

ST
1-

```

SSSSSSSS TTTTTTTTTT RRRRRRRR RRRRRRRR EEEEEEEEEEE PPPPPPPP LL          AAAAAA CCCCCCCC
SSSSSSSS TTTTTTTTTT RRRRRRRR RRRRRRRR EEEEEEEEEEE PPPPPPPP PP          AAAAAA CCCCCCCC
SS      TT      RR      RR      EE      PP      PP      LL      AA      AA      CC
SS      TT      RR      RR      EE      PP      PP      LL      AA      AA      CC
SS      TT      RR      RR      EE      PP      PP      LL      AA      AA      CC
SS      TT      RR      RR      EE      PP      PP      LL      AA      AA      CC
      SSSSSS      TT      RRRRRRRR RRRRRRRR EEEEEEEE PPPPPPPP LL      AA      AA      CC
      SSSSSS      TT      RRRRRRRR RRRRRRRR EEEEEEEE PPPPPPPP LL      AA      AA      CC
      SS      TT      RR      RR      EE      PP      PP      LL      AAAAAAAAAA CC
      SS      TT      RR      RR      EE      PP      PP      LL      AAAAAAAAAA CC
      SS      TT      RR      RR      EE      PP      PP      LL      AA      AA      CC
      SS      TT      RR      RR      EE      PP      PP      LL      AA      AA      CC
SSSSSSSS TT      RR      RR      RR      EEEEEEEEEEE PP      LLLLLLLLLL AA      AA      CCCCCCCC
SSSSSSSS TT      RR      RR      RR      EEEEEEEEEEE PP      LLLLLLLLLL AA      AA      CCCCCCCC

```

```

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
LLLLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLLLL IIIIII SSSSSSSS

```



```
1 0001 0 MODULE STR$REPLACE ( ! replace a substring in a string
2 0002 0
3 0003 0 IDENT = '1-005' ! File: STRREPLAC.B32 Edit: RKR1005
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 take an input string, an output string, and a
38 0038 1 replacement string of any supported class or dtype. It copies
39 0039 1 the source string to the destination string replacing the part
40 0040 1 of the source string specified by a starting position and an
41 0041 1 ending position by the replacement string.
42 0042 1
43 0043 1 ENVIRONMENT: User mode, AST level or not or mixed
44 0044 1
45 0045 1 AUTHOR: R. Will, CREATION DATE: 2-Dec-79
46 0046 1
47 0047 1 MODIFIED BY:
48 0048 1
49 0049 1 R. Will, 2-Dec-79: VERSION 01
50 0050 1 1-001 - original
51 0051 1 1-002 - String speedup, make JSB entry do code in line.
52 0052 1 RW 8-Jan-1980
53 0053 1 1-003 - Enhance to recognize additional classes of descriptors by
54 0054 1 using $STR$GET_LEN ADDR to extract length and address of
55 0055 1 first data byte. Remove string interlocking code.
56 0056 1 RKR 27-APR-81
57 0057 1 1-004 - Speed up code. RKR 7-OCT-1981.
```

