


```

1 0001 0 MODULE STR$COPY ( ! Copy one string to another
2 0002 0
3 0003 0 IDENT = '1-015' ! File: STR$COPY.B32 Edit: DG1015
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 copies one string of any supported class and any
38 0038 1 supported dtype to a second string of any supported class and any
39 0039 1 supported dtype.
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: 20-Jan-79
44 0044 1
45 0045 1 MODIFIED BY:
46 0046 1
47 0047 1 R. Will, 20-Jan-79: VERSION 01
48 0048 1 01 - original
49 0049 1 02 - add by reference entries, rename by descriptor entries 15-Mar-79 RW
50 0050 1 1-003 - Change FILL_CHAR to STR$K_FILL_CHAR. JBS 09-APR-1979
51 0051 1 1-004 - Make CALL entry points accept scalars by ref. RW 21-MAY-79
52 0052 1 1-005 - Make Length references word instead of long. RW 23-May-79
53 0053 1 1-006 - Change linkage names to start with STR$. JBS 04-JUN-1979
54 0054 1 1-007 - Change order of src and dest parameters. RW 16-Jul-79
55 0055 1 1-008 - Correct a typo in edit 007. JBS 17-JUL-1979
56 0056 1 1-009 - Use new interlock macros. JBS 06-NOV-1979
57 0057 1 1-010 - The interlock macros cannot be used from a routine called

```

```

: 58      0058 1 | with a JBS because of the ENABLE. JBS 15-NOV-1979
: 59      0059 1 | 1-011 - String speedup, remove edit 10. RW 8-Jan-1980
: 60      0060 1 | 1-012 - Reorganize string copying routines to use the corresponding
: 61      0061 1 | LIB$SCOPY_xxx entry points to do the real work.
: 62      0062 1 | This makes them sensitive to the newly-added classes of
: 63      0063 1 | descriptors. Remove string interlocking code. RKR 31-MAR-1981
: 64      0064 1 | 1-013 - Speed up by special-casing classes of descriptors that 'read'
: 65      0065 1 | like fixed string descriptors.
: 66      0066 1 | To bring performance back to Version 2 levels, it became
: 67      0067 1 | necessary to replicate the logic found in LIB$SCOPY_R_DX6
: 68      0068 1 | in STRSCOPY_R_R8.
: 69      0069 1 | RKR 18-NOV-1981
: 70      0070 1 | 1-014 - Add support for class S0 string descriptor. DG 3-OCT-1983.
: 71      0071 1 | 1-015 - Change class S0 string descriptor to SB. DG 27-Feb-1984.
: 72      0072 1 | --
: 73      0073 1 |
: 74      0074 1 | !<BLF/PAGE>
```

```
76 0075 1 |
77 0076 1 | SWITCHES:
78 0077 1 |
79 0078 1 |
80 0079 1 SWITCHES ADDRESSING MODE
81 0080 1 (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
82 0081 1 !SWITCHES LIST (EXPAND);
83 0082 1 |
84 0083 1 |
85 0084 1 | LINKAGES:
86 0085 1 |
87 0086 1 REQUIRE 'RTLIN:STRLNK'; ! Use require file with string linkage
88 0271 1 |
89 0272 1 |
90 0273 1 | TABLE OF CONTENTS:
91 0274 1 |
92 0275 1 |
93 0276 1 FORWARD ROUTINE
94 0277 1 STRSCOPY_DX, ! Copy string to another,
95 0278 1 ! CALL entry
96 0279 1 STRSCOPY_R, ! Copy string by ref, CALL entry
97 0280 1 STRSCOPY_DX_RB : STR$JSB_COPY_DX, ! Copy string to another,
98 0281 1 ! JSB entry
99 0282 1 STRSCOPY_R_RB : STR$JSB_COPY_R; ! Copy string by ref, JSB entry
100 0283 1 |
101 0284 1 |
102 0285 1 | INCLUDE FILES:
103 0286 1 |
104 0287 1 |
105 0288 1 REQUIRE 'RTLIN:RTLPSECT'; ! Use to declare PSECTS
106 0383 1 |
107 0384 1 REQUIRE 'RTLIN:STRMACROS'; ! Use string macros to code
108 1300 1 |
109 1301 1 LIBRARY 'RTLSTARLE'; ! STARLET library for macros and symbol
110 1302 1 |
111 1303 1 |
112 1304 1 | MACROS:
113 1305 1 |
114 1306 1 NONE
115 1307 1 |
116 1308 1 | EQUATED SYMBOLS:
117 1309 1 |
118 1310 1 LITERAL
119 1311 1 MAX_SIZE = 65535 ; ! Maximum string size
120 1312 1 |
121 1313 1 | PSECT DECLARATIONS
122 1314 1 |
123 1315 1 |
124 1316 1 DECLARE_PSECTS (STR);
125 1317 1 |
126 1318 1 |
127 1319 1 | OWN STORAGE:
128 1320 1 |
129 1321 1 NONE
130 1322 1 |
131 1323 1 | EXTERNAL REFERENCES:
132 1324 1 |
```

