

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	SSSSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLLLLLL

```
BBBBBBBBB      AAAAAA      SSSSSSSS      RRRRRRRR      TTTTTTTTTT      DDDDDDDD      IIIIII      MM      MM
BBBBBBBBB      AAAAAA      SSSSSSSS      RRRRRRRR      TTTTTTTTTT      DDDDDDDD      IIIIII      MM      MM
BB      BB      AA      AA      SS      SS      RR      RR      TT      DD      DD      II      MMMM      MMMM
BB      BB      AA      AA      SS      SS      RR      RR      TT      DD      DD      II      MMMM      MMMM
BB      BB      AA      AA      SS      SS      RR      RR      TT      DD      DD      II      MM      MM
BBBBBBBBB      AA      AA      SSSSSS      RRRRRRRR      TT      DD      DD      II      MM      MM
BBBBBBBBB      AA      AA      SSSSSS      RRRRRRRR      TT      DD      DD      II      MM      MM
BB      BB      AAAAAAAAAA      SS      RR      RR      TT      DD      DD      II      MM      MM
BB      BB      AAAAAAAAAA      SS      RR      RR      TT      DD      DD      II      MM      MM
BB      BB      AA      AA      SS      SS      RR      RR      TT      DD      DD      II      MM      MM
BB      BB      AA      AA      SSSSSSSS      RR      RR      TT      DDDDDDDD      IIIIII      MM      MM
BBBBBBBBB      AA      AA      SSSSSSSS      RR      RR      TT      DDDDDDDD      IIIIII      MM      MM

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 BASRT_DIM (
2 0002 0 IDENT = '1-011'
3 0003 0 ) =
4 0004 0
5 0005 0 *****
6 0006 0 *
7 0007 0 *
8 0008 0 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 0 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 0 * ALL RIGHTS RESERVED.
11 0011 0 *
12 0012 0 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 0 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 0 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 0 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 0 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 0 * TRANSFERRED.
18 0018 0 *
19 0019 0 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 0 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 0 * CORPORATION.
22 0022 0 *
23 0023 0 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 0 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 0 *
26 0026 0 *****
27 0027 0
28 0028 1 BEGIN
29 0029 1
30 0030 1 ++
31 0031 1 FACILITY: Basic Language Support
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1 This routine is called by the generated code to dimension an
36 0036 1 array at run-time.
37 0037 1
38 0038 1 ENVIRONMENT: Runs at any access mode - AST reentrant
39 0039 1
40 0040 1 AUTHOR: Pamela L. Levesque, CREATION DATE: 14-May-1982
41 0041 1
42 0042 1 MODIFIED BY:
43 0043 1
44 0044 1 1-001 Original. PLL 14-May-1982
45 0045 1 1-002 Allow an upper bound of 0. PLL 20-May-1982
46 0046 1 1-003 Make routine global. PLL 3-Jun-1982
47 0047 1 1-004 Check return status from LIB$FREE_VM and LIB$GET_VM. PLL 7-Jun-1982
48 0048 1 1-005 Row 0 should be included in ARSIZE. Also, the counter in the INCR
49 0049 1 loop for setting the bounds should start at 0, not 1. PLL 30-Jun-1982
50 0050 1 1-006 Initialize BNDSARG outside of the loop which sets the lower and
51 0051 1 upper bounds. PLL 27-Sep-1982
52 0052 1 1-007 Fix initialization of array elements to not write beyond the space
53 0053 1 allocated by LIB$GET_VM. (End_addr was incorrectly computed.)
54 0054 1 PLL 1-Oct-1982
55 0055 1 1-008 - Add RECORD datatype. Since there is no datatype by this name on the
56 0056 1 VAX, Basic uses DSC$K_DTYPE_2 (none). MDL 26-May-1983
57 0057 1 1-009 - correct array initialization. use a loop of CH$FILL's; the MOVCS

```

:	58	0058	1	:		
:	59	0059	1	:		
:	60	0060	1	:	1-010	-
:	61	0061	1	:		
:	62	0062	1	:		
:	63	0063	1	:	1-011	-
:	64	0064	1	:		
:	65	0065	1	:		
:	66	0066	1	:	--	
:	67	0067	1	:		

generated by one CH\$X comes nowhere near writing all of a large array,
as the length operands for the instruction are words. MDL 20-Dec-1983
the size to return to FREE_VM should be calculated from the array size
field in the array descriptor, in the same fashion as for the call to
GET_VM when the array is first allocated. MDL 9-Feb-1984
check to see if there's enough virtual address space available before
calling GET_VM. also, free dynamic strings if returning an array of
them. MDL 22-Feb-1984