STR\$REPLACE
1-005

F 4
16-Sep-1984 01:47:28
14-Sep-1984 12:40:13

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]STRREPLAC.B32;1

Page 2
(1)

```
: 58      0058 1 ! 1-005 - Use STR$COPY R R8 for copy operation. Use $STR$SIGNAL_FATAL
: 59      0059 1 !      instead of $STR$CHECK_STATUS. RKR 18-NOV-1981.
: 60      0060 1 !--
: 61      0061 1
: 62      0062 1 !<BLF/PAGE>
```



```

64 0063 1 |
65 0064 1 | SWITCHES:
66 0065 1 |
67 0066 1 |
68 0067 1 | SWITCHES ADDRESSING MODE
69 0068 1 | (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
70 0069 1 |
71 0070 1 |
72 0071 1 | LINKAGES:
73 0072 1 |
74 0073 1 |
75 0074 1 | REQUIRE 'RTLIN:STRLNK'; ! Use require file with string linkage
76 0259 1 |
77 0260 1 |
78 0261 1 | TABLE OF CONTENTS:
79 0262 1 |
80 0263 1 |
81 0264 1 | FORWARD ROUTINE
82 0265 1 | STR$REPLACE, ! replace a substring
83 0266 1 | STR$REPLACE_R8 : STR$JSB_REPLACE; ! replace a substring, JSB entry
84 0267 1 |
85 0268 1 |
86 0269 1 | INCLUDE FILES:
87 0270 1 |
88 0271 1 |
89 0272 1 | REQUIRE 'RTLIN:RTLPSECT'; ! Declare PSECTs code
90 0367 1 |
91 0368 1 | REQUIRE 'RTLIN:STRMACROS'; ! use string macros to code
92 1284 1 |
93 1285 1 | LIBRARY 'RTLSTARLE'; ! STARLET library for macros and symbol
94 1286 1 |
95 1287 1 |
96 1288 1 | MACROS: NONE
97 1289 1 |
98 1290 1 |
99 1291 1 |
100 1292 1 | EQUATED SYMBOLS: NONE
101 1293 1 |
102 1294 1 |
103 1295 1 |
104 1296 1 | PSECT DECLARATIONS
105 1297 1 |
106 1298 1 |
107 1299 1 | DECLARE_PSECTS (STR);
108 1300 1 |
109 1301 1 |
110 1302 1 | OWN STORAGE: NONE
111 1303 1 |
112 1304 1 |
113 1305 1 |
114 1306 1 | EXTERNAL REFERENCES:
115 1307 1 |
116 1308 1 |
117 1309 1 | EXTERNAL ROUTINE
118 1310 1 | LIB$STOP, ! Routine to signal error
119 1311 1 | STR$COPY_R_R8 : STR$JSB_COPY_R ; ! Routine to do the copy
120 1312 1 |
```

STR\$REPLACE
1-005

:	121	1313	1	EXTERNAL LITERAL
:	122	1314	1	STR\$ NORMAL,
:	123	1315	1	STR\$_STRTOOLON,
:	124	1316	1	
:	125	1317	1	STR\$_ILLSTRSPE,
:	126	1318	1	STR\$_ILLSTRPOS;

H 4
16-Sep-1984 01:47:28 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:40:13 [LIBRTL.SRC]STRREPLAC.B32;1

Page 4
(2)

! successful completion
! tried to create string longer
! than 65535
! illegal string specification
! illegal string position


```
128 1319 1 GLOBAL ROUTINE STR$REPLACE (          ! Replace a substring
129 1320 1
130 1321 1     DEST_DESC,          ! Pointer to destination descriptor
131 1322 1     SRC_DESC,         ! Pointer to source descriptor
132 1323 1     START_POS,       ! First character to be included
133 1324 1     END_POS,         ! Last character position to include
134 1325 1     REPLACE_DESC    ! pointer to replacement string desc
135 1326 1
136 1327 1     ) =
137 1328 1
138 1329 1 ++
139 1330 1 FUNCTIONAL DESCRIPTION:
140 1331 1
141 1332 1     This routine copies to the destination string, the characters
142 1333 1     from the beginning of the source string through the character
143 1334 1     before that specified by the start position (3rd input), then
144 1335 1     it continues copying to the destination string taking the
145 1336 1     input from the replacement string (the 5th input).  When the
146 1337 1     replacement string is exhausted, the copy continues from the
147 1338 1     source string starting at the character position after the end
148 1339 1     position (4th input parameter) until the source string is
149 1340 1     exhausted.  The string is built in a temporary and copied to
150 1341 1     the destination string (by JSB to STR$COPY_R_R8) according
151 1342 1     to the syntax of the class of the destination string.  The
152 1343 1     following conditions will return warnings, but the listed
153 1344 1     actions will be taken.
154 1345 1     If the starting position is < 1, 1 is used.  If the ending
155 1346 1     position is > length of the source string, the length of the
156 1347 1     source string is used.  If the starting position > the ending
157 1348 1     position, the overlapping portions of the source string will be
158 1349 1     copied twice.  The CALL entry is implemented by JSBing to the
159 1350 1     JSB entry.
160 1351 1
161 1352 1 FORMAL PARAMETERS:
162 1353 1
163 1354 1     DEST_DESC.wt.dx    pointer to destination string descriptor
164 1355 1     SRC_DESC.rt.dx     pointer to source string descriptor
165 1356 1     START_POS.rl.r     character position in source to start
166 1357 1                     substring
167 1358 1     END_POS.rl.r       character position in source to end
168 1359 1                     substring
169 1360 1     REPLACE_DESC.rt.dx pointer to replacement string descriptor
170 1361 1
171 1362 1 IMPLICIT INPUTS:
172 1363 1
173 1364 1     NONE
174 1365 1
175 1366 1 IMPLICIT OUTPUTS:
176 1367 1
177 1368 1     NONE
178 1369 1
179 1370 1 COMPLETION CODES:
180 1371 1
181 1372 1     any codes returned by STR$REPLACE_R8
182 1373 1
183 1374 1 SIDE EFFECTS:
184 1375 1
```

