

BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTTTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL

BBBBBBBB	AAAAAA	SSSSSSSS	RRRRRRRR	SSSSSSSS	EEEEEEEEEE	TTTTTTTTTT	
BBBBBBBB	AAAAAA	SSSSSSSS	RRRRRRRR	SSSSSSSS	EEEEEEEEEE	TTTTTTTTTT	
BB	AA	SS	RR	SS	EE	TT	
BB	AA	SS	RR	SS	EE	TT	
BB	AA	SS	RR	SS	EE	TT	
BB	AA	SS	RR	SS	EE	TT	
BBBBBBBB	AA	SSSSSS	RRRRRRRR	SSSSSS	EEEEEEEEEE	TT	
BBBBBBBB	AA	SSSSSS	RRRRRRRR	SSSSSS	EEEEEEEEEE	TT	
BB	AAAAAAAAAA	SS	RR	SS	EE	TT	
BB	AAAAAAAAAA	SS	RR	SS	EE	TT	
BB	AA	SS	RR	SS	EE	TT	
BB	AA	SS	RR	SS	EE	TT	
BBBBBBBB	AA	SSSSSSSS	RR	SSSSSSSS	EEEEEEEEEE	TT	
BBBBBBBB	AA	SSSSSSSS	RR	SSSSSSSS	EEEEEEEEEE	TT	

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


```

1 0001 0 MODULE BASSRSET (
2 0002 0 IDENT = '1-005'
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1 ++
31 0031 1 FACILITY: String support library
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1 Copy one string of any class or dtype to another string of any class
36 0036 1 or dtype. Right justify the source string in the destination string
37 0037 1 using fixed length semantics on the destination string, ie. do not
38 0038 1 alter the current length for dynamic strings (or varying strings if
39 0039 1 they are implemented). Blank pad or truncate on the left.
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: 02-Apr-79
44 0044 1
45 0045 1 MODIFIED BY:
46 0046 1
47 0047 1 R. Will, 02-Apr-79: VERSION 01
48 0048 1 01 - original
49 0049 1 1-002 - Use STR$K_FILL_CHAR. JBS 15-APR-1979
50 0050 1 1-003 - Add string by ref entry. RW 18-JUL-79
51 0051 1 1-004 - String clean up, don't use $STR$ macros. RW 29-Oct-79
52 0052 1 1-005 - Free temp string before exiting. Not doing this caused a
53 0053 1 field test site's large file update to run out of virtual
54 0054 1 storage after 8 hours! JBS 10-MAR-1980
55 0055 1 --
56 0056 1
57 0057 1 !<BLF/PAGE>

```

! Move string right justified, fixed semantics
! File: BASRSET.B32, Edit: JBS1005

```

59 0058 1 !
60 0059 1 ! SWITCHES:
61 0060 1 !
62 0061 1 !
63 0062 1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
64 0063 1 !
65 0064 1 !
66 0065 1 ! LINKAGES:
67 0066 1 !
68 0067 1 !
69 0068 1 REQUIRE 'RTLIN:STRLNK'; ! Use require file with string linkage
70 0253 1 !
71 0254 1 !
72 0255 1 ! TABLE OF CONTENTS:
73 0256 1 !
74 0257 1 !
75 0258 1 FORWARD ROUTINE
76 0259 1     BASRSET : NOVALUE, ! Copy right justified using fixed sem
77 0260 1     BASRSET_R : NOVALUE; ! RSET with string by ref
78 0261 1 !
79 0262 1 !
80 0263 1 ! INCLUDE FILES:
81 0264 1 !
82 0265 1 !
83 0266 1 REQUIRE 'RTLIN:RTLPSECT'; ! Use to declare PSECTs
84 0361 1 !
85 0362 1 LIBRARY 'RTLSTARLE'; ! STARLET library for macros and symb
86 0363 1 !
87 0364 1 !
88 0365 1 ! MACROS: NONE
89 0366 1 !
90 0367 1 !
91 0368 1 ! EQUATED SYMBOLS: NONE
92 0369 1 !
93 0370 1 !
94 0371 1 ! PSECT DECLARATIONS
95 0372 1 !
96 0373 1 DECLARE_PSECTS (BAS);
97 0374 1 !
98 0375 1 ! OWN STORAGE: NONE
99 0376 1 !
100 0377 1 !
101 0378 1 ! EXTERNAL REFERENCES:
102 0379 1 !
103 0380 1 !
104 0381 1 EXTERNAL ROUTINE
105 0382 1     STR$FREE1_DX_R4 : STR$JSB GETFRE, ! deallocate a string
106 0383 1     STR$COPY_DX_R8 : STR$JSB_COPY_DX; ! copy string to and from temp
107 0384 1 !

