

BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTT		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


```
BBBBBBBBB      AAAAAA      SSSSSSSS      RRRRRRRR      SSSSSSSS      TTTTTTTTTT      SSSSSSSS      CCCCCCCC      VV      VV
BBBBBBBBB      AAAAAA      SSSSSSSS      RRRRRRRR      SSSSSSSS      TTTTTTTTTT      SSSSSSSS      CCCCCCCC      VV      VV
BB      BB      AA      AA      SS      RR      RR      SS      TT      SS      CC      VV      VV
BB      BB      AA      AA      SS      RR      RR      SS      TT      SS      CC      VV      VV
BB      BB      AA      AA      SS      RR      RR      SS      TT      SS      CC      VV      VV
BBBBBBBBB      AA      AA      SSSSSS      RRRRRRRR      SSSSSS      TT      SSSSSS      CC      VV      VV
BBBBBBBBB      AA      AA      SSSSSS      RRRRRRRR      SSSSSS      TT      SSSSSS      CC      VV      VV
BB      BB      AAAAAAAAAA      SS      RR      RR      SS      TT      SS      CC      VV      VV
BB      BB      AAAAAAAAAA      SS      RR      RR      SS      TT      SS      CC      VV      VV
BB      BB      AA      AA      SS      RR      RR      SS      TT      SS      CC      VV      VV
BB      BB      AA      AA      SS      RR      RR      SS      TT      SS      CC      VV      VV
BBBBBBBBB      AA      AA      SSSSSSSS      RR      RR      SSSSSSSS      TT      SSSSSSSS      CC      VV      VV
BBBBBBBBB      AA      AA      SSSSSSSS      RR      RR      SSSSSSSS      TT      SSSSSSSS      CC      VV      VV

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
LLLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLLL      IIIIII      SSSSSSSS
```



```
1 0001 0 MODULE BAS$RSTS_CVT (
2 0002 0 IDENT = '1-005'
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1 ++
31 0031 1 FACILITY: BASIC-PLUS-2 Miscellaneous
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1 This module contains the RSTS-compatible CVT functions: CVT$,
36 0036 1 CVT$, CVT$, CVT$, and the double precision versions of
37 0037 1 CVT$ and CVT$.
38 0038 1
39 0039 1 ENVIRONMENT: VAX-11 User Mode
40 0040 1
41 0041 1 AUTHOR: John Sauter, CREATION DATE: 26-FEB-1979
42 0042 1
43 0043 1 MODIFIED BY:
44 0044 1
45 0045 1 1-001 - Original. JBS 26-FEB-1979
46 0046 1 1-002 - If the string provided is too short for the conversion function,
47 0047 1 pad with zeros. JBS 27-FEB-1979
48 0048 1 1-003 - Change LIB$$ and OTS$$ to STR$. JBS 21-MAY-1979
49 0049 1 1-004 - Make BAS$CVT_D_S take its input by value. JBS 20-AUG-1979
50 0050 1 1-005 - Sign-extend the result of BAS$CVT_S_W to 32 bits. JBS 24-SEP-1979
51 0051 1 --
52 0052 1
53 0053 1 !<BLF/PAGE>
```



```

55 0054 1 |
56 0055 1 | SWITCHES:
57 0056 1 |
58 0057 1 |
59 0058 1 | SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
60 0059 1 |
61 0060 1 |
62 0061 1 | LINKAGES:
63 0062 1 |
64 0063 1 |     NONE
65 0064 1 |
66 0065 1 | TABLE OF CONTENTS:
67 0066 1 |
68 0067 1 |
69 0068 1 | FORWARD ROUTINE
70 0069 1 |     BASS$CVT_W_S : NOVALUE,      | Convert word to string
71 0070 1 |     BASS$CVT_S_W,      | Convert string to word
72 0071 1 |     BASS$CVT_F_S : NOVALUE,      | Convert floating to string
73 0072 1 |     BASS$CVT_D_S : NOVALUE,      | Convert double to string
74 0073 1 |     BASS$CVT_S_F,      | Convert string to floating
75 0074 1 |     BASS$CVT_S_D : NOVALUE;      | Convert string to double
76 0075 1 |
77 0076 1 |
78 0077 1 | INCLUDE FILES:
79 0078 1 |
80 0079 1 |
81 0080 1 | REQUIRE 'RTLIN:RTLPSECT';        | Macros for defining psects
82 0175 1 |
83 0176 1 | LIBRARY 'RTLSTARLE';            | System definitions
84 0177 1 |
85 0178 1 |
86 0179 1 | MACROS:
87 0180 1 |
88 0181 1 |     NONE
89 0182 1 |
90 0183 1 | EQUATED SYMBOLS:
91 0184 1 |
92 0185 1 |     NONE
93 0186 1 |
94 0187 1 | PSECTS:
95 0188 1 |
96 0189 1 | DECLARE_PSECTS (BAS);           | Declare psects for BASS$ facility
97 0190 1 |
98 0191 1 | OWN STORAGE:
99 0192 1 |
100 0193 1 |     NONE
101 0194 1 |
102 0195 1 | EXTERNAL REFERENCES:
103 0196 1 |
104 0197 1 |
105 0198 1 | EXTERNAL ROUTINE
106 0199 1 |     BASS$STOP : NOVALUE,      | signals fatal error
107 0200 1 |     STR$GET1_DX,      | Allocate a string
108 0201 1 |     STR$FREED_DX;      | Deallocate a string
109 0202 1 |
```



```
111 0203 1 GLOBAL ROUTINE BASSCVT_W_S (
112 0204 1     STRING_DESC,
113 0205 1     VAL
114 0206 1     ) : NOVALUE =
115 0207 1
116 0208 1 ++
117 0209 1 FUNCTIONAL DESCRIPTION:
118 0210 1
119 0211 1     Changes a word to a string, permuting the bytes in the process.
120 0212 1     There is no justification for permuting the bytes except
121 0213 1     compatability with RSTS/E BASIC-PLUS.
122 0214 1
123 0215 1 FORMAL PARAMETERS:
124 0216 1
125 0217 1     STRING_DESC.wt.d     Descriptor for the result string
126 0218 1     VAL.rw.v             The word to be "converted"
127 0219 1
128 0220 1 IMPLICIT INPUTS:
129 0221 1
130 0222 1     NONE
131 0223 1
132 0224 1 IMPLICIT OUTPUTS:
133 0225 1
134 0226 1     NONE
135 0227 1
136 0228 1 ROUTINE VALUE:
137 0229 1 COMPLETION CODES:
138 0230 1
139 0231 1     NONE
140 0232 1
141 0233 1 SIDE EFFECTS:
142 0234 1
143 0235 1     NONE
144 0236 1
145 0237 1 --
146 0238 1
147 0239 2 BEGIN
148 0240 2
149 0241 2 MAP
150 0242 2     VAL : VECTOR [2, BYTE],
151 0243 2     STRING_DESC : REF BLOCK [8, BYTE];
152 0244 2
153 0245 2 LOCAL
154 0246 2     STRING : REF VECTOR [65535, BYTE];
155 0247 2
156 0248 2 +
157 0249 2 Be sure the result string has only enough storage to hold the
158 0250 2 two bytes we will be putting in it.
159 0251 2 -
160 0252 2     STR$FREE1_DX (.STRING_DESC);
161 0253 2     STR$GET1_DX (%REF (2), .STRING_DESC);
162 0254 2 +
163 0255 2 Now permute the bytes, putting the word into the string.
164 0256 2 -
165 0257 2     STRING = .STRING_DESC [DSC$A_POINTER];
166 0258 2     STRING [0] = .VAL [1];
167 0259 2     STRING [1] = .VAL [0];
```


