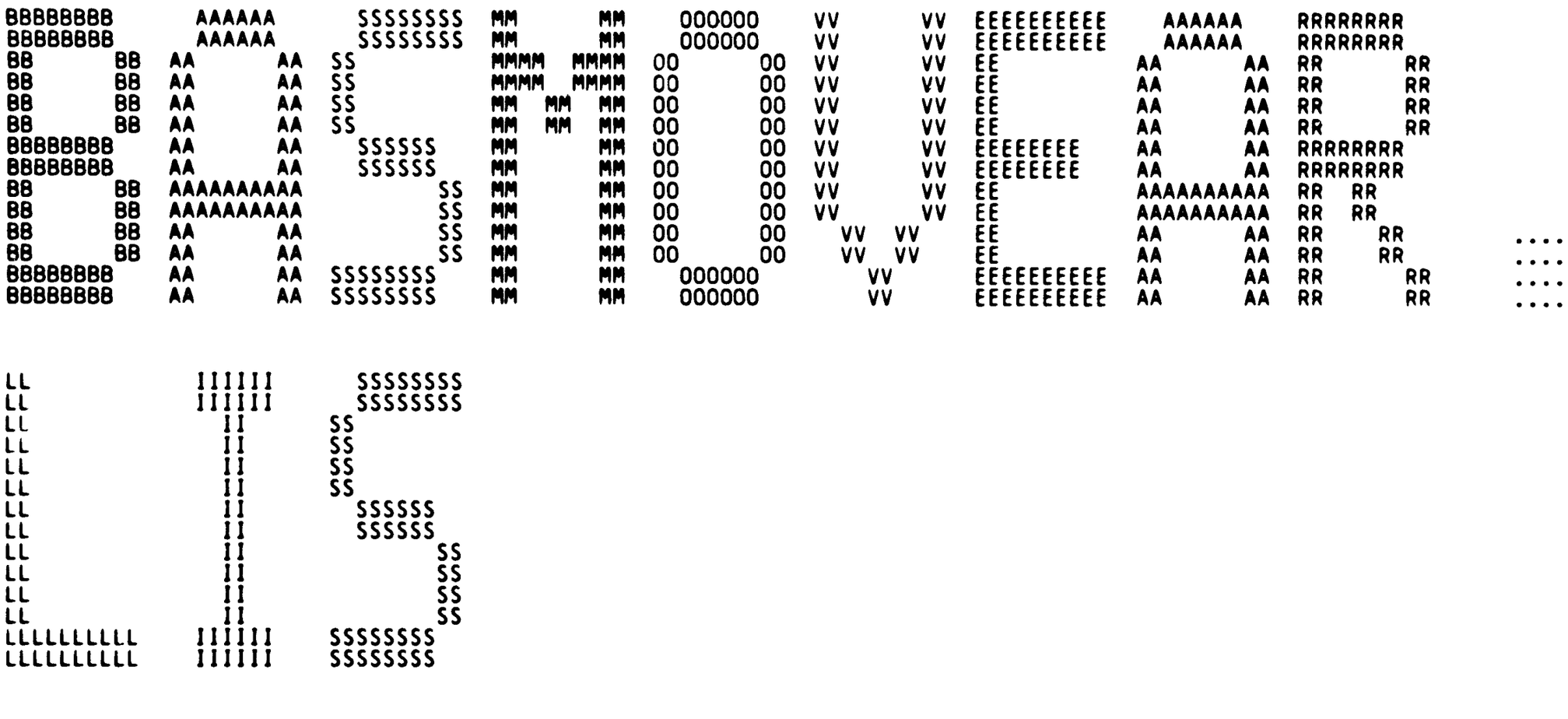
```
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBB         BBB   AAA         AAA  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


BBBBBBBB AAAAAA SSSSSSSS MM MM 000000 VV VV EEEEEEEEEE AAAAAA RRRRRRRR
BBBBBBBB AAAAAA SSSSSSSS MM MM 000000 VV VV EEEEEEEEEE AAAAAA RRRRRRRR
BB BB AA AA SS MMMM MMMM 00 00 VV VV EE AA AA RR RR
BB BB AA AA SS MMMM MMMM 00 00 VV VV EE AA AA RR RR
BB BB AA AA SS MM MM MM 00 00 VV VV EE AA AA RR RR
BB BB AA AA SS MM MM MM 00 00 VV VV EE AA AA RR RR
BBBBBBBB AA AA SSSSSS MM MM 00 00 VV VV EEEEEEEE AA AA RRRRRRRR
BBBBBBBB AA AA SSSSSS MM MM 00 00 VV VV EEEEEEEE AA AA RRRRRRRR
BB BB AAAAAAAAAA SS MM MM 00 00 VV VV EE AAAAAAAAAA RR RR
BB BB AAAAAAAAAA SS MM MM 00 00 VV VV EE AAAAAAAAAA RR RR
BB BB AA AA SS MM MM 00 00 VV VV EE AA AA RR RR
BB BB AA AA SS MM MM 00 00 VV VV EE AA AA RR RR
BBBBBBBB AA AA SSSSSSSS MM MM 000000 VV EEEEEEEEEE AA AA RR RR
BBBBBBBB AA AA SSSSSSSS MM MM 000000 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

```
;     1          0001  0 MODULE BAS$MOVE_ARRAY (                       !
;     2          0002  0                     IDENT = '1-020'           ! File: BASMOVEAR.B32 Edit: MDL1020
;     3          0003  0                     ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !*****************************************************************************
;     7          0007  1 !*                                                                           *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                      *
;    11          0011  1 !*                                                                           *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    17          0017  1 !*  TRANSFERRED.                                                              *
;    18          0018  1 !*                                                                           *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
;    21          0021  1 !*  CORPORATION.                                                              *
;    22          0022  1 !*                                                                           *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*****************************************************************************
;    28          0028  1 !
;    29          0029  1
;    30          0030  1 !++
;    31          0031  1 ! FACILITY:  BASIC-PLUS-2 Miscellaneous I/O
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:
;    34          0034  1 !
;    35          0035  1 !       This module contains the routines called by compiled code
;    36          0036  1 !       for the MOVE FROM and MOVE TO statements, when applied to
;    37          0037  1 !       arrays.
;    38          0038  1 !
;    39          0039  1 ! ENVIRONMENT:  VAX-11 User Mode
;    40          0040  1 !
;    41          0041  1 ! AUTHOR: John Sauter, CREATION DATE: 30-AUG-1979
;    42          0042  1 !
;    43          0043  1 ! MODIFIED BY:
;    44          0044  1 !
;    45          0045  1 ! 1-001 - Original.
;    46          0046  1 ! 1-002 - Version 001 was just a skeleton, code single-dimensioned
;    47          0047  1 !           numeric arrays moving TO the buffer, so we can see how
;    48          0048  1 !           the algorithms work.  JBS 31-AUG-1979
;    49          0049  1 ! 1-003 - Code DEST_NA.  JBS 06-SEP-1979
;    50          0050  1 ! 1-004 - Code SRC_SA.  JBS 10-SEP-1979
;    51          0051  1 ! 1-005 - Code the other effectors.  All coding is now complete.
;    52          0052  1 !           JBS 13-SEP-1979
;    53          0053  1 ! 1-006 - When initializing a string descriptor, be sure the pointer
;    54          0054  1 !           is zero.  JBS 18-SEP-1979
;    55          0055  1 ! 1-007 - When the source is a string array and a length is specified,
;    56          0056  1 !           always put the specified number of characters in the buffer.
;    57          0057  1 !           JBS 22-OCT-1979
```

