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

FILEID**BASMOVEAR

```
BBBBBBBB      AAAAAA      SSSSSSSS  MM      MM    OOOOOO    VV      VV  EEEEEEEEEE    AAAAAA    RRRRRRRR
BBBBBBBB      AAAAAA      SSSSSSSS  MM      MM    OOOOOO    VV      VV  EEEEEEEEEE    AAAAAA    RRRRRRRR
BB      BB  AA      AA  SS          MMMM  MMMM  OO      OO  VV      VV  EE          AA      AA  RR      RR
BB      BB  AA      AA  SS          MMMM  MMMM  OO      OO  VV      VV  EE          AA      AA  RR      RR
BB      BB  AA      AA  SS          MM  MM  MM  OO      OO  VV      VV  EE          AA      AA  RR      RR
BB      BB  AA      AA  SS          MM  MM  MM  OO      OO  VV      VV  EE          AA      AA  RR      RR
BBBBBBBB    AA      AA    SSSSSS    MM      MM  OO      OO  VV      VV  EEEEEEEE    AA      AA  RRRRRRRR
BBBBBBBB    AA      AA    SSSSSS    MM      MM  OO      OO  VV      VV  EEEEEEEE    AA      AA  RRRRRRRR
BB      BB  AAAAAAAAAA          SS  MM      MM  OO      OO  VV      VV  EE          AAAAAAAAAA  RR  RR
BB      BB  AAAAAAAAAA          SS  MM      MM  OO      OO  VV      VV  EE          AAAAAAAAAA  RR  RR
BB      BB  AA      AA          SS  MM      MM  OO      OO    VV  VV    EE          AA      AA  RR    RR        ....
BB      BB  AA      AA          SS  MM      MM  OO      OO    VV  VV    EE          AA      AA  RR    RR        ....
BBBBBBBB    AA      AA  SSSSSSSS    MM      MM    OOOOOO        VV      EEEEEEEEEE  AA      AA  RR      RR      ....
BBBBBBBB    AA      AA  SSSSSSSS    MM      MM    OOOOOO        VV      EEEEEEEEEE  AA      AA  RR      RR      ....

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
```

```
0 MODULE BAS$MOVE_ARRAY (
0               IDENT = '1-020'          ! File: BASMOVEAR.B32 Edit: MDL1020
0               ) =
1 BEGIN
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                *
1 !*   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                 *
1 !*   ALL RIGHTS RESERVED.                                                   *
1 !*                                                                          *
1 !*   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
1 !*   ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
1 !*   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
1 !*   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
1 !*   OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
1 !*   TRANSFERRED.                                                           *
1 !*                                                                          *
1 !*   THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
1 !*   AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
1 !*   CORPORATION.                                                           *
1 !*                                                                          *
1 !*   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
1 !*   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! FACILITY:  BASIC-PLUS-2 Miscellaneous I/O
1 !
1 ! ABSTRACT:
1 !
1 !       This module contains the routines called by compiled code
1 !       for the MOVE FROM and MOVE TO statements, when applied to
1 !       arrays.
1 !
1 ! ENVIRONMENT:  VAX-11 User Mode
1 !
1 ! AUTHOR: John Sauter, CREATION DATE: 30-AUG-1979
1 !
1 ! MODIFIED BY:
1 !
1 ! 1-001 - Original.
1 ! 1-002 - Version 001 was just a skeleton, code single-dimensioned
1 !               numeric arrays moving TO the buffer, so we can see how
1 !               the algorithms work.  JBS 31-AUG-1979
1 ! 1-003 - Code DEST_NA.  JBS 06-SEP-1979
1 ! 1-004 - Code SRC_SA.  JBS 10-SEP-1979
1 ! 1-005 - Code the other effectors.  All coding is now complete.
1 !               JBS 13-SEP-1979
1 ! 1-006 - When initializing a string descriptor, be sure the pointer
1 !               is zero.  JBS 18-SEP-1979
1 ! 1-007 - When the source is a string array and a length is specified,
1 !               always put the specified number of characters in the buffer.
1 !               JBS 22-OCT-1979
```

```
1 ! 1-008 - SRC_NA, SRC_SA, DEST_NA, and DEST_SA must access the array
1 !         in row major order.  PL 1-Jul-81
1 ! 1-009 - Add support for records.  MOVE_ARRAY, SRC_SA, and DEST_SA
1 !         have been modified for dtype Z.  PLL 26-Feb-82
1 ! 1-010 - Add support for dynamically mapped arrays.  New entry points
1 !         have been added, SRC_DSC and DEST_DSC.  PLL 2-Mar-82
1 ! 1-011 - Add support for decimal arrays.  PLL 12-Mar-82
1 ! 1-012 - Don't try to free value descrip in SRC_DSC and DEST_DSC.
1 !         Also, DEST_SA (and SRC_SA) should set the dtype of the src buffer to Z only
1 !         if the array is dtype Z.  BAS$STORE_BFA requires the two data
1 !         types to match.  PLL 19-Mar-1982
1 ! 1-013 - Add support for multiply dimensioned arrays.  PLL 5-Apr-1982
1 ! 1-014 - Fix CH$MOVE in DEST_NA.  PLL 4-May-1982
1 ! 1-015 - Fix DEST_SA.  STR$COPY_R expects a descriptor for the destination,
1 !         and this descriptor must be constructed for static string arrays.
1 !         PLL 5-May-1982
1 ! 1-016 - Fix DEST_SA again.  CH$MOVE updates destination pointer, not source
1 !         pointer.  PLL 10-May-1982
1 ! 1-017 - 1. Fix SRC_SA for virtual arrays - nulls are stripped off on a per
1 !         element basis.
1 !         2. Fix DEST_SA for virtual arrays - dest len. must reflect length
1 !         given in statement or be default string length.
1 !         3. Fix DEST_SA for arrays of RFAs or RECORDs - length found in
1 !         descriptor must be used.  DG 25-Jan-1984
1 ! 1-018 - Fix 1-017 (2).  By simply null-filling the rest of the virtual
1 !         array element, the LEN function will work correctly.  DG 14-Feb-1984
1 ! 1-019 - Fixed some bugs in SRC_SA:
1 !         1. Virt arrays of RFAs or RECORDs must have their element length
1 !         be a power of 2 (not found when looking in descriptor)
1 !         2. Virt arrays of RFAs or RECORDs should not go thru the null
1 !         stripping procedure, again because the length in the descriptor is the
1 !         actual element length.
1 !         3. CH$COPY is used for virtual arrays (instead of CH$MOVE), so that
1 !         if a LENGTH is specified and the element length is less, blank
1 !         padding will be accomplished
1 ! 1-020 - on a MOVE FROM to a static string array, if the length is less than
1 !         then element length, we should blank fill.  MDL 12-Apr-1984
1 !--
1
1 !<BLF/PAGE>
```

```
!
! SWITCHES:
!

SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);

!
! LINKAGES:
!

REQUIRE 'RTLIN:OTSLNK';                               ! Define linkages

!
! TABLE OF CONTENTS:
!

FORWARD ROUTINE
    BAS$MOVE_ARRAY : NOVALUE,                         ! MOVE an array
    SRC_NA : CALL_CCB NOVALUE,                        ! Source is numeric array
    SRC_SA : CALL_CCB NOVALUE,                        ! Source is string array
    DEST_NA : CALL_CCB NOVALUE,                       ! Destination is numeric array
    DEST_SA : CALL_CCB NOVALUE,                       ! Destination is string array
    SRC_DSC : CALL_CCB NOVALUE,                       ! Source is descriptor
    DEST_DSC : CALL_CCB NOVALUE;                      ! Destination is descriptor

!
! INCLUDE FILES:
!

REQUIRE 'RTLML:OTSLUB';                               ! LUB definitions

REQUIRE 'RTLML:OTSISB';                               ! ISB definitions

REQUIRE 'RTLIN:RTLPSECT';                             ! Macros for defining psects

LIBRARY 'RTLSTARLE';                                  ! System symbols

!
! MACROS:
!
!       NONE
!
! EQUATED SYMBOLS:
!
!       NONE
!
! PSECTS:
!
DECLARE_PSECTS (BAS);                                 ! Declare psects for BAS$ facility
!
! OWN STORAGE:
!
!       NONE
!
! EXTERNAL REFERENCES:
!
```

```
EXTERNAL ROUTINE
    BAS$$CB_GET : JSB_CB_GET NOVALUE,          ! Load current CCB
    BAS$$STOP : NOVALUE,                       ! Signal fatal error
    BAS$$STOP_IO : NOVALUE,                    ! Signal fatal I/O error
    BAS$$WHOLE_VA_FETCH : NOVALUE,             ! Fetch entire virtual array
    BAS$$WHOLE_VA_STORE : NOVALUE,             ! Store entire virtual array
    LIB$FREE_VM,                               ! Dealloc memory
    LIB$GET_VM,                                ! Alloc memory
    STR$COPY_R;                                ! Copy a string by reference
!+
! The following are the error codes used in this module.
!-

EXTERNAL LITERAL
    BAS$K_PROLOSSOR : UNSIGNED (8),            ! Program lost, sorry
    BAS$K_MOVOVEBUF : UNSIGNED (8),            ! MOVE overflows buffer
    BAS$K_NOTIMP : UNSIGNED (8);               ! Not implemented
```

```
1 GLOBAL ROUTINE BAS$MOVE_ARRAY (                          ! MOVE an array
1         SRC : REF BLOCK [8, BYTE],                       ! Source: buffer or array
1         DEST : REF BLOCK [8, BYTE],                      ! Destination: array or buffer
1         LEN                                              ! Optional length, for strings
1     ) : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Within a MOVE statement, move an array.  This routines decodes its
1 !       parameters to move either an array to the buffer, or the buffer
1 !       to an array.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       SRC.mq.r        The source, which has the form of either a static
1 !                       string (the buffer) or an array.
1 !       DEST.mq.r       The destination, which has the other form.
1 !       LEN.rl.v        Optional length, used with strings.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       OTS$$A_CUR_LUB, accessed using BAS$$CB_GET, points to the current
1 !                       LUB.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Signals if an error is encountered.
1 !       Updates the buffer's address and count to reflect the movement
1 !       of the array.
1 !
1 !--
1
2     BEGIN
2
2     GLOBAL REGISTER
2         CCB = K_CCB_REG : REF BLOCK [, BYTE];
2
2     BUILTIN
2         ACTUALCOUNT;
2
2     LOCAL
2         LENGTH;                                          ! Passed length, or -1
2
2     LENGTH = (IF (ACTUALCOUNT () GEQ 3) THEN .LEN ELSE -1);
2 !+
2 ! Set up register CCB for error messages.
2 !-
```

```
2      BAS$$CB_GET ();
2  !+
2  ! If we are not doing I/O, or if the I/O is not a MOVE statement,
2  ! then the compiled code has called this routine in the wrong context.