STR\$REPLACE
1-005

J 4
16-Sep-1984 01:47:28
14-Sep-1984 12:40:13

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]STRREPLAC.B32;1

Page 6
(3)

```
: 185      1376  1  | any side effects of STR$REPLACE_R8
: 186      1377  1  |
: 187      1378  1  |
: 188      1379  1  |
: 189      1380  2  | BEGIN
: 190      1381  2  |
: 191      1382  2  | MAP
: 192      1383  2  |   SRC_DESC : REF $STR$DESCRIPTOR,
: 193      1384  2  |   REPLACE_DESC : REF $STR$DESCRIPTOR,
: 194      1385  2  |   DEST_DESC : REF $STR$DESCRIPTOR;
: 195      1386  2  |
: 196      1387  2  | RETURN STR$REPLACE_R8 (
: 197      1388  2  |   DEST_DESC [0,0,0,0],
: 198      1389  2  |   SRC_DESC [0,0,0,0],
: 199      1390  2  |   ..START_POS,
: 200      1391  2  |   ..END_POS,
: 201      1392  2  |   REPLACE_DESC [0,0,0,0]);
: 202      1393  2  |
: 203      1394  1  | END;
```

!End of STR\$REPLACE

.TITLE STR\$REPLACE
.IDENT \1-005\

.EXTRN LIB\$STOP, STR\$COPY_R_R8
.EXTRN STR\$_NORMAL, STR\$_STRTOOLON
.EXTRN STR\$_ILLSTR\$PE, STR\$_ILLSTR\$POS

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

.ENTRY STR\$REPLACE, Save R2,R3,R4,R5,R6,R7,R8
MOVL REPLACE_DESC, R4
MOVL @END_POS, R3
MOVL @START_POS, R2
MOVQ DEST_DESC, R0
BSBW STR\$REPLACE_R8
RET