```
;    58          0058  1 ! 1-008 - SRC_NA, SRC_SA, DEST_NA, and DEST_SA must access the array
;    59          0059  1 !         in row major order.  PL 1-Jul-81
;    60          0060  1 ! 1-009 - Add support for records.  MOVE_ARRAY, SRC_SA, and DEST_SA
;    61          0061  1 !         have been modified for dtype Z.  PLL 26-Feb-82
;    62          0062  1 ! 1-010 - Add support for dynamically mapped arrays.  New entry points
;    63          0063  1 !         have been added, SRC_DSC and DEST_DSC.  PLL 2-Mar-82
;    64          0064  1 ! 1-011 - Add support for decimal arrays.  PLL 12-Mar-82
;    65          0065  1 ! 1-012 - Don't try to free value_descrip in SRC_DSC and DEST_DSC.
;    66          0066  1 !         Also, DEST_SA (and SRC_SA) should set the dtype of the src buffer to Z only
;    67          0067  1 !         if the array is dtype Z.  BAS$STORE_BFA requires the two data
;    68          0068  1 !         types to match.  PLL 19-Mar-1982
;    69          0069  1 ! 1-013 - Add support for multiply dimensioned arrays.  PLL 5-Apr-1982
;    70          0070  1 ! 1-014 - Fix CH$MOVE in DEST_NA.  PLL 4-May-1982
;    71          0071  1 ! 1-015 - Fix DEST_SA.  STR$COPY_R expects a descriptor for the destination,
;    72          0072  1 !         and this descriptor must be constructed for static string arrays.
;    73          0073  1 !         PLL 5-May-1982
;    74          0074  1 ! 1-016 - Fix DEST_SA again.  CH$MOVE updates destination pointer, not source
;    75          0075  1 !         pointer.  PLL 10-May-1982
;    76          0076  1 ! 1-017 - 1. Fix SRC_SA for virtual arrays - nulls are stripped off on a per
;    77          0077  1 !         element basis.
;    78          0078  1 !         2. Fix DEST_SA for virtual arrays - dest len. must reflect length
;    79          0079  1 !         given in statement or be default string length.
;    80          0080  1 !         3. Fix DEST_SA for arrays of RFAs or RECORDs - length found in
;    81          0081  1 !         descriptor must be used.  DG 25-Jan-1984
;    82          0082  1 ! 1-018 - Fix 1-017 (2).  By simply null-filling the rest of the virtual
;    83          0083  1 !         array element, the LEN function will work correctly.  DG 14-Feb-1984
;    84          0084  1 ! 1-019 - Fixed some bugs in SRC_SA:
;    85          0085  1 !         1. Virt arrays of RFAs or RECORDs must have their element length
;    86          0086  1 !         be a power of 2 (not found when looking in descriptor)
;    87          0087  1 !         2. Virt arrays of RFAs or RECORDs should not go thru the null
;    88          0088  1 !         stripping procedure, again because the length in the descriptor is the
;    89          0089  1 !         actual element length.
;    90          0090  1 !         3. CH$COPY is used for virtual arrays (instead of CH$MOVE), so that
;    91          0091  1 !         if a LENGTH is specified and the element length is less, blank
;    92          0092  1 !         padding will be accomplished
;    93          0093  1 ! 1-020 - on a MOVE FROM to a static string array, if the length is less than
;    94          0094  1 !         then element length, we should blank fill.  MDL 12-Apr-1984
;    95          0095  1 !--
;    96          0096  1
;    97          0097  1 !<BLF/PAGE>
```

```
;   99          0098  1 !
;  100          0099  1 ! SWITCHES:
;  101          0100  1 !
;  102          0101  1
;  103          0102  1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;  104          0103  1
;  105          0104  1 !
;  106          0105  1 ! LINKAGES:
;  107          0106  1 !
;  108          0107  1
;  109          0108  1 REQUIRE 'RTLIN:OTSLNK';                         ! Define linkages
;  110          0537  1
;  111          0538  1 !
;  112          0539  1 ! TABLE OF CONTENTS:
;  113          0540  1 !
;  114          0541  1
;  115          0542  1 FORWARD ROUTINE
;  116          0543  1     BAS$MOVE_ARRAY : NOVALUE,                   ! MOVE an array
;  117          0544  1     SRC_NA : CALL_CCB NOVALUE,                  ! Source is numeric array
;  118          0545  1     SRC_SA : CALL_CCB NOVALUE,                  ! Source is string array
;  119          0546  1     DEST_NA : CALL_CCB NOVALUE,                 ! Destination is numeric array
;  120          0547  1     DEST_SA : CALL_CCB NOVALUE,                 ! Destination is string array
;  121          0548  1     SRC_DSC : CALL_CCB NOVALUE,                 ! Source is descriptor
;  122          0549  1     DEST_DSC : CALL_CCB NOVALUE;                ! Destination is descriptor
;  123          0550  1
;  124          0551  1 !
;  125          0552  1 ! INCLUDE FILES:
;  126          0553  1 !
;  127          0554  1
;  128          0555  1 REQUIRE 'RTLML:OTSLUB';                         ! LUB definitions
;  129          0695  1
;  130          0696  1 REQUIRE 'RTLML:OTSISB';                         ! ISB definitions
;  131          0864  1
;  132          0865  1 REQUIRE 'RTLIN:RTLPSECT';                       ! Macros for defining psects
;  133          0960  1
;  134          0961  1 LIBRARY 'RTLSTARLE';                            ! System symbols
;  135          0962  1
;  136          0963  1 !
;  137          0964  1 ! MACROS:
;  138          0965  1 !
;  139          0966  1 !       NONE
;  140          0967  1 !
;  141          0968  1 ! EQUATED SYMBOLS:
;  142          0969  1 !
;  143          0970  1 !       NONE
;  144          0971  1 !
;  145          0972  1 ! PSECTS:
;  146          0973  1 !
;  147          0974  1 DECLARE_PSECTS (BAS);                           ! Declare psects for BAS$ facility
;  148          0975  1 !
;  149          0976  1 ! OWN STORAGE:
;  150          0977  1 !
;  151          0978  1 !       NONE
;  152          0979  1 !
;  153          0980  1 ! EXTERNAL REFERENCES:
;  154          0981  1 !
;  155          0982  1
```

```
;  156          0983  1 EXTERNAL ROUTINE
;  157          0984  1     BAS$$CB_GET : JSB_CB_GET NOVALUE,          ! Load current CCB
;  158          0985  1     BAS$$STOP : NOVALUE,                       ! Signal fatal error
;  159          0986  1     BAS$$STOP_IO : NOVALUE,                    ! Signal fatal I/O error
;  160          0987  1     BAS$$WHOLE_VA_FETCH : NOVALUE,             ! Fetch entire virtual array
;  161          0988  1     BAS$$WHOLE_VA_STORE : NOVALUE,             ! Store entire virtual array
;  162          0989  1     LIB$FREE_VM,                               ! Dealloc memory
;  163          0990  1     LIB$GET_VM,                                ! Alloc memory
;  164          0991  1     STR$COPY_R;                                ! Copy a string by reference
;  165          0992  1 !+
;  166          0993  1 ! The following are the error codes used in this module.
;  167          0994  1 !-
;  168          0995  1
;  169          0996  1 EXTERNAL LITERAL
;  170          0997  1     BAS$K_PROLOSSOR : UNSIGNED (8),            ! Program lost, sorry
;  171          0998  1     BAS$K_MOVOVEBUF : UNSIGNED (8),            ! MOVE overflows buffer
;  172          0999  1     BAS$K_NOTIMP : UNSIGNED (8);               ! Not implemented
;  173          1000  1
```

```
;  175          1001  1 GLOBAL ROUTINE BAS$MOVE_ARRAY (                      ! MOVE an array
;  176          1002  1         SRC : REF BLOCK [8, BYTE],                  ! Source: buffer or array
;  177          1003  1         DEST : REF BLOCK [8, BYTE],                 ! Destination: array or buffer
;  178          1004  1         LEN                                         ! Optional length, for strings
;  179          1005  1     ) : NOVALUE =
;  180          1006  1
;  181          1007  1 !++
;  182          1008  1 ! FUNCTIONAL DESCRIPTION:
;  183          1009  1 !
;  184          1010  1 !       Within a MOVE statement, move an array.  This routines decodes its
;  185          1011  1 !       parameters to move either an array to the buffer, or the buffer
;  186          1012  1 !       to an array.
;  187          1013  1 !
;  188          1014  1 ! FORMAL PARAMETERS:
;  189          1015  1 !
;  190          1016  1 !       SRC.mq.r        The source, which has the form of either a static
;  191          1017  1 !                       string (the buffer) or an array.
;  192          1018  1 !       DEST.mq.r       The destination, which has the other form.
;  193          1019  1 !       LEN.rl.v        Optional length, used with strings.
;  194          1020  1 !
;  195          1021  1 ! IMPLICIT INPUTS:
;  196          1022  1 !
;  197          1023  1 !       OTS$$A_CUR_LUB, accessed using BAS$$CB_GET, points to the current
;  198          1024  1 !                       LUB.
;  199          1025  1 !
;  200          1026  1 ! IMPLICIT OUTPUTS:
;  201          1027  1 !
;  202          1028  1 !       NONE
;  203          1029  1 !
;  204          1030  1 ! ROUTINE VALUE:
;  205          1031  1 ! COMPLETION CODES:
;  206          1032  1 !
;  207          1033  1 !       NONE
;  208          1034  1 !
;  209          1035  1 ! SIDE EFFECTS:
;  210          1036  1 !
;  211          1037  1 !       Signals if an error is encountered.
;  212          1038  1 !       Updates the buffer's address and count to reflect the movement
;  213          1039  1 !       of the array.
;  214          1040  1 !
;  215          1041  1 !--
;  216          1042  1
;  217          1043  2     BEGIN
;  218          1044  2
;  219          1045  2     GLOBAL REGISTER
;  220          1046  2         CCB = K_CCB_REG : REF BLOCK [, BYTE];
;  221          1047  2
;  222          1048  2     BUILTIN
;  223          1049  2         ACTUALCOUNT;
;  224          1050  2
;  225          1051  2     LOCAL
;  226          1052  2         LENGTH;                                     ! Passed length, or -1
;  227          1053  2
;  228          1054  2     LENGTH = (IF (ACTUALCOUNT () GEQ 3) THEN .LEN ELSE -1);
;  229          1055  2 !+
;  230          1056  2 ! Set up register CCB for error messages.
;  231          1057  2 !-
```