F 16
16-Sep-1984 01:06:52 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 11:56:37 [BASRTL.SRC]BASRSTSCV.B32;1

Page 4
(3)

```
! end of BAS$CVT_W_S
```

```
.TITLE  BAS$RSTS_CVT
.IDENT  \1-005\
```

```

EXTRN  BAS$$STOP, STR$GET1_DX
EXTRN  STR$FREE1_DX

```

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

```

.ENTRY  BAS$CVT_W_S, Save R2

```

```

SUBL2    #4, SP
MOVL     STRING_DESC, R2

```

```

MOVE     STRING_DESC, R2
PUSHL    R2
CALLS    #1, STR$FREE1_DX

```

```
CALLS #1, STRSFREET_DA
PUSHL R2
MOVL #2, 4(SP)
```

```

MOVLE    #2, 4(SP)
PUSHAB   4(SP)
CALLS    #2, STR$GET1, DX

```

```
CALLS #2, STR$GETT DX
MOVL 4(R2), STRING
MOVB VAL+1, (STRING)
```

```

MOVB    VAL+1, (STRING)
MOVB    VAL, 1(SSTRING)
RET

```

RET

				0004	00000
	5E		04	C2	00002
	52	04	AC	D0	00005
			52	DD	00009
000000000G	00		01	FB	0000B
			52	DD	00012
04	AE		02	D0	00014
		04	AE	9F	00018
000000000G	00		02	FB	0001B
	50	04	A2	D0	00022
	60	09	AC	90	00026
01	A0	08	AC	90	0002A
				04	0002F

```
; Routine Size: 48 bytes,    Routine Base: _BAS$CODE + 0000
```

