```
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
BBBBBBBB      AAAAAA      SSSSSSSS  MM      MM    000000    VV      VV  EEEEEEEEEE
BB      BB  AA      AA  SS        MMMM  MMMM  OO      OO  VV      VV  EE
BB      BB  AA      AA  SS        MMMM  MMMM  OO      OO  VV      VV  EE
BB      BB  AA      AA  SS        MM  MM  MM  OO      OO  VV      VV  EE
BB      BB  AA      AA  SS        MM  MM  MM  OO      OO  VV      VV  EE
BBBBBBBB    AA      AA    SSSSSS  MM      MM  OO      OO  VV      VV  EEEEEEEE
BBBBBBBB    AA      AA    SSSSSS  MM      MM  OO      OO  VV      VV  EEEEEEEE
BB      BB  AAAAAAAAAA          SS  MM      MM  OO      OO  VV      VV  EE
BB      BB  AAAAAAAAAA          SS  MM      MM  OO      OO  VV      VV  EE
BB      BB  AA      AA          SS  MM      MM  OO      OO    VV  VV    EE              ....
BB      BB  AA      AA          SS  MM      MM  OO      OO    VV  VV    EE              ....
BBBBBBBB    AA      AA  SSSSSSSS    MM      MM    000000      VV      EEEEEEEEEE      ....
BBBBBBBB    AA      AA  SSSSSSSS    MM      MM    000000      VV      EEEEEEEEEE      ....

        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 (                                    !
;     2          0002  0                 IDENT = '1-006'                     ! File: BASMOVE.B32 EDIT:FM1006
;     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.
;    37          0037  1 !
;    38          0038  1 ! ENVIRONMENT:  VAX-11 User Mode
;    39          0039  1 !
;    40          0040  1 ! AUTHOR: John Sauter, CREATION DATE: 21-MAY-1979
;    41          0041  1 !
;    42          0042  1 ! MODIFIED BY:
;    43          0043  1 !
;    44          0044  1 ! 1-001 - Original.
;    45          0045  1 ! 1-002 - Call BAS$$CB_GET, so this module does not have to be in the
;    46          0046  1 !           sharable library.  JBS 22-AUG-1979
;    47          0047  1 ! 1-003 - Reverse the order of arguments to BAS$MOVE_BEG.  JBS 04-SEP-1979
;    48          0048  1 ! 1-004 - Add MOVE_TO and MOVE_FROM entry points, which we will split later.
;    49          0049  1 !           JBS 30-NOV-1979
;    50          0050  1 ! 1-005 - Channel zero should translate to the appropriate BASIC LUN (BAS$K_LUN_INPU).
;    51          0051  1 !           Call it an error if user calls with a negative channel, at the same
;    52          0052  1 !         time add 2 to lub$k_ilun_min to force bas$$cb_push to signal an
;    53          0053  1 !         error if LUNs -7 or -8 are being pushed, the later is a temporary
;    54          0054  1 !         fix for #0 syntax in BASIC until the standard comm. decides on the
;    55          0055  1 !         issue.
;    56          0056  1 ! 1-006 - Undo 5.  We can now do I/O to #0, because BAS$PUT will use foreign
;    57          0057  1 !         buffer mechanism to do PUTs to #0. FM 9-JUL-81.
```

```
;    58          0058  1 !--
;    59          0059  1
;    60          0060  1 !<BLF/PAGE>
```

```
;    62          0061  1 !
;    63          0062  1 ! SWITCHES:
;    64          0063  1 !
;    65          0064  1
;    66          0065  1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;    67          0066  1
;    68          0067  1 !
;    69          0068  1 ! LINKAGES:
;    70          0069  1 !
;    71          0070  1
;    72          0071  1 REQUIRE 'RTLIN:OTSLNK';                         ! Define linkages
;    73          0500  1
;    74          0501  1 !
;    75          0502  1 ! TABLE OF CONTENTS:
;    76          0503  1 !
;    77          0504  1
;    78          0505  1 FORWARD ROUTINE
;    79          0506  1     BAS$MOVE_BEG : NOVALUE,                     ! Start of MOVE statement
;    80          0507  1     BAS$MOVE_END : NOVALUE;                     ! End of MOVE statement
;    81          0508  1
;    82          0509  1 !
;    83          0510  1 ! INCLUDE FILES:
;    84          0511  1 !
;    85          0512  1
;    86          0513  1 REQUIRE 'RTLML:OTSLUB';                         ! Get LUB definitions
;    87          0653  1
;    88          0654  1 REQUIRE 'RTLML:OTSISB';                         ! Get ISB definitions
;    89          0822  1
;    90          0823  1 REQUIRE 'RTLIN:RTLPSECT';                       ! Macros for defining psects
;    91          0918  1
;    92          0919  1 LIBRARY 'RTLSTARLE';                            ! System symbols
;    93          0920  1
;    94          0921  1 !
;    95          0922  1 ! MACROS:
;    96          0923  1 !
;    97          0924  1 !       NONE
;    98          0925  1 !
;    99          0926  1 ! EQUATED SYMBOLS:
;   100          0927  1 !
;   101          0928  1
;   102          0929  1 GLOBAL BIND
;   103          0930  1     ROUTINE
;   104          0931  1     BAS$MOVE_TO = BAS$MOVE_BEG;
;   105          0932  1
;   106          0933  1 GLOBAL BIND
;   107          0934  1     ROUTINE
;   108          0935  1     BAS$MOVE_FROM = BAS$MOVE_BEG;
;   109          0936  1
;   110          0937  1 !
;   111          0938  1 ! PSECTS:
;   112          0939  1 !
;   113          0940  1 DECLARE_PSECTS (BAS);                           ! Declare psects for BAS$ facility
;   114          0941  1 !
;   115          0942  1 ! OWN STORAGE:
;   116          0943  1 !
;   117          0944  1 !       NONE
;   118          0945  1 !
```

```
;  119          0946  1 ! EXTERNAL REFERENCES:
;  120          0947  1 !
;  121          0948  1
;  122          0949  1 EXTERNAL ROUTINE
;  123          0950  1     BAS$$CB_PUSH : JSB_CB_PUSH NOVALUE,          ! Load register CCB
;  124          0951  1     BAS$$CB_POP : JSB_CB_POP NOVALUE,            ! Done with register CCB
;  125          0952  1     BAS$$CB_GET : JSB_CB_GET NOVALUE,            ! Load current CCB
;  126          0953  1     BAS$$STOP : NOVALUE,                         ! Signal fatal error
;  127          0954  1     BAS$$STOP_IO : NOVALUE,                      ! Signal fatal I/O error
;  128          0955  1     BAS$$OPEN_ZERO : NOVALUE;                    ! OPEN channel 0
;  129          0956  1
;  130          0957  1 !+
