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

```
BBBBBBBB      AAAAAA      SSSSSSSS  RRRRRRRR    EEEEEEEEEE  MM      MM    AAAAAA    PPPPPPPP
BBBBBBBB      AAAAAA      SSSSSSSS  RRRRRRRR    EEEEEEEEEE  MM      MM    AAAAAA    PPPPPPPP
BB      BB  AA      AA  SS        RR      RR  EE          MMMM  MMMM  AA      AA  PP      PP
BB      BB  AA      AA  SS        RR      RR  EE          MMMM  MMMM  AA      AA  PP      PP
BB      BB  AA      AA  SS        RR      RR  EE          MM  MM  MM  AA      AA  PP      PP
BB      BB  AA      AA  SS        RR      RR  EE          MM  MM  MM  AA      AA  PP      PP
BBBBBBBB    AA      AA    SSSSSS  RRRRRRRR    EEEEEEEE    MM      MM  AA      AA  PPPPPPPP
BBBBBBBB    AA      AA    SSSSSS  RRRRRRRR    EEEEEEEE    MM      MM  AA      AA  PPPPPPPP
BB      BB  AAAAAAAAAA          SS  RR  RR      EE        MM      MM  AAAAAAAAAA  PP
BB      BB  AAAAAAAAAA          SS  RR  RR      EE        MM      MM  AAAAAAAAAA  PP
BB      BB  AA      AA          SS  RR    RR    EE        MM      MM  AA      AA  PP
BB      BB  AA      AA          SS  RR    RR    EE        MM      MM  AA      AA  PP
BBBBBBBB    AA      AA  SSSSSSSS    RR      RR  EEEEEEEEEE  MM      MM  AA      AA  PP
BBBBBBBB    AA      AA  SSSSSSSS    RR      RR  EEEEEEEEEE  MM      MM  AA      AA  PP

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
```

```
;  1          0001  0 %TITLE 'BAS$REMAP_ARRAY - Remap an array'
;  2          0002  0 MODULE BAS$REMAP_ARRAY (                        ! Remap an array
;  3          0003  0                         IDENT = '1-010'         ! File: BASREMAP.B32 Edit: PLL1010
;  4          0004  0                         ) =
;  5          0005  1 BEGIN
;  6          0006  1
;  7          0007  1 !****************************************************************************
;  8          0008  1 !*                                                                          *
;  9          0009  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
; 10          0010  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
; 11          0011  1 !*  ALL RIGHTS RESERVED.                                                    *
; 12          0012  1 !*                                                                          *
; 13          0013  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
; 14          0014  1 !*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE          *
; 15          0015  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER     *
; 16          0016  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
; 17          0017  1 !*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
; 18          0018  1 !*  TRANSFERRED.                                                            *
; 19          0019  1 !*                                                                          *
; 20          0020  1 !*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE    *
; 21          0021  1 !*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT       *
; 22          0022  1 !*  CORPORATION.                                                            *
; 23          0023  1 !*                                                                          *
; 24          0024  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS     *
; 25          0025  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
; 26          0026  1 !*                                                                          *
; 27          0027  1 !*                                                                          *
; 28          0028  1 !****************************************************************************
; 29          0029  1 !
; 30          0030  1
; 31          0031  1 !++
; 32          0032  1 ! FACILITY:     Basic Language Support
; 33          0033  1 !
; 34          0034  1 ! ABSTRACT:
; 35          0035  1 !
; 36          0036  1 !       This routine is called by the compiled code to remap an array.
; 37          0037  1 !       The array will be an array of descriptors, since all dynamic
; 38          0038  1 !       variables are stored as descriptors.
; 39          0039  1 !
; 40          0040  1 ! ENVIRONMENT:  Runs at any access mode - AST reentrant
; 41          0041  1 !
; 42          0042  1 ! AUTHOR: Pamela L. Levesque, CREATION DATE: 1-Mar-1982
; 43          0043  1 !
; 44          0044  1 ! MODIFIED BY:
; 45          0045  1 !
; 46          0046  1 ! 1-001 - Original.  PLL 1-Mar-1982
; 47          0047  1 ! 1-002 - Make FETCH_DESC a separate module.  PLL 2-Mar-82
; 48          0048  1 ! 1-003 - Correct calculation of length of decimal values.  PLL 15-Mar-1982
; 49          0049  1 ! 1-004 - Make sure a length is passed for records.  PLL 16-Mar-1982
; 50          0050  1 ! 1-005 - Make routine global.  PLL 17-Mar-1982
; 51          0051  1 ! 1-006 - BAS$K_FATINTERR should be OTS$_FATINTERR.  PLL 18-Mar-1982
; 52          0052  1 ! 1-007 - Always use the length in the descriptor for records.  PLL 12-Apr-1982
; 53          0053  1 ! 1-008 - Add support for multi dimensioned arrays.  PLL 21-May-1982
; 54          0054  1 ! 1-009 - Write the updated buffer pointer into the buffer descriptor.  PLL 28-Jun-1982
; 55          0055  1 ! 1-010 - Update the length in the buffer descriptor also.  PLL 29-Jun-1982
; 56          0056  1 !--
; 57          0057  1
```

```
;   59          0058  1 %SBTTL 'Declarations'
;   60          0059  1 !
;   61          0060  1 ! SWITCHES:
;   62          0061  1 !
;   63          0062  1
;   64          0063  1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;   65          0064  1
;   66          0065  1 !
;   67          0066  1 ! LINKAGES:
;   68          0067  1 !
;   69          0068  1 !       NONE
;   70          0069  1 !
;   71          0070  1 ! TABLE OF CONTENTS:
;   72          0071  1 !
;   73          0072  1
;   74          0073  1 FORWARD ROUTINE
;   75          0074  1     BAS$REMAP_ARRAY : NOVALUE;                  ! Remap an array
;   76          0075  1
;   77          0076  1 !
;   78          0077  1 ! INCLUDE FILES:
;   79          0078  1 !
;   80          0079  1
;   81          0080  1 LIBRARY 'RTLSTARLE';                            ! System symbols, typically from SYS$LIBRARY:STARLET.L32
;   82          0081  1
;   83          0082  1 REQUIRE 'RTLIN:RTLPSECT';                       ! Define PSECT declarations macros
;   84          0177  1
;   85          0178  1 !
;   86          0179  1 ! MACROS:
;   87          0180  1 !
;   88          0181  1 !       NONE
;   89          0182  1 !
;   90          0183  1 ! EQUATED SYMBOLS:
;   91          0184  1 !
;   92          0185  1 !       NONE
;   93          0186  1 !
;   94          0187  1 ! FIELDS:
;   95          0188  1 !
;   96          0189  1 !       NONE
;   97          0190  1 !
;   98          0191  1 ! PSECTS:
;   99          0192  1 !
;  100          0193  1 DECLARE_PSECTS (BAS);                           ! Declare PSECTs for BAS$ facility
;  101          0194  1 !
;  102          0195  1 ! OWN STORAGE:
;  103          0196  1 !
;  104          0197  1 !       NONE
;  105          0198  1 !
;  106          0199  1 ! EXTERNAL REFERENCES:
;  107          0200  1 !
;  108          0201  1
;  109          0202  1 EXTERNAL ROUTINE
;  110          0203  1     BAS$$STOP : NOVALUE,                        ! Signal fatal basic error
;  111          0204  1     LIB$STOP : NOVALUE;                         ! Signal fatal error
;  112          0205  1
;  113          0206  1 EXTERNAL LITERAL                                ! Condition value symbols
;  114          0207  1     BAS$K_REMOVEBUF : UNSIGNED (8);             ! REMAP overflows buffer
;  115          0208  1
```

```
;  117          0209  1 %SBTTL 'BAS$REMAP_ARRAY - Remap an array'
;  118          0210  1 GLOBAL ROUTINE BAS$REMAP_ARRAY (                          ! Remap an array
;  119          0211  1         BUFFER,                                           ! buffer desc
;  120          0212  1         ARRAY,                                            ! array desc
;  121          0213  1         LENGTH                                            ! length for strings or records
;  122          0214  1     ) : NOVALUE =
;  123          0215  1
;  124          0216  1 !++
;  125          0217  1 ! FUNCTIONAL DESCRIPTION:
;  126          0218  1 !
;  127          0219  1 !       This routine is called by the compiled code to remap an array of
;  128          0220  1 !       descriptors.  'Remapping' an array involves updating the pointer
;  129          0221  1 !       field in the descriptor, and the length field for strings or
;  130          0222  1 !       records.
;  131          0223  1 !
;  132          0224  1 ! CALLING SEQUENCE:
;  133          0225  1 !
;  134          0226  1 !       BAS$REMAP_ARRAY (buffer.rx.ds, array.mx.da, length.rl.v)
;  135          0227  1 !
;  136          0228  1 ! FORMAL PARAMETERS:
;  137          0229  1 !
;  138          0230  1 !       buffer          addr of desc for MAP buffer
;  139          0231  1 !       array           addr of array desc
;  140          0232  1 !       length          longword length for strings or records
;  141          0233  1 !                       (-1 for default length, 16, for strings)
;  142          0234  1 !
;  143          0235  1 ! IMPLICIT INPUTS:
;  144          0236  1 !
;  145          0237  1 !       NONE
;  146          0238  1 !
;  147          0239  1 ! IMPLICIT OUTPUTS:
;  148          0240  1 !
;  149          0241  1 !       NONE
;  150          0242  1 !
;  151          0243  1 ! COMPLETION STATUS: (or ROUTINE VALUE:)
;  152          0244  1 !
;  153          0245  1 !       NONE
;  154          0246  1 !
;  155          0247  1 ! SIDE EFFECTS:
;  156          0248  1 !
;  157          0249  1 !       Will signal if an error occurs
;  158          0250  1 !
;  159          0251  1 !--
;  160          0252  1
;  161          0253  2     BEGIN
;  162          0254  2
;  163          0255  2     MAP
;  164          0256  2         BUFFER : REF BLOCK [8, BYTE],                   ! buffer desc
;  165          0257  2         ARRAY : REF BLOCK [,BYTE];                      ! array desc
;  166          0258  2
;  167          0259  2     LOCAL
;  168          0260  2         END_ADDR,                                       ! addr of last array element
;  169          0261  2         MAX_BUF_ADDR;                                   ! max addr in buffe
;  170          0262  2
;  171          0263  2
;  172          0264  2 !+
;  173          0265  2 ! Compute the largest possible address in the buffer.
```

```
;  174          0266  2 !-
;  175          0267  2
;  176          0268  2     MAX_BUF_ADDR = .BUFFER [DSC$A_POINTER] + .BUFFER [DSC$W_LENGTH];
;  177          0269  2
;  178          0270  2 !+
;  179          0271  2 ! Loop through the elements of the array.  Update the pointer and length, if
;  180          0272  2 ! necessary, of each element.  Give an error if the maximum size of the MAP
;  181          0273  2 ! buffer is exceeded.
;  182          0274  2 !-
;  183          0275  2
;  184          0276  2     END_ADDR = .ARRAY [DSC$A_POINTER] + .ARRAY [DSC$L_ARSIZE] - .ARRAY [DSC$W_LENGTH];
;  185          0277  2     INCR VALUE_DESCRIP FROM .ARRAY [DSC$A_POINTER] TO .END_ADDR
;  186          0278  2         BY .ARRAY [DSC$W_LENGTH] DO
;  187          0279  3         BEGIN
;  188          0280  3         MAP
;  189          0281  3             VALUE_DESCRIP : REF BLOCK [8, BYTE];
;  190          0282  3
;  191          0283  3         VALUE_DESCRIP [DSC$A_POINTER] = .BUFFER [DSC$A_POINTER];
;  192          0284  3         IF .VALUE_DESCRIP [DSC$B_DTYPE] EQL DSC$K_DTYPE_T
;  193          0285  3         THEN                        ! set length for strings
;  194          0286  4             VALUE_DESCRIP [DSC$W_LENGTH] = (IF .LENGTH LSS 0 THEN 16
;  195          0287  3                                             ELSE .LENGTH);
;  196          0288  3 !+
;  197          0289  3 ! Update pointer into buffer to reflect space that has been 'used'.
;  198          0290  3 !-
;  199          0291  3
;  200          0292  3         IF .VALUE_DESCRIP [DSC$B_DTYPE] NEQ DSC$K_DTYPE_P
;  201          0293  3         THEN
;  202          0294  4             BEGIN
;  203          0295  4             BUFFER [DSC$A_POINTER] = .BUFFER [DSC$A_POINTER] + .VALUE_DESCRIP [DSC$W_LENGTH];
;  204          0296  4             BUFFER [DSC$W_LENGTH] = .BUFFER [DSC$W_LENGTH] - .VALUE_DESCRIP [DSC$W_LENGTH];
;  205          0297  4             END
;  206          0298  3         ELSE
;  207          0299  4             BEGIN
;  208          0300  4             LOCAL
;  209          0301  4                 LEN;
;  210          0302  4
;  211          0303  4             LEN = .VALUE_DESCRIP [DSC$W_LENGTH]/2 + 1;
;  212          0304  4             BUFFER [DSC$A_POINTER] = .BUFFER [DSC$A_POINTER] + .LEN;
;  213          0305  4             BUFFER [DSC$W_LENGTH] = .BUFFER [DSC$W_LENGTH] - .LEN;
;  214          0306  3             END
;  215          0307  3         IF .BUFFER [DSC$A_POINTER] GTRU .MAX_BUF_ADDR
;  216          0308  3         THEN
;  217          0309  3             BAS$$STOP (BAS$K_REMOVEBUF);
;  218          0310  2         END;
;  219          0311  2
;  220          0312  1     END;                                    ! End of routine BAS$REMAP_ARRAY


;

                                                .TITLE  BAS$REMAP_ARRAY BAS$REMAP_ARRAY - Remap an arra
                                                                                y
                                                .IDENT  \1-010\

                                                .EXTRN  BAS$$STOP, LIB$STOP
                                                .EXTRN  BAS$K_REMOVEBUF

                                                .PSECT  _BAS$CODE,NOWRT,  SHR,  PIC,2
```

```
                                 00FC 00000          .ENTRY  BAS$REMAP_ARRAY, Save R2,R3,R4,R5,R6,R7      ; 0210
                          53     04 AC  D0 00002     MOVL    BUFFER, R3                                   ; 0268
                          54     04 A3  9E 00006     MOVAB   4(R3), R4                                    ;
                          56        63  3C 0000A     MOVZWL  (R3), MAX_BUF_ADDR                           ;
                          56        64  C0 0000D     ADDL2   (R4), MAX_BUF_ADDR                           ;
                          50     08 AC  D0 00010     MOVL    ARRAY, R0                                    ; 0276
                    51    04 A0     0C A0  C1 00014     ADDL3   12(R0), 4(R0), R1                        ;
                          55        60  3C 0001A     MOVZWL  (R0), R5                                     ;
                    57            51     55  C3 0001D     SUBL3   R5, R1, END_ADDR                         ;
                          52     04 A0  D0 00021     MOVL    4(R0), VALUE_DESCRIP                         ; 0307
                                    4D  11 00025     BRB     8$                                           ;
                       04 A2           64  D0 00027 1$:     MOVL    (R4), 4(VALUE_DESCRIP)               ; 0283
                          0E     02 A2  91 0002B     CMPB    2(VALUE_DESCRIP), #14                        ; 0284
                                    11  12 0002F     BNEQ    4$                                           ;
                          0C     AC  D5 00031     TSTL    LENGTH                                       ; 0286
                                    05  18 00034     BGEQ    2$                                           ;
                          50        10  D0 00036     MOVL    #16, R0                                      ;
                                    04  11 00039     BRB     3$                                           ;
                          50     0C AC  D0 0003B 2$:  MOVL    LENGTH, R0                                   ; 0287
                          62        50  B0 0003F 3$:  MOVW    R0, (VALUE_DESCRIP)                          ; 0286
                          15     02 A2  91 00042 4$:  CMPB    2(VALUE_DESCRIP), #21                        ; 0292
                                    0B  13 00046     BEQL    5$                                           ;
                          50        62  3C 00048     MOVZWL  (VALUE_DESCRIP), R0                          ; 0295
                          64        50  C0 0004B     ADDL2   R0, (R4)                                     ;
                          63        62  A2 0004E     SUBW2   (VALUE_DESCRIP), (R3)                        ; 0296
                                    0E  11 00051     BRB     6$                                           ; 0292
                          50        62  3C 00053 5$:  MOVZWL  (VALUE_DESCRIP), R0                          ; 0303
                          50        02  C6 00056     DIVL2   #2, R0                                       ;
                                    50  D6 00059     INCL    LEN                                          ;
                          64        50  C0 0005B     ADDL2   LEN, (R4)                                    ; 0304
                          63        50  A2 0005E     SUBW2   LEN, (R3)                                    ; 0305
                          56        64  D1 00061 6$:  CMPL    (R4), MAX_BUF_ADDR                           ; 0307
                                    0B  1B 00064     BLEQU   7$                                           ;
                    ,'    7E     00G 8F  9A 00066     MOVZBL  #BAS$K_REMOVEBUF, -(SP)                      ; 0309
                00000000G 00           01  FB 0006A     CALLS   #1, BAS$$STOP                            ;
                          52        55  C0 00071 7$:  ADDL2   R5, VALUE_DESCRIP                          ; 0277
                          57        52  D1 00074 8$:  CMPL    VALUE_DESCRIP, END_ADDR                      ;
                                    AE  15 00077     BLEQ    1$                                           ;
                                        04 00079     RET                                                  ; 0312

; Routine Size:  122 bytes,    Routine Base:  _BAS$CODE + 0000

;  221          0313  1 !<BLF/PAGE>
```

```
;  223          0314  1 END                                        ! End of module BAS$REMAP_ARRAY
;  224          0315  1
;  225          0316  0 ELUDOM
```

```
;                                  PSECT SUMMARY
;
;      Name                      Bytes                           Attributes
;
;  _BAS$CODE                        122  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
```

```
;                                Library Statistics
;
;                                          -------- Symbols --------      Pages       Processing
;      File                                Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1        9776         6         0       581          00:01.0
```

```
;                                     COMMAND QUALIFIERS

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

; Size:          122 code + 0 data bytes
; Run Time:        00:05.3
; Elapsed Time:    00:15.4
; Lines/CPU Min:    3577
; Lexemes/CPU-Min: 18588
; Memory Used:  69 pages
; Compilation Complete
```

BASRSTSUB
LIS

BASRAD50
LIS

BASRSET
LIS

BASPUT
LIS

BASRECPRO
LIS

BASRESTAR
LIS

BASRANDOM
LIS

BASREMAP
LIS

BASRESTOR
LIS

BASRIGHT
LIS