Declarations

```

: 69      0068 1 %SBTTL 'Declarations'
: 70      0069 1
: 71      0070 1 SWITCHES:
: 72      0071 1
: 73      0072 1
: 74      0073 1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
: 75      0074 1
: 76      0075 1
: 77      0076 1 LINKAGES:
: 78      0077 1
: 79      0078 1 NONE
: 80      0079 1
: 81      0080 1 TABLE OF CONTENTS:
: 82      0081 1
: 83      0082 1
: 84      0083 1 FORWARD ROUTINE
: 85      0084 1 BASSRT_DIM : NOVALUE; ! run time dimension
: 86      0085 1
: 87      0086 1
: 88      0087 1 INCLUDE FILES:
: 89      0088 1
: 90      0089 1
: 91      0090 1 LIBRARY 'RTLSTARLE'; ! System symbols, typically from SYS$LIBRARY:STARLET.L32
: 92      0091 1
: 93      0092 1 REQUIRE 'RTLIN:RTLPSECT'; ! Define PSECT declarations macros
: 94      0187 1
: 95      0188 1
: 96      0189 1 MACROS:
: 97      0190 1
: 98      0191 1 NONE
: 99      0192 1
: 100     0193 1 EQUATED SYMBOLS:
: 101     0194 1
: 102     0195 1 NONE
: 103     0196 1
: 104     0197 1 FIELDS:
: 105     0198 1
: 106     0199 1 NONE
: 107     0200 1
: 108     0201 1 PSECTS:
: 109     0202 1
: 110     0203 1 DECLARE_PSECTS (BAS); ! Declare PSECTs for BASS$ facility
: 111     0204 1
: 112     0205 1 OWN STORAGE:
: 113     0206 1
: 114     0207 1 NONE
: 115     0208 1
: 116     0209 1 EXTERNAL REFERENCES:
: 117     0210 1
: 118     0211 1
: 119     0212 1 EXTERNAL ROUTINE
: 120     0213 1 BASS$STOP : NOVALUE, ! Signal a fatal error
: 121     0214 1 LIB$GET_EF, ! get an event flag
: 122     0215 1 LIB$FREE_EF, ! free an event flag
: 123     0216 1 LIB$STOP : NOVALUE, ! signal if error from GET_EF or $GETJPI
: 124     0217 1 LIB$FREE_VM, ! Deallocate virtual space
: 125     0218 1 LIB$GET_VM, ! Allocate virtual space
```

BASRT_DIM
1-011

Declarations

```
: 126      0219 1    STR$FREE1_DX;  
: 127      0220 1  
: 128      0221 1  EXTERNAL LITERAL  
: 129      0222 1    BAS$K_ARGDONMAT : UNSIGNED (8),  
: 130      0223 1    BAS$K_NOTIMP : UNSIGNED (8),  
: 131      0224 1    BAS$K_SUBOUTRAN : UNSIGNED (8),  
: 132      0225 1    BAS$K_TOOFEWARG : UNSIGNED (8),  
: 133      0226 1    BAS$K_TOOMANARG : UNSIGNED (8),  
: 134      0227 1    BAS$K_MAXMEMEXC : UNSIGNED (8),  
: 135      0228 1    BAS$K_PROLOSSOR : UNSIGNED (8);  
: 136      0229 1
```

6 5
16-Sep-1984 01:09:09
14-Sep-1984 11:56:38

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

Page 4
(2)

```
: Deallocate string space  
: Condition value symbols  
: Arguments don't match  
: Not implemented  
: Subscript out of range  
: Too few arguments  
: Too many arguments  
: Maximum memory exceeded  
: Program lost sorry
```

BASSRT_DIM - run time dimension

```
138 0230 1 %SBTTL 'BASSRT_DIM - run time dimension'
139 0231 1 GLOBAL ROUTINE-BASSRT_DIM (
140 0232 1     ARRAY,
141 0233 1     BOUND1
142 0234 1 ) : NOVALUE =
143 0235 1
144 0236 1 ++
145 0237 1 FUNCTIONAL DESCRIPTION:
146 0238 1
147 0239 1     This routine changes the size of an array. It does this by
148 0240 1     deleting any virtual memory previously allocated for the array
149 0241 1     elements, updating the upper bounds with the parameters passed,
150 0242 1     and then allocating enough virtual memory for the new size.
151 0243 1
152 0244 1     Numeric array elements are initialized to zero, and string array
153 0245 1     elements (descriptors) have their length and pointer set to zero.
154 0246 1
155 0247 1 CALLING SEQUENCE:
156 0248 1
157 0249 1     BASSRT_DIM (ARRAY.mx.da, BOUND1.rl.v [,BOUND2.rl.v, ... BOUND32.rl.v])
158 0250 1
159 0251 1 FORMAL PARAMETERS:
160 0252 1
161 0253 1     ARRAY          desc. of array to re-dimension
162 0254 1     BOUND1         new value of 1st upper bound
163 0255 1     [BOUND2,...   optional parameters - the number of bounds
164 0256 1     BOUND32]     passed must agree with the number of dimensions
165 0257 1                up to 32 dimensions are possible
166 0258 1
167 0259 1 IMPLICIT INPUTS:
168 0260 1
169 0261 1     NONE
170 0262 1
171 0263 1 IMPLICIT OUTPUTS:
172 0264 1
173 0265 1     The following fields in the array descriptor are updated:
174 0266 1         DSCSA_POINTER
175 0267 1         DSCSA_AO
176 0268 1         DCSL_ARSIZE
177 0269 1         bounds and multipliers
178 0270 1
179 0271 1 COMPLETION STATUS:
180 0272 1
181 0273 1     Signals if any error is encountered
182 0274 1
183 0275 1 SIDE EFFECTS:
184 0276 1
185 0277 1     NONE
186 0278 1
187 0279 1 --
188 0280 1
189 0281 2 BEGIN
190 0282 2
191 0283 2 BUILTIN
192 0284 2     ACTUALCOUNT,
193 0285 2     ACTUALPARAMETER,
194 0286 2     ASHP;
```

```
195 0287 2
196 0288
197 0289 MAP
198 0290 ARRAY : REF BLOCK [,BYTE];
199 0291
200 0292 LOCAL
201 0293 AR SIZE : INITIAL (1),
202 0294 BNDSARG : INITIAL (2),
203 0295 BOUNDS,
204 0296 MULTIPLIERS,
205 0297 LENGTH,
206 0298 STATUS,
207 0299 EVENT FLAG,
208 0300 NUM_FREE_PAGES: INITIAL(0),
209 0301 JPI_RETURN_LENGTH: INITIAL(0),
210 0302 JPI_ITEMS: VECTOR [4, LONG] INITIAL (
211 0303
212 0304
213 0305
214 0306
215 0307
216 0308
217 0309
218 0310
219 0311 IF (ACTUALCOUNT () - 1) NEQ .ARRAY [DSC$B_DIMCT]
220 0312 THEN
221 0313 BEGIN
222 0314 IF (ACTUALCOUNT () - 1) LSS .ARRAY [DSC$B_DIMCT]
223 0315 THEN
224 0316 BASS$STOP (BASSK_TOOFEWARG)
225 0317 ELSE
226 0318 BASS$STOP (BASSK_TOOMANARG);
227 0319 END;
228 0320
229 0321 IF (NOT (.ARRAY [DSC$V_FL_COEFF] AND .ARRAY [DSC$V_FL_BOUNDS]))
230 0322 THEN
231 0323 BASS$STOP (BASSK_ARGDONMAT);
232 0324
233 0325
234 0326
235 0327
236 0328
237 0329
238 0330
239 0331
240 0332
241 0333
242 0334
243 0335
244 0336
245 0337
246 0338
247 0339
248 0340
249 0341
250 0342
251 0343
```

