M 1

```
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSSS  RRRRRRRRRRR      TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSSS  RRRRRRRRRRR      TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSSS  RRRRRRRRRRR      TTTTTTTTTTTTTTT  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     SSSSSSSSS     RRRRRRRRRRR            TTT        LLL
BBBBBBBBBBBB     AAA         AAA     SSSSSSSSS     RRRRRRRRRRR            TTT        LLL
BBBBBBBBBBBB     AAA         AAA     SSSSSSSSS     RRRRRRRRRRR            TTT        LLL
BBB         BBB  AAAAAAAAAAAAAAA              SSS  RRR   RRR              TTT        LLL
BBB         BBB  AAAAAAAAAAAAAAA              SSS  RRR   RRR              TTT        LLL
BBB         BBB  AAAAAAAAAAAAAAA              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  SSSSSSSSSSSS     RRR         RRR        TTT        LLLLLLLLLLLLLLL
BBBBBBBBBBBB     AAA         AAA  SSSSSSSSSSSS     RRR         RRR        TTT        LLLLLLLLLLLLLLL
BBBBBBBBBBBB     AAA         AAA  SSSSSSSSSSSS     RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

FILEID**BASMATRED

```
BBBBBBBB      AAAAAA      SSSSSSSS  MM          MM    AAAAAA    TTTTTTTTTT  RRRRRRRR     EEEEEEEEEE  DDDDDDDD
BBBBBBBB      AAAAAA      SSSSSSSS  MM          MM    AAAAAA    TTTTTTTTTT  RRRRRRRR     EEEEEEEEEE  DDDDDDDD
BB      BB  AA      AA  SS          MMMM    MMMM  AA      AA      TT      RR      RR   EE          DD      DD
BB      BB  AA      AA  SS          MMMM    MMMM  AA      AA      TT      RR      RR   EE          DD      DD
BB      BB  AA      AA  SS          MM  MM  MM    AA      AA      TT      RR      RR   EE          DD      DD
BB      BB  AA      AA  SS          MM  MM  MM    AA      AA      TT      RR      RR   EE          DD      DD
BBBBBBBB    AA      AA    SSSSSS    MM      MM    AA      AA      TT      RRRRRRRR     EEEEEEEE    DD      DD
BBBBBBBB    AA      AA    SSSSSS    MM      MM    AA      AA      TT      RRRRRRRR     EEEEEEEE    DD      DD
BB      BB  AAAAAAAAAA          SS  MM      MM    AAAAAAAAAA      TT      RR  RR       EE          DD      DD
BB      BB  AAAAAAAAAA          SS  MM      MM    AAAAAAAAAA      TT      RR  RR       EE          DD      DD
BB      BB  AA      AA          SS  MM      MM    AA      AA      TT      RR    RR     EE          DD      DD      ....
BB      BB  AA      AA          SS  MM      MM    AA      AA      TT      RR    RR     EE          DD      DD      ....
BBBBBBBB    AA      AA  SSSSSSSS    MM      MM    AA      AA      TT      RR      RR   EEEEEEEEEE  DDDDDDDD        ....
BBBBBBBB    AA      AA  SSSSSSSS    MM      MM    AA      AA      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
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000            .TITLE  BAS$MAT_REDIM           ; Redimension a BASIC-PLUS-2 array
0000            .IDENT  /1-002/         ; File: BASMATRED.MAR Edit:RNH1002
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000
0000    ;++
0000    ; FACILITY: BASIC code support
0000    ;
0000    ; ABSTRACT:
0000    ;
0000    ;       This module is the routine to redimension BASIC arrays.  It is
0000    ;       called from both the RTL and code generated by the compiler.
0000    ;
0000    ; ENVIRONMENT: User Mode, AST Reentrant
0000    ;
0000    ;--
0000    ; AUTHOR: R. WILL, CREATION DATE: 17-Apr-79
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       , : VERSION 00
0000    ; 0-001 - Original
0000    ; 1-002 - Changed shared external references to G^ RNH 25-Sep-81
```