: 170 0262 1


```

172 0263 1 GLOBAL ROUTINE BASSCVT_S_W ( ! Convert string to word
173 0264 1 STRING_DESC ! Descriptor for string
174 0265 1 ) =
175 0266 1
176 0267 1 !++
177 0268 1 FUNCTIONAL DESCRIPTION:
178 0269 1
179 0270 1 Changes a string to a word, permuting the bytes in the process.
180 0271 1 There is no justification for permuting the bytes except
181 0272 1 compatability with RSTS/E BASIC-PLUS.
182 0273 1
183 0274 1 FORMAL PARAMETERS:
184 0275 1
185 0276 1 STRING_DESC.rt.d Descriptor for the string
186 0277 1
187 0278 1 IMPLICIT INPUTS:
188 0279 1
189 0280 1 NONE
190 0281 1
191 0282 1 IMPLICIT OUTPUTS:
192 0283 1
193 0284 1 NONE
194 0285 1
195 0286 1 ROUTINE VALUE:
196 0287 1
197 0288 1 The resultant 16-bit word.
198 0289 1
199 0290 1 SIDE EFFECTS:
200 0291 1
201 0292 1 NONE
202 0293 1
203 0294 1 --
204 0295 1
205 0296 2 BEGIN
206 0297 2
207 0298 2 MAP
208 0299 2 STRING_DESC : REF BLOCK [8, BYTE];
209 0300 2
210 0301 2 LOCAL
211 0302 2 RESULT : VECTOR [4, BYTE, SIGNED],
212 0303 2 STRING : REF VECTOR [, BYTE, SIGNED];
213 0304 2
214 0305 2 !+
215 0306 2 Permute the bytes, putting the string into the word.
216 0307 2 !-
217 0308 2 STRING = .STRING_DESC [DSC$A_POINTER];
218 0309 2 RESULT = 0;
219 0310 2 RESULT [0] = (IF (.STRING_DESC [DSC$W_LENGTH] LEQU 1) THEN 0 ELSE .STRING [1]);
220 0311 2 RESULT [1] = (IF (.STRING_DESC [DSC$W_LENGTH] LEQU 0) THEN 0 ELSE .STRING [0]);
221 0312 2 !+
222 0313 2 Sign-extend the result, so if the caller is storing the result in a longword a
223 0314 2 sign test will still work.
224 0315 2 !-
225 0316 2 RESULT [2] = (IF (.RESULT [1] LSS 0) THEN -1 ELSE 0);
226 0317 2 RESULT [3] = (IF (.RESULT [1] LSS 0) THEN -1 ELSE 0);
227 0318 2 RETURN (.RESULT);
228 0319 1 END; ! end of BASSCVT_W_S

```


			0004 00000	.ENTRY	BASSCVT S.W, Save R2	: 0263
52	04	AC	D0 00002	MOVL	STRING_DESC, R2	: 0308
50	04	A2	D0 00006	MOVL	4(R2), -STRING	
		7E	D4 0000A	CLRL	RESULT	: 0309
01		62	B1 0000C	CMPW	(R2), #1	: 0310
		04	1A 0000F	BGTRU	1\$	
		51	D4 00011	CLRL	R1	
		04	11 00013	BRB	2\$	
51	01	A0	98 00015	CVTBL	1(String), R1	
6E		51	90 00019	MOVB	R1, RESULT	
		62	B5 0001C	TSTW	(R2)	: 0311
		04	12 0001E	BNEQ	3\$	
		50	D4 00020	CLRL	R0	
		03	11 00022	BRB	4\$	
		60	98 00024	CVTBL	(String), R0	
01	50	50	90 00027	MOVB	R0, RESULT+1	
AE		05	18 0002B	BGEQ	5\$	: 0316
	50	01	CE 0002D	MNEGL	#1, R0	
		02	11 00030	BRB	6\$	
		50	D4 00032	CLRL	R0	
02	AE	50	90 00034	MOVB	R0, RESULT+2	
		AE	95 00038	TSTB	RESULT+1	: 0317
		05	18 0003B	BGEQ	7\$	
	50	01	CE 0003D	MNEGL	#1, R0	
		02	11 00040	BRB	8\$	
		50	D4 00042	CLRL	R0	
03	AE	50	90 00044	MOVB	R0, RESULT+3	
	50	6E	D0 00048	MOVL	RESULT, R0	: 0318
			04 0004B	RET		: 0319