temp used to compute array size
1st bounds is 2nd arg
upper or lower bounds of array
mult in array desc
temp used to compute element size
return status from calls
for use with \$GETJPI
written by \$GETJPI
bytes returned from \$GETJPI
((JPI\$ FREPTECNT*16) OR 4),
NUM_FREE_PAGES,
JPI_RETURN_LENGTH,
0);

First perform some checks to make sure the proper number of bounds have been passed, etc.

Free previously used space for elements, if any.

if this was an array of descriptors, there is some cleanup necessary prior to freeing the space for the array itself.
for dynamic string descriptors, each descriptor with space allocated must be freed.
a static string descriptor implies that the element is FIELDed, and the FIELD variable must be marked invalid.

IF .ARRAY [DSC\$B_DTYPE] EQL DSC\$K_DTYPE_DSC
THEN
BEGIN
LOCAL

```
252 0344 4      END ADDR;  
253 0345 4      END_ADDR = .ARRAY [DSC$A_POINTER] + .ARRAY [DSC$L_ARSIZE] -  
254 0346 4      .ARRAY [DSC$W_LENGTH];  
255 0347 4      !+  
256 0348 4      !- walk through the descriptors one at a time.  
257 0349 4      INCR DSC_ADDR FROM .ARRAY [DSC$A_POINTER] TO .END_ADDR BY .ARRAY [DSC$W_LENGTH] DO  
258 0350 4      BEGIN  
259 0351 3      MAP  
260 0352 3      DSC_ADDR : REF BLOCK [8, BYTE];  
261 0353 3  
262 0354 3      !+  
263 0355 3      !- dynamic? call STR$FREE1_DX.  
264 0356 3      IF .DSC_ADDR [DSC$B_CLASS] EQL DSC$K_CLASS_D  
265 0357 3      THEN  
266 0358 3      BEGIN  
267 0359 3      STATUS = STR$FREE1_DX (.DSC_ADDR);  
268 0360 6      END;  
269 0361 6  
270 0362 3      !+  
271 0363 3      !- static? mark length as 0 to prevent illicit usage as  
272 0364 3      a FIELDed variable.  
273 0365 3      IF .DSC_ADDR [DSC$B_CLASS] EQL DSC$K_CLASS_S  
274 0366 3      THEN  
275 0367 3      BEGIN  
276 0368 3      DSC_ADDR [DSC$W_LENGTH] = 0;  
277 0369 3      END;  
278 0370 6  
279 0371 6      END;      ! Incr dsc_addr  
280 0372 3  
281 0373 3      END;      ! If .array [dsc$b_dtype] eql dsc$k_dtype_dsc  
282 0374 4  
283 0375 4  
284 0376 3      !+  
285 0377 3      !- finally, free the old array itself up.  
286 0378 3      LENGTH = (IF .ARRAY [DSC$B_DTYPE] EQL DSC$K_DTYPE_P  
287 0379 3      THEN  
288 0380 4      (.ARRAY [DSC$L_ARSIZE]/ .ARRAY [DSC$W_LENGTH]) *  
289 0381 4      (.ARRAY [DSC$W_LENGTH]/ 2 + 1) ! compute length in bytes  
290 0382 4      ELSE  
291 0383 4      .ARRAY [DSC$L_ARSIZE]);  
292 0384 5      STATUS = LIB$FREE_VM (LENGTH, ARRAY [DSC$A_POINTER]);  
293 0385 4      IF NOT (.STATUS)  
294 0386 3      THEN  
295 0387 3      BASS$STOP (BASS$K_PROLOSSOR);  
296 0388 4  
297 0389 3      END;      ! If .array [dsc$a_pointer] neq 0  
298 0390 3  
299 0391 3      !+  
300 0392 3      !- Calculate the amount of space needed for the new size.  
301 0393 3  
302 0394 3      INCR COUNTER FROM 1 TO .ARRAY [DSC$B_DIMCT] DO  
303 0395 3      BEGIN  
304 0396 3      IF ACTUALPARAMETER (.COUNTER + 1) LSS 0  
305 0397 3  
306 0398 3  
307 0399 3  
308 0400 3
```

```
309 0401      THEN
310 0402          BASS$STOP (BASS$K_SUBOUTRAN);          ! don't allow bounds < 0
311 0403      AR SIZE = .AR_SIZE * (ACTUALPARAMETER (.COUNTER + 1) + 1);
312 0404      END;
313 0405
314 0406      AR SIZE = .AR_SIZE * .ARRAY [DSC$W_LENGTH];
315 0407      ARRAY [DSC$L_ARSIZE] = .AR_SIZE;
316 0408
317 0409      +
318 0410      Update the bounds and multipliers in the array descriptor.
319 0411      -
320 0412
321 0413      BOUNDS = ARRAY [DSC$L_M1] + (%UPVAL * .ARRAY [DSC$B_DIMCT]);
322 0414      MULTIPLIERS = ARRAY [DSC$L_M1];
323 0415
324 0416      INCR COUNTER FROM 0 TO ((.ARRAY [DSC$B_DIMCT] * 2) - 1) BY 2 DO
325 0417      BEGIN
326 0418          MAP
327 0419              BOUNDS : REF VECTOR;
328 0420
329 0421              BOUNDS [.COUNTER] = 0;          ! set lower bounds to 0
330 0422              BOUNDS [.COUNTER + 1] = ACTUALPARAMETER (.BNDSARG);
331 0423              ! take upper bound that was passed
332 0424              BNDSARG = .BNDSARG + 1;          ! point to next bounds arg, if any
333 0425
334 0426      END;
335 0427
336 0428      INCR COUNTER FROM 0 TO (.ARRAY [DSC$B_DIMCT] - 1) DO
337 0429      BEGIN
338 0430          MAP
339 0431              MULTIPLIERS : REF VECTOR;
340 0432              MULTIPLIERS [.COUNTER] = ACTUALPARAMETER (.COUNTER + 2) + 1;
