```
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
```

FILEID**BASMATINI

```
BBBBBBBB      AAAAAA      SSSSSSSS  MM      MM    AAAAAA    TTTTTTTTTT     IIIIII    NN      NN    IIIIII
BBBBBBBB      AAAAAA      SSSSSSSS  MM      MM    AAAAAA    TTTTTTTTTT     IIIIII    NN      NN    IIIIII
BB      BB  AA      AA  SS        MMMM  MMMM  AA      AA      TT           II      NN      NN      II
BB      BB  AA      AA  SS        MMMM  MMMM  AA      AA      TT           II      NN      NN      II
BB      BB  AA      AA  SS        MM  MM  MM  AA      AA      TT           II      NNNN    NN      II
BB      BB  AA      AA  SS        MM  MM  MM  AA      AA      TT           II      NNNN    NN      II
BBBBBBBB    AA      AA    SSSSSS  MM      MM  AA      AA      TT           II      NN  NN  NN      II
BBBBBBBB    AA      AA    SSSSSS  MM      MM  AA      AA      TT           II      NN  NN  NN      II
BB      BB  AAAAAAAAAA          SS  MM      MM  AAAAAAAAAA      TT           II      NN    NNNN      II
BB      BB  AAAAAAAAAA          SS  MM      MM  AAAAAAAAAA      TT           II      NN    NNNN      II
BB      BB  AA      AA          SS  MM      MM  AA      AA      TT           II      NN      NN      II
BB      BB  AA      AA          SS  MM      MM  AA      AA      TT           II      NN      NN      II
BBBBBBBB    AA      AA  SSSSSSSS  MM      MM  AA      AA      TT         IIIIII    NN      NN    IIIIII
BBBBBBBB    AA      AA  SSSSSSSS  MM      MM  AA      AA      TT         IIIIII    NN      NN    IIIIII

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_INIT
0000     2          .IDENT  /1-010/                 ; File: BASMATINI.MAR Edit: PLL1010
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 each element of a matrix to the input
0000    35 ;       constant.
0000    36 ;
0000    37 ; ENVIRONMENT: User Mode, AST Reentrant
0000    38 ;
0000    39 ;--
0000    40 ; AUTHOR: R. Will, CREATION DATE: 23-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 - Add support for byte, g and h floating.  PLL 17-Sep-81
0000    47 ; 1-004 - Change shared external references to G^ RNH 25-Sep-81
0000    48 ; 1-005 - Substitute a macro for the calls to the store routines.
0000    49 ;         This should speed things up.  PLL 6-Nov-81
0000    50 ; 1-006 - STORE macro must handle g & h floating.  PLL 11-Nov-81
0000    51 ; 1-007 - Correct a run-time expression in the FETCH and STORE macros.
0000    52 ;         PLL 20-Jan-82
0000    53 ; 1-008 - Correct another bug in the STORE macro.  Does not compute
0000    54 ;         linear index for one dimensional arrays properly.  PLL 23-Feb-82
0000    55 ; 1-009 - Add code in mainline code to support arrays of descriptors.
0000    56 ;         LEB 28-JUN-1982.
0000    57 ; 1-010 - Change own storage to stack storage.  PLL 9-Jul-1982
```

```
0000     58 ;--
```

```
                        .SBTTL  DECLARATIONS
                ;
                ; INCLUDE FILES:
                ;

                        $DSCDEF
                        $SFDEF

                ;
                ; EXTERNAL DECLARATIONS:
                ;
                        .DSABL  GBL                     ; Prevent undeclared
                                                        ;  symbols from being
                                                        ;  automatically global.
                        .EXTRN  BAS$K_ARGDONMAT         ; signalled if all 3 blocks
                                                        ;  not present in array desc
                                                        ;  or dimct = 0
                        .EXTRN  BAS$K_DATTYPERR         ; signalled if dtype of array
                                                        ; isn't word long float double
                        .EXTRN  BAS$STO_FA_B_R8         ; array element store for byte
                        .EXTRN  BAS$STO_FA_W_R8         ; array element store for word
                        .EXTRN  BAS$STO_FA_L_R8         ; array element store for long
                        .EXTRN  BAS$STO_FA_F_R8         ; array element store - float
                        .EXTRN  BAS$STO_FA_D_R8         ; array element store - double
                        .EXTRN  BAS$STO_FA_G_R8         ; array element store - gfloat
                        .EXTRN  BAS$STO_FA_H_R8         ; array element store - hfloat
                        .EXTRN  BAS$$SCALE_RT           ; get the scale for double
                        .EXTRN  BAS$$STOP               ; signal fatal errors
                        .EXTRN  BAS$STORE_BFA

                ;
                ; MACROS:
                ;

                ;       $BAS$MAT_INIT   see below, defines entire initialization algorithm
                ;       STORE           store an element into an array

                ;
                ; EQUATED SYMBOLS:
                ;

00000000                lower_bnd2 = 0                  ; stack offset for temp
00000004                lower_bnd1 = 4                  ; stack offset for temp
00000008                upper_bnd1 = 8                  ; stack offset for temp
0000000C                value_desc = 12                 ; output descriptor
0000000C                str_len = 12                    ; length field within desc
0000000E                dtype = 14                      ; data type field in desc
0000000F                class = 15                      ; class field in desc
00000010                pointer = 16                    ; pointer field in desc
00000014                data = 20                       ; data field (4 longwords)
00000024                constant_cvt = 36               ; stack offset, converted const
                                                        ;  may be hfloat
00000018                dsc$l_l1_1 = 24                 ; desc offset if 1 sub
0000001C                dsc$l_u1_1 = 28                 ; desc offset if 1 sub
0000001C                dsc$l_l1_2 = 28                 ; desc offset if 2 sub
00000020                dsc$l_u1_2 = 32                 ; desc offset if 2 sub
00000024                dsc$l_l2_2 = 36                 ; desc offset if 2 sub
```

```
00000028  0000   117          dsc$l_u2_2 = 40                         ; desc offset if 2 sub
          0000   118
          0000   119 ;
          0000   120 ; OWN STORAGE:
          0000   121 ;
          0000   122
          0000   123 ;          NONE
          0000   124
          0000   125 ;
          0000   126 ; PSECT DECLARATIONS:
          0000   127 ;
          00000000   128          .PSECT _BAS$CODE PIC, USR, CON, REL, LCL, SHR, -
          0000   129                         EXE, RD, NOWRT, LONG
          0000   130
```

```
                     0000   132          .SBTTL  BAS$MAT_INIT  - Initialize a  matrix
                     0000   133 ;++
                     0000   134 ; FUNCTIONAL DESCRIPTION:
                     0000   135 ;
                     0000   136 ;       This routine initializes each element of a matrix to the
                     0000   137 ;       input constant.  The algorithm is the same for all the supported
                     0000   138 ;       BASIC data types.  In order to keep the code for all data types
                     0000   139 ;       the same and to simplify the reading, the code has been done as
                     0000   140 ;       a macro, which all the data types use varying only the letters
                     0000   141 ;       (B, W, L, F, D, G, H) in converting the constant, in passing the constant
                     0000   142 ;       and calling the array store routines.
                     0000   143 ;
                     0000   144 ; CALLING SEQUENCE:
                     0000   145 ;
                     0000   146 ;       CALL BASMAT_INIT (matrix.wx.da, constant.rl.v)
                     0000   147 ;
                     0000   148 ; INPUT PARAMETERS:
                     0000   149 ;
00000008             0000   150          constant = 8
                     0000   151 ;
                     0000   152 ; IMPLICIT INPUTS:
                     0000   153 ;
                     0000   154 ;       NONE
                     0000   155 ;
                     0000   156 ; OUTPUT PARAMETERS:
                     0000   157 ;
00000004             0000   158          matrix = 4
                     0000   159 ;
                     0000   160 ; IMPLICIT OUTPUTS:
                     0000   161 ;
                     0000   162 ;       NONE
                     0000   163 ;
                     0000   164 ; FUNCTION VALUE:
                     0000   165 ; COMPLETION CODES:
                     0000   166 ;
                     0000   167 ;       NONE
                     0000   168 ;
                     0000   169 ; SIDE EFFECTS:
                     0000   170 ;
                     0000   171 ;       This routine calls the BASIC matrix store routines, and may cause
                     0000   172 ;       any of their errors to be signalled.  It also may signal any of the
                     0000   173 ;       errors listed in the externals area.
                     0000   174 ;
                     0000   175 ;--
                     0000   176
                     0000   177
```

```
0000    179          .LIST   ME
0000    180          .MACRO  $BAS$MAT_INIT dtype             ; initialize algorithm
0000    181
0000    182 ;+
0000    183 ;       REGISTER USAGE
0000    184 ;       R0 - R8 destroyed by store routines
0000    185 ;       R9      upper bound for 2nd subscript
0000    186 ;       R10     pointer to array descriptor
0000    187 ;       R11     current value of 2nd subscript
0000    188 ;-
0000    189
0000    190 ;+
0000    191 ; Set up limits for looping through all elements
0000    192 ;-
0000    193
0000    194          .IF     IDN     dtype, L
0000    195          .IFT                                    ; data type is long
0000    196          MOVL    constant(AP), -(SP)            ; move constant
0000    197          .IFF                                    ; data type is not long
0000    198          CVTL'dtype'     constant(AP), -(SP)    ; make constant same datatype
0000    199                                                  ;  as array, save on stack
0000    200          .ENDC
0000    201          .IF     IDN     dtype, D               ; if array is double
0000    202          MOVL    SF$L_SAVE_FP(FP), R0           ; pass FP to get scale
0000    203          JSB     G^BAS$$SCALE_R1                ; get scale in R0 & R1
0000    204                                                  ; call a BLISS routine because
0000    205                                                  ; the frame offsets are only
0000    206                                                  ; defined for BLISS
0000    207          MULD2   R0, (SP)                       ; scale
0000    208          .ENDC
0000    209
0000    210 ;+
0000    211 ; Allocate data and value_desc on the stack.  This applies to both
0000    212 ; one and two dimensions.
0000    213 ;-
0000    214
0000    215          CLRQ    -(SP)                          ; space for data
0000    216          CLRQ    -(SP)                          ;  may be hfloat
0000    217          CLRQ    -(SP)                          ; space for value_desc
0000    218
0000    219          CMPB    DSC$B_DIMCT(R10), #1           ; determine # of subscripts
0000    220          BEQLU   INIT_ONE_SUB'dtype'            ; 1 sub, go init
0000    221          BGTRU   INIT_TWO_SUBS'dtype'           ; >=2 subs, go init
0000    222          BRW     ERR_ARGDONMAT                  ; 0 subs, error
0000    223
0000    224 ;+
0000    225 ; There is only 1 subscript.  Make both upper and lower bound for 2nd
0000    226 ; subscript a 1.  The second subscript will be passed to and ignored by the
0000    227 ; store routine.
0000    228 ;-
0000    229
0000    230 INIT_ONE_SUB'dtype':
0000    231          PUSHL   dsc$l_u1_1(R10)                ; 1st upper bound
0000    232          PUSHL   dsc$l_l1_1(R10)                ; 1st lower bound
0000    233          BGTR    1$                             ; not 0 or neg, do 2nd sub
0000    234          MOVL    #1, (SP)                       ; don't alter col 0
0000    235 1$:      MOVL    #1, R9                         ; dummy 2nd lower bound
```

```
0000   236          PUSHL   #1                                  ; dummy 2nd upper bound
0000   237          BRB     LOOP_2ND_SUB'dtype'                 ; go loop
0000   238
0000   239 ;+
0000   240 ; There are 2 subscripts.  Put the upper bound for both subscripts on the
0000   241 ; stack and make sure that the lower bound for both subscripts will start
0000   242 ; at 1 (do not alter row or col 0)
0000   243 ;-
0000   244
0000   245 INIT_TWO_SUBS'dtype':
0000   246          PUSHL   dsc$l_u1_2(R10)                     ; 1st upper bound
0000   247          PUSHL   dsc$l_l1_2(R10)                     ; 1st lower bound
0000   248          BGTR    1$                                  ; not row 0 or neg, do cols
0000   249          MOVL    #1, (SP)                            ; start with row 1
0000   250 1$:      MOVL    dsc$l_u2_2(R10), R9                 ; 2nd upper bound
0000   251          PUSHL   dsc$l_l2_2(R10)                     ; 2nd lower bound
0000   252          BGTR    LOOP_1ST_SUB'dtype'                 ; not col 0, go loop
0000   253          MOVL    #1, (SP)                            ; start with col 1
0000   254
0000   255 ;+
0000   256 ; Loop through all the rows.  Row and column upper and lower bounds have been
0000   257 ; initialized on the stack.
0000   258 ;-
0000   259
0000   260 LOOP_1ST_SUB'dtype':
0000   261          MOVL    lower_bnd2(SP), R11                 ; R11 has 2nd lower bound
0000   262
0000   263 ;+
0000   264 ; Loop through all the elements (columns) of the current row.  Column lower
0000   265 ; bound is initialized in R11.  Column upper bound is on the stack.
0000   266 ; Distinguish array by data type so that the correct store routine can be
0000   267 ; called and the constant can be converted to the correct type.
0000   268 ;-
0000   269
0000   270 LOOP_2ND_SUB'dtype':
0000   271
0000   272          MOV'dtype'      constant_cvt(SP), R0        ; put constant into R0
0000   273                                                      ;  R0 & R1 for double
0000   274 ;+
0000   275 ; When passed by value, hfloat takes 4 words, gfloat and double take 2 words,
0000   276 ; and all other data types take 1 longword.
0000   277 ;-
0000   278
0000   279          .IF     IDN     dtype, H                    ; data type is hfloat
0000   280          MOVL    R10, R4                             ; pointer to array desc
0000   281          MOVL    lower_bnd1(SP), R5                  ; current row
0000   282          MOVL    R11, R6                             ; current column
0000   283          .IFF
0000   284          .IF     IDN     dtype, G                    ; data type is gfloat
0000   285          MOVL    R10, R2                             ; pointer to array desc
0000   286          MOVL    lower_bnd1(SP), R3                  ; current row
0000   287          MOVL    R11, R4                             ; current column
0000   288          .IFF
0000   289          .IF     IDN     dtype, D                    ; data type is double
0000   290          MOVL    R10, R2                             ; pointer to array desc
0000   291          MOVL    lower_bnd1(SP), R3                  ; current row
0000   292          MOVL    R11, R4                             ; current column
```

```
0000   293          .IFF                                    ; all other data types
0000   294          MOVL    R10, R1                         ; pointer to array desc
0000   295          MOVL    lower_bnd1(SP), R2              ; current row
0000   296          MOVL    R11, R3                         ; current column
0000   297          .ENDC
0000   298          .ENDC
0000   299          .ENDC
0000   300          MOV'dtype'      R0, data(SP)            ; store value in value_desc
0000   301          STORE   'dtype'                         ; store in array
0000   302          INCL    R11                             ; get next column
0000   303          CMPL    R11, R9                         ; see if last column done
0000   304          BGTR    2$
0000   305          BRW     LOOP_2ND_SUB'dtype'             ; no, continue inner loop
0000   306
0000   307 ;+
0000   308 ; Have completed entire row.  See if it was the last row.  If not,
0000   309 ; continue with next row.
0000   310 ;-
0000   311
0000   312 2$:      INCL    lower_bnd1(SP)                  ; get next row
0000   313          CMPL    lower_bnd1(SP), upper_bnd1(SP)  ; see if last row done
0000   314          BGTR    3$
0000   315          BRW     LOOP_1ST_SUB'dtype'             ; no, continue outer loop
0000   316
0000   317 3$:      RET                                     ; yes, finished
0000   318          .ENDM
```

```
                        4FFC  0000   320          .ENTRY BAS$MAT_INIT , ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,IV>
                              0002   321
                              0002   322 ;+
                              0002   323 ; Put routine arguments into registers for ease of use.
                              0002   324 ; If block 2 of array descriptor (multipliers) is not present then error.
                              0002   325 ;-
                              0002   326
                    5A   04 AC   D0   0002   327          MOVL    matrix(AP), R10                 ; ptr to array descr in R10
                 3F 0A AA    07   E1   0006   328          BBC     #DSC$V_FL_BOUNDS, DSC$B_AFLAGS(R10), ERR_ARGDONMAT
                                       000B   329                                                  ; exit if block 3 not
                                       000B   330                                                  ; present in descriptor
                                       000B   331
                                       000B   332 ;+
                                       000B   333 ; Algorithm now differs according to data types
                                       000B   334 ;-
                                       000B   335
                       55   5A   D0   000B   336          MOVL    R10, R5                         ; save original pointer
              05   06    02 A5   8F   000E   337 4$:      CASEB   DSC$B_DTYPE(R5), #DSC$K_DTYPE_B, #<DSC$K_DTYPE_D - DSC$K_DTYPE_B>
                                0044' 0013   338 1$:      .WORD   BYTE-1$                         ; code for byte dtype
                                0180' 0015   339          .WORD   WORD-1$                         ; code for word dtype
                                02BC' 0017   340          .WORD   LONG-1$                         ; code for long dtype
                                002A' 0019   341          .WORD   ERR_DATTYPERR-1$                ; quad not supported
                                03F8' 001B   342          .WORD   FLOAT-1$                        ; code for float dtype
                                0534' 001D   343          .WORD   DOUBLE-1$                       ; code for double dtype
                                      001F   344
                                      001F   345 ;+
                                      001F   346 ; G and H floating fall outside the range of the CASEB.
                                      001F   347 ;-
                                      001F   348
                          1B    02 A5   91   001F   349          CMPB    DSC$B_DTYPE(R5), #DSC$K_DTYPE_G
                                03   12   0023   350          BNEQ    2$
                                 0668   31   0025   351          BRW     GFLOAT                          ; code for gfloat dtype
                                        0028   352
                          1C    02 A5   91   0028   353 2$:      CMPB    DSC$B_DTYPE(R5), #DSC$K_DTYPE_H
                                03   12   002C   354          BNEQ    3$
                                 07A2   31   002E   355          BRW     HFLOAT
                                        0031   356
                          18    02 A5   91   0031   357 3$:      CMPB    DSC$B_DTYPE(R5), #DSC$K_DTYPE_DSC
                                06   12   0035   358          BNEQ    ERR_DATTYPERR
                          55    04 A5   D0   0037   359          MOVL    4(R5), R5                       ; R5 <-- addr of desc
                                D1   11   003B   360          BRB     4$                              ; CASE again on dtype in desc
                                      003D   361
                                      003D   362 ERR_DATTYPERR:
                    00000000'8F   DD   003D   363          PUSHL   #BAS$K_DATTYPERR                ; Signal error, unsupported
          00000000'GF   01   FB   0043   364          CALLS   #1, G^BAS$$STOP                 ; dtype in array desc
                                      004A   365
                                      004A   366 ERR_ARGDONMAT:
                    00000000'8F   DD   004A   367          PUSHL   #BAS$K_ARGDONMAT                ; signal error, 0 for dimct
          00000000'GF   01   FB   0050   368          CALLS   #1, G^BAS$$STOP                 ; or block 2 or 3 absent
                                      0057   369
```

```
                     0057   371 BYTE:   $BAS$MAT_INIT B                          ; expand to byte operations
                     0057
                     0057       ;+
                     0057       ;       REGISTER USAGE
                     0057       ;       R0 - R8 destroyed by store routines
                     0057       ;       R9      upper bound for 2nd subscript
                     0057       ;       R10     pointer to array descriptor
                     0057       ;       R11     current value of 2nd subscript
                     0057       ;-
                     0057
                     0057       ;+
                     0057       ; Set up limits for looping through all elements
                     0057       ;-
                     0057
                     0057               .IF     IDN     B, L
                     0057               .IFT
                     0057               MOVL    constant(AP), -(SP)                ; data type is long
                     0057                                                          ; move constant
                     0057               .IFF                                       ; data type is not long
7E   08 AC   F6      0057               CVTLB   constant(AP), -(SP)        ; make constant same datatype
                     005B                                                          ;  as array, save on stack
                     005B               .ENDC
                     005B               .IF     IDN     B, D               ; if array is double
                     005B               MOVL    SF$L_SAVE_FP(FP), R0               ; pass FP to get scale
                     005B               JSB     G^BAS$$SCALE_R1                    ; get scale in R0 & R1
                     005B                                                          ; call a BLISS routine because
                     005B                                                          ; the frame offsets are only
                     005B                                                          ; defined for BLISS
                     005B               MULD2   R0, (SP)                           ; scale
                     005B               .ENDC
                     005B
                     005B       ;+
                     005B       ; Allocate data and value_desc on the stack.  This applies to both
                     005B       ; one and two dimensions.
                     005B       ;-
                     005B
        7E   7C      005B               CLRQ    -(SP)                              ; space for data
        7E   7C      005D               CLRQ    -(SP)                              ;  may be hfloat
        7E   7C      005F               CLRQ    -(SP)                              ; space for value_desc
                     0061
01   0B AA   91      0061               CMPB    DSC$B_DIMCT(R10), #1               ; determine # of subscripts
        05   13      0065               BEQLU   INIT_ONE_SUBB              ; 1 sub, go init
        15   1A      0067               BGTRU   INIT_TWO_SUBSB             ; >=2 subs, go init
      FFDE   31      0069               BRW     ERR_ARGDONMAT                      ; 0 subs, error
                     006C
                     006C       ;+
                     006C       ; There is only 1 subscript.  Make both upper and lower bound for 2nd
                     006C       ; subscript a 1.  The second subscript will be passed to and ignored by the
                     006C       ; store routine.