```



```
109 0385 1 GLOBAL ROUTINE BASSRSET (
110 0386 1     DEST_DESC,
111 0387 1     SRC_DESC)
112 0388 1     : NOVALOE =
113 0389 1
114 0390 1 ++
115 0391 1 FUNCTIONAL DESCRIPTION:
116 0392 1
117 0393 1     This routine copies one string of any class or dtype to
118 0394 1     a second string of any class or dtype but which is treated as
119 0395 1     a fixed length string. The source is right justified in the
120 0396 1     destination with truncation or blank padding on the left.
121 0397 1     This is accomplished by creating a temporary string with
122 0398 1     the attributes of the destination string and using the temporary
123 0399 1     so that the string cannot be moved from under us, since we
124 0400 1     can't use the $STR$ interlock macros.
125 0401 1
126 0402 1 FORMAL PARAMETERS:
127 0403 1
128 0404 1     DEST_DESC.wx.dx     pointer to destination string descriptor
129 0405 1     SRC_DESC.rx.dx      pointer to source string descriptor
130 0406 1
131 0407 1 IMPLICIT INPUTS:
132 0408 1
133 0409 1     NONE
134 0410 1
135 0411 1 IMPLICIT OUTPUTS:
136 0412 1
137 0413 1     NONE
138 0414 1
139 0415 1 ROUTINE VALUE:
140 0416 1 COMPLETION CODES:
141 0417 1
142 0418 1     NONE
143 0419 1
144 0420 1 SIDE EFFECTS:
145 0421 1
146 0422 1     This routine calls STR$COPY and therefore may signal any of its errors.
147 0423 1     It may also allocate and/or deallocate space in the destination string.
148 0424 1     It will allocate and deallocate space for a temporary string.
149 0425 1
150 0426 1 --
151 0427 1
152 0428 2 BEGIN
153 0429 2
154 0430 2 MAP
155 0431 2     DEST_DESC : REF BLOCK [8, BYTE];
156 0432 2
157 0433 2 MAP
158 0434 2     SRC_DESC : REF BLOCK [8, BYTE];
159 0435 2
160 0436 2 LOCAL
161 0437 2     TEMP_DESC : BLOCK [8, BYTE];
162 0438 2
163 0439 2 ++
164 0440 2 Copy the destination string to a temporary so that no one else may
165 0441 2 change its length or pointer if it is a dynamic string.
```

```
166 0442 2 !-
167 0443 2 TEMP_DESC [DSC$W_LENGTH] = 0;
168 0444 2 TEMP_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
169 0445 2 TEMP_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
170 0446 2 TEMP_DESC [DSC$A_POINTER] = 0;
171 0447 2 STR$COPY_DX_R8 (TEMP_DESC [0, 0, 0, 0], DEST_DESC [0, 0, 0, 0]);
172 0448 2 !+
173 0449 2 Treat the temporary destination as a fixed length string
174 0450 2 !-
175 0451 2
176 0452 2 IF .TEMP_DESC [DSC$W_LENGTH] LEQ .SRC_DESC [DSC$W_LENGTH]
177 0453 2 !+
178 0454 2 The source string is longer than the destination
179 0455 2 string. Compute the amount of the source string which must
180 0456 2 be truncated from the left and move the source starting there.
181 0457 2 !-
182 0458 2 THEN
183 0459 2 CH$MOVE (                                ! copy the string
184 0460 2     .TEMP_DESC [DSC$W_LENGTH],          ! move enuf chars to fill dest
185 0461 2     CH$PLUS (                             ! start after trunc in source
186 0462 2     CH$PLUS (.SRC_DESC [DSC$A_POINTER], .SRC_DESC [DSC$W_LENGTH]), -.TEMP_DESC [DSC$W_LENGTH]),
187 0463 2     .TEMP_DESC [DSC$A_POINTER])           ! start of dest string
188 0464 2 !+
189 0465 2 The source string is shorter than or the same length as the
190 0466 2 destination. Fill the number of characters to pad
191 0467 2 and then move the source.
192 0468 2 !-
193 0469 2 ELSE
194 0470 2 CH$MOVE (                                ! fill the temp right justified
195 0471 2     .SRC_DESC [DSC$W_LENGTH],              ! move length of source
196 0472 2     .SRC_DESC [DSC$A_POINTER],            ! start at beginning of source
197 0473 2     CH$FILL (' ',                        ! start move after pad blanks
198 0474 2     .TEMP_DESC [DSC$W_LENGTH] - .SRC_DESC [DSC$W_LENGTH], !
199 0475 2     .TEMP_DESC [DSC$A_POINTER]));
200 0476 2
201 0477 2 !+
202 0478 2 Now put the string in the true destination
203 0479 2 !-
204 0480 2 STR$COPY_DX_R8 (DEST_DESC [0, 0, 0, 0], TEMP_DESC [0, 0, 0, 0]);
205 0481 2 !+
206 0482 2 Free the temporary string.
207 0483 2 !-
208 0484 2 STR$FREE1_DX_R4 (TEMP_DESC [0, 0, 0, 0]);
209 0485 2 END;
210 0486 1 !End of BASSRSET
```

```
.TITLE BASSRSET
.IDENT \1-005\
```

```
.EXTRN STR$FREE1_DX_R4
.EXTRN STR$COPY_DX_R8
```

```
.PSECT _BASSCODE, NOWRT, SHR, PIC, 2
```

```
.ENTRY BASSRSET, Save R2, R3, R4, R5, R6, R7, R8, R9
MOVAB STR$COPY_DX_R8, R9
```

```
03FC 0000
59 00000000G 00 9E 00002
```

```
: 0385
:
```