341 0433      END;
342 0434
343 0435      +
344 0436      compute the total amount of space needed for the array, in bytes.
345 0437      -
346 0438      LENGTH = (IF .ARRAY [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
347 0439      THEN
348 0440          (.ARRAY [DSC$L_ARSIZE] / .ARRAY [DSC$W_LENGTH]) *
349 0441          (.ARRAY [DSC$W_LENGTH] / 2 + 1) ! compute length in bytes
350 0442      ELSE
351 0443          .ARRAY [DSC$L_ARSIZE]);
352 0444
353 0445      +
354 0446      see if there's enough room to expand this process' virtual address space
355 0447      that much. get the number of free pages available, multiply it by 512
356 0448      to get number of free bytes available, and see if this is more than the
357 0449      number of bytes needed (computed above).
358 0450      -
359 0451      STATUS = LIB$GET_EF (EVENT_FLAG);
360 0452      IF (NOT .STATUS) THEN LIB$STOP (.STATUS);
361 0453
362 0454      STATUS = $GETJPIW (EFN = .EVENT_FLAG, ITMLST = JPI_ITEMS);
363 0455      IF (NOT .STATUS) THEN LIB$STOP (.STATUS);
364 0456
365 0457      STATUS = LIB$FREE_EF (EVENT_FLAG);
```

```
366 0458 2 IF (NOT .STATUS) THEN LIB$STOP (.STATUS);
367 0459
368 0460
369 0461 IF (.NUM_FREE_PAGES * 512) LSSU .LENGTH
370 0462 THEN
371 0463     BASS$STOP (BASS$K_MAXMEMEXC);
372 0464
373 0465 + Allocate space for the new size and store it's address in the pointer field
374 0466 of the array descriptor.
375 0467 -
376 0468     STATUS = LIB$GET_VM (LENGTH, ARRAY [DSC$A_POINTER]);
377 0469     IF NOT (.STATUS)
378 0470     THEN
379 0471         BASS$STOP (BASS$K_MAXMEMEXC);
380 0472     ARRAY [DSC$A_A0] = .ARRAY [DSC$A_POINTER];
381 0473
382 0474 + Initialize the space according to the data type of the array elements.
383 0475 -
384 0476
385 0477     SELECTONE .ARRAY [DSC$B_DTYPE] OF
386 0478     SET
387 0479
388 0480     [DSC$K_DTYPE_B, DSC$K_DTYPE_W, DSC$K_DTYPE_L, DSC$K_DTYPE_F,
389 0481     DSC$K_DTYPE_D, DSC$K_DTYPE_G, DSC$K_DTYPE_H, DSC$K_DTYPE_Z] :
390 0482     BEGIN
391 0483     LOCAL
392 0484         CURR_SIZE:      INITIAL(.ARRAY [DSC$L_ARSIZE]),
393 0485         CURR_ADDR:      INITIAL(.ARRAY [DSC$A_POINTER]);
394 0486
395 0487
396 0488     + initialize the array in chunks of %Xffff bytes at a time. We do this
397 0489     because CH$FILL generates a MOVCS, and the length operand for MOVCS
398 0490     is a WORD. The total length of the array can be much longer than can
399 0491     be addressed by a WORD.
400 0492     -
401 0493     WHILE .CURR_ADDR LSS (.ARRAY [DSC$A_POINTER] + .ARRAY [DSC$L_ARSIZE]) DO
402 0494     BEGIN
403 0495         CH$FILL (%C'', MIN (.CURR_SIZE, %X'0000FFFF'), .CURR_ADDR);
404 0496         CURR_SIZE = .CURR_SIZE - %X'0000FFFF';
405 0497         CURR_ADDR = .CURR_ADDR + %X'0000FFFF';
406 0498     END;
407 0499
408 0500     END;
409 0501
410 0502 [DSC$K_DTYPE_P] :
411 0503     BEGIN
412 0504     LOCAL
413 0505         COUNT,
414 0506         END_ADDR;
415 0507     BIND
416 0508         PACKED_ZERO = UPLIT BYTE (REP 15 OF (%X'00'), %X'0C');
417 0509     COUNT = - .ARRAY [DSC$B_SCALE];
418 0510     END_ADDR = .ARRAY [DSC$A_POINTER] + .LENGTH -
419 0511     - (.ARRAY [DSC$W_LENGTH] / 2 + 1);
420 0512     INCR ELE_PTR FROM .ARRAY [DSC$A_POINTER] TO .END_ADDR
421 0513     BY (.ARRAY [DSC$W_LENGTH] / 2 + 1) DO
422 0514     BEGIN
```

```
423      ASHP (COUNT, %REF(15), PACKED ZERO, %REF(0),  
424           ARRAY [DSC$W_LENGTH], .ELE_PTR);  
425      END;  
426      END;  
427      ! ele by ele copy to  
428      ! accomodate scaling  
429  
430      [DSC$K_DTYPE_DSC] :  
431      BEGIN  
432      LOCAL  
433      END_ADDR;  
434      END_ADDR = .ARRAY [DSC$A_POINTER] + .ARRAY [DSC$L_ARSIZE] -  
435      .ARRAY [DSC$W_LENGTH];  
436      INCR DSC_ADDR FROM .ARRAY [DSC$A_POINTER] TO .END_ADDR BY .ARRAY [DSC$W_LENGTH] DO  
437      BEGIN  
438      MAP  
439      DSC_ADDR : REF BLOCK [8, BYTE];  
440      DSC_ADDR [DSC$B_CLASS] = DSC$K_CLASS_D;  
441      DSC_ADDR [DSC$B_DTYPE] = DSC$K_DTYPE_T;  
442      DSC_ADDR [DSC$W_LENGTH] = 0;  
443      DSC_ADDR [DSC$A_POINTER] = 0;  
444      END;  
445      END;  
446      [OTHERWISE] :  
447      BASS$STOP (BASS$K_NOTIMP);  
448      TES;  
449  
450      END;  
451
```

! End of routine BASSRT_DIM

```
.TITLE BASSRT_DIM  
.IDENT \1-011\  
.PSECT _BASS$CODE, NOWRT, SHR, PIC, 2  
00000000 04150004 00000 P.AAA: .LONG 68485124  
00000000 00000000 00004 .LONG 0  
00000000 00000000 00008 .LONG 0, 0  
00# 00010 P.AAB: .BYTE 0[15]  
0C 0001F .BYTE 12
```

```
PACKED_ZERO= P.AAB  
.EXTRN BASS$STOP, LIB$GET_EF  
.EXTRN LIB$FREE_EF, LIB$STOP  
.EXTRN LIB$FREE_VM, LIB$GET_VM  
.EXTRN STR$FREE_DX, BASS$K_ARGDONMAT  
.EXTRN BASS$K_NOTIMP, BASS$K_SUBOUTRAN  
.EXTRN BASS$K_TOOFEWARG  
.EXTRN BASS$K_TOOMANARG  
.EXTRN BASS$K_MAXMEMEXC  
.EXTRN BASS$K_PROLOSSOR  
.EXTRN SYSS$GETJPIW
```

```
OFFC 00000 .ENTRY BASSRT_DIM, Save R2,R3,R4,R5,R6,R7,R8,R9,- : 0231  
5B 00000000G 00 9E 00002 MOVAB BASS$STOP, R11 :  
:
```