: 1319
: 1392
:
:
:
:
: 1394

; Routine Size: 22 bytes, Routine Base: _STR\$CODE + 0000

; 204 1395 1


```
206 1396 1 GLOBAL ROUTINE STR$REPLACE_R8 (      ! replace a substring
207 1397 1
208 1398 1     DEST_DESC,      ! Pointer to destination descriptor
209 1399 1     SRC_DESC,      ! Pointer to source descriptor
210 1400 1     START_POS,    ! First character to be included
211 1401 1     END_POS,      ! Last character position to include
212 1402 1     REPLACE_DESC  ! Pointer to replacement string desc
213 1403 1
214 1404 1     ) : STR$JSB_REPLACE =
215 1405 1
216 1406 1 ++
217 1407 1 FUNCTIONAL DESCRIPTION:
218 1408 1
219 1409 1     This routine copies to the destination string, the characters
220 1410 1     from the beginning of the source string through the character
221 1411 1     before that specified by the start position (3rd input), then
222 1412 1     it continues copying to the destination string taking the input
223 1413 1     from the replacement string (the 5th input). When the
224 1414 1     replacement string is exhausted, the copy continues from the
225 1415 1     source string starting at the character position after the end
226 1416 1     position (4th input parameter) until the source string is
227 1417 1     exhausted. The string is built in a
228 1418 1     temporary and copied to the destination string
229 1419 1     (by JSB to STR$COPY_R_R8) according
230 1420 1     to the syntax of the class of the destination string. The
231 1421 1     following conditions will return warnings, but the listed
232 1422 1     actions will be taken.
233 1423 1     If the starting position is < 1, 1 is used. If the ending
234 1424 1     position is > length of the source string, the length of the
235 1425 1     source string is used. If the starting position > the ending
236 1426 1     position, the overlapping portions of the source string will
237 1427 1     be copied twice.
238 1428 1
239 1429 1 FORMAL PARAMETERS:
240 1430 1
241 1431 1     DEST_DESC.wt.dx    pointer to destination string descriptor
242 1432 1     SRC_DESC.rt.dx     pointer to source string descriptor
243 1433 1     START_POS.rl.v     character position in source to start
244 1434 1                   substring
245 1435 1     END_POS.rl.v       character position in source to end
246 1436 1                   substring
247 1437 1     REPLACE_DESC.rt.dx  pointer to replacement string descriptor
248 1438 1
249 1439 1 IMPLICIT INPUTS:
250 1440 1
251 1441 1     NONE
252 1442 1
253 1443 1 IMPLICIT OUTPUTS:
254 1444 1
255 1445 1     NONE
256 1446 1
257 1447 1 COMPLETION CODES:
258 1448 1
259 1449 1     STR$_NORMAL        successful completion
260 1450 1     STR$_TRU           string was truncated when copied into
261 1451 1                   destination
262 1452 1     STR$_ILLSTRPOS     if START_POS or END_POS is < 0 or >
```

```
263 1453 1 | source-length
264 1454 1 | if START_POS > END_POS
265 1455 1 |
266 1456 1 | SIDE EFFECTS:
267 1457 1 |
268 1458 1 | allocation or deallocation of dynamic string space
269 1459 1 | may signal STR$_INSVIRMEM, STR$_ILLSTRCLA or STR$_FATINTERR
270 1460 1 |
271 1461 1 | --
272 1462 1 |
273 1463 2 | BEGIN
274 1464 2 |
275 1465 2 | LOCAL
276 1466 2 | COPY_LENGTH, | length of string built
277 1467 2 | ALLOC_LENGTH : UNSIGNED WORD, | actual length
278 1468 2 | NEW_POINTER, | keep place in temp string
279 1469 2 | ACTUAL_START, | 1 <= actual_start <= srclen
280 1470 2 | ACTUAL_END, | 1 <= actual_end <= srclen
281 1471 2 | STATUS, | status returned by copy
282 1472 2 | RETURN_STATUS, | save status for this routine
283 1473 2 | IN_LEN, | length of input string
284 1474 2 | IN_ADDR, | addr of 1st byte of input
285 1475 2 | REPL_LEN, | length of replacement string
286 1476 2 | REPL_ADDR, | addr of 1st byte of replace.
287 1477 2 | TEMP_DESC : $STR$DESCRIPTOR; | area to build output string
288 1478 2 |
289 1479 2 | MAP
290 1480 2 | SRC_DESC : REF $STR$DESCRIPTOR,
291 1481 2 | REPLACE_DESC : REF $STR$DESCRIPTOR,
292 1482 2 | DEST_DESC : REF $STR$DESCRIPTOR;
293 1483 2 |
294 1484 2 | RETURN_STATUS = 1 ; | Assume success to follow
295 1485 2 |
296 1486 2 | +
297 1487 2 | precompute lengths and addresses involved
298 1488 2 | -
299 1489 2 | $STR$GET_LEN_ADDR ( SRC_DESC, IN_LEN, IN_ADDR ) ;
300 1490 2 | $STR$GET_LEN_ADDR ( REPLACE_DESC, REPL_LEN, REPL_ADDR ) ;
301 1491 2 |
302 1492 2 | ACTUAL_START = .START_POS;
303 1493 2 | ACTUAL_END = .END_POS;
304 1494 2 |
305 1495 2 | +
306 1496 2 | detect error if START_POS > END_POS
307 1497 2 | -
308 1498 2 |
309 1499 2 | IF .ACTUAL_END LSS .ACTUAL_START THEN
310 1500 2 | RETURN_STATUS = STR$_ILLSTRSPE; | START_POS > END_POS
311 1501 2 |
312 1502 2 | +
313 1503 2 | actual_end is >= 0 and <= source string length
314 1504 2 | -
315 1505 2 |
316 1506 2 | IF .IN_LEN LSS .ACTUAL_END
317 1507 2 | THEN
318 1508 2 | BEGIN