BASSRSET
1-005

J 15
16-Sep-1984 01:06:20
14-Sep-1984 11:56:37

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASRSET.B32;1

Page 5
(3)

5E		04	C2	00009	SUBL2	#4, SP	:	
	020E0000	8F	DD	0000C	PUSHL	#34471936	:	0443
		04	AE	D4	00012	CLRL	:	0446
50		6E	9E	00015	MOVAB	TEMP_DESC, R0	:	0447
51		04	AC	D0	00018	MOVL	:	
		69	16	0001C	JSB	STR\$COPY_DX_R8	:	
56		08	AC	D0	0001E	MOVL	:	0462
08	BC	6E	B1	00022	CMPW	TEMP_DESC, @SRC_DESC	:	0452
		15	1A	00026	BGTRU	1\$	:	
50		08	BC	3C	00028	MOVZWL	:	0462
50		04	A6	C0	0002C	ADDL2	:	
51			6E	3C	00030	MOVZWL	:	
50			51	C2	00033	SUBL2	:	
	04	BE	6E	28	00036	MOVC3	:	0464
60			17	11	0003B	BRB	:	0462
50			6E	3C	0003D	MOVZWL	:	0475
51		08	BC	3C	00040	MOVZWL	:	
50			51	C2	00044	SUBL2	:	
6E			00	2C	00047	MOVC5	:	0476
		04	BE		0004C		:	
		63	04	B6	08	BC	:	0474
				51	6E	9E	:	0481
			04	AC	D0	00057	:	
			69	16	0005B	JSB	:	
50			6E	9E	0005D	MOVAB	:	0485
	00000000G	00	16	00060	JSB	STR\$FREE1_DX_R4	:	
		04	00066		RET		:	0486