2  !-
2
2      IF (.CCB EQLA 0) THEN BAS$$STOP (BAS$K_PROLOSSOR);
2
2      IF ( NOT .CCB [LUB$V_OPENED]) THEN BAS$$STOP_IO (BAS$K_PROLOSSOR);
2
2      IF (.CCB [ISB$B_STTM_TYPE] NEQ ISB$K_ST_TY_MOV) THEN BAS$$STOP_IO (BAS$K_PROLOSSOR);
2
2  !+
2  ! Dispatch based on the class and data type of the source and destination
2  !-
2
2      SELECTONE .SRC [DSC$B_CLASS] OF
2          SET
2
2          [DSC$K_CLASS_A, DSC$K_CLASS_BFA] :
3              BEGIN                                   ! Source is array
3
3              IF (.DEST [DSC$B_CLASS] NEQ DSC$K_CLASS_S) THEN BAS$$STOP_IO (BAS$K_PROLOSSOR);
3
3              SELECTONE .SRC [DSC$B_DTYPE] OF
3                  SET
3
3                  [DSC$K_DTYPE_B, DSC$K_DTYPE_W, DSC$K_DTYPE_L, DSC$K_DTYPE_F,
3                   DSC$K_DTYPE_D, DSC$K_DTYPE_G, DSC$K_DTYPE_H, DSC$K_DTYPE_P] :
4                      BEGIN                           ! Source is numeric array
4                      SRC_NA (.SRC, .DEST);
3                      END;
3
3                  [DSC$K_DTYPE_T, DSC$K_DTYPE_Z] :
4                      BEGIN                           ! Source is string array or record
4                      SRC_SA (.SRC, .DEST, .LENGTH);
3                      END;
3
3                  [DSC$K_DTYPE_DSC] :
4                      BEGIN                           ! Source is descriptor
4                      SRC_DSC (.SRC, .DEST, .LENGTH);
3                      END;
3
3                  [OTHERWISE] :
3                      BAS$$STOP_IO (BAS$K_PROLOSSOR);
3                  TES;
3
2              END;
2
2          [DSC$K_CLASS_S] :
3              BEGIN                                   ! Source is buffer
3
3              SELECTONE .DEST [DSC$B_CLASS] OF
3                  SET
3
3                  [DSC$K_CLASS_A, DSC$K_CLASS_BFA] :
4                      BEGIN                           ! Destination is array
```

```
4
4                SELECTONE .DEST [DSC$B_DTYPE] OF
4                    SET
4
4                    [DSC$K_DTYPE_B, DSC$K_DTYPE_W, DSC$K_DTYPE_L,
4                     DSC$K_DTYPE_F, DSC$K_DTYPE_D, DSC$K_DTYPE_G,
4                     DSC$K_DTYPE_H, DSC$K_DTYPE_P] :
5                        BEGIN              ! Destination is numeric array
5                        DEST_NA (.SRC, .DEST);
4                        END;
4
4                    [DSC$K_DTYPE_T, DSC$K_DTYPE_Z] :
5                        BEGIN              ! Destination is string array
5                        DEST_SA (.SRC, .DEST, .LENGTH);
4                        END;
4
4                    [DSC$K_DTYPE_DSC] :
5                        BEGIN              ! Destination is descriptor
5                        DEST_DSC (.SRC, .DEST, .LENGTH);
4                        END;
4
4                    [OTHERWISE] :
4                        BAS$$STOP_IO (BAS$K_PROLOSSOR);
4                    TES;
3
3                END;
3
3            [OTHERWISE] :
3                BAS$$STOP_IO (BAS$K_PROLOSSOR);
3            TES;
2
2        END;
2
2    [OTHERWISE] :
2        BAS$$STOP_IO (BAS$K_PROLOSSOR);
2    TES;
2
2    RETURN;
1    END;                                        ! end of BAS$MOVE_ARRAY

                                                  .TITLE  BAS$MOVE_ARRAY
                                                  .IDENT  \1-020\

                                                  .EXTRN  BAS$$CB_GET, BAS$$STOP
                                                  .EXTRN  BAS$$STOP_IO, BAS$$WHOLE_VA_FETCH
                                                  .EXTRN  BAS$$WHOLE_VA_STORE
                                                  .EXTRN  LIB$FREE_VM, LIB$GET_VM
                                                  .EXTRN  STR$COPY_R, BAS$K_PROLOSSOR
                                                  .EXTRN  BAS$K_MOVOVEBUF
                                                  .EXTRN  BAS$K_NOTIMP

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

                            08FC 00000            .ENTRY  BAS$MOVE_ARRAY, Save R2,R3,R4,R5,R6,R7,R11   ; 1001
         57        00G  8F   9A 00002            MOVZBL  #BAS$K_PROLOSSOR, R7                          ;
         56 00000000G  00   9E 00006            MOVAB   BAS$$STOP_IO, R6                              ;
```

```
            03               6C  91 0000D          CMPB    (AP), #3                        ; 1054
                             06  1F 00010          BLSSU   1$                              ;
            55          0C   AC  D0 00012          MOVL    LEN, LENGTH                     ;
                             03  11 00016          BRB     2$                              ;
            55               01  CE 00018 1$:      MNEGL   #1, LENGTH                      ;
                    00000000G 00  16 0001B 2$:     JSB     BAS$$CB_GET                     ; 1058
                             5B  D5 00021          TSTL    CCB                             ; 1064
                             0A  12 00023          BNEQ    3$                              ;
            7E               57  9A 00025          MOVZBL  R7, -(SP)                       ;
 00000000G  00               01  FB 00028          CALLS   #1, BAS$$STOP                   ;
            06          FC   AB  E8 0002F 3$:      BLBS    -4(CCB), 4$                     ; 1066
            7E               57  9A 00033          MOVZBL  R7, -(SP)                       ;
            66               01  FB 00036          CALLS   #1, BAS$$STOP_IO                ;
            2E        FF71   CB  91 00039 4$:      CMPB    -143(CCB), #46                  ; 1068
                             06  13 0003E          BEQL    5$                              ;
            7E               57  9A 00040          MOVZBL  R7, -(SP)                       ;
            66               01  FB 00043          CALLS   #1, BAS$$STOP_IO                ;
            53          04   AC  D0 00046 5$:      MOVL    SRC, R3                         ; 1074
            50          03   A3  9A 0004A          MOVZBL  3(R3), R0                       ;
            04               50  91 0004E          CMPB    R0, #4                          ; 1077
                             06  13 00051          BEQL    6$                              ;
      BF    8F               50  91 00053          CMPB    R0, #191                        ;
                             5D  12 00057          BNEQ    14$                             ;
            54          08   AC  D0 00059 6$:      MOVL    DEST, R4                        ; 1080
            01          03   A4  91 0005D          CMPB    3(R4), #1                       ;
                             06  13 00061          BEQL    7$                              ;
            7E               57  9A 00063          MOVZBL  R7, -(SP)                       ;
            66               01  FB 00066          CALLS   #1, BAS$$STOP_IO                ;
            52          02   A3  9A 00069 7$:      MOVZBL  2(R3), R2                       ; 1082
            06               52  91 0006D          CMPB    R2, #6                          ; 1085
                             05  1F 00070          BLSSU   8$                              ;
            08               52  91 00072          CMPB    R2, #8                          ;
                             19  1B 00075          BLEQU   10$                             ;
            0A               52  91 00077 8$:      CMPB    R2, #10                         ;
                             05  1F 0007A          BLSSU   9$                              ;
            0B               52  91 0007C          CMPB    R2, #11                         ;
                             0F  1B 0007F          BLEQU   10$                             ;
            15               52  91 00081 9$:      CMPB    R2, #21                         ;
                             0A  13 00084          BEQL    10$                             ;
            1B               52  91 00086          CMPB    R2, #27                         ;
                             0D  1F 00089          BLSSU   11$                             ;
            1C               52  91 0008B          CMPB    R2, #28                         ;
                             08  1A 0008E          BGTRU   11$                             ;
                             18  BB 00090 10$:     PUSHR   #^M<R3,R4>                      ; 1088
     0000V  CF               02  FB 00092          CALLS   #2, SRC_NA                      ;
                                 04 00097          RET                                     ; 1082
                             52  D5 00098 11$:     TSTL    R2                              ; 1091
                             05  13 0009A          BEQL    12$                             ;
            0E               52  91 0009C          CMPB    R2, #14                         ;
                             08  12 0009F          BNEQ    13$                             ;
                             38  BB 000A1 12$:     PUSHR   #^M<R3,R4,R5>                   ; 1093
     0000V  CF               03  FB 000A3          CALLS   #3, SRC_SA                      ;
                                 04 000A8          RET                                     ; 1082
            18               52  91 000A9 13$:     CMPB    R2, #24                         ; 1096
                             6D  12 000AC          BNEQ    22$                             ;
                             38  BB 000AE          PUSHR   #^M<R3,R4,R5>                   ; 1098
     0000V  CF               03  FB 000B0          CALLS   #3, SRC_DSC                     ;
```

```
                                  04 000B5          RET                                ; 1082
             01          50       91 000B6  14$:    CMPB    R0, #1                     ; 1107
                         60       12 000B9          BNEQ    22$
             54    08    AC       D0 000BB          MOVL    DEST, R4                   ; 1110
             50    03    A4       9A 000BF          MOVZBL  3(R4), R0
             04          50       91 000C3          CMPB    R0, #4                     ; 1113
                         06       13 000C6          BEQL    15$
     BF      8F          50       91 000C8          CMPB    R0, #191
                         4D       12 000CC          BNEQ    22$
             52    02    A4       9A 000CE  15$:    MOVZBL  2(R4), R2                  ; 1116
             06          52       91 000D2          CMPB    R2, #6                     ; 1119
                         05       1F 000D5          BLSSU   16$
             08          52       91 000D7          CMPB    R2, #8
                         19       1B 000DA          BLEQU   18$
             0A          52       91 000DC  16$:    CMPB    R2, #10
                         05       1F 000DF          BLSSU   17$
             0B          52       91 000E1          CMPB    R2, #11
                         0F       1B 000E4          BLEQU   18$
             15          52       91 000E6  17$:    CMPB    R2, #21
                         0A       13 000E9          BEQL    18$
             1B          52       91 000EB          CMPB    R2, #27
                         0D       1F 000EE          BLSSU   19$
             1C          52       91 000F0          CMPB    R2, #28
                         08       1A 000F3          BGTRU   19$
                         18       BB 000F5  18$:    PUSHR   #^M<R3,R4>                 ; 1123
  0000V      CF          02       FB 000F7          CALLS   #2, DEST_NA
                                  04 000FC          RET                                ; 1116
                         52       D5 000FD  19$:    TSTL    R2                         ; 1126
                         05       13 000FF          BEQL    20$
             0E          52       91 00101          CMPB    R2, #14
                         08       12 00104          BNEQ    21$
                         38       BB 00106  20$:    PUSHR   #^M<R3,R4,R5>              ; 1128
  0000V      CF          03       FB 00108          CALLS   #3, DEST_SA
                                  04 0010D          RET                                ; 1116
             18          52       91 0010E  21$:    CMPB    R2, #24                    ; 1131
                         08       12 00111          BNEQ    22$
                         38       BB 00113          PUSHR   #^M<R3,R4,R5>              ; 1133
  0000V      CF          03       FB 00115          CALLS   #3, DEST_DSC
                                  04 0011A          RET                                ; 1116
             7E          57       9A 0011B  22$:    MOVZBL  R7, -(SP)                  ; 1149
             66          01       FB 0011E          CALLS   #1, BAS$$STOP_IO
                                  04 00121          RET                                ; 1153

; Routine Size:  290 bytes,    Routine Base:  _BAS$CODE + 0000

;  328          1154  1
```

```
1 ROUTINE SRC_NA (                                      ! MOVE a numeric array
1       SRC : REF BLOCK [, BYTE],                       ! Source: array
1       DEST : REF BLOCK [8, BYTE]                      ! Destination: buffer
1    ) : CALL_CCB NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Within a MOVE statement, move a numeric array to the buffer.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       SRC.rx.da       The source, a numeric array.
1 !       DEST.mq.r       The destination, the I/O buffer.  It is updated to
1 !                       reflect the array.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       CCB, which is used only to provide good error messages.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Signals if an error is encountered.
1 !       Updates the buffer's address and count to reflect the movement
1 !       of the array.
1 !
1 !--
1
2    BEGIN
2
2    EXTERNAL REGISTER
2       CCB : REF BLOCK [, BYTE];
2
2    LOCAL
2       LEN;                                            ! length of array elements
2                                                       ! (needed for decimal)
2 !+
2 ! The total number of bytes in the array must not be greater than the
2 ! remaining length of the buffer.  If this is an array of numeric descriptors,
2 ! SRC_DSC has already checked for the MOVE overflowing the buffer.