319 1509 3 | RETURN_STATUS = STR$_ILLSTRPOS; | END_POS is too large
```



```
320      1510      3      ACTUAL_END = .IN_LEN ;          ! use srclen, remember
321      1511      3      ! error
322      1512      3      END
323      1513      2      ELSE IF .ACTUAL_END LSS 0
324      1514      2      THEN
325      1515      2      BEGIN
326      1516      2      RETURN_STATUS = STR$_ILLSTRPOS;    ! END_POS is too small
327      1517      2      ACTUAL_END = 0;                    ! Use 1, remember error
328      1518      2      END;
329      1519      2
330      1520      2      !+
331      1521      2      !- actual_start is >= 1 and <= source string length + 1
332      1522      2
333      1523      2
334      1524      2      IF .IN_LEN + 1 LSS .ACTUAL_START
335      1525      2      THEN
336      1526      2      BEGIN
337      1527      2      RETURN_STATUS = STR$_ILLSTRPOS;    ! START_POS is too large
338      1528      2      ACTUAL_START = .IN_LEN + 1;        ! use srclen, remember
339      1529      2      ! error
340      1530      2      END
341      1531      2      ELSE IF .ACTUAL_START LSS 1
342      1532      2      THEN
343      1533      2      BEGIN
344      1534      2      RETURN_STATUS = STR$_ILLSTRPOS;    ! START_POS is too small
345      1535      2      ACTUAL_START = 1;                    ! use 1, remember error
346      1536      2      END;
347      1537      2
348      1538      2      !+
349      1539      2      !- Create a temp descriptor and build destination string in it.
350      1540      2
351      1541      2
352      1542      2
353      1543      2      TEMP_DESC [DSC$W_LENGTH] = 0 ;      ! initialize descriptor
354      1544      2      TEMP_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T ;
355      1545      2      TEMP_DESC [DSC$B_CLASS] = DSC$K_CLASS_D ;
356      1546      2      TEMP_DESC [DSC$A_POINTER] = 0 ;
357      1547      2
358      1548      2      !+
359      1549      2      !- Number to copy is number from src + number from replace
360      1550      2
361      1551      2      COPY_LENGTH = .IN_LEN - .ACTUAL_END + .ACTUAL_START - 1 + .REPL_LEN;
362      1552      2
363      1553      2      IF .COPY_LENGTH GTR 65535
364      1554      2      THEN RETURN_STATUS = STR$_STRTOOLON    ! remember fatal error
365      1555      2      ELSE
366      1556      2      BEGIN
367      1557      2      ALLOC_LENGTH = .COPY_LENGTH;        ! need word unsigned for macro
368      1558      2
369      1559      2      ! allocate space to temp
370      1560      4      IF (NOT (STATUS = $STR$ALLOC_TMP (.ALLOC_LENGTH, TEMP_DESC)))
371      1561      3      THEN RETURN_STATUS = .STATUS          ! if fails, remember failure
372      1562      3      ELSE                                  ! if it succeeds continue
373      1563      4      BEGIN
374      1564      4
375      1565      4      !+
376      1566      4      !- copy up to start_pos from source string into temp string
```

```
377 1567 4      !-
378 1568 4      NEW_POINTER = CH$MOVE (.ACTUAL_START - 1, .IN_ADDR,
379 1569 4      .TEMP_DESC [DSC$A_POINTER]);
380 1570 4
381 1571 4      !+
382 1572 4      ! copy entire replacement string onto end of temp string
383 1573 4      !-
384 1574 4      NEW_POINTER = CH$MOVE (.REPL_LEN, .REPL_ADDR, .NEW_POINTER);
385 1575 4
386 1576 4      !+
387 1577 4      ! copy from END_POS to end in source string onto end of temp
388 1578 4      ! string.
389 1579 4      !-
390 1580 4      CH$MOVE (.IN_LEN - .ACTUAL_END,
391 1581 4      CH$PLUS (.IN_ADDR, .ACTUAL_END),
392 1582 4      .NEW_POINTER);
393 1583 4
394 1584 4      !+
395 1585 4      ! Copy the temp to the destination string and deallocate
396 1586 4      ! the temp
397 1587 4      !-
398 1588 4
399 1589 4      STATUS = STR$COPY_R_R8 (
400 1590 4      .DEST_DESC,
401 1591 4      .TEMP_DESC [DSC$W_LENGTH],
402 1592 4      .TEMP_DESC [DSC$A_POINTER] ) ;
403 1593 4      IF .STATUS NEQ SS$NORMAL
404 1594 4      THEN RETURN_STATUS = .STATUS;
405 1595 4      ! copy truncated, or
406 1596 4      ! fatal error, return
407 1597 4      ! instead of previous
408 1598 4      ! status
409 1599 5      IF (NOT (STATUS = $STR$DEALLOC_TMP (TEMP_DESC))) ! deallocate
410 1600 5      ! the temp
411 1601 4      THEN RETURN_STATUS = .STATUS;
412 1602 3      END;
413 1603 3      ! end of allocate else
414 1604 2      END;
415 1605 2      ! end of STRTOOLON else
416 1606 2      $STR$SIGNAL FATAL (RETURN_STATUS);
417 1607 2      RETURN .RETURN_STATUS;
418 1608 1      ! signal severe errors
      !End of STR$REPLACE
```

```
SE      18 C2 00000 STR$REPLACE R8::
      50 DD 00003 SUBC2 #24, SP
      01 DD 00005 PUSHL R0
      02 03 A1 91 00007 PUSHL #1
      CMPB 3(SRC_DESC), #2
```

```
.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$FREE_VM
.EXTRN STR$_FATINTERR
```

```
: 1396
: 1484
: 1489
```