; Routine Size: 103 bytes, Routine Base: _BAS\$CODE + 0000

; 211 0487 1

```
213 0488 1 GLOBAL ROUTINE BASSRSET_R (
214 0489 1     DEST_DESC,
215 0490 1     SRC_LENGTH,
216 0491 1     SRC_POINTER)
217 0492 1     : NOVALUE =
218 0493 1
219 0494 1 ++
220 0495 1 FUNCTIONAL DESCRIPTION:
221 0496 1
222 0497 1     This routine copies one string of any class or dtype to
223 0498 1     a second string of any class or dtype but which is treated as
224 0499 1     a fixed length string. The source is right justified in the
225 0500 1     destination with truncation or blank padding on the left.
226 0501 1
227 0502 1 FORMAL PARAMETERS:
228 0503 1
229 0504 1     DEST_DESC.wx.dx     pointer to destination string descriptor
230 0505 1     SRC_LENGTH.rlu.v    source string length
231 0506 1     SRC_POINTER.rlu.v   pointer to source string
232 0507 1
233 0508 1 IMPLICIT INPUTS:
234 0509 1
235 0510 1     NONE
236 0511 1
237 0512 1 IMPLICIT OUTPUTS:
238 0513 1
239 0514 1     NONE
240 0515 1
241 0516 1 ROUTINE VALUE:
242 0517 1 COMPLETION CODES:
243 0518 1
244 0519 1     NONE
245 0520 1
246 0521 1 SIDE EFFECTS:
247 0522 1
248 0523 1     This routine calls STR$COPY and therefore may signal any of its errors.
249 0524 1     It may also allocate and/or deallocate space in the destination string.
250 0525 1     It will allocate and deallocate space for a temporary string.
251 0526 1
252 0527 1 --
253 0528 1
254 0529 2 BEGIN
255 0530 2
256 0531 2 MAP
257 0532 2     DEST_DESC : REF BLOCK [8, BYTE];
258 0533 2
259 0534 2 LOCAL
260 0535 2     TEMP_DESC : BLOCK [8, BYTE];
261 0536 2
262 0537 2 ++
263 0538 2 Copy the destination string to a temporary to preserve its length, so
264 0539 2 that no one may move the string from under us or change it's length
265 0540 2 while we are processing it.
266 0541 2 --
267 0542 2     TEMP_DESC [DSC$W_LENGTH] = 0;
268 0543 2     TEMP_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
269 0544 2     TEMP_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
```



```
270 0545 2 TEMP_DESC [DSC$A_POINTER] = 0;
271 0546 2 STR$COPY_DX_R8 (TEMP_DESC [0, 0, 0, 0], DEST_DESC [0, 0, 0, 0]);
272 0547 2
273 0548 2 + Treat the destination as a fixed length string
274 0549 2 -
275 0550 2
276 0551 2 IF .TEMP_DESC [DSC$W_LENGTH] LEQ .SRC_LENGTH
277 0552 2 +
278 0553 2 The source string is longer than the destination
279 0554 2 string. Compute the amount of the source string which must
280 0555 2 be truncated from the left and move the source starting there.
281 0556 2 -
282 0557 2 THEN
283 0558 2 CH$MOVE (.TEMP_DESC [DSC$W_LENGTH], ! move enuf chars to fill dest
284 0559 2 .SRC_POINTER + ! start after trunc in source
285 0560 2 .SRC_LENGTH - .TEMP_DESC [DSC$W_LENGTH], .TEMP_DESC [DSC$A_POINTER])
286 0561 2 ! start of dest string
287 0562 2 +
288 0563 2 The source string is shorter than or the same length as the
289 0564 2 destination. Fill the number of characters to pad
290 0565 2 and then move the source.
291 0566 2 -
292 0567 2 ELSE
293 0568 2 CH$MOVE (.SRC_LENGTH, ! move length of source
294 0569 2 .SRC_POINTER, ! start at beginning of source
295 0570 2 CH$FILL (' ', ! start move after pad blanks
296 0571 2 .TEMP_DESC [DSC$W_LENGTH] - .SRC_LENGTH, !
297 0572 2 .TEMP_DESC [DSC$A_POINTER]));
298 0573 2
299 0574 2 +
300 0575 2 Now copy the temporary back into the destination
301 0576 2 -
302 0577 2 STR$COPY_DX_R8 (DEST_DESC [0, 0, 0, 0], TEMP_DESC [0, 0, 0, 0]);
303 0578 2 +
304 0579 2 Free the temporary string.
305 0580 2 -
306 0581 2 STR$FREE1_DX_R4 (TEMP_DESC [0, 0, 0, 0]);
307 0582 1 END; !End of BASSRSET_R
```