2 !-
2
2    IF (.SRC [DSC$B_DTYPE] NEQ DSC$K_DTYPE_DSC) AND
3       (.SRC [DSC$L_ARSIZE] GTRU .DEST [DSC$W_LENGTH])
2    THEN
2        BAS$$STOP_IO (BAS$K_MOVOVEBUF);
2
2 !+
```

```
2 ! A memory numeric array can be moved in one instruction.  A virtual array,
2 ! however, must call the fetch routine.  A dynamically mapped numeric array
2 ! is in memory but elements may not be stored contiguously.  Therefore if
2 ! this is an array of descriptors, perform the move element by element.
2 !-
2
2     IF .SRC [DSC$B_CLASS] EQL DSC$K_CLASS_A
2     THEN
3         BEGIN                                       ! memory array
3         IF .SRC [DSC$B_DTYPE] EQL DSC$K_DTYPE_DSC
3         THEN
4             BEGIN                                   ! dynamically mapped array
4             LOCAL
4                 END_ADDR;
4             END_ADDR = .SRC [DSC$A_POINTER] + .SRC [DSC$L_ARSIZE] - .SRC [DSC$W_LENGTH];
4                                                     ! addr of last desc
4             INCR DSC_PTR FROM .SRC [DSC$A_POINTER] TO .END_ADDR
4                 BY .SRC [DSC$W_LENGTH] DO
5                 BEGIN
5                 MAP
5                     DSC_PTR : REF BLOCK [8,BYTE];
6                 LEN = (IF .DSC_PTR [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
7                         THEN (.DSC_PTR [DSC$W_LENGTH]/2 + 1)
5                         ELSE .DSC_PTR [DSC$W_LENGTH]);
5                 DEST [DSC$A_POINTER] = CH$MOVE (.LEN, .DSC_PTR [DSC$A_POINTER],
5                         .DEST [DSC$A_POINTER]);
5                 DEST [DSC$W_LENGTH] = .DEST [DSC$W_LENGTH] - .LEN;
4                 END;
4             END
3         ELSE
4             BEGIN                                   ! numeric array
5             LEN = (IF .SRC [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
6                     THEN (.SRC [DSC$W_LENGTH]/2 + 1) * (.SRC [DSC$L_ARSIZE]/ .SRC [DSC$W_LENGTH])
4                     ELSE .SRC [DSC$L_ARSIZE]);
4             DEST [DSC$A_POINTER] = CH$MOVE (.LEN, .SRC [DSC$A_POINTER],
4                 .DEST [DSC$A_POINTER]);
4             DEST [DSC$W_LENGTH] = .DEST [DSC$W_LENGTH] - .LEN;
3             END;
3         END
2     ELSE
3         BEGIN                                       ! virtual array
3         BAS$$WHOLE_VA_FETCH (.SRC, .DEST [DSC$A_POINTER]);
4         LEN = (IF .SRC [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
5                 THEN (.SRC [DSC$W_LENGTH]/2 + 1) * (.SRC [DSC$L_ARSIZE]/ .SRC [DSC$W_LENGTH])
3                 ELSE .SRC [DSC$L_ARSIZE]);
3         DEST [DSC$A_POINTER] = .DEST [DSC$A_POINTER] + .LEN;
3         DEST [DSC$W_LENGTH] = .DEST [DSC$W_LENGTH] - .LEN;
2         END;
2
1     END;                                            ! of SRC_NA


                         07FC 00000 SRC_NA: .WORD    Save R2,R3,R4,R5,R6,R7,R8,R9,R10        ; 1155
              5E      04  C2 00002          SUBL2    #4, SP                                  ;
```

```
                                  52     04   AC  D0 00005           MOVL    SRC, R2                               ; 1206
                                  18     02   A2  91 00009           CMPB    2(R2), #24                            ;
                                              14  13 0000D           BEQL    1$                                    ;
  0C   A2   08   BC               10          00  ED 0000F           CMPZV   #0, #16, @DEST, 12(R2)                ; 1207
                                              0B  1E 00016           BGEQU   1$                                    ;
                                  7E    00G   8F  9A 00018           MOVZBL  #BAS$K_MOVOVEBUF, -(SP)               ; 1209
                      00000000G   00          01  FB 0001C           CALLS   #1, BAS$$STOP_IO                      ;
                                  53     0C   A2  9E 00023  1$:      MOVAB   12(R2), R3                            ; 1226
                                  58     08   AC  D0 00027           MOVL    DEST, R8                              ; 1236
                                  59     04   A8  9E 0002B           MOVAB   4(R8), R9                             ;
                                  04     03   A2  91 0002F           CMPB    3(R2), #4                             ; 1218
                                              6D  12 00033           BNEQ    9$                                    ;
                                  18     02   A2  91 00035           CMPB    2(R2), #24                            ; 1221
                                              3C  12 00039           BNEQ    6$                                    ;
                 51        04     A2          63  C1 0003B           ADDL3   (R3), 4(R2), R1                       ; 1226
                                  5A          62  3C 00040           MOVZWL  (R2), R10                             ;
                 6E               51          5A  C3 00043           SUBL3   R10, R1, END_ADDR                     ;
                                  56     04   A2  D0 00047           MOVL    4(R2), DSC_PTR                        ; 1229
                                              24  11 0004B           BRB     5$                                    ;
                                  15     02   A6  91 0004D  2$:      CMPB    2(DSC_PTR), #21                       ; 1233
                                              0C  12 00051           BNEQ    3$                                    ;
                                  50          66  3C 00053           MOVZWL  (DSC_PTR), R0                         ; 1234
                                  50          02  C6 00056           DIVL2   #2, R0                                ;
                                  57     01   A0  9E 00059           MOVAB   1(R0), LEN                            ;
                                              03  11 0005D           BRB     4$                                    ;
                                  57          66  3C 0005F  3$:      MOVZWL  (DSC_PTR), LEN                        ; 1235
            00   B9        04     B6          57  28 00062  4$:      MOVC3   LEN, @4(DSC_PTR), @0(R9)              ; 1237
                                  69          53  D0 00068           MOVL    R3, (R9)                              ;
                                  68          57  A2 0006B           SUBW2   LEN, (R8)                             ; 1238
                                  56          5A  C0 0006E           ADDL2   R10, DSC_PTR                          ; 1228
                                  6E          56  D1 00071  5$:      CMPL    DSC_PTR, END_ADDR                     ;
                                              D7  15 00074           BLEQ    2$                                    ;
                                                  04 00076           RET                                           ; 1221
                                  15     02   A2  91 00077  6$:      CMPB    2(R2), #21                            ; 1243
                                              17  12 0007B           BNEQ    7$                                    ;
                                  50          62  3C 0007D           MOVZWL  (R2), R0                              ; 1244
                                  50          02  C6 00080           DIVL2   #2, R0                                ;
                                  51     01   A0  9E 00083           MOVAB   1(R0), R1                             ;
                                  50          62  3C 00087           MOVZWL  (R2), R0                              ;
                 50               63          50  C7 0008A           DIVL3   R0, (R3), R0                          ;
                 57               50          51  C5 0008E           MULL3   R1, R0, LEN                           ;
                                              03  11 00092           BRB     8$                                    ;
                                  57          63  D0 00094  7$:      MOVL    (R3), LEN                             ; 1245
            00   B9        04     B2          57  28 00097  8$:      MOVC3   LEN, @4(R2), @0(R9)                   ; 1247
                                  69          53  D0 0009D           MOVL    R3, (R9)                              ;
                                              2E  11 000A0           BRB     12$                                   ; 1248
                                              69  DD 000A2  9$:      PUSHL   (R9)                                  ; 1253
                                              52  DD 000A4           PUSHL   R2                                    ;
                      00000000G   00          02  FB 000A6           CALLS   #2, BAS$$WHOLE_VA_FETCH               ;
                                  15     02   A2  91 000AD           CMPB    2(R2), #21                            ; 1254
                                              17  12 000B1           BNEQ    10$                                   ;
                                  50          62  3C 000B3           MOVZWL  (R2), R0                              ; 1255
                                  50          02  C6 000B6           DIVL2   #2, R0                                ;
                                  51     01   A0  9E 000B9           MOVAB   1(R0), R1                             ;
                                  50          62  3C 000BD           MOVZWL  (R2), R0                              ;
                 50               63          50  C7 000C0           DIVL3   R0, (R3), R0                          ;
                 57               50          51  C5 000C4           MULL3   R1, R0, LEN                           ;
```

```
                          03  11 000C8           BRB     11$                          ;
              57          63  D0 000CA  10$:     MOVL    (R3), LEN                    ; 1256
              69          57  C0 000CD  11$:     ADDL2   LEN, (R9)                    ; 1257
              68          57  A2 000D0  12$:     SUBW2   LEN, (R8)                    ; 1258
                              04 000D3           RET                                  ; 1261

; Routine Size:  212 bytes,    Routine Base:  _BAS$CODE + 0122
```

```
1 ROUTINE SRC_SA (                                          ! MOVE a string array
1         SRC : REF BLOCK [, BYTE],                         ! Source: array
1         DEST : REF BLOCK [, BYTE],                        ! Destination: buffer
1         LENGTH                                            ! Limit on string size
1     ) : CALL_CCB NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Within a MOVE statement, move a string array or record to the buffer.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       SRC.rx.da       The source, a string array or record.
1 !       DEST.mq.r       The destination, the I/O buffer.  It is updated to
1 !                       reflect the array.
1 !       LENGTH.rl.v     -1 or limit of length of string to copy
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       CCB, which is used only to provide good error messages.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Signals if an error is encountered.
1 !       Updates the buffer's address and count to reflect the movement
1 !       of the array.
1 !
1 !--
1
2     BEGIN
2
2     EXTERNAL REGISTER
2         CCB : REF BLOCK [, BYTE];
2
2     LOCAL
2         POWER_OF_2: VECTOR [10, WORD]
2             !+
2             ! Only have to check up to 512 because the compiler
2             ! allocates virtual arrays so that they will not
2             ! cross block boundaries.
2             !-
2             INITIAL ( WORD(1,2,4,8,16,32,64,128,256,512) ),
2         LEN;
2
2 !+
2 ! Dynamic string arrays will be arrays of descriptors, so the pointer field
2 ! must be used to get to the string.  The elements in a static string array
```

```
2 ! will be the strings themselves.  In both cases do an element by element move
2 ! in order to accomodate padding, etc. which may be necessary.  Virtual arrays
2 ! must call the virtual array fetch routine.
2 !-
2
2       IF .SRC [DSC$B_CLASS] EQL DSC$K_CLASS_A
2       THEN
3           BEGIN
3           !+
3           ! memory array
3           !-
3           IF .SRC [DSC$B_DTYPE] EQL DSC$K_DTYPE_DSC
3           THEN
4               BEGIN
4               !+
4               ! dynamic strings
4               !-
4               LOCAL
4                   END_ADDR;                           ! addr of last elem of array
4
4               END_ADDR = .SRC [DSC$A_POINTER] + .SRC [DSC$L_ARSIZE] - .SRC [DSC$W_LENGTH];
4               !+
4               ! Loop thru array moving each element to buffer.