: Routine Size: 76 bytes, Routine Base: _BASSCODE + 0030

: 229 0320 1


```
231 0321 1 GLOBAL ROUTINE BASSCVT_F_S (
232 0322 1     STRING_DESC,
233 0323 1     VAL
234 0324 1     ) : NOVALUE =
235 0325 1
236 0326 1 ++
237 0327 1 FUNCTIONAL DESCRIPTION:
238 0328 1
239 0329 1     Changes a floating value to a string, permuting the bytes in the process.
240 0330 1     There is no justification for permuting the bytes except
241 0331 1     compatability with RSTS/E BASIC-PLUS.
242 0332 1
243 0333 1 FORMAL PARAMETERS:
244 0334 1
245 0335 1     STRING_DESC.wt.d     Descriptor for the result string
246 0336 1     VAL.rf.v             The floating value to be "converted"
247 0337 1
248 0338 1 IMPLICIT INPUTS:
249 0339 1
250 0340 1     NONE
251 0341 1
252 0342 1 IMPLICIT OUTPUTS:
253 0343 1
254 0344 1     NONE
255 0345 1
256 0346 1 ROUTINE VALUE:
257 0347 1 COMPLETION CODES:
258 0348 1
259 0349 1     NONE
260 0350 1
261 0351 1 SIDE EFFECTS:
262 0352 1
263 0353 1     NONE
264 0354 1
265 0355 1 --
266 0356 1
267 0357 2 BEGIN
268 0358 2
269 0359 2 MAP
270 0360 2     VAL : VECTOR [4, BYTE],
271 0361 2     STRING_DESC : REF BLOCK [8, BYTE];
272 0362 2
273 0363 2 LOCAL
274 0364 2     STRING : REF VECTOR [65535, BYTE];
275 0365 2
276 0366 2 ++
277 0367 2 Be sure the result string has only enough storage to hold the
278 0368 2 four bytes we will be putting in it.
279 0369 2 --
280 0370 2     STR$FREE1_DX (.STRING_DESC);
281 0371 2     STR$GET1_DX (%REF (4), .STRING_DESC);
282 0372 2 ++
283 0373 2 Now permute the bytes, putting the floating value into the string.
284 0374 2 --
285 0375 2     STRING = .STRING_DESC [DSC$A_POINTER];
286 0376 2     STRING [0] = .VAL [3];
287 0377 2     STRING [1] = .VAL [2];
```


BASSRSTS_CVT
1-005

J 16
16-Sep-1984 01:06:52
14-Sep-1984 11:56:37

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

Page 8
(5)

```
: 288      0378 2   STRING [2] = .VAL [1];
: 289      0379 2   STRING [3] = .VAL [0];
: 290      0380 2   RETURN;
: 291      0381 1   END;
```

! end of BASSCVT_F_S

```
                                0004 00000
                                04  C2 00002
                                04  AC  D0 00005
                                52  DD 00009
00000000G 00                   01  FB 0000B
                                52  DD 00012
                                04  D0 00014
                                04  AE  9F 00018
00000000G 00                   02  FB 0001B
                                04  A2  D0 00022
                                0B  AC  90 00026
                                01  A0  0A  AC  90 0002A
                                02  A0  09  AC  90 0002F
                                03  A0  08  AC  90 00034
                                04 00039
```

```
.ENTRY BASSCVT_F_S, Save R2
SUBL2  #4, SP
MOVL   STRING_DESC, R2
PUSHL  R2
CALLS  #1, STR$FREE1_DX
PUSHL  R2
MOVL   #4, 4(SP)
PUSHAB 4(SP)
CALLS  #2, STR$GET1_DX
MOVL   4(R2), STRING
MOVB   VAL+3, (STRING)
MOVB   VAL+2, 1(STRING)
MOVB   VAL+1, 2(STRING)
MOVB   VAL, 3(STRING)
RET
```

```
: 0321
:
: 0370
:
: 0371
:
:
: 0375
: 0376
: 0377
: 0378
: 0379
: 0381
```

; Routine Size: 58 bytes, Routine Base: _BAS\$CODE + 007C

; 292 0382 1


```
294 0383 1 GLOBAL ROUTINE BASSCVT_D_S (      ! Convert double to string
295 0384 1     STRING_DESC,                  ! Descriptor for returned string
296 0385 1     VAL                          ! Double number to convert
297 0386 1 ) : NOVALUE =
298 0387 1
299 0388 1 ++
300 0389 1 FUNCTIONAL DESCRIPTION:
301 0390 1
302 0391 1     Changes a double value to a string, permuting the bytes in the process.
303 0392 1     There is no justification for permuting the bytes except
304 0393 1     compatability with RSTS/E BASIC-PLUS.
305 0394 1
306 0395 1 FORMAL PARAMETERS:
307 0396 1
308 0397 1     STRING_DESC.wt.d      Descriptor for the result string
309 0398 1     VAL.rd.v              The double value to be "converted"
310 0399 1
311 0400 1 IMPLICIT INPUTS:
312 0401 1
313 0402 1     NONE
314 0403 1
315 0404 1 IMPLICIT OUTPUTS:
316 0405 1
317 0406 1     NONE
318 0407 1
319 0408 1 ROUTINE VALUE:
320 0409 1 COMPLETION CODES:
321 0410 1
322 0411 1     NONE
323 0412 1
324 0413 1 SIDE EFFECTS:
325 0414 1
326 0415 1     NONE
327 0416 1
328 0417 1 --
329 0418 1
330 0419 2 BEGIN
331 0420 2
332 0421 2 MAP
333 0422 2     VAL : VECTOR [8, BYTE],
334 0423 2     STRING_DESC : REF BLOCK [8, BYTE];
335 0424 2
336 0425 2 LOCAL
337 0426 2     STRING : REF VECTOR [65535, BYTE];
338 0427 2
339 0428 2 +
340 0429 2 Be sure the result string has only enough storage to hold the
341 0430 2 eight bytes we will be putting in it.
342 0431 2 -
343 0432 2     STR$FREE1 DX (.STRING_DESC);
344 0433 2     STR$GET1_DX (%REF (8), .STRING_DESC);
345 0434 2 +
346 0435 2 Now permute the bytes, putting the double value into the string.
347 0436 2 -
348 0437 2     STRING = .STRING_DESC [DSC$A_POINTER];
349 0438 2     STRING [0] = .VAL [7];
350 0439 2     STRING [1] = .VAL [6];
```