10	AE		0B	1A	0000B	BGTRU	1\$		
OC	AE	04	61	3C	0000D	MOVZWL	(SRC_DESC), IN_LEN		
			A1	D0	00011	MOVL	4(SRC_DESC), IN_ADDR		
	50		11	11	00016	BRB	2\$		
		00000000G	51	D0	00018	1\$:	MOVL	SRC_DESC, R0	
			00	16	0001B	JSB	STR\$ANALYZE_SDESC_R1		
10	AE		50	D0	00021	MOVL	R0, 16(SP)		
OC	AE		51	D0	00025	MOVL	R1, 12(SP)		
	02	03	A4	91	00029	2\$:	CMPB	3(REPLACE_DESC), #2	1490
			0A	1A	0002D	BGTRU	3\$		
	57		64	3C	0002F	MOVZWL	(REPLACE_DESC), REPL_LEN		
14	AE	04	A4	D0	00032	MOVL	4(REPLACE_DESC), REPC_ADDR		
			10	11	00037	BRB	4\$		
	50		54	D0	00039	3\$:	MOVL	REPLACE_DESC, R0	
		00000000G	00	16	0003C	JSB	STR\$ANALYZE_SDESC_R1		
	57		50	D0	00042	MOVL	R0, R7		
14	AE		51	D0	00045	MOVL	R1, 20(SP)		
	56		53	D0	00049	4\$:	MOVL	END_POS, ACTUAL_END	1493
	52		56	D1	0004C	CMPL	ACTUAL_END, ACTUAL_START		1499
			07	18	0004F	BGEQ	5\$		
	6E	00000000G	8F	D0	00051	MOVL	#STR\$_ILLSTRSPE, RETURN_STATUS		1500
	56	10	AE	D1	00058	5\$:	CMPL	IN_LEN, ACTUAL_END	1506
			0D	18	0005C	BGEQ	6\$		
	6E	00000000G	8F	D0	0005E	MOVL	#STR\$_ILLSTRPOS, RETURN_STATUS		1509
	56	10	AE	D0	00065	MOVL	IN_LEN, ACTUAL_END		1510
			0D	11	00069	BRB	7\$		1506
			56	D5	0006B	6\$:	TSTL	ACTUAL_END	1513
			09	18	0006D	BGEQ	7\$		
	6E	00000000G	8F	D0	0006F	MOVL	#STR\$_ILLSTRPOS, RETURN_STATUS		1516
			56	D4	00076	CLRL	ACTUAL_END		1517
50	10	AE	01	C1	00078	7\$:	ADDL3	#1, IN_LEN, R0	1524
		52	50	D1	0007D	CMPL	R0, ACTUAL_START		
			0C	18	00080	BGEQ	8\$		
	6E	00000000G	8F	D0	00082	MOVL	#STR\$_ILLSTRPOS, RETURN_STATUS		1527
	52		50	D0	00089	MOVL	R0, ACTUAL_START		1528
			0E	11	0008C	BRB	9\$		1524
			52	D5	0008E	8\$:	TSTL	ACTUAL_START	1531
			0A	14	00090	BGTR	9\$		
	6E	00000000G	8F	D0	00092	MOVL	#STR\$_ILLSTRPOS, RETURN_STATUS		1534
	52		01	D0	00099	MOVL	#1, ACTUAL_START		1535
18	AE	020E0000	8F	D0	0009C	9\$:	MOVL	#34471936, TEMP_DESC	1543
		1C	AE	D4	000A4	CLRL	TEMP_DESC+4		1546
50	10	AE	56	C3	000A7	SUBL3	ACTUAL_END, IN_LEN, R0		1551
		50	52	C0	000AC	ADDL2	ACTUAL_START, R0		
		50	74	9E	000AF	MOVAB	-1(REPC_LEN)[R0], COPY_LENGTH		
0000FFFF	8F		50	D1	000B4	CMPL	COPY_LENGTH, #65535		1553
			0A	15	000BB	BLEQ	10\$		
	6E	00000000G	8F	D0	000BD	MOVL	#STR\$_STRTOOLON, RETURN_STATUS		1554
			01	16	000C4	BRW	25\$		
	53		50	B0	000C7	10\$:	MOVW	COPY_LENGTH, ALLOC_LENGTH	1557
	07	00000000G	00	E8	000CA	BLBS	STR\$\$V_INIT, 11\$		1560
00000000G	00		00	FB	000D1	CALLS	#0, STR\$\$INIT		
	50	00000000G	8F	D0	000D8	11\$:	MOVL	#STR\$_NORMAL, RETURN_STATUS	
	00F0		53	B1	000DF	CMPL	ALLOC_LENGTH, #240		
			40	1A	000E4	BGTRU	17\$		
			53	B5	000E6	TSTW	ALLOC_LENGTH		
			04	12	000E8	BNEQ	12\$		