;  131          0958  1 ! The following are the error codes used in this module.
;  132          0959  1 !-
;  133          0960  1
;  134          0961  1 EXTERNAL LITERAL
;  135          0962  1     BAS$K_PROLOSSOR : UNSIGNED (8),              ! Program lost, sorry
;  136          0963  1     BAS$K_IO_CHANOT : UNSIGNED (8),              ! I/O channel not open
;  137          0964  1     BAS$K_ILLIO_CHA : UNSIGNED (8);              ! Illegal I/O channel
```

```
;  139          0965  1 GLOBAL ROUTINE BAS$MOVE_BEG (                    ! Start of MOVE statement
;  140          0966  1         DESC,                                     ! Descriptor of buffer
;  141          0967  1         UNIT                                      ! Channel to start MOVE on
;  142          0968  1     ) : NOVALUE =
;  143          0969  1
;  144          0970  1 !++
;  145          0971  1 ! FUNCTIONAL DESCRIPTION:
;  146          0972  1 !
;  147          0973  1 !       Start a MOVE statement on the specified channel.  We do this
;  148          0974  1 !       by returning a descriptor of the buffer, which the compiled
;  149          0975  1 !       code will use when accessing the I/O buffer.
;  150          0976  1 !
;  151          0977  1 ! FORMAL PARAMETERS:
;  152          0978  1 !
;  153          0979  1 !       DESC.wq.r       Descriptor of the I/O buffer
;  154          0980  1 !       UNIT.rl.v       The channel whose buffer descriptor to return.
;  155          0981  1 !
;  156          0982  1 ! IMPLICIT INPUTS:
;  157          0983  1 !
;  158          0984  1 !       The LUB$W_RBUF_SIZE field of the LUB of the specified channel.
;  159          0985  1 !       Also, LUB$A_RBUF_ADR.
;  160          0986  1 !
;  161          0987  1 ! IMPLICIT OUTPUTS:
;  162          0988  1 !
;  163          0989  1 !       NONE
;  164          0990  1 !
;  165          0991  1 ! ROUTINE VALUE:
;  166          0992  1 ! COMPLETION CODES:
;  167          0993  1 !
;  168          0994  1 !       NONE
;  169          0995  1 !
;  170          0996  1 ! SIDE EFFECTS:
;  171          0997  1 !
;  172          0998  1 !       Signals if an error is encountered.
;  173          0999  1 !       BAS$$CB_PUSH will signal if the channel number is invalid.
;  174          1000  1 !       Leaves the channel active.
;  175          1001  1 !
;  176          1002  1 !--
;  177          1003  1
;  178          1004  2     BEGIN
;  179          1005  2
;  180          1006  2     BUILTIN FP;
;  181          1007  2
;  182          1008  2     GLOBAL REGISTER
;  183          1009  2         CCB = K_CCB_REG : REF BLOCK [, BYTE];
;  184          1010  2
;  185          1011  2     MAP
;  186          1012  2         DESC : REF BLOCK [8, BYTE];
;  187          1013  2
;  188          1014  2     LOCAL
;  189          1015  2         FMP : REF BLOCK [, BYTE],
;  190          1016  2         ACTUAL_UNIT;
;  191          1017  2
;  192          1018  2 !+
;  193          1019  2 ! Store away FP
;  194          1020  2 !-
;  195          1021  2     FMP = .FP;
```

```
;  196          1022  2 !+
;  197          1023  2 ! If the unit is zero, use the appropriate unit for input side of channel 0.
;  198          1024  2 ! Load register CCB with a pointer to the LUB/ISB/RAB for the channel.
;  199          1025  2 ! Signal an invalid channel number.
;  200          1026  2 !-
;  201          1027  2     IF (.UNIT LSS 0) THEN BAS$$STOP(BAS$K_ILLIO_CHA);
;  202          1028  2     ACTUAL_UNIT = (IF .UNIT EQL 0 THEN LUB$K_LUN_INPU ELSE .UNIT);
;  203          1029  2
;  204          1030  2     BAS$$CB_PUSH (.ACTUAL_UNIT, LUB$K_ILUN_MIN);
;  205          1031  2     CCB [ISB$A_USER_FP] = .FMP [SF$L_SAVE_FP];
;  206          1032  2 !+
;  207          1033  2 ! If the channel number is zero, make sure it is open.
;  208          1034  2 !-
;  209          1035  3     IF ( NOT .CCB[LUB$V_OPENED] )
;  210          1036  2     THEN
;  211          1037  2
;  212          1038  3         IF (.ACTUAL_UNIT EQL LUB$K_LUN_INPU)
;  213          1039  2         THEN
;  214          1040  3                 BEGIN
;  215          1041  3                 BAS$$OPEN_ZERO(.FMP[SF$L_SAVE_FP])
;  216          1042  3                 END
;  217          1043  2         ELSE
;  218          1044  3                 BEGIN
;  219          1045  3                 BAS$$STOP_IO(BAS$K_IO_CHANOT);
;  220          1046  2                 END;
;  221          1047  2
;  222          1048  2 !+
;  223          1049  2 ! Mark this as a MOVE statement.
;  224          1050  2 !-
;  225          1051  2     CCB [ISB$B_STTM_TYPE] = ISB$K_ST_TY_MOV;
;  226          1052  2
;  227          1053  2 !+
;  228          1054  2 ! Return the caller a descriptor of the buffer.
;  229          1055  2 !-
;  230          1056  2     DESC [DSC$W_LENGTH] = .CCB [LUB$W_RBUF_SIZE];
;  231          1057  2     DESC [DSC$B_DTYPE] = DSC$K_DTYPE_Z;
;  232          1058  2     DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
;  233          1059  2     DESC [DSC$A_POINTER] = .CCB [LUB$A_RBUF_ADR];
;  234          1060  2 !+
;  235          1061  2 ! All done.
;  236          1062  2 !-
;  237          1063  2     RETURN
;  238          1064  1     END;                                          ! end of BAS$MOVE_BEG


                                                                        .TITLE  BAS$MOVE
                                                                        .IDENT  \1-006\

                                                                        .EXTRN  BAS$$CB_PUSH, BAS$$CB_POP
                                                                        .EXTRN  BAS$$CB_GET, BAS$$STOP
                                                                        .EXTRN  BAS$$STOP_IO, BAS$$OPEN_ZERO
                                                                        .EXTRN  BAS$K_PROCESSOR
                                                                        .EXTRN  BAS$K_IO_CHANOT
                                                                        .EXTRN  BAS$K_ILLIO_CHA

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

```
                                    080C 00000          .ENTRY  BAS$MOVE_BEG, Save R2,R3,R11                ; 0965
                          53        5D   D0 00002          MOVL    FP, FMP                                     ; 1021
                          52   08   AC   D0 00005          MOVL    UNIT, R2                                    ; 1027
                               08   0B   18 00009          BGEQ    1$                                          ;
                          7E   00G  8F   9A 0000B          MOVZBL  #BAS$K_ILLIO_CHA, -(SP)                     ;
              00000000G   00        01   FB 0000F          CALLS   #1, BAS$$STOP                               ;
                                    52   D5 00016 1$:      TSTL    R2                                          ; 1028
                                    03   12 00018          BNEQ    2$                                          ;
                          52        07   CE 0001A          MNEGL   #7, ACTUAL_UNIT                             ;
                          50        08   CE 0001D 2$:      MNEGL   #8, R0                                      ; 1030
                        0000v000G   00   16 00020          JSB     BAS$$CB_PUSH                                ;
                   FF4C CB     0C   A3   D0 00026          MOVL    12(FMP), -180(CCB)                          ; 1031
                               FC   AB   E8 0002C          BLBS    -4(CCB), 4$                                 ; 1035
              FFFFFFF9 8F              52   D1 00030          CMPL    ACTUAL_UNIT, #-7                            ; 1038
                                    0C   12 00037          BNEQ    3$                                          ;
                               0C   A3   DD 00039          PUSHL   12(FMP)                                     ; 1041
              00000000G   00        01   FB 0003C          CALLS   #1, BAS$$OPEN_ZERO                          ;
                                    0B   11 00043          BRB     4$                                          ; 1040
                          7E   00G  8F   9A 00045 3$:      MOVZBL  #BAS$K_IO_CHANOT, -(SP)                     ; 1045
              00000000G   00        01   FB 00049          CALLS   #1, BAS$$STOP_IO                            ;
                   FF71   CB             2E   90 00050 4$:      MOVB    #46, -143(CCB)                          ; 1051
                          50   04   AC   D0 00055          MOVL    DESC, R0                                    ; 1056
                          60   D2   AB   B0 00059          MOVW    -46(CCB), (R0)                              ;
                       02 A0   0100 8F   B0 0005D          MOVW    #256, 2(R0)                                 ; 1057
                       04 A0   EC   AB   D0 00063          MOVL    -20(CCB), 4(R0)                             ; 1059
                                    04   00068          RET                                                 ; 1064

; Routine Size:  105 bytes,    Routine Base:  _BAS$CODE + 0000

;  239          1065  1
```

```
;  241     1066  1 GLOBAL ROUTINE BAS$MOVE_END                    ! End of MOVE statement
;  242     1067  1     : NOVALUE =
;  243     1068  1
;  244     1069  1 !++
;  245     1070  1 ! FUNCTIONAL DESCRIPTION:
;  246     1071  1 !
;  247     1072  1 !       End a MOVE statement.  This is needed so we know when all uses
;  248     1073  1 !       of the I/O buffer are done, so we can release the buffer after
;  249     1074  1 !       a CLOSE.
;  250     1075  1 !
;  251     1076  1 ! FORMAL PARAMETERS:
;  252     1077  1 !
;  253     1078  1 !       NONE
;  254     1079  1 !
;  255     1080  1 ! IMPLICIT INPUTS:
;  256     1081  1 !
;  257     1082  1 !       OTS$$A_CUR_LUB, the current logical unit.  This had better be
;  258     1083  1 !       doing a MOVE statement.
;  259     1084  1 !
;  260     1085  1 ! IMPLICIT OUTPUTS:
;  261     1086  1 !
;  262     1087  1 !       NONE
;  263     1088  1 !
;  264     1089  1 ! ROUTINE VALUE:
;  265     1090  1 ! COMPLETION CODES:
;  266     1091  1 !
;  267     1092  1 !       NONE
;  268     1093  1 !
;  269     1094  1 ! SIDE EFFECTS:
;  270     1095  1 !
;  271     1096  1 !       Signals if there is no current I/O, or if the current I/O
;  272     1097  1 !       is not a MOVE statement.
;  273     1098  1 !
;  274     1099  1 !--
;  275     1100  1
;  276     1101  2     BEGIN
;  277     1102  2
;  278     1103  2     GLOBAL REGISTER
;  279     1104  2         CCB = K_CCB_REG : REF BLOCK [, BYTE];
;  280     1105  2
;  281     1106  2     BAS$$CB_GET ();
;  282     1107  2
;  283     1108  2     IF (.CCB EQL 0) THEN BAS$$STOP (BAS$K_PROLOSSOR);
;  284     1109  2
;  285     1110  2 !+
;  286     1111  2 ! The channel might not be open due to a CLOSE at AST level.  Give
;  287     1112  2 ! an error in this case.  If we ignored this condition, the following
;  288     1113  2 ! GET or PUT would give the error, so we might as well give it as
;  289     1114  2 ! soon as possible.
;  290     1115  2 !-
;  291     1116  2
;  292     1117  2     IF ( NOT .CCB [LUB$V_OPENED]) THEN BAS$$STOP_IO (BAS$K_IO_CHANOT);
;  293     1118  2
;  294     1119  2 !+
;  295     1120  2 ! If the channel is not doing a MOVE statement we have a serious
;  296     1121  2 ! problem, probably a wild branch into the I/O list.
;  297     1122  2 !-
```

```
;  298          1123  2
;  299          1124  2      IF (.CCB [ISB$B_STTM_TYPE] NEQ ISB$K_ST_TY_MOV) THEN BAS$$STOP_IO (BAS$K_PROLOSSOR);
;  300          1125  2
;  301          1126  2  !+
;  302          1127  2  ! If all those tests succeed, pop the I/O system.
;  303          1128  2  !-
;  304          1129  2      BAS$$CB_POP ();
;  305          1130  2  !+
;  306          1131  2  ! All done.
;  307          1132  2  !-
;  308          1133  2      RETURN
;  309          1134  1      END;                                    ! end of BAS$MOVE_END

                                        0804 00000          .ENTRY  BAS$MOVE_END, Save R2,R11                ; 1066
                              52 00000000G   00   9E 00002  MOVAB   BAS$$STOP_IO, R2                         ;
                                 00000000G   00   16 00009  JSB     BAS$$CB_GET                              ; 1106
                                             5B   D5 0000F  TSTL    CCB                                      ; 1108
                                             0B   12 00011  BNEQ    1$                                       ;
                              7E        00G   8F   9A 00013  MOVZBL  #BAS$K_PROLOSSOR, -(SP)                 ;
                    00000000G 00            01   FB 00017  CALLS   #1, BAS$$STOP                            ;
                              07        FC    AB   E8 0001E 1$:     BLBS    -4(CCB), 2$                      ; 1117
                              7E        00G   8F   9A 00022  MOVZBL  #BAS$K_IO_CHANOT, -(SP)                 ;
                              62              01   FB 00026  CALLS   #1, BAS$$STOP_IO                        ;
                              2E      FF71    CB   91 00029 2$:     CMPB    -143(CCB), #46                   ; 1124
                                              07   13 0002E  BEQL    3$                                       ;
                              7E        00G   8F   9A 00030  MOVZBL  #BAS$K_PROLOSSOR, -(SP)                 ;
                              62              01   FB 00034  CALLS   #1, BAS$$STOP_IO                        ;
                                 00000000G    00   16 00037 3$:     JSB     BAS$$CB_POP                      ; 1129
                                              04   0003D  RET                                                ; 1134

; Routine Size:  62 bytes,    Routine Base:  _BAS$CODE + 0069

;  310          1135  1
;  311          1136  1 END                                         ! end of module BAS$MOVE
;  312          1137  1
;  313          1138  0 ELUDOM
```

```
BAS$MOVE_FROM==     BAS$MOVE_BEG
BAS$MOVE_TO==       BAS$MOVE_BEG
```

```
;                              PSECT SUMMARY
;
;      Name                    Bytes                           Attributes
;
;  _BAS$CODE                     167  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        7         0        581          00:01.2




;                                      COMMAND QUALIFIERS

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

; Size:         167 code + 0 data bytes
; Run Time:       00:10.3
; Elapsed Time:   00:26.9
; Lines/CPU Min:   6661
; Lexemes/CPU-Min: 38985
; Memory Used:  127 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