

BBBBBBBBBBBB		AAAAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAAAA		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	SSSSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSSSS		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
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL


```
BBBBBBBBB      AAAAAA      SSSSSSSS      MM      MM      AAAAAA      TTTTTTTTTT      RRRRRRRR      EEEEEEEEEE      DDDDDDDD
BBBBBBBBB      AAAAAA      SSSSSSSS      MM      MM      AAAAAA      TTTTTTTTTT      RRRRRRRR      EEEEEEEEEE      DDDDDDDD
BB      BB      AA      AA      SS      SS      SSSSSSSS      MM      MM      AA      AA      TT      TT      RR      RR      EE      EE      DD      DD
BB      BB      AA      AA      SS      SS      SSSSSSSS      MM      MM      AA      AA      TT      TT      RR      RR      EE      EE      DD      DD
BB      BB      AA      AA      SS      SS      SSSSSSSS      MM      MM      AA      AA      TT      TT      RR      RR      EE      EE      DD      DD
BB      BB      AA      AA      SS      SS      SSSSSSSS      MM      MM      AA      AA      TT      TT      RR      RR      EE      EE      DD      DD
BBBBBBBBB      AA      AA      SSSSSSSS      MM      MM      AAAAAA      TT      TT      RRRRRRRR      EEEEEEEE      DD      DD
BBBBBBBBB      AA      AA      SSSSSSSS      MM      MM      AAAAAA      TT      TT      RRRRRRRR      EEEEEEEE      DD      DD
BB      BB      AAAAAAAAAA      SS      MM      MM      AAAAAAAAAA      TT      TT      RR      RR      EE      EE      DD      DD
BB      BB      AAAAAAAAAA      SS      MM      MM      AAAAAAAAAA      TT      TT      RR      RR      EE      EE      DD      DD
BB      BB      AA      AA      SS      SS      MM      MM      AA      AA      TT      TT      RR      RR      EE      EE      DD      DD
BB      BB      AA      AA      SS      SS      MM      MM      AA      AA      TT      TT      RR      RR      EE      EE      DD      DD
BBBBBBBBB      AA      AA      SSSSSSSS      MM      MM      AA      AA      TT      TT      RR      RR      EEEEEEEEEE      DDDDDDDD
BBBBBBBBB      AA      AA      SSSSSSSS      MM      MM      AA      AA      TT      TT      RR      RR      EEEEEEEEEE      DDDDDDDD
                                     ....
                                     ....
                                     ....
                                     ....

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
```