BASSRSTS_CVT
1-005

L 16
16-Sep-1984 01:06:52
14-Sep-1984 11:56:37

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

Page 10
(6)

```
: 351      0440  2  STRING [2] = .VAL [5]:  
: 352      0441  2  STRING [3] = .VAL [4]:  
: 353      0442  2  STRING [4] = .VAL [3]:  
: 354      0443  2  STRING [5] = .VAL [2]:  
: 355      0444  2  STRING [6] = .VAL [1]:  
: 356      0445  2  STRING [7] = .VAL [0]:  
: 357      0446  2  RETURN;  
: 358      0447  1  END;
```

! end of BASSCVT_D_S

```
0004 00000  
5E      04  C2 00002  
52      04  AC  D0 00005  
00000000G 00  52  DD 00009  
04      AE      01  FB 0000B  
00000000G 00  52  DD 00012  
04      AE      08  D0 00014  
04      AE      08  9F 00018  
00000000G 00  02  FB 0001B  
50      04  A2  D0 00022  
60      0F  AC  90 00026  
01      A0      0E  AC  90 0002A  
02      A0      0D  AC  90 0002F  
03      A0      0C  AC  90 00034  
04      A0      0B  AC  90 00039  
05      A0      0A  AC  90 0003E  
06      A0      09  AC  90 00043  
07      A0      08  AC  90 00048  
04      04  0004D
```

```
.ENTRY BASSCVT_D_S, Save R2  
SUBL2  #4, SP  
MOVL   STRING_DESC, R2  
PUSHL  R2  
CALLS  #1, STR$FREE1_DX  
PUSHL  R2  
MOVL   #8, 4(SP)  
PUSHAB 4(SP)  
CALLS  #2, STR$GET1_DX  
MOVL   4(R2), STRING  
MOVB   VAL+7, (STRING)  
MOVB   VAL+6, 1(STRING)  
MOVB   VAL+5, 2(STRING)  
MOVB   VAL+4, 3(STRING)  
MOVB   VAL+3, 4(STRING)  
MOVB   VAL+2, 5(STRING)  
MOVB   VAL+1, 6(STRING)  
MOVB   VAL, 7(STRING)  
RET
```

```
: 0383  
: 0432  
: 0433  
: 0437  
: 0438  
: 0439  
: 0440  
: 0441  
: 0442  
: 0443  
: 0444  
: 0445  
: 0447
```

; Routine Size: 78 bytes, Routine Base: _BASSCODE + 00B6

; 359 0448 1


```

361 0449 1 GLOBAL ROUTINE BASSCVT_S_F ( ! Convert string to floating
362 0450 1 STRING_DESC ! Descriptor for string
363 0451 1 ) =
364 0452 1
365 0453 1 !++
366 0454 1 FUNCTIONAL DESCRIPTION:
367 0455 1
368 0456 1 Changes a string to a floating value, permuting the bytes in the process.
369 0457 1 There is no justification for permuting the bytes except
370 0458 1 compatability with RSTS/E BASIC-PLUS.
371 0459 1
372 0460 1 FORMAL PARAMETERS:
373 0461 1
374 0462 1 STRING_DESC.rt.d Descriptor for the string
375 0463 1
376 0464 1 IMPLICIT INPUTS:
377 0465 1
378 0466 1 NONE
379 0467 1
380 0468 1 IMPLICIT OUTPUTS:
381 0469 1
382 0470 1 NONE
383 0471 1
384 0472 1 ROUTINE VALUE:
385 0473 1 COMPLETION CODES:
386 0474 1
387 0475 1 The resultant floating value.
388 0476 1
389 0477 1 SIDE EFFECTS:
390 0478 1
391 0479 1 NONE
392 0480 1
393 0481 1 --
394 0482 1
395 0483 2 BEGIN
396 0484 2
397 0485 2 MAP
398 0486 2 STRING_DESC : REF BLOCK [8, BYTE];
399 0487 2
400 0488 2 LOCAL
401 0489 2 RESULT : VECTOR [4, BYTE],
402 0490 2 STRING : REF VECTOR [, BYTE];
403 0491 2
404 0492 2 !+
405 0493 2 ! Permute the bytes, putting the string into the floating result.
406 0494 2 !-
407 0495 2 STRING = .STRING_DESC [DSC$A POINTER];
408 0496 2 RESULT [0] = (IF (.STRING_DESC [DSC$W_LENGTH] LEQU 3) THEN 0 ELSE .STRING [3]);
409 0497 2 RESULT [1] = (IF (.STRING_DESC [DSC$W_LENGTH] LEQU 2) THEN 0 ELSE .STRING [2]);
410 0498 2 RESULT [2] = (IF (.STRING_DESC [DSC$W_LENGTH] LEQU 1) THEN 0 ELSE .STRING [1]);
411 0499 2 RESULT [3] = (IF (.STRING_DESC [DSC$W_LENGTH] LEQU 0) THEN 0 ELSE .STRING [0]);
412 0500 2 RETURN (.RESULT);
413 0501 1 END; ! end of BASSCVT_F_S