                     006C       ;-
                     006C
                     006C       INIT_ONE_SUBB:
     1C AA   DD      006C               PUSHL   dsc$l_u1_1(R10)                    ; 1st upper bound
     18 AA   DD      006F               PUSHL   dsc$l_l1_1(R10)                    ; 1st lower bound
        03   14      0072               BGTR    1$                                 ; not 0 or neg, do 2nd sub
   6E   01   D0      0074               MOVL    #1, (SP)                           ; don't alter col 0
   59   01   D0      0077       1$:     MOVL    #1, R9                             ; dummy 2nd lower bound
        01   DD      007A               PUSHL   #1                                 ; dummy 2nd upper bound
```

```
                1A   11  007C                BRB     LOOP_2ND_SUBB           ; go loop
                    007E
                    007E        ;+
                    007E        ; There are 2 subscripts.  Put the upper bound for both subscripts on the
                    007E        ; stack and make sure that the lower bound for both subscripts will start
                    007E        ; at 1 (do not alter row or col 0)
                    007E        ;-
                    007E
                    007E        INIT_TWO_SUBSB:
              20 AA DD  007E                PUSHL   dsc$l_u1_2(R10)                 ; 1st upper bound
              1C AA DD  0081                PUSHL   dsc$l_l1_2(R10)                 ; 1st lower bound
                 03 14  0084                BGTR    1$                              ; not row 0 or neg, do cols
           6E    01 D0  0086                MOVL    #1, (SP)                        ; start with row 1
        59    28 AA D0  0089        1$:     MOVL    dsc$l_u2_2(R10), R9             ; 2nd upper bound
              24 AA DD  008D                PUSHL   dsc$l_l2_2(R10)                 ; 2nd lower bound
                 03 14  0090                BGTR    LOOP_1ST_SUBB           ; not col 0, go loop
           6E    01 D0  0092                MOVL    #1, (SP)                        ; start with col 1
                    0095
                    0095        ;+
                    0095        ; Loop through all the rows.  Row and column upper and lower bounds have been
                    0095        ; initialized on the stack.
                    0095        ;-
                    0095
                    0095        LOOP_1ST_SUBB:
           5B   6E   D0  0095                MOVL    lower_bnd2(SP), R11             ; R11 has 2nd lower bound
                    0098
                    0098        ;+
                    0098        ; Loop through all the elements (columns) of the current row.  Column lower
                    0098        ; bound is initialized in R11.  Column upper bound is on the stack.
                    0098        ; Distinguish array by data type so that the correct store routine can be
                    0098        ; called and the constant can be converted to the correct type.
                    0098        ;-
                    0098
                    0098        LOOP_2ND_SUBB:
                    0098
        50    24 AE   90  0098                MOVB    constant_cvt(SP), R0    ; put constant into R0
                    009C                                                        ;  R0 & R1 for double
                    009C        ;+
                    009C        ; When passed by value, hfloat takes 4 words, gfloat and double take 2 words,
                    009C        ; and all other data types take 1 longword.
                    009C        ;-
                    009C
                    009C                .IF     IDN     B, H            ; data type is hfloat
                    009C                MOVL    R10, R4                         ; pointer to array desc
                    009C                MOVL    lower_bnd1(SP), R5              ; current row
                    009C                MOVL    R11, R6                         ; current column
                    009C                .IFF
                    009C                .IF     IDN     B, G            ; data type is gfloat
                    009C                MOVL    R10, R2                         ; pointer to array desc
                    009C                MOVL    lower_bnd1(SP), R3              ; current row
                    009C                MOVL    R11, R4                         ; current column
                    009C                .IFF
                    009C                .IF     IDN     B, D            ; data type is double
                    009C                MOVL    R10, R2                         ; pointer to array desc
                    009C                MOVL    lower_bnd1(SP), R3              ; current row
                    009C                MOVL    R11, R4                         ; current column
                    009C                .IFF                                    ; all other data types
```

```
                51    5A    D0  009C                MOVL    R10, R1                         ; pointer to array desc
            52    04 AE    D0  009F                MOVL    lower_bnd1(SP), R2              ; current row
                53    5B    D0  00A3                MOVL    R11, R3                         ; current column
                                00A6                .ENDC
                                00A6                .ENDC
                                00A6                .ENDC
            14 AE    50    90  00A6                MOVB    R0, data(SP)            ; store value in value_desc
                                00AA                STORE   B                               ; store in array
                                00AA                .IF     IDN     B, H
                                00AA                CMPB    dsc$b_dtype(R4), #dsc$k_dtype_dsc
                                00AA                BNEQ    30009$
                                00AA                MOVL    4(R4), R0
                                00AA                MOVB    dsc$b_dtype(R0), dtype(SP)
                                00AA                MOVB    dsc$b_class(R0), class(SP)
                                00AA                MOVAL   data(SP), pointer (SP)
                                00AA                MOVW    #10, str_len(SP)
                                00AA                CMPB    dsc$b_dimct(R4), #1
                                00AA                BNEQ    30011$
                                00AA                PUSHL   R5
                                00AA                PUSHL   R4
                                00AA                PUSHAL  value_desc+8(SP)
                                00AA                CALLS   #3,G^BAS$STORE_BFA
                                00AA                BRW     30008$
                                00AA        30011$: PUSHL   R6
                                00AA                PUSHL   R5
                                00AA                PUSHL   R4
                                00AA                PUSHAL  value_desc+12(SP)
                                00AA                CALLS   #4,G^BAS$STORE_BFA
                                00AA                BRW     30008$
                                00AA        30009$: CMPB    dsc$b_class(R4), #dsc$k_class_bfa
                                00AA                BNEQ    30000$
                                00AA                JSB     G^BAS$STO_FA_9_R8
                                00AA                BRW     30008$
                                00AA        30000$: BBS     #5, 10(R4), 30001$
                                00AA                CMPB    dsc$b_dimct(R4), #1
                                00AA                BNEQ    30010$
                                00AA                MOVZWL  dsc$w_length(R4), R8
                                00AA                INDEX   R5, dsc$l_l1_1(R4), dsc$l_u1_1(R4), R8, #0, R7
                                00AA                ADDL    dsc$a_a0(R4), R7
                                00AA                MOVB    R0, (R7)
                                00AA                BRW     30008$
                                00AA        30010$: INDEX   R5, dsc$l_l1_2(R4), dsc$l_u1_2(R4), dsc$l_m2(R4), #0, R7
                                00AA                MOVZWL  dsc$w_length(R4), R8
                                00AA                INDEX   R6, dsc$l_l2_2(R4), dsc$l_u2_2(R4), R8, R7, R7
                                00AA                ADDL    dsc$a_a0(R4), R7
                                00AA                MOVB    R0, (R7)
                                00AA                BRW     30008$
                                00AA        30001$: CMPB    dsc$b_dimct(R4), #1
                                00AA                BNEQ    30012$
                                00AA                MOVZWL  dsc$w_length(R4), R8
                                00AA                INDEX   R6, dsc$l_l1_1(R4), dsc$l_u1_1(R4), R8, #0, R7
                                00AA                ADDL    dsc$a_a0(R4), R7
                                00AA                MOVB    R0, (R7)
                                00AA                BRW     30008$
                                00AA        30012$: INDEX   R6, dsc$l_l2_2(R4), dsc$l_u2_2(R4), dsc$l_m1(R4), #0, R7
                                00AA                MOVZWL  dsc$w_length(R4), R8
                                00AA                INDEX   R5, dsc$l_l1_2(R4), dsc$l_u1_2(R4), R8, R7, R7
```

```
00AA                    ADDL    dsc$a_a0(R7), R7
00AA                    MOVB    R0, (R7)
00AA                    .IFF
00AA                    .IF     IDN     B, G
00AA                    CMPB    dsc$b_dtype(R2), #dsc$k_dtype_dsc
00AA                    BNEQ    30013$
00AA                    MOVL    4(R2), R0
00AA                    MOVB    dsc$b_dtype(R0), dtype(SP)
00AA                    MOVB    dsc$b_class(R0), class(SP)
00AA                    MOVAL   data(SP), pointer (SP)
00AA                    MOVW    #10, str_len(SP)
00AA                    CMPB    dsc$b_dimct(R2), #1
00AA                    BNEQ    30015$
00AA                    PUSHL   R3
00AA                    PUSHL   R2
00AA                    PUSHAL  value_desc+8(SP)
00AA                    CALLS   #3,G^BAS$STORE_BFA
00AA                    BRW     30008$
00AA            30015$: PUSHL   R4
00AA                    PUSHL   R3
00AA                    PUSHL   R2
00AA                    PUSHAL  value_desc+12(SP)
00AA                    CALLS   #4,G^BAS$STORE_BFA
00AA                    BRW     30008$
00AA            30013$: CMPB    dsc$b_class(R2), #dsc$k_class_bfa
00AA                    BNEQ    30002$
00AA                    JSB     G^BAS$STO_FA_B_R8
00AA                    BRW     30008$
00AA            30002$: BBS     #5, 10(R2), 30003$
00AA                    CMPB    dsc$b_dimct(R2), #1
00AA                    BNEQ    30014$
00AA                    MOVZWL  dsc$w_length(R2), R6
00AA                    INDEX   R3, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
00AA                    ADDL    dsc$a_a0(R2), R5
00AA                    MOVB    R0, (R5)
00AA                    BRW     30008$
00AA            30014$: INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), dsc$l_m2(R2), #0, R5
00AA                    MOVZWL  dsc$w_length(R2), R6
00AA                    INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), R6, R5, R5
00AA                    ADDL    dsc$a_a0(R2), R5
00AA                    MOVB    R0, (R5)
00AA                    BRW     30008$
00AA            30003$: CMPB    dsc$b_dimct(R2), #1
00AA                    BNEQ    30016$
00AA                    MOVZWL  dsc$w_length(R2), R6
00AA                    INDEX   R4, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
00AA                    ADDL    dsc$a_a0(R2), R5
00AA                    MOVB    R0, (R5)
00AA                    BRW     30008$
00AA            30016$: INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), dsc$l_m1(R2), #0, R5
00AA                    MOVZWL  dsc$w_length(R2), R6
00AA                    INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), R6, R5, R5
00AA                    ADDL    dsc$a_a0(R2), R5
00AA                    MOVB    R0, (R5)
00AA                    .IFF
00AA                    .IF     IDN     B, D
00AA                    CMPB    dsc$b_dtype(R2), #dsc$k_dtype_dsc
```

```
                00AA                BNEQ    30017$
                00AA                MOVL    4(R2), R0
                00AA                MOVB    dsc$b_dtype(R0), dtype(SP)
                00AA                MOVB    dsc$b_class(R0), class(SP)
                00AA                MOVAL   data(SP), pointer (SP)
                00AA                MOVW    #10, str_len(SP)
                00AA                CMPB    dsc$b_dimct(R2), #1
                00AA                BNEQ    30019$
                00AA                PUSHL   R3
                00AA                PUSHL   R2
                00AA                PUSHAL  value_desc+8(SP)
                00AA                CALLS   #3,G^BAS$STORE_BFA
                00AA                BRW     30008$
                00AA        30019$: PUSHL   R4
                00AA                PUSHL   R3
                00AA                PUSHL   R2
                00AA                PUSHAL  value_desc+12(SP)
                00AA                CALLS   #4,G^BAS$STORE_BFA
                00AA                BRW     30008$
                00AA        30017$: CMPB    dsc$b_class(R2), #dsc$k_class_bfa
                00AA                BNEQ    30004$
                00AA                JSB     G^BAS$STO_FA_B_R8
                00AA                BRW     30008$
                00AA        30004$: BBS     #5, 10(R2), 30005$
                00AA                CMPB    dsc$b_dimct(R2), #1
                00AA                BNEQ    30018$
                00AA                MOVZWL  dsc$w_length(R2), R6
                00AA                INDEX   R3, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                00AA                ADDL    dsc$a_a0(R2), R5
                00AA                MOVB    R0, (R5)
                00AA                BRW     30008$
                00AA        30018$: INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), dsc$l_m2(R2), #0, R5
                00AA                MOVZWL  dsc$w_length(R2), R6
                00AA                INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), R6, R5, R5
                00AA                ADDL    dsc$a_a0(R2), R5
                00AA                MOVB    R0, (R5)
                00AA                BRW     30008$
                00AA        30005$: CMPB    dsc$b_dimct(R2), #1
                00AA                BNEQ    30020$
                00AA                MOVZWL  dsc$w_length(R2), R6
                00AA                INDEX   R4, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                00AA                ADDL    dsc$a_a0(R2), R5
                00AA                MOVB    R0, (R5)
                00AA                BRW     30008$
                00AA        30020$: INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), dsc$l_m1(R2), #0, R5
                00AA                MOVZWL  dsc$w_length(R2), R6
                00AA                INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), R6, R5, R5
                00AA                ADDL    dsc$a_a0(R2), R5
                00AA                MOVB    R0, (R5)
                00AA                .IFF
    18  02 A1   91   00AA          CMPB    dsc$b_dtype(R1), #dsc$k_dtype_dsc
       41   12   00AE               BNEQ    30021$
    50  04 A1   D0   00B0          MOVL    4(R1), R0
0E AE  02 A0    90   00B4          MOVB    dsc$b_dtype(R0), dtype(SP)
0F AE  03 A0    90   00B9          MOVB    dsc$b_class(R0), class(SP)
10 AE  14 AE    DE   00BE          MOVAL   data(SP), pointer (SP)
   0C AE  0A    B0   00C3          MOVW    #10, str_len(SP)
```

```
                    01    0B A1   91  00C7                CMPB    dsc$b_dimct(R1), #1
                                11  12  00CB                BNEQ    30023$
                                52  DD  00CD                PUSHL   R2
                                51  DD  00CF                PUSHL   R1
                              14 AE  DF  00D1                PUSHAL  value_desc+8(SP)
                00000000'GF    03  FB  00D4                CALLS   #3,G^BAS$STORE_BFA
                             009D  31  00DB                BRW     30008$
                                53  DD  00DE    30023$:     PUSHL   R3
                                52  DD  00E0                PUSHL   R2
                                51  DD  00E2                PUSHL   R1
                              18 AE  DF  00E4                PUSHAL  value_desc+12(SP)
                00000000'GF    04  FB  00E7                CALLS   #4,G^BAS$STORE_BFA
                             008A  31  00EE                BRW     30008$
                   BF 8F    03 A1  91  00F1    30021$:     CMPB    dsc$b_class(R1), #dsc$k_class_bfa
                                09  12  00F6                BNEQ    30006$
                  00000000'GF      16  00F8                JSB     G^BAS$STO_FA_B_R8
                             007A  31  00FE                BRW     30008$
                   3C 0A A1    05  E0  0101    30006$:     BBS     #5, 10(R1), 30007$
                      01    0B A1  91  0106                CMPB    dsc$b_dimct(R1), #1
                                16  12  010A                BNEQ    30022$
                           55   61  3C  010C                MOVZWL  dsc$w_length(R1), R5
00   55   1C A1    18 A1    52  0A  010F                INDEX   R2, dsc$l_l1_1(R1), dsc$l_u1_1(R1), R5, #0, R4
                                54      0117
                        54   10 A1  C0  0118                ADDL    dsc$a_a0(R1), R4
                           64   50  90  011C                MOVB    R0, (R4)
                             0059  31  011F                BRW     30008$
  18 A1    20 A1    1C A1    52  0A  0122    30022$:     INDEX   R2, dsc$l_l1_2(R1), dsc$l_u1_2(R1), dsc$l_m2(R1), #0, R4
                                54  00  012A
                           55   61  3C  012C                MOVZWL  dsc$w_length(R1), R5
54   55   28 A1    24 A1    53  0A  012F                INDEX   R3, dsc$l_l2_2(R1), dsc$l_u2_2(R1), R5, R4, R4
                                54      0137
                        54   10 A1  C0  0138                ADDL    dsc$a_a0(R1), R4
                           64   50  90  013C                MOVB    R0, (R4)
                             0039  31  013F                BRW     30008$
                      01    0B A1  91  0142    30007$:     CMPB    dsc$b_dimct(R1), #1
                                16  12  0146                BNEQ    30024$
                           55   61  3C  0148                MOVZWL  dsc$w_length(R1), R5
00   55   1C A1    18 A1    53  0A  014B                INDEX   R3, dsc$l_l1_1(R1), dsc$l_u1_1(R1), R5, #0, R4
                                54      0153
                        54   10 A1  C0  0154                ADDL    dsc$a_a0(R1), R4
                           64   50  90  0158                MOVB    R0, (R4)
                             001D  31  015B                BRW     30008$
  14 A1    28 A1    24 A1    53  0A  015E    30024$:     INDEX   R3, dsc$l_l2_2(R1), dsc$l_u2_2(R1), dsc$l_m1(R1), #0, R4
                                54  00  0166
                           55   61  3C  0168                MOVZWL  dsc$w_length(R1), R5
54   55   20 A1    1C A1    52  0A  016B                INDEX   R2, dsc$l_l1_2(R1), dsc$l_u1_2(R1), R5, R4, R4
                                54      0173
                        54   10 A1  C0  0174                ADDL    dsc$a_a0(R1), R4
                           64   50  90  0178                MOVB    R0, (R4)
                                        017B                .ENDC
                                        017B                .ENDC
                                        017B                .ENDC
                                        017B    30008$:
                                        017B
                                5B  D6  017B                INCL    R11                             ; get next column
                           59   5B  D1  017D                CMPL    R11, R9                         ; see if last column done
                                03  14  0180                BGTR    2$
```

```
                FF13   31  0182                BRW     LOOP_2ND_SUBB           ; no, continue inner loop
                           0185
                           0185        ;+
                           0185        ; Have completed entire row.  See if it was the last row.  If not,
                           0185        ; continue with next row.
                           0185        ;-
                           0185
            04 AE      D6  0185        2$:     INCL    lower_bnd1(SP)                  ; get next row
      08 AE 04 AE      D1  0188                CMPL    lower_bnd1(SP), upper_bnd1(SP)  ; see if last row done
                  03   14  018D                BGTR    3$
                FF03   31  018F                BRW     LOOP_1ST_SUBB           ; no, continue outer loop
                           0192
                      04   0192        3$:     RET                             ; yes, finished
                           0193
```

```
                     0193   373 WORD:   $BAS$MAT_INIT W                        ; expand to word operations
                     0193
                     0193       ;+
                     0193       ;       REGISTER USAGE
                     0193       ;       R0 - R8 destroyed by store routines
                     0193       ;       R9      upper bound for 2nd subscript
                     0193       ;       R10     pointer to array descriptor
                     0193       ;       R11     current value of 2nd subscript
                     0193       ;-
                     0193
                     0193       ;+
                     0193       ; Set up limits for looping through all elements
                     0193       ;-
                     0193
                     0193               .IF     IDN     W, L
                     0193               .IFT                                    ; data type is long
                     0193               MOVL    constant(AP), -(SP)             ; move constant
                     0193               .IFF                                    ; data type is not long
                7E   08 AC   F7 0193           CVTLW   constant(AP), -(SP)     ; make constant same datatype
                     0197                                                      ;  as array, save on stack
                     0197               .ENDC
                     0197               .IF     IDN     W, D                    ; if array is double
                     0197               MOVL    SF$L_SAVE_FP(FP), R0            ; pass FP to get scale
                     0197               JSB     G^BAS$$SCALE_R1                 ; get scale in R0 & R1
                     0197                                                       ; call a BLISS routine because
                     0197                                                       ; the frame offsets are only
                     0197                                                       ; defined for BLISS
                     0197               MULD2   R0, (SP)                        ; scale
                     0197               .ENDC
                     0197
                     0197       ;+
                     0197       ; Allocate data and value_desc on the stack.  This applies to both
                     0197       ; one and two dimensions.
                     0197       ;-
                     0197
                7E   7C 0197               CLRQ    -(SP)                       ; space for data
                7E   7C 0199               CLRQ    -(SP)                       ;  may be hfloat
                7E   7C 019B               CLRQ    -(SP)                       ; space for value_desc
                     019D
             01 0B AA   91 019D            CMPB    DSC$B_DIMCT(R10), #1        ; determine # of subscripts
                05   13 01A1               BEQLU   INIT_ONE_SUBW       ; 1 sub, go init
                15   1A 01A3               BGTRU   INIT_TWO_SUBSW      ; >=2 subs, go init
              FEA2   31 01A5               BRW     ERR_ARGDONMAT               ; 0 subs, error
                     01A8
                     01A8       ;+
                     01A8       ; There is only 1 subscript.  Make both upper and lower bound for 2nd
                     01A8       ; subscript a 1.  The second subscript will be passed to and ignored by the
                     01A8       ; store routine.