			5E		1C	C2	00009	SUBL2	#28, SP			
			59		01	D0	0000C	MOVL	#1, AR SIZE	0281		
			56		02	D0	0000F	MOVL	#2, BND\$ARG			
					7E	D4	00012	CLRL	NUM_FREE_PAGES			
					AE	D4	00014	CLRL	JPI_RETURN_LENGTH			
	10	AE	C5	AF	10	28	00017	MOV3	#16, P.AAA, JPI_ITEMS	0304		
			14	AE	6E	9E	0001D	MOVAB	NUM_FREE_PAGES, JPI_ITEMS+4	0281		
			18	AE	AE	9E	00021	MOVAB	JPI_RETURN_LENGTH, JPI_ITEMS+8			
				50	6C	9A	00026	MOVZBL	(AP), R0	0311		
					50	D7	00029	DECL	R0			
50				57	04	AC	D0	0002B	MOVL	ARRAY, R7		
	0B	A7		08	00	ED	0002F	CMPZV	#0, #8, 11(R7), R0			
					1A	13	00035	BEQL	3\$			
				50	6C	9A	00037	MOVZBL	(AP), R0	0314		
					50	D7	0003A	DECL	R0			
50				08	00	ED	0003C	CMPZV	#0, #8, 11(R7), R0			
	0B	A7			06	15	00042	BLEQ	1\$			
				7E	00G	8F	9A	00044	MOVZBL	#BASSK_TOOFEWARG, -(SP)	0316	
					04	11	00048	BRB	2\$			
				7E	00G	8F	9A	0004A	MOVZBL	#BASSK_TOOMANARG, -(SP)	0318	
				6B	01	FB	0004E	CALLS	#1, BASS\$STOP			
	05			A7	06	E1	00051	BBC	#6, 10(R7), 4\$	0321		
					0A	A7	95	00056	TSTB	10(R7)		
					07	19	00059	BLSS	5\$			
				7E	00G	8F	9A	0005B	MOVZBL	#BASSK_ARGDONMAT, -(SP)	0323	
				6B	01	FB	0005F	CALLS	#1, BASS\$STOP			
				58	04	A7	9E	00062	MOVAB	4(R7), R8	0328	
					68	D5	00066	TSTL	(R8)			
					75	13	00068	BEQL	13\$			
				18	02	A7	91	0006A	CMPB	2(R7), #24	0340	
					33	12	0006E	BNEQ	10\$			
54				68	0C	A7	C1	00070	ADDL3	12(R7), (R8), R4	0345	
				53		67	3C	00075	MOVZWL	(R7), R3	0346	
55				54		53	C3	00078	SUBL3	R3, R4, END_ADDR		
				52		68	D0	0007C	MOVL	(R8), DSC_ADDR	0350	
					1D	11	0007F	BRB	9\$			
				02	03	A2	91	00081	CMPB	3(DSC_ADDR), #2	0358	
					0C	12	00085	BNEQ	7\$			
					52	DD	00087	PUSHL	DSC_ADDR	0361		
				00	01	FB	00089	CALLS	#1, STR\$FREE1_DX			
	00000000G			54	50	D0	00090	MOVL	R0, STATUS			
				01	03	A2	91	00093	CMPB	3(DSC_ADDR), #1	0368	
					02	12	00097	BNEQ	8\$			
					62	B4	00099	CLRW	(DSC_ADDR)	0371		
				52	53	C0	0009B	ADDL2	R3, DSC_ADDR	0350		
				55	52	D1	0009E	CMPL	DSC_ADDR, END_ADDR			
					DE	15	000A1	BLEQ	6\$			
				15	02	A7	91	000A3	CMPB	2(R7), #21	0381	
					15	12	000A7	BNEQ	11\$			
				51	67	3C	000A9	MOVZWL	(R7), R1	0383		
51				A7	51	C7	000AC	DIVL3	R1, 12(R7), R1			
				50	67	3C	000B1	MOVZWL	(R7), R0	0384		
				50	02	C6	000B4	DIVL2	#2, R0			
					50	D6	000B7	INCL	R0			
					51	C4	000B9	MULL2	R1, R0			
				50	04	11	000BC	BRB	12\$	0383		
					50	OC	A7	D0	000BE	MOVL	12(R7), R0	0386