```
    0000          .SBTTL  DECLARATIONS
    0000  ;
    0000  ; INCLUDE FILES:
    0000  ;
    0000
    0000          $DSCDEF                                 ; Define descriptor offsets
    0000
    0000  ;
    0000  ; EXTERNAL DECLARATIONS:
    0000  ;
    0000          .DSABL  GBL                             ; Prevent undeclared
    0000                                                  ; symbols from being
    0000                                                  ; automatically global.
    0000
    0000          .EXTRN  BAS$$STOP                       ; Routine to signal errors
    0000
    0000  ;
    0000  ;       The following are error messages that may be signalled
    0000  ;
    0000          .EXTRN  BAS$K_CANCHAARR         ; Cannot Change Array Dimensions
    0000                                          ; DSC$V_FL_REDIM is not set or
    0000                                          ; DSC$V_FL_BOUNDS is not set
    0000          .EXTRN  BAS$K_MATDIMERR         ; Matrix Dimension Error
    0000                                          ; Number of input parameters differs
    0000                                          ;  from number of dimensions in array
    0000          .EXTRN  BAS$K_REDARR            ; Redimensioned Array
    0000                                          ; Not enough space allocated for input
    0000                                          ;  dimension parameters
    0000          .EXTRN  BAS$K_SUBOUTRAN         ; Subscript Out Of Range
    0000                                          ; One of the new bounds specified was
    0000                                          ;  negative or zero
    0000
    0000  ;
    0000  ; MACROS:
    0000  ;
    0000
    0000  ;
    0000  ; EQUATED SYMBOLS:
    0000  ;
    0000
    0000  ;
    0000  ; OWN STORAGE:
    0000  ;
    0000
    0000  ;
    0000  ; PSECT DECLARATIONS:
    0000  ;
00000000          .PSECT _BAS$CODE PIC, USR, CON, REL, LCL, SHR, -
    0000                          EXE, RD, NOWRT, LONG
    0000
```

```
                0000             .SBTTL  BAS$MAT_REDIM  - Redimension a BASIC array
                0000     ;++
                0000     ; FUNCTIONAL DESCRIPTION:
                0000     ;
                0000     ;       This routine redimensions arrays for BASIC-PLUS-2.  It first checks to
                0000     ;       see if the array is currently the size that the call is requesting
                0000     ;       redimensioning to.  If so, the routine will return (so that if
                0000     ;       DSC$V_FL_REDIM is not set an error will not be signalled).  Otherwise
                0000     ;       it will signal an error if DSC$V_FL_REDIM is not set, or if
                0000     ;       2 dimensions are input and the matrix currently only has 1 dimension
                0000     ;       or only 1 dimension is input and the matrix currently has 2 dimensions
                0000     ;       or if the input dimensions require more space than originally
                0000     ;       allocated to the array (DSC$L_ARSIZE).  If there are no errors, then
                0000     ;       the routine will write either 1 or 2 upper bounds and the same
                0000     ;       number of multipliers, and set to 0 the same number of lower bounds.
                0000     ;       DSC$A_A0 will be set to DSC$A_POINTER.  Also
                0000     ;       note that total array size will always remain the initial allocated
                0000     ;       length.  Integer overflow is disabled so that a BASIC-PLUS-2 error
                0000     ;       can be signalled if the space needed is too large instead of getting
                0000     ;       a hardware error.
                0000     ;
                0000     ; CALLING SEQUENCE:
                0000     ;
                0000     ;       CALL BAS$MAT_REDIM (matrix.wx.da, rows.rl.v [, cols.rl.v])
                0000     ;
                0000     ; INPUT PARAMETERS:
                0000     ;
       00000008 0000             row_upr_bnd = 8
       0000000C 0000             col_upr_bnd = 12
                0000     ;
                0000     ; IMPLICIT INPUTS:
                0000     ;
                0000     ;       NONE
                0000     ;
                0000     ; OUTPUT PARAMETERS:
                0000     ;
       00000004 0000             matrix = 4
                0000     ;
                0000     ; IMPLICIT OUTPUTS:
                0000     ;
                0000     ;       NONE
                0000     ;
                0000     ; FUNCTION VALUE:
                0000     ; COMPLETION CODES:
                0000     ;
                0000     ;       NONE
                0000     ;
                0000     ; SIDE EFFECTS:
                0000     ;
                0000     ;       Errors list under externals may be signalled.  The matrix parameter
                0000     ;       may have different dimensions after routine execution.
                0000     ;
                0000     ;--
                0000
           001C 0000             .ENTRY BAS$MAT_REDIM , ^M<R2,R3,R4>
                0002                                                     ; Routine to redimension array
                0002                                                     ; according to BASIC-PLUS-2
```

```
                              0002                                                       ; syntax
                              0002
                              0002  ;+
                              0002  ; Register usage
                              0002  ;         R0, R1  computation temps
                              0002  ;         R2      pointer to output matrix descriptor
                              0002  ;         R3      upper bound for subscript 1 (rows)
                              0002  ;         R4      upper bound for subscript 2 (columns)
                              0002  ;-
                              0002
          52    04 AC    D0   0002          MOVL    matrix(AP), R2                   ; pointer to array descriptor
                              0006
                              0006  ;+
                              0006  ; First check to see if the bounds are set in the array descriptor
                              0006  ; if they are not, we cannot change them.  \is this really an error\
                              0006  ;-
                              0006
       0B 0A A2    07    E0   0006          BBS     #DSC$V_FL_BOUNDS, DSC$B_AFLAGS(R2), 1$  ; If bounds are
                              000B                                                       ; present, go check them
          7E     00'8F   9A   000B          MOVZBL  #BAS$K_CANCHAARR, -(SP)          ; Bounds are not present
    00000000'GF    01    FB   000F          CALLS   #1, G^BAS$$STOP                  ; signal redimension error
                              0016
                              0016  ;+
                              0016  ; Bounds are present.  Take execution path depending on number
                              0016  ; of dimensions.
                              0016  ;-
                              0016
             02    6C    91   0016  1$:     CMPB    (AP), #2                         ; find # of bounds input
                   69    1F   0019          BLSSU   RETURN                           ; no bounds input, so exit
                              001B                                                   ; \should that be an error\
                              001B                                                   ; \should I even check for it\
                   74    1A   001B          BGTRU   TWO_DIMS                         ; go do more than 1 bound
                              001D
```

```
                         001D      ;+
                         001D      ; One dimension was input.
                         001D      ;-
                         001D
                         001D      ;+
                         001D      ; Put new bound into R3.  If it is negative or zero signal an error
                         001D      ;-
                         001D
      53    08 AC   D0   001D               MOVL    row_upr_bnd(AP), R3             ; get the new bound
            56      15   0021               BLEQ    ERR4                            ; error
                         0023
                         0023      ;+
                         0023      ; Check to see if array is one dimensional.  If not signal an error.
                         0023      ;-
                         0023
      01    08 A2   91   0023               CMPB    DSC$B_DIMCT(R2), #1             ; One dimensional array?
            2F      12   0027               BNEQU   ERR1                            ; No, go signal error
                         0029
                         0029      ;+
                         0029      ; Check to see if the new bound is the same as the old bound.
                         0029      ; If so return.
                         0029      ;-
                         0029
   08 AC    1C A2   D1   0029               CMPL    DSC$L_M1+8(R2), row_upr_bnd(AP) ; Yes, compare bounds
            54      13   002E               BEQL    RETURN                          ; Array is already desired
                         0030                                                       ; size, so return
                         0030
                         0030      ;+
                         0030      ; See if array is redimensionable.  If not, signal an error.  Note that
                         0030      ; we must check for correct size before redimensionability so that we
                         0030      ; won't give the 'can't redimension' error when array is correct size.
                         0030      ;-
                         0030
   39 0A A2    04    E1  0030               BBC     #DSC$V_FL_REDIM, DSC$B_AFLAGS(R2), ERR3 ; if can't redimension
                         0035                                                       ; array, go signal an error
                         0035
                         0035      ;+
                         0035      ; Compute array size needed for new bounds.  If more space is needed
                         0035      ; than is currently allocated to the array signal an error.
                         0035      ;-
                         0035
        50    62    3C   0035               MOVZWL  DSC$W_LENGTH(R2), R0            ; make item length a longword
  51    01    08 AC C1   0038               ADDL3   row_upr_bnd(AP), #1, R1         ; Add 1 to upper bound since
                         003D                                                       ; BASIC-PLUS-2 ARRAYS have 0
                         003D                                                       ; for a lower bound
        51    50    C4   003D               MULL2   R0, R1                          ; find space need w/ new bound
              21    1D   0040               BVS     ERR2                            ; not enuf space, go signal
     0C A2    51    D1   0042               CMPL    R1, DSC$L_ARSIZE(R2)            ; see if too much space needed
              'B    14   0046               BGTR    ERR2                            ; too much space, go signal
                         0048      ;\what is supposed to to unsigned and what is not?\
                         0048
                         0048      ;+
                         0048      ; All errors have been caught.  Now redimension array.
                         0048      ;-
                         0048
   1C A2    08 AC   D0   0048               MOVL    row_upr_bnd(AP), DSC$L_M1+8(R2) ; write new upper bound
            18 A2   D4   004D               CLRL    DSC$L_M1+4(R2)                  ; set lower bound to 0 since
```

```
                         0050   247                                                   ; B+2 arrays have 0 lower bnd
      14 A2   01   08 AC   C1  0050   248        ADDL3   row_upr_bnd(AP), #1, DSC$L_M1(R2) ; compute and write new
                         0056   249                                                   ; multiplier, assuming that
                         0056   250                                                   ; all B+2 arrays have 0 for L1
                    2C   11  0056   251        BRB     RETURN                         ; and exit
                         0058   252
                         0058   253
```

```
                               0058      ;+
                               0058      ; Signal errors
                               0058      ;-
                               0058
         7E   00'8F       9A   0058      ERR1:   MOVZBL  #BAS$K_MATDIMERR, -(SP)        ; Matrix Dimension Error
   00000000'GF      01    FB   005C              CALLS   #1, G^BAS$$STOP               ; Signal the error
                               0063
         7E   00'8F       9A   0063      ERR2:   MOVZBL  #BAS$K_REDARR, -(SP)           ; Redimensioned Array
   00000000'GF      01    FB   0067              CALLS   #1, G^BAS$$STOP               ; Signal the error
                               006E
         7E   00'8F       9A   006E      ERR3:   MOVZBL  #BAS$K_CANCHAARR, -(SP)        ; Can't Change Array Dimension
   00000000'GF      01    FB   0072              CALLS   #1, G^BAS$$STOP
                               0079
         7E   00'8F       9A   0079      ERR4:   MOVZBL  #BAS$K_SUBOUTRAN, -(SP)        ; Subscript out of range
   00000000'GF      01    FB   007D              CALLS   #1, G^BAS$$STOP
                               0084
                          04   0084      RETURN: RET                                    ; and exit
                               0085
```

```
                       0085  ;+
                       0085  ; Two dimensions were input.
                       0085  ;-
                       0085
                       0085  ;+
                       0085  ; Put the 2 new upper bounds into registers.  If either is negative or 0,
                       0085  ; signal an error.
                       0085  ;-
                       0085
      53    08 AC  D0  0085          MOVL    row_upr_bnd(AP), R3             ; new upper bound
               EE  15  0089          BLEQ    ERR4                            ; error
      54    0C AC  D0  008B          MOVL    col_upr_bnd(AP), R4             ; new upper bound
               E8  15  008F          BLEQ    ERR4                            ; error
                       0091
                       0091  ;+
                       0091  ; Check to see if array is two dimensional.  If not signal an error.
                       0091  ;-
                       0091
                       0091  TWO_DIMS:
      02    0B A2  91  0091          CMPB    DSC$B_DIMCT(R2), #2             ; Two dimensional array?
               C1  12  0095          BNEQU   ERR1                            ; No, go signal error
                       0097
                       0097  ;+
                       0097  ; Check to see if the new bounds are the same as the old bounds.
                       0097  ; If so return.
                       0097  ;-
                       0097
   08 AC    20 A2  D1  0097          CMPL    DSC$L_M2+8(R2), row_upr_bnd(AP) ; Yes, compare number of rows
               07  12  009C          BNEQ    25$                             ; Not =, continue redimension
   0C AC    28 A2  D1  009E          CMPL    DSC$L_M2+16(R2), col_upr_bnd(AP) ; compare number of columns
               DF  13  00A3          BEQL    RETURN                          ; Array is already desired
                       00A5                                                  ; size, so return
                       00A5
                       00A5  ;+
                       00A5  ; See if array is redimensionable.  If not, signal an error.  Note that
                       00A5  ; we must check for correct size before redimensionability so that we
                       00A5  ; won't give the 'can't redimension' error when array is correct size.
                       00A5  ;-
                       00A5
 C4 0A A2       04  E1  00A5  25$:     BBC     #DSC$V_FL_REDIM, DSC$B_AFLAGS(R2), ERR3 ; if can't redimension
                       00AA                                                  ; array, go signal an error
                       00AA
                       00AA  ;+
                       00AA  ; Compute array size needed for new bounds.  If more space is needed
                       00AA  ; than is currently allocated to the array signal an error.
                       00AA  ;-
                       00AA
         50    62  3C  00AA          MOVZWL  DSC$W_LENGTH(R2), R0            ; make item length a longword
  51  01    08 AC  C1  00AD          ADDL3   row_upr_bnd(AP), #1, R1         ; Add 1 to upper bound since
                       00B2                                                  ; BASIC-PLUS-2 arrays have 0
                       00B2                                                  ; for a lower bound for rows
               AF  1D  00B2          BVS     ERR2                            ; not enuf space, go signal
         51    50  C4  00B4          MULL2   R0, R1                          ; find space need for a column
               AA  1D  00B7          BVS     ERR2                            ; not enuf space, go signal
  50  01    0C AC  C1  00B9          ADDL3   col_upr_bnd(AP), #1, R0         ; Add 1 to upper bound since
                       00BE                                                  ; BASIC-PLUS-2 arrays have 0
                       00BE                                                  ; for a lower bound for column
```

```
                 A3  1D  00BE   331            BVS     ERR2                            ; not enuf space, go signal
             51  50  C4  00C0   332            MULL2   R0, R1                          ; find total space needed
                 9E  1D  00C3   333            BVS     ERR2                            ; not enuf space, go signal
          0C A2  51  D1  00C5   334            CMPL    R1, DSC$L_ARSIZE(R2)            ; see if too much space needed
                 98  14  00C9   335            BGTR    ERR2                            ; too much space, go signal
                         00CB   336 ;\what is supposed to to unsigned and what is not?\
                         00CB   337
                         00CB   338 ;+
                         00CB   339 ; All errors have been caught.  Now redimension array.
                         00CB   340 ;-
                         00CB   341
              1C A2  D4  00CB   342            CLRL    DSC$L_M2+4(R2)                  ; set lower bound to 0
       20 A2  08 AC  D0  00CE   343            MOVL    row_upr_bnd(AP), DSC$L_M2+8(R2) ; write new upper bnd for rows
  14 A2   01  08 AC  C1  00D3   344            ADDL3   row_upr_bnd(AP), #1, DSC$L_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$L_M2+12(R2)                 ; set lower bound to 0
       28 A2  0C AC  D0  00DC   348            MOVL    col_upr_bnd(AP), DSC$L_M2+16(R2) ; write new upper bnd for col
  18 A2   01  0C AC  C1  00E1   349            ADDL3   col_upr_bnd(AP), #1, DSC$L_M2(R2) ; compute and write new col
                         00E7   350                                                    ; multiplier, assuming that
                         00E7   351                                                    ; all B+2 arrays have 0 for L2
                         00E7   352
                 9B  11  00E7   353            BRB     RETURN                          ; and exit
                         00E9   354
                         00E9   355            .END                                    ; End of BAS$MAT_REDIM
```

```
BAS$$STOP                        ********   X   00
BAS$K_CANCHAARR                  ********   X   00
BAS$K_MATDIMERR                  ********   X   00
BAS$K_REDARR                     ********   X   00
BAS$K_SUBOUTRAN                  ********   X   00
BAS$MAT_REDIM                    00000000 RG    02
COL_UPR_BND                    = 0000000C
DSC$B_AFLAGS                   = 0000000A
DSC$B_DIMCT                    = 0000000B
DSC$L_ARSIZE                   = 0000000C
DSC$L_M1                       = 00000014
DSC$L_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
$ABS$                           00000000  (    0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
_BAS$CODE                       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

BASMATMUL
LIS

BASMATIO
LIS

BASMATRED
LIS

BASMATNUL
LIS

BASMATINV
LIS

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

BASMATSUB
LIS

BASMATSCA
LIS

BASMATTRN
LIS