                                                  ;
                                  2D   11 00125         BRB     26$                                                   ;
                 51          FF   A6   9E 00127 22$:    MOVAB   -1(R6), R1                                            ;
                 51               07   8A 0012B         BICB2   #7, R1                                                ;
                 55 00000000G0041      9E 0012E         MOVAB   STR$$Q_SHORT_Q[R1], REMQUE_ADDR                       ;
                 54          00   B5   0F 00136 23$:    REMQUE  @0(REMQUE_ADDR), TEMP                                 ;
                                  05   1D 0013A         BVS     24$                                                   ;
                 53               01   D0 0013C         MOVL    #1, ALLOC_DONE                                        ;
                                  0B   11 0013F         BRB     25$                                                   ;
                                  53   D4 00141 24$:    CLRL    ALLOC_DONE                                            ;
                                  56   DD 00143         PUSHL   TOT_LEN                                               ;
       00000000G  00              01   FB 00145         CALLS   #1, STR$$ALOC_SHORT                                   ;
                  05              53   E8 0014C 25$:    BLBS    ALLOC_DONE, 26$                                       ;
                  2C              50   E9 0014F         BLBC    RETURN_STATUS, 29$                                    ;
                                  E2   11 00152         BRB     23$                                                   ;
                  27              50   E9 00154 26$:    BLBC    RETURN_STATUS, 29$                                    ;
              10  AE              54   D0 00157         MOVL    TEMP, TEMP_DESC+4                                     ;
                                  1D   11 0015B         BRB     28$                                                   ;
                             10   AE   9F 0015D 27$:    PUSHAB  TEMP_DESC+4                                           ;
              04  AE              56   D0 00160         MOVL    TOT_LEN, 4(SP)                                        ;
                             04   AE   9F 00164         PUSHAB  4(SP)                                                 ;
       00000000G  00              02   FB 00167         CALLS   #2, LIB$GET_VM                                        ;
                  09              50   E8 0016E         BLBS    RETURN_STATUS, 28$                                    ;
                 50 00000000G     8F   D0 00171         MOVL    #STR$_INSVIRMEM, RETURN_STATUS                        ;
                                  04   11 00178         BRB     29$                                                   ;
              0C  AE              56   B0 0017A 28$:    MOVW    TOT_LEN, TEMP_DESC                                    ;
                  5B              50   D0 0017E 29$:    MOVL    RETURN_STATUS, RETURN_STATUS                          ;
                  7F              50   E9 00181         BLBC    RETURN_STATUS, 32$                                    ;
  10   BE         62              58   28 00184         MOVC3   DEST_LEN, (DEST_ADDR), @TEMP_DESC+4                   ; 1435
  10 BE48         69              5A   28 00189         MOVC3   SRC_LEN, (SRC_ADDR), @TEMP_DESC+4[DEST_LEN]           ; 1439
                  51         04   AC   D0 0018F         MOVL    DEST_DESC, R1                                         ; 1441
              04  AE              61   B0 00193         MOVW    (R1), $STR$TEMP_DESC                                  ;
              08  AE         04   A1   D0 00197         MOVL    4(R1), $STR$TEMP_DESC+4                               ;
              0E  AE         02   A1   B0 0019C         MOVW    2(R1), TEMP_DESC+2                                    ;
                  50         0C   AE   9E 001A1         MOVAB   TEMP_DESC, R0                                         ;
                    00000000G     00   16 001A5         JSB     STR$$MOVQ_R1                                          ;
              0C  AE         04   AE   B0 001AB         MOVW    $STR$TEMP_DESC, TEMP_DESC                             ;
              10  AE         08   AE   D0 001B0         MOVL    $STR$TEMP_DESC+4, TEMP_DESC+4                         ;
                 50 00000000G     8F   D0 001B5         MOVL    #STR$_NORMAL, RETURN_STATUS                           ; 1445
                 52          10   AE   D0 001BC         MOVL    TEMP_DESC+4, R2                                       ;
                                  3E   13 001C0         BEQL    31$                                                   ;
            00F0  8F         0C   AE   B1 001C2         CMPW    TEMP_DESC, #240                                       ;
                                  1A   1A 001C8         BGTRU   30$                                                   ;
                 51               52   D0 001CA         MOVL    R2, STRING_BLOCK                                      ;
                 51          FE   A1   3C 001CD         MOVZWL  -2(STRING_BLOCK), ALLOC_LENGTH                        ;
                                  51   D7 001D1         DECL    R1                                                    ;
                 51               07   8A 001D3         BICB2   #7, R1                                                ;
                 51 00000000G0041      9E 001D6         MOVAB   STR$$Q_SHORT_Q[R1], INSQUE_ADDR                       ;
              00  B1              62   0E 001DE         INSQUE  (R2), @0(INSQUE_ADDR)                                 ;
```