OC	AE	50	D0	000C2	12\$:	MOVL	R0, LENGTH	:	0381		
		58	DD	000C6		PUSHL	R8	:	0387		
		10	AE	9F	000C8	PUSHAB	LENGTH	:			
00000000G	00	02	FB	000CB		CALLS	#2, LIB\$FREE_VM	:			
	54	50	D0	000D2		MOVL	R0, STATUS	:			
	07	54	E8	000D5		BLBS	STATUS, 13\$	:	0388		
	7E	00G	8F	9A	000D8	MOVZBL	#BAS\$K_PROLOSSOR, -(SP)	:	0390		
	6B	01	FB	000DC		CALLS	#1, BAS\$\$\$STOP	:			
	53	0B	A7	9A	000DF	13\$:	MOVZBL	11(R7), R3	:	0398	
			52	D4	000E3		CLRL	COUNTER	:		
			16	11	000E5		BRB	16\$	:		
		04	AC	42	D5	000E7	14\$:	TSTL	4(AP)[COUNTER]	:	0400
			07	18	000EB		BGEQ	15\$	:		
	7E	00G	8F	9A	000ED		MOVZBL	#BAS\$K_SUBOUTRAN, -(SP)	:	0402	
	6B	01	FB	000F1		CALLS	#1, BAS\$\$\$STOP	:			
50	04	AC	42	01	C1	000F4	15\$:	ADDL3	#1, 4(AP)[COUNTER], R0	:	0403
				50	C4	000FA		MULL2	R0, AR_SIZE	:	
E6				53	F3	000FD	16\$:	AOBLEQ	R3, COUNTER, 14\$	:	0398
				67	3C	00101		MOVZWL	(R7), R0	:	0406
				50	C4	00104		MULL2	R0, AR_SIZE	:	
		0C	A7	9E	00107		MOVAB	12(R7), R10	:	0407	
			59	D0	0010B		MOVL	AR_SIZE, (R10)	:		
		0B	A7	9A	0010E		MOVZBL	11(R7), R2	:	0413	
		14	A7	42	DE	00112		MOVAL	20(R7)[R2], BOUNDS	:	
		14	A7	9E	00117		MOVAB	20(R7), MULTIPLIERS	:	0414	
55			01	78	0011B		ASHL	#1, R2, R5	:	0416	
			55	D7	0011F		DECL	R5	:		
			02	CE	00121		MNEGL	#2, COUNTER	:	0421	
			0B	11	00124		BRB	18\$	:		
		63	41	D4	00126	17\$:	CLRL	(BOUNDS)[COUNTER]	:		
	04	A3	41	6C	46	D0	00129	MOVL	(AP)[BND\$ARG], 4(BOUNDS)[COUNTER]	:	0422
				56	D6	0012F		INCL	BND\$ARG	:	0424
FFEF	51			55	F1	00131	18\$:	ACBL	R5, #2, COUNTER, 17\$	:	0416
				01	CE	00137		MNEGL	#1, COUNTER	:	0432
				07	11	0013A		BRB	20\$	:	
6041	08	AC	41	01	C1	0013C	19\$:	ADDL3	#1, 8(AP)[COUNTER], (MULTIPLIERS)[COUNTER]	:	
F5				52	F2	00143	20\$:	AOBLSS	R2, COUNTER, 19\$	:	0428
				15	A7	91	00147	CMPB	2(R7), #21	:	0438
				14	12	0014B		BNEQ	21\$	:	
				67	3C	0014D		MOVZWL	(R7), R1	:	0440
51				51	C7	00150		DIVL3	R1, (R10), R1	:	
				67	3C	00154		MOVZWL	(R7), R0	:	0441
				50	C6	00157		DIVL2	#2, R0	:	
				50	D6	0015A		INCL	R0	:	
				51	C4	0015C		MULL2	R1, R0	:	
				03	11	0015F		BRB	22\$	:	0440
				6A	D0	00161	21\$:	MOVL	(R10), R0	:	0443
	OC	AE		50	D0	00164	22\$:	MOVL	R0, LENGTH	:	0438
				08	AE	9F	00168	PUSHAB	EVENT_FLAG	:	0451
00000000G	00			01	FB	0016B		CALLS	#1, LIB\$GET_EF	:	
	54			50	D0	00172		MOVL	R0, STATUS	:	
	09			54	E8	00175		BLBS	STATUS, 23\$	:	0452
				54	DD	00178		PUSHL	STATUS	:	
00000000G	00			01	FB	0017A		CALLS	#1, LIB\$STOP	:	
				7E	7C	00181	23\$:	CLRQ	-(SP)	:	0454
				7E	D4	00183		CLRL	-(SP)	:	
				1C	AE	9F	00185	PUSHAB	JPI_ITEMS	:	

			7E	7C	00188	CLRQ	-(SP)	
			AE	DD	0018A	PUSHL	EVENT FLAG	
			07	FB	0018D	CALLS	#7, SYS\$GETJPIW	
			50	DO	00194	MOVL	R0, STATUS	
			54	E8	00197	BLBS	STATUS, 24\$	0455
			54	DD	0019A	PUSHL	STATUS	
			01	FB	0019C	CALLS	#1, LIB\$STOP	
			AE	9F	001A3	PUSHAB	EVENT FLAG	0457
			01	FB	001A6	CALLS	#1, LIB\$FREE_EF	
			50	DO	001AD	MOVL	R0, STATUS	
			54	E8	001B0	BLBS	STATUS, 25\$	0458
			54	DD	001B3	PUSHL	STATUS	
			01	FB	001B5	CALLS	#1, LIB\$STOP	
50			09	78	001BC	ASHL	#9, NUM_FREE_PAGES, R0	0460
			50	D1	001C0	CMPL	R0, LENGTH	
			07	1E	001C4	BGEQU	26\$	
			8F	9A	001C6	MOVZBL	#BASSK_MAXMEMEXC, -(SP)	0462
			01	FB	001CA	CALLS	#1, BASS\$STOP	
			58	DD	001CD	PUSHL	R8	0468
			AE	9F	001CF	PUSHAB	LENGTH	
			02	FB	001D2	CALLS	#2, LIB\$GET_VM	
			50	DO	001D9	MOVL	R0, STATUS	
			54	E8	001DC	BLBS	STATUS, 27\$	0469
			8F	9A	001DF	MOVZBL	#BASSK_MAXMEMEXC, -(SP)	0471
			01	FB	001E3	CALLS	#1, BASS\$STOP	
			68	DO	001E6	MOVL	(R8), 16(R7)	0472
			A7	9A	001EA	MOVZBL	2(R7), R0	0478
			1E	13	001EE	BEQL	30\$	0481
			50	91	001F0	CMPB	R0, #6	
			05	1F	001F3	BLSSU	28\$	
			50	91	001F5	CMPB	R0, #8	
			14	1B	001F8	BLEQU	30\$	
			50	91	001FA	CMPB	R0, #10	
			05	1F	001FD	BLSSU	29\$	
			50	91	001FF	CMPB	R0, #11	
			0A	1B	00202	BLEQU	30\$	
			50	91	00204	CMPB	R0, #27	
			3C	1F	00207	BLSSU	34\$	
			50	91	00209	CMPB	R0, #28	
			37	1A	0020C	BGTRU	34\$	
			6A	DO	0020E	MOVL	(R10), CURR_SIZE	0485
			68	DO	00211	MOVL	(R8), CURR_ADDR	0486
			6A	C1	00214	ADDL3	(R10), (R8), R4	0494
			59	D1	00218	CMPL	CURR_ADDR, R4	
			01	19	0021B	BLSS	32\$	
				04	0021D	RET		
			56	DO	0021E	MOVL	CURR_SIZE, R0	0496
			50	D1	00221	CMPL	R0, #65535	
			05	15	00228	BLEQ	33\$	
			8F	3C	0022A	MOVZWL	#65535, R0	
			00	2C	0022F	MOVCS	#0, (SP), #0, R0, (CURR_ADDR)	
			69		00234			
			E6	9E	00235	MOVAB	-65535(R6), CURR_SIZE	0497
			E9	9E	0023C	MOVAB	65535(R9), CURR_ADDR	0498
			CF	11	00243	BRB	31\$	0494
			50	91	00245	CMPB	R0, #21	0502
			32	12	00248	BNEQ	37\$	