			55	D4	000EA	CLRL	TEMP		
			2F	11	000EC	BRB	16\$		
	51		53	3C	000EE	12\$:	MOVZWL	ALLOC_LENGTH, R1	
			51	D7	000F1	DECL	R1		
	51		07	8A	000F3	BICB2	#7, R1		
	58	00000000G	00	41	9E	000F6	MOVAB	STR\$\$Q SHORT Q[R1], REMQUE_ADDR	
	55		00	B8	0F	000FE	13\$:	REMQUE	@0(REMQUE_ADDR), TEMP
			05	1D	00102	BVS	14\$		
	54		01	D0	00104	MOVL	#1, ALLOC_DONE		
			0C	11	00107	BRB	15\$		
			54	D4	00109	14\$:	CLRL	ALLOC_DONE	
	7E		53	3C	0010B	MOVZWL	ALLOC_LENGTH, -(SP)		
00000000G	00		01	FB	0010E	CALLS	#1, STR\$\$ALOC SHORT		
	05		54	E8	00115	15\$:	BLBS	ALLOC_DONE, 18\$	
	2C		50	E9	00118	BLBC	RETURN_STATUS, 19\$		
			E1	11	0011B	BRB	13\$		
	27		50	E9	0011D	16\$:	BLBC	RETURN_STATUS, 19\$	
1C	AE		55	D0	00120	MOVL	TEMP, TEMP_DESC+4		
			1D	11	00124	BRB	18\$		
		1C	AE	9F	00126	17\$:	PUSHAB	TEMP_DESC+4	
	0C	AE	53	3C	00129	MOVZWL	ALLOC_LENGTH, 12(SP)		
		0C	AE	9F	0012D	PUSHAB	12(SP)		
00000000G	00		02	FB	00130	CALLS	#2, LIB\$GET VM		
	09		50	E8	00137	BLBS	RETURN_STATUS, 18\$		
	50	00000000G	8F	D0	0013A	MOVL	#STR\$_INSVIRMEM, RETURN_STATUS		
			04	11	00141	BRB	19\$		
	18	AE	53	B0	00143	18\$:	MOVW	ALLOC_LENGTH, TEMP_DESC	
	58		50	D0	00147	19\$:	MOVL	RETURN_STATUS, STATUS	
	03		50	E8	0014A	BLBS	RETURN_STATUS, 20\$		
			008A	31	0014D	BRW	24\$		
			52	D7	00150	20\$:	DECL	R2	1568
1C	BE	0C	52	28	00152	MOV3	R2, @IN_ADDR, @TEMP_DESC+4		1569
63	14	BE	57	28	00158	MOV3	REPL_LEN, @REPL_ADDR, (NEW_POINTER)		1574
50	10	AE	56	C3	0015D	SUBL3	ACTUAL_END, IN_LEN, R0		1580
			57	D0	00162	MOVL	IN_ADDR, R7		1582
63		0C	50	28	00166	MOV3	R0, (ACTUAL_END)[R7], (NEW_POINTER)		
	6647		50	D0	0016B	MOVL	TEMP_DESC+4, R2		1590
		1C	51	AE	3C	0016F	MOVZWL	TEMP_DESC, R1	
		18	50	AE	D0	00173	MOVL	DEST_DESC, R0	
		04		00	16	00177	JSB	STR\$COPY R-R8	
		00000000G	58	D0	0017D	MOVL	R0, STATUS		
			01	58	D1	00180	CMPL	STATUS, #1	1593
				03	13	00183	BEQL	21\$	
			58	D0	00185	MOVL	STATUS, RETURN_STATUS		1594
	6E		8F	D0	00188	21\$:	MOVL	#STR\$ NORMAL, RETURN_STATUS	1599
	50	00000000G	1C	AE	D5	0018F	TSTL	TEMP_DESC+4	
			40	13	00192	BEQL	23\$		
	00F0	8F	18	AE	B1	00194	CMPW	TEMP_DESC, #240	
			1C	1A	0019A	BGTRU	22\$		
		1C	AE	D0	0019C	MOVL	TEMP_DESC+4, STRING_BLOCK		
		FE	A1	3C	001A0	MOVZWL	-2(STRING_BLOCK), ACLOC_LENGTH		
			51	D7	001A4	DECL	R1		
			07	8A	001A6	BICB2	#7, R1		
	51	00000000G	00	41	9E	001A9	MOVAB	STR\$\$Q SHORT Q[R1], INSQUE_ADDR	
	51		1C	BE	0E	001B1	INSQUE	@TEMP_DESC+4, @0(INSQUE_ADDR)	
			1C	11	001B6	BRB	23\$		
		1C	AE	9F	001B8	22\$:	PUSHAB	TEMP_DESC+4	