4               !-
4               INCR DSC_PTR FROM .SRC [DSC$A_POINTER] TO .END_ADDR BY .SRC [DSC$W_LENGTH] DO
5                   BEGIN
5
5                   MAP
5                       DSC_PTR : REF BLOCK [8,BYTE];
5
5                   LEN = (IF .LENGTH LSS 0 THEN .DSC_PTR [DSC$W_LENGTH] ELSE .LENGTH);
5                                                       ! dest length depends on if
5                                                       !  LENGTH parameter given
5                   IF .DEST [DSC$W_LENGTH] LSSU .LEN
5                   THEN
5                       BAS$$STOP_IO (BAS$K_MOVOVEBUF);
5                   DEST [DSC$A_POINTER] = CH$COPY (.DSC_PTR [DSC$W_LENGTH],
5                       .DSC_PTR [DSC$A_POINTER], %C' ', .LEN,
5                       .DEST [DSC$A_POINTER]);         ! move string from array to
5                                                       !  buffer, blank padding if
5                                                       !  necessary
5                   DEST [DSC$W_LENGTH] = .DEST [DSC$W_LENGTH] - .LEN;
5                                                       ! buffer len must be decreased
5                                                       !  by amt moved so check can be
5                                                       !  done so that further moves
5                                                       !  will not overflow buffer
4                   END;
4               END
3           ELSE
4               BEGIN
4               !+
4               ! static strings or records
4               !-
4               LOCAL
4                   END_ADDR;                           ! addr of last elem of array
4
4               LEN = (IF .LENGTH LSS 0 THEN .SRC [DSC$W_LENGTH] ELSE .LENGTH);
```

```
                                                    ! dest length depends on if
                                                    !  LENGTH parameter given
            END_ADDR = .SRC [DSC$A_POINTER] + .SRC [DSC$L_ARSIZE] - .SRC [DSC$W_LENGTH];
            !+
            ! Loop thru array moving each element to buffer.
            !-
            INCR ELEM_PTR FROM .SRC [DSC$A_POINTER] TO .END_ADDR BY .SRC [DSC$W_LENGTH] DO
                BEGIN

                IF .DEST [DSC$W_LENGTH] LSSU .LEN
                THEN
                    BAS$$STOP_IO (BAS$K_MOVOVEBUF);
                DEST [DSC$A_POINTER] = CH$COPY (.SRC [DSC$W_LENGTH],
                    .ELEM_PTR, %C' ', .LEN,
                    .DEST [DSC$A_POINTER]);         ! move string from array to
                                                    !  buffer, blank padding if
                                                    !  necessary
                DEST [DSC$W_LENGTH] = .DEST [DSC$W_LENGTH] - .LEN;
                                                    ! buffer len must be decreased
                                                    !  by amt moved so check can be
                                                    !  done so that further moves
                                                    !  will not overflow buffer
                END;
            END;
        END
    ELSE
            IF .SRC [DSC$B_CLASS] EQL DSC$K_CLASS_BFA
            THEN
                BEGIN
                !+
                ! virtual array
                !-
                LOCAL
                    DEST_LEN,                       ! buffer element length
                    ELEM_LEN,                       ! array element length
                    MEM_LOC;                        ! virtual memory pointer

                LIB$GET_VM (SRC [DSC$L_ARSIZE], MEM_LOC);
                                                    ! alloc memory for virtual array
                BAS$$WHOLE_VA_FETCH (.SRC, .MEM_LOC);
                                                    ! move array to memory
                ELEM_LEN = .SRC [DSC$W_LENGTH];
                !+
                ! If we are dealing with an array of RFAs or RECORDs,
                ! ELEM_LEN must be changed to a power of 2 since that is
                ! the virtual array element's true size.
                !-
                IF .SRC [DSC$B_DTYPE] EQL DSC$K_DTYPE_Z
                THEN
                    BEGIN

                    ELEM_LEN = (INCR I FROM 0 TO 9
                    DO
                        IF .ELEM_LEN LEQ .POWER_OF_2 [.I]
                        THEN
                            EXITLOOP .POWER_OF_2 [.I]);
```

```
3            END;
3        !+
3        ! Loop thru array moving each element to buffer.
3        !-
4        INCR PTR FROM .MEM_LOC TO (.MEM_LOC + .SRC [DSC$L_ARSIZE] - .ELEM_LEN)
3            BY .ELEM_LEN DO
3
4            BEGIN
4
4            LOCAL
4                END_PTR;                    ! ptr to end of array element,
4                                            ! not including nulls
4
4            !+
4            ! Trailing nulls must be removed.  Calculate the length of
4            ! actual characters not including nulls on a per character
4            ! basis.
4            ! NOTE - virtual arrays of RECORDs or RFAs should not have
4            ! the nulls removed.  Reason being desc length reflects the
4            ! RECORD or RFA size and not the virt array element size.
4            !-
4            END_PTR = .PTR + .SRC[DSC$W_LENGTH] - 1;
4                                            ! point to last char in array
4                                            !  element
4            IF .SRC [DSC$B_DTYPE] NEQ DSC$K_DTYPE_Z
4            THEN
4                UNTIL .END_PTR EQL (.PTR - 1) DO
5                    BEGIN
5                    !+
5                    ! Go backwards to 1st char looking for 1st non null.
5                    !-
5                    IF CH$FAIL (CH$FIND_NOT_CH (1, .END_PTR, %B'0') )
5                    THEN
5                        END_PTR = .END_PTR - 1
5                    ELSE
5                        EXITLOOP;
5
4                    END;
4            LEN = .END_PTR - .PTR + 1;  ! lenght of element w/o nulls
4            DEST_LEN = (IF .LENGTH LSS 0 THEN .LEN ELSE .LENGTH);
4                                            ! dest length depends on if
4                                            !  LENGTH parameter given
4            IF (.DEST [DSC$W_LENGTH] LSSU .LEN) THEN BAS$$STOP_IO (BAS$K_MOVOVEBUF);
4            DEST [DSC$A_POINTER] = CH$COPY (.LEN, .PTR, %C' ',
4                                            .DEST_LEN, .DEST [DSC$A_POINTER]);
4                                            ! move string from array to
4                                            !  buffer, blank padding if
4                                            !  necessary
4            DEST [DSC$W_LENGTH] = .DEST [DSC$W_LENGTH] - .DEST_LEN;
4                                            ! buffer len must be decreased
4                                            !  by amt moved so check can be
4                                            !  done so that further moves
4                                            !  will not overflow buffer
3            END;
3
3        LIB$FREE_VM (SRC [DSC$L_ARSIZE], MEM_LOC);
3                                            ! dealloc memory used for array
```

```
;  666          1490  2                  END;
;  667          1491  2
;  668          1492  1        END;                                        ! of SRC_SA



                                                               001F6          .BLKB    2
0200  0100  0080  0040  0020  0010  0008  0004  0002  0001    001F8 P.AAA:   .WORD    1, 2, 4, 8, 16, 32, 64, 128, 256, 512          ;



                                                         07FC 00000 SRC_SA:  .WORD    Save R2,R3,R4,R5,R6,R7,R8,R9,R10          ; 1262
                                  5E               24    C2 00002            SUBL2    #36, SP                                   ;
              10    AE       E3   AF               14    28 00005            MOVC3    #20, P.AAA, POWER_OF_2                    ; 1313
                                  58       04      AC    D0 0000B            MOVL     SRC, R8                                   ; 1324
                                  04       03      A8    91 0000F            CMPB     3(R8), #4                                 ;
                                                   03    13 00013            BEQL     1$                                        ;
                                                 00A6    31 00015            BRW      13$                                       ;
                                  5A       0C      AC    D0 00018 1$:        MOVL     LENGTH, R10                               ; 1349
                                  56       08      AC    D0 0001C            MOVL     DEST, R6                                  ; 1352
                                  18       02      A8    91 00020            CMPB     2(R8), #24                                ; 1330
                                                   4E    12 00024            BNEQ     7$                                        ;
                    50       04   A8       0C      A8    C1 00026            ADDL3    12(R8), 4(R8), R0                         ; 1339
                             08   AE               68    3C 0002C            MOVZWL   (R8), 8(SP)                               ;
              04    AE            50       08      AE    C3 00030            SUBL3    8(SP), R0, END_ADDR                       ;
                                  59       04      A8    D0 00036            MOVL     4(R8), DSC_PTR                            ; 1343
                                                   31    11 0003A            BRB      6$                                        ;
                                                   5A    D5 0003C 2$:        TSTL     R10                                       ; 1349
                                                   05    18 0003E            BGEQ     3$                                        ;
                                  57               69    3C 00040            MOVZWL   (DSC_PTR), LEN                            ;
                                                   03    11 00043            BRB      4$                                        ;
                                  57               5A    D0 00045 3$:        MOVL     R10, LEN                                  ;
        57          66            10               00    ED 00048 4$:        CMPZV    #0, #16, (R6), LEN                        ; 1352
                                                   0B    1E 0004D            BGEQU    5$                                        ;
                                  7E      00G      8F    9A 0004F            MOVZBL   #BAS$K_MOVOVEBUF, -(SP)                   ; 1354
                    00000000G     00               01    FB 00053            CALLS    #1, BAS$$STOP_IO                          ;
        57          20       04   B9               69    2C 0005A 5$:        MOVC5    (DSC_PTR), @4(DSC_PTR), #32, LEN, @4(R6)  ; 1357
                                           04      B6       00060                                                       ;
                             04   A6               53    D0 00062            MOVL     R3, 4(R6)                                 ;
                                  66               57    A2 00066            SUBW2    LEN, (R6)                                 ; 1360
                                  59       08      AE    C0 00069            ADDL2    8(SP), DSC_PTR                            ; 1343
                             04   AE               59    D1 0006D 6$:        CMPL     DSC_PTR, END_ADDR                         ;
                                                   C9    15 00071            BLEQ     2$                                        ;
                                                         04 00073            RET                                                ; 1330
                                                   5A    D5 00074 7$:        TSTL     R10                                       ; 1375
                                                   05    18 00076            BGEQ     8$                                        ;
                                  57               68    3C 00078            MOVZWL   (R8), LEN                                 ;
                                                   03    11 0007B            BRB      9$                                        ;
                                  57               5A    D0 0007D 8$:        MOVL     R10, LEN                                  ;
                    50       04   A8       0C      A8    C1 00080 9$:        ADDL3    12(R8), 4(R8), R0                         ; 1378
                                  59               68    3C 00086            MOVZWL   (R8), R9                                  ;
              08    AE            50               59    C3 00089            SUBL3    R9, R0, END_ADDR                          ;
                                  5A       04      A8    D0 0008E            MOVL     4(R8), ELEM_PTR                           ; 1385
                                                   23    11 00092            BRB      12$                                       ;
        57          66            10               00    ED 00094 10$:       CMPZV    #0, #16, (R6), LEN                        ;
                                                   0B    1E 00099            BGEQU    11$                                       ;
                                  7E      00G      8F    9A 0009B            MOVZBL   #BAS$K_MOVOVEBUF, -(SP)                   ; 1387
```

```
                    00000000G  00            01  FB 0009F         CALLS   #1, BAS$$STOP_IO                  :
        57       20                 6A       59  2C 000A6  11$:   MOVC5   R9, (ELEM_PTR), #32, LEN, @4(R6)  : 1390
                                        04   B6     000AB                                                 :
                         04  A6              53  D0 000AD         MOVL    R3, 4(R6)                         :
                             66              57  A2 000B1         SUBW2   LEN, (R6)                         : 1393
                             5A              59  C0 000B4         ADDL2   R9, ELEM_PTR                      : 1382
                         08  AE              5A  D1 000B7  12$:   CMPL    ELEM_PTR, END_ADDR                :
                                             D7  15 000BB         BLEQ    10$                               :
                                             04     000BD         RET                                       : 1324
                         BF  8F         03   A8  91 000BE  13$:   CMPB    3(R8), #191                       : 1402
                                             01  13 000C3         BEQL    14$                               :
                                             04     000C5         RET                                       :