00000000G 00 20
54
09
00000000G 00
00000000G 00 08
54
09
00000000G 00
50 6E
OC AE
7E
6B
00G
00000000G 00 10
54
07
7E
6B
00G
10 A7
50 02
06
08
0A
0B
1B
1C
56
59
54
0000FFFF 50
8F
50 FFFF
6E
56 FFFF0001
59 0000FFFF
15

BASSRT_DIM
1-011

BASSRT_DIM - run time dimension

D 6
16-Sep-1984 01:09:09
14-Sep-1984 11:56:38

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

Page 14
(3)

			59	08	A7	98	0024A		CVTBL	8(R7), COUNT	: 0509
			59		59	CE	0024E		MNEGL	COUNT, COUNT	: 0510
	54		68	0C	AE	C1	00251		ADDL3	LENGTH, (R8), R4	: 0511
			50		67	3C	00256		MOVZWL	(R7), R0	: 0510
			50		02	C6	00259		DIVL2	#2, R0	: 0511
			54		50	C2	0025C		SUBL2	R0, R4	: 0510
					54	D7	0025F		DECL	END_ADDR	: 0513
			56	01	A0	9E	00261		MOVAB	1(R0), R6	: 0513
			55		68	D0	00265		MOVL	(R8), ELE_PTR	: 0516
					0C	11	00268		BRB	36\$	: 0512
00	FD80	CF	0F		59	F8	0026A	35\$:	ASHP	COUNT, #15, PACKED_ZERO, #0, (R7), -	: 0516
			65		67		00271			(ELE_PTR)	: 0512
			55		56	C0	00273		ADDL2	R6, ELE_PTR	: 0478
			54		55	D1	00276	36\$:	CMPL	ELE_PTR, END_ADDR	: 0520
					EF	15	00279		BLEQ	35\$	: 0524
						04	0027B		RET		: 0525
			18		50	91	0027C	37\$:	CMPB	R0, #24	: 0526
					23	12	0027F		BNEQ	40\$	: 0532
	54		68		6A	C1	00281		ADDL3	(R10), (R8), R4	: 0533
			51		67	3C	00285		MOVZWL	(R7), R1	: 0526
	52		54		51	C3	00288		SUBL3	R1, R4, END_ADDR	: 0538
			50		68	D0	0028C		MOVL	(R8), DSC_ADDR	: 0543
					0D	11	0028F		BRB	39\$	: 0532
			60	020E0000	8F	D0	00291	38\$:	MOVL	#34471936, (DSC_ADDR)	: 0526
				04	A0	D4	00298		CLRL	4(DSC_ADDR)	: 0532
			50		51	C0	0029B		ADDL2	R1, DSC_ADDR	: 0538
			52		50	D1	0029E	39\$:	CMPL	DSC_ADDR, END_ADDR	: 0543
					EE	15	002A1		BLEQ	38\$	: 0478
						04	002A3		RET		: 0538
			7E	00G	8F	9A	002A4	40\$:	MOVZBL	#BASSK_NOTIMP, -(SP)	: 0543
			68		01	FB	002AB		CALLS	#1, BASS\$STOP	: 0543
					04	002AB		RET			: 0543

; Routine Size: 684 bytes, Routine Base: _BASSCODE + 0020

; 452 0544 1 !<BLF/PAGE>

BASSRT_DIM
1-011

BASSRT_DIM - run time dimension

E 6
16-Sep-1984 01:09:09
14-Sep-1984 11:56:38

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

Page 15
(4)

: 454 0545 1 END
: 455 0546 1
: 456 0547 0 ELUDOM

! End of module BASSRT_DIM

PSECT SUMMARY

: Name Bytes Attributes
: _BASSCODE 716 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 28 0 581 00:01.1

COMMAND QUALIFIERS

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

: Size: 684 code + 32 data bytes
: Run Time: 00:14.4
: Elapsed Time: 00:30.9
: Lines/CPU Min: 2283
: Lexemes/CPU-Min: 17657
: Memory Used: 211 pages
: Compilation Complete

0031

AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY