```
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  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   SSSSSSSSSSS     RRR         RRR      TTT        LLLLLLLLLLLLLLL
BBBBBBBBBBBB      AAA         AAA   SSSSSSSSSSS     RRR         RRR      TTT        LLLLLLLLLLLLLLL
BBBBBBBBBBBB      AAA         AAA   SSSSSSSSSSS     RRR         RRR      TTT        LLLLLLLLLLLLLLL
```

```
BBBBBBBB      AAAAAA      SSSSSSSS  MM      MM    AAAAAA    TTTTTTTTTT  NN      NN  UU      UU  LL
BBBBBBBB      AAAAAA      SSSSSSSS  MM      MM    AAAAAA    TTTTTTTTTT  NN      NN  UU      UU  LL
BB      BB   AA      AA  SS        MMMM  MMMM   AA      AA      TT      NN      NN  UU      UU  LL
BB      BB   AA      AA  SS        MMMM  MMMM   AA      AA      TT      NN      NN  UU      UU  LL
BB      BB   AA      AA  SS        MM  MM  MM   AA      AA      TT      NNNN    NN  UU      UU  LL
BB      BB   AA      AA  SS        MM  MM  MM   AA      AA      TT      NNNN    NN  UU      UU  LL
BBBBBBBB     AA      AA    SSSSSS  MM      MM   AA      AA      TT      NN  NN  NN  UU      UU  LL
BBBBBBBB     AA      AA    SSSSSS  MM      MM   AA      AA      TT      NN  NN  NN  UU      UU  LL
BB      BB   AAAAAAAAAA          SS  MM      MM   AAAAAAAAAA      TT      NN    NNNN  UU      UU  LL
BB      BB   AAAAAAAAAA          SS  MM      MM   AAAAAAAAAA      TT      NN    NNNN  UU      UU  LL
BB      BB   AA      AA          SS  MM      MM   AA      AA      TT      NN      NN  UU      UU  LL
BB      BB   AA      AA          SS  MM      MM   AA      AA      TT      NN      NN  UU      UU  LL
BBBBBBBB     AA      AA  SSSSSSSS    MM      MM   AA      AA      TT      NN      NN  UUUUUUUUUU  LLLLLLLLLL
BBBBBBBB     AA      AA  SSSSSSSS    MM      MM   AA      AA      TT      NN      NN  UUUUUUUUUU  LLLLLLLLLL
```

```
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
```

```
0000     1          .TITLE  BAS$MAT_NULL
0000     2          .IDENT  /1-004/             ; File: BASMATNUL.MAR EDIT:MDL1004
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 initializes a string matrix to have a null string
0000    35  ;       for each element.
0000    36  ;
0000    37  ; ENVIRONMENT: User Mode, AST Reentrant
0000    38  ;
0000    39  ;--
0000    40  ; AUTHOR: R. Will, CREATION DATE: 30-May-79
0000    41  ;
0000    42  ; MODIFIED BY:
0000    43  ;
0000    44  ; 1-001 - Original
0000    45  ; 1-002 - Make references to bounds signed.  Rw 7-Jun-79
0000    46  ; 1-003 - Change shared external references to G^ RNH 25-Sep-81
0000    47  ; 1-004 - make ALL external references G^.  MDL 26-May-1983
```

```
                     0000     49          .SBTTL  DECLARATIONS
                     0000     50 ;
                     0000     51 ; INCLUDE FILES:
                     0000     52 ;
                     0000     53
                     0000     54          $DSCDEF                                 ; define descriptors
                     0000     55
                     0000     56 ;
                     0000     57 ; EXTERNAL DECLARATIONS:
                     0000     58 ;
                     0000     59
                     0000     60          .DSABL  GBL                             ; Prevent undeclared
                     0000     61                                                  ;  symbols from being
                     0000     62                                                  ;  automatically global.
                     0000     63          .EXTRN  BAS$K_ARGDONMAT                 ; signalled if all 3 blocks
                     0000     64                                                  ;  not present in array desc
                     0000     65                                                  ;  or dimct = 0
                     0000     66          .EXTRN  BAS$STORE_BFA                   ; array element store
                     0000     67          .EXTRN  BAS$$STOP                       ; signal fatal errors
                     0000     68
                     0000     69 ;
                     0000     70 ; MACROS:
                     0000     71 ;
                     0000     72
                     0000     73 ;
                     0000     74 ; EQUATED SYMBOLS:
                     0000     75 ;
                     0000     76
0000000C             0000     77          index1 = 12                             ; stack offset of index1
                     0000     78                                                  ;  before call to store
00000010             0000     79          index2 = 16                             ; stack offset of index2
                     0000     80                                                  ;  before call to store
00000018             0000     81          dsc$l_l1_1 = 24                         ; desc offset if 1 sub
0000001C             0000     82          dsc$l_u1_1 = 28                         ; desc offset if 1 sub
0000001C             0000     83          dsc$l_l1_2 = 28                         ; desc offset if 2 sub
00000020             0000     84          dsc$l_u1_2 = 32                         ; desc offset if 2 sub
00000024             0000     85          dsc$l_l2_2 = 36                         ; desc offset if 2 sub
00000028             0000     86          dsc$l_u2_2 = 40                         ; desc offset if 2 sub
                     0000     87
                     0000     88 ;
                     0000     89 ; OWN STORAGE:
                     0000     90 ;
                     0000     91
                     0000     92 ;
                     0000     93 ; PSECT DECLARATIONS:
                     0000     94 ;
                00000000     95          .PSECT _BAS$CODE PIC, USR, CON, REL, LCL, SHR, -
                     0000     96                      EXE, RD, NOWRT, LONG
                     0000     97
```

```
                     0000    99          .SBTTL  BAS$MAT_NULL  - Initialize a string matrix to null matrix
                     0000   100 ;++
                     0000   101 ; FUNCTIONAL DESCRIPTION:
                     0000   102 ;
                     0000   103 ;       This routine initializes each element of a string array to the
                     0000   104 ;       null string.  It does this by copying a null string to each elemnt.
                     0000   105 ;
                     0000   106 ; CALLING SEQUENCE:
                     0000   107 ;
                     0000   108 ;       CALL BASMAT_NULL (matrix.wx.da)
                     0000   109 ;
                     0000   110 ; INPUT PARAMETERS:
                     0000   111 ;
                     0000   112 ;       NONE
                     0000   113 ;
                     0000   114 ; IMPLICIT INPUTS:
                     0000   115 ;
                     0000   116 ;       NONE
                     0000   117 ;
                     0000   118 ; OUTPUT PARAMETERS:
                     0000   119 ;
00000004             0000   120 ;       matrix = 4
                     0000   121 ;
                     0000   122 ; IMPLICIT OUTPUTS:
                     0000   123 ;
                     0000   124 ;       NONE
                     0000   125 ;
                     0000   126 ; FUNCTION VALUE:
                     0000   127 ; COMPLETION CODES:
                     0000   128 ;
                     0000   129 ;       NONE
                     0000   130 ;
                     0000   131 ; SIDE EFFECTS:
                     0000   132 ;
                     0000   133 ;       This routine will call the BASIC array fetch and store routines and so
                     0000   134 ;       may cause any of their errors to be signalled.  It may also signal any
                     0000   135 ;       of the errors listed in the externals section.
                     0000   136 ;
                     0000   137 ;--
                     0000   138
                     0000   139
                     0000   140 ;+
                     0000   141 ;       REGISTER USAGE
                     0000   142 ;       R2      current bound for 1st subscript
                     0000   143 ;       R3      upper bound for 1st subscript
                     0000   144 ;       R4      lower bound for 2nd subscript
                     0000   145 ;       R5      upper bound for 2nd subscript
                     0000   146 ;       R6      current value of 2nd subscript
                     0000   147 ;-
                     0000   148
                 407C 0000  149          .ENTRY BAS$MAT_NULL, ^M<R2,R3,R4,R5,R6,IV>
                     0002   150
                     0002   151 ;+
                     0002   152 ; If block 2 of array descriptor (multipliers) is not present then error.
                     0002   153 ;-
                     0002   154
  50   04 AC    D0  0002   155          MOVL    matrix(AP), R0                  ; get pointer to mat descriptr
```

```
                 6E 0A A0    07   E1  0006   156          BBC     #DSC$V_FL_BOUNDS, DSC$B_AFLAGS(R0), ERR_ARGDONMAT
                                      000B   157                                                      ; exit if block 3 not
                                      000B   158                                                      ; present in descriptor
                                      000B   159
                                      000B   160 ;+
                                      000B   161 ; Initialize stack for CALLG to element store routine.  Then divide algc  thm
                                      000B   162 ; based on number of subscripts.
                                      000B   163 ;-
                                      000B   164
                           7E    D4   000B   165          CLRL    -(SP)                                   ; pointer of descriptor
                   010E0000 8F    DD   000D   166          PUSHL   #<<DSC$K_CLASS_S @ 24> + <DSC$K_DTYPE_T @ 16>> ; 1st desc word
                                      0013   167                                                      ; class, type and 0 length
                           7E    7C   0013   168          CLRQ    -(SP)                                   ; space for indices
                           50    DD   0015   169          PUSHL   R0                                      ; ptr to dest desc for call
                        0C AE    DF   0017   170          PUSHAL  12(SP)                                  ; pointer to NULL descriptor
                     01 0B A0    91   001A   171          CMPB    DSC$B_DIMCT(R0), #1                     ; determine # of subscripts
                           05    13   001E   172          BEQLU   INIT_ONE_SUB                            ; 1 sub, go init
                           1A    1A   0020   173          BGTRU   INIT_TWO_SUBS                           ; >=2 subs, go init
                         0054    31   0022   174          BRW     ERR_ARGDONMAT                           ; 0 subs, error
                                      0025   175
                                      0025   176 ;+
                                      0025   177 ; There is only 1 subscript.  Make both upper and lower bound for 2nd
                                      0025   178 ; subscript a 1.  A second subscript will be passed to and ignored by the
                                      0025   179 ; store routine because the argcount in the arglist of the CALL will not
                                      0025   180 ; include the 2nd subscript.  Put bound for 1st subscript into registers.
                                      0025   181 ;-
                                      0025   182
                                      0025   183 INIT_ONE_SUB:
                           03    DD   0025   184          PUSHL   #3                                      ; 3 arguments to store routine
                                      0027   185                                                      ;  ignores 2nd index
                     53    1C A0    D0   0027   186          MOVL    dsc$l_u1_1(R0), R3                      ; 1st upper bound
                     52    18 A0    D0   002B   187          MOVL    dsc$l_l1_1(R0), R2                      ; 1st lower bound
                           03    14   002F   188          BGTR    1$                                      ; not 0 or neg, init 2nd bound
                        52 01    D0   0031   189          MOVL    #1, R2                                  ; don't alter row 0
                        54 01    D0   0034   190 1$:      MOVL    #1, R4                                  ; set 2nd lower bnd to 1
                        55 01    D0   0037   191          MOVL    #1, R5                                  ; set 2nd upper bnd to 1
                           1C    11   003A   192          BRB     LOOP_1ST_SUB                            ; go loop
                                      003C   193
                                      003C   194 ;+
                                      003C   195 ; There are 2 subscripts.  Put the upper bound for both subscripts in
                                      003C   196 ; registers and make sure that the lower bound for both subscripts will start
                                      003C   197 ; at 1 (do not alter row or col 0)
                                      003C   198 ;-
                                      003C   199
                                      003C   200 INIT_TWO_SUBS:
                           04    DD   003C   201          PUSHL   #4                                      ; 4 arguments to store routine
                     55    28 A0    D0   003E   202          MOVL    dsc$l_u2_2(R0), R5                      ; 2nd upper bound
                     54    24 A0    D0   0042   203          MOVL    dsc$l_l2_2(R0), R4                      ; 2nd lower bound
                           03    14   0046   204          BGTR    1$                                      ; not col 0 or neg, do cols
                        54 01    D0   0048   205          MOVL    #1, R4                                  ; start with col 1
                     53    20 A0    D0   004B   206 1$:      MOVL    dsc$l_u1_2(R0), R3                      ; 1st upper bound
                     52    1C A0    D0   004F   207          MOVL    dsc$l_l1_2(R0), R2                      ; 1st lower bound
                           03    14   0053   208          BGTR    LOOP_1ST_SUB                            ; not row 0 or neg, go loop
                        52 01    D0   0055   209          MOVL    #1, R2                                  ; start with row 1
                                      0058   210
                                      0058   211 ;+
                                      0058   212 ; Loop through all the rows.  Row and column upper and lower bounds have been
```

```
                     0058   213 ; initialized in registers.
                     0058   214 ;-
                     0058   215
                     0058   216 LOOP_1ST_SUB:
          56   54    D0  0058   217         MOVL    R4, R6                          ; R6 has 1st lower bound
                     005B   218
                     005B   219 ;+
                     005B   220 ; Loop through all the elements (columns) of the current row.  Column lower
                     005B   221 ; bound is initialized in R6.  Column upper bound is in R5.
                     005B   222 ;-
                     005B   223
                     005B   224 LOOP_2ND_SUB:
                     005B   225
       10 AE    56   D0  005B   226         MOVL    R6, index2(SP)                  ; current column
       0C AE    52   D0  005F   227         MOVL    R2, index1(SP)                  ; current row
00000000'GF     6E   FA  0063   228         CALLG   (SP), G^BAS$STORE_BFA           ; store in array
          56   56   D6  006A   229         INCL    R6                              ; get next column
       55   56   D1  006C   230         CMPL    R6, R5                          ; see if last column done
            EA   15  006F   231         BLEQ    LOOP_2ND_SUB                    ; no, continue inner loop
                     0071   232
                     0071   233 ;+
                     0071   234 ; Have completed entire row.  See if it was the last row.  If not,
                     0071   235 ; continue with next row.
                     0071   236 ;-
                     0071   237
          52   D6  0071   238         INCL    R2                              ; get next row
       53   52   D1  0073   239         CMPL    R2, R3                          ; see if last row done
            E0   15  0076   240         BLEQ    LOOP_1ST_SUB                    ; no, continue outer loop
                     0078   241
               04    0078   242         RET                                     ; yes, finished
                     0079   243
                     0079   244 ERR_ARGDONMAT:
     0000000C'8F    DD  0079   245         PUSHL   #BAS$K_ARGDONMAT                ; signal error, 0 for dimct
00000000'GF    01    FB  007F   246         CALLS   #1, G^BAS$$STOP                 ; or block 2 or 3 absent
                     0086   247
                     0086   248         .END                                    ; end of BAS$MAT_NULL
```

```
BAS$$STOP                       ********  X   00
BAS$K_ARGDONMAT                 ********  X   00
BAS$MAT_NULL                    00000000 RG   02
BAS$STORE_BFA                   ********  X   00
DSC$B_AFLAGS                  = 0000000A
DSC$B_DIMCT                   = 0000000B
DSC$K_CLASS_S                 = 00000001
DSC$K_DTYPE_T                 = 0000000E
DSC$L_L1_1                    = 00000018
DSC$L_L1_2                    = 0000001C
DSC$L_L2_2                    = 00000024
DSC$L_U1_1                    = 0000001C
DSC$L_U1_2                    = 00000020
DSC$L_U2_2                    = 00000028
DSC$V_FL_BOUNDS               = 00000007
ERR_ARGDONMAT                   00000079 R    02
INDEX1                        = 0000000C
INDEX2                        = 00000010
INIT_ONE_SUB                    00000025 R    02
INIT_TWO_SUBS                   0000003C R    02
LOOP_1ST_SUB                    00000058 R    02
LOOP_2ND_SUB                    0000005B R    02
MATRIX                        = 00000004
```

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                       00000086  (  134.)  02 (  2.)    PIC   USR   CON   REL   LCL   SHR  EXE   RD  NOWRT NOVEC LONG
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	28	00:00:00.08	00:00:00.43
Command processing	103	00:00:00.50	00:00:01.89
Pass 1	135	00:00:01.81	00:00:05.79
Symbol table sort	0	00:00:00.18	00:00:00.34
Pass 2	55	00:00:00.63	00:00:01.85
Symbol table output	4	00:00:00.03	00:00:00.04
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	329	00:00:03.26	00:00:10.36

```
The working set limit was 1350 pages.
9282 bytes (19 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 149 non-local and 2 local symbols.
248 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$:BASMATNUL/OBJ=OBJ$:BASMATNUL MSRC$:BASMATNUL/UPDATE=(ENH$:BASMATNUL)
```

BASMATMUL
LIS

BASMATIO
LIS

BASMATRED
LIS

BASMATNUL
LIS

BASMATINV
LIS