```
;  232          1058  2     BAS$$CB_GET ();
;  233          1059  2 !+
;  234          1060  2 ! If we are not doing I/O, or if the I/O is not a MOVE statement,
;  235          1061  2 ! then the compiled code has called this routine in the wrong context.
;  236          1062  2 !-
;  237          1063  2
;  238          1064  2     IF (.CCB EQLA 0) THEN BAS$$STOP (BAS$K_PROLOSSOR);
;  239          1065  2
;  240          1066  2     IF ( NOT .CCB [LUB$V_OPENED]) THEN BAS$$STOP_IO (BAS$K_PROLOSSOR);
;  241          1067  2
;  242          1068  2     IF (.CCB [ISB$B_STTM_TYPE] NEQ ISB$K_ST_TY_MOV) THEN BAS$$STOP_IO (BAS$K_PROLOSSOR);
;  243          1069  2
;  244          1070  2 !+
;  245          1071  2 ! Dispatch based on the class and data type of the source and destination
;  246          1072  2 !-
;  247          1073  2
;  248          1074  2     SELECTONE .SRC [DSC$B_CLASS] OF
;  249          1075  2         SET
;  250          1076  2
;  251          1077  2         [DSC$K_CLASS_A, DSC$K_CLASS_BFA] :
;  252          1078  3             BEGIN                                   ! Source is array
;  253          1079  3
;  254          1080  3             IF (.DEST [DSC$B_CLASS] NEQ DSC$K_CLASS_S) THEN BAS$$STOP_IO (BAS$K_PROLOSSOR);
;  255          1081  3
;  256          1082  3             SELECTONE .SRC [DSC$B_DTYPE] OF
;  257          1083  3                 SET
;  258          1084  3
;  259          1085  3                 [DSC$K_DTYPE_B, DSC$K_DTYPE_W, DSC$K_DTYPE_L, DSC$K_DTYPE_F,
;  260          1086  3                  DSC$K_DTYPE_D, DSC$K_DTYPE_G, DSC$K_DTYPE_H, DSC$K_DTYPE_P] :
;  261          1087  4                     BEGIN                           ! Source is numeric array
;  262          1088  4                     SRC_NA (.SRC, .DEST);
;  263          1089  3                     END;
;  264          1090  3
;  265          1091  3                 [DSC$K_DTYPE_T, DSC$K_DTYPE_Z] :
;  266          1092  4                     BEGIN                           ! Source is string array or record
;  267          1093  4                     SRC_SA (.SRC, .DEST, .LENGTH);
;  268          1094  3                     END;
;  269          1095  3
;  270          1096  3                 [DSC$K_DTYPE_DSC] :
;  271          1097  4                     BEGIN                           ! Source is descriptor
;  272          1098  4                     SRC_DSC (.SRC, .DEST, .LENGTH);
;  273          1099  3                     END;
;  274          1100  3
;  275          1101  3                 [OTHERWISE] :
;  276          1102  3                     BAS$$STOP_IO (BAS$K_PROLOSSOR);
;  277          1103  3                 TES;
;  278          1104  3
;  279          1105  2             END;
;  280          1106  2
;  281          1107  2         [DSC$K_CLASS_S] :
;  282          1108  3             BEGIN                                   ! Source is buffer
;  283          1109  3
;  284          1110  3             SELECTONE .DEST [DSC$B_CLASS] OF
;  285          1111  3                 SET
;  286          1112  3
;  287          1113  3                 [DSC$K_CLASS_A, DSC$K_CLASS_BFA] :
;  288          1114  4                     BEGIN                           ! Destination is array
```

```
;  289          1115  4
;  290          1116  4                SELECTONE .DEST [DSC$B_DTYPE] OF
;  291          1117  4                    SET
;  292          1118  4
;  293          1119  4                    [DSC$K_DTYPE_B, DSC$K_DTYPE_W, DSC$K_DTYPE_L,
;  294          1120  4                     DSC$K_DTYPE_F, DSC$K_DTYPE_D, DSC$K_DTYPE_G,
;  295          1121  4                     DSC$K_DTYPE_H, DSC$K_DTYPE_P] :
;  296          1122  5                        BEGIN                ! Destination is numeric array
;  297          1123  5                        DEST_NA (.SRC, .DEST);
;  298          1124  4                        END;
;  299          1125  4
;  300          1126  4                    [DSC$K_DTYPE_T, DSC$K_DTYPE_Z] :
;  301          1127  5                        BEGIN                ! Destination is string array
;  302          1128  5                        DEST_SA (.SRC, .DEST, .LENGTH);
;  303          1129  4                        END;
;  304          1130  4
;  305          1131  4                    [DSC$K_DTYPE_DSC] :
;  306          1132  5                        BEGIN                ! Destination is descriptor
;  307          1133  5                        DEST_DSC (.SRC, .DEST, .LENGTH);
;  308          1134  4                        END;
;  309          1135  4
;  310          1136  4                    [OTHERWISE] :
;  311          1137  4                        BAS$$STOP_IO (BAS$K_PROLOSSOR);
;  312          1138  4                    TES;
;  313          1139  4
;  314          1140  3                END;
;  315          1141  3
;  316          1142  3            [OTHERWISE] :
;  317          1143  3                BAS$$STOP_IO (BAS$K_PROLOSSOR);
;  318          1144  3            TES;
;  319          1145  3
;  320          1146  2        END;
;  321          1147  2
;  322          1148  2            [OTHERWISE] :
;  323          1149  2                BAS$$STOP_IO (BAS$K_PROLOSSOR);
;  324          1150  2            TES;
;  325          1151  2
;  326          1152  2        RETURN;
;  327          1153  1        END;                                    ! end of BAS$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$:   MNEGL   #1, LENGTH
                    00000000G   00  FB  0001B 2$:       JSB     BAS$$CB_GET                     ;1058
                                38  D5  0002?           TSTL    CCB                             ;1064
                                0A  12  00025           BNEQ    3$
                7E                  57  9A  00025       MOVZBL  R7, -(SP)
      00000000G 00                  01  FB  00028       CALLS   #1, BAS$$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
```

```
;  330     1155  1 ROUTINE SRC_NA (                                  ! MOVE a numeric array
;  331     1156  1         SRC : REF BLOCK [, BYTE],                  ! Source: array
;  332     1157  1         DEST : REF BLOCK [8, BYTE]                 ! Destination: buffer
;  333     1158  1     ) : CALL_CCB NOVALUE =
;  334     1159  1
;  335     1160  1 !++
;  336     1161  1 ! FUNCTIONAL DESCRIPTION:
;  337     1162  1 !
;  338     1163  1 !       Within a MOVE statement, move a numeric array to the buffer.
;  339     1164  1 !
;  340     1165  1 ! FORMAL PARAMETERS:
;  341     1166  1 !
;  342     1167  1 !       SRC.rx.da       The source, a numeric array.
;  343     1168  1 !       DEST.mq.r       The destination, the I/O buffer.  It is updated to
;  344     1169  1 !                       reflect the array.
;  345     1170  1 !
;  346     1171  1 ! IMPLICIT INPUTS:
;  347     1172  1 !
;  348     1173  1 !       CCB, which is used only to provide good error messages.
;  349     1174  1 !
;  350     1175  1 ! IMPLICIT OUTPUTS:
;  351     1176  1 !
;  352     1177  1 !       NONE
;  353     1178  1 !
;  354     1179  1 ! ROUTINE VALUE:
;  355     1180  1 ! COMPLETION CODES:
;  356     1181  1 !
;  357     1182  1 !       NONE
;  358     1183  1 !
;  359     1184  1 ! SIDE EFFECTS:
;  360     1185  1 !
;  361     1186  1 !       Signals if an error is encountered.
;  362     1187  1 !       Updates the buffer's address and count to reflect the movement
;  363     1188  1 !       of the array.
;  364     1189  1 !
;  365     1190  1 !--
;  366     1191  1
;  367     1192  2     BEGIN
;  368     1193  2
;  369     1194  2     EXTERNAL REGISTER
;  370     1195  2         CCB : REF BLOCK [, BYTE];
;  371     1196  2
;  372     1197  2     LOCAL
;  373     1198  2         LEN;                                       ! length of array elements
;  374     1199  2                                                    ! (needed for decimal)
;  375     1200  2 !+
;  376     1201  2 ! The total number of bytes in the array must not be greater than the
;  377     1202  2 ! remaining length of the buffer.  If this is an array of numeric descriptors,
;  378     1203  2 ! SRC_DSC has already checked for the MOVE overflowing the buffer.
;  379     1204  2 !-
;  380     1205  2
;  381     1206  2     IF (.SRC [DSC$B_DTYPE] NEQ DSC$K_DTYPE_DSC) AND
;  382     1207  3        (.SRC [DSC$L_ARSIZE] GTRU .DEST [DSC$W_LENGTH])
;  383     1208  2     THEN
;  384     1209  2          BAS$$STOP_IO (BAS$K_MOVOVEBUF);
;  385     1210  2
;  386     1211  2 !+
```