BASSMAT_REDIM
Table of contents

; Redimension a BASIC-PLUS-2 array ^{H 16}

15-SEP-1984 23:50:44 VAX/VMS Macro V04-00

Page 0

(2) 48
(3) 99

DECLARATIONS
BASSMAT_REDIM - Redimension a BASIC array


```
0000 1      .TITLE BASSMAT_REDIM
0000 2      .IDENT /1-002/
0000 3
0000 4
0000 5 :*****
0000 6 :*
0000 7 :*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 :*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 :*  ALL RIGHTS RESERVED.
0000 10 :*
0000 11 :*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 :*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 :*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 :*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 :*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 :*  TRANSFERRED.
0000 17 :*
0000 18 :*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 :*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 :*  CORPORATION.
0000 21 :*
0000 22 :*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 :*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 :*
0000 25 :*
0000 26 :*****
0000 27 :
0000 28
0000 29 :++
0000 30 : FACILITY: BASIC code support
0000 31 :
0000 32 : ABSTRACT:
0000 33 :
0000 34 :     This module is the routine to redimension BASIC arrays. It is
0000 35 :     called from both the RTL and code generated by the compiler.
0000 36 :
0000 37 : ENVIRONMENT: User Mode, AST Reentrant
0000 38 :
0000 39 : --
0000 40 : AUTHOR: R. WILL, CREATION DATE: 17-Apr-79
0000 41 :
0000 42 : MODIFIED BY:
0000 43 :
0000 44 :     : VERSION 00
0000 45 : 0-001 - Original
0000 46 : 1-002 - Changed shared external references to G^ RNH 25-Sep-81
```



```

0000 48      .SBTTL  DECLARATIONS
0000 49      :
0000 50      : INCLUDE FILES:
0000 51      :
0000 52
0000 53      $DSCDEF                      ; Define descriptor offsets
0000 54
0000 55      :
0000 56      : EXTERNAL DECLARATIONS:
0000 57      :
0000 58      .DSABL  GBL                      ; Prevent undeclared
0000 59                      ; symbols from being
0000 60                      ; automatically global.
0000 61
0000 62      .EXTRN  BAS$$STOP                ; Routine to signal errors
0000 63
0000 64      :
0000 65      : The following are error messages that may be signalled
0000 66      :
0000 67      .EXTRN  BAS$K_CANCHAARR          ; Cannot Change Array Dimensions
0000 68                      ; DSCSV_FL_REDIM is not set or
0000 69                      ; DSCSV_FL_BOUNDS is not set
0000 70      .EXTRN  BAS$K_MATDIMERR        ; Matrix Dimension Error
0000 71                      ; Number of input parameters differs
0000 72                      ; from number of dimensions in array
0000 73      .EXTRN  BAS$K_REDARR            ; Redimensioned Array
0000 74                      ; Not enough space allocated for input
0000 75                      ; dimension parameters
0000 76      .EXTRN  BAS$K_SUBOUTRAN         ; Subscript Out Of Range
0000 77                      ; One of the new bounds specified was
0000 78                      ; negative or zero
0000 79
0000 80      :
0000 81      : MACROS:
0000 82      :
0000 83
0000 84      :
0000 85      : EQUATED SYMBOLS:
0000 86      :
0000 87
0000 88      :
0000 89      : OWN STORAGE:
0000 90      :
0000 91
0000 92      :
0000 93      : PSECT DECLARATIONS:
0000 94      :
0000 95      .PSECT _BAS$CODE PIC, USR, CON, REL, LCL, SHR, -
0000 96                      EXE, RD, NOWRT, LONG
0000 97

```



```
0000 99 .SBTTL BASSMAT_REDIM - Redimension a BASIC array
0000 100 :++
0000 101 : FUNCTIONAL DESCRIPTION:
0000 102 :
0000 103 : This routine redimensions arrays for BASIC-PLUS-2. It first checks to
0000 104 : see if the array is currently the size that the call is requesting
0000 105 : redimensioning to. If so, the routine will return (so that if
0000 106 : DSC$V_FL_REDIM is not set an error will not be signalled). Otherwise
0000 107 : it will signal an error if DSC$V_FL_REDIM is not set, or if
0000 108 : 2 dimensions are input and the matrix currently only has 1 dimension
0000 109 : or only 1 dimension is input and the matrix currently has 2 dimensions
0000 110 : or if the input dimensions require more space than originally
0000 111 : allocated to the array (DSC$L_ARSIZE). If there are no errors, then
0000 112 : the routine will write either 1 or 2 upper bounds and the same
0000 113 : number of multipliers, and set to 0 the same number of lower bounds.
0000 114 : DSC$A_A0 will be set to DSC$A_POINTER. Also
0000 115 : note that total array size will always remain the initial allocated
0000 116 : length. Integer overflow is disabled so that a BASIC-PLUS-2 error
0000 117 : can be signalled if the space needed is too large instead of getting
0000 118 : a hardware error.
0000 119 :
0000 120 : CALLING SEQUENCE:
0000 121 :
0000 122 : CALL BASSMAT_REDIM (matrix.wx.da, rows.rl.v [, cols.rl.v])
0000 123 :
0000 124 : INPUT PARAMETERS:
0000 125 :
00000008 0000 126 : row_upr_bnd = 8
0000000C 0000 127 : col_upr_bnd = 12
0000 128 :
0000 129 : IMPLICIT INPUTS:
0000 130 :
0000 131 : NONE
0000 132 :
0000 133 : OUTPUT PARAMETERS:
00000004 0000 134 :
0000 135 : matrix = 4
0000 136 :
0000 137 : IMPLICIT OUTPUTS:
0000 138 :
0000 139 : NONE
0000 140 :
0000 141 : FUNCTION VALUE:
0000 142 : COMPLETION CODES:
0000 143 :
0000 144 : NONE
0000 145 :
0000 146 : SIDE EFFECTS:
0000 147 :
0000 148 : Errors list under externals may be signalled. The matrix parameter
0000 149 : may have different dimensions after routine execution.
0000 150 :
0000 151 :--
001C 0000 152 :
0002 0000 153 : .ENTRY BASSMAT_REDIM , ^M<R2,R3,R4>
0002 0002 154 :
0002 0002 155 : ; Routine to redimension array
; according to BASIC-PLUS-2
```


BASSMAT_REDIM
1-002

L 16

: Redimension a BASIC-PLUS-2 array 15-SEP-1984 23:50:44 VAX/VMS Macro V04-00 Page 4
BASSMAT_REDIM - Redimension a BASIC arr 6-SEP-1984 10:30:46 [BASRTL.SRC]BASMATRED.MAR;1 (3)

```
0002 156 ; syntax
0002 157
0002 158 ;+
0002 159 : Register usage
0002 160 R0, R1 computation temps
0002 161 R2 pointer to output matrix descriptor
0002 162 R3 upper bound for subscript 1 (rows)
0002 163 R4 upper bound for subscript 2 (columns)
0002 164 :-
0002 165
52 04 AC D0 0002 166 MOVL matrix(AP), R2 ; pointer to array descriptor
0006 167
0006 168 ;+
0006 169 : First check to see if the bounds are set in the array descriptor
0006 170 : if they are not, we cannot change them. \is this really an error\
0006 171 :-
0006 172
0B 0A A2 07 E0 0006 173 BBS #DSC$V_FL_BOUNDS, DSC$B_AFLAGS(R2), 1$ ; If bounds are
000B 174 ; present, go check them
7E 00'8F 9A 000B 175 MOVZBL #BASSK CANCHAARR, -(SP) ; Bounds are not present
00000000'GF 01 FB 000F 176 CALLS #1, G^BASS$STOP ; signal redimension error
0016 177
0016 178 ;+
0016 179 : Bounds are present. Take execution path depending on number
0016 180 : of dimensions.
0016 181 :-
0016 182
02 6C 91 0016 183 1$: CMPB (AP), #2 ; find # of bounds input
69 1F 0019 184 BLSSU RETURN ; no bounds input, so exit
001B 185 ; \should that be an error\
001B 186 ; \should I even check for it\
74 1A 001B 187 BGTRU TWO_DIMS ; go do more than 1 bound
001D 188
```



```
001D 190 :+
001D 191 : One dimension was input.
001D 192 :-
001D 193 :+
001D 194 : Put new bound into R3. If it is negative or zero signal an error
001D 195 :-
001D 196 :-
001D 197 :-
53 08 AC D0 001D 198      MOVL    row_upr_bnd(AP), R3      ; get the new bound
      56 15 0021 199      BLEQ    ERR4                    ; error
0023 200
0023 201 :+
0023 202 : Check to see if array is one dimensional. If not signal an error.
0023 203 :-
0023 204 :-
01 0B A2 91 0023 205      CMPB    DSC$B_DIMCT(R2), #1      ; One dimensional array?
      2F 12 0027 206      BNEQU    ERR1                    ; No, go signal error
0029 207
0029 208 :+
0029 209 : Check to see if the new bound is the same as the old bound.
0029 210 : If so return.
0029 211 :-
0029 212 :-
08 AC 1C A2 D1 0029 213      CMPL    DSC$L_M1+8(R2), row_upr_bnd(AP) ; Yes, compare bounds
      54 13 002E 214      BEQL     RETURN                  ; Array is already desired
0030 215      ; size, so return
0030 216
0030 217 :+
0030 218 : See if array is redimensionable. If not, signal an error. Note that
0030 219 : we must check for correct size before redimensionability so that we
0030 220 : won't give the 'can't redimension' error when array is correct size.
0030 221 :-
0030 222 :-
39 0A A2 04 E1 0030 223      BBC     #DSC$V_FL_REDIM, DSC$B_AFLAGS(R2), ERR3 ; if can't redimension
0035 224      ; array, go signal an error
0035 225
0035 226 :+
0035 227 : Compute array size needed for new bounds. If more space is needed
0035 228 : than is currently allocated to the array signal an error.
0035 229 :-
0035 230 :-
51 01 50 62 3C 0035 231      MOVZWL  DSC$W_LENGTH(R2), R0      ; make item length a longword
      08 AC C1 0038 232      ADDL3    row_upr_bnd(AP), #1, R1    ; Add 1 to upper bound since
003D 233      ; BASIC-PLUS-2 ARRAYS have 0
003D 234      ; for a lower bound
      51 50 C4 003D 235      MULL2    R0, R1                    ; find space need w/ new bound
      21 1D 0040 236      BVS       ERR2                    ; not enuf space, go signal
0C A2 51 D1 0042 237      CMPL     R1, DSC$L_ARSIZE(R2)        ; see if too much space needed
      1B 14 0046 238      BGTR      ERR2                    ; too much space, go signal
0048 239 : \what is supposed to to unsigned and what is not? \
0048 240
0048 241 :+
0048 242 : All errors have been caught. Now redimension array.
0048 243 :-
0048 244 :-
1C A2 08 AC D0 0048 245      MOVL     row_upr_bnd(AP), DSC$L_M1+8(R2) ; write new upper bound
      18 A2 D4 004D 246      CLRL     DSC$L_MT+4(R2)          ; set lower bound to 0 since
```