                                        0C   AE  9F 000C6  14$:   PUSHAB  MEM_LOC                           : 1413
                                        0C   A8  9F 000C9         PUSHAB  12(R8)                            :
                    00000000G  00            02  FB 000CC         CALLS   #2, LIB$GET_VM                    :
                                        0C   AE  DD 000D3         PUSHL   MEM_LOC                           : 1415
                                             58  DD 000D6         PUSHL   R8                                :
                    00000000G  00            02  FB 000D8         CALLS   #2, BAS$$WHOLE_VA_FETCH           :
                         04  AE              68  3C 000DF         MOVZWL  (R8), 4(SP)                       : 1417
                         08  AE         04   AE  D0 000E3         MOVL    4(SP), ELEM_LEN                   :
                                        02   A8  95 000E8         TSTB    2(R8)                             : 1423
                                             1E  12 000EB         BNEQ    17$                               :
                                             50  D4 000ED         CLRL    I                                 : 1429
                                        10 AE40  3F 000EF  15$:   PUSHAW  POWER_OF_2[I]                     :
  08   AE                9E          10          00  ED 000F3         CMPZV   #0, #16, @(SP)+, ELEM_LEN         :
                                             08  19 000F9         BLSS    16$                               :
                         08  AE         10 AE40  3C 000FB         MOVZWL  POWER_OF_2[I], ELEM_LEN           : 1431
                                             08  11 00101         BRB     17$                               :
                 E8          50              09  F3 00103  16$:   AOBLEQ  #9, I, 15$                        : 1429
                         08  AE              01  CE 00107         MNEGL   #1, ELEM_LEN                      : 1427
                 6E      0C  AE         0C   A8  C1 0010B  17$:   ADDL3   12(R8), MEM_LOC, (SP)             : 1437
                             6E         08   AE  C2 00111         SUBL2   ELEM_LEN, (SP)                    :
                             59         08   AC  D0 00115         MOVL    DEST, R9                          : 1476
                             56         0C   AE  D0 00119         MOVL    MEM_LOC, PTR                      :
                                             62  11 0011D         BRB     25$                               :
                 50      04  AE              01  C3 0011F  18$:   SUBL3   #1, 4(SP), R0                     : 1454
                 52          56              50  C1 00124         ADDL3   R0, PTR, END_PTR                  :
                                        02   A8  95 00128         TSTB    2(R8)                             : 1457
                                             19  13 0012B         BEQL    21$                               :
                             53         FF   A6  9E 0012D         MOVAB   -1(R6), R3                        : 1459
                             53              52  D1 00131  19$:   CMPL    END_PTR, R3                       :
                                             10  13 00134         BEQL    21$                               :
                 62          01              00  3B 00136         SKPC    #0, #1, (END_PTR)                 : 1464
                                             02  12 0013A         BNEQ    20$                               :
                                             51  D4 0013C         CLRL    R1                                :
                                             51  D5 0013E  20$:   TSTL    R1                                :
                                             04  12 00140         BNEQ    21$                               :
                                             52  D7 00142         DECL    END_PTR                           : 1466
                                             EB  11 00144         BRB     19$                               :
                             52              56  C2 00146  21$:   SUBL2   PTR, R2                           : 1471
                             57         01   A2  9E 00149         MOVAB   1(R2), LEN                        :
                                        0C   AC  D5 0014D         TSTL    LENGTH                            : 1472
                                             05  18 00150         BGEQ    22$                               :
                             5A              57  D0 00152         MOVL    LEN, DEST_LEN                     :
                                             04  11 00155         BRB     23$                               :
                             5A         0C   AC  D0 00157  22$:   MOVL    LENGTH, DEST_LEN                  :
        57       08      BC          10          00  ED 0015B  23$:   CMPZV   #0, #16, @DEST, LEN               : 1475
```

```
                                         0B  1E 00161        BGEQU   24$                                      ;
                        7E          00G  8F  9A 00163        MOVZBL  #BAS$K_MOVOVEBUF, -(SP)                  ;
              00000000G 00               01  FB 00167        CALLS   #1, BAS$$STOP_IO                         ;
      5A             20 66               57  2C 0016E 24$:   MOVC5   LEN, (PTR), #32, DEST_LEN, @4(R9)        ; 1477
                                    04   B9     00173                                                         ;
                     04 A9               53  D0 00175        MOVL    R3, 4(R9)                                ;
                     08 BC               5A  A2 00179        SUBW2   DEST_LEN, @DEST                          ; 1481
                        56          08   AE  C0 0017D        ADDL2   ELEM_LEN, PTR                            ; 1437
                        6E               56  D1 00181 25$:   CMPL    PTR, (SP)                                ;
                                         99  15 00184        BLEQ    18$                                      ;
                                    0C   AE  9F 00186        PUSHAB  MEM_LOC                                  ; 1488
                                    0C   A8  9F 00189        PUSHAB  12(R8)                                   ;
              00000000G 00               02  FB 0018C        CALLS   #2, LIB$FREE_VM                          ;
                                             04 00193        RET                                              ; 1492

; Routine Size:  404 bytes,    Routine Base:  _BAS$CODE + 020C
```

```
ROUTINE SRC_DSC (                                   ! MOVE an array of dsc
        SRC : REF BLOCK [, BYTE],                   ! Source: array
        DEST : REF BLOCK [, BYTE],                  ! Destination: buffer
        LENGTH                                      ! Limit on string size
    ) : CALL_CCB NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Within a MOVE statement, move an array of descriptors to a buffer.
!       The descriptors may be string descriptors or numeric descriptors
!       (in the case of dynamically mapped variables).  This routine
!       determines the dtype of the descriptors and dispatches to the
!       appropriate routine.
!
! FORMAL PARAMETERS:
!
!       SRC.rx.da       The source, an array of desc
!       DEST.mq.r       The destination, the I/O buffer.  It is updated to
!                       reflect the array.
!       LENGTH.rl.v     -1 or limit of length of string to copy
!
! IMPLICIT INPUTS:
!
!       CCB, which is used only to provide good error messages.
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       NONE
!
! SIDE EFFECTS:
!
!       Signals if an error is encountered.
!
!--

    BEGIN

    EXTERNAL REGISTER
        CCB : REF BLOCK [, BYTE];

    LOCAL
        ELEM_DSC : REF BLOCK [8,BYTE],
        NUM_ELEMS;


!+
! Set up a pointer to the first element of the array so that the dtype
! and length of the descriptor elements can be determined.
!-

    ELEM_DSC = .SRC [DSC$A_A0];
```

```
;  727     1550  2  !+
;  728     1551  2  !+
;  729     1552  2  ! Now just dispatch to the appropriate routine based on the dtype in the
;  730     1553  2  ! descriptor element.
;  731     1554  2  !-
;  732     1555  2
;  733     1556  2      SELECTONEU .ELEM_DSC [DSC$B_DTYPE] OF
;  734     1557  2          SET
;  735     1558  2
;  736     1559  2          [DSC$K_DTYPE_B, DSC$K_DTYPE_W, DSC$K_DTYPE_L, DSC$K_DTYPE_F,
;  737     1560  2           DSC$K_DTYPE_D, DSC$K_DTYPE_G, DSC$K_DTYPE_H, DSC$K_DTYPE_P]:
;  738     1561  3              BEGIN                                   ! dynamically mapped array
;  739     1562  3              !+
;  740     1563  3              ! Check for overflowing the buffer here to save doing it in
;  741     1564  3              ! SRC_NA.
;  742     1565  3              !-
;  743     1566  3              NUM_ELEMS = .SRC [DSC$L_ARSIZE]/ .SRC [DSC$W_LENGTH];
;  744     1567  3              IF (.NUM_ELEMS * .ELEM_DSC [DSC$W_LENGTH]) GTRU .DEST [DSC$W_LENGTH]
;  745     1568  3              THEN
;  746     1569  3                  BAS$$STOP_IO (BAS$K_MOVOVEBUF);
;  747     1570  3              SRC_NA (.SRC, .DEST);
;  748     1571  2              END;
;  749     1572  2
;  750     1573  2          [DSC$K_DTYPE_T, DSC$K_DTYPE_Z]:
;  751     1574  2              SRC_SA (.SRC, .DEST, .LENGTH);          ! string or record array
;  752     1575  2
;  753     1576  2          [OTHERWISE]:
;  754     1577  2              BAS$$STOP_IO (BAS$K_PROLOSSOR);
;  755     1578  2
;  756     1579  2          TES;
;  757     1580  2
;  758     1581  1      END;                                        ! of SRC_DSC
```

```
                      001C 00000 SRC_DSC:.WORD   Save R2,R3,R4                     ; 1493
54 0000000G     00    9E 00002          MOVAB   BAS$$STOP_IO, R4                  ;
52        04    AC    D0 00009          MOVL    SRC, R2                           ; 1549
51        10    A2    D0 0000D          MOVL    16(R2), ELEM_DSC                  ;
50        02    A1    9A 00011          MOVZBL  2(ELEM_DSC), R0                   ; 1556
06              50    91 00015          CMPB    R0, #6                            ; 1559
                05    1F 00018          BLSSU   1$                                ;
08              50    91 0001A          CMPB    R0, #8                            ;
                19    1B 0001D          BLEQU   3$                                ;
0A              50    91 0001F 1$:      CMPB    R0, #10                           ;
                05    1F 00022          BLSSU   2$                                ;
0B              50    91 00024          CMPB    R0, #11                           ;
                0F    1B 00027          BLEQU   3$                                ;
15              50    91 00029 2$:      CMPB    R0, #21                           ;
                0A    13 0002C          BEQL    3$                                ;
1B              50    91 0002E          CMPB    R0, #27                           ;
                2D    1F 00031          BLSSU   5$                                ;
1C              50    91 00033          CMPB    R0, #28                           ;
                28    1A 00036          BGTRU   5$                                ;
50              62    3C 00038 3$:      MOVZWL  (R2), NUM_ELEMS                   ; 1566
```

```
                  50      0C   A2             50  C7 0003B        DIVL3   NUM_ELEMS, 12(R2), NUM_ELEMS      ;
                               53             61  3C 00040        MOVZWL  (ELEM_DSC), R3                    ; 1567
                               50             53  C4 00043        MULL2   R3, R0                            ;
        50        08   BC      10             00  ED 00046        CMPZV   #0, #16, @DEST, R0                ;
                                              07  1E 0004C        BGEQU   4$                                ;
                               7E       00G   8F  9A 0004E        MOVZBL  #BAS$K_MOVOVEBUF, -(SP)           ; 1569
                               64             01  FB 00052        CALLS   #1, BAS$$STOP_IO                  ;
                                        08    AC  DD 00055 4$:    PUSHL   DEST                              ; 1570
                                              52  DD 00058        PUSHL   R2                                ;
                     FD23      CF             02  FB 0005A        CALLS   #2, SRC_NA                        ;
                                                  04 0005F        RET                                       ; 1556
                                              50  D5 00060 5$:    TSTL    R0                                ; 1573
                                              05  13 00062        BEQL    6$                                ;
                               0E             50  91 00064        CMPB    R0, #14                           ;
                                              0C  12 00067        BNEQ    7$                                ;
                               7E       08    AC  7D 00069 6$:    MOVQ    DEST, -(SP)                       ; 1574
                                              52  DD 0006D        PUSHL   R2                                ;
                     FDF8      CF             03  FB 0006F        CALLS   #3, SRC_SA                        ;
                                                  04 00074        RET                                       ;
                               7E       00G   8F  9A 00075 7$:    MOVZBL  #BAS$K_PROLOSSOR, -(SP)           ; 1577
                               64             01  FB 00079        CALLS   #1, BAS$$STOP_IO                  ;
                                                  04 0007C        RET                                       ; 1581

; Routine Size:  125 bytes,    Routine Base:  _BAS$CODE + 03A0
```

```
ROUTINE DEST_NA (                                       ! MOVE a numeric array
        SRC : REF BLOCK [8, BYTE],                      ! Source: buffer
        DEST : REF BLOCK [, BYTE]                       ! Destination: numeric array
    ) : CALL_CCB NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Within a MOVE statement, move the I/O buffer to a numeric array.
!
! FORMAL PARAMETERS:
!
!       SRC.mq.r        The source, the I/O buffer.  It is updated to account for the
!                       bytes taken from it.
!       DEST.wx.da      The destination, a numeric array.
!
! IMPLICIT INPUTS:
!
!       CCB, which is used only to provide good error messages.
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       NONE
!
! SIDE EFFECTS:
!
!       Signals if an error is encountered.
!       Updates the buffer's address and count to reflect the movement
!       of the array.
!