```
                                        1C  11 001E2       BRB     31$                                            ;
                                   10   AE  9F 001E4 30$:  PUSHAB  TEMP_DESC+4                                    ;
                   04   AE         10   AE  3C 001E7       MOVZWL  TEMP_DESC, 4(SP)                               ;
                                   04   AE  9F 001EC       PUSHAB  4(SP)                                          ;
           00000000G    00              02  FB 001EF       CALLS   #2, LIB$FREE_VM                                ;
                        07              50  E8 001F6       BLBS    RETURN_STATUS, 31$                             ;
                        50 00000000G    8F  D0 001F9       MOVL    #STR$_FATINTERR, RETURN_STATUS                 ;
                        5B              50  D0 00200 31$:  MOVL    RETURN_STATUS, RETURN_STATUS                   ;
                                        40  11 00203 32$:  BRB     36$                                            ; 1415
               6842     69              5A  28 00205 33$:  MOVC3   SRC_LEN, (SRC_ADDR), (DEST_LEN)[DEST_ADDR]     ; 1451
                        67              56  B0 0020A       MOVW    TOT_LEN, (R7)                                  ; 1453
                                        36  11 0020D       BRB     36$                                            ; 1405
      56        67      10              00  ED 0020F 34$:  CMPZV   #0, #16, (R7), TOT_LEN                         ; 1464
                                        21  1E 00214       BGEQU   35$                                            ;
                        51              67  3C 00216       MOVZWL  (R7), R1                                       ; 1474
                        50         04   B7  3C 00219       MOVZWL  @4(R7), R0                                     ;
                        51              50  C2 0021D       SUBL2   R0, R1                                         ;
                50      58         04   A7  C1 00220       ADDL3   4(R7), DEST_LEN, R0                            ; 1477
             02 A0      69              51  28 00225       MOVC3   R1, (SRC_ADDR), 2(R0)                          ;
                   04   B7              67  B0 0022A       MOVW    (R7), @4(R7)                                   ; 1483
                        5B 00000000G    8F  D0 0022E       MOVL    #STR$_TRU, RETURN_STATUS                       ; 1485
                                        0E  11 00235       BRB     36$                                            ; 1464
                50      58         04   A7  C1 00237 35$:  ADDL3   4(R7), DEST_LEN, R0                            ; 1497
             02 A0      69              5A  28 0023C       MOVC3   SRC_LEN, (SRC_ADDR), 2(R0)                     ; 1496
                   04   B7              56  B0 00241       MOVW    TOT_LEN, @4(R7)                                ; 1502
                        10              5B  E8 00245 36$:  BLBS    RETURN_STATUS, 37$                             ; 1515
      04        5B      03              00  ED 00248       CMPZV   #0, #3, RETURN_STATUS, #4                      ;
                                        09  12 0024D       BNEQ    37$                                            ;
                                        5B  DD 0024F       PUSHL   RETURN_STATUS                                  ;
           00000000G    00              01  FB 00251       CALLS   #1, LIB$STOP                                   ;
                        50              5B  D0 00258 37$:  MOVL    RETURN_STATUS, R0                              ; 1516
                                            04 0025B       RET                                                    ; 1518

; Routine Size:  604 bytes,    Routine Base:  _STR$CODE + 0000
```

```
;                1 END                                   .End of module
;                0 ELUDOM
```

PSECT SUMMARY

Name	Bytes	Attributes
_STR$CODE	604	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	-------- Symbols -------- Total	Loaded	Percent	Pages Mapped	Processing Time
_255DUA28:[SYSLIB]STARLET.L32;1	9776	11	0	581	00:00.8

COMMAND QUALIFIERS

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

; Size:          604 code + 0 data bytes
; Run Time:         00:10.5
; Elapsed Time:     00:43.6
; Lines/CPU Min:     8669
; Lexemes/CPU-Min: 37768
; Memory Used:  202 pages
; Compilation Complete
```

OTSPKDIVL
LIS

OTSPKDIVS
LIS

STRABMUL
LIS

STRCHESTA
LIS

RTLLIB
LIS

STRAPPEND
LIS

STRARITH
LIS

STRALLOC
LIS

STRANASTR
LIS

STRCOMCAS
LIS

OTSSCOPY
LIS

OTSTERMIO
LIS