```
;  387          1212  2 ! A memory numeric array can be moved in one instruction.  A virtual array,
;  388          1213  2 ! however, must call the fetch routine.  A dynamically mapped numeric array
;  389          1214  2 ! is in memory but elements may not be stored contiguously.  Therefore if
;  390          1215  2 ! this is an array of descriptors, perform the move element by element.
;  391          1216  2 !-
;  392          1217  2
;  393          1218  2     IF .SRC [DSC$B_CLASS] EQL DSC$K_CLASS_A
;  394          1219  2     THEN
;  395          1220  3         BEGIN                                       ! memory array
;  396          1221  3         IF .SRC [DSC$B_DTYPE] EQL DSC$K_DTYPE_DSC
;  397          1222  3         THEN
;  398          1223  4             BEGIN                                   ! dynamically mapped array
;  399          1224  4             LOCAL
;  400          1225  4                 END_ADDR;
;  401          1226  4             END_ADDR = .SRC [DSC$A_POINTER] + .SRC [DSC$L_ARSIZE] - .SRC [DSC$W_LENGTH];
;  402          1227  4                                                     ! addr of last desc
;  403          1228  4             INCR DSC_PTR FROM .SRC [DSC$A_POINTER] TO .END_ADDR
;  404          1229  4                 BY .SRC [DSC$W_LENGTH] DO
;  405          1230  5                 BEGIN
;  406          1231  5                 MAP
;  407          1232  5                     DSC_PTR : REF BLOCK [8,BYTE];
;  408          1233  6                 LEN = (IF .DSC_PTR [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
;  409          1234  7                        THEN (.DSC_PTR [DSC$W_LENGTH]/2 + 1)
;  410          1235  5                        ELSE .DSC_PTR [DSC$W_LENGTH]);
;  411          1236  5                 DEST [DSC$A_POINTER] = CH$MOVE (.LEN, .DSC_PTR [DSC$A_POINTER],
;  412          1237  5                                                .DEST [DSC$A_POINTER]);
;  413          1238  5                 DEST [DSC$W_LENGTH] = .DEST [DSC$W_LENGTH] - .LEN;
;  414          1239  4                 END;
;  415          1240  4             END
;  416          1241  3         ELSE
;  417          1242  4             BEGIN                                   ! numeric array
;  418          1243  5             LEN = (IF .SRC [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
;  419          1244  6                    THEN (.SRC [DSC$W_LENGTH]/2 + 1) * (.SRC [DSC$L_ARSIZE]/ .SRC [DSC$W_LENGTH])
;  420          1245  4                    ELSE .SRC [DSC$L_ARSIZE]);
;  421          1246  4             DEST [DSC$A_POINTER] = CH$MOVE (.LEN, .SRC [DSC$A_POINTER],
;  422          1247  4                                            .DEST [DSC$A_POINTER]);
;  423          1248  4             DEST [DSC$W_LENGTH] = .DEST [DSC$W_LENGTH] - .LEN;
;  424          1249  3             END;
;  425          1250  3         END
;  426          1251  2     ELSE
;  427          1252  3         BEGIN                                       ! virtual array
;  428          1253  3         BAS$$WHOLE_VA_FETCH (.SRC, .DEST [DSC$A_POINTER]);
;  429          1254  4         LEN = (IF .SRC [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
;  430          1255  5                THEN (.SRC [DSC$W_LENGTH]/2 + 1) * (.SRC [DSC$L_ARSIZE]/ .SRC [DSC$W_LENGTH])
;  431          1256  3                ELSE .SRC [DSC$L_ARSIZE]);
;  432          1257  3         DEST [DSC$A_POINTER] = .DEST [DSC$A_POINTER] + .LEN;
;  433          1258  3         DEST [DSC$W_LENGTH] = .DEST [DSC$W_LENGTH] - .LEN;
;  434          1259  2         END;
;  435          1260  2
;  436          1261  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
```

```
;  438          1262  1 ROUTINE SRC_SA (                                  ! MOVE a string array
;  439          1263  1         SRC : REF BLOCK [, BYTE],                 ! Source: array
;  440          1264  1         DEST : REF BLOCK [, BYTE],                ! Destination: buffer
;  441          1265  1         LENGTH                                    ! Limit on string size
;  442          1266  1     ) : CALL_CCB NOVALUE =
;  443          1267  1
;  444          1268  1 !++
;  445          1269  1 ! FUNCTIONAL DESCRIPTION:
;  446          1270  1 !
;  447          1271  1 !       Within a MOVE statement, move a string array or record to the buffer.
;  448          1272  1 !
;  449          1273  1 ! FORMAL PARAMETERS:
;  450          1274  1 !
;  451          1275  1 !       SRC.rx.da       The source, a string array or record.
;  452          1276  1 !       DEST.mq.r       The destination, the I/O buffer.  It is updated to
;  453          1277  1 !                       reflect the array.
;  454          1278  1 !       LENGTH.rl.v     -1 or limit of length of string to copy
;  455          1279  1 !
;  456          1280  1 ! IMPLICIT INPUTS:
;  457          1281  1 !
;  458          1282  1 !       CCB, which is used only to provide good error messages.
;  459          1283  1 !
;  460          1284  1 ! IMPLICIT OUTPUTS:
;  461          1285  1 !
;  462          1286  1 !       NONE
;  463          1287  1 !
;  464          1288  1 ! ROUTINE VALUE:
;  465          1289  1 ! COMPLETION CODES:
;  466          1290  1 !
;  467          1291  1 !       NONE
;  468          1292  1 !
;  469          1293  1 ! SIDE EFFECTS:
;  470          1294  1 !
;  471          1295  1 !       Signals if an error is encountered.
;  472          1296  1 !       Updates the buffer's address and count to reflect the movement
;  473          1297  1 !       of the array.
;  474          1298  1 !
;  475          1299  1 !--
;  476          1300  1
;  477          1301  2     BEGIN
;  478          1302  2
;  479          1303  2     EXTERNAL REGISTER
;  480          1304  2         CCB : REF BLOCK [, BYTE];
;  481          1305  2
;  482          1306  2     LOCAL
;  483          1307  2         POWER_OF_2: VECTOR [10, WORD]
;  484          1308  2                 !+
;  485          1309  2                 ! Only have to check up to 512 because the compiler
;  486          1310  2                 ! allocates virtual arrays so that they will not
;  487          1311  2                 ! cross block boundaries.
;  488          1312  2                 !-
;  489          1313  2                 INITIAL ( WORD(1,2,4,8,16,32,64,128,256,512) ),
;  490          1314  2         LEN;
;  491          1315  2
;  492          1316  2 !+
;  493          1317  2 ! Dynamic string arrays will be arrays of descriptors, so the pointer field
;  494          1318  2 ! must be used to get to the string.  The elements in a static string array
```