!--

    BEGIN

    EXTERNAL REGISTER
        CCB : REF BLOCK [, BYTE];

    LOCAL
        LEN;

!+
! The total number of bytes in the array must not be greater than the
! remaining length of the buffer.
!-

    IF (.SRC [DSC$B_DTYPE] NEQ DSC$K_DTYPE_DSC) AND
        (.SRC [DSC$W_LENGTH] LSSU .DEST [DSC$L_ARSIZE])
    THEN
         BAS$$STOP_IO (BAS$K_MOVOVEBUF);

!+
! Values can be moved to a memory array in one instruction.  If the array
```

```
! is virtual, however, the store routine must be called.  If the array is
! dynamically mapped, the elements are not contiguous and values must be
! moved one by one.
!-

    IF .DEST [DSC$B_CLASS] EQL DSC$K_CLASS_A
    THEN
        BEGIN                                         ! memory array
        IF .DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_DSC
        THEN
            BEGIN                                     ! dynamically mapped array
            LOCAL
                END_ADDR;                             ! addr of last elem of array
            END_ADDR = .DEST [DSC$A_POINTER] + .DEST [DSC$L_ARSIZE] - .DEST [DSC$W_LENGTH];
                                                      ! addr of last desc
            INCR DSC_PTR FROM .DEST [DSC$A_POINTER] TO .END_ADDR BY .DEST [DSC$W_LENGTH]  DO
                BEGIN
                MAP
                    DSC_PTR : REF BLOCK [8,BYTE];
                LEN = (IF .DSC_PTR [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
                       THEN (.DSC_PTR [DSC$W_LENGTH]/2 + 1)
                       ELSE .DSC_PTR [DSC$W_LENGTH]);
                CH$MOVE (.LEN, .SRC [DSC$A_POINTER], .DSC_PTR [DSC$A_POINTER]);
                SRC [DSC$A_POINTER] = .SRC [DSC$A_POINTER] + .LEN;
                SRC [DSC$W_LENGTH] = .SRC [DSC$W_LENGTH] - .LEN;
                END;
            END
        ELSE
            BEGIN                                     ! numeric array
            LEN = (IF .DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
                    THEN (.DEST [DSC$W_LENGTH]/2 + 1) * (.DEST [DSC$L_ARSIZE]/.DEST [DSC$W_LENGTH])
                    ELSE .DEST [DSC$L_ARSIZE]);
            CH$MOVE (.LEN, .SRC [DSC$A_POINTER], .DEST [DSC$A_POINTER]);
            SRC [DSC$A_POINTER] = .SRC [DSC$A_POINTER] + .LEN;
            SRC [DSC$W_LENGTH] = .SRC [DSC$W_LENGTH] - .LEN;
            END;
        END
    ELSE
        BEGIN                                         ! virtual array
        BAS$$WHOLE_VA_STORE (.DEST, .SRC [DSC$A_POINTER]);
        LEN = (IF .DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
                THEN (.DEST [DSC$W_LENGTH]/2 + 1) * (.DEST [DSC$L_ARSIZE]/.DEST [DSC$W_LENGTH])
                ELSE .DEST [DSC$L_ARSIZE]);
        SRC [DSC$A_POINTER] = .SRC [DSC$A_POINTER] + .LEN;
        SRC [DSC$W_LENGTH] = .SRC [DSC$W_LENGTH] - .LEN;
        END;

    END;                                              ! of DEST_NA



                                07FC 00000 DEST_NA:.WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10          ; 1582
          5E              04    C2 00002           SUBL2   #4, SP                                    ;
          59        04    AC    D0 00005           MOVL    SRC, R9                                   ; 1632
          18        02    A9    91 00009           CMPB    2(R9), #24                                ;
```

```
                                      17   13 0000D         BEQL    1$                                      ;
                          50      08  AC   D0 0000F         MOVL    DEST, R0                                ; 1633
  0C   A0         69              10  00   ED 00013         CMPZV   #0, #16, (R9), 12(R0)                   ;
                                      0B   1E 00019         BGEQU   1$                                      ;
                          7E     00G  8F   9A 0001B         MOVZBL  #BAS$K_MOVOVEBUF, -(SP)                 ; 1635
               00000000G  00          01   FB 0001F         CALLS   #1, BAS$$STOP_IO                        ;
                          52      08  AC   D0 00026  1$:    MOVL    DEST, R2                                ; 1644
                          53      0C  A2   9E 0002A         MOVAB   12(R2), R3                              ; 1652
                          58      04  A9   9E 0002E         MOVAB   4(R9), R8                               ; 1661
                          04      03  A2   91 00032         CMPB    3(R2), #4                               ; 1644
                                      6A   12 00036         BNEQ    9$                                      ;
                          18      02  A2   91 00038         CMPB    2(R2), #24                              ; 1647
                                      3C   12 0003C         BNEQ    6$                                      ;
                  51      04      A2  63   C1 0003E         ADDL3   (R3), 4(R2), R1                         ; 1652
                                  5A  62   3C 00043         MOVZWL  (R2), R10                               ;
                  6E              51  5A   C3 00046         SUBL3   R10, R1, END_ADDR                       ;
                          56      04  A2   D0 0004A         MOVL    4(R2), DSC_PTR                          ; 1654
                                      24   11 0004E         BRB     5$                                      ;
                          15      02  A6   91 00050  2$:    CMPB    2(DSC_PTR), #21                         ; 1658
                                      0C   12 00054         BNEQ    3$                                      ;
                          50          66   3C 00056         MOVZWL  (DSC_PTR), R0                           ; 1659
                          50          02   C6 00059         DIVL2   #2, R0                                  ;
                          57      01  A0   9E 0005C         MOVAB   1(R0), LEN                              ;
                                      03   11 00060         BRB     4$                                      ;
                          57          66   3C 00062  3$:    MOVZWL  (DSC_PTR), LEN                          ; 1660
          04   B6         00      B8  57   28 00065  4$:    MOVC3   LEN, @0(R8), @4(DSC_PTR)                ; 1661
                          68          57   C0 0006B         ADDL2   LEN, (R8)                               ; 1662
                          69          57   A2 0006E         SUBW2   LEN, (R9)                               ; 1663
                          56          5A   C0 00071         ADDL2   R10, DSC_PTR                            ; 1654
                          6E          56   D1 00074  5$:    CMPL    DSC_PTR, END_ADDR                       ;
                                      D7   15 00077         BLEQ    2$                                      ;
                                           04 00079         RET                                             ; 1647
                          15      02  A2   91 0007A  6$:    CMPB    2(R2), #21                              ; 1668
                                      17   12 0007E         BNEQ    7$                                      ;
                          50          62   3C 00080         MOVZWL  (R2), R0                                ; 1669
                          50          02   C6 00083         DIVL2   #2, R0                                  ;
                          51      01  A0   9E 00086         MOVAB   1(R0), R1                               ;
                          50          62   3C 0008A         MOVZWL  (R2), R0                                ;
                  50      63          50   C7 0008D         DIVL3   R0, (R3), R0                            ;
                  57      50          51   C5 00091         MULL3   R1, R0, LEN                             ;
                                      03   11 00095         BRB     8$                                      ;
                          57          63   D0 00097  7$:    MOVL    (R3), LEN                               ; 1670
          04   B2         00      B8  57   28 0009A  8$:    MOVC3   LEN, @0(R8), @4(R2)                     ; 1671
                                      2B   11 000A0         BRB     11$                                     ; 1672
                                      68   DD 000A2  9$:    PUSHL   (R8)                                    ; 1678
                                      52   DD 000A4         PUSHL   R2                                      ;
               00000000G  00          02   FB 000A6         CALLS   #2, BAS$$WHOLE_VA_STORE                 ;
                          15      02  A2   91 000AD         CMPB    2(R2), #21                              ; 1679
                                      17   12 000B1         BNEQ    10$                                     ;
                          50          62   3C 000B3         MOVZWL  (R2), R0                                ; 1680
                          50          02   C6 000B6         DIVL2   #2, R0                                  ;
                          51      01  A0   9E 000B9         MOVAB   1(R0), R1                               ;
                          50          62   3C 000BD         MOVZWL  (R2), R0                                ;
                  50      63          50   C7 000C0         DIVL3   R0, (R3), R0                            ;
                  57      50          51   C5 000C4         MULL3   R1, R0, LEN                             ;
                                      03   11 000C8         BRB     11$                                     ;
                          57          63   D0 000CA  10$:   MOVL    (R3), LEN                               ; 1681
```

```
        68              57  C0 000CD 11$:   ADDL2   LEN, (R8)                       ; 1682
        69              57  A2 000D0        SUBW2   LEN, (R9)                       ; 1683
                            04 000D3        RET                                     ; 1686

; Routine Size:  212 bytes,    Routine Base:  _BAS$CODE + 041D
```

```
ROUTINE DEST_SA (                                       ! MOVE a string array
        SRC : REF BLOCK [8, BYTE],                      ! Source: buffer
        DEST : REF BLOCK [, BYTE],                      ! Destination: array
        LENGTH                                          ! -1 or length of each string
    ) : CALL_CCB NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Within a MOVE statement, move the I/O buffer to a string array or
!       array of records.
!
! FORMAL PARAMETERS:
!
!       SRC.mq.r        The source, the I/O buffer.  This is updated to account for
!                       the bytes taken from it.
!       DEST.wx.da      The destination, a string array or array of records.
!
! IMPLICIT INPUTS:
!
!       CCB, which is used only to provide good error messages.
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       NONE
!
! SIDE EFFECTS:
!
!       Signals if an error is encountered.
!       Updates the buffer's address and count to reflect the movement
!       of the array.
!
!--

    BEGIN

    EXTERNAL REGISTER
        CCB : REF BLOCK [, BYTE];

    LITERAL
        K_DEFAULT_STR_LEN = 16;

    LOCAL
        POWER_OF_2: VECTOR [10, WORD]
            !+
            ! Only have to check up to 512 because the compiler
            ! allocates virtual arrays so that they will not
            ! cross block boundaries.
            !-
            INITIAL ( WORD(1,2,4,8,16,32,64,128,256,512) ),
        LEN;
```

```
2 !+
2 ! Dynamic string arrays are arrays of descriptors, while static string
2 ! arrays contain the strings themselves.  In both cases, do an element
2 ! by element move so that padding can be done if necessary.  Virtual
2 ! arrays must use the virtual array store routine.
2 !-
2
2     IF .DEST [DSC$B_CLASS] EQL DSC$K_CLASS_A
2     THEN
3         BEGIN
3         !+
3         ! memory array
3         !-
3         IF .DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_DSC
3         THEN
4             BEGIN
4             !+
4             ! static strings or records
4             !-
4             LOCAL
4                 END_ADDR;                          ! addr of last elem of array
4
4             LEN = (IF .LENGTH LSS 0 THEN K_DEFAULT_STR_LEN ELSE .LENGTH);
4                                                    ! dest length depends on if
4                                                    !  LENGTH parameter given,
4                                                    !  defaults to 16
4             IF .DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_Z
4             THEN
4                 LEN = .DEST [DSC$W_LENGTH];        ! if we are dealing with an
4                                                    !  array of RFAs or RECORDs,
4                                                    !  get the length from the descr
4
4             END_ADDR = .DEST [DSC$A_POINTER] + .DEST [DSC$L_ARSIZE] -
4                 .DEST [DSC$W_LENGTH];
4             !+
4             ! Loop thru array moving each element from buffer to it.