STRSCOPY
1-015

L⁵
16-Sep-1984 01:35:39
14-Sep-1984 12:40:04

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

Page 4
(2)

ST
1-

```
.. 133      1325 1
... 134      1326 1 EXTERNAL LITERAL
... 135      1327 1 STRS_FATINTERR,      ! Fatal internal error
... 136      1328 1 STRS_INSVIRMEM,    ! Insufficient virt. mem
... 137      1329 1 STRS_ILLSTRCLA,  ! Illegal string desc
... 138      1330 1 STRS_TRU,        ! String truncated
... 139      1331 1 STRS_NORMAL;      ! success
... 140      1332 1
... 141      1333 1 EXTERNAL ROUTINE
... 142      1334 1 LIB$STOP;      ! signal errors
```

```

144 1335 1 GLOBAL ROUTINE STR$COPY_DX (                ! Copy string
145 1336 1
146 1337 1     DEST_DESC,                ! Pointer to dest str desc
147 1338 1     SRC_DESC                ! Pointer to input str desc
148 1339 1     ) =
149 1340 1
150 1341 1 ++
151 1342 1 FUNCTIONAL DESCRIPTION:
152 1343 1     This routine copies a source string to a destination string
153 1344 1     where both the source and destination may be of any class or any
154 1345 1     dtype. This is the CALL entry point, it puts the parameters in the
155 1346 1     correct place and JSBs to the JSB entry point.
156 1347 1
157 1348 1 FORMAL PARAMETERS:
158 1349 1
159 1350 1     DEST_DESC.wt.dx    pointer to destination string descriptor
160 1351 1     SRC_DESC.rt.dx    pointer to source string descriptor
161 1352 1
162 1353 1 IMPLICIT INPUTS:
163 1354 1
164 1355 1     NONE
165 1356 1
166 1357 1 IMPLICIT OUTPUTS:
167 1358 1
168 1359 1     NONE
169 1360 1
170 1361 1 COMPLETION CODES:
171 1362 1
172 1363 1     SSS_NORMAL        Success
173 1364 1     STR$_TRU          Truncation occured. Warning.
174 1365 1
175 1366 1 SIDE EFFECTS:
176 1367 1
177 1368 1 Will signal STR$ INSVIRMEM if no heap memory to allocate strings or
178 1369 1 STR$_ILLSTRCLA if class in descriptor is not supported.
179 1370 1
180 1371 1 --
181 1372 1 BEGIN
182 1373 2     RETURN (STR$COPY_DX_R8 ( .DEST_DESC, .SRC_DESC ) );
183 1374 2     END;
184 1375 1

```

```

.TITLE STR$COPY
.IDENT \1-015\

.EXTRN STR$_FATINTERR, STR$_INSVIRMEM
.EXTRN STR$_ILLSTRCLA, STR$_TRU
.EXTRN STR$_NORMAL, LIB$STOP

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

.ENTRY STR$COPY_DX, Save R2,R3,R4,R5,R6,R7,R8 : 1335
MOVQ DEST_DESC, R0 : 1374
BSBW STR$COPY_DX_R8 : 1375
RET : 1375

```

```

50      04      01FC 0000
        AC      7D 0002
        0000V 30 0006
        04      0009

```

STRSCOPY
1-015

N 5
16-Sep-1984 01:35:39
14-Sep-1984 12:40:04

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

Page 6
(3)

: Routine Size: 10 bytes, Routine Base: _STR\$CODE + 0C00

ST
1-


```

186 1376 1 GLOBAL ROUTINE STR$COPY_R (           ! Copy a string
187 1377 1
188 1378 1     DEST_DESC,           ! Pointer to dest str desc
189 1379 1     SRC_LEN,           ! Value of src string length
190 1380 1     SRC_ADDR           ! Pointer to source string
191 1381 1           ) =
192 1382 1
193 1383 1 ++
194 1384 1 FUNCTIONAL DESCRIPTION:
195 1385 1
196 1386 1     This routine copies a source string to a destination string
197 1387 1     where both the source and destination may be of any class or any
198 1388 1     dtype. This is the CALL with source string by reference
199 1389 1     entry point, it puts the parameters in the correct
200 1390 1     place and JSBs to the JSB entry point.
201 1391 1
202 1392 1 FORMAL PARAMETERS:
203 1393 1
204 1394 1     DEST_DESC.wt.dx     pointer to destination string descriptor
205 1395 1     SRC_LEN.rwu.r       addr of value of length of source string
206 1396 1     SRC_ADDR.rt.r       pointer to source string
207 1397 1
208 1398 1 IMPLICIT INPUTS:
209 1399 1
210 1400 1     NONE
211 1401 1
212 1402 1 IMPLICIT OUTPUTS:
213 1403 1
214 1404 1     NONE
215 1405 1
216 1406 1 COMPLETION CODES:
217 1407 1
218 1408 1     SSS_NORMAL          Success
219 1409 1     STR$_TRU            Truncation occured. Warning.
220 1410 1
221 1411 1 SIDE EFFECTS:
222 1412 1
223 1413 1 Will signal STR$ INSVIRMEM if no heap memory to allocate strings or
224 1414 1 STR$_ILLSTRCLA if class in descriptor is not supported.
225 1415 1
226 1416 1 --
227 1417 1
228 1418 1
229 1419 2 BEGIN
230 1420 2 RETURN (STR$COPY_R_R8 (.DEST_DESC, ..SRC_LEN, .SRC_ADDR) ) ;
231 1421 1 END;

```

```

          01FC 00000          .ENTRY STR$COPY_R, Save R2,R3,R4,R5,R6,R7,R8
52      0C   AC  D0 00002          MOVL SRC_ADDR, R2          : 1376
51      08   BC  D0 00006          MOVL @SRC_LEN, R1         : 1420
50      04   AC  D0 0000A          MOVL DEST_DESC, R0
          0000V 30 0000E          BSBW STR$COPY_R_R8
          04 00011          RET          : 1421

```

STRSCOPY
1-015

⁶
16-Sep-1984 01:35:39
14-Sep-1984 12:40:04

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

Page 8
(4)

; Routine Size: 18 bytes, Routine Base: _STR\$CODE + 000A

```
233 1422 1 GLOBAL ROUTINE STR$COPY_DX_R8 (      ! Copy string
234 1423 1
235 1424 1     DEST_DESC,      ! Pointer to dest str desc
236 1425 1     SRC_DESC      ! Pointer to input str desc
237 1426 1
238 1427 1     ) : STR$JSB_COPY_DX =
239 1428 1
240 1429 1 ++
241 1430 1 FUNCTIONAL DESCRIPTION:
242 1431 1
243 1432 1     This routine copies a source string to a destination string
244 1433 1     where both the source and destination may be of any supported class.
245 1434 1     It JSBs to the routine which does the actual copy by reference on
246 1435 1     the source.
247 1436 1
248 1437 1 FORMAL PARAMETERS:
249 1438 1
250 1439 1     DEST_DESC.wt.dx     pointer to destination string descriptor
251 1440 1     SRC_DESC.rt.dx      pointer to source string descriptor
252 1441 1
253 1442 1 IMPLICIT INPUTS:
254 1443 1
255 1444 1     NONE
256 1445 1
257 1446 1 IMPLICIT OUTPUTS:
258 1447 1
259 1448 1     NONE
260 1449 1
261 1450 1 COMPLETION CODES:
262 1451 1
263 1452 1     $$$ NORMAL         Success.
264 1453 1     STR$_TRU           Truncation occurred. Warning.
265 1454 1
266 1455 1 SIDE EFFECTS:
267 1456 1
268 1457 1 Will signal STR$_INSVIRMEM if no heap memory to allocate strings or
269 1458 1 STR$_ILLSTRCLA if class in descriptor is not supported.
270 1459 1
271 1460 1 --
272 1461 1
273 1462 2 BEGIN
274 1463 2
275 1464 2 MAP
276 1465 2     SRC_DESC : REF BLOCK [,BYTE];
277 1466 2
278 1467 2 IF .SRC_DESC [DSC$B_CLASS] LEQU DSC$K_CLASS_D
279 1468 2 THEN
280 1469 3     BEGIN
281 1470 4     RETURN (STR$COPY_R_R8 ( .DEST_DESC,
282 1471 4     .SRC_DESC [DSC$W_LENGTH],
283 1472 3     .SRC_DESC [DSC$A_POINTER] ) ) ;
284 1473 3     END
285 1474 2 ELSE
286 1475 2 BEGIN
287 1476 2 LOCAL
288 1477 2     IN_LEN,
289 1478 2     IN_ADDR ;
```



```
296 1484 1 GLOBAL ROUTINE STR$COPY_R_RB (           ! Copy a string
297 1485 1
298 1486 1     DEST_DESC,           ! Pointer to dest str desc
299 1487 1     SRC_LEN,         ! Value of len of src string
300 1488 1     SRC_ADDR         ! Pointer to source string
301 1489 1
302 1490 1           ) : STR$JSB_COPY_R =
303 1491 1
304 1492 1 ++
305 1493 1 FUNCTIONAL DESCRIPTION:
306 1494 1
307 1495 1     This routine copies a source string specified by a length and a
308 1496 1     pointer, to a destination string specified by a descriptor,
309 1497 1     where both the source and destination may be of any class or any
310 1498 1     dtype. This routine uses the macros to prevent ASTs on the source and
311 1499 1     destination strings, and then JSBs to the routine which does
312 1500 1     the actual copy by reference on the source
313 1501 1
314 1502 1 FORMAL PARAMETERS:
315 1503 1
316 1504 1     DEST_DESC.wt.dx     pointer to destination string descriptor
317 1505 1     SRC_LEN.rwu.v      value of source string length
318 1506 1     SRC_ADDR.rt.r      pointer to the source string
319 1507 1
320 1508 1 IMPLICIT INPUTS:
321 1509 1
322 1510 1     NONE
323 1511 1
324 1512 1 IMPLICIT OUTPUTS:
325 1513 1
326 1514 1     NONE
327 1515 1
328 1516 1 COMPLETION CODES:
329 1517 1
330 1518 1     $$$ NORMAL        Success.
331 1519 1     STR$_TRU          Truncation occurred. Warning.
332 1520 1
333 1521 1 SIDE EFFECTS:
334 1522 1
335 1523 1     Will signal STR$ INSVIRMEM if no heap memory to allocate to
336 1524 1     strings, and STR$_ILLSTRCLA if class in descriptor is not one
337 1525 1     supported by SRM
338 1526 1
339 1527 1 --
340 1528 1
341 1529 2     BEGIN
342 1530 2
343 1531 2     LOCAL
344 1532 2     RETURN_STATUS;
345 1533 2     MAP
346 1534 2     DEST_DESC : REF BLOCK [,BYTE],
347 1535 2     SRC_LEN : WORD UNSIGNED ;
348 1536 2 ++
349 1537 2     Select the class of descriptor.
350 1538 2     Return the status resulting from the copy operation.
351 1539 2 --
352 1540 2
```



```

384 1571 3 | +
385 1572 3 | dynamic destination string
386 1573 3 | *****
387 1574 3 | -
388 1575 3 |
389 1576 4 | [DSC$K CLASS_D] :
390 1577 4 | BEGIN
391 1578 5 | IF $STR$NEED_ALLOC (.SRC_LEN,
392 1579 5 | ($STR$DYN_AL_LEN (DEST_DESC)) )
393 1580 5 | %IF %BLISS (BLISS16) OR %BLISS (BLISS36) ! if not VAX must not
394 1581 5 | %THEN ! CH$MOVE with overlap
395 1582 5 | OR $STR$OVERLAP (.SRC_ADDR, SRC_LEN,
396 1583 5 | .DEST_DESC [DSC$A_POINTER], .SRC_LEN)
397 1584 5 | %FI
398 1585 4 | THEN
399 1586 5 | BEGIN ! cannot directly fill dest
400 1587 5 | LOCAL
401 1588 5 | LOC_RET_STAT, ! status of calls to Allocate
402 1589 5 | and Deallocate
403 1590 5 | TEMP_DESC : $STR$DESCRIPTOR; ! create temp
404 1591 5 |
405 1592 5 | LOC_RET_STAT = $STR$ALLOCATE (.SRC_LEN, TEMP_DESC);
406 1593 5 | ! alloc temp
407 1594 5 |
408 1595 5 | +
409 1596 5 | Allocate will only return STR$ NORMAL or
410 1597 5 | STR$ INSVIRMEM, therefore if it wasn't success,
411 1598 5 | don't continue copying
412 1599 5 | -
413 1600 6 | IF (.LOC_RET_STAT?)
414 1601 5 | THEN
415 1602 6 | BEGIN ! successful allocate
416 1603 6 | CH$MOVE (.SRC_LEN, .SRC_ADDR, ! copy to temp
417 1604 6 | .TEMP_DESC [DSC$A_POINTER]);
418 1605 6 | $STR$EXCH_DESCS (TEMP_DESC, DEST_DESC);
419 1606 6 | ! switch temp
420 1607 6 | ! and dest
421 1608 6 | LOC_RET_STAT = $STR$DEALLOCATE (TEMP_DESC);
422 1609 6 | ! return former
423 1610 6 | ! string
424 1611 6 |
425 1612 6 | +
426 1613 6 | $STR$DEALLOCATE returns either STR$ NORMAL
427 1614 6 | or STR$ FATINTERR.
428 1615 6 |
429 1616 6 | IF NOT .LOC_RET_STAT
430 1617 6 | THEN
431 1618 6 | RETURN_STATUS = STR$ FATINTERR ;
432 1619 5 | ! successful allocate
433 1620 5 | ELSE
434 1621 5 | RETURN_STATUS = STR$ INSVIRMEM ;
435 1622 5 | ! cannot directly fill dest
436 1623 4 | END
437 1624 4 | ELSE
438 1625 5 | BEGIN ! directly fill dest
439 1626 5 | CH$MOVE (.SRC_LEN, .SRC_ADDR, ! write dest
440 1627 5 | .DEST_DESC [DSC$A_POINTER]);

```

STRCOPY
1-015

6
16-Sep-1984 01:35:39
14-Sep-1984 12:40:04

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

Page 14
(7)

```
: 441      1628  5  
: 442      1629  4  
: 443      1630  4  
: 444      1631  4  
: 445      1632  3  
: 446      1633  3
```

```
DEST_DESC [DSC$W_LENGTH] = .SRC LEN;  
END;                ! directly fill dest  
  
.RETURN_STATUS      ! return the status  
END;
```

ST
1-

:


```

: 448 1634 3
: 449 1635 3
: 450 1636 3
: 451 1637 3
: 452 1638 3
: 453 1639 3
: 454 1640 4
: 455 1641 4
: 456 1642 4
: 457 1643 4
: 458 1644 4
: 459 1645 4
: 460 1646 4
: 461 1647 4
: 462 1648 4
: 463 1649 4
: 464 1650 4
: 465 1651 4
: 466 1652 4
: 467 1653 4
: 468 1654 4
: 469 1655 4
: 470 1656 3

+
Class A and NCA array descriptor
*****
-
[DSC$K_CLASS_A,      ! Class A Array descriptor
DSC$K_CLASS_NCA]:   ! Class NCA array descriptor
BEGIN
  BUILTIN RO; ! len of uncopied src from MOVCS

  IF .DEST_DESC [DSC$L_ARSIZE] GTR MAX_SIZE ! If size>max
  THEN STR$_ILLSTRCLA ; ! then quit

  CH$COPY (.SRC_LEN, .SRC_ADDR, STR$_FILL_CHAR,
           .DEST_DESC [DSC$L_ARSIZE],
           .DEST_DESC [DSC$A_POINTER]); ! do copy

  IF .RO EQLU 0 ! if no uncopied src
  THEN
    RETURN_STATUS = SSS_NORMAL ! then success
  ELSE
    RETURN_STATUS = STR$_TRU !else truncation

END ; ! of Class A and NCA Array Descriptor
```

```

472 1657 3 !+
473 1658 3 Varying string descriptor
474 1659 3 *****
475 1660 3 -
476 1661 3
477 1662 3 [DSC$K_CLASS_VS]: ! Varying string descriptor
478 1663 3 BEGIN
479 1664 3 IF (.SRC_LEN LEQU .DEST_DESC [DSC$W_MAXSTRLN] )
480 1665 3 THEN ! fits within MAXLEN, copy and update CURLN
481 1666 3 BEGIN
482 1667 3 CH$MOVE (.SRC_LEN, .SRC_ADDR,
483 1668 3 .DEST_DESC [DSC$A_POINTER] + 2);
484 1669 3 (.DEST_DESC [DSC$A_POINTER])<0,16> = .SRC_LEN ;
485 1670 3 $$$_NORMAL ! return success status
486 1671 3 END
487 1672 3
488 1673 3 ELSE ! Won't fit within MAXLEN. Only copy MAXLEN's
489 1674 3 ! worth of data and update CURLN to MAXLEN
490 1675 3
491 1676 3 BEGIN
492 1677 3 CH$MOVE (.DEST_DESC [DSC$W_MAXSTRLN], .SRC_ADDR,
493 1678 3 .DEST_DESC [DSC$A_POINTER] + 2);
494 1679 3 (.DEST_DESC [DSC$A_POINTER])<0,16> =
495 1680 3 .DEST_DESC [DSC$W_MAXSTRLN] ;
496 1681 3 STR$_TRU ! return truncation status
497 1682 3 END
498 1683 3
499 1684 3 END ; ! of Varying string descriptor
500 1685 3
501 1686 3 !+
502 1687 3 Unsupported class descriptor
503 1688 3 *****
504 1689 3 -
505 1690 3
506 1691 3
507 1692 3 [INRANGE, OVRANGE]: ! Unsupported class of descriptor
508 1693 3 STR$_ILLSTRCLA ;
509 1694 3 TES); ! end of set on class code
510 1695 3 $$STR$SIGNAL FATAL (RETURN_STATUS) ;
511 1696 3 RETURN .RETURN_STATUS;
512 1697 3 END; !End of STR$COPY_R_R8

```

```

.EXTRN STR$$INIT, STR$$V_INIT
.EXTRN STR$$ALOC_SHORT
.EXTRN STR$$Q_SHORT_Q, LIB$GET_VM
.EXTRN STR$$MOVQ_R1, LIB$FREE_VM

```

0020	OF	00	03	002A	8F	0000	1\$:	CASEB	3(DEST_DESC), #0, #15	
	003C	002A	0012	1\$:				.WORD	3\$-1\$, =	
		5E	14	C2	0000		STR\$COPY R R8::			
			52	DD	00003		SUBL2	#20, SP		: 1484
		58	51	DD	00005		PUSHL	R2		
		56	50	DD	00008		MOVL	R1, R8		
			01	DD	0000B		MOVL	R0, R6		
			01	DD	0000B		PUSHL	#1		: 1541
			A6	8F	0000D		CASEB	3(DEST_DESC), #0, #15		: 1542
					00012		.WORD	3\$-1\$, =		

0020
01DF
002A

0020
01CD
0020

0020
002A
0020

01CD
0020
0020

0001A
00022
0002A

3\$-1\$,-
5\$-1\$,-
2\$-1\$,-
32\$-1\$,-
2\$-1\$,-
2\$-1\$,-
2\$-1\$,-
2\$-1\$,-
2\$-1\$,-
3\$-1\$,-
32\$-1\$,-
33\$-1\$,-
2\$-1\$,-
2\$-1\$,-
2\$-1\$,-
3\$-1\$

			6E	00000000G	8F	D0	00032	2\$:	MOVL	#STR\$ _ILLSTRCLA, RETURN_STATUS	
					01E2	31	00039		BRW	37\$	
66	20	04	BE		58	2C	0003C	3\$:	MOVCS	SRC_LEN, @SRC_ADDR, #32, (DEST_DESC), -	1562
				04	B6		00042			@4(DEST_DESC)	
					50	D5	00044		TSTL	R0	1564
					03	12	00046		BNEQ	4\$	
					01BB	31	00048		BRW	34\$	
					01C9	31	0004B	4\$:	BRW	36\$	
			51	04	A6	D0	0004E	5\$:	MOVL	4(DEST_DESC), R1	1578
					52	D4	00052		CLRL	R2	
					51	D5	00054		TSTL	R1	
					06	12	00056		BNEQ	6\$	
					52	D6	00058		INCL	R2	
					50	D4	0005A		CLRL	R0	
					13	11	0005C		BRB	8\$	
		00F0	8F		66	B1	0005E	6\$:	CMPW	(DEST_DESC), #240	
					05	1B	00063		BLEQU	7\$	
			50		66	3C	00065		MOVZWL	(DEST_DESC), R0	
					07	11	00068		BRB	8\$	
			50		51	D0	0006A	7\$:	MOVL	R1, STRING_BLOCK	
			50	FE	A0	3C	0006D		MOVZWL	-2(STRING_BLOCK), R0	
		000000F0	8F		50	D1	00071	8\$:	CMP	R0, #240	
					23	1F	00078		BLSSU	12\$	
			04		52	E9	0007A		BLBC	R2, 9\$	
					50	D4	0007D		CLRL	R0	
					13	11	0007F		BRB	11\$	
		00F0	8F		66	B1	00081	9\$:	CMPW	(DEST_DESC), #240	
					05	1B	00086		BLEQU	10\$	
			50		66	3C	00088		MOVZWL	(DEST_DESC), R0	
					07	11	0008B		BRB	11\$	
			50		51	D0	0008D	10\$:	MOVL	R1, STRING_BLOCK	
			50	FE	A0	3C	00090		MOVZWL	-2(STRING_BLOCK), R0	
50	58		10		00	ED	00094	11\$:	CMPZV	#0, #16, SRC_LEN, R0	
					23	13	00099		BEQL	16\$	
					24	11	0009B		BRB	17\$	
			04		52	E9	0009D	12\$:	BLBC	R2, 13\$	
					50	D4	000A0		CLRL	R0	
					13	11	000A2		BRB	15\$	
		00F0	8F		66	B1	000A4	13\$:	CMPW	(DEST_DESC), #240	
					05	1B	000A9		BLEQU	14\$	
			50		66	3C	000AB		MOVZWL	(DEST_DESC), R0	
					07	11	000AE		BRB	15\$	

50		50		51	DO	000B0	14\$:	MOVL	R1, STRING_BLOCK		
		50	FE	A0	3C	000B3		MOVZWL	-2(STRING_BLOCK), R0		
		10		00	ED	000B7	15\$:	CMPZV	#0, #16, SRC_LEN, R0		
				03	1A	000BC		BGTRU	17\$		
				0114	31	000BE	16\$:	BRW	31\$		
		07	00000000G	00	E8	000C1	17\$:	BLBS	STR\$\$V_INIT, 18\$	1592	
		00		00	FB	000C8		CALLS	#0, STR\$\$INIT		
		50	00000000G	8F	DO	000CF	18\$:	MOVL	#STR\$ NORMAL, RETURN_STATUS		
		00F0		8F	B1	000D6		CMPW	SRC_LEN, #240		
				40	1A	000DB		BGTRU	24\$		
				58	B5	000DD		TSTW	SRC_LEN		
				04	12	000DF		BNEQ	19\$		
				53	D4	000E1		CLRL	TEMP		
				2F	11	000E3		BRB	23\$		
		51		58	3C	000E5	19\$:	MOVZWL	SRC_LEN, R1		
				51	D7	000E8		DECL	R1		
		51		07	8A	000EA		BICB2	#7, R1		
		54	00000000G00	41	9E	000ED		MOVAB	STR\$\$Q SHORT Q[R1], REMQUE_ADDR		
		53	00	B4	0F	000F5	20\$:	REMQUE	@(REMQUE_ADDR), TEMP		
				05	1D	000F9		BVS	21\$		
		52		01	DO	000FB		MOVL	#1, ALLOC_DONE		
				0C	11	000FE		BRB	22\$		
				52	D4	00100	21\$:	CLRL	ALLOC_DONE		
		7E		58	3C	00102		MOVZWL	SRC_LEN, -(SP)		
		00000000G		01	FB	00105		CALLS	#1, STR\$\$ALLOC_SHORT		
				52	E8	0010C	22\$:	BLBS	ALLOC_DONE, 23\$		
		05		50	E9	0010F		BLBC	RETURN_STATUS, 26\$		
		2C		E1	11	00112		BRB	20\$		
				50	E9	00114	23\$:	BLBC	RETURN_STATUS, 26\$		
		18	AE	53	DO	00117		MOVL	TEMP, TEMP_DESC+4		
				1D	11	0011B		BRB	25\$		
			18	AE	9F	0011D	24\$:	PUSHAB	TEMP_DESC+4		
		0C	AE	58	3C	00120		MOVZWL	SRC_LEN, 12(SP)		
			0C	AE	9F	00124		PUSHAB	12(SP)		
		00000000G		02	FB	00127		CALLS	#2, LIB\$GET_VM		
				50	E8	0012E		BLBS	RETURN_STATUS, 25\$		
		50	00000000G	8F	DO	00131		MOVL	#STR\$_INSVIRMEM, RETURN_STATUS		
				04	11	00138		BRB	26\$		
		14	AE	58	B0	0013A	25\$:	MOVW	SRC_LEN, TEMP_DESC		
				50	DO	0013E	26\$:	MOVL	RETURN_STATUS, LOC_RET_STAT		
				57	E8	00141		BLBS	LOC_RET_STAT, 27\$	1600	
				0085	31	00144		BRW	30\$		
		18	BE	58	28	00147	27\$:	MOV3	SRC_LEN, @SRC_ADDR, @TEMP_DESC+4	1604	
				0C	AE	0014D		MOVW	(DEST_DESC), \$STR\$TEMP_DESC	1605	
				10	AE	00151		MOVL	4(DEST_DESC), \$STR\$TEMP_DESC+4		
				16	AE	00156		MOVW	2(DEST_DESC), TEMP_DESC+2		
				50	AE	0015B		MOVAB	TEMP_DESC, R0		
				51	DO	0015F		MOVL	DEST_DESC, R1		
				00000000G	00	16	00162	JSB	STR\$\$MOVQ_R1		
				14	AE	B0	00168	MOVW	\$STR\$TEMP_DESC, TEMP_DESC		
				18	AE	DO	0016D	MOVL	\$STR\$TEMP_DESC+4, TEMP_DESC+4		
		50	00000000G	8F	DO	00172		MOVL	#STR\$ NORMAL, RETURN_STATUS	1608	
				52	AE	DO	00179	MOVL	TEMP_DESC+4, R2		
					3E	13	0017D	BEQL	29\$		
		00F0		8F	AE	B1	0017F	CMPW	TEMP_DESC, #240		
					1A	1A	00185	BGTRU	28\$		
				51	DO	00187		MOVL	R2, STRING_BLOCK		

		51	FE	A1	3C	0018A	MOVZWL	-2(STRING_BLOCK), ALLOC_LENGTH						
		51		51	D7	0018E	DECL	R1						
		51		07	8A	00190	BICB2	#7, R1						
		51	00000000G	00	41	9E	00193	MOVAB	STR\$\$Q SHORT Q[R1], INSQUE_ADDR					
		B1			62	0E	0019B	INSQUE	(R2), #0(INSQUE_ADDR)					
					1C	11	0019F	BRB	29\$					
				18	AE	9F	001A1	28\$:	PUSHAB	TEMP_DESC+4				
		OC		18	AE	3C	001A4	MOVZWL	TEMP_DESC, 12(SP)					
				OC	AE	9F	001A9	PUSHAB	12(SP)					
		00000000G			02	FB	001AC	CALLS	#2, LIB\$FREE_VM					
					07	50	E8	001B3	BLBS	RETURN_STATUS, 29\$				
					50	00000000G	8F	D0	001B6	MOVL	#STR\$ FATINTERR, RETURN_STATUS			
					57		50	D0	001BD	29\$:	MOVL	RETURN_STATUS, LOC_RET_STAT		
					5B		57	E8	001C0	BLBS	LOC_RET_STAT, 37\$			
					6E	00000000G	8F	D0	001C3	MOVL	#STR\$ FATINTERR, RETURN_STATUS			
							52	11	001CA	BRB	37\$			
					6E	00000000G	8F	D0	001CC	30\$:	MOVL	#STR\$ INSVIRMEM, RETURN_STATUS		
							49	11	001D3	BRB	37\$			
		61		04	BE		58	28	001D5	31\$:	MOVC3	SRC_LEN, @SRC_ADDR, (R1)		
					66		58	B0	001DA	MOVW	SRC_LEN, (DEST_DESC)			
							3F	11	001DD	BRB	37\$			
		OC	A6	20	04	BE		58	2C	001DF	32\$:	MOVC5	SRC_LEN, @SRC_ADDR, #32, 12(DEST_DESC), -	
							04	B6	001E6			@4(DEST_DESC)		
							50	D5	001E8	TSTL	R0			
							2B	12	001EA	BNEQ	36\$			
					6E		01	D0	001EC	MOVL	#1, RETURN_STATUS			
							15	11	001EF	BRB	34\$			
							50	04	A6	9E	001F1	33\$:	MOVAB	4(DEST_DESC), R0
							66		58	B1	001F5	CMPW	SRC_LEN, (DEST_DESC)	
									11	1A	001F8	BGTRU	35\$	
							57		60	D0	001FA	MOVL	(R0), R7	
		02	A7	04	BE		58	28	001FD	MOVC3	SRC_LEN, @SRC_ADDR, 2(R7)			
					67		58	B0	00203	MOVW	SRC_LEN, (R7)			
					6E		01	D0	00206	34\$:	MOVL	#1, RETURN_STATUS		
							13	11	00209	BRB	37\$			
							57		60	D0	0020B	35\$:	MOVL	(R0), R7
		02	A7	04	BE		66	28	0020E	MOVC3	(DEST_DESC), @SRC_ADDR, 2(R7)			
					67		66	B0	00214	MOVW	(DEST_DESC), (R7)			
					6E	00000000G	8F	D0	00217	36\$:	MOVL	#STR\$ TRU, RETURN_STATUS		
					10		6E	E8	0021E	37\$:	BLBS	RETURN_STATUS, 38\$		
		04		6E	03		00	E0	00221	CMPZV	#0, #3, RETURN_STATUS, #4			
							09	12	00226	BNEQ	38\$			
							6E	DD	00228	PUSHL	RETURN_STATUS			
					00000000G		01	FB	0022A	CALLS	#1, LIB\$STOP			
							50		8E	D0	00231	38\$:	MOVL	RETURN_STATUS, R0
							5E		18	C0	00234	ADDL2	#24, SP	
							05	00237	RSB					

; Routine Size: 568 bytes. Routine Base: _STR\$CODE + 0068

STR\$COPY
1-015

B 7
16-Sep-1984 01:35:39
14-Sep-1984 12:40:04

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

Page 20
(10)

ST
1-

: 514 1698 1 END
: 515 1699 1
: 516 1700 0 ELUDOM

!End of module

PSECT SUMMARY

Name Bytes Attributes
:_STR\$CODE 672 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	17	0	581	00:00.8

COMMAND QUALIFIERS

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

: Size: 672 code + 0 data bytes
: Run Time: 00:12.2
: Elapsed Time: 00:47.6
: Lines/CPU Min: 8340
: Lexemes/CPU-Min: 33895
: Memory Used: 202 pages
: Compilation Complete