                     01A8       ;-
                     01A8
                     01A8       INIT_ONE_SUBW:
             1C AA   DD 01A8               PUSHL   dsc$l_u1_1(R10)             ; 1st upper bound
             18 AA   DD 01AB               PUSHL   dsc$l_l1_1(R10)             ; 1st lower bound
                03   14 01AE               BGTR    1$                          ; not 0 or neg, do 2nd sub
          6E    01   D0 01B0               MOVL    #1, (SP)                    ; don't alter col 0
          59    01   D0 01B3       1$:     MOVL    #1, R9                      ; dummy 2nd lower bound
                01   DD 01B6               PUSHL   #1                          ; dummy 2nd upper bound
```

```
                 1A   11  01B8                BRB     LOOP_2ND_SUBW           ; go loop
                         01BA
                         01BA        ;+
                         01BA        ; There are 2 subscripts.  Put the upper bound for both subscripts on the
                         01BA        ; stack and make sure that the lower bound for both subscripts will start
                         01BA        ; at 1 (do not alter row or col 0)
                         01BA        ;-
                         01BA
                         01BA        INIT_TWO_SUBSW:
                   20 AA DD  01BA                PUSHL   dsc$l_u1_2(R10)                 ; 1st upper bound
                   1C AA DD  01BD                PUSHL   dsc$l_l1_2(R10)                 ; 1st lower bound
                      03 14  01C0                BGTR    1$                              ; not row 0 or neg, do cols
                6E    01 D0  01C2                MOVL    #1, (SP)                        ; start with row 1
             59   28 AA DD  01C5        1$:     MOVL    dsc$l_u2_2(R10), R9             ; 2nd upper bound
                   24 AA DD  01C9                PUSHL   dsc$l_l2_2(R10)                 ; 2nd lower bound
                      03 14  01CC                BGTR    LOOP_TST_SUBW           ; not col 0, go loop
                6E    01 D0  01CE                MOVL    #1, (SP)                        ; start with col 1
                         01D1
                         01D1        ;+
                         01D1        ; Loop through all the rows.  Row and column upper and lower bounds have been
                         01D1        ; initialized on the stack.
                         01D1        ;-
                         01D1
                         01D1        LOOP_1ST_SUBW:
                5B    6E  D0  01D1                MOVL    lower_bnd2(SP), R11             ; R11 has 2nd lower bound
                         01D4
                         01D4        ;+
                         01D4        ; Loop through all the elements (columns) of the current row.  Column lower
                         01D4        ; bound is initialized in R11.  Column upper bound is on the stack.
                         01D4        ; Distinguish array by data type so that the correct store routine can be
                         01D4        ; called and the constant can be converted to the correct type.
                         01D4        ;-
                         01D4
                         01D4        LOOP_2ND_SUBW:
                         01D4
             50    24 AE  B0  01D4                MOVW    constant_cvt(SP), R0    ; put constant into R0
                         01D8                                                        ;  R0 & R1 for double
                         01D8        ;+
                         01D8        ; When passed by value, hfloat takes 4 words, gfloat and double take 2 words,
                         01D8        ; and all other data types take 1 longword.
                         01D8        ;-
                         01D8
                         01D8                .IF     IDN     W, H            ; data type is hfloat
                         01D8                MOVL    R10, R4                         ; pointer to array desc
                         01D8                MOVL    lower_bnd1(SP), R5              ; current row
                         01D8                MOVL    R11, R6                         ; current column
                         01D8                .IFF
                         01D8                .IF     IDN     W, G            ; data type is gfloat
                         01D8                MOVL    R10, R2                         ; pointer to array desc
                         01D8                MOVL    lower_bnd1(SP), R3              ; current row
                         01D8                MOVL    R11, R4                         ; current column
                         01D8                .IFF
                         01D8                .IF     IDN     W, D            ; data type is double
                         01D8                MOVL    R10, R2                         ; pointer to array desc
                         01D8                MOVL    lower_bnd1(SP), R3              ; current row
                         01D8                MOVL    R11, R4                         ; current column
                         01D8                .IFF                                    ; all other data types
```

```
                   51   5A    D0  01D8              MOVL    R10, R1                         ; pointer to array desc
               52   04 AE    D0  01DB              MOVL    lower_bnd1(SP), R2              ; current row
                   53   5B    D0  01DF              MOVL    R11, R3                         ; current column
                                  01E2              .ENDC
                                  01E2              .ENDC
                                  01E2              .ENDC
               14 AE    50    B0  01E2              MOVW    R0, data(SP)            ; store value in value_desc
                                  01E6              STORE   W                               ; store in array
                                  01E6              .IF     IDN     W, H
                                  01E6              CMPB    dsc$b_dtype(R4), #dsc$k_dtype_dsc
                                  01E6              BNEQ    30034$
                                  01E6              MOVL    4(R4), R0
                                  01E6              MOVB    dsc$b_dtype(R0), dtype(SP)
                                  01E6              MOVB    dsc$b_class(R0), class(SP)
                                  01E6              MOVAL   data(SP), pointer (SP)
                                  01E6              MOVW    #10, str_len(SP)
                                  01E6              CMPB    dsc$b_dimct(R4), #1
                                  01E6              BNEQ    30036$
                                  01E6              PUSHL   R5
                                  01E6              PUSHL   R4
                                  01E6              PUSHAL  value_desc+8(SP)
                                  01E6              CALLS   #3,G^BAS$STORE_BFA
                                  01E6              BRW     30033$
                                  01E6      30036$: PUSHL   R6
                                  01E6              PUSHL   R5
                                  01E6              PUSHL   R4
                                  01E6              PUSHAL  value_desc+12(SP)
                                  01E6              CALLS   #4,G^BAS$STORE_BFA
                                  01E6              BRW     30033$
                                  01E6      30034$: CMPB    dsc$b_class(R4), #dsc$k_class_bfa
                                  01E6              BNEQ    30025$
                                  01E6              JSB     G^BAS$STO_FA_W_R8
                                  01E6              BRW     30033$
                                  01E6      30025$: BBS     #5, 10(R4), 30026$
                                  01E6              CMPB    dsc$b_dimct(R4), #1
                                  01E6              BNEQ    30035$
                                  01E6              MOVZWL  dsc$w_length(R4), R8
                                  01E6              INDEX   R5, dsc$l_l1_1(R4), dsc$l_u1_1(R4), R8, #0, R7
                                  01E6              ADDL    dsc$a_a0(R4), R7
                                  01E6              MOVW    R0, (R7)
                                  01E6              BRW     30033$
                                  01E6      30035$: INDEX   R5, dsc$l_l1_2(R4), dsc$l_u1_2(R4), dsc$l_m2(R4), #0, R7
                                  01E6              MOVZWL  dsc$w_length(R4), R8
                                  01E6              INDEX   R6, dsc$l_l2_2(R4), dsc$l_u2_2(R4), R8, R7, R7
                                  01E6              ADDL    dsc$a_a0(R4), R7
                                  01E6              MOVW    R0, (R7)
                                  01E6              BRW     30033$
                                  01E6      30026$: CMPB    dsc$b_dimct(R4), #1
                                  01E6              BNEQ    30037$
                                  01E6              MOVZWL  dsc$w_length(R4), R8
                                  01E6              INDEX   R6, dsc$l_l1_1(R4), dsc$l_u1_1(R4), R8, #0, R7
                                  01E6              ADDL    dsc$a_a0(R4), R7
                                  01E6              MOVW    R0, (R7)
                                  01E6              BRW     30033$
                                  01E6      30037$: INDEX   R6, dsc$l_l2_2(R4), dsc$l_u2_2(R4), dsc$l_m1(R4), #0, R7
                                  01E6              MOVZWL  dsc$w_length(R4), R8
                                  01E6              INDEX   R5, dsc$l_l1_2(R4), dsc$l_u1_2(R4), R8, R7, R7
```

```
01E6                    ADDL    dsc$a_a0(R4), R7
01E6                    MOVW    R0, (R7)
01E6                    .IFF
01E6                    .IF     IDN     W, G
01E6                    CMPB    dsc$b_dtype(R2), #dsc$k_dtype_dsc
01E6                    BNEQ    30038$
01E6                    MOVL    4(R2), R0
01E6                    MOVB    dsc$b_dtype(R0), dtype(SP)
01E6                    MOVB    dsc$b_class(R0), class(SP)
01E6                    MOVAL   data(SP), pointer (SP)
01E6                    MOVW    #10, str_len(SP)
01E6                    CMPB    dsc$b_dimct(R2), #1
01E6                    BNEQ    30040$
01E6                    PUSHL   R3
01E6                    PUSHL   R2
01E6                    PUSHAL  value_desc+8(SP)
01E6                    CALLS   #3,G^BAS$STORE_BFA
01E6                    BRW     30033$
01E6            30040$: PUSHL   R4
01E6                    PUSHL   R3
01E6                    PUSHL   R2
01E6                    PUSHAL  value_desc+12(SP)
01E6                    CALLS   #4,G^BAS$STORE_BFA
01E6                    BRW     30033$
01E6            30038$: CMPB    dsc$b_class(R2), #dsc$k_class_bfa
01E6                    BNEQ    30027$
01E6                    JSB     G^BAS$STO_FA_W_R8
01E6                    BRW     30033$
01E6            30027$: BBS     #5, 10(R2), 30028$
01E6                    CMPB    dsc$b_dimct(R2), #1
01E6                    BNEQ    30039$
01E6                    MOVZWL  dsc$w_length(R2), R6
01E6                    INDEX   R3, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
01E6                    ADDL    dsc$a_a0(R2), R5
01E6                    MOVW    R0, (R5)
01E6                    BRW     30033$
01E6            30039$: INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), dsc$l_m2(R2), #0, R5
01E6                    MOVZWL  dsc$w_length(R2), R6
01E6                    INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), R6, R5, R5
01E6                    ADDL    dsc$a_a0(R2), R5
01E6                    MOVW    R0, (R5)
01E6                    BRW     30033$
01E6            30028$: CMPB    dsc$b_dimct(R2), #1
01E6                    BNEQ    30041$
01E6                    MOVZWL  dsc$w_length(R2), R6
01E6                    INDEX   R4, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
01E6                    ADDL    dsc$a_a0(R2), R5
01E6                    MOVW    R0, (R5)
01E6                    BRW     30033$
01E6            30041$: INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), dsc$l_m1(R2), #0, R5
01E6                    MOVZWL  dsc$w_length(R2), R6
01E6                    INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), R6, R5, R5
01E6                    ADDL    dsc$a_a0(R2), R5
01E6                    MOVW    R0, (R5)
01E6                    .IFF
01E6                    .IF     IDN     W, D
01E6                    CMPB    dsc$b_dtype(R2), #dsc$k_dtype_dsc
```

```
                01E6                    BNEQ    30042$
                01E6                    MOVL    4(R2), R0
                01E6                    MOVB    dsc$b_dtype(R0), dtype(SP)
                01E6                    MOVB    dsc$b_class(R0), class(SP)
                01E6                    MOVAL   data(SP), pointer (SP)
                01E6                    MOVW    #10, str_len(SP)
                01E6                    CMPB    dsc$b_dimct(R2), #1
                01E6                    BNEQ    30044$
                01E6                    PUSHL   R3
                01E6                    PUSHL   R2
                01E6                    PUSHAL  value_desc+8(SP)
                01E6                    CALLS   #3,G^BAS$STORE_BFA
                01E6                    BRW     30033$
                01E6            30044$: PUSHL   R4
                01E6                    PUSHL   R3
                01E6                    PUSHL   R2
                01E6                    PUSHAL  value_desc+12(SP)
                01E6                    CALLS   #4,G^BAS$STORE_BFA
                01E6                    BRW     30033$
                01E6            30042$: CMPB    dsc$b_class(R2), #dsc$k_class_bfa
                01E6                    BNEQ    30029$
                01E6                    JSB     G^BAS$STO_FA_W_R8
                01E6                    BRW     30033$
                01E6            30029$: BBS     #5, 10(R2), 30030$
                01E6                    CMPB    dsc$b_dimct(R2), #1
                01E6                    BNEQ    30043$
                01E6                    MOVZWL  dsc$w_length(R2), R6
                01E6                    INDEX   R3, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                01E6                    ADDL    dsc$a_a0(R2), R5
                01E6                    MOVW    R0, (R5)
                01E6                    BRW     30033$
                01E6            30043$: INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2)  dsc$l_m2(R2), #0, R5
                01E6                    MOVZWL  dsc$w_length(R2), R6
                01E6                    INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), R6, R5, R5
                01E6                    ADDL    dsc$a_a0(R2), R5
                01E6                    MOVW    R0, (R5)
                01E6                    BRW     30033$
                01E6            30030$: CMPB    dsc$b_dimct(R2), #1
                01E6                    BNEQ    30045$
                01E6                    MOVZWL  dsc$w_length(R2), R6
                01E6                    INDEX   R4, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                01E6                    ADDL    dsc$a_a0(R2), R5
                01E6                    MOVW    R0, (R5)
                01E6                    BRW     30033$
                01E6            30045$: INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), dsc$l_m1(R2), #0, R5
                01E6                    MOVZWL  dsc$w_length(R2), R6
                01E6                    INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), R6, R5, R5
                01E6                    ADDL    dsc$a_a0(R2), R5
                01E6                    MOVW    R0, (R5)
                01E6                    .IFF
    18  02 A1  91  01E6                CMPB    dsc$b_dtype(R1), #dsc$k_dtype_dsc
           41  12  01EA                BNEQ    30046$
    50  04 A1  D0  01EC                MOVL    4(R1), R0
0E AE  02 A0  90  01F0                MOVB    dsc$b_dtype(R0), dtype(SP)
0F AE  03 A0  90  01F5                MOVB    dsc$b_class(R0), class(SP)
10 AE  14 AE  DE  01FA                MOVAL   data(SP), pointer (SP)
   0C AE   0A  B0  01FF                MOVW    #10, str_len(SP)
```

```
                    01    0B A1    91  0203                CMPB    dsc$b_dimct(R1), #1
                                11    12  0207                BNEQ    30048$
                                52    DD  0209                PUSHL   R2
                                51    DD  020B                PUSHL   R1
                             14 AE    DF  020D                PUSHAL  value_desc+8(SP)
                00000000'GF    03    FB  0210                CALLS   #3,G^BAS$STORE_BFA
                              009D    31  0217                BRW     30033$
                                53    DD  021A    30048$:     PUSHL   R3
                                52    DD  021C                PUSHL   R2
                                51    DD  021E                PUSHL   R1
                             18 AE    DF  0220                PUSHAL  value_desc+12(SP)
                00000000'GF    04    FB  0223                CALLS   #4,G^BAS$STORE_BFA
                              008A    31  022A                BRW     30033$
                   BF 8F    03 A1    91  022D    30046$:     CMPB    dsc$b_class(R1), #dsc$k_class_bfa
                                09    12  0232                BNEQ    30031$
                     00000000'GF    16  0234                JSB     G^BAS$STO_FA_W_R8
                              007A    31  023A                BRW     30033$
                   3C 0A A1    05    E0  023D    30031$:     BBS     #5, 10(R1), 30032$
                      01    0B A1    91  0242                CMPB    dsc$b_dimct(R1), #1
                                16    12  0246                BNEQ    30047$
                             55    61    3C  0248                MOVZWL  dsc$w_length(R1), R5
00   55   1C A1    18 A1    52    0A  024B                INDEX   R2, dsc$l_l1_1(R1), dsc$l_u1_1(R1), R5, #0, R4
                                54        0253
                      54    10 A1    C0  0254                ADDL    dsc$a_a0(R1), R4
                         64    50    B0  0258                MOVW    R0, (R4)
                              0059    31  025B                BRW     30033$
  18 A1   20 A1    1C A1    52    0A  025E    30047$:     INDEX   R2, dsc$l_l1_2(R1), dsc$l_u1_2(R1), dsc$l_m2(R1), #0, R4
                             54    00        0266
                             55    61    3C  0268                MOVZWL  dsc$w_length(R1), R5
54   55   28 A1    24 A1    53    0A  026B                INDEX   R3, dsc$l_l2_2(R1), dsc$l_u2_2(R1), R5, R4, R4
                                54        0273
                      54    10 A1    C0  0274                ADDL    dsc$a_a0(R1), R4
                         64    50    B0  0278                MOVW    R0, (R4)
                              0039    31  027B                BRW     30033$
                      01    0B A1    91  027E    30032$:     CMPB    dsc$b_dimct(R1), #1
                                16    12  0282                BNEQ    30049$
                             55    61    3C  0284                MOVZWL  dsc$w_length(R1), R5
00   55   1C A1    18 A1    53    0A  0287                INDEX   R3, dsc$l_l1_1(R1), dsc$l_u1_1(R1), R5, #0, R4
                                54        028F
                      54    10 A1    C0  0290                ADDL    dsc$a_a0(R1), R4
                         64    50    B0  0294                MOVW    R0, (R4)
                              001D    31  0297                BRW     30033$
  14 A1   28 A1    24 A1    53    0A  029A    30049$:     INDEX   R3, dsc$l_l2_2(R1), dsc$l_u2_2(R1), dsc$l_m1(R1), #0, R4
                             54    00        02A2
                             55    61    3C  02A4                MOVZWL  dsc$w_length(R1), R5
54   55   20 A1    1C A1    52    0A  02A7                INDEX   R2, dsc$l_l1_2(R1), dsc$l_u1_2(R1), R5, R4, R4
                                54        02AF
                      54    10 A1    C0  02B0                ADDL    dsc$a_a0(R1), R4
                         64    50    B0  02B4                MOVW    R0, (R4)
                                        02B7                .ENDC
                                        02B7                .ENDC
                                        02B7                .ENDC
                                        02B7    30033$:
                                        02B7
                             5B    D6  02B7                INCL    R11                             ; get next column
                          59    5B    D1  02B9                CMPL    R11, R9                         ; see if last column done
                                03    14  02BC                BGTR    2$
```

```
                FF13    31  02BE                BRW     LOOP_2ND_SUBW           ; no, continue inner loop
                            02C1
                            02C1        ;+
                            02C1        ; Have completed entire row.  See if it was the last row.  If not,
                            02C1        ; continue with next row.
                            02C1        ;-
                            02C1
            04 AE   D6      02C1        2$:     INCL    lower_bnd1(SP)                  ; get next row
    08 AE   04 AE   D1      02C4                CMPL    lower_bnd1(SP), upper_bnd1(SP)  ; see if last row done
                03  14      02C9                BGTR    3$
                FF03    31  02CB                BRW     LOOP_1ST_SUBW           ; no, continue outer loop
                            02CE
                    04      02CE        3$:     RET                             ; yes, finished
                            02CF
```

```
                     02CF    375 LONG:   $BAS$MAT_INIT L                         ; expand to long operations
                     02CF
                     02CF        ;+
                     02CF        ;       REGISTER USAGE
                     02CF        ;       R0 - R8 destroyed by store routines
                     02CF        ;       R9      upper bound for 2nd subscript
                     02CF        ;       R10     pointer to array descriptor
                     02CF        ;       R11     current value of 2nd subscript
                     02CF        ;-
                     02CF
                     02CF        ;+
                     02CF        ; Set up limits for looping through all elements
                     02CF        ;-
                     02CF
                     02CF                .IF     IDN     L, L
                     02CF                .IFT                                    ; data type is long
                7E   08 AC   D0  02CF    MOVL    constant(AP), -(SP)             ; move constant
                     02D3                .IFF                                    ; data type is not long
                     02D3                CVTLL   constant(AP), -(SP)     ; make constant same datatype
                     02D3                                                        ;  as array, save on stack
                     02D3                .ENDC
                     02D3                .IF     IDN     L, D            ; if array is double
                     02D3                MOVL    SF$L_SAVE_FP(FP), R0            ; pass FP to get scale
                     02D3                JSB     G^BAS$$SCALE_R1                 ; get scale in R0 & R1
                     02D3                                                        ; call a BLISS routine because
                     02D3                                                        ; the frame offsets are only
                     02D3                                                        ; defined for BLISS
                     02D3                MULD2   R0, (SP)                        ; scale
                     02D3                .ENDC
                     02D3
                     02D3        ;+
                     02D3        ; Allocate data and value_desc on the stack.  This applies to both
                     02D3        ; one and two dimensions.
                     02D3        ;-
                     02D3
                7E   7C  02D3                CLRQ    -(SP)                   ; space for data
                7E   7C  02D5                CLRQ    -(SP)                   ;  may be hfloat
                7E   7C  02D7                CLRQ    -(SP)                   ; space for value_desc
                     02D9
           01   0B AA   91  02D9                CMPB    DSC$B_DIMCT(R10), #1    ; determine # of subscripts
                05   13  02DD                BEQLU   INIT_ONE_SUBL           ; 1 sub, go init
                15   1A  02DF                BGTRU   INIT_TWO_SUBSL          ; >=2 subs, go init
              FD66   31  02E1                BRW     ERR_ARGDONMAT                   ; 0 subs, error
                     02E4
                     02E4        ;+
                     02E4        ; There is only 1 subscript.  Make both upper and lower bound for 2nd
                     02E4        ; subscript a 1.  The second subscript will be passed to and ignored by the
                     02E4        ; store routine.
                     02E4        ;-
                     02E4
                     02E4        INIT_ONE_SUBL:
             1C AA   DD  02E4                PUSHL   dsc$l_u1_1(R10)                 ; 1st upper bound
             18 AA   DD  02E7                PUSHL   dsc$l_l1_1(R10)                 ; 1st lower bound
                03   14  02EA                BGTR    1$                              ; not 0 or neg, do 2nd sub
          6E    01   D0  02EC                MOVL    #1, (SP)                        ; don't alter col 0
          59    01   D0  02EF        1$:     MOVL    #1, R9                          ; dummy 2nd lower bound
                01   DD  02F2                PUSHL   #1                              ; dummy 2nd upper bound
```

```
                1A   11  02F4                BRB     LOOP_2ND_SUBL           ; go loop
                     02F6
                     02F6        ;+
                     02F6        ; There are 2 subscripts.  Put the upper bound for both subscripts on the
                     02F6        ; stack and make sure that the lower bound for both subscripts will start
                     02F6        ; at 1 (do not alter row or col 0)
                     02F6        ;-
                     02F6
                     02F6        INIT_TWO_SUBSL:
              20 AA  DD  02F6            PUSHL   dsc$l_u1_2(R10)                 ; 1st upper bound
              1C AA  DD  02F9            PUSHL   dsc$l_l1_2(R10)                 ; 1st lower bound
                 03  14  02FC            BGTR    1$                              ; not row 0 or neg, do cols
           6E    01  D0  02FE            MOVL    #1, (SP)                        ; start with row 1
        59    28 AA  D0  0301  1$:       MOVL    dsc$l_u2_2(R10), R9             ; 2nd upper bound
              24 AA  DD  0305            PUSHL   dsc$l_l2_2(R10)                 ; 2rd lower bound
                 03  14  0308            BGTR    LOOP_TST_SUBL           ; not col 0, go loop
           6E    01  D0  030A            MOVL    #1, (SP)                        ; start with col 1
                     030D
                     030D        ;+
                     030D        ; Loop through all the rows.  Row and column upper and lower bounds have been
                     030D        ; initialized on the stack.
                     030D        ;-
                     030D
                     030D        LOOP_1ST_SUBL:
           5B    6E  D0  030D            MOVL    lower_bnd2(SP), R11             ; R11 has 2nd lower bound
                     0310
                     0310        ;+
                     0310        ; Loop through all the elements (columns) of the current row.  Column lower
                     0310        ; bound is initialized in R11.  Column upper bound is on the stack.
                     0310        ; Distinguish array by data type so that the correct store routine can be
                     0310        ; called and the constant can be converted to the correct type.
                     0310        ;-
                     0310
                     0310        LOOP_2ND_SUBL:
                     0310
        50    24 AE  D0  0310            MOVL    constant_cvt(SP), R0    ; put constant into R0
                     0314                                                    ;  R0 & R1 for double
                     0314        ;+
                     0314        ; When passed by value, hfloat takes 4 words, gfloat and double take 2 words,
                     0314        ; and all other data types take 1 longword.
                     0314        ;-
                     0314
                     0314            .IF     IDN     L, H            ; data type is hfloat
                     0314            MOVL    R10, R4                         ; pointer to array desc
                     0314            MOVL    lower_bnd1(SP), R5              ; current row
                     0314            MOVL    R11, R6                         ; current column
                     0314            .IFF
                     0314            .IF     IDN     L, G            ; data type is gfloat
                     0314            MOVL    R10, R2                         ; pointer to array desc
                     0314            MOVL    lower_bnd1(SP), R3              ; current row
                     0314            MOVL    R11, R4                         ; current column
                     0314            .IFF
                     0314            .IF     IDN     L, D            ; data type is double
                     0314            MOVL    R10, R2                         ; pointer to array desc
                     0314            MOVL    lower_bnd1(SP), R3              ; current row
                     0314            MOVL    R11, R4                         ; current column
                     0314            .IFF                                    ; all other data types
```

```
                51    5A    D0  0314                MOVL    R10, R1                         ; pointer to array desc
            52    04 AE    D0  0317                MOVL    lower_bnd1(SP), R2              ; current row
                53    5B    D0  031B                MOVL    R11, R3                         ; current column
                                031E                .ENDC
                                031E                .ENDC
                                031E                .ENDC
            14 AE    50    D0  031E                MOVL    R0, data(SP)            ; store value in value_desc
                                0322                STORE   L                               ; store in array
                                0322                .IF     IDN     L, H
                                0322                CMPB    dsc$b_dtype(R4), #dsc$k_dtype_dsc
                                0322                BNEQ    30059$
                                0322                MOVL    4(R4), R0
                                0322                MOVB    dsc$b_dtype(R0), dtype(SP)
                                0322                MOVB    dsc$b_class(R0), class(SP)
                                0322                MOVAL   data(SP), pointer (SP)
                                0322                MOVW    #10, str_len(SP)
                                0322                CMPB    dsc$b_dimct(R4), #1
                                0322                BNEQ    30061$
                                0322                PUSHL   R5
                                0322                PUSHL   R4
                                0322                PUSHAL  value_desc+8(SP)
                                0322                CALLS   #3,G^BAS$STORE_BFA
                                0322                BRW     30058$
                                0322        30061$: PUSHL   R6
                                0322                PUSHL   R5
                                0322                PUSHL   R4
                                0322                PUSHAL  value_desc+12(SP)
                                0322                CALLS   #4,G^BAS$STORE_BFA
                                0322                BRW     30058$
                                0322        30059$: CMPB    dsc$b_class(R4), #dsc$k_class_bfa
                                0322                BNEQ    30050$
                                0322                JSB     G^BAS$STO_FA_L_R8
                                0322                BRW     30058$
                                0322        30050$: BBS     #5, 10(R4), 30051$
                                0322                CMPB    dsc$b_dimct(R4), #1
                                0322                BNEQ    30060$
                                0322                MOVZWL  dsc$w_length(R4), R8
                                0322                INDEX   R5, dsc$l_l1_1(R4), dsc$l_u1_1(R4), R8, #0, R7
                                0322                ADDL    dsc$a_a0(R4), R7
                                0322                MOVL    R0, (R7)
                                0322                BRW     30058$
                                0322        30060$: INDEX   R5, dsc$l_l1_2(R4), dsc$l_u1_2(R4), dsc$l_m2(R4), #0, R7
                                0322                MOVZWL  dsc$w_length(R4), R8
                                0322                INDEX   R6, dsc$l_l2_2(R4), dsc$l_u2_2(R4), R8, R7, R7
                                0322                ADDL    dsc$a_a0(R4), R7
                                0322                MOVL    R0, (R7)
                                0322                BRW     30058$
                                0322        30051$: CMPB    dsc$b_dimct(R4), #1
                                0322                BNEQ    30062$
                                0322                MOVZWL  dsc$w_length(R4), R8
                                0322                INDEX   R6, dsc$l_l1_1(R4), dsc$l_u1_1(R4), R8, #0, R7
                                0322                ADDL    dsc$a_a0(R4), R7
                                0322                MOVL    R0, (R7)
                                0322                BRW     30058$
                                0322        30062$: INDEX   R6, dsc$l_l2_2(R4), dsc$l_u2_2(R4), dsc$l_m1(R4), #0, R7
                                0322                MOVZWL  dsc$w_length(R4), R8
                                0322                INDEX   R5, dsc$l_l1_2(R4), dsc$l_u1_2(R4), R8, R7, R7
```

```
0322                    ADDL    dsc$a_a0(R4), R7
0322                    MOVL    R0, (R7)
0322                    .IFF
0322                    .IF     IDN     L, G
0322                    CMPB    dsc$b_dtype(R2), #dsc$k_dtype_dsc
0322                    BNEQ    30063$
0322                    MOVL    4(R2), R0
0322                    MOVB    dsc$b_dtype(R0), dtype(SP)
0322                    MOVB    dsc$b_class(R0), class(SP)
0322                    MOVAL   data(SP), pointer (SP)
0322                    MOVW    #10, str_len(SP)
0322                    CMPB    dsc$b_dimct(R2), #1
0322                    BNEQ    30065$
0322                    PUSHL   R3
0322                    PUSHL   R2
0322                    PUSHAL  value_desc+8(SP)
0322                    CALLS   #3,G^BAS$STORE_BFA
0322                    BRW     30058$
0322            30065$: PUSHL   R4
0322                    PUSHL   R3
0322                    PUSHL   R2
0322                    PUSHAL  value_desc+12(SP)
0322                    CALLS   #4,G^BAS$STORE_BFA
0322                    BRW     30058$
0322            30063$: CMPB    dsc$b_class(R2), #dsc$k_class_bfa
0322                    BNEQ    30052$
0322                    JSB     G^BAS$STO_FA_L_R8
0322                    BRW     30058$
0322            30052$: BBS     #5, 10(R2), 30053$
0322                    CMPB    dsc$b_dimct(R2), #1
0322                    BNEQ    30064$
0322                    MOVZWL  dsc$w_length(R2), R6
0322                    INDEX   R3, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
0322                    ADDL    dsc$a_a0(R2), R5
0322                    MOVL    R0, (R5)
0322                    BRW     30058$
0322            30064$: INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), dsc$l_m2(R2), #0, R5
0322                    MOVZWL  dsc$w_length(R2), R6
0322                    INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), R6, R5, R5
0322                    ADDL    dsc$a_a0(R2), R5
0322                    MOVL    R0, (R5)
0322                    BRW     30058$
0322            30053$: CMPB    dsc$b_dimct(R2), #1
0322                    BNEQ    30066$
0322                    MOVZWL  dsc$w_length(R2), R6
0322                    INDEX   R4, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
0322                    ADDL    dsc$a_a0(R2), R5
0322                    MOVL    R0, (R5)
0322                    BRW     30058$
0322            30066$: INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), dsc$l_m1(R2), #0, R5
0322                    MOVZWL  dsc$w_length(R2), R6
0322                    INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), R6, R5, R5
0322                    ADDL    dsc$a_a0(R2), R5
0322                    MOVL    R0, (R5)
0322                    .IFF
0322                    .IF     IDN     L, D
0322                    CMPB    dsc$b_dtype(R2), #dsc$k_dtype_dsc
```

```
                                0322                BNEQ    30067$
                                0322                MOVL    4(R2), R0
                                0322                MOVB    dsc$b_dtype(R0), dtype(SP)
                                0322                MOVB    dsc$b_class(R0), class(SP)
                                0322                MOVAL   data(SP), pointer (SP)
                                0322                MOVW    #10, str_len(SP)
                                0322                CMPB    dsc$b_dimct(R2), #1
                                0322                BNEQ    30069$
                                0322                PUSHL   R3
                                0322                PUSHL   R2
                                0322                PUSHAL  value_desc+8(SP)
                                0322                CALLS   #3,G^BAS$STORE_BFA
                                0322                BRW     30058$
                                0322        30069$: PUSHL   R4
                                0322                PUSHL   R3
                                0322                PUSHL   R2
                                0322                PUSHAL  value_desc+12(SP)
                                0322                CALLS   #4,G^BAS$STORE_BFA
                                0322                BRW     30058$
                                0322        30067$: CMPB    dsc$b_class(R2), #dsc$k_class_bfa
                                0322                BNEQ    30054$
                                0322                JSB     G^BAS$STO_FA_L_R8
                                0322                BRW     30058$
                                0322        30054$: BBS     #5, 10(R2), 30055$
                                0322                CMPB    dsc$b_dimct(R2), #1
                                0322                BNEQ    30068$
                                0322                MOVZWL  dsc$w_length(R2), R6
                                0322                INDEX   R3, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                                0322                ADDL    dsc$a_a0(R2), R5
                                0322                MOVL    R0, (R5)
                                0322                BRW     30058$
                                0322        30068$: INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), dsc$l_m2(R2), #0, R5
                                0322                MOVZWL  dsc$w_length(R2), R6
                                0322                INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), R6, R5, R5
                                0322                ADDL    dsc$a_a0(R2), R5
                                0322                MOVL    R0, (R5)
                                0322                BRW     30058$
                                0322        30055$: CMPB    dsc$b_dimct(R2), #1
                                0322                BNEQ    30070$
                                0322                MOVZWL  dsc$w_length(R2), R6
                                0322                INDEX   R4, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                                0322                ADDL    dsc$a_a0(R2), R5
                                0322                MOVL    R0, (R5)
                                0322                BRW     30058$
                                0322        30070$: INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), dsc$l_m1(R2), #0, R5
                                0322                MOVZWL  dsc$w_length(R2), R6
                                0322                INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), R6, R5, R5
                                0322                ADDL    dsc$a_a0(R2), R5
                                0322                MOVL    R0, (R5)
                                0322                .IFF
    18    02 A1    91         0322                CMPB    dsc$b_dtype(R1), #dsc$k_dtype_dsc
          41 12               0326                BNEQ    30071$
    50    04 A1    D0         0328                MOVL    4(R1), R0
0E  AE    02 A0    90         032C                MOVB    dsc$b_dtype(R0), dtype(SP)
0F  AE    03 A0    90         0331                MOVB    dsc$b_class(R0), class(SP)
10  AE     14 AE   DE         0336                MOVAL   data(SP), pointer (SP)
    0C AE   0A     B0         033B                MOVW    #10, str_len(SP)
```

```
                     01    0B A1    91  033F                CMPB    dsc$b_dimct(R1), #1
                                11    12  0343                BNEQ    30073$
                                52    DD  0345                PUSHL   R2
                                51    DD  0347                PUSHL   R1
                             14 AE    DF  0349                PUSHAL  value_desc+8(SP)
                00000000'GF    03    FB  034C                CALLS   #3,G^BAS$STORE_BFA
                              009D    31  0353                BRW     30058$
                                53    DD  0356        30073$: PUSHL   R3
                                52    DD  0358                PUSHL   R2
                                51    DD  035A                PUSHL   R1
                             18 AE    DF  035C                PUSHAL  value_desc+12(SP)
                00000000'GF    04    FB  035F                CALLS   #4,G^BAS$STORE_BFA
                              008A    31  0366                BRW     30058$
                   BF 8F    03 A1    91  0369        30071$: CMPB    dsc$b_class(R1), #dsc$k_class_bfa
                                09    12  036E                BNEQ    30056$
                  00000000'GF        16  0370                JSB     G^BAS$STO_FA_L_R8
                              007A    31  0376                BRW     30058$
                3C 0A A1       05    E0  0379        30056$: BBS     #5, 10(R1), 30057$
                   01    0B A1    91  037E                CMPB    dsc$b_dimct(R1), #1
                                16    12  0382                BNEQ    30072$
                             55    61  0384                MOVZWL  dsc$w_length(R1), R5
00    55    1C A1    18 A1    52    0A  0387                INDEX   R2, dsc$l_l1_1(R1), dsc$l_u1_1(R1), R5, #0, R4
                                54        038F
                      54    10 A1    C0  0390                ADDL    dsc$a_a0(R1), R4
                         64    50    D0  0394                MOVL    R0, (R4)
                               0059    31  0397                BRW     30058$
  18 A1    20 A1    1C A1    52    0A  039A        30072$: INDEX   R2, dsc$l_l1_2(R1), dsc$l_u1_2(R1), dsc$l_m2(R1), #0, R4
                                54    00  03A2
                             55    61  03A4                MOVZWL  dsc$w_length(R1), R5
54    55    28 A1    24 A1    53    0A  03A7                INDEX   R3, dsc$l_l2_2(R1), dsc$l_u2_2(R1), R5, R4, R4
                                54        03AF
                      54    10 A1    C0  03B0                ADDL    dsc$a_a0(R1), R4
                         64    50    D0  03B4                MOVL    R0, (R4)
                               0039    31  03B7                BRW     30058$
                   01    0B A1    91  03BA        30057$: CMPB    dsc$b_dimct(R1), #1
                                16    12  03BE                BNEQ    30074$
                             55    61  03C0                MOVZWL  dsc$w_length(R1), R5
00    55    1C A1    18 A1    53    0A  03C3                INDEX   R3, dsc$l_l1_1(R1), dsc$l_u1_1(R1), R5, #0, R4
                                54        03CB
                      54    10 A1    C0  03CC                ADDL    dsc$a_a0(R1), R4
                         64    50    D0  03D0                MOVL    R0, (R4)
                               001D    31  03D3                BRW     30058$
  14 A1    28 A1    24 A1    53    0A  03D6        30074$: INDEX   R3, dsc$l_l2_2(R1), dsc$l_u2_2(R1), dsc$l_m1(R1), #0, R4
                                54    00  03DE
                             55    61  03E0                MOVZWL  dsc$w_length(R1), R5
54    55    20 A1    1C A1    52    0A  03E3                INDEX   R2, dsc$l_l1_2(R1), dsc$l_u1_2(R1), R5, R4, R4
                                54        03EB
                      54    10 A1    C0  03EC                ADDL    dsc$a_a0(R1), R4
                         64    50    D0  03F0                MOVL    R0, (R4)
                                        03F3                .ENDC
                                        03F3                .ENDC
                                        03F3                .ENDC
                                        03F3        30058$:
                                        03F3
                                    5B  D6  03F3                INCL    R11                             ; get next column
                                 59 5B  D1  03F5                CMPL    R11, R9                         ; see if last column done
                                    03  14  03F8                BGTR    2$
```

```
                FF13    31  03FA              BRW     LOOP_2ND_SUBL           ; no, continue inner loop
                        03FD
                        03FD        ;+
                        03FD        ; Have completed entire row.  See if it was the last row.  If not,
                        03FD        ; continue with next row.
                        03FD        ;-
                        03FD
            04 AE   D6  03FD        2$:     INCL    lower_bnd1(SP)                  ; get next row
    08 AE   04 AE   D1  0400                CMPL    lower_bnd1(SP), upper_bnd1(SP)  ; see if last row done
               03   14  0405                BGTR    3$
                FF03    31  0407            BRW     LOOP_1ST_SUBL           ; no, continue outer loop
                        040A
                    04  040A        3$:     RET                                     ; yes, finished
                        040B
```

```
                040B   377 FLOAT:  $BAS$MAT_INIT F                          ; expand to float operations
                040B
                040B       ;+
                040B       ;       REGISTER USAGE
                040B       ;       R0 - R8 destroyed by store routines
                040B       ;       R9      upper bound for 2nd subscript
                040B       ;       R10     pointer to array descriptor
                040B       ;       R11     current value of 2nd subscript
                040B       ;-
                040B
                040B       ;+
                040B       ; Set up limits for looping through all elements
                040B       ;-
                040B
                040B               .IF     IDN     F, L
                040B               .IFT                                     ; data type is long
                040B               MOVL    constant(AP), -(SP)              ; move constant
                040B               .IFF                                     ; data type is not long
      7E  08 AC  4E 040B               CVTLF   constant(AP), -(SP)      ; make constant same datatype
                040F                                                        ;  as array, save on stack
                040F               .ENDC
                040F               .IF     IDN     F, D                 ; if array is double
                040F               MOVL    SF$L_SAVE_FP(FP), R0             ; pass FP to get scale
                040F               JSB     G^BAS$$SCALE_R1                  ; get scale in R0 & R1
                040F                                                        ; call a BLISS routine because
                040F                                                        ; the frame offsets are only
                040F                                                        ; defined for BLISS
                040F               MULD2   R0, (SP)                         ; scale
                040F               .ENDC
                040F
                040F       ;+
                040F       ; Allocate data and value_desc on the stack.  This applies to both
                040F       ; one and two dimensions.
                040F       ;-
                040F
        7E  7C  040F               CLRQ    -(SP)                            ; space for data
        7E  7C  0411               CLRQ    -(SP)                            ;  may be hfloat
        7E  7C  0413               CLRQ    -(SP)                            ; space for value_desc
                0415
01  0B AA  91   0415               CMPB    DSC$B_DIMCT(R10), #1             ; determine # of subscripts
     05 13  0419               BEQLU   INIT_ONE_SUBF            ; 1 sub, go init
     15 1A  041B               BGTRU   INIT_TWO_SUBSF           ; >=2 subs, go init
   FC2A 31  041D               BRW     ERR_ARGDONMAT                    ; 0 subs, error
                0420
                0420       ;+
                0420       ; There is only 1 subscript.  Make both upper and lower bound for 2nd
                0420       ; subscript a 1.  The second subscript will be passed to and ignored by the
                0420       ; store routine.
                0420       ;-
                0420
                0420       INIT_ONE_SUBF:
    1C AA  DD   0420               PUSHL   dsc$l_u1_1(R10)                  ; 1st upper bound
    18 AA  DD   0423               PUSHL   dsc$l_l1_1(R10)                  ; 1st lower bound
     03 14  0426               BGTR    1$                               ; not 0 or neg, do 2nd sub
  6E 01 D0  0428               MOVL    #1, (SP)                         ; don't alter col 0
  59 01 D0  042B       1$:     MOVL    #1, R9                           ; dummy 2nd lower bound
     01 DD  042E               PUSHL   #1                               ; dummy 2nd upper bound
```

```
                1A   11  0430                BRB     LOOP_2ND_SUBF           ; go loop
                    0432
                    0432        ;+
                    0432        ; There are 2 subscripts.  Put the upper bound for both subscripts on the
                    0432        ; stack and make sure that the lower bound for both subscripts will start
                    0432        ; at 1 (do not alter row or col 0)
                    0432        ;-
                    0432
                    0432        INIT_TWO_SUBSF:
              20 AA DD  0432                PUSHL   dsc$l_u1_2(R10)                 ; 1st upper bound
              1C AA DD  0435                PUSHL   dsc$l_l1_2(R10)                 ; 1st lower bound
                 03 14  0438                BGTR    1$                              ; not row 0 or neg, do cols
           6E    01 D0  043A                MOVL    #1, (SP)                        ; start with row 1
        59    28 AA D0  043D        1$:     MOVL    dsc$l_u2_2(R10), R9             ; 2nd upper bound
              24 AA DD  0441                PUSHL   dsc$l_l2_2(R10)                 ; 2nd lower bound
                 03 14  0444                BGTR    LOOP_TST_SUBF           ; not col 0, go loop
           6E    01 D0  0446                MOVL    #1, (SP)                        ; start with col 1
                    0449
                    0449        ;+
                    0449        ; Loop through all the rows.  Row and column upper and lower bounds have been
                    0449        ; initialized on the stack.
                    0449        ;-
                    0449
                    0449        LOOP_1ST_SUBF:
           5B   6E   D0  0449                MOVL    lower_bnd2(SP), R11             ; R11 has 2nd lower bound
                    044C
                    044C        ;+
                    044C        ; Loop through all the elements (columns) of the current row.  Column lower
                    044C        ; bound is initialized in R11.  Column upper bound is on the stack.
                    044C        ; Distinguish array by data type so that the correct store routine can be
                    044C        ; called and the constant can be converted to the correct type.
                    044C        ;-
                    044C
                    044C        LOOP_2ND_SUBF:
                    044C
     50    24 AE   50  044C                MOVF    constant_cvt(SP), R0    ; put constant into R0
                    0450                                                        ;  R0 & R1 for double
                    0450        ;+
                    0450        ; When passed by value, hfloat takes 4 words, gfloat and double take 2 words,
                    0450        ; and all other data types take 1 longword.
                    0450        ;-
                    0450
                    0450                .IF     IDN     F, H            ; data type is hfloat
                    0450                MOVL    R10, R4                         ; pointer to array desc
                    0450                MOVL    lower_bnd1(SP), R5              ; current row
                    0450                MOVL    R11, R6                         ; current column
                    0450                .IFF
                    0450                .IF     IDN     F, G            ; data type is gfloat
                    0450                MOVL    R10, R2                         ; pointer to array desc
                    0450                MOVL    lower_bnd1(SP), R3              ; current row
                    0450                MOVL    R11, R4                         ; current column
                    0450                .IFF
                    0450                .IF     IDN     F, D            ; data type is double
                    0450                MOVL    R10, R2                         ; pointer to array desc
                    0450                MOVL    lower_bnd1(SP), R3              ; current row
                    0450                MOVL    R11, R4                         ; current column
                    0450                .IFF                                    ; all other data types
```

```
                51    5A    D0  0450                MOVL    R10, R1                         ; pointer to array desc
            52   04 AE    D0  0453                MOVL    lower_bnd1(SP), R2              ; current row
                53    5B    D0  0457                MOVL    R11, R3                         ; current column
                                045A                .ENDC
                                045A                .ENDC
                                045A                .ENDC
            14 AE    50    50  045A                MOVF    R0, data(SP)            ; store value in value_desc
                                045E                STORE   F                               ; store in array
                                045E                .IF     IDN     F, H
                                045E                CMPB    dsc$b_dtype(R4), #dsc$k_dtype_dsc
                                045E                BNEQ    30084$
                                045E                MOVL    4(R4), R0
                                045E                MOVB    dsc$b_dtype(R0), dtype(SP)
                                045E                MOVB    dsc$b_class(R0), class(SP)
                                045E                MOVAL   data(SP), pointer (SP)
                                045E                MOVW    #10, str_len(SP)
                                045E                CMPB    dsc$b_dimct(R4), #1
                                045E                BNEQ    30086$
                                045E                PUSHL   R5
                                045E                PUSHL   R4
                                045E                PUSHAL  value_desc+8(SP)
                                045E                CALLS   #3,G^BAS$STORE_BFA
                                045E                BRW     30083$
                                045E        30086$: PUSHL   R6
                                045E                PUSHL   R5
                                045E                PUSHL   R4
                                045E                PUSHAL  value_desc+12(SP)
                                045E                CALLS   #4,G^BAS$STORE_BFA
                                045E                BRW     30083$
                                045E        30084$: CMPB    dsc$b_class(R4), #dsc$k_class_bfa
                                045E                BNEQ    30075$
                                045E                JSB     G^BAS$STO_FA_F_R8
                                045E                BRW     30083$
                                045E        30075$: BBS     #5, 10(R4), 30076$
                                045E                CMPB    dsc$b_dimct(R4), #1
                                045E                BNEQ    30085$
                                045E                MOVZWL  dsc$w_length(R4), R8
                                045E                INDEX   R5, dsc$l_l1_1(R4), dsc$l_u1_1(R4), R8, #0, R7
                                045E                ADDL    dsc$a_a0(R4), R7
                                045E                MOVF    R0, (R7)
                                045E                BRW     30083$
                                045E        30085$: INDEX   R5, dsc$l_l1_2(R4), dsc$l_u1_2(R4), dsc$l_m2(R4), #0, R7
                                045E                MOVZWL  dsc$w_length(R4), R8
                                045E                INDEX   R6, dsc$l_l2_2(R4), dsc$l_u2_2(R4), R8, R7, R7
                                045E                ADDL    dsc$a_a0(R4), R7
                                045E                MOVF    R0, (R7)
                                045E                BRW     30083$
                                045E        30076$: CMPB    dsc$b_dimct(R4), #1
                                045E                BNEQ    30087$
                                045E                MOVZWL  dsc$w_length(R4), R8
                                045E                INDEX   R6, dsc$l_l1_1(R4), dsc$l_u1_1(R4), R8, #0, R7
                                045E                ADDL    dsc$a_a0(R4), R7
                                045E                MOVF    R0, (R7)
                                045E                BRW     30083$
                                045E        30087$: INDEX   R6, dsc$l_l2_2(R4), dsc$l_u2_2(R4), dsc$l_m1(R4), #0, R7
                                045E                MOVZWL  dsc$w_length(R4), R8
                                045E                INDEX   R5, dsc$l_l1_2(R4), dsc$l_u1_2(R4), R8, R7, R7
```

```
045E                    ADDL    dsc$a_a0(R4), R7
045E                    MOVF    R0, (R7)
045E                    .IFF
045E                    .IF     IDN     F, G
045E                    CMPB    dsc$b_dtype(R2), #dsc$k_dtype_dsc
045E                    BNEQ    30088$
045E                    MOVL    4(R2), R0
045E                    MOVB    dsc$b_dtype(R0), dtype(SP)
045E                    MOVB    dsc$b_class(R0), class(SP)
045E                    MOVAL   data(SP), pointer (SP)
045E                    MOVW    #10, str_len(SP)
045E                    CMPB    dsc$b_dimct(R2), #1
045E                    BNEQ    30090$
045E                    PUSHL   R3
045E                    PUSHL   R2
045E                    PUSHAL  value_desc+8(SP)
045E                    CALLS   #3,G^BAS$STORE_BFA
045E                    BRW     30083$
045E            30090$: PUSHL   R4
045E                    PUSHL   R3
045E                    PUSHL   R2
045E                    PUSHAL  value_desc+12(SP)
045E                    CALLS   #4,G^BAS$STORE_BFA
045E                    BRW     30083$
045E            30088$: CMPB    dsc$b_class(R2), #dsc$k_class_bfa
045E                    BNEQ    30077$
045E                    JSB     G^BAS$STO_FA_F_R8
045E                    BRW     30083$
045E            30077$: BBS     #5, 10(R2), 30078$
045E                    CMPB    dsc$b_dimct(R2), #1
045E                    BNEQ    30089$
045E                    MOVZWL  dsc$w_length(R2), R6
045E                    INDEX   R3, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
045E                    ADDL    dsc$a_a0(R2), R5
045E                    MOVF    R0, (R5)
045E                    BRW     30083$
045E            30089$: INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), dsc$l_m2(R2), #0, R5
045E                    MOVZWL  dsc$w_length(R2), R6
045E                    INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), R6, R5, R5
045E                    ADDL    dsc$a_a0(R2), R5
045E                    MOVF    R0, (R5)
045E                    BRW     30083$
045E            30078$: CMPB    dsc$b_dimct(R2), #1
045E                    BNEQ    30091$
045E                    MOVZWL  dsc$w_length(R2), R6
045E                    INDEX   R4, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
045E                    ADDL    dsc$a_a0(R2), R5
045E                    MOVF    R0, (R5)
045E                    BRW     30083$
045E            30091$: INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), dsc$l_m1(R2), #0, R5
045E                    MOVZWL  dsc$w_length(R2), R6
045E                    INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), R6, R5, R5
045E                    ADDL    dsc$a_a0(R2), R5
045E                    MOVF    R0, (R5)
045E                    .IFF
045E                    .IF     IDN     F, D
045E                    CMPB    dsc$b_dtype(R2), #dsc$k_dtype_dsc
```

```
                045E                    BNEQ    30092$
                045E                    MOVL    4(R2), R0
                045E                    MOVB    dsc$b_dtype(R0), dtype(SP)
                045E                    MOVB    dsc$b_class(R0), class(SP)
                045E                    MOVAL   data(SP), pointer (SP)
                045E                    MOVW    #10, str_len(SP)
                045E                    CMPB    dsc$b_dimct(R2), #1
                045E                    BNEQ    30094$
                045E                    PUSHL   R3
                045E                    PUSHL   R2
                045E                    PUSHAL  value_desc+8(SP)
                045E                    CALLS   #3,G^BAS$STORE_BFA
                045E                    BRW     30083$
                045E            30094$: PUSHL   R4
                045E                    PUSHL   R3
                045E                    PUSHL   R2
                045E                    PUSHAL  value_desc+12(SP)
                045E                    CALLS   #4,G^BAS$STORE_BFA
                045E                    BRW     30083$
                045E            30092$: CMPB    dsc$b_class(R2), #dsc$k_class_bfa
                045E                    BNEQ    30079$
                045E                    JSB     G^BAS$STO_FA_F_R8
                045E                    BRW     30083$
                045E            30079$: BBS     #5, 10(R2), 30080$
                045E                    CMPB    dsc$b_dimct(R2), #1
                045E                    BNEQ    30093$
                045E                    MOVZWL  dsc$w_length(R2), R6
                045E                    INDEX   R3, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                045E                    ADDL    dsc$a_a0(R2), R5
                045E                    MOVF    R0, (R5)
                045E                    BRW     30083$
                045E            30093$: INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), dsc$l_m2(R2), #0, R5
                045E                    MOVZWL  dsc$w_length(R2), R6
                045E                    INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), R6, R5, R5
                045E                    ADDL    dsc$a_a0(R2), R5
                045E                    MOVF    R0, (R5)
                045E                    BRW     30083$
                045E            30080$: CMPB    dsc$b_dimct(R2), #1
                045E                    BNEQ    30095$
                045E                    MOVZWL  dsc$w_length(R2), R6
                045E                    INDEX   R4, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                045E                    ADDL    dsc$a_a0(R2), R5
                045E                    MOVF    R0, (R5)
                045E                    BRW     30083$
                045E            30095$: INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), dsc$l_m1(R2), #0, R5
                045E                    MOVZWL  dsc$w_length(R2), R6
                045E                    INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), R6, R5, R5
                045E                    ADDL    dsc$a_a0(R2), R5
                045E                    MOVF    R0, (R5)
                045E                    .IFF
    18  02 A1   91  045E                CMPB    dsc$b_dtype(R1), #dsc$k_dtype_dsc
        41      12  0462                BNEQ    30096$
    50  04 A1   D0  0464                MOVL    4(R1), R0
0E AE   02 A0   90  0468                MOVB    dsc$b_dtype(R0), dtype(SP)
0F AE   03 A0   90  046D                MOVB    dsc$b_class(R0), class(SP)
10 AE   14 AE   DE  0472                MOVAL   data(SP), pointer (SP)
   0C AE    0A   B0  0477                MOVW    #10, str_len(SP)
```

```
                    01    0B A1   91   047B                CMPB    dsc$b_dimct(R1), #1
                                11   12   047F                BNEQ    30098$
                                52   DD   0481                PUSHL   R2
                                51   DD   0483                PUSHL   R1
                             14 AE   DF   0485                PUSHAL  value_desc+8(SP)
                00000000'GF    03   FB   0488                CALLS   #3,G^BAS$STORE_BFA
                             009D   31   048F                BRW     30083$
                                53   DD   0492    30098$:     PUSHL   R3
                                52   DD   0494                PUSHL   R2
                                51   DD   0496                PUSHL   R1
                             18 AE   DF   0498                PUSHAL  value_desc+12(SP)
                00000000'GF    04   FB   049B                CALLS   #4,G^BAS$STORE_BFA
                             008A   31   04A2                BRW     30083$
                   BF 8F    03 A1   91   04A5    30096$:     CMPB    dsc$b_class(R1), #dsc$k_class_bfa
                                09   12   04AA                BNEQ    30081$
                  00000000'GF       16   04AC                JSB     G^BAS$STO_FA_F_R8
                             007A   31   04B2                BRW     30083$
                   3C 0A A1    05   E0   04B5    30081$:     BBS     #5, 10(R1), 30082$
                    01    0B A1   91   04BA                CMPB    dsc$b_dimct(R1), #1
                                16   12   04BE                BNEQ    30097$
                          55    61   3C   04C0                MOVZWL  dsc$w_length(R1), R5
 00    55    1C A1    18 A1    52   0A   04C3                INDEX   R2, dsc$l_l1_1(R1), dsc$l_u1_1(R1), R5, #0, R4
                                54        04CB
                    54    10 A1   C0   04CC                ADDL    dsc$a_a0(R1), R4
                       64    50   50   04D0                MOVF    R0, (R4)
                             0059   31   04D3                BRW     30083$
   18 A1    20 A1    1C A1    52   0A   04D6    30097$:     INDEX   R2, dsc$l_l1_2(R1), dsc$l_u1_2(R1), dsc$l_m2(R1), #0, R4
                          54    00        04DE
                          55    61   3C   04E0                MOVZWL  dsc$w_length(R1), R5
 54    55    28 A1    24 A1    53   0A   04E3                INDEX   R3, dsc$l_l2_2(R1), dsc$l_u2_2(R1), R5, R4, R4
                                54        04EB
                    54    10 A1   C0   04EC                ADDL    dsc$a_a0(R1), R4
                       64    50   50   04F0                MOVF    R0, (R4)
                             0039   31   04F3                BRW     30083$
                    01    0B A1   91   04F6    30082$:     CMPB    dsc$b_dimct(R1), #1
                                16   12   04FA                BNEQ    30099$
                          55    61   3C   04FC                MOVZWL  dsc$w_length(R1), R5
 00    55    1C A1    18 A1    53   0A   04FF                INDEX   R3, dsc$l_l1_1(R1), dsc$l_u1_1(R1), R5, #0, R4
                                54        0507
                    54    10 A1   C0   0508                ADDL    dsc$a_a0(R1), R4
                       64    50   50   050C                MOVF    R0, (R4)
                             001D   31   050F                BRW     30083$
   14 A1    28 A1    24 A1    53   0A   0512    30099$:     INDEX   R3, dsc$l_l2_2(R1), dsc$l_u2_2(R1), dsc$l_m1(R1), #0, R4
                          54    00        051A
                          55    61   3C   051C                MOVZWL  dsc$w_length(R1), R5
 54    55    20 A1    1C A1    52   0A   051F                INDEX   R2, dsc$l_l1_2(R1), dsc$l_u1_2(R1), R5, R4, R4
                                54        0527
                    54    10 A1   C0   0528                ADDL    dsc$a_a0(R1), R4
                       64    50   50   052C                MOVF    R0, (R4)
                                      052F                .ENDC
                                      052F                .ENDC
                                      052F                .ENDC
                                      052F    30083$:
                                      052F
                          5B    D6   052F                INCL    R11                             ; get next column
                       59    5B    D1   0531                CMPL    R11, R9                         ; see if last column done
                             03    14   0534                BGTR    2$
```

```
                FF13   31  0536                BRW     LOOP_2ND_SUBF           ; no, continue inner loop
                       0539
                       0539        ;+
                       0539        ; Have completed entire row.  See if it was the last row.  If not,
                       0539        ; continue with next row.
                       0539        ;-
                       0539
            04 AE      D6  0539    2$:     INCL    lower_bnd1(SP)                  ; get next row
      08 AE 04 AE      D1  053C            CMPL    lower_bnd1(SP), upper_bnd1(SP)  ; see if last row done
                 03    14  0541            BGTR    3$
                FF03   31  0543            BRW     LOOP_1ST_SUBF           ; no, continue outer loop
                       0546
                       04  0546    3$:     RET                                     ; yes, finished
                           0547
```

```
                     0547   379 DOUBLE: $BAS$MAT_INIT D                         ; expand to double operations
                     0547
                     0547       ;+
                     0547       ;       REGISTER USAGE
                     0547       ;       R0 - R8 destroyed by store routines
                     0547       ;       R9      upper bound for 2nd subscript
                     0547       ;       R10     pointer to array descriptor
                     0547       ;       R11     current value of 2nd subscript
                     0547       ;-
                     0547
                     0547       ;+
                     0547       ; Set up limits for looping through all elements
                     0547       ;-
                     0547
                     0547               .IF     IDN     D, L
                     0547               .IFT                                    ; data type is long
                     0547               MOVL    constant(AP), -(SP)             ; move constant
                     0547               .IFF                                    ; data type is not long
   7E    08 AC    6E  0547               CVTLD   constant(AP), -(SP)     ; make constant same datatype
                     054B                                                       ;  as array, save on stack
                     054B               .ENDC
                     054B               .IF     IDN     D, D            ; if array is double
   50    0C AD    D0  054B               MOVL    SF$L_SAVE_FP(FP), R0            ; pass FP to get scale
00000000'GF    16    054F               JSB     G^BAS$$SCALE_R1                 ; get scale in R0 & R1
                     0555                                                       ; call a BLISS routine because
                     0555                                                       ; the frame offsets are only
                     0555                                                       ; defined for BLISS
   6E    50    64  0555               MULD2   R0, (SP)                        ; scale
                     0558               .ENDC
                     0558
                     0558       ;+
                     0558       ; Allocate data and value_desc on the stack.  This applies to both
                     0558       ; one and two dimensions.
                     0558       ;-
                     0558
   7E    7C    0558               CLRQ    -(SP)                               ; space for data
   7E    7C    055A               CLRQ    -(SP)                               ;  may be hfloat
   7E    7C    055C               CLRQ    -(SP)                               ; space for value_desc
                     055E
01 0B AA    91    055E               CMPB    DSC$B_DIMCT(R10), #1                ; determine # of subscripts
   05    13    0562               BEQLU   INIT_ONE_SUBD           ; 1 sub, go init
   15    1A    0564               BGTRU   INIT_TWO_SUBSD          ; >=2 subs, go init
 FAE1    31    0566               BRW     ERR_ARGDONMAT                       ; 0 subs, error
                     0569
                     0569       ;+
                     0569       ; There is only 1 subscript.  Make both upper and lower bound for 2nd
                     0569       ; subscript a 1.  The second subscript will be passed to and ignored by the
                     0569       ; store routine.
                     0569       ;-
                     0569
                     0569       INIT_ONE_SUBD:
   1C AA    DD    0569               PUSHL   dsc$l_u1_1(R10)                     ; 1st upper bound
   18 AA    DD    056C               PUSHL   dsc$l_l1_1(R10)                     ; 1st lower bound
   03    14    056F               BGTR    1$                                  ; not 0 or neg, do 2nd sub
6E 01    D0    0571               MOVL    #1, (SP)                            ; don't alter col 0
59 01    D0    0574       1$:     MOVL    #1, R9                              ; dummy 2nd lower bound
   01    DD    0577               PUSHL   #1                                  ; dummy 2nd upper bound
```

```
                1A   11  0579                BRB     LOOP_2ND_SUBD           ; go loop
                     0578
                     0578        ;+
                     0578        ; There are 2 subscripts.  Put the upper bound for both subscripts on the
                     0578        ; stack and make sure that the lower bound for both subscripts will start
                     0578        ; at 1 (do not alter row or col 0)
                     0578        ;-
                     0578
                     0578        INIT_TWO_SUBSD:
              20 AA  DD  057B                PUSHL   dsc$l_u1_2(R10)                 ; 1st upper bound
              1C AA  DD  057E                PUSHL   dsc$l_l1_2(R10)                 ; 1st lower bound
                 03  14  0581                BGTR    1$                              ; not row 0 or neg, do cols
           6E    01  D0  0583                MOVL    #1, (SP)                        ; start with row 1
        59 28 AA  D0  0586        1$:       MOVL    dsc$l_u2_2(R10), R9             ; 2nd upper bound
              24 AA  DD  058A                PUSHL   dsc$l_l2_2(R10)                 ; 2nd lower bound
                 03  14  058D                BGTR    LOOP_1ST_SUBD           ; not col 0, go loop
           6E    01  D0  058F                MOVL    #1, (SP)                        ; start with col 1
                     0592
                     0592        ;+
                     0592        ; Loop through all the rows.  Row and column upper and lower bounds have been
                     0592        ; initialized on the stack.
                     0592        ;-
                     0592
                     0592        LOOP_1ST_SUBD:
           5B    6E  D0  0592                MOVL    lower_bnd2(SP), R11             ; R11 has 2nd lower bound
                     0595
                     0595        ;+
                     0595        ; Loop through all the elements (columns) of the current row.  Column lower
                     0595        ; bound is initialized in R11.  Column upper bound is on the stack.
                     0595        ; Distinguish array by data type so that the correct store routine can be
                     0595        ; called and the constant can be converted to the correct type.
                     0595        ;-
                     0595
                     0595        LOOP_2ND_SUBD:
                     0595
        50    24 AE   70  0595                MOVD    constant_cvt(SP), R0    ; put constant into R0
                     0599                                                        ;  R0 & R1 for double
                     0599        ;+
                     0599        ; When passed by value, hfloat takes 4 words, gfloat and double take 2 words,
                     0599        ; and    other data types take 1 longword.
                     0599        ;-
                     0599
                     0599                .IF     IDN     D, H                    ; data type is hfloat
                     0599                MOVL    R10, R4                         ; pointer to array desc
                     0599                MOVL    lower_bnd1(SP), R5              ; current row
                     0599                MOVL    R11, R6                         ; current column
                     0599                .IFF
                     0599                .IF     IDN     D, G                    ; data type is gfloat
                     0599                MOVL    R10, R2                         ; pointer to array desc
                     0599                MOVL    lower_bnd1(SP), R3              ; current row
                     0599                MOVL    R11, R4                         ; current column
                     0599                .IFF
                     0599                .IF     IDN     D, D                    ; data type is double
           52    5A  D0  0599                MOVL    R10, R2                         ; pointer to array desc
        53    04 AE  D0  059C                MOVL    lower_bnd1(SP), R3              ; current row
           54    5B  D0  05A0                MOVL    R11, R4                         ; current column
                     05A3                .IFF                                    ; all other data types
```

```
                     05A3                MOVL    R10, R1                         ; pointer to array desc
                     05A3                MOVL    lower_bnd1(SP), R2              ; current row
                     05A3                MOVL    R11, R3                         ; current column
                     05A3                .ENDC
                     05A3                .ENDC
                     05A3                .ENDC
14 AE   50   70      05A3                MOVD    R0, data(SP)            ; store value in value_desc
                     05A7                STORE   D                               ; store in array
                     05A7                .IF     IDN     D, H
                     05A7                CMPB    dsc$b_dtype(R4), #dsc$k_dtype_dsc
                     05A7                BNEQ    30109$
                     05A7                MOVL    4(R4), R0
                     05A7                MOVB    dsc$b_dtype(R0), dtype(SP)
                     05A7                MOVB    dsc$b_class(R0), class(SP)
                     05A7                MOVAL   data(SP), pointer (SP)
                     05A7                MOVW    #10, str_len(SP)
                     05A7                CMPB    dsc$b_dimct(R4), #1
                     05A7                BNEQ    30111$
                     05A7                PUSHL   R5
                     05A7                PUSHL   R4
                     05A7                PUSHAL  value_desc+8(SP)
                     05A7                CALLS   #3,G^BAS$STORE_BFA
                     05A7                BRW     30108$
                     05A7        30111$: PUSHL   R6
                     05A7                PUSHL   R5
                     05A7                PUSHL   R4
                     05A7                PUSHAL  value_desc+12(SP)
                     05A7                CALLS   #4,G^BAS$STORE_BFA
                     05A7                BRW     30108$
                     05A7        30109$: CMPB    dsc$b_class(R4), #dsc$k_class_bfa
                     05A7                BNEQ    30100$
                     05A7                JSB     G^BAS$STO_FA_D_R8
                     05A7                BRW     30108$
                     05A7        30100$: BBS     #5, 10(R4), 30101$
                     05A7                CMPB    dsc$b_dimct(R4), #1
                     05A7                BNEQ    30110$
                     05A7                MOVZWL  dsc$w_length(R4), R8
                     05A7                INDEX   R5, dsc$l_l1_1(R4), dsc$l_u1_1(R4), R8, #0, R7
                     05A7                ADDL    dsc$a_a0(R4), R7
                     05A7                MOVD    R0, (R7)
                     05A7                BRW     30108$
                     05A7        30110$: INDEX   R5, dsc$l_l1_2(R4), dsc$l_u1_2(R4), dsc$l_m2(R4), #0, R7
                     05A7                MOVZWL  dsc$w_length(R4), R8
                     05A7                INDEX   R6, dsc$l_l2_2(R4), dsc$l_u2_2(R4), R8, R7, R7
                     05A7                ADDL    dsc$a_a0(R4), R7
                     05A7                MOVD    R0, (R7)
                     05A7                BRW     30108$
                     05A7        30101$: CMPB    dsc$b_dimct(R4), #1
                     05A7                BNEQ    30112$
                     05A7                MOVZWL  dsc$w_length(R4), R8
                     05A7                INDEX   R6, dsc$l_l1_1(R4), dsc$l_u1_1(R4), R8, #0, R7
                     05A7                ADDL    dsc$a_a0(R4), R7
                     05A7                MOVD    R0, (R7)
                     05A7                BRW     30108$
                     05A7        30112$: INDEX   R6, dsc$l_l2_2(R4), dsc$l_u2_2(R4), dsc$l_m1(R4), #0, R7
                     05A7                MOVZWL  dsc$w_length(R4), R8
                     05A7                INDEX   R5, dsc$l_l1_2(R4), dsc$l_u1_2(R4), R8, R7, R7
```

```
                05A7                    ADDL    dsc$a_a0(R4), R7
                05A7                    MOVD    R0, (R7)
                05A7                    .IFF
                05A7                    .IF     IDN     D, G
                05A7                    CMPB    dsc$b_dtype(R2), #dsc$k_dtype_dsc
                05A7                    BNEQ    30113$
                05A7                    MOVL    4(R2), R0
                05A7                    MOVB    dsc$b_dtype(R0), dtype(SP)
                05A7                    MOVB    dsc$b_class(R0), class(SP)
                05A7                    MOVAL   data(SP), pointer (SP)
                05A7                    MOVW    #10, str_len(SP)
                05A7                    CMPB    dsc$b_dimct(R2), #1
                05A7                    BNEQ    30115$
                05A7                    PUSHL   R3
                05A7                    PUSHL   R2
                05A7                    PUSHAL  value_desc+8(SP)
                05A7                    CALLS   #3,G^BAS$STORE_BFA
                05A7                    BRW     30108$
                05A7            30115$: PUSHL   R4
                05A7                    PUSHL   R3
                05A7                    PUSHL   R2
                05A7                    PUSHAL  value_desc+12(SP)
                05A7                    CALLS   #4,G^BAS$STORE_BFA
                05A7                    BRW     30108$
                05A7            30113$: CMPB    dsc$b_class(R2), #dsc$k_class_bfa
                05A7                    BNEQ    30102$
                05A7                    JSB     G^BAS$STO_FA_D_R8
                05A7                    BRW     30108$
                05A7            30102$: BBS     #5, 10(R2), 30103$
                05A7                    CMPB    dsc$b_dimct(R2), #1
                05A7                    BNEQ    30114$
                05A7                    MOVZWL  dsc$w_length(R2), R6
                05A7                    INDEX   R3, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                05A7                    ADDL    dsc$a_a0(R2), R5
                05A7                    MOVD    R0, (R5)
                05A7                    BRW     30108$
                05A7            30114$: INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), dsc$l_m2(R2), #0, R5
                05A7                    MOVZWL  dsc$w_length(R2), R6
                05A7                    INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), R6, R5, R5
                05A7                    ADDL    dsc$a_a0(R2), R5
                05A7                    MOVD    R0, (R5)
                05A7                    BRW     30108$
                05A7            30103$: CMPB    dsc$b_dimct(R2), #1
                05A7                    BNEQ    30116$
                05A7                    MOVZWL  dsc$w_length(R2), R6
                05A7                    INDEX   R4, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                05A7                    ADDL    dsc$a_a0(R2), R5
                05A7                    MOVD    R0, (R5)
                05A7                    BRW     30108$
                05A7            30116$: INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), dsc$l_m1(R2), #0, R5
                05A7                    MOVZWL  dsc$w_length(R2), R6
                05A7                    INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), R6, R5, R5
                05A7                    ADDL    dsc$a_a0(R2), R5
                05A7                    MOVD    R0, (R5)
                05A7                    .IFF
                05A7                    .IF     IDN     D, D
18   02 A2   91 05A7                    CMPB    dsc$b_dtype(R2), #dsc$k_dtype_dsc
```

```
                     41   12  05AB                BNEQ    30117$
                50   04 A2   D0  05AD                MOVL    4(R2), R0
             0E AE   02 A0   90  05B1                MOVB    dsc$b_dtype(R0), dtype(SP)
             0F AE   03 A0   90  05B6                MOVB    dsc$b_class(R0), class(SP)
             10 AE   14 AE   DE  05BB                MOVAL   data(SP), pointer (SP)
                0C AE   0A   B0  05C0                MOVW    #10, str_len(SP)
                01   0B A2   91  05C4                CMPB    dsc$b_dimct(R2), #1
                     11   12  05C8                BNEQ    30119$
                     53   DD  05CA                PUSHL   R3
                     52   DD  05CC                PUSHL   R2
                  14 AE   DF  05CE                PUSHAL  value_desc+8(SP)
          00000000'GF   03   FB  05D1                CALLS   #3,G^BAS$STORE_BFA
                   009D   31  05D8                BRW     30108$
                     54   DD  05DB      30119$:   PUSHL   R4
                     53   DD  05DD                PUSHL   R3
                     52   DD  05DF                PUSHL   R2
                  18 AE   DF  05E1                PUSHAL  value_desc+12(SP)
          00000000'GF   04   FB  05E4                CALLS   #4,G^BAS$STORE_BFA
                   008A   31  05EB                BRW     30108$
             BF 8F   03 A2   91  05EE      30117$:   CMPB    dsc$b_class(R2), #dsc$k_class_bfa
                     09   12  05F3                BNEQ    30104$
                00000000'GF   16  05F5                JSB     G^BAS$STO_FA_D_R8
                   007A   31  05FB                BRW     30108$
             3C 0A A2   05   E0  05FE      30104$:   BBS     #5, 10(R2), 30105$
                01   0B A2   91  0603                CMPB    dsc$b_dimct(R2), #1
                     16   12  0607                BNEQ    30118$
                  56   62   3C  0609                MOVZWL  dsc$w_length(R2), R6
00   56   1C A2   18 A2   53   0A  060C                INDEX   R3, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                     55       0614
                55   10 A2   C0  0615                ADDL    dsc$a_a0(R2), R5
                   65   50   70  0619                MOVD    R0, (R5)
                    0059   31  061C                BRW     30108$
  18 A2   20 A2   1C A2   53   0A  061F      30118$:   INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), dsc$l_m2(R2), #0, R5
                     55   00      0627
                  56   62   3C  0629                MOVZWL  dsc$w_length(R2), R6
55   56   28 A2   24 A2   54   0A  062C                INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), R6, R5, R5
                     55       0634
                55   10 A2   C0  0635                ADDL    dsc$a_a0(R2), R5
                   65   50   70  0639                MOVD    R0, (R5)
                    0039   31  063C                BRW     30108$
                01   0B A2   91  063F      30105$:   CMPB    dsc$b_dimct(R2), #1
                     16   12  0643                BNEQ    30120$
                  56   62   3C  0645                MOVZWL  dsc$w_length(R2), R6
00   56   1C A2   18 A2   54   0A  0648                INDEX   R4, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                     55       0650
                55   10 A2   C0  0651                ADDL    dsc$a_a0(R2), R5
                   65   50   70  0655                MOVD    R0, (R5)
                    001D   31  0658                BRW     30108$
  14 A2   28 A2   24 A2   54   0A  065B      30120$:   INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), dsc$l_m1(R2), #0, R5
                     55   00      0663
                  56   62   3C  0665                MOVZWL  dsc$w_length(R2), R6
55   56   20 A2   1C A2   53   0A  0668                INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), R6, R5, R5
                     55       0670
                55   10 A2   C0  0671                ADDL    dsc$a_a0(R2), R5
                   65   50   70  0675                MOVD    R0, (R5)
                                0678                .IFF
                                0678                CMPB    dsc$b_dtype(R1), #dsc$k_dtype_dsc
```

```
                0678                    BNEQ    30121$
                0678                    MOVL    4(R1), R0
                0678                    MOVB    dsc$b_dtype(R0), dtype(SP)
                0678                    MOVB    dsc$b_class(R0), class(SP)
                0678                    MOVAL   data(SP), pointer (SP)
                0678                    MOVW    #10, str_len(SP)
                0678                    CMPB    dsc$b_dimct(R1), #1
                0678                    BNEQ    30123$
                0678                    PUSHL   R2
                0678                    PUSHL   R1
                0678                    PUSHAL  value_desc+8(SP)
                0678                    CALLS   #3,G^BAS$STORE_BFA
                0678                    BRW     30108$
                0678            30123$: PUSHL   R3
                0678                    PUSHL   R2
                0678                    PUSHL   R1
                0678                    PUSHAL  value_desc+12(SP)
                0678                    CALLS   #4,G^BAS$STORE_BFA
                0678                    BRW     30108$
                0678            30121$: CMPB    dsc$b_class(R1), #dsc$k_class_bfa
                0678                    BNEQ    30106$
                0678                    JSB     G^BAS$STO_FA_D_R8
                0678                    BRW     30108$
                0678            30106$: BBS     #5, 10(R1), 30107$
                0678                    CMPB    dsc$b_dimct(R1), #1
                0678                    BNEQ    30122$
                0678                    MOVZWL  dsc$w_length(R1), R5
                0678                    INDEX   R2, dsc$l_l1_1(R1), dsc$l_u1_1(R1), R5, #0, R4
                0678                    ADDL    dsc$a_a0(R1), R4
                0678                    MOVD    R0, (R4)
                0678                    BRW     30108$
                0678            30122$: INDEX   R2, dsc$l_l1_2(R1), dsc$l_u1_2(R1), dsc$l_m2(R1), #0, R4
                0678                    MOVZWL  dsc$w_length(R1), R5
                0678                    INDEX   R3, dsc$l_l2_2(R1), dsc$l_u2_2(R1), R5, R4, R4
                0678                    ADDL    dsc$a_a0(R1), R4
                0678                    MOVD    R0, (R4)
                0678                    BRW     30108$
                0678            30107$: CMPB    dsc$b_dimct(R1), #1
                0678                    BNEQ    30124$
                0678                    MOVZWL  dsc$w_length(R1), R5
                0678                    INDEX   R3, dsc$l_l1_1(R1), dsc$l_u1_1(R1), R5, #0, R4
                0678                    ADDL    dsc$a_a0(R1), R4
                0678                    MOVD    R0, (R4)
                0678                    BRW     30108$
                0678            30124$: INDEX   R3, dsc$l_l2_2(R1), dsc$l_u2_2(R1), dsc$l_m1(R1), #0, R4
                0678                    MOVZWL  dsc$w_length(R1), R5
                0678                    INDEX   R2, dsc$l_l1_2(R1), dsc$l_u1_2(R1), R5, R4, R4
                0678                    ADDL    dsc$a_a0(R1), R4
                0678                    MOVD    R0, (R4)
                0678                    .ENDC
                0678                    .ENDC
                0678                    .ENDC
                0678            30108$:
                0678
    5B    D6    0678                    INCL    R11                             ; get next column
59  5B    D1    067A                    CMPL    R11, R9                         ; see if last column done
    03    14    067D                    BGTR    2$
```

```
                FF13   31  067F              BRW     LOOP_2ND_SUBD           ; no, continue inner loop
                       0682
                       0682          ;+
                       0682          ; Have completed entire row.  See if it was the last row.  If not,
                       0682          ; continue with next row.
                       0682          ;-
                       0682
            04 AE      D6  0682      2$:     INCL    lower_bnd1(SP)                  ; get next row
08 AE       04 AE      D1  0685              CMPL    lower_bnd1(SP), upper_bnd1(SP)  ; see if last row done
                03     14  068A              BGTR    3$
                FF03   31  068C              BRW     LOOP_1ST_SUBD           ; no, continue outer loop
                       068F
                       04  068F      3$:     RET                             ; yes, finished
                       0690
```

```
                     0690   381 GFLOAT: $BAS$MAT_INIT G                        ; expand to gfloat operations
                     0690
                     0690       ;+
                     0690       ;       REGISTER USAGE
                     0690       ;       R0 - R8 destroyed by store routines
                     0690       ;       R9      upper bound for 2nd subscript
                     0690       ;       R10     pointer to array descriptor
                     0690       ;       R11     current value of 2nd subscript
                     0690       ;-
                     0690
                     0690       ;+
                     0690       ; Set up limits for looping through all elements
                     0690       ;-
                     0690
                     0690               .IF     IDN     G, L
                     0690               .IFT                                    ; data type is long
                     0690               MOVL    constant(AP), -(SP)             ; move constant
                     0690               .IFF                                    ; data type is not long
7E   08 AC 4EFD      0690               CVTLG   constant(AP), -(SP)     ; make constant same datatype
                     0695                                                       ;  as array, save on stack
                     0695               .ENDC
                     0695               .IF     IDN     G, D            ; if array is double
                     0695               MOVL    SF$L_SAVE_FP(FP), R0            ; pass FP to get scale
                     0695               JSB     G^BAS$$SCALE_R1                 ; get scale in R0 & R1
                     0695                                                       ; call a BLISS routine because
                     0695                                                       ; the frame offsets are only
                     0695                                                       ; defined for BLISS
                     0695               MULD2   R0, (SP)                        ; scale
                     0695               .ENDC
                     0695
                     0695       ;+
                     0695       ; Allocate data and value_desc on the stack.  This applies to both
                     0695       ; one and two dimensions.
                     0695       ;-
                     0695
     7E   7C         0695               CLRQ    -(SP)                           ; space for data
     7E   7C         0697               CLRQ    -(SP)                           ;  may be hfloat
     7E   7C         0699               CLRQ    -(SP)                           ; space for value_desc
                     069B
01   0B AA   91      069B               CMPB    DSC$B_DIMCT(R10), #1            ; determine # of subscripts
     05      13      069F               BEQLU   INIT_ONE_SUBG           ; 1 sub, go init
     15      1A      06A1               BGTRU   INIT_TWO_SUBSG          ; >=2 subs, go init
   F9A4      31      06A3               BRW     ERR_ARGDONMAT                   ; 0 subs, error
                     06A6
                     06A6       ;+
                     06A6       ; There is only 1 subscript.  Make both upper and lower bound for 2nd
                     06A6       ; subscript a 1.  The second subscript will be passed to and ignored by the
                     06A6       ; store routine.
                     06A6       ;-
                     06A6
                     06A6       INIT_ONE_SUBG:
   1C AA   DD        06A6               PUSHL   dsc$l_u1_1(R10)                 ; 1st upper bound
   18 AA   DD        06A9               PUSHL   dsc$l_l1_1(R10)                 ; 1st lower bound
     03    14        06AC               BGTR    1$                              ; not 0 or neg, do 2nd sub
6E   01   D0        06AE               MOVL    #1, (SP)                        ; don't alter col 0
59   01   D0        06B1       1$:     MOVL    #1, R9                          ; dummy 2nd lower bound
     01   DD        06B4               PUSHL   #1                              ; dummy 2nd upper bound
```

```
                1A   11  06B6                BRB     LOOP_2ND_SUBG           ; go loop
                        06B8
                        06B8        ;+
                        06B8        ; There are 2 subscripts.  Put the upper bound for both subscripts on the
                        06B8        ; stack and make sure that the lower bound for both subscripts will start
                        06B8        ; at 1 (do not alter row or col 0)
                        06B8        ;-
                        06B8
                        06B8        INIT_TWO_SUBSG:
              20 AA     DD  06B8          PUSHL   dsc$l_u1_2(R10)                 ; 1st upper bound
              1C AA     DD  06BB          PUSHL   dsc$l_l1_2(R10)                 ; 1st lower bound
                 03     14  06BE          BGTR    1$                              ; not row 0 or neg, do cols
           6E    01     D0  06C0          MOVL    #1, (SP)                        ; start with row 1
        59    28 AA     D0  06C3  1$:     MOVL    dsc$l_u2_2(R10), R9             ; 2nd upper bound
              24 AA     DD  06C7          PUSHL   dsc$l_l2_2(R10)                 ; 2nd lower bound
                 03     14  06CA          BGTR    LOOP_TST_SUBG           ; not col 0, go loop
           6E    01     D0  06CC          MOVL    #1, (SP)                        ; start with col 1
                        06CF
                        06CF        ;+
                        06CF        ; Loop through all the rows.  Row and column upper and lower bounds have been
                        06CF        ; initialized on the stack.
                        06CF        ;-
                        06CF
                        06CF        LOOP_1ST_SUBG:
           5B   6E      D0  06CF          MOVL    lower_bnd2(SP), R11             ; R11 has 2nd lower bound
                        06D2
                        06D2        ;+
                        06D2        ; Loop through all the elements (columns) of the current row.  Column lower
                        06D2        ; bound is initialized in R11.  Column upper bound is on the stack.
                        06D2        ; Distinguish array by data type so that the correct store routine can be
                        06D2        ; called and the constant can be converted to the correct type.
                        06D2        ;-
                        06D2
                        06D2        LOOP_2ND_SUBG:
                        06D2
        50    24 AE 50FD  06D2          MOVG    constant_cvt(SP), R0    ; put constant into R0
                        06D7                                            ;  R0 & R1 for double
                        06D7        ;+
                        06D7        ; When passed by value, hfloat takes 4 words, gfloat and double take 2 words,
                        06D7        ; and all other data types take 1 longword.
                        06D7        ;-
                        06D7
                        06D7            .IF     IDN     G, H            ; data type is hfloat
                        06D7            MOVL    R10, R4                         ; pointer to array desc
                        06D7            MOVL    lower_bnd1(SP), R5              ; current row
                        06D7            MOVL    R11, R6                         ; current column
                        06D7            .IFF
                        06D7            .IF     IDN     G, G            ; data type is gfloat
              52   5A   D0  06D7        MOVL    R10, R2                         ; pointer to array desc
        53    04 AE    D0  06DA        MOVL    lower_bnd1(SP), R3              ; current row
              54   5B   D0  06DE        MOVL    R11, R4                         ; current column
                        06E1            .IFF
                        06E1            .IF     IDN     G, D            ; data type is double
                        06E1            MOVL    R10, R2                         ; pointer to array desc
                        06E1            MOVL    lower_bnd1(SP), R3              ; current row
                        06E1            MOVL    R11, R4                         ; current column
                        06E1            .IFF                                    ; all other data types
```

```
                06E1                    MOVL    R10, R1                         ; pointer to array desc
                06E1                    MOVL    lower_bnd1(SP), R2              ; current row
                06E1                    MOVL    R11, R3                         ; current column
                06E1                    .ENDC
                06E1                    .ENDC
                06E1                    .ENDC
    14 AE   50 50FD 06E1                MOVG    R0, data(SP)            ; store value in value_desc
                06E6                    STORE   G                               ; store in array
                06E6                    .IF     IDN     G, H
                06E6                    CMPB    dsc$b_dtype(R4), #dsc$k_dtype_dsc
                06E6                    BNEQ    30134$
                06E6                    MOVL    4(R4), R0
                06E6                    MOVB    dsc$b_dtype(R0), dtype(SP)
                06E6                    MOVB    dsc$b_class(R0), class(SP)
                06E6                    MOVAL   data(SP), pointer (SP)
                06E6                    MOVW    #10, str_len(SP)
                06E6                    CMPB    dsc$b_dimct(R4), #1
                06E6                    BNEQ    30136$
                06E6                    PUSHL   R5
                06E6                    PUSHL   R4
                06E6                    PUSHAL  value_desc+8(SP)
                06E6                    CALLS   #3,G^BAS$STORE_BFA
                06E6                    BRW     30133$
                06E6            30136$: PUSHL   R6
                06E6                    PUSHL   R5
                06E6                    PUSHL   R4
                06E6                    PUSHAL  value_desc+12(SP)
                06E6                    CALLS   #4,G^BAS$STORE_BFA
                06E6                    BRW     30133$
                06E6            30134$: CMPB    dsc$b_class(R4), #dsc$k_class_bfa
                06E6                    BNEQ    30125$
                06E6                    JSB     G^BAS$STO_FA_G_R8
                06E6                    BRW     30133$
                06E6            30125$: BBS     #5, 10(R4), 30126$
                06E6                    CMPB    dsc$b_dimct(R4), #1
                06E6                    BNEQ    30135$
                06E6                    MOVZWL  dsc$w_length(R4), R8
                06E6                    INDEX   R5, dsc$l_l1_1(R4), dsc$l_u1_1(R4), R8, #0, R7
                06E6                    ADDL    dsc$a_a0(R4), R7
                06E6                    MOVG    R0, (R7)
                06E6                    BRW     30133$
                06E6            30135$: INDEX   R5, dsc$l_l1_2(R4), dsc$l_u1_2(R4), dsc$l_m2(R4), #0, R7
                06E6                    MOVZWL  dsc$w_length(R4), R8
                06E6                    INDEX   R6, dsc$l_l2_2(R4), dsc$l_u2_2(R4), R8, R7, R7
                06E6                    ADDL    dsc$a_a0(R4), R7
                06E6                    MOVG    R0, (R7)
                06E6                    BRW     30133$
                06E6            30126$: CMPB    dsc$b_dimct(R4), #1
                06E6                    BNEQ    30137$
                06E6                    MOVZWL  dsc$w_length(R4), R8
                06E6                    INDEX   R6, dsc$l_l1_1(R4), dsc$l_u1_1(R4), R8, #0, R7
                06E6                    ADDL    dsc$a_a0(R4), R7
                06E6                    MOVG    R0, (R7)
                06E6                    BRW     30133$
                06E6            30137$: INDEX   R6, dsc$l_l2_2(R4), dsc$l_u2_2(R4), dsc$l_m1(R4), #0, R7
                06E6                    MOVZWL  dsc$w_length(R4), R8
                06E6                    INDEX   R5, dsc$l_l1_2(R4), dsc$l_u1_2(R4), R8, R7, R7
```

```
                                06E6                    ADDL    dsc$a_a0(R4), R7
                                06E6                    MOVG    R0, (R7)
                                06E6                    .IFF
                                06E6                    .IF     IDN     G, G
          18    02 A2     91    06E6                    CMPB    dsc$b_dtype(R2), #dsc$k_dtype_dsc
                  41      12    06EA                    BNEQ    30138$
          50    04 A2     D0    06EC                    MOVL    4(R2), R0
       0E AE    02 A0     90    06F0                    MOVB    dsc$b_dtype(R0), dtype(SP)
       0F AE    03 A0     90    06F5                    MOVB    dsc$b_class(R0), class(SP)
       10 AE    14 AE     DE    06FA                    MOVAL   data(SP), pointer (SP)
          0C AE   0A       B0    06FF                    MOVW    #10, str_len(SP)
          01    0B A2     91    0703                    CMPB    dsc$b_dimct(R2), #1
                  11      12    0707                    BNEQ    30140$
                  53      DD    0709                    PUSHL   R3
                  52      DD    070B                    PUSHL   R2
                14 AE     DF    070D                    PUSHAL  value_desc+8(SP)
00000000'GF       03      FB    0710                    CALLS   #3,G^BAS$STORE_BFA
                  00A1    31    0717                    BRW     30133$
                  54      DD    071A    30140$:         PUSHL   R4
                  53      DD    071C                    PUSHL   R3
                  52      DD    071E                    PUSHL   R2
                18 AE     DF    0720                    PUSHAL  value_desc+12(SP)
00000000'GF       04      FB    0723                    CALLS   #4,G^BAS$STORE_BFA
                  008E    31    072A                    BRW     30133$
    BF 8F       03 A2     91    072D    30138$:         CMPB    dsc$b_class(R2), #dsc$k_class_bfa
                  09      12    0732                    BNEQ    30127$
          00000000'GF       16    0734                JSB     G^BAS$STO_FA_G_R8
                  007E      31    073A                BRW     30133$
             3E 0A A2       05    E0    073D    30127$:   BBS     #5, 10(R2), 30128$
                01    0B A2     91    0742              CMPB    dsc$b_dimct(R2), #1
                        17      12    0746              BNEQ    30139$
                      56        62    3C    0748        MOVZWL  dsc$w_length(R2), R6
00    56    1C A2    18 A2      53    0A    074B        INDEX   R3, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                                55          0753
                      55    10 A2       C0    0754      ADDL    dsc$a_a0(R2), R5
                         65   50    50FD    0758        MOVG    R0, (R5)
                            005C      31    075C        BRW     30133$
  18 A2    20 A2    1C A2      53    0A    075F  30139$: INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), dsc$l_m2(R2), #0, R5
                                55    00    0767
                      56        62    3C    0769        MOVZWL  dsc$w_length(R2), R6
55    56    28 A2    24 A2      54    0A    076C        INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), R6, R5, R5
                                55          0774
                      55    10 A2       C0    0775      ADDL    dsc$a_a0(R2), R5
                         65   50    50FD    0779        MOVG    R0, (R5)
                            003B      31    077D        BRW     30133$
                01    0B A2     91    0780  30128$:   CMPB    dsc$b_dimct(R2), #1
                        17      12    0784              BNEQ    30141$
                      56        62    3C    0786        MOVZWL  dsc$w_length(R2), R6
00    56    1C A2    18 A2      54    0A    0789        INDEX   R4, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                                55          0791
                      55    10 A2       C0    0792      ADDL    dsc$a_a0(R2), R5
                         65   50    50FD    0796        MOVG    R0, (R5)
                            001E      31    079A        BRW     30133$
  14 A2    28 A2    24 A2      54    0A    079D  30141$: INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), dsc$l_m1(R2), #0, R5
                                55    00    07A5
                      56        62    3C    07A7        MOVZWL  dsc$w_length(R2), R6
55    56    20 A2    1C A2      53    0A    07AA        INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), R6, R5, R5
```

```
                55        07B2
          55    10 A2  C0 07B3                ADDL    dsc$a_a0(R2), R5
             65    50 50FD 07B7               MOVG    R0, (R5)
                           07BB               .IFF
                           07BB               .IF     IDN     G, D
                           07BB               CMPB    dsc$b_dtype(R2), #dsc$k_dtype_dsc
                           07BB               BNEQ    30142$
                           07BB               MOVL    4(R2), R0
                           07BB               MOVB    dsc$b_dtype(R0), dtype(SP)
                           07BB               MOVB    dsc$b_class(R0), class(SP)
                           07BB               MOVAL   data(SP), pointer (SP)
                           07BB               MOVW    #10, str_len(SP)
                           07BB               CMPB    dsc$b_dimct(R2), #1
                           07BB               BNEQ    30144$
                           07BB               PUSHL   R3
                           07BB               PUSHL   R2
                           07BB               PUSHAL  value_desc+8(SP)
                           07BB               CALLS   #3,G^BAS$STORE_BFA
                           07BB               BRW     30133$
                           07BB       30144$: PUSHL   R4
                           07BB               PUSHL   R3
                           07BB               PUSHL   R2
                           07BB               PUSHAL  value_desc+12(SP)
                           07BB               CALLS   #4,G^BAS$STORE_BFA
                           07BB               BRW     30133$
                           07BB       30142$: CMPB    dsc$b_class(R2), #dsc$k_class_bfa
                           07BB               BNEQ    30129$
                           07BB               JSB     G^BAS$STO_FA_G_R8
                           07BB               BRW     30133$
                           07BB       30129$: BBS     #5, 10(R2), 30130$
                           07BB               CMPB    dsc$b_dimct(R2), #1
                           07BB               BNEQ    30143$
                           07BB               MOVZWL  dsc$w_length(R2), R6
                           07BB               INDEX   R3, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                           07BB               ADDL    dsc$a_a0(R2), R5
                           07BB               MOVG    R0, (R5)
                           07BB               BRW     30133$
                           07BB       30143$: INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), dsc$l_m2(R2), #0, R5
                           07BB               MOVZWL  dsc$w_length(R2), R6
                           07BB               INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), R6, R5, R5
                           07BB               ADDL    dsc$a_a0(R2), R5
                           07BB               MOVG    R0, (R5)
                           07BB               BRW     30133$
                           07BB       30130$: CMPB    dsc$b_dimct(R2), #1
                           07BB               BNEQ    30145$
                           07BB               MOVZWL  dsc$w_length(R2), R6
                           07BB               INDEX   R4, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                           07BB               ADDL    dsc$a_a0(R2), R5
                           07BB               MOVG    R0, (R5)
                           07BB               BRW     30133$
                           07BB       30145$: INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), dsc$l_m1(R2), #0, R5
                           07BB               MOVZWL  dsc$w_length(R2), R6
                           07BB               INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), R6, R5, R5
                           07BB               ADDL    dsc$a_a0(R2), R5
                           07BB               MOVG    R0, (R5)
                           07BB               .IFF
                           07BB               CMPB    dsc$b_dtype(R1), #dsc$k_dtype_dsc
```

```
                07BB                    JNEQ    30146$
                07BB                    MOVL    4(R1), R0
                07BB                    MOVB    dsc$b_dtype(R0), dtype(SP)
                07BB                    MOVB    dsc$b_class(R0), class(SP)
                07BB                    MOVAL   data(SP), pointer (SP)
                07BB                    MOVW    #10, str_len(SP)
                07BB                    CMPB    dsc$b_dimct(R1), #1
                07BB                    BNEQ    30148$
                07BB                    PUSHL   R2
                07BB                    PUSHL   R1
                07BB                    PUSHAL  value_desc+8(SP)
                07BB                    CALLS   #3,G^BAS$STORE_BFA
                07BB                    BRW     30133$
                07BB            30148$: PUSHL   R3
                07BB                    PUSHL   R2
                07BB                    PUSHL   R1
                07BB                    PUSHAL  value_desc+12(SP)
                07BB                    CALLS   #4,G^BAS$STORE_BFA
                07BB                    BRW     30133$
                07BB            30146$: CMPB    dsc$b_class(R1), #dsc$k_class_bfa
                07BB                    BNEQ    30131$
                07BB                    JSB     G^BAS$STO_FA_G_R8
                07BB                    BRW     30133$
                07BB            30131$: BBS     #5, 10(R1), 30132$
                07BB                    CMPB    dsc$b_dimct(R1), #1
                07BB                    BNEQ    30147$
                07BB                    MOVZWL  dsc$w_length(R1), R5
                07BB                    INDEX   R2, dsc$l_l1_1(R1), dsc$l_u1_1(R1), R5, #0, R4
                07BB                    ADDL    dsc$a_a0(R1), R4
                07BB                    MOVG    R0, (R4)
                07BB                    BRW     30133$
                07BB            30147$: INDEX   R2, dsc$l_l1_2(R1), dsc$l_u1_2(R1), dsc$l_m2(R1), #0, R4
                07BB                    MOVZWL  dsc$w_length(R1), R5
                07BB                    INDEX   R3, dsc$l_l2_2(R1), dsc$l_u2_2(R1), R5, R4, R4
                07BB                    ADDL    dsc$a_a0(R1), R4
                07BB                    MOVG    R0, (R4)
                07BB                    BRW     30133$
                07BB            30132$: CMPB    dsc$b_dimct(R1), #1
                07BB                    BNEQ    30149$
                07BB                    MOVZWL  dsc$w_length(R1), R5
                07BB                    INDEX   R3, dsc$l_l1_1(R1), dsc$l_u1_1(R1), R5, #0, R4
                07BB                    ADDL    dsc$a_a0(R1), R4
                07BB                    MOVG    R0, (R4)
                07BB                    BRW     30133$
                07BB            30149$: INDEX   R3, dsc$l_l2_2(R1), dsc$l_u2_2(R1), dsc$l_m1(R1), #0, R4
                07BB                    MOVZWL  dsc$w_length(R1), R5
                07BB                    INDEX   R2, dsc$l_l1_2(R1), dsc$l_u1_2(R1), R5, R4, R4
                07BB                    ADDL    dsc$a_a0(R1), R4
                07BB                    MOVG    R0, (R4)
                07BB                    .ENDC
                07BB                    .ENDC
                07BB                    .ENDC
                07BB            30133$:
                07BB
     5B   D6    07BB                    INCL    R11                             ; get next column
59   5B   D1    07BD                    CMPL    R11, R9                         ; see if last column done
     03   14    07C0                    BGTR    2$
```

```
                FF0D   31  07C2                 BRW     LOOP_2ND_SUBG           ; no, continue inner loop
                       07C5
                       07C5         ;+
                       07C5         ; Have completed entire row.  See if it was the last row.  If not,
                       07C5         ; continue with next row.
                       07C5         ;-
                       07C5
            04 AE      D6  07C5     2$:    INCL    lower_bnd1(SP)                  ; get next row
      08 AE 04 AE      D1  07C8            CMPL    lower_bnd1(SP), upper_bnd1(SP)  ; see if last row done
                03     14  07CD            BGTR    3$
                FEFD   31  07CF            BRW     LOOP_1ST_SUBG           ; no, continue outer loop
                       07D2
                04     07D2     3$:    RET                                     ; yes, finished
                       07D3
```

```
                     07D3   383 HFLOAT: $BAS$MAT_INIT H                          ; expand to hfloat operations
                     07D3
                     07D3       ;+
                     07D3       ;       REGISTER USAGE
                     07D3       ;       R0 - R8 destroyed by store routines
                     07D3       ;       R9      upper bound for 2nd subscript
                     07D3       ;       R10     pointer to array descriptor
                     07D3       ;       R11     current value of 2nd subscript
                     07D3       ;-
                     07D3
                     07D3       ;+
                     07D3       ; Set up limits for looping through all elements
                     07D3       ;-
                     07D3
                     07D3               .IF     IDN     H, L
                     07D3               .IFT                                    ; data type is long
                     07D3               MOVL    constant(AP), -(SP)             ; move constant
                     07D3               .IFF                                    ; data type is not long
                7E   08 AC 6EFD 07D3   CVTLH   constant(AP), -(SP)     ; make constant same datatype
                     07D8                                                      ;  as array, save on stack
                     07D8               .ENDC
                     07D8               .IF     IDN     H, D            ; if array is double
                     07D8               MOVL    SF$L_SAVE_FP(FP), R0            ; pass FP to get scale
                     07D8               JSB     G^BAS$$SCALE_R1                 ; get scale in R0 & R1
                     07D8                                                       ; call a BLISS routine because
                     07D8                                                       ; the frame offsets are only
                     07D8                                                       ; defined for BLISS
                     07D8               MULD2   R0, (SP)                        ; scale
                     07D8               .ENDC
                     07D8
                     07D8       ;+
                     07D8       ; Allocate data and value_desc on the stack.  This applies to both
                     07D8       ; one and two dimensions.
                     07D8       ;-
                     07D8
                7E   7C 07D8               CLRQ    -(SP)                           ; space for data
                7E   7C 07DA               CLRQ    -(SP)                           ;  may be hfloat
                7E   7C 07DC               CLRQ    -(SP)                           ; space for value_desc
                     07DE
           01   0B AA   91 07DE               CMPB    DSC$B_DIMCT(R10), #1            ; determine # of subscripts
                05   13 07E2               BEQLU   INIT_ONE_SUBH           ; 1 sub, go init
                15   1A 07E4               BGTRU   INIT_TWO_SUBSH          ; >=2 subs, go init
              F861   31 07E6               BRW     ERR_ARGDONMAT                   ; 0 subs, error
                     07E9
                     07E9       ;+
                     07E9       ; There is only 1 subscript.  Make both upper and lower bound for 2nd
                     07E9       ; subscript a 1.  The second subscript will be passed to and ignored by the
                     07E9       ; store routine.
                     07E9       ;-
                     07E9
                     07E9       INIT_ONE_SUBH:
             1C  AA   DD 07E9               PUSHL   dsc$l_u1_1(R10)                 ; 1st upper bound
             18  AA   DD 07EC               PUSHL   dsc$l_l1_1(R10)                 ; 1st lower bound
                03   14 07EF               BGTR    1$                              ; not 0 or neg, do 2nd sub
           6E   01   D0 07F1               MOVL    #1, (SP)                        ; don't alter col 0
           59   01   D0 07F4       1$:     MOVL    #1, R9                          ; dummy 2nd lower bound
                01   DD 07F7               PUSHL   #1                              ; dummy 2nd upper bound
```

```
                 1A   11  07F9                BRB     LOOP_2ND_SUBH           ; go loop
                         07FB
                         07FB        ;+
                         07FB        ; There are 2 subscripts.  Put the upper bound for both subscripts on the
                         07FB        ; stack and make sure that the lower bound for both subscripts will start
                         07FB        ; at 1 (do not alter row or col 0)
                         07FB        ;-
                         07FB
                         07FB        INIT_TWO_SUBSH:
               20 AA     DD  07FB            PUSHL   dsc$l_u1_2(R10)                 ; 1st upper bound
               1C AA     DD  07FE            PUSHL   dsc$l_l1_2(R10)                 ; 1st lower bound
                  03     14  0801            BGTR    1$                              ; not row 0 or neg, do cols
               6E 01     D0  0803            MOVL    #1, (SP)                        ; start with row 1
            59 28 AA     D0  0806    1$:     MOVL    dsc$l_u2_2(R10), R9             ; 2nd upper bound
               24 AA     DD  080A            PUSHL   dsc$l_l2_2(R10)                 ; 2nd lower bound
                  03     14  080D            BGTR    LOOP_TST_SUBH           ; not col 0, go loop
               6E 01     D0  080F            MOVL    #1, (SP)                        ; start with col 1
                         0812
                         0812        ;+
                         0812        ; Loop through all the rows.  Row and column upper and lower bounds have been
                         0812        ; initialized on the stack.
                         0812        ;-
                         0812
                         0812        LOOP_1ST_SUBH:
               5B 6E     D0  0812            MOVL    lower_bnd2(SP), R11             ; R11 has 2nd lower bound
                         0815
                         0815        ;+
                         0815        ; Loop through all the elements (columns) of the current row.  Column lower
                         0815        ; bound is initialized in R11.  Column upper bound is on the stack.
                         0815        ; Distinguish array by data type so that the correct store routine can be
                         0815        ; called and the constant can be converted to the correct type.