					03FC 00000	.ENTRY	BASSRSET_R, Save R2,R3,R4,R5,R6,R7,R8,R9	: 0488
		59	00000000G	00	9E 00002	MOVAB	STR\$COPY_DX_R8, R9	:
		5E		04	C2 00009	SUBL2	#4, SP	:
			020E0000	8F	DD 0000C	PUSHL	#34471936	: 0542
			04	AE	D4 00012	CLRL	TEMP_DESC+4	: 0545
		50		6E	9E 00015	MOVAB	TEMP_DESC, R0	: 0546
		51	04	AC	D0 00018	MOVL	DEST_DESC, R1	:
				69	16 0001C	JSB	STR\$COPY_DX_R8	:
08	AC			00	ED 0001E	CMPZV	#0, #16, TEMP_DESC, SRC_LENGTH	: 0551
				13	14 00024	BGTR	1\$	:
		50	0C	AC	C1 00026	ADDL3	SRC_LENGTH, SRC_POINTER, R0	: 0560
		51		6E	3C 0002C	MOVZWL	TEMP_DESC, R1	:
		50		51	C2 0002F	SUBL2	R1, R0	:
	04	BE		6E	28 00032	MOVC3	TEMP_DESC, (R0), @TEMP_DESC+4	:

BASSRSET
1-005

M 15
16-Sep-1984 01:06:20
14-Sep-1984 11:56:37

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASRSET.B32;1

Page 8
(4)

50	20	6E	08	AC	14	11	00037	BRB	2\$			
		50		6E	3C	00039	1\$:	MOVZWL	TEMP_DESC, R0			0558
		6E		00	C2	0003C		SUBL2	SRC_LENGTH, R0			0571
				04	2C	00040		MOVC5	#0, -(SP), #32, R0, @TEMP_DESC+4			0572
				08	BE	00045						
	63	0C	BC	08	AC	28	00047	MOVC3	SRC_LENGTH, @SRC_POINTER, (R3)			0570
			51	04	6E	9E	0004D	2\$:	MOVAB	TEMP_DESC, R1		0577
			50		AC	D0	00050	MOVL	DEST_DESC, R0			
					69	16	00054	JSB	STR\$COPY_DX_R8			
			50		6E	9E	00056	MOVAB	TEMP_DESC, R0			0581
					00	16	00059	JSB	STR\$FREE1_DX_R4			
					04	0005F		RET				0582

00000000G

; Routine Size: 96 bytes, Routine Base: _BAS\$CODE + 0067

: 308 0583 1
: 309 0584 1 END
: 310 0585 1
: 311 0586 0 ELUDOM

!End of module

PSECT SUMMARY

Name	Bytes	Attributes
_BAS\$CODE	199	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	6	0	581	00:01.1

COMMAND QUALIFIERS

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

; Size: 199 code + 0 data bytes
; Run Time: 00:07.4
; Elapsed Time: 00:17.8
; Lines/CPU Min: 4744
; Lexemes/CPU-Min: 23546

BASSRSET
1-005

N 15
16-Sep-1984 01:06:20

VAX-11 Bliss-32 V4.0-742

Page 9

; Memory Used: 63 pages
; Compilation Complete

0030 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