STR\$REPLACE
1-005

D 5
16-Sep-1984 01:47:28
14-Sep-1984 12:40:13

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]STRREPLAC.B32;1

Page 13
(4)

04	18	AE	1C	AE	3C	001BB	MOVZWL	TEMP DESC, 24(SP)	
			18	AE	9F	001C0	PUSHAB	24(SP)	
	00000000G	00		02	FB	001C3	CALLS	#2, LIB\$FREE VM	
		07		50	E8	001CA	BLBS	RETURN STATUS, 23\$	
		50	00000000G	8F	D0	001CD	MOVL	#STR\$ FATINTERR, RETURN_STATUS	
		58		50	D0	001D4	MOVL	RETURN STATUS, STATUS	
		03		50	E8	001D7	BLBS	RETURN STATUS, 25\$	
		6E		58	D0	001DA	MOVL	STATUS, RETURN STATUS	1601
		10		6E	E8	001DD	BLBS	RETURN STATUS, 26\$	1606
		03		00	ED	001E0	CMPZV	#0, #3, RETURN STATUS, #4	
				09	12	001E5	BNEQ	26\$	
				6E	DD	001E7	PUSHL	RETURN STATUS	
	00000000G	00		01	FB	001E9	CALLS	#1, LIB\$STOP	
		50		8E	D0	001F0	MOVL	RETURN STATUS, R0	1607
		5E		1C	C0	001F3	ADDL2	#28, SP	1608
					05	001F6	RSB		

; Routine Size: 503 bytes, Routine Base: _STR\$CODE + 0016

:	419	1609	1	
:	420	1610	1	END
:	421	1611	1	
:	422	1612	0	ELUDOM

!End of module

PSECT SUMMARY

Name	Bytes	Attributes
_STR\$CODE	525	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	9	0	581	00:00.8

COMMAND QUALIFIERS

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

STR\$REPLACE
1-005

E 5
16-Sep-1984 01:47:28

VAX-11 Bliss-32 V4.0-742

Page 14

: Size: 525 code + 0 data bytes
: Run Time: 00:10.2
: Elapsed Time: 00:42.6
: Lines/CPU Min: 9491
: Lexemes/CPU-Min: 29799
: Memory Used: 177 pages
: Compilation Complete

ST

0215 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