                         0815        ;-
                         0815
                         0815        LOOP_2ND_SUBH:
                         0815
            50 24 AE 70FD    0815            MOVH    constant_cvt(SP), R0    ; put constant into R0
                         081A                                                    ;  R0 & R1 for double
                         081A        ;+
                         081A        ; When passed by value, hfloat takes 4 words, gfloat and double take 2 words,
                         081A        ; and all other data types take 1 longword.
                         081A        ;-
                         081A
                         081A            .IF     IDN     H, H                    ; data type is hfloat
               54 5A     D0  081A        MOVL    R10, R4                         ; pointer to array desc
            55 04 AE     D0  081D        MOVL    lower_bnd1(SP), R5              ; current row
               56 5B     D0  0821        MOVL    R11, R6                         ; current column
                         0824            .IFF
                         0824            .IF     IDN     H, G                    ; data type is gfloat
                         0824        MOVL    R10, R2                         ; pointer to array desc
                         0824        MOVL    lower_bnd1(SP), R3              ; current row
                         0824        MOVL    R11, R4                         ; current column
                         0824            .IFF
                         0824            .IF     IDN     H, D                    ; data type is double
                         0824        MOVL    R10, R2                         ; pointer to array desc
                         0824        MOVL    lower_bnd1(SP), R3              ; current row
                         0824        MOVL    R11, R4                         ; current column
                         0824            .IFF                                    ; all other data types
```

```
                                0824                MOVL    R10, R1                         ; pointer to array desc
                                0824                MOVL    lower_bnd1(SP), R2              ; current row
                                0824                MOVL    R11, R3                         ; current column
                                0824                .ENDC
                                0824                .ENDC
                                0824                .ENDC
      14 AE    50 70FD          0824                MOVH    R0, data(SP)            ; store value in value_desc
                                0829                STORE   H                               ; store in array
                                0829                .IF     IDN     H, H
      18   02 A4    91          0829                CMPB    dsc$b_dtype(R4), #dsc$k_dtype_dsc
                41    12          082D                BNEQ    30159$
      50   04 A4    D0          082F                MOVL    4(R4), R0
   0E AE   02 A0    90          0833                MOVB    dsc$b_dtype(R0), dtype(SP)
   0F AE   03 A0    90          0838                MOVB    dsc$b_class(R0), class(SP)
   10 AE   14 AE    DE          083D                MOVAL   data(SP), pointer (SP)
      0C AE   0A    B0          0842                MOVW    #10, str_len(SP)
      01   0B A4    91          0846                CMPB    dsc$b_dimct(R4), #1
                11    12          084A                BNEQ    30161$
                55    DD          084C                PUSHL   R5
                54    DD          084E                PUSHL   R4
            14 AE     DF          0850                PUSHAL  value_desc+8(SP)
00000000'GF   03      FB          0853                CALLS   #3,G^BAS$STORE_BFA
             00A1     31          085A                BRW     30158$
                56    DD          085D        30161$: PUSHL   R6
                55    DD          085F                PUSHL   R5
                54    DD          0861                PUSHL   R4
            18 AE     DF          0863                PUSHAL  value_desc+12(SP)
00000000'GF   04      FB          0866                CALLS   #4,G^BAS$STORE_BFA
              008E    31          086D                BRW     30158$
   BF 8F    03 A4    91          0870        30159$: CMPB    dsc$b_class(R4), #dsc$k_class_bfa
                09    12          0875                BNEQ    30150$
          00000000'GF   16        0877                JSB     G^BAS$STO_FA_H_R8
              007E    31          087D                BRW     30158$
   3E 0A A4    05    E0          0880        30150$: BBS     #5, 10(R4), 30151$
      01   0B A4    91          0885                CMPB    dsc$b_dimct(R4), #1
                17    12          0889                BNEQ    30160$
             58     64    3C      088B                MOVZWL  dsc$w_length(R4), R8
00   58   1C A4   18 A4    55    0A  088E            INDEX   R5, dsc$l_l1_1(R4), dsc$l_u1_1(R4), R8, #0, R7
                57            0896
             57   10 A4    C0    0897                ADDL    dsc$a_a0(R4), R7
                67   50 70FD    089B                MOVH    R0, (R7)
                   005C    31    089F                BRW     30158$
  18 A4   20 A4   1C A4    55    0A  08A2    30160$: INDEX   R5, dsc$l_l1_2(R4), dsc$l_u1_2(R4), dsc$l_m2(R4), #0, R7
                57   00           08AA
             58     64    3C      08AC                MOVZWL  dsc$w_length(R4), R8
57   58   28 A4   24 A4    56    0A  08AF            INDEX   R6, dsc$l_l2_2(R4), dsc$l_u2_2(R4), R8, R7, R7
                57            08B7
             57   10 A4    C0    08B8                ADDL    dsc$a_a0(R4), R7
                67   50 70FD    08BC                MOVH    R0, (R7)
                   003B    31    08C0                BRW     30158$
                01   0B A4    91    08C3    30151$: CMPB    dsc$b_dimct(R4), #1
                17    12          08C7                BNEQ    30162$
             58     64    3C      08C9                MOVZWL  dsc$w_length(R4), R8
00   58   1C A4   18 A4    56    0A  08CC            INDEX   R6, dsc$l_l1_1(R4), dsc$l_u1_1(R4), R8, #0, R7
                57            08D4
             57   10 A4    C0    08D5                ADDL    dsc$a_a0(R4), R7
                67   50 70FD    08D9                MOVH    R0, (R7)
```

```
                     001E   31  08DD                BRW     30158$
    14 A4   28 A4    24 A4   56  0A  08E0    30162$: INDEX   R6, dsc$l_l2_2(R4), dsc$l_u2_2(R4), dsc$l_m1(R4), #0, R7
                     57      00      08E8
                     58      64  3C  08EA            MOVZWL  dsc$w_length(R4), R8
 57  58  20 A4    1C A4   55  0A  08ED                INDEX   R5, dsc$l_l1_2(R4), dsc$l_u1_2(R4), R8, R7, R7
                     57          08F5
                 57    10 A4   C0  08F6                ADDL    dsc$a_a0(R4), R7
                    67   50 70FD  08FA                MOVH    R0, (R7)
                             08FE                    .IFF
                             08FE                    .IF     IDN     H, G
                             08FE                    CMPB    dsc$b_dtype(R2), #dsc$k_dtype_dsc
                             08FE                    BNEQ    30163$
                             08FE                    MOVL    4(R2), R0
                             08FE                    MOVB    dsc$b_dtype(R0), dtype(SP)
                             08FE                    MOVB    dsc$b_class(R0), class(SP)
                             08FE                    MOVAL   data(SP), pointer (SP)
                             08FE                    MOVW    #10, str_len(SP)
                             08FE                    CMPB    dsc$b_dimct(R2), #1
                             08FE                    BNEQ    30165$
                             08FE                    PUSHL   R3
                             08FE                    PUSHL   R2
                             08FE                    PUSHAL  value_desc+8(SP)
                             08FE                    CALLS   #3,G^BAS$STORE_BFA
                             08FE                    BRW     30158$
                             08FE    30165$: PUSHL   R4
                             08FE                    PUSHL   R3
                             08FE                    PUSHL   R2
                             08FE                    PUSHAL  value_desc+12(SP)
                             08FE                    CALLS   #4,G^BAS$STORE_BFA
                             08FE                    BRW     30158$
                             08FE    30163$: CMPB    dsc$b_class(R2), #dsc$k_class_bfa
                             08FE                    BNEQ    30152$
                             08FE                    JSB     G^BAS$STO_FA_H_R8
                             08FE                    BRW     30158$
                             08FE    30152$: BBS     #5, 10(R2), 30153$
                             08FE                    CMPB    dsc$b_dimct(R2), #1
                             08FE                    BNEQ    30164$
                             08FE                    MOVZWL  dsc$w_length(R2), R6
                             08FE                    INDEX   R3, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                             08FE                    ADDL    dsc$a_a0(R2), R5
                             08FE                    MOVH    R0, (R5)
                             08FE                    BRW     30158$
                             08FE    30164$: INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), dsc$l_m2(R2), #0, R5
                             08FE                    MOVZWL  dsc$w_length(R2), R6
                             08FE                    INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), R6, R5, R5
                             08FE                    ADDL    dsc$a_a0(R2), R5
                             08FE                    MOVH    R0, (R5)
                             08FE                    BRW     30158$
                             08FE    30153$: CMPB    dsc$b_dimct(R2), #1
                             08FE                    BNEQ    30166$
                             08FE                    MOVZWL  dsc$w_length(R2), R6
                             08FE                    INDEX   R4, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                             08FE                    ADDL    dsc$a_a0(R2), R5
                             08FE                    MOVH    R0, (R5)
                             08FE                    BRW     30158$
                             08FE    30166$: INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), dsc$l_m1(R2), #0, R5
                             08FE                    MOVZWL  dsc$w_length(R2), R6
```

```
                08FE                    INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), R6, R5, R5
                08FE                    ADDL    dsc$a_a0(R2), R5
                08FE                    MOVH    R0, (R5)
                08FE                    .IFF
                08FE                    .IF     IDN     H, D
                08FE                    CMPB    dsc$b_dtype(R2), #dsc$k_dtype_dsc
                08FE                    BNEQ    30167$
                08FE                    MOVL    4(R2), R0
                08FE                    MOVB    dsc$b_dtype(R0), dtype(SP)
                08FE                    MOVB    dsc$b_class(R0), class(SP)
                08FE                    MOVAL   data(SP), pointer (SP)
                08FE                    MOVW    #10, str_len(SP)
                08FE                    CMPB    dsc$b_dimct(R2), #1
                08FE                    BNEQ    30169$
                08FE                    PUSHL   R3
                08FE                    PUSHL   R2
                08FE                    PUSHAL  value_desc+8(SP)
                08FE                    CALLS   #3,G^BAS$STORE_BFA
                08FE                    BRW     30158$
                08FE            30169$: PUSHL   R4
                08FE                    PUSHL   R3
                08FE                    PUSHL   R2
                08FE                    PUSHAL  value_desc+12(SP)
                08FE                    CALLS   #4,G^BAS$STORE_BFA
                08FE                    BRW     30158$
                08FE            30167$: CMPB    dsc$b_class(R2), #dsc$k_class_bfa
                08FE                    BNEQ    30154$
                08FE                    JSB     G^BAS$STO_FA_H_R8
                08FE                    BRW     30158$
                08FE            30154$: BBS     #5, 10(R2), 30155$
                08FE                    CMPB    dsc$b_dimct(R2), #1
                08FE                    BNEQ    30168$
                08FE                    MOVZWL  dsc$w_length(R2), R6
                08FE                    INDEX   R3, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                08FE                    ADDL    dsc$a_a0(R2), R5
                08FE                    MOVH    R0, (R5)
                08FE                    BRW     30158$
                08FE            30168$: INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), dsc$l_m2(R2), #0, R5
                08FE                    MOVZWL  dsc$w_length(R2), R6
                08FE                    INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), R6, R5, R5
                08FE                    ADDL    dsc$a_a0(R2), R5
                08FE                    MOVH    R0, (R5)
                08FE                    BRW     30158$
                08FE            30155$: CMPB    dsc$b_dimct(R2), #1
                08FE                    BNEQ    30170$
                08FE                    MOVZWL  dsc$w_length(R2), R6
                08FE                    INDEX   R4, dsc$l_l1_1(R2), dsc$l_u1_1(R2), R6, #0, R5
                08FE                    ADDL    dsc$a_a0(R2), R5
                08FE                    MOVH    R0, (R5)
                08FE                    BRW     30158$
                08FE            30170$: INDEX   R4, dsc$l_l2_2(R2), dsc$l_u2_2(R2), dsc$l_m1(R2), #0, R5
                08FE                    MOVZWL  dsc$w_length(R2), R6
                08FE                    INDEX   R3, dsc$l_l1_2(R2), dsc$l_u1_2(R2), R6, R5, R5
                08FE                    ADDL    dsc$a_a0(R2), R5
                08FE                    MOVH    R0, (R5)
                08FE                    .IFF
                08FE                    CMPB    dsc$b_dtype(R1), #dsc$k_dtype_dsc
```

```
                08FE                BNEQ    30171$
                08FE                MOVL    4(R1), R0
                08FE                MOVB    dsc$b_dtype(R0), dtype(SP)
                08FE                MOVB    dsc$b_class(R0), class(SP)
                08FE                MOVAL   data(SP), pointer (SP)
                08FE                MOVW    #10, str_len(SP)
                08FE                CMPB    dsc$b_dimct(R1), #1
                08FE                BNEQ    30173$
                08FE                PUSHL   R2
                08FE                PUSHL   R1
                08FE                PUSHAL  value_desc+8(SP)
                08FE                CALLS   #3,G^BAS$STORE_BFA
                08FE                BRW     30158$
                08FE        30173$: PUSHL   R3
                08FE                PUSHL   R2
                08FE                PUSHL   R1
                08FE                PUSHAL  value_desc+12(SP)
                08FE                CALLS   #4,G^BAS$STORE_BFA
                08FE                BRW     30158$
                08FE        30171$: CMPB    dsc$b_class(R1), #dsc$k_class_bfa
                08FE                BNEQ    30156$
                08FE                JSB     G^BAS$STO_FA_H_R8
                08FE                BRW     30158$
                08FE        30156$: BBS     #5, 10(R1), 30157$
                08FE                CMPB    dsc$b_dimct(R1), #1
                08FE                BNEQ    30172$
                08FE                MOVZWL  dsc$w_length(R1), R5
                08FE                INDEX   R2, dsc$l_l1_1(R1), dsc$l_u1_1(R1), R5, #0, R4
                08FE                ADDL    dsc$a_a0(R1), R4
                08FE                MOVH    R0, (R4)
                08FE                BRW     30158$
                08FE        30172$: INDEX   R2, dsc$l_l1_2(R1), dsc$l_u1_2(R1), dsc$l_m2(R1), #0, R4
                08FE                MOVZWL  dsc$w_length(R1), R5
                08FE                INDEX   R3, dsc$l_l2_2(R1), dsc$l_u2_2(R1), R5, R4, R4
                08FE                ADDL    dsc$a_a0(R1), R4
                08FE                MOVH    R0, (R4)
                08FE                BRW     30158$
                08FE        30157$: CMPB    dsc$b_dimct(R1), #1
                08FE                BNEQ    30174$
                08FE                MOVZWL  dsc$w_length(R1), R5
                08FE                INDEX   R3, dsc$l_l1_1(R1), dsc$l_u1_1(R1), R5, #0, R4
                08FE                ADDL    dsc$a_a0(R1), R4
                08FE                MOVH    R0, (R4)
                08FE                BRW     30158$
                08FE        30174$: INDEX   R3, dsc$l_l2_2(R1), dsc$l_u2_2(R1), dsc$l_m1(R1), #0, R4
                08FE                MOVZWL  dsc$w_length(R1), R5
                08FE                INDEX   R2, dsc$l_l1_2(R1), dsc$l_u1_2(R1), R5, R4, R4
                08FE                ADDL    dsc$a_a0(R1), R4
                08FE                MOVH    R0, (R4)
                08FE                .ENDC
                08FE                .ENDC
                08FE                .ENDC
                08FE        30158$:
                08FE
     5B   D6    08FE                INCL    R11                             ; get next column
59   5B   D1    0900                CMPL    R11, R9                         ; see if last column done
     03   14    0903                BGTR    2$
```

```
                FF0D   31  0905                 BRW     LOOP_2ND_SUBH           ; no, continue inner loop
                           0908
                           0908         ;+
                           0908         ; Have completed entire row.  See if it was the last row.  If not,
                           0908         ; continue with next row.
                           0908         ;-
                           0908
             04 AE   D6    0908         2$:     INCL    lower_bnd1(SP)                  ; get next row
      08 AE  04 AE   D1    090B                 CMPL    lower_bnd1(SP), upper_bnd1(SP)  ; see if last row done
                03   14    0910                 BGTR    3$
                FEFD   31  0912                 BRW     LOOP_1ST_SUBH           ; no, continue outer loop
                           0915
                       04  0915         3$:     RET                             ; yes, finished
                           0916
                           0916  384             .END                            ; end of BAS$MAT_INIT
```

```
BAS$$SCALE_R1                   ********    X   00
BAS$$STOP                       ********    X   00
BAS$K_ARGDONMAT                 ********    X   00
BAS$K_DATTYPERR                 ********    X   00
BAS$MAT_INIT                    00000000 RG     02
BAS$STORE_BFA                   ********    X   00
BAS$STO_FA_B_R8                 ********    X   00
BAS$STO_FA_D_R8                 ********    X   00
BAS$STO_FA_F_R8                 ********    X   00
BAS$STO_FA_G_R8                 ********    X   00
BAS$STO_FA_H_R8                 ********    X   00
BAS$STO_FA_L_R8                 ********    X   00
BAS$STO_FA_W_R8                 ********    X   00
BYTE                            00000057 R      02
CLASS                       =   0000000F
CONSTANT                    =   00000008
CONSTANT_CVT                =   00000024
DATA                        =   00000014
DOUBLE                          00000547 R      02
DSC$A_AO                    =   00000010
DSC$B_AFLAGS                =   0000000A
DSC$B_CLASS                 =   00000003
DSC$B_DIMCT                 =   0000000B
DSC$B_DTYPE                 =   00000002
DSC$K_CLASS_BFA             =   000000BF
DSC$K_DTYPE_B               =   00000006
DSC$K_DTYPE_D               =   0000000B
DSC$K_DTYPE_DSC             =   00000018
DSC$K_DTYPE_G               =   0000001B
DSC$K_DTYPE_H               =   0000001C
DSC$L_L1_1                  =   00000018
DSC$L_L1_2                  =   0000001C
DSC$L_L2_2                  =   00000024
DSC$L_M1                    =   00000014
DSC$L_M2                    =   00000018
DSC$L_U1_1                  =   0000001C
DSC$L_U1_2                  =   00000020
DSC$L_U2_2                  =   00000028
DSC$V_FL_BOUNDS             =   00000007
DSC$W_LENGTH                =   00000000
DTYPE                       =   0000000E
ERR_ARGDONMAT                   0000004A R      02
ERR_DATTYPERR                   0000003D R      02
FLOAT                           0000040B R      02
GFLOAT                          00000690 R      02
HFLOAT                          000007D3 R      02
INIT_ONE_SUBB                   0000006C R      02
INIT_ONE_SUBD                   00000569 R      02
INIT_ONE_SUBF                   00000420 R      02
INIT_ONE_SUBG                   000006A6 R      02
INIT_ONE_SUBH                   000007E9 R      02
INIT_ONE_SUBL                   000002E4 R      02
INIT_ONE_SUBW                   000001A8 R      02
INIT_TWO_SUBSB                  0000007E R      02
INIT_TWO_SUBSD                  0000057B R      02
INIT_TWO_SUBSF                  00000432 R      02
INIT_TWO_SUBSG                  000006B8 R      02
INIT_TWO_SUBSH                  000007FB R      02
INIT_TWO_SUBSL                  000002F6 R      02
INIT_TWO_SUBSW                  000001BA R      02
LONG                            000002CF R      02
LOOP_1ST_SUBB                   00000095 R      02
LOOP_1ST_SUBD                   00000592 R      02
LOOP_1ST_SUBF                   00000449 R      02
LOOP_1ST_SUBG                   000006CF R      02
LOOP_1ST_SUBH                   00000812 R      02
LOOP_1ST_SUBL                   0000030D R      02
LOOP_1ST_SUBW                   000001D1 R      02
LOOP_2ND_SUBB                   00000098 R      02
LOOP_2ND_SUBD                   00000595 R      02
LOOP_2ND_SUBF                   0000044C R      02
LOOP_2ND_SUBG                   000006D2 R      02
LOOP_2ND_SUBH                   00000815 R      02
LOOP_2ND_SUBL                   00000310 R      02
LOOP_2ND_SUBW                   000001D4 R      02
LOWER_BND1                  =   00000004
LOWER_BND2                  =   00000000
MATRIX                      =   00000004
POINTER                     =   00000010
SF$L_SAVE_FP                =   0000000C
STR_LEN                     =   0000000C
UPPER_BND1                  =   00000008
VALUE_DESC                  =   0000000C
WORD                            00000193 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	00000916 (2326.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.05	00:00:00.55
Command processing	114	00:00:00.57	00:00:02.69
Pass 1	374	00:00:07.49	00:00:15.89
Symbol table sort	0	00:00:00.36	00:00:00.40
Pass 2	406	00:00:03.56	00:00:08.40
Symbol table output	1	00:00:00.08	00:00:00.09
Psect synopsis output	1	00:00:00.02	00:00:00.05
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	928	00:00:12.13	00:00:28.07

The working set limit was 1800 pages.
107967 bytes (211 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 228 non-local and 81 local symbols.
384 source lines were read in Pass 1, producing 18 object records in Pass 2.
30 pages of virtual memory were used to define 10 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[BASRTL.OBJ]BASRTL.MLB;1	1
_255DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	6

436 GETS were required to define 6 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS$:BASMATINI/OBJ=OBJ$:BASMATINI MSRC$:BASMATINI/UPDATE=(ENH$:BASMATINI)+LI

BASMATINT
LIS

BASMATION
LIS

BASMATASS
LIS

BASMATMUL
LIS

BASMATIO
LIS

BASMATRED
LIS

BASMATNUL
LIS

BASMATINV
LIS