```
;  495          1319  2 ! will be the strings themselves.  In both cases do an element by element move
;  496          1320  2 ! in order to accomodate padding, etc. which may be necessary.  Virtual arrays
;  497          1321  2 ! must call the virtual array fetch routine.
;  498          1322  2 !-
;  499          1323  2
;  500          1324  2     IF .SRC [DSC$B_CLASS] EQL DSC$K_CLASS_A
;  501          1325  2     THEN
;  502          1326  3         BEGIN
;  503          1327  3         !+
;  504          1328  3         ! memory array
;  505          1329  3         !-
;  506          1330  3         IF .SRC [DSC$B_DTYPE] EQL DSC$K_DTYPE_DSC
;  507          1331  3         THEN
;  508          1332  4             BEGIN
;  509          1333  4             !+
;  510          1334  4             ! dynamic strings
;  511          1335  4             !-
;  512          1336  4             LOCAL
;  513          1337  4                 END_ADDR;                       ! addr of last elem of array
;  514          1338  4
;  515          1339  4             END_ADDR = .SRC [DSC$A_POINTER] + .SRC [DSC$L_ARSIZE] - .SRC [DSC$W_LENGTH];
;  516          1340  4             !+
;  517          1341  4             ! Loop thru array moving each element to buffer.
;  518          1342  4             !-
;  519          1343  4             INCR DSC_PTR FROM .SRC [DSC$A_POINTER] TO .END_ADDR BY .SRC [DSC$W_LENGTH] DO
;  520          1344  5                 BEGIN
;  521          1345  5
;  522          1346  5                 MAP
;  523          1347  5                     DSC_PTR : REF BLOCK [8,BYTE];
;  524          1348  5
;  525          1349  5                 LEN = (IF .LENGTH LSS 0 THEN .DSC_PTR [DSC$W_LENGTH] ELSE .LENGTH);
;  526          1350  5                                                 ! dest length depends on if
;  527          1351  5                                                 ! LENGTH parameter given
;  528          1352  5                 IF .DEST [DSC$W_LENGTH] LSSU .LEN
;  529          1353  5                 THEN
;  530          1354  5                     BAS$$STOP_IO (BAS$K_MOVOVEBUF);
;  531          1355  5                 DEST [DSC$A_POINTER] = CH$COPY (.DSC_PTR [DSC$W_LENGTH],
;  532          1356  5                     .DSC_PTR [DSC$A_POINTER], %C' ', .LEN,
;  533          1357  5                     .DEST [DSC$A_POINTER]);     ! move string from array to
;  534          1358  5                                                 ! buffer, blank padding if
;  535          1359  5                                                 ! necessary
;  536          1360  5                 DEST [DSC$W_LENGTH] = .DEST [DSC$W_LENGTH] - .LEN;
;  537          1361  5                                                 ! buffer len must be decreased
;  538          1362  5                                                 ! by amt moved so check can be
;  539          1363  5                                                 ! done so that further moves
;  540          1364  5                                                 ! will not overflow buffer
;  541          1365  4                 END;
;  542          1366  4             END
;  543          1367  3         ELSE
;  544          1368  4             BEGIN
;  545          1369  4             !+
;  546          1370  4             ! static strings or records
;  547          1371  4             !-
;  548          1372  4             LOCAL
;  549          1373  4                 END_ADDR;                       ! addr of last elem of array
;  550          1374  4
;  551          1375  4             LEN = (IF .LENGTH LSS 0 THEN .SRC [DSC$W_LENGTH] ELSE .LENGTH);
```

```
;  552          1376  4                                        . dest length depends on if
;  553          1377  4                                        ! LENGTH parameter given
;  554          1378  4              END_ADDR = .SRC [DSC$A_POINTER] + .SRC [DSC$L_ARSIZE] - .SRC [DSC$W_LENGTH];
;  555          1379  4              !+
;  556          1380  4              ! Loop thru array moving each element to buffer.
;  557          1381  4              !-
;  558          1382  4              INCR ELEM_PTR FROM .SRC [DSC$A_POINTER] TO .END_ADDR BY .SRC [DSC$W_LENGTH] DO
;  559          1383  5                  BEGIN
;  560          1384  5
;  561          1385  5                  IF .DEST [DSC$W_LENGTH] LSSU .LEN
;  562          1386  5                  THEN
;  563          1387  5                      BAS$$STOP_IO (BAS$K_MOVOVEBUF);
;  564          1388  5                  DEST [DSC$A_POINTER] = CH$COPY (.SRC [DSC$W_LENGTH],
;  565          1389  5                      .ELEM_PTR, %C' ', .LEN,
;  566          1390  5                      .DEST [DSC$A_POINTER]);      ! move string from array to
;  567          1391  5                                                   !  buffer, blank padding if
;  568          1392  5                                                   !  necessary
;  569          1393  5                  DEST [DSC$W_LENGTH] = .DEST [DSC$W_LENGTH] - .LEN;
;  570          1394  5                                                   ! buffer len must be decreased
;  571          1395  5                                                   !  by amt moved so check can be
;  572          1396  5                                                   !  done so that further moves
;  573          1397  5                                                   !  will not overflow buffer
;  574          1398  4                  END;
;  575          1399  3              END;
;  576          1400  3          END
;  577          1401  2      ELSE
;  578          1402  2              IF .SRC [DSC$B_CLASS] EQL DSC$K_CLASS_BFA
;  579          1403  2              THEN
;  580          1404  3                  BEGIN
;  581          1405  3                  !+
;  582          1406  3                  ! virtual array
;  583          1407  3                  !-
;  584          1408  3                  LOCAL
;  585          1409  3                      DEST_LEN,                    ! buffer element length
;  586          1410  3                      ELEM_LEN,                    ! array element length
;  587          1411  3                      MEM_LOC;                     ! virtual memory pointer
;  588          1412  3
;  589          1413  3                  LIB$GET_VM (SRC [DSC$L_ARSIZE], MEM_LOC);
;  590          1414  3                                                   ! alloc memory for virtual array
;  591          1415  3                  BAS$$WHOLE_VA_FETCH (.SRC, .MEM_LOC);
;  592          1416  3                                                   ! move array to memory
;  593          1417  3                  ELEM_LEN = .SRC [DSC$W_LENGTH];
;  594          1418  3                  !+
;  595          1419  3                  ! If we are dealing with an array of RFAs or RECORDs,
;  596          1420  3                  ! ELEM_LEN must be changed to a power of 2 since that is
;  597          1421  3                  ! the virtual array element's true size.
;  598          1422  3                  !-
;  599          1423  3                  IF .SRC [DSC$B_DTYPE] EQL DSC$K_DTYPE_Z
;  600          1424  3                  THEN
;  601          1425  4                      BEGIN
;  602          1426  4
;  603          1427  5                      ELEM_LEN = (INCR I FROM 0 TO 9
;  604          1428  5                      DO
;  605          1429  5                          IF .ELEM_LEN LEQ .POWER_OF_2 [.I]
;  606          1430  5                          THEN
;  607          1431  4                              EXITLOOP .POWER_OF_2 [.I]);
;  608          1432  4
```

```
;  609          1433  3                END;
;  610          1434  3            !+
;  611          1435  3            ! Loop thru array moving each element to buffer.
;  612          1436  3            !-
;  613          1437  4            INCR PTR FROM .MEM_LOC TO (.MEM_LOC + .SRC [DSC$L_ARSIZE] - .ELEM_LEN)
;  614          1438  3                BY .ELEM_LEN DO
;  615          1439  3
;  616          1440  4                BEGIN
;  617          1441  4
;  618          1442  4                LOCAL
;  619          1443  4                    END_PTR;                ! ptr to end of array element,
;  620          1444  4                                            ! not including nulls
;  621          1445  4
;  622          1446  4                !+
;  623          1447  4                ! Trailing nulls must be removed.  Calculate the length of
;  624          1448  4                ! actual characters not including nulls on a per character
;  625          1449  4                ! basis.
;  626          1450  4                ! NOTE - virtual arrays of RECORDs or RFAs should not have
;  627          1451  4                ! the nulls removed.  Reason being desc length reflects the
;  628          1452  4                ! RECORD or RFA size and not the virt array element size.
;  629          1453  4                !-
;  630          1454  4                END_PTR = .PTR + .SRC[DSC$W_LENGTH] - 1;
;  631          1455  4                                            ! point to last char in array
;  632          1456  4                                            !  element
;  633          1457  4                IF .SRC [DSC$B_DTYPE] NEQ DSC$K_DTYPE_Z
;  634          1458  4                THEN
;  635          1459  4                    UNTIL .END_PTR EQL (.PTR - 1) DO
;  636          1460  5                        BEGIN
;  637          1461  5                        !+
;  638          1462  5                        ! Go backwards to 1st char looking for 1st non null.
;  639          1463  5                        !-
;  640          1464  5                        IF CH$FAIL (CH$FIND_NOT_CH (1, .END_PTR, %B'0') )
;  641          1465  5                        THEN
;  64^          1466  5                            END_PTR = .END_PTR - 1
;  643          1467  5                        ELSE
;  644          1468  5                            EXITLOOP;
;  645          1469  5
;  646          1470  4                        END;
;  647          1471  4                LEN = .END_PTR - .PTR + 1;  ! lenght of element w/o nulls
;  648          1472  4                DEST_LEN = (IF .LENGTH LSS 0 THEN .LEN ELSE .LENGTH);
;  649          1473  4                                            ! dest length depends on if
;  650          1474  4                                            !  LENGTH parameter given
;  651          1475  4                IF (.DEST [DSC$W_LENGTH] LSSU .LEN) THEN BAS$$STOP_IO (BAS$K_MOVOVEBUF);
;  652          1476  4                DEST [DSC$A_POINTER] = CH$COPY (.LEN, .PTR, %C' '
;  653          1477  4                                            ,.DEST_LEN, .DEST [DSC$A_POINTER]);
;  654          1478  4                                            ! move string from array to
;  655          1479  4                                            !  buffer, blank padding if
;  656          1480  4                                            !  necessary
;  657          1481  4                DEST [DSC$W_LENGTH] = .DEST [DSC$W_LENGTH] - .DEST_LEN;
;  658          1482  4                                            ! buffer len must be decreased
;  659          1483  4                                            !  by amt moved so check can be
;  660          1484  4                                            !  done so that further moves
;  661          1485  4                                            !  will not overflow buffer
;  662          1486  3                END;
;  663          1487  3
;  664          1488  3            LIB$FREE_VM (SRC [DSC$L_ARSIZE], MEM_LOC);
;  665          1489  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$:     PUSHAW  POWER_OF_2[I]
          08   AE          9E           10      00  ED 000F3          CMPZV   #0, #T6, @(SP)+, ELEM_LEN
                                           08  19 000F9          BLSS    16$
                     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
```

```
;  670          1493  1 ROUTINE SRC_DSC (                                   ! MOVE an array of dsc
;  671          1494  1         SRC : REF BLOCK [, BYTE],                   ! Source: array
;  672          1495  1         DEST : REF BLOCK [, BYTE],                  ! Destination: buffer
;  673          1496  1         LENGTH                                      ! Limit on string size
;  674          1497  1     ) : CALL_CCB NOVALUE =
;  675          1498  1
;  676          1499  1 !++
;  677          1500  1 ! FUNCTIONAL DESCRIPTION:
;  678          1501  1 !
;  679          1502  1 !       Within a MOVE statement, move an array of descriptors to a buffer.
;  680          1503  1 !       The descriptors may be string descriptors or numeric descriptors
;  681          1504  1 !       (in the case of dynamically mapped variables).  This routine
;  682          1505  1 !       determines the dtype of the descriptors and dispatches to the
;  683          1506  1 !       appropriate routine.
;  684          1507  1 !
;  685          1508  1 ! FORMAL PARAMETERS:
;  686          1509  1 !
;  687          1510  1 !       SRC.rx.da       The source, an array of desc
;  688          1511  1 !       DEST.mq.r       The destination, the I/O buffer.  It is updated to
;  689          1512  1 !                       reflect the array.
;  690          1513  1 !       LENGTH.rl.v     -1 or limit of length of string to copy
;  691          1514  1 !
;  692          1515  1 ! IMPLICIT INPUTS:
;  693          1516  1 !
;  694          1517  1 !       CCB, which is used only to provide good error messages.
;  695          1518  1 !
;  696          1519  1 ! IMPLICIT OUTPUTS:
;  697          1520  1 !
;  698          1521  1 !       NONE
;  699          1522  1 !
;  700          1523  1 ! ROUTINE VALUE:
;  701          1524  1 ! COMPLETION CODES:
;  702          1525  1 !
;  703          1526  1 !       NONE
;  704          1527  1 !
;  705          1528  1 ! SIDE EFFECTS:
;  706          1529  1 !
;  707          1530  1 !       Signals if an error is encountered.
;  708          1531  1 !
;  709          1532  1 !--
;  710          1533  1
;  711          1534  2     BEGIN
;  712          1535  2
;  713          1536  2     EXTERNAL REGISTER
;  714          1537  2         CCB : REF BLOCK [, BYTE];
;  715          1538  2
;  716          1539  2     LOCAL
;  717          1540  2         ELEM_DSC : REF BLOCK [8,BYTE],
;  718          1541  2         NUM_ELEMS;
;  719          1542  2
;  720          1543  2
;  721          1544  2 !+
;  722          1545  2 ! Set up a pointer to the first element of the array so that the dtype
;  723          1546  2 ! and length of the descriptor elements can be determined.
;  724          1547  2 !-
;  725          1548  2
;  726          1549  2     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 00000000G    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
```

```
;  760          1582  1 ROUTINE DEST_NA (                             ! MOVE a numeric array
;  761          1583  1         SRC : REF BLOCK [8, BYTE],            ! Source: buffer
;  762          1584  1         DEST : REF BLOCK [, BYTE]             ! Destination: numeric array
;  763          1585  1     ) : CALL_CCB NOVALUE =
;  764          1586  1
;  765          1587  1 !++
;  766          1588  1 ! FUNCTIONAL DESCRIPTION:
;  767          1589  1 !
;  768          1590  1 !       Within a MOVE statement, move the I/O buffer to a numeric array.
;  769          1591  1 !
;  770          1592  1 ! FORMAL PARAMETERS:
;  771          1593  1 !
;  772          1594  1 !       SRC.mq.r        The source, the I/O buffer.  It is updated to account for the
;  773          1595  1 !                       bytes taken from it.
;  774          1596  1 !       DEST.wx.da      The destination, a numeric array.
;  775          1597  1 !
;  776          1598  1 ! IMPLICIT INPUTS:
;  777          1599  1 !
;  778          1600  1 !       CCB, which is used only to provide good error messages.
;  779          1601  1 !
;  780          1602  1 ! IMPLICIT OUTPUTS:
;  781          1603  1 !
;  782          1604  1 !       NONE
;  783          1605  1 !
;  784          1606  1 ! ROUTINE VALUE:
;  785          1607  1 ! COMPLETION CODES:
;  786          1608  1 !
;  787          1609  1 !       NONE
;  788          1610  1 !
;  789          1611  1 ! SIDE EFFECTS:
;  790          1612  1 !
;  791          1613  1 !       Signals if an error is encountered.
;  792          1614  1 !       Updates the buffer's address and count to reflect the movement
;  793          1615  1 !       of the array.
;  794          1616  1 !
;  795          1617  1 !--
;  796          1618  1
;  797          1619  2     BEGIN
;  798          1620  2
;  799          1621  2     EXTERNAL REGISTER
;  800          1622  2         CCB : REF BLOCK [, BYTE];
;  801          1623  2
;  802          1624  2     LOCAL
;  803          1625  2         LEN;
;  804          1626  2
;  805          1627  2 !+
;  806          1628  2 ! The total number of bytes in the array must not be greater than the
;  807          1629  2 ! remaining length of the buffer.
;  808          1630  2 !-
;  809          1631  2
;  810          1632  2     IF (.SRC [DSC$B_DTYPE] NEQ DSC$K_DTYPE_DSC) AND
;  811          1633  3        (.SRC [DSC$W_LENGTH] LSSU .DEST [DSC$L_ARSIZE])
;  812          1634  2     THEN
;  813          1635  2         BAS$$STOP_IO (BAS$K_MOVOVEBUF);
;  814          1636  2
;  815          1637  2 !+
;  816          1638  2 ! Values can be moved to a memory array in one instruction.  If the array
```

```
;  817          1639  2 ! is virtual, however, the store routine must be called.  If the array is
;  818          1640  2 ! dynamically mapped, the elements are not contiguous and values must be
;  819          1641  2 ! moved one by one.
;  820          1642  2 !-
;  821          1643  2
;  822          1644  2     IF .DEST [DSC$B_CLASS] EQL D SC$K_CLASS_A
;  823          1645  2     THEN
;  824          1646  3         BEGIN                                   ! memory array
;  825          1647  3         IF .DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_DSC
;  826          1648  3         THEN
;  827          1649  4             BEGIN                               ! dynamically mapped array
;  828          1650  4             LOCAL
;  829          1651  4                 END_ADDR;                       ! addr of last elem of array
;  830          1652  4             END_ADDR = .DEST [DSC$A_POINTER] + .DEST [DSC$L_ARSIZE] - .DEST [DSC$W_LENGTH];
;  831          1653  4                                                 ! addr of last desc
;  832          1654  4             INCR DSC_PTR FROM .DEST [DSC$A_POINTER] TO .END_ADDR BY .DEST [DSC$W_LENGTH]  DO
;  833          1655  5                 BEGIN
;  834          1656  5                 MAP
;  835          1657  5                     DSC_PTR : REF BLOCK [8,BYTE];
;  836          1658  6                 LEN = (IF .DSC_PTR [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
;  837          1659  7                        THEN (.DSC_PTR [DSC$W_LENGTH]/2 + 1)
;  838          1660  5                        ELSE .DSC_PTR [DSC$W_LENGTH]);
;  839          1661  5                 CH$MOVE (.LEN, .SRC [DSC$A_POINTER], .DSC_PTR [DSC$A_POINTER]);
;  840          1662  5                 SRC [DSC$A_POINTER] = .SRC [DSC$A_POINTER] + .LEN;
;  841          1663  5                 SRC [DSC$W_LENGTH] = .SRC [DSC$W_LENGTH] - .LEN;
;  842          1664  4                 END;
;  843          1665  4             END
;  844          1666  3         ELSE
;  845          1667  4             BEGIN                               ! numeric array
;  846          1668  5             LEN = (IF .DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
;  847          1669  6                    THEN (.DEST [DSC$W_LENGTH]/2 + 1) * (.DEST [DSC$L_ARSIZE]/.DEST [DSC$W_LENGTH])
;  848          1670  4                    ELSE .DEST [DSC$L_ARSIZE]);
;  849          1671  4             CH$MOVE (.LEN, .SRC [DSC$A_POINTER], .DEST [DSC$A_POINTER]);
;  850          1672  4             SRC [DSC$A_POINTER] = .SRC [DSC$A_POINTER] + .LEN;
;  851          1673  4             SRC [DSC$W_LENGTH] = .SRC [DSC$W_LENGTH] - .LEN;
;  852          1674  3             END;
;  853          1675  3         END
;  854          1676  2     ELSE
;  855          1677  3         BEGIN                                   ! virtual array
;  856          1678  3         BAS$$WHOLE_VA_STORE (.DEST, .SRC [DSC$A_POINTER]);
;  857          1679  4         LEN = (IF .DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_P
;  858          1680  5                THEN (.DEST [DSC$W_LENGTH]/2 + 1) * (.DEST [DSC$L_ARSIZE]/.DEST [DSC$W_LENGTH])
;  859          1681  3                ELSE .DEST [DSC$L_ARSIZE]);
;  860          1682  3         SRC [DSC$A_POINTER] = .SRC [DSC$A_POINTER] + .LEN;
;  861          1683  3         SRC [DSC$W_LENGTH] = .SRC [DSC$W_LENGTH] - .LEN;
;  862          1684  2         END;
;  863          1685  2
;  864          1686  1     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
```

```
;  866          1687  1 ROUTINE DEST_SA (                             ! MOVE a string array
;  867          1688  1         SRC : REF BLOCK [8, BYTE],           ! Source: buffer
;  868          1689  1         DEST : REF BLOCK [, BYTE],           ! Destination: array
;  869          1690  1         LENGTH                               ! -1 or length of each string
;  870          1691  1     ) : CALL_CCB NOVALUE =
;  871          1692  1
;  872          1693  1 !++
;  873          1694  1 ! FUNCTIONAL DESCRIPTION:
;  874          1695  1 !
;  875          1696  1 !       Within a MOVE statement, move the I/O buffer to a string array or
;  876          1697  1 !       array of records.
;  877          1698  1 !
;  878          1699  1 ! FORMAL PARAMETERS:
;  879          1700  1 !
;  880          1701  1 !       SRC.mq.r        The source, the I/O buffer.  This is updated to account for
;  881          1702  1 !                       the bytes taken from it.
;  882          1703  1 !       DEST.wx.da      The destination, a string array or array of records.
;  883          1704  1 !
;  884          1705  1 ! IMPLICIT INPUTS:
;  885          1706  1 !
;  886          1707  1 !       CCB, which is used only to provide good error messages.
;  887          1708  1 !
;  888          1709  1 ! IMPLICIT OUTPUTS:
;  889          1710  1 !
;  890          1711  1 !       NONE
;  891          1712  1 !
;  892          1713  1 ! ROUTINE VALUE:
;  893          1714  1 ! COMPLETION CODES:
;  894          1715  1 !
;  895          1716  1 !       NONE
;  896          1717  1 !
;  897          1718  1 ! SIDE EFFECTS:
;  898          1719  1 !
;  899          1720  1 !       Signals if an error is encountered.
;  900          1721  1 !       Updates the buffer's address and count to reflect the movement
;  901          1722  1 !       of the array.
;  902          1723  1 !
;  903          1724  1 !--
;  904          1725  1
;  905          1726  2     BEGIN
;  906          1727  2
;  907          1728  2     EXTERNAL REGISTER
;  908          1729  2         CCB : REF BLOCK [, BYTE];
;  909          1730  2
;  910          1731  2     LITERAL
;  911          1732  2         K_DEFAULT_STR_LEN = 16;
;  912          1733  2
;  913          1734  2     LOCAL
;  914          1735  2         POWER_OF_2: VECTOR [10, WORD]
;  915          1736  2                 !
;  916          1737  2                 ! Only have to check up to 512 because the compiler
;  917          1738  2                 ! allocates virtual arrays so that they will not
;  918          1739  2                 ! cross block boundaries.
;  919          1740  2                 !-
;  920          1741  2                 INITIAL ( WORD(1,2,4,8,16,32,64,128,256,512) ),
;  921          1742  2         LEN;
;  922          1743  2
```

```
;  923          1744  2 !+
;  924          1745  2 ! Dynamic string arrays are arrays of descriptors, while static string
;  925          1746  2 ! arrays contain the strings themselves.  In both cases, do an element
;  926          1747  2 ! by element move so that padding can be done if necessary.  Virtual
;  927          1748  2 ! arrays must use the virtual array store routine.
;  928          1749  2 !-
;  929          1750  2
;  930          1751  2     IF .DEST [DSC$B_CLASS] EQL DSC$K_CLASS_A
;  931          1752  2     THEN
;  932          1753  3         BEGIN
;  933          1754  3         !+
;  934          1755  3         ! memory array
;  935          1756  3         !-
;  936          1757  3         IF .DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_DSC
;  937          1758  3         THEN
;  938          1759  4             BEGIN
;  939          1760  4             !+
;  940          1761  4             ! static strings or records
;  941          1762  4             !-
;  942          1763  4             LOCAL
;  943          1764  4                 END_ADDR;                          ! addr of last elem of array
;  944          1765  4
;  945          1766  4             LEN = (IF .LENGTH LSS 0 THEN K_DEFAULT_STR_LEN ELSE .LENGTH);
;  946          1767  4                                                    ! dest length depends on if
;  947          1768  4                                                    !  LENGTH parameter given,
;  948          1769  4                                                    !  defaults to 16
;  949          1770  4             IF .DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_Z
;  950          1771  4             THEN
;  951          1772  4                 LEN = .DEST [DSC$W_LENGTH];        ! if we are dealing with an
;  952          1773  4                                                    !  array of RFAs or RECORDs,
;  953          1774  4                                                    !  get the length from the descr
;  954          1775  4
;  955          1776  4             END_ADDR = .DEST [DSC$A_POINTER] + .DEST [DSC$L_ARSIZE] -
;  956          1777  4                 .DEST [DSC$W_LENGTH];
;  957          1778  4             !+
;  958          1779  4             ! Loop thru array moving each element from buffer to it.
;  959          1780  4             !-
;  960          1781  4             INCR ELEM_PTR FROM .DEST [DSC$A_POINTER] TO .END_ADDR
;  961          1782  4                 BY .DEST [DSC$W_LENGTH] DO
;  962          1783  5                 BEGIN
;  963          1784  5
;  964          1785  6                 IF (.SRC [DSC$W_LENGTH] LSSU .LEN)
;  965          1786  5                 THEN
;  966          1787  5                     BAS$$STOP_IO (BAS$K_MOVOVEBUF);
;  967          1788  5                 CH$COPY (.LEN, .SRC [DSC$A_POINTER], %C' ', .DEST [DSC$W_LENGTH], .ELEM_PTR);
;  968          1789  5                                                    !  array, Blank padding if
;  969          1790  5                                                    !  necessary
;  970          1791  5                 SRC [DSC$A_POINTER] = .SRC [DSC$A_POINTER] + .LEN;
;  971          1792  5                                                    ! update buffer pointer
;  972          1793  5                 SRC [DSC$W_LENGTH] = .SRC [DSC$W_LENGTH] - .LEN;
;  973          1794  5                                                    ! buffer len must be decreased
;  974          1795  5                                                    !  by amt moved so check can be
;  975          1796  5                                                    !  done so that further moves
;  976          1797  5                                                    !  will not overflow buffer
;  977          1798  4                 END;
;  978          1799  4             END
;  979          1800  3         ELSE
```

```
;  980          1801  4          BEGIN
;  981          1802  4          !+
;  982          1803  4          ! dynamic strings
;  983          1804  4          !-
;  984          1805  4          LOCAL
;  985          1806  4              DEST_ITEM : REF BLOCK [8, BYTE],! desc for string within array
;  986          1807  4              END_ADDR;                       ! addr of last elem of array
;  987          1808  4
;  988          1809  4          LEN = (IF .LENGTH LSS 0 THEN K_DEFAULT_STR_LEN ELSE .LENGTH);
;  989          1810  4                                              ! dest length depends on if
;  990          1811  4                                              !  LENGTH parameter given,
;  991          1812  4                                              !  defaults to 16
;  992          1813  4          DEST_ITEM = .DEST [DSC$A_POINTER];  ! set up desc for actual item
;  993          1814  4          IF .DEST_ITEM [DSC$B_DTYPE] EQL DSC$K_DTYPE_Z
;  994          1815  4          THEN
;  995          1816  4              LEN = .DEST_ITEM [DSC$W_LENGTH];! if we are dealing with an
;  996          1817  4                                              !  array of RFAs or RECORDs,
;  997          1818  4                                              !  get the length from the descr
;  998          1819  4
;  999          1820  4          END_ADDR = .DEST [DSC$A_POINTER] + .DEST [DSC$L_ARSIZE] -
; 1000          1821  4              .DEST [DSC$W_LENGTH];
; 1001          1822  4          !+
; 1002          1823  4          ! Loop thru array moving each element from buffer to it.
; 1003          1824  4          !-
; 1004          1825  4          INCR DSC_PTR FROM .DEST [DSC$A_POINTER] TO .END_ADDR
; 1005          1826  4              BY .DEST [DSC$W_LENGTH] DO
; 1006          1827  5              BEGIN
; 1007          1828  5
; 1008          1829  5              MAP
; 1009          1830  5                  DSC_PTR: REF BLOCK [8,BYTE];! desc for string within array
; 1010          1831  5
; 1011          1832  6              IF (.SRC [DSC$W_LENGTH] LSSU .LEN)
; 1012          1833  5              THEN
; 1013          1834  5                  BAS$$STOP_IO (BAS$K_MOVOVEBUF);
; 1014          1835  5              STR$COPY_R (.DSC_PTR, LEN, .SRC [DSC$A_POINTER]);
; 1015          1836  5                                              ! move string from buffer to
; 1016          1837  5                                              !  array, null padding if
; 1017          1838  5                                              !  necessary
; 1018          1839  5              SRC [DSC$A_POINTER] = .SRC [DSC$A_POINTER] + .LEN;
; 1019          1840  5                                              ! update buffer pointer
; 1020          1841  5              SRC [DSC$W_LENGTH] = .SRC [DSC$W_LENGTH] - .LEN;
; 1021          1842  5                                              ! buffer len must be decreased
; 1022          1843  5                                              !  by amt moved so check can be
; 1023          1844  5                                              !  done so that further moves
; 1024          1845  5                                              !  will not overflow buffer
; 1025          1846  4              END;
; 1026          1847  4          END
; 1027          1848  3      END
; 1028          1849  2  ELSE
; 1029          1850  2      IF .DEST [DSC$B_CLASS] EQL DSC$K_CLASS_BFA
; 1030          1851  2      THEN
; 1031          1852  3          BEGIN
; 1032          1853  3          !+
; 1033          1854  3          ! virtual array
; 1034          1855  3          !-
; 1035          1856  3          LOCAL
; 1036          1857  3              SRC_LEN,                        ! buffer element length
```

```
;1037     1858  3          MEM_LOC;                        ! virtual memory pointer
;1038     1859  3
;1039     1860  3      LIB$GET_VM (DEST [DSC$L_ARSIZE], MEM_LOC);
;1040     1861  3                                          ! alloc memory for virtual array
;1041     1862  3      SRC_LEN =
;1042     1863  3          (IF .LENGTH LSS 0 THEN K_DEFAULT_STR_LEN ELSE .LENGTH);
;1043     1864  3                                          ! dest length depends on if
;1044     1865  3                                          !  LENGTH parameter given,
;1045     1866  3                                          !  defaults to 16
;1046     1867  3      LEN = .DEST [DSC$W_LENGTH];         ! array element length
;1047     1868  3      !+
;1048     1869  3      ! If we are dealing with an array of RFAs or RECORDs,
;1049     1870  3      ! LEN must be changed to a power of 2 since that is the
;1050     1871  3      ! virtual array element's true size.
;1051     1872  3      !-
;1052     1873  3      IF .DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_Z
;1053     1874  3      THEN
;1054     1875  4          BEGIN
;1055     1876  4
;1056     1877  5          LEN = (INCR I FROM 0 TO 9
;1057     1878  5          DO
;1058     1879  5              IF .LEN LEQ .POWER_OF_2 [.I]
;1059     1880  5              THEN
;1060     1881  4                  EXITLOOP .POWER_OF_2 [.I]);
;1061     1882  4          SRC_LEN = .DEST [DSC$W_LENGTH];
;1062     1883  4
;1063     1884  3          END;
;1064     1885  3
;1065     1886  3      IF (.LEN GTRU .SRC [DSC$W_LENGTH])
;1066     1887  3      THEN
;1067     1888  3          BAS$$STOP_IO (BAS$K_MOVOVEBUF);
;1068     1889  3      !+
;1069     1890  3      ! Loop thru array moving each element from buffer to it.
;1070     1891  3      !-
;1071     1892  4      INCR PTR FROM .MEM_LOC TO (.MEM_LOC + .DEST [DSC$L_ARSIZE] -
;1072     1893  3          .LEN) BY .LEN DO
;1073     1894  4          BEGIN
;1074     1895  4          CH$COPY (.SRC_LEN, .SRC [DSC$A_POINTER],
;1075     1896  4              %X'0', .LEN, .PTR);         ! move string from buffer to
;1076     1897  4                                          !  array, null padding if
;1077     1898  4                                          !  necessary
;1078     1899  4          SRC [DSC$A_POINTER] = .SRC [DSC$A_POINTER] + .SRC_LEN;
;1079     1900  4                                          ! update buffer pointer
;1080     1901  4          SRC [DSC$W_LENGTH] = .SRC [DSC$W_LENGTH] - .SRC_LEN;
;1081     1902  4                                          ! buffer len must be decreased
;1082     1903  4                                          !  by amt moved so check can be
;1083     1904  4                                          !  done so that further moves
;1084     1905  4                                          !  will not overflow buffer
;1085     1906  3          END;
;1086     1907  3
;1087     1908  3      BAS$$WHOLE_VA_STORE (.DEST, .MEM_LOC);
;1088     1909  3                                          ! store array from memory
;1089     1910  3      LIB$FREE_VM (DEST [DSC$L_ARSIZE], MEM_LOC);
;1090     1911  3                                          ! dealloc memory used for array
;1091     1912  2      END;
;1092     1913  2
;1093     1914  1  END;                                    ! of DEST_SA
```

```
                                                  004F3                 .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$:      ADDL3   12(R7), 4(R7), R0                     ; 1776
                                              59          67  3C 00049          MOVZWL  (R7), R9                              ; 1777
                                        6E                50  C3 0004C          SUBL3   R9, R0, END_ADDR                      ;
                                              5A    04    A7  D0 00050          MOVL    4(R7), ELEM_PTR                       ; 1785
                                                          26  11 00054          BRB     7$                                    ;
                              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
                                    000000000G    00          01  FB 00062          CALLS   #1, BA$$$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
```

```
;1095     1915  1 ROUTINE DEST_DSC (                            ! MOVE a descriptor
;1096     1916  1         SRC : REF BLOCK [8, BYTE],            ! Source: buffer
;1097     1917  1         DEST : REF BLOCK [, BYTE],            ! Destination: array
;1098     1918  1         LENGTH                                ! -1 or length of each string
;1099     1919  1     ) : CALL_CCB NOVALUE =
;1100     1920  1
;1101     1921  1 !++
;1102     1922  1 ! FUNCTIONAL DESCRIPTION:
;1103     1923  1 !
;1104     1924  1 !       Within a MOVE statement, move the I/O buffer to an array of
;1105     1925  1 !       descriptors.  The descriptors may be string descriptors or
;1106     1926  1 !       descriptors for dynamic variables (probably numeric).  So
;1107     1927  1 !       determine the dtype of the descriptor and then call SRC_SA or
;1108     1928  1 !       SRC_NA to perform the work.
;1109     1929  1 !
;1110     1930  1 ! FORMAL PARAMETERS:
;1111     1931  1 !
;1112     1932  1 !       SRC.mq.r        The source, the I/O buffer.  This is updated to account for
;1113     1933  1 !                       the bytes taken from it.
;1114     1934  1 !       DEST.wx.da      The destination, an array of descriptors
;1115     1935  1 !
;1116     1936  1 ! IMPLICIT INPUTS:
;1117     1937  1 !
;1118     1938  1 !       CCB, which is used only to provide good error messages.
;1119     1939  1 !
;1120     1940  1 ! IMPLICIT OUTPUTS:
;1121     1941  1 !
;1122     1942  1 !       NONE
;1123     1943  1 !
;1124     1944  1 ! ROUTINE VALUE:
;1125     1945  1 ! COMPLETION CODES:
;1126     1946  1 !
;1127     1947  1 !       NONE
;1128     1948  1 !
;1129     1949  1 ! SIDE EFFECTS:
;1130     1950  1 !
;1131     1951  1 !       Signals if an error is encountered.
;1132     1952  1 !
;1133     1953  1 !--
;1134     1954  1
;1135     1955  2     BEGIN
;1136     1956  2
;1137     1957  2     EXTERNAL REGISTER
;1138     1958  2         CCB : REF BLOCK [, BYTE];
;1139     1959  2
;1140     1960  2     LOCAL
;1141     1961  2         ELEM_DSC : REF BLOCK [8,BYTE],
;1142     1962  2         NUM_ELEMS;
;1143     1963  2
;1144     1964  2
;1145     1965  2 !+
;1146     1966  2 ! Set up pointer to the first element descriptor in the array so that the
;1147     1967  2 ! dtype and length can be determined.
;1148     1968  2 !-
;1149     1969  2
;1150     1970  2     ELEM_DSC = .DEST [DSC$A_A0];
;1151     1971  2
```

```
;  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