4             !-
4             INCR ELEM_PTR FROM .DEST [DSC$A_POINTER] TO .END_ADDR
4                 BY .DEST [DSC$W_LENGTH] DO
5                 BEGIN
5
6                 IF (.SRC [DSC$W_LENGTH] LSSU .LEN)
5                 THEN
5                     BAS$$STOP_IO (BAS$K_MOVOVEBUF);
5                 CH$COPY (.LEN, .SRC [DSC$A_POINTER], %C' ', .DEST [DSC$W_LENGTH], .ELEM_PTR);
5                                                    !  array, Blank padding if
5                                                    !  necessary
5                 SRC [DSC$A_POINTER] = .SRC [DSC$A_POINTER] + .LEN;
5                                                    ! update buffer pointer
5                 SRC [DSC$W_LENGTH] = .SRC [DSC$W_LENGTH] - .LEN;
5                                                    ! buffer len must be decreased
5                                                    !  by amt moved so check can be
5                                                    !  done so that further moves
5                                                    !  will not overflow buffer
4                 END;
4             END
3         ELSE
```

```
                    BEGIN
                    !+
                    ! dynamic strings
                    !-
                    LOCAL
                        DEST_ITEM : REF BLOCK [8, BYTE],! desc for string within array
                        END_ADDR;                       ! addr of last elem of array

                    LEN = (IF .LENGTH LSS 0 THEN K_DEFAULT_STR_LEN ELSE .LENGTH);
                                                        ! dest length depends on if
                                                        !  LENGTH parameter given,
                                                        !  defaults to 16
                    DEST_ITEM = .DEST [DSC$A_POINTER];  ! set up desc for actual item
                    IF .DEST_ITEM [DSC$B_DTYPE] EQL DSC$K_DTYPE_Z
                    THEN
                        LEN = .DEST_ITEM [DSC$W_LENGTH];! if we are dealing with an
                                                        !  array of RFAs or RECORDs,
                                                        !  get the length from the descr

                    END_ADDR = .DEST [DSC$A_POINTER] + .DEST [DSC$L_ARSIZE] -
                        .DEST [DSC$W_LENGTH];
                    !+
                    ! Loop thru array moving each element from buffer to it.
                    !-
                    INCR DSC_PTR FROM .DEST [DSC$A_POINTER] TO .END_ADDR
                        BY .DEST [DSC$W_LENGTH] DO
                        BEGIN

                        MAP
                            DSC_PTR: REF BLOCK [8,BYTE];! desc for string within array

                        IF (.SRC [DSC$W_LENGTH] LSSU .LEN)
                        THEN
                            BAS$$STOP_IO (BAS$K_MOVOVEBUF);
                        STR$COPY_R (.DSC_PTR, LEN, .SRC [DSC$A_POINTER]);
                                                        ! move string from buffer to
                                                        !  array, null padding if
                                                        !  necessary
                        SRC [DSC$A_POINTER] = .SRC [DSC$A_POINTER] + .LEN;
                                                        ! update buffer pointer
                        SRC [DSC$W_LENGTH] = .SRC [DSC$W_LENGTH] - .LEN;
                                                        ! buffer len must be decreased
                                                        !  by amt moved so check can be
                                                        !  done so that further moves
                                                        !  will not overflow buffer
                        END;
                    END
                END
            ELSE
                IF .DEST [DSC$B_CLASS] EQL DSC$K_CLASS_BFA
                THEN
                    BEGIN
                    !+
                    ! virtual array
                    !-
                    LOCAL
                        SRC_LEN,                        ! buffer element length
```

```
3                    MEM_LOC;                       ! virtual memory pointer
3
3               LIB$GET_VM (DEST [DSC$L_ARSIZE], MEM_LOC);
3                                                   ! alloc memory for virtual array
3               SRC_LEN =
3                   (IF .LENGTH LSS 0 THEN K_DEFAULT_STR_LEN ELSE .LENGTH);
3                                                   ! dest length depends on if
3                                                   !  LENGTH parameter given,
3                                                   !  defaults to 16
3               LEN = .DEST [DSC$W_LENGTH];         ! array element length
3               !+
3               ! If we are dealing with an array of RFAs or RECORDs,
3               ! LEN must be changed to a power of 2 since that is the
3               ! virtual array element's true size.
3               !-
3               IF .DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_Z
3               THEN
4                   BEGIN
4
5                   LEN = (INCR I FROM 0 TO 9
5                   DO
5                       IF .LEN LEQ .POWER_OF_2 [.I]
5                       THEN
4                           EXITLOOP .POWER_OF_2 [.I]);
4                   SRC_LEN = .DEST [DSC$W_LENGTH];
4
3                   END;
3
4               IF (.LEN GTRU .SRC [DSC$W_LENGTH])
3               THEN
3                   BAS$$STOP_IO (BAS$K_MOVOVEBUF);
3               !+
3               ! Loop thru array moving each element from buffer to it.
3               !-
4               INCR PTR FROM .MEM_LOC TO (.MEM_LOC + .DEST [DSC$L_ARSIZE] -
3                   .LEN) BY .LEN DO
4                   BEGIN
4                   CH$COPY (.SRC_LEN, .SRC [DSC$A_POINTER],
4                       %X'0', .LEN, .PTR);         ! move string from buffer to
4                                                   !  array, null padding if
4                                                   !  necessary
4                   SRC [DSC$A_POINTER] = .SRC [DSC$A_POINTER] + .SRC_LEN;
4                                                   ! update buffer pointer
4                   SRC [DSC$W_LENGTH] = .SRC [DSC$W_LENGTH] - .SRC_LEN;
4                                                   ! buffer len must be decreased
4                                                   !  by amt moved so check can be
4                                                   !  done so that further moves
4                                                   !  will not overflow buffer
3                   END;
3
3               BAS$$WHOLE_VA_STORE (.DEST, .MEM_LOC);
3                                                   ! store array from memory
3               LIB$FREE_VM (DEST [DSC$L_ARSIZE], MEM_LOC);
3                                                   ! dealloc memory used for array
2               END;
2
1           END;                                    ! of DEST_SA
```

```
                                                                   004F1           .BLKB   3
0200  0100  0080  0040  0020  0010  0008  0004  0002  0001         004F4 P.AAB:    .WORD   1, 2, 4, 8, 16, 32, 64, 128, 256, 512                    ;


                                                      07FC  00000 DEST_SA:.WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10          ; 1687
                              5E               20  C2  00002           SUBL2   #32, SP                                     ;
              0C    AE    E3  AF               14  28  00005           MOVC3   #20, P.AAB, POWER_OF_2                     ; 1741
                              57         08    AC  D0  0000B           MOVL    DEST, R7                                    ; 1751
                              04         03    A7  91  0000F           CMPB    3(R7), #4                                   ;
                                               03  13  00013           BEQL    1$                                          ;
                                             00D1  31  00015           BRW     15$                                         ;
                              51         0C    AC  D0  00018 1$:       MOVL    LENGTH, R1                                  ; 1766
                              56         04    AC  D0  0001C           MOVL    SRC, R6                                     ; 1785
                              58         04    A6  9E  00020           MOVAB   4(R6), R8                                   ; 1788
                              18         02    A7  91  00024           CMPB    2(R7), #24                                  ; 1757
                                               58  13  00028           BEQL    8$                                          ;
                                               51  D5  0002A           TSTL    R1                                          ; 1766
                                               05  18  0002C           BGEQ    2$                                          ;
                              50               10  D0  0002E           MOVL    #16, R0                                     ;
                                               03  11  00031           BRB     3$                                          ;
                              50               51  D0  00033 2$:       MOVL    R1, R0                                      ;
                        04    AE               50  D0  00036 3$:       MOVL    R0, LEN                                     ;
                                         02    A7  95  0003A           TSTB    2(R7)                                       ; 1770
                                               04  12  0003D           BNEQ    4$                                          ;
                        04    AE               67  3C  0003F           MOVZWL  (R7), LEN                                   ; 1772
                  50    04    A7         0C    A7  C1  00043 4$:       ADDL3   12(R7), 4(R7), R0                           ; 1776
                              59               67  3C  00049           MOVZWL  (R7), R9                                    ; 1777
                  6E          50               59  C3  0004C           SUBL3   R9, R0, END_ADDR                            ;
                              5A         04    A7  D0  00050           MOVL    4(R7), ELEM_PTR                             ; 1785
                                               26  11  00054           BRB     7$                                          ;
  04    AE        66          10               00  ED  00056 5$:       CMPZV   #0, #16, (R6), LEN                          ;
                                               0B  1E  0005C           BGEQU   6$                                          ;
                              7E        00G    8F  9A  0005E           MOVZBL  #BAS$K_MOVOVEBUF, -(SP)                     ; 1787
                   00000000G  00               01  FB  00062           CALLS   #1, BAS$$STOP_IO                            ;
        59        20          00  B8     04    AE  2C  00069 6$:       MOVC5   LEN, @0(R8), #32, R9, (ELEM_PTR)            ; 1788
                                  6A                   00070
                              68         04    AE  C0  00071           ADDL2   LEN, (R8)                                   ; 1791
                              66         04    AE  A2  00075           SUBW2   LEN, (R6)                                   ; 1793
                              5A               59  C0  00079           ADDL2   R9, ELEM_PTR                                ; 1781
                              6E               5A  D1  0007C 7$:       CMPL    ELEM_PTR, END_ADDR                          ;
                                               D5  15  0007F           BLEQ    5$                                          ;
                                                   04  00081           RET                                                 ; 1753
                                               51  D5  00082 8$:       TSTL    R1                                          ; 1809
                                               05  18  00084           BGEQ    9$                                          ;
                              50               10  D0  00086           MOVL    #16, R0                                     ;
                                               03  11  00089           BRB     10$                                         ;
                              50               51  D0  0008B 9$:       MOVL    R1, R0                                      ;
                        04    AE               50  D0  0008E 10$:      MOVL    R0, LEN                                     ;
                              50         04    A7  D0  00092           MOVL    4(R7), DEST_ITEM                            ; 1813
                                         02    A0  95  00096           TSTB    2(DEST_ITEM)                                ; 1814
                                               04  12  00099           BNEQ    11$                                         ;
                        04    AE               60  3C  0009B           MOVZWL  (DEST_ITEM), LEN                            ; 1816
                  50    04    A7         0C    A7  C1  0009F 11$:      ADDL3   12(R7), 4(R7), R0                           ; 1820
```

```
                                    54            67  3C 000A5        MOVZWL  (R7), R4                                    ; 1821
                      55            50            54  C3 000A8        SUBL3   R4, R0, END_ADDR                            ;
                                    52       04   AE  D0 000AC        MOVL    LEN, R2                                     ; 1832
                                    53       04   A7  D0 000B0        MOVL    4(R7), DSC_PTR                              ;
                                                  2D  11 000B4        BRB     14$                                         ;
            52        66            10            00  ED 000B6 12$:   CMPZV   #0, #16, (R6), R2                           ;
                                                  0B  1E 000BB        BGEQU   13$                                         ;
                                    7E       00G  8F  9A 000BD        MOVZBL  #BAS$K_MOVOVEBUF, -(SP)                     ; 1834
                         00000000G  00            01  FB 000C1        CALLS   #1, BAS$$STOP_IO                            ;
                                                  68  DD 000C8 13$:   PUSHL   (R8)                                        ; 1835
                                             08   AE  9F 000CA        PUSHAB  LEN                                         ;
                                                  53  DD 000CD        PUSHL   DSC_PTR                                     ;
                         00000000G  00            03  FB 000CF        CALLS   #3, STR$COPY_R                              ;
                                    52       04   AE  D0 000D6        MOVL    LEN, R2                                     ; 1839
                                    68            52  C0 000DA        ADDL2   R2, (R8)                                    ;
                                    66            52  A2 000DD        SUBW2   R2, (R6)                                    ; 1841
                                    53            54  C0 000E0        ADDL2   R4, DSC_PTR                                 ; 1825
                                    55            53  D1 000E3 14$:   CMPL    DSC_PTR, END_ADDR                           ;
                                                  CE  15 000E6        BLEQ    12$                                         ;
                                                      04 000E8        RET                                                 ; 1753
                              BF    8F       03   A7  91 000E9 15$:   CMPB    3(R7), #191                                 ; 1850
                                                  01  13 000EE        BEQL    16$                                         ;
                                                      04 000F0        RET                                                 ;
                                             08   AE  9F 000F1 16$:   PUSHAB  MEM_LOC                                     ; 1860
                                             0C   A7  9F 000F4        PUSHAB  12(R7)                                      ;
                         00000000G  00            02  FB 000F7        CALLS   #2, LIB$GET_VM                              ;
                                             0C   AC  D5 000FE        TSTL    LENGTH                                      ; 1863
                                                  05  18 00101        BGEQ    17$                                         ;
                                    59            10  D0 00103        MOVL    #16, SRC_LEN                                ;
                                                  04  11 00106        BRB     18$                                         ;
                                    59       0C   AC  D0 00108 17$:   MOVL    LENGTH, SRC_LEN                             ;
                              04    AE            67  3C 0010C 18$:   MOVZWL  (R7), LEN                                   ; 1867
                                             02   A7  95 00110        TSTB    2(R7)                                       ; 1873
                                                  23  12 00113        BNEQ    22$                                         ;
                                                  50  D4 00115        CLRL    I                                           ; 1879
                                          0C AE40 3F 00117 19$:      PUSHAW  POWER_OF_2[I]                               ;
     04   AE             9E         10            00  ED 0011B        CMPZV   #0, #16, @(SP)+, LEN                        ;
                                                  07  19 00121        BLSS    20$                                         ;
                                    50    0C AE40 3C 00123            MOVZWL  POWER_OF_2[I], R0                           ; 1881
                                                  07  11 00128        BRB     21$                                         ;
                         E9         50            09  F3 0012A 20$:   AOBLEQ  #9, I, 19$                                  ; 1879
                                    50            01  CE 0012E        MNEGL   #1, R0                                      ; 1877
                              04    AE            50  D0 00131 21$:   MOVL    R0, LEN                                     ;
                                    59            67  3C 00135        MOVZWL  (R7), SRC_LEN                               ; 1882
     04   AE        04   BC         10            00  ED 00138 22$:   CMPZV   #0, #16, @SRC, LEN                          ; 1886
                                                  0B  1E 0013F        BGEQU   23$                                         ;
                                    7E       00G  8F  9A 00141        MOVZBL  #BAS$K_MOVOVEBUF, -(SP)                     ; 1888
                         00000000G  00            01  FB 00145        CALLS   #1, BAS$$STOP_IO                            ;
                         5A   08    AE       0C   A7  C1 0014C 23$:   ADDL3   12(R7), MEM_LOC, R10                        ; 1892
                                    5A       04   AE  C2 00152        SUBL2   LEN, R10                                    ; 1893
                                    56       04   AC  D0 00156        MOVL    SRC, R6                                     ; 1895
                                    58       08   AE  D0 0015A        MOVL    MEM_LOC, PTR                                ;
                                                  14  11 0015E        BRB     25$                                         ;
     04   AE             00   04    B6            59  2C 00160 24$:   MOVC5   SRC_LEN, @4(R6), #0, LEN, (PTR)             ; 1896
                                                  68     00167                                                            ;
                              04    A6            59  C0 00168        ADDL2   SRC_LEN, 4(R6)                              ; 1899
                              04    BC            59  A2 0016C        SUBW2   SRC_LEN, @SRC                               ; 1901
```

```
              58      04  AE  C0 00170        ADDL2   LEN, PTR                          ; 1892
              5A          58  D1 00174 25$:   CMPL    PTR, R10                          ;
                          E7  15 00177        BLEQ    24$                               ;
                      08  AE  DD 00179        PUSHL   MEM_LOC                           ; 1908
                          57  DD 0017C        PUSHL   R7                                ;
    00000000G 00          02  FB 0017E        CALLS   #2, BAS$$WHOLE_VA_STORE           ;
                      08  AE  9F 00185        PUSHAB  MEM_LOC                           ; 1910
                      0C  A7  9F 00188        PUSHAB  12(R7)                            ;
    00000000G 00          02  FB 0018B        CALLS   #2, LIB$FREE_VM                   ;
                              04 00192        RET                                       ; 1914

; Routine Size:  403 bytes,    Routine Base:  _BAS$CODE + 0508
```

```
ROUTINE DEST_DSC (                                      ! MOVE a descriptor
        SRC : REF BLOCK [8, BYTE],                      ! Source: buffer
        DEST : REF BLOCK [, BYTE],                      ! Destination: array
        LENGTH                                          ! -1 or length of each string
    ) : CALL_CCB NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Within a MOVE statement, move the I/O buffer to an array of
!       descriptors.  The descriptors may be string descriptors or
!       descriptors for dynamic variables (probably numeric).  So
!       determine the dtype of the descriptor and then call SRC_SA or
!       SRC_NA to perform the work.
!
! FORMAL PARAMETERS:
!
!       SRC.mq.r        The source, the I/O buffer.  This is updated to account for
!                       the bytes taken from it.
!       DEST.wx.da      The destination, an array of descriptors
!
! IMPLICIT INPUTS:
!
!       CCB, which is used only to provide good error messages.
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
! COMPLETION CODES:
!
!       NONE
!
! SIDE EFFECTS:
!
!       Signals if an error is encountered.
!
!--

    BEGIN

    EXTERNAL REGISTER
        CCB : REF BLOCK [, BYTE];

    LOCAL
        ELEM_DSC : REF BLOCK [8,BYTE],
        NUM_ELEMS;

!+
! Set up pointer to the first element descriptor in the array so that the
! dtype and length can be determined.
!-

    ELEM_DSC = .DEST [DSC$A_A0];

```

```
; 1152      1972  2 !+
; 1153      1973  2 ! Now just dispatch to the appropriate routine based on the dtype in the
; 1154      1974  2 ! descriptor element.
; 1155      1975  2 !-
; 1156      1976  2
; 1157      1977  2     SELECTONEU .ELEM_DSC [DSC$B_DTYPE] OF
; 1158      1978  2         SET
; 1159      1979  2
; 1160      1980  2         [DSC$K_DTYPE_B, DSC$K_DTYPE_W, DSC$K_DTYPE_L, DSC$K_DTYPE_F,
; 1161      1981  2          DSC$K_DTYPE_D, DSC$K_DTYPE_G, DSC$K_DTYPE_H, DSC$K_DTYPE_P]:
; 1162      1982  3             BEGIN                                   ! dynamically mapped array
; 1163      1983  3             !+
; 1164      1984  3             ! Check for overflowing the buffer here to save doing it in
; 1165      1985  3             ! DEST_NA.
; 1166      1986  3             !-
; 1167      1987  3             NUM_ELEMS = .DEST [DSC$L_ARSIZE]/ .DEST [DSC$W_LENGTH];
; 1168      1988  4             IF .SRC [DSC$W_LENGTH] LSSU (.NUM_ELEMS * .ELEM_DSC [DSC$W_LENGTH])
; 1169      1989  3             THEN
; 1170      1990  3                 BAS$$STOP_IO (BAS$K_MOVOVEBUF);
; 1171      1991  3             DEST_NA (.SRC, .DEST);
; 1172      1992  2             END;
; 1173      1993  2
; 1174      1994  2         [DSC$K_DTYPE_T, DSC$K_DTYPE_Z]:
; 1175      1995  2             DEST_SA (.SRC, .DEST, .LENGTH);       ! string or record array
; 1176      1996  2
; 1177      1997  2         [OTHERWISE]:
; 1178      1998  2             BAS$$STOP_IO (BAS$K_PROLOSSOR);
; 1179      1999  2
; 1180      2000  2         TES;
; 1181      2001  2
; 1182      2002  1     END;                                            ! of DEST_DSC



                               001C 00000 DEST_DSC:
                                                  .WORD   Save R2,R3,R4                        ; 1915
          54 00000000G   00   9E 00002            MOVAB   BAS$$STOP_IO, R4
          52         08  AC   D0 00009            MOVL    DEST, R2                             ; 1970
          51         10  A2   D0 0000D            MOVL    16(R2), ELEM_DSC
          50         02  A1   9A 00011            MOVZBL  2(ELEM_DSC), R0                      ; 1977
          06             50   91 00015            CMPB    R0, #6                               ; 1980
                         05   1F 00018            BLSSU   1$                                   ;
          08             50   91 0001A            CMPB    R0, #8                               ;
                         19   1B 0001D            BLEQU   3$                                   ;
          0A             50   91 0001F 1$:        CMPB    R0, #10                              ;
                         05   1F 00022            BLSSU   2$                                   ;
          0B             50   91 00024            CMPB    R0, #11                              ;
                         0F   1B 00027            BLEQU   3$                                   ;
          15             50   91 00029 2$:        CMPB    R0, #21                              ;
                         0A   13 0002C            BEQL    3$                                   ;
          1B             50   91 0002E            CMPB    R0, #27                              ;
                         2D   1F 00031            BLSSU   5$                                   ;
          1C             50   91 00033            CMPB    R0, #28                              ;
                         28   1A 00036            BGTRU   5$                                   ;
          50             62   3C 00038 3$:        MOVZWL  (R2), NUM_ELEMS                      ; 1987
```

```
                    50      0C   A2              50  C7 0003B          DIVL3   NUM_ELEMS, 12(R2), NUM_ELEMS          ;
                                 53              61  3C 00040          MOVZWL  (ELEM_DSC), R3                        ; 1988
                                 50              53  C4 00043          MULL2   R3, R0                                ;
        50      04      BC       10              00  ED 00046          CMPZV   #0, #16, @SRC, R0                     ;
                                                 07  1E 0004C          BGEQU   4$                                    ;
                                 7E        00G   8F  9A 0004E          MOVZBL  #BAS$K_MOVOVEBUF, -(SP)               ; 1990
                                 64              01  FB 00052          CALLS   #1, BAS$$STOP_IO                      ;
                                                 52  DD 00055 4$:      PUSHL   R2                                    ; 1991
                                           04    AC  DD 00057          PUSHL   SRC                                   ;
                         FD23    CF              02  FB 0005A          CALLS   #2, DEST_NA                           ;
                                                     04 0005F          RET                                           ; 1977
                                                 50  D5 00060 5$:      TSTL    R0                                    ; 1994
                                                 05  13 00062          BEQL    6$                                    ;
                                 0E              50  91 00064          CMPB    R0, #14                               ;
                                                 0E  12 00067          BNEQ    7$                                    ;
                                           0C    AC  DD 00069 6$:      PUSHL   LENGTH                                ; 1995
                                                 52  DD 0006C          PUSHL   R2                                    ;
                                           04    AC  DD 0006E          PUSHL   SRC                                   ;
                         FDF7    CF              03  FB 00071          CALLS   #3, DEST_SA                           ;
                                                     04 00076          RET                                           ;
                                 7E        00G   8F  9A 00077 7$:      MOVZBL  #BAS$K_PROLOSSOR, -(SP)               ; 1998
                                 64              01  FB 0007B          CALLS   #1, BAS$$STOP_IO                      ;
                                                     04 0007E          RET                                           ; 2002

; Routine Size:  127 bytes,    Routine Base:  _BAS$CODE + 069B

; 1183          2003  1 END                                                   ! end of module BAS$MOVE_ARRAY
; 1184          2004  1
; 1185          2005  0 ELUDOM
```

```
;                                   PSECT SUMMARY
;
;       Name                    Bytes                      Attributes
;
;   _BAS$CODE                      1818  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       20         0          581          00:01.2
```

```
;                                       COMMAND QUALIFIERS

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

; Size:          1773 code + 45 data bytes
; Run Time:         00:37.0
; Elapsed Time:     01:18.8
; Lines/CPU Min:     3248
; Lexemes/CPU-Min: 25224
; Memory Used:  220 pages
; Compilation Complete
```

BASMID
LIS

BASMULDDI
LIS

BASNOTIMP
LIS

BASMOVEAR
LIS

BASMSGDEF
LIS

BASMSGGEN
LIS

BASONECHR
LIS

BASMOVE
LIS

BASNUM
LIS

BASNAMEAS
LIS

BASNUM1
LIS