BASSMAT_REDIM
1-002

B 1
; Redimension a BASIC-PLUS-2 array
BASSMAT_REDIM - Redimension a BASIC arr

15-SEP-1984 23:50:44
6-SEP-1984 10:30:46

VAX/VMS Macro V04-00
[BASRTL.SRC]BASMATRED.MAR;1

Page 6
(3)

14 A2 01 08 AC C1 0050 247
0050 248
0056 249
0056 250
2C 11 0056 251
0058 252
0058 253

ADDL3 row_upr_bnd(AP), #1, DSC\$\$_M1(R2)

BRB RETURN

; B+2 arrays have 0 lower bnd
; compute and write new
; multiplier, assuming that
; all B+2 arrays have 0 for L1
; and exit

BASSMAT_REDIM
1-002

C 1
; Redimension a BASIC-PLUS-2 array 15-SEP-1984 23:50:44 VAX/VMS Macro V04-00 Page 7
BASSMAT_REDIM - Redimension a BASIC arr 6-SEP-1984 10:30:46 [BASRTL.SRC]BASMATRED.MAR;1 (3)

```

0058 255 ;+
0058 256 ; Signal errors
0058 257 ; -
0058 258
0058 259 ERR1: MOVZBL #BASSK_MATDIMERR, -(SP) ; Matrix Dimension Error
005C 260 CALLS #1, G^BASS$STOP ; Signal the error
0063 261
0063 262 ERR2: MOVZBL #BASSK_REDARR, -(SP) ; Redimensioned Array
0067 263 CALLS #1, G^BASS$STOP ; Signal the error
006E 264
006E 265 ERR3: MOVZBL #BASSK_CANCHAARR, -(SP) ; Can't Change Array Dimension
0072 266 CALLS #1, G^BASS$STOP
0079 267
0079 268 ERR4: MOVZBL #BASSK_SUBOUTRAN, -(SP) ; Subscript out of range
007D 269 CALLS #1, G^BASS$STOP
0084 270
04 0084 271 RETURN: RET ; and exit
0085 272
```



```
0085 274 ;+
0085 275 ; Two dimensions were input.
0085 276 ; -
0085 277 ;+
0085 278 ; Put the 2 new upper bounds into registers. If either is negative or 0,
0085 279 ; signal an error.
0085 280 ; -
0085 281 ; -
0085 282 ; -
53 08 AC D0 0085 283      MOVL    row_upr_bnd(AP), R3      ; new upper bound
    EE 15 0089 284      BLEQ    ERR4                  ; error
54 0C AC D0 008B 285      MOVL    col_upr_bnd(AP), R4    ; new upper bound
    EB 15 008F 286      BLEQ    ERR4                  ; error
0091 287 ;+
0091 288 ; Check to see if array is two dimensional. If not signal an error.
0091 289 ; -
0091 290 ; -
0091 291 ; -
0091 292 TWO_DIMS:
    0B A2 91 0091 293      CMPB    DSC$B_DIMCT(R2), #2      ; Two dimensional array?
    C1 12 0095 294      BNEQU    ERR1                    ; No, go signal error
0097 295 ;+
0097 296 ; Check to see if the new bounds are the same as the old bounds.
0097 297 ; If so return.
0097 298 ; -
0097 299 ; -
0097 300 ; -
08 AC 20 A2 D1 0097 301      CMPL    DSC$L_M2+8(R2), row_upr_bnd(AP) ; Yes, compare number of rows
    07 12 009C 302      BNEQ     25$                    ; Not =, continue redimension
0C AC 28 A2 D1 009E 303      CMPL    DSC$L_M2+16(R2), col_upr_bnd(AP) ; compare number of columns
    DF 13 00A3 304      BEQL     RETURN                  ; Array is already desired
00A5 305 ; size, so return
00A5 306 ;+
00A5 307 ; See if array is redimensionable. If not, signal an error. Note that
00A5 308 ; we must check for correct size before redimensionability so that we
00A5 309 ; won't give the 'can't redimension' error when array is correct size.
00A5 310 ; -
00A5 311 ; -
00A5 312 ; -
C4 0A A2 04 E1 00A5 313 25$: BBC      #DSC$V_FL_REDIM, DSC$B_AFLAGS(R2), ERR3 ; if can't redimension
00AA 314 ; array, go signal an error
00AA 315 ;+
00AA 316 ; Compute array size needed for new bounds. If more space is needed
00AA 317 ; than is currently allocated to the array signal an error.
00AA 318 ; -
00AA 319 ; -
00AA 320 ; -
51 01 50 62 3C 00AA 321      MOVZWL  DSC$W_LENGTH(R2), R0      ; make item length a longword
    08 AC C1 00AD 322      ADDL3    row_upr_bnd(AP), #1, R1    ; Add 1 to upper bound since
00B2 323 ; BASIC-PLUS-2 arrays have 0
00B2 324 ; for a lower bound for rows
00B2 325      BVS      ERR2      ; not enuf space, go signal
    51 AF 1D 00B4 326      MULL2    R0, R1                  ; find space need for a column
    50 AA 1D 00B7 327      BVS      ERR2      ; not enuf space, go signal
50 01 0C AC C1 00B9 328      ADDL3    col_upr_bnd(AP), #1, R0    ; Add 1 to upper bound since
00BE 329 ; BASIC-PLUS-2 arrays have 0
00BE 330 ; for a lower bound for column
```



```

      51      A3      1D      00BE      331      BVS      ERR2      ; not enuf space, go signal
      50      C4      00C0      332      MULL2      R0, R1      ; find total space needed
      9E      1D      00C3      333      BVS      ERR2      ; not enuf space, go signal
OC A2      51      D1      00C5      334      CMPL      R1, DSC$$_ARSIZE(R2) ; see if too much space needed
      98      14      00C9      335      BGTR      ERR2      ; too much space, go signal
      00CB      336      ;\what is supposed to to unsigned and what is not?\
      00CB      337
      00CB      338      ;+
      00CB      339      ; All errors have been caught. Now redimension array.
      00CB      340      ; -
      00CB      341
      1C A2      D4      00CB      342      CLRL      DSC$$_M2+4(R2) ; set lower bound to 0
      08 AC      D0      00CE      343      MOVL      row_upr_bnd(AP), DSC$$_M2+8(R2) ; write new upper bnd for rows
      08 AC      C1      00D3      344      ADDL3      row_upr_bnd(AP), #1, DSC$$_M1(R2) ; compute and write new row
      00D9      345      ; multiplier, assuming that
      00D9      346      ; all B+2 arrays have 0 for L1
      24 A2      D4      00D9      347      CLRL      DSC$$_M2+12(R2) ; set lower bound to 0
      0C AC      D0      00DC      348      MOVL      col_upr_bnd(AP), DSC$$_M2+16(R2) ; write new upper bnd for col
      0C AC      C1      00E1      349      ADDL3      col_upr_bnd(AP), #1, DSC$$_M2(R2) ; compute and write new col
      00E7      350      ; multiplier, assuming that
      00E7      351      ; all B+2 arrays have 0 for L2
      00E7      352
      9B      11      00E7      353      BRB      RETURN ; and exit
      00E9      354
      00E9      355      .END ; End of BASSMAT_REDIM
```


BASSMAT REDIM
Symbol Table

; Redimension a BASIC-PLUS-2 array F 1

15-SEP-1984 23:50:44
6-SEP-1984 10:30:46

VAX/VMS Macro V04-00
[BASRTL.SRC]BASMATRED.MAR;1

Page 10
(3)

```
BASS$STOP          ***** X 00
BASS$CANCHAARR      ***** X 00
BASS$MATDIMERR      ***** X 00
BASS$REDARR         ***** X 00
BASS$SUBOUTRAN      ***** X 00
BASSMAT REDIM       00000000 RG 02
COL UPR_BND         = 0000000C
DSC$B_AFLAGS        = 0000000A
DSC$B_DIMCT         = 0000000B
DSC$B_ARSIZE        = 0000000C
DSC$B_M1            = 00000014
DSC$B_M2            = 00000018
DSC$V_FL_BOUNDS     = 00000007
DSC$V_FL_REDIM      = 00000004
DSC$W_LENGTH        = 00000000
ERR1                00000058 R 02
ERR2                00000063 R 02
ERR3                0000006E R 02
ERR4                00000079 R 02
MATRIX              = 00000004
RETURN              00000084 R 02
ROW UPR_BND         = 00000008
TWO_DIMS            00000091 R 02
```

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes															
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE					
\$ABSS	00000000 (0.)	01 (1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE					
_BASSCODE	000000E9 (233.)	02 (2.)	PIC	USR	CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC	LONG					

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.06	00:00:00.24
Command processing	107	00:00:00.49	00:00:02.20
Pass 1	137	00:00:02.05	00:00:06.62
Symbol table sort	0	00:00:00.17	00:00:00.18
Pass 2	73	00:00:00.84	00:00:02.55
Symbol table output	3	00:00:00.04	00:00:00.03
Psect synopsis output	3	00:00:00.03	00:00:00.22
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	357	00:00:03.68	00:00:12.05

The working set limit was 1200 pages.
10708 bytes (21 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 146 non-local and 2 local symbols.
355 source lines were read in Pass 1, producing 13 object records in Pass 2.
8 pages of virtual memory were used to define 7 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name

Macros defined

_ \$255\$DUA28:[SYSLIB]STARLET.MLB;2

4

190 GETS were required to define 4 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS\$:BASMATRED/OBJ=OBJ\$:BASMATRED MSRC\$:BASMATRED/UPDATE=(ENH\$:BASMATRED)

0026 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

0027 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

BASMATSUB
LIS

BASMATSCA
LIS

BASMATRN
LIS