```


			0004 00000	.ENTRY	BASSCVT_S_F, Save R2		0449
	5E		04 C2 00002	SUBL2	#4, SP		
	52	04	AC D0 00005	MOVL	STRING_DESC, R2		0495
	50	04	A2 D0 00009	MOVL	4(R2), STRING		
	03		62 B1 0000D	CMPW	(R2), #3		0496
			04 1A 00010	BGTRU	1\$		
			51 D4 00012	CLRL	R1		
			04 11 00014	BRB	2\$		
	51	03	A0 9A 00016 1\$:	MOVZBL	3(STRING), R1		
	6E		51 90 0001A 2\$:	MOVB	R1, RESULT		
	02		62 B1 0001D	CMPW	(R2), #2		0497
			04 1A 00020	BGTRU	3\$		
			51 D4 00022	CLRL	R1		
			04 11 00024	BRB	4\$		
	51	02	A0 9A 00026 3\$:	MOVZBL	2(STRING), R1		
01	AE		51 90 0002A 4\$:	MOVB	R1, RESULT+1		
	01		62 B1 0002E	CMPW	(R2), #1		0498
			04 1A 00031	BGTRU	5\$		
			51 D4 00033	CLRL	R1		
			04 11 00035	BRB	6\$		
	51	01	A0 9A 00037 5\$:	MOVZBL	1(STRING), R1		
02	AE		51 90 0003B 6\$:	MOVB	R1, RESULT+2		
			62 B5 0003F	TSTW	(R2)		0499
			04 12 00041	BNEQ	7\$		
			50 D4 00043	CLRL	R0		
			03 11 00045	BRB	8\$		
	50		60 9A 00047 7\$:	MOVZBL	(STRING), R0		
03	AE		50 90 0004A 8\$:	MOVB	R0, RESULT+3		
	50		6E D0 0004E	MOVL	RESULT, R0		0500
			04 00051	RET			0501

; Routine Size: 82 bytes, Routine Base: _BASSCODE + 0104

; 414 0502 1


```
: 416      0503 1 GLOBAL ROUTINE BASS$CVT_S_D (      ! Convert string to double
: 417      0504 1      STRING_DESC                ! Descriptor for string
: 418      0505 1      ) : NOVALUE =
: 419      0506 1
: 420      0507 1  !++
: 421      0508 1  FUNCTIONAL DESCRIPTION:
: 422      0509 1
: 423      0510 1      Changes a string to a double value, permuting the bytes in the process.
: 424      0511 1      There is no justification for permuting the bytes except
: 425      0512 1      compatability with RSTS/E BASIC-PLUS.
: 426      0513 1
: 427      0514 1  FORMAL PARAMETERS:
: 428      0515 1
: 429      0516 1      STRING_DESC.rt.d      Descriptor for the string
: 430      0517 1
: 431      0518 1  IMPLICIT INPUTS:
: 432      0519 1
: 433      0520 1      NONE
: 434      0521 1
: 435      0522 1  IMPLICIT OUTPUTS:
: 436      0523 1
: 437      0524 1      NONE
: 438      0525 1
: 439      0526 1  ROUTINE VALUE:
: 440      0527 1  COMPLETION CODES:
: 441      0528 1
: 442      0529 1      The resultant double value.
: 443      0530 1
: 444      0531 1  SIDE EFFECTS:
: 445      0532 1
: 446      0533 1      NONE
: 447      0534 1
: 448      0535 1  --
: 449      0536 1
: 450      0537 2      BEGIN
: 451      0538 2
: 452      0539 2      MAP
: 453      0540 2      STRING_DESC : REF BLOCK [8, BYTE];
: 454      0541 2
: 455      0542 2      LOCAL
: 456      0543 2      RESULT : VECTOR [8, BYTE],
: 457      0544 2      STRING : REF VECTOR [, BYTE];
: 458      0545 2
: 459      0546 2  !+
: 460      0547 2  !- Permute the bytes, putting the string into the floating result.
: 461      0548 2
: 462      0549 2      STRING = .STRING_DESC [DSC$A POINTER];
: 463      0550 2      RESULT [0] = (IF (.STRING_DESC [DSC$W_LENGTH] LEQU 7) THEN 0 ELSE .STRING [7]);
: 464      0551 2      RESULT [1] = (IF (.STRING_DESC [DSC$W_LENGTH] LEQU 6) THEN 0 ELSE .STRING [6]);
: 465      0552 2      RESULT [2] = (IF (.STRING_DESC [DSC$W_LENGTH] LEQU 5) THEN 0 ELSE .STRING [5]);
: 466      0553 2      RESULT [3] = (IF (.STRING_DESC [DSC$W_LENGTH] LEQU 4) THEN 0 ELSE .STRING [4]);
: 467      0554 2      RESULT [4] = (IF (.STRING_DESC [DSC$W_LENGTH] LEQU 3) THEN 0 ELSE .STRING [3]);
: 468      0555 2      RESULT [5] = (IF (.STRING_DESC [DSC$W_LENGTH] LEQU 2) THEN 0 ELSE .STRING [2]);
: 469      0556 2      RESULT [6] = (IF (.STRING_DESC [DSC$W_LENGTH] LEQU 1) THEN 0 ELSE .STRING [1]);
: 470      0557 2      RESULT [7] = (IF (.STRING_DESC [DSC$W_LENGTH] LEQU 0) THEN 0 ELSE .STRING [0]);
: 471      0558 2  !+
: 472      0559 2  ! Returning a double precision number is a little tricky in BLISS.
```



```

: 473      0560 2  !-
: 474      0561
: 475      0562 BEGIN
: 476      0563 REGISTER
: 477      0564 R0 = 0;
: 478      0565 R1 = 1;
: 479      0566
: 480      0567 MAP
: 481      0568 RESULT : BLOCK [8, BYTE];
: 482      0569
: 483      0570 R0 = .RESULT [0, 0, %BPVAL, 0];
: 484      0571 R1 = .RESULT [%UPVAL, 0, %BPVAL, 0];
: 485      0572 RETURN;
: 486      0573 END;
: 487      0574 1  END;

```

! end of BASSCVT_F_S

			0004 00000	.ENTRY	BASSCVT_S_D, Save R2	: 0503
	5E		08 C2 00002	SUBL2	#8, SP	: 0549
	52	04	AC D0 00005	MOVL	STRING_DESC, R2	: 0550
	50	04	A2 D0 00009	MOVL	4(R2), -STRING	
	07		62 B1 0000D	CMPW	(R2), #7	
			04 1A 00010	BGTRU	1\$	
			51 D4 00012	CLRL	R1	
			04 11 00014	BRB	2\$	
	51	07	A0 9A 00016 1\$:	MOVZBL	7(STRING), R1	
	6E		51 90 0001A 2\$:	MOVB	R1, RESULT	
	06		62 B1 0001D	CMPW	(R2), #6	: 0551
			04 1A 00020	BGTRU	3\$	
			51 D4 00022	CLRL	R1	
			04 11 00024	BRB	4\$	
	51	06	A0 9A 00026 3\$:	MOVZBL	6(STRING), R1	
01	AE		51 90 0002A 4\$:	MOVB	R1, RESULT+1	
	05		62 B1 0002E	CMPW	(R2), #5	: 0552
			04 1A 00031	BGTRU	5\$	
			51 D4 00033	CLRL	R1	
			04 11 00035	BRB	6\$	
	51	05	A0 9A 00037 5\$:	MOVZBL	5(STRING), R1	
02	AE		51 90 0003B 6\$:	MOVB	R1, RESULT+2	
	04		62 B1 0003F	CMPW	(R2), #4	: 0553
			04 1A 00042	BGTRU	7\$	
			51 D4 00044	CLRL	R1	
			04 11 00046	BRB	8\$	
	51	04	A0 9A 00048 7\$:	MOVZBL	4(STRING), R1	
03	AE		51 90 0004C 8\$:	MOVB	R1, RESULT+3	
	03		62 B1 00050	CMPW	(R2), #3	: 0554
			04 1A 00053	BGTRU	9\$	
			51 D4 00055	CLRL	R1	
			04 11 00057	BRB	10\$	
	51	03	A0 9A 00059 9\$:	MOVZBL	3(STRING), R1	
04	AE		51 90 0005D 10\$:	MOVB	R1, RESULT+4	
	02		62 B1 00061	CMPW	(R2), #2	: 0555
			04 1A 00064	BGTRU	11\$	
			51 D4 00066	CLRL	R1	

05	51 AE 01	02	04 11 00068 A0 9A 0006A 11\$: 51 90 0006E 12\$: 62 B1 00072 04 1A 00075 51 D4 00077 04 11 00079	BRB 12\$ MOVZBL 2(String), R1 MOVB R1, RESULT+5 CMPW (R2), #1 BGTRU 13\$ CLRL R1 BRB 14\$
06	51 AE	01	A0 9A 0007B 13\$: 51 90 0007F 14\$: 62 B5 00083 04 12 00085 50 D4 00087 03 11 00089	MOVZBL 1(String), R1 MOVB R1, RESULT+6 TSTW (R2) BNEQ 15\$ CLRL R0 BRB 16\$
07	50 AE 50		60 9A 0008B 15\$: 50 90 0008E 16\$: 6E 7D 00092 04 00095	MOVZBL (String), R0 MOVB R0, RESULT+7 MOVQ RESULT, R0 RET

0556

0557

0570

0574

; Routine Size: 150 bytes, Routine Base: _BAS\$CODE + 0156

: 488 0575 1
: 489 0576 1 END
: 490 0577 1
: 491 0578 0 ELUDOM

! end of module BASSRSTS_CVT

PSECT SUMMARY

Name	Bytes	Attributes
_BAS\$CODE	492	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_S255\$DUA28:[SYSLIB]STARLET.L32;1	9776	2	0	581	00:01.0

COMMAND QUALIFIERS

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

BASRSTS_CVT
1-005

F 1
16-Sep-1984 01:06:52
14-Sep-1984 11:56:37

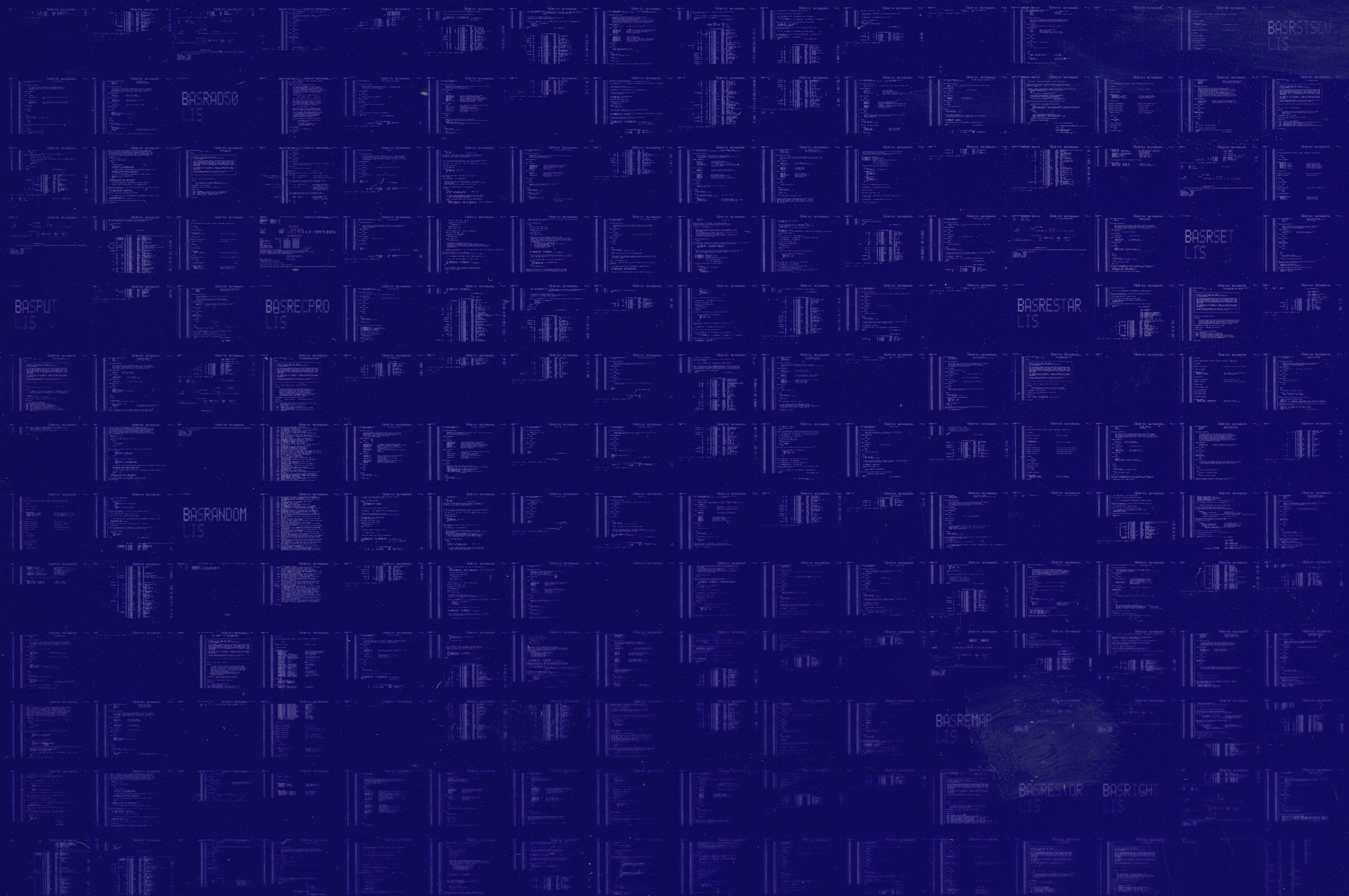
VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASRSTSCV.B32;1

Page 16
(8)

: Size: 492 code + 0 data bytes
: Run Time: 00:11.6
: Elapsed Time: 00:26.2
: Lines/CPU Min: 3000
: Lexemes/CPU-Min: 21498
: Memory Used: 72 pages
: Compilation Complete

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0031

AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY