C 12

```
   SSSSSSSSSSSS  MMM         MMM        GGGGGGGGGGG   RRRRRRRRRRRR     TTTTTTTTTTTTTTT   LLL
   SSSSSSSSSSSS  MMM         MMM        GGGGGGGGGGG   RRRRRRRRRRRR     TTTTTTTTTTTTTTT   LLL
   SSSSSSSSSSSS  MMM         MMM        GGGGGGGGGGG   RRRRRRRRRRRR     TTTTTTTTTTTTTTT   LLL
SSS              MMMMMM   MMMMMM  GGG              RRR         RRR        TTT        LLL
SSS              MMMMMM   MMMMMM  GGG              RRR         RRR        TTT        LLL
SSS              MMMMMM   MMMMMM  GGG              RRR         RRR        TTT        LLL
SSS              MMM   MMM   MMM  GGG              RRR         RRR        TTT        LLL
SSS              MMM   MMM   MMM  GGG              RRR         RRR        TTT        LLL
SSS              MMM   MMM   MMM  GGG              RRR         RRR        TTT        LLL
   SSSSSSSSS     MMM         MMM  GGG              RRRRRRRRRRRR           TTT        LLL
   SSSSSSSSS     MMM         MMM  GGG              RRRRRRRRRRRR           TTT        LLL
   SSSSSSSSS     MMM         MMM  GGG              RRRRRRRRRRRR           TTT        LLL
            SSS  MMM         MMM  GGG   GGGGGGGGG  RRR   RRR              TTT        LLL
            SSS  MMM         MMM  GGG   GGGGGGGGG  RRR   RRR              TTT        LLL
            SSS  MMM         MMM  GGG   GGGGGGGGG  RRR   RRR              TTT        LLL
            SSS  MMM         MMM  GGG         GGG  RRR      RRR           TTT        LLL
            SSS  MMM         MMM  GGG         GGG  RRR      RRR           TTT        LLL
            SSS  MMM         MMM  GGG         GGG  RRR      RRR           TTT        LLL
SSSSSSSSSSSS     MMM         MMM     GGGGGGGGG     RRR         RRR        TTT        LLLLLLLLLLLLLLL
SSSSSSSSSSSS     MMM         MMM     GGGGGGGGG     RRR         RRR        TTT        LLLLLLLLLLLLLLL
SSSSSSSSSSSS     MMM         MMM     GGGGGGGGG     RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

```
_$2

Val
---
001
001
001
001
001
001
001
001
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
```

```
**FILE**ID**SMGDISINP                                      L 9                                                              SMG
                                                                                                                            1-0

  SSSSSSSS  MM      MM    GGGGGGGG  DDDDDDDD      IIIIII      SSSSSSSS    IIIIII    NN      NN  PPPPPPPP
  SSSSSSSS  MM      MM    GGGGGGGG  DDDDDDDD      IIIIII      SSSSSSSS    IIIIII    NN      NN  PPPPPPPP
SS          MMMM  MMMM  GG          DD      DD      II      SS              II      NN      NN  PP      PP
SS          MMMM  MMMM  GG          DD      DD      II      SS              II      NN      NN  PP      PP
SS          MM  MM  MM  GG          DD      DD      II      SS              II      NNNN    NN  PP      PP
SS          MM  MM  MM  GG          DD      DD      II      SS              II      NNNN    NN  PP      PP
  SSSSSS    MM      MM  GG          DD      DD      II        SSSSSS        II      NN  NN  NN  PPPPPPPP
  SSSSSS    MM      MM  GG          DD      DD      II        SSSSSS        II      NN  NN  NN  PPPPPPPP
        SS  MM      MM  GG  GGGGGG  DD      DD      II              SS      II      NN    NNNN  PP
        SS  MM      MM  GG  GGGGGG  DD      DD      II              SS      II      NN    NNNN  PP
        SS  MM      MM  GG      GG  DD      DD      II              SS      II      NN      NN  PP            ....
        SS  MM      MM  GG      GG  DD      DD      II              SS      II      NN      NN  PP            ....
SSSSSSSS    MM      MM    GGGGGG    DDDDDDDD      IIIIII    SSSSSSSS      IIIIII    NN      NN  PP            ....
SSSSSSSS    MM      MM    GGGGGG    DDDDDDDD      IIIIII    SSSSSSSS      IIIIII    NN      NN  PP            ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0 %TITLE 'SMG$$DISPLAY_INPUT - Input support routines'
0 MODULE SMG$$DISPLAY_INPUT (
0               IDENT = '1-026' ! File: SMGDISINP.B32 Edit: STAN1026
0               ) =
1 BEGIN
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! FACILITY:     Screen Management
1 !
1 ! ABSTRACT:
1 !       The procedures in this module act as interfaces between the
1 ! virtual displays and pasteboards and associated data structures, and
1 ! the keyboard input side of SMG.  These routines are called to pass
1 ! information about changes to the physical screen that have been
1 ! brought about by input-related activities.
1 !
1 ! ENVIRONMENT:  User mode, Shared library routines.
1 !
1 ! AUTHOR: R. Reichert, CREATION DATE: 9-Mar-1983
1 !
1 ! MODIFIED BY:
1 !
1 ! 1-001 - Original.  Skeleton for future code.  RKR 9-Mar-1983
1 ! 1-002 - Correct names of data structures and macros.  PLL 15-Mar-1983
1 ! 1-003 - Add $SMG$VALIDATE_ARGCOUNT invocations.  RKR 30-Mar-1983.
1 ! 1-004 - Flesh out some of the routines.  RKR 11-APR-1983.
1 ! 1-005 - Add SMG$$GET_PASTEBOARD_ID.  RKR 14-APR-1983.
1 ! 1-006 - Tap into output side.  RKR 26-APR-1983.
1 ! 1-007 - Debug SMG$$SET_PHYSICAL_CURSOR. RKR 26-APR-1983
1 ! 1-008 - Start debugging SMG$$REPORT_CHANGE_xxxxx. RKR 29-APR-1983
1 ! 1-009 - Take advantage of some new routines.
1 !         RKR 15-MAY-1983.
1 ! 1-010 - Delete external references to DD_ structures and counts -- no
```

```
1 !             longer needed (or available).
1 !             RKR 20-MAY-1983.
1 ! 1-011 - Start using fields in PBCB which tell SMG$$MIN_UPD what part
1 !             of WCB buffers changed.
1 !             RKR 28-MAY-1983.
1 ! 1-012 - Inform output that the physical cursor position is unknown.
1 !             Fix bug wherein NUM_CHARS was treated as the address of
1 !             a longword instead of a word.
1 !             STAN 28-may-1983.
1 ! 1-013 - Further debugging of SMG$$SET_PHYSICAL_CURSOR.
1 !             RKR 27-MAY-1983.
1 ! 1-014 - Rename PBCB_B_COLS to PBCB_W_WIDTH
1 !             STAN 1-May-1983
1 ! 1-015 - Make changed_row and changed_col optional arguments to
1 !             SMG$$REPORT_CHANGE_REPLACE.  PLL 16-Jun-1983
1 ! 1-016 - Rearrange order of arguments to SMG$$SET_PHYSICAL_CURSOR, and
1 !             add two new arguments.  PLL 21-Jun-1983
1 ! 1-017 - In SMG$$SET_PHYSICAL_CURSOR:
1 !                     Validate right number of arguements.
1 !                     Conditionalize some code so that some computations are
1 !                     done only if optional paramter(s) are present.
1 !             In SMG$$MOVE_TEXT_TO_SCREEN_BUFFER:
1 !                     Optimize some code paths to achieve same optimizations
1 !                     as in SMG$$MOVE_TEXT_TO_WINDOW_BUF (in SMGDISOUT.B32).
1 !             Throughout, switch to data structure named declarations.
1 !             RKR 24-JUN-1983.
1 ! 1-018 - Cleanup calling sequence to SMG$$SET_PHYSICAL_CURSOR, based on
1 !             how the input routines call it.  PLL 27-Jun-1983
1 ! 1-019 - Pass physical device type to SMG$$SET_CURSOR_ABS_R4 if the
1 !             device is a foreign terminal.  PLL 2-Aug-1983
1 ! 1-020 - Add logic to SMG$$SET_PHYSICAL_CURSOR to set scrolling region
1 !             to full screen if we are about to do input from a VT100.
1 !             RKR 3-AUG-1983.
1 ! 1-021 - Speed up SMG$$SET_PHYSICAL_CURSOR by looking at occluded
1 !             bit in PP to bypass occlusion tests.  RKR 12-SEP-1983.
1 ! 1-022 - Fix the way SMG$$SET_PHYSICAL_CURSOR calculates remaining columns
1 !             available for input.  No properly recognizing where the virtual
1 !             display is pasted.
1 !             RKR 14-DEC-1983
1 ! 1-023 - Minor changes for TERMTABLE support. STAN 22-JAN-1984
1 ! 1-024 - More. STAN 14-Feb-1984.
1 ! 1-025 - STAN 7-Mar-1984. Moved report_change routines to file SMGPRVINP.B32
1 ! 1-026 - STAN 19-Mar-1984. Fix bug in remaining columns calculation for
1 !             unoccluded virtual display not pasted to column 1 of pasteboard.
1 !--
```

```
1 %SBTTL 'Declarations'
1 !
1 ! SWITCHES:
1 !
1
1 !
1 ! LINKAGES:
1 !
1 !       NONE
1 !
1 ! TABLE OF CONTENTS:
1 !
1
1 FORWARD ROUTINE
1
1 ! Private entry points:
1
1     SMG$$GET_PASTEBOARD_ID,            ! Find pasteboard id to match
1                                        ! device name.
1     SMG$$MOVE_TEXT_TO_SCREEN_BUF,      ! Map a single virtual display
1                                        ! to the WCB screen buffer.
1
1     SMG$$SET_PHYSICAL_CURSOR;          ! Set physical cursor on screen
1
1 !
1 ! INCLUDE FILES
1 !
1
1 REQUIRE 'RTLIN:SMGPROLOG';             ! defines psects, macros, tcb,
1                                        !  wcb, & terminal symbols
1
1 !
1 ! EXTERNAL REFERENCES
1 !
1
1 EXTERNAL
1     PBD_L_COUNT,        ! No. of pasteboards we currently have
1
1     PBD_A_PBCB : VECTOR [PBD_K_MAX_PB, LONG],
1                         ! Table of addresses of PBCB's
1
1     PBD_V_PB_AVAIL : BITVECTOR [PBD_K_MAX_PB];
1                         ! Bit vector or pasteboard id numbers in use.
1
1 EXTERNAL ROUTINE
1     LIB$GET_VM,         ! Allocate heap storage
1
1     SMG$INSERT_CHARS,             ! Insert char in virtual display buffer
1                                   ! and cause output.
1
1     SMG$$FILL_WINDOW_BUFFER,      ! Map all virtual display buffers to
1                                   ! the window buffer for a given PBCB
1
1     SMG$$FLUSH_BUFFER,  ! Flush any pending output to terminal
1
1     SMG$$FORCE_SCROLL_REG,        ! Force scroll region to specified
1                                   ! lines.
```

```
1
1       SMG$$LOCATE_PP,      ! Locate pasting packet that joins a virtual
1                            ! display to a pasteboard.
1
1       SMG$$MOVE_TEXT_TO_WINDOW_BUF,       ! Map single virtual display to
1                                           ! window buffer.
1
1       SMG$$OCCLUDE,        ! Determine overlap between two rectangular
1                            ! areas.
1       SMG$$MIN_UPD,        ! Minimum output routine
1
1       SMG$$PUT_TEXT_TO_BUFFER,     ! Text to virtual display buffer
1
1       SMG$$REPORT_CHANGE_INSERT,          ! Report change to physical
1                                           ! screen in insert mode.
1
1       SMG$$REPORT_CHANGE_REPLACE;         ! Report change to physical
1                                           ! screen in replaec mode.
1
1 EXTERNAL LITERAL
1
1       SMG$_FATERRLIB,      ! Fatal error in library procedure
1       SMG$_INVARG,         ! Invalid argument
1       SMG$_INVCOL,         ! Invalid column number
1       SMG$_INVDIS_ID,      ! Invalid virtual display id
1       SMG$_INVPAS_ID,      ! Invalid pasteboard id
1       SMG$_INVROW;         ! Invalid row number
1
1 !<BLF/PAGE>
```

```
;  191         0266  1 %SBTTL 'SMG$$GET_PASTEBOARD_ID - Get pasteboard id for device'
;  192         0267  1 GLOBAL ROUTINE SMG$$GET_PASTEBOARD_ID (
;  193         0268  1                                    DEVNAM_LEN : REF VECTOR [,WORD],
;  194         0269  1                                    DEVNAM_ADDR,
;  195         0270  1                                    PASTEBOARD_ID
;  196         0271  1                                  ) =
;  197         0272  1 !++
;  198         0273  1 ! FUNCTIONAL DESCRIPTION:
;  199         0274  1 !
;  200         0275  1 !       Try to find a pasteboard which is associated with the device
;  201         0276  1 !       name string provided.   If match found, return pasteboard id.
;  202         0277  1 !       If not found, return SMG$_INVARG.
;  203         0278  1 !
;  204         0279  1 ! CALLING SEQUENCE:
;  205         0280  1 !
;  206         0281  1 !       ret_status.wlc.v = SMG$$GET_PASTEBOARD_ID (
;  207         0282  1 !                                       DEVNAM_LEN.rwu.r,
;  208         0283  1 !                                       DEVNAM_ADDR.rt.r,
;  209         0284  1 !                                       PASTEBOARD_ID.wlu.r)
;  210         0285  1 !
;  211         0286  1 ! FORMAL PARAMETERS:
;  212         0287  1 !
;  213         0288  1 !       DEVNAM_LEN.rwu.r        Address a word containing the length of
;  214         0289  1 !                               the device name string whose pasteboard
;  215         0290  1 !                               id counterpart is sought.
;  216         0291  1 !
;  217         0292  1 !       DEVNAM_ADDR.rt.r        Address of a buffer containing the
;  218         0293  1 !                               device name string whose pasteboard id
;  219         0294  1 !                               is sought.
;  220         0295  1 !
;  221         0296  1 !       PASTEBOARD_ID.wlu.r     Address of the longword to receive the
;  222         0297  1 !                               the pasteboard id that is allocated
;  223         0298  1 !                               to the specified device name.
;  224         0299  1 !
;  225         0300  1 ! IMPLICIT INPUTS:
;  226         0301  1 !
;  227         0302  1 !       Data pertaining to pasteboards currently known.
;  228         0303  1 !
;  229         0304  1 ! IMPLICIT OUTPUTS:
;  230         0305  1 !
;  231         0306  1 !       NONE
;  232         0307  1 !
;  233         0308  1 ! COMPLETION STATUS:
;  234         0309  1 !
;  235         0310  1 !       SS$_NORMAL      Normal successful completion
;  236         0311  1 !       SMG$_WRONUMARG  Wrong number of arguments
;  237         0312  1 !       SMG$_INVARG     No pasteboard on file matches given device
;  238         0313  1 !                       name string.
;  239         0314  1 !
;  240         0315  1 ! SIDE EFFECTS:
;  241         0316  1 !
;  242         0317  1 !       NONE
;  243         0318  1 !--
;  244         0319  1
;  245         0320  2     BEGIN
;  246         0321  2     LOCAL
;  247         0322  2         PBCB : REF $PBCB_DECL;          ! Address of a pasteboard
```

```
2                                       . control block.
2
2    $SMG$VALIDATE_ARGCOUNT (3, 3);      ! Test for right no. of args
2
2 !+
2 ! If we don't have any pasteboards yet, it can't match.  Return error.
2 !-
2    IF .PBD_L_COUNT LEQ 0
2    THEN
2        RETURN ( SMG$_INVARG);
2
2 !+
2 ! Loop through all the pasteboards we've got trying to match name.
2 !-
2    INCR I FROM 0 TO .PBD_L_COUNT -1
2    DO
3        BEGIN   ! Loop thru pasteboards
3        IF (PBCB = .PBD_A_PBCB [.I]) NEQ 0
3        THEN
4            BEGIN       ! A valid pasteboard address
4            IF .DEVNAM_LEN [0] EQL .PBCB [PBCB_W_DEVNAM_LEN]
4            THEN
5                BEGIN ! Lengths match
5                IF CH$EQL ( .DEVNAM_LEN [0],             ! length
5                            .DEVNAM_ADDR,                 ! address
5                            .PBCB [PBCB_W_DEVNAM_LEN],    ! length
5                            PBCB [PBCB_T_DEVNAM])         ! address
5                THEN
6                    BEGIN       ! Match found
6                    .PASTEBOARD_ID = .I;
6                    RETURN ( SS$_NORMAL);
5                    END;        ! Match found
4                END;  ! Lengths match
3            END;        ! A valid pasteboard address
2        END;    ! Loop thru pasteboards
2
2 !+
2 ! If we fall out of loop, none of our pasteboards are associated with
2 ! the given string.  Return error code.
2 !-
2    RETURN (SMG$_INVARG);
1    END;                    ! End of routine SMG$$GET_PASTEBOARD_ID


;                                                        .TITLE  SMG$$DISPLAY_INPUT SMG$$DISPLAY_INPUT - Input s
                                                                                   upport routines
                                                         .IDENT  \1-026\

                                                         .EXTRN  PBD_L_COUNT, PBD_A_PBCB
                                                         .EXTRN  PBD_V_PB_AVAIL, LIB$GET_VM
                                                         .EXTRN  SMG$INSERT_CHARS
                                                         .EXTRN  SMG$$FILL_WINDOW_BUFFER
                                                         .EXTRN  SMG$$FLUSH_BUFFER
                                                         .EXTRN  SMG$$FORCE_SCROLL_REG
                                                         .EXTRN  SMG$$LOCATE_PP, SMG$$MOVE_TEXT_TO_WINDOW_BUF
                                                         .EXTRN  SMG$$OCCLUDE, SMG$$MIN_UPD
                                                         .EXTRN  SMG$$PUT_TEXT_TO_BUFFER
```

```
                                                                   F 10
SMG$$DISPLAY_IN SMG$$DISPLAY_INPUT - Input support routines       16-Sep-1984 00:27:47     VAX-11 Bliss-32 V4.0-742              Page   7         SM
1-026           SMG$$GET_PASTEBOARD_ID - Get pasteboard id for    14-Sep-1984 13:09:42     [SMGRTL.SRC]SMGDISINP.B32;1                (3)          1-

                                                                             .EXTRN  SMG$$REPORT_CHANGE_INSERT
                                                                             .EXTRN  SMG$$REPORT_CHANGE_REPLACE
                                                                             .EXTRN  SMG$_FATERRLIB, SMG$_INVARG
                                                                             .EXTRN  SMG$_INVCOL, SMG$_INVDIS_ID
                                                                             .EXTRN  SMG$_INVPAS_ID, SMG$_INVROW
                                                                             .EXTRN  SMG$_WRONUMARG

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

                                                         007C 00000          .ENTRY  SMG$$GET_PASTEBOARD_ID, Save R2,R3,R4,R5,R6 ; 0267
                                        03             6C  91 00002          CMPB    (AP), #3                                      ; 0325
                                                       08  13 00005          BEQL    1$                                            ;
                                        50 00000000G   8F  D0 00007          MOVL    #SMG$_WRONUMARG, R0                           ;
                                                           04 0000E          RET                                                   ;
                                        56 00000000G   00  D0 0000F 1$:      MOVL    PBD_L_COUNT, R6                               ; 0330
                                                       2E  15 00016          BLEQ    4$                                            ;
                                        55             01  CE 00018          MNEGL   #1, I                                         ; 0337
                                                       25  11 0001B          BRB     3$                                            ;
                                        54 00000000G0045  D0 0001D 2$:       MOVL    PBD_A_PBCB[I], PBCB                           ; 0340
                                                       1B  13 00025          BEQL    3$                                            ;
                                   12   A4        04   BC  B1 00027          CMPW    @DEVNAM_LEN, 18(PBCB)                         ; 0343
                                                       14  12 0002C          BNEQ    3$                                            ;
      12   A4             00       08   BC        04   BC  2D 0002E          CMPC5   @DEVNAM_LEN, @DEVNAM_ADDR, #0, 18(PBCB), -    ; 0349
                                                  18   A4     00036                  24(PBCB)                                      ;
                                                       08  12 00038          BNEQ    3$                                            ;
                                   0C   BC             55  D0 0003A          MOVL    I, @PASTEBOARD_ID                             ; 0352
                                        50             01  D0 0003E          MOVL    #1, R0                                        ; 0353
                                                           04 00041          RET                                                   ;
                          D7            55             56  F2 00042 3$:      AOBLSS  R6, I, 2$                                     ; 0337
                                        50 00000000G   8F  D0 00046 4$:      MOVL    #SMG$_INVARG, R0                              ; 0363
                                                           04 0004D          RET                                                   ; 0364

; Routine Size:  78 bytes,    Routine Base:  _SMG$CODE + 0000

;  290         0365  1 !<BLF/PAGE>
```

```
%SBTTL 'SMG$$MOVE_TEXT_TO_SCREEN_BUF - Move text from display buf. to screen buf.'
GLOBAL ROUTINE SMG$$MOVE_TEXT_TO_SCREEN_BUF (
                                  PP : REF BLOCK [,BYTE]
                                            ) =
!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine moves text from the buffer located at
!       .DCB [DCB_A_TEXT_BUF] into the screen text buffer.
!       Array of bytes at .DCB [ DCB_A_ATTR_BUF ] describe the
!       rendition this text must assume and is moved into the
!       associated screen attribute buffer.
!       Similarly, if the alternate character set buffer at
!       .DCB [DCB_A_CHAR_SET_BUF] exists, it must be mapped into the
!       screen alternate character set buffer.
!
! CALLING SEQUENCE:
!
!       ret_status.wlc.v = SMG$$MOVE_TEXT_TO_SCREEN_BUF ( PP.rab.r)
!
! FORMAL PARAMETERS:
!
!       PP.rab.r                Address of pasting packet.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! COMPLETION STATUS:
!
!       SS$_NORMAL      Normal successful completion
!
! SIDE EFFECTS:
!
!       NONE
!--

    BEGIN
    LOCAL
        DCB  : REF $DCB_DECL,           ! Addr of virtual display
                                        ! control block.
        PBCB : REF $PBCB_DECL,          ! Addr of pasteboard control
                                        ! block
        WCB  : REF $WCB_DECL,           ! Addr of window control block
        FROM_INDEX,                     ! Index into source buffer
        TO_INDEX;                       ! Index into destination buffer

    DCB  = .PP [PP_A_DCB_ADDR];
    PBCB = .PP [PP_A_PBCB_ADDR];
    WCB  = .PBCB [PBCB_A_WCB];

    FROM_INDEX = .PP [PP_W_FROM_INDEX];
    TO_INDEX   = .PP [PP_W_TO_INDEX];
```

```
2
2 !+
2 ! Before diverging on two copying paths, check to see if we are going
2 ! to get involved with alternate character set buffers.  If one exists
2 ! in the DCB but does not yet exist in the WCB, we have to allocate
2 ! one for the WCB and initialize it.
2 !-
2     IF .DCB [DCB_A_CHAR_SET_BUF]     NEQ 0      AND
2        .WCB [WCB_A_SCR_CHAR_SET_BUF] EQL 0
2     THEN
3         BEGIN   ! Alloc. and init. window alternate char. set buffer
3         LOCAL
3             STATUS;     ! Status of LIB$GET_VM call
3
4         IF NOT (STATUS = LIB$GET_VM ( WCB [WCB   _BUFSIZE],
4                                       WCB [W^B_A_SCR_CHAR_SET_BUF]))
3         THEN
3             RETURN (.STATUS);
3
3         CH$FILL ( 0, .WCB [WCB_L_BUFSIZE], .WCB [WCB_A_SCR_CHAR_SET_BUF]);
2         END;    ! Alloc. and init. window alternate char. set buffer
2
2 !+
2 ! Check to see if we can do it with a single CH$MOVE or whether we must
2 ! do it a row at a time.
2 !-
2     IF .PP [PP_V_CONTIG]
2     THEN
3         BEGIN   ! Can be done in single move
3
3         ! Move text
3         !
3         CH$MOVE ( .PP [PP_L_MOVE_SIZE],                           ! length
3                   .DCB [DCB_A_TEXT_BUF] + .FROM_INDEX,            ! source
3                   .WCB [WCB_A_SCR_TEXT_BUF] + .TO_INDEX);         ! dest.
3
3         ! Move attributes
3         !
3         CH$MOVE ( .PP [PP_L_MOVE_SIZE],                           ! length
3                   .DCB [DCB_A_ATTR_BUF] + .FROM_INDEX,            ! source
3                   .WCB [WCB_A_SCR_ATTR_BUF] + .TO_INDEX);         ! dest.
3
3         ! Move alternate character set buffer pieces, if necessary
3         !
3         IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
3         THEN
4             BEGIN       ! Map alternate character set
4             CH$MOVE ( .PP [PP_L_MOVE_SIZE],                       ! Length
4                       .DCB [DCB_A_CHAR_SET_BUF] + .FROM_INDEX,    ! Source
4                       .WCB [WCB_A_SCR_CHAR_SET_BUF] + .TO_INDEX);! Dest.
3             END;        ! Map alternate character set
3         END             ! Can be done in single move
3
2     ELSE
2
3         BEGIN   ! Must be done row at a time
3         LOCAL
```

```
3               DCB_COLS,     ! = .DCB [DCB_W_NO_COLS]
3               WCB_COLS;     ! = .WCB [WCB_W_NO_COLS]
3
3           !+
3           ! Extracting out these two words as local longwords makes
3           ! compiler generate better code for this critical loop.
3           !-
3           DCB_COLS = .DCB [DCB_W_NO_COLS];
3           WCB_COLS = .WCB [WCB_W_NO_COLS];
3
3           INCR R FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
3           DO
4               BEGIN         ! For all rows in this display
4               ! Move text
4               !
4               CH$MOVE ( .PP [PP_W_MOVE_LENGTH],                     ! length
4                         .DCB [DCB_A_TEXT_BUF] + .FROM_INDEX,        ! source
4                         .WCB [WCB_A_SCR_TEXT_BUF] + .TO_INDEX);     ! dest.
4
4               ! Move attributes
4               !
4               CH$MOVE ( .PP [PP_W_MOVE_LENGTH],                     ! length
4                         .DCB [DCB_A_ATTR_BUF] + .FROM_INDEX,        ! source
4                         .WCB [WCB_A_SCR_ATTR_BUF] + .TO_INDEX);     ! dest
4
4               FROM_INDEX = .FROM_INDEX + .DCB_COLS;
4               TO_INDEX   = .TO_INDEX   + .WCB_COLS;
3               END;          ! For all rows in this display
3
3           !+
3           ! Move alternate character set buffer pieces, if necessary.
3           !-
3           IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
3           THEN
4               BEGIN         ! Map alt. char. set buffer
4               FROM_INDEX = .PP [PP_W_FROM_INDEX];
4               TO_INDEX   = .PP [PP_W_TO_INDEX];
4
4               INCR R FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
4               DO
5                   BEGIN
5                   CH$MOVE ( .PP [PP_W_MOVE_LENGTH],                 ! length
5                         .DCB [DCB_A_CHAR_SET_BUF] + .FROM_INDEX,    ! source
5                         .WCB [WCB_A_SCR_CHAR_SET_BUF] + .TO_INDEX);! dest.
5
5                   FROM_INDEX = .FROM_INDEX + .DCB_COLS;
5                   TO_INDEX   = .TO_INDEX   + .WCB_COLS;
4                   END;
3               END;          ! Map alt. char. set buffer
2           END;    ! Must be done row at a time
2
2       RETURN (SS$_NORMAL);
1       END;                 ! End of routine SMG$$MOVE_TEXT_TO_SCREEN_BUF
```

```
                                         0FFC 00000         .ENTRY  SMG$$MOVE_TEXT_TO_SCREEN_BUF, Save R2,R3,-  ; 0367
                                                                    R4,R5,R6,R7,R8,R9,R10,R11                   ;
                   5E          10   C2 00002         SUBL2   #16, SP                                     ;
                   57     04   AC   D0 00005         MOVL    PP, R7                                      ; 0417
                   59     10   A7   D0 00009         MOVL    16(R7), DCB                                 ;
                   50     14   A7   D0 0000D         MOVL    20(R7), PBCB                                ; 0418
                   56     08   A0   D0 00011         MOVL    8(PBCB), WCB                                ; 0419
                   5A     1E   A7   3C 00015         MOVZWL  30(R7), FROM_INDEX                          ; 0421
                   58     20   A7   3C 00019         MOVZWL  32(R7), TO_INDEX                            ; 0422
                   5B     18   A9   D0 0001D         MOVL    24(DCB), R11                                ; 0430
                               7E   D4 00021         CLRL    -(SP)                                       ;
                               5B   D5 00023         TSTL    R11                                         ;
                               20   13 00025         BEQL    2$                                          ;
                               6E   D6 00027         INCL    (SP)                                        ;
                          1C   A6   D5 00029         TSTL    28(WCB)                                     ; 0431
                               19   12 0002C         BNEQ    2$                                          ;
                          1C   A6   9F 0002E         PUSHAB  28(WCB)                                     ; 0438
                          28   A6   9F 00031         PUSHAB  40(WCB)                                     ; 0437
            00000000G  00          02   FB 00034         CALLS   #2, LIB$GET_VM                              ; 0438
                       01          50   E8 0003B         BLBC    STATUS, 1$                                  ;
                                   04 0003E         RET                                                 ;
 28    A6        00        6E          00   2C 0003F 1$:     MOVC5   #0, (SP), #0, 40(WCB), @28(WCB)             ; 0442
                          1C   B6      00045                                                     ;
           1F        2A   A7          01   E1 00047 2$:     BBC     #1, 42(R7), 3$                              ; 0449
       14 B648       10 B94A     2B   A7   28 0004C         MOVC3   43(R7), @16(DCB)[FROM_INDEX], @20(WCB)-     ; 0457
                                                                    [TO_INDEX]                                  ;
       18 B648       14 B94A     2B   A7   28 00055         MOVC3   43(R7), @20(DCB)[FROM_INDEX], @24(WCB)-     ; 0463
                                                                    [TO_INDEX]                                  ;
                        62          6E   E9 0005E         BLBC    (SP), 8$                                    ; 0467
       1C B648        6A4B     2B   A7   28 00061         MOVC3   43(R7), (FROM_INDEX)[R11], @28(WCB)-        ; 0472
                                                                    [TO_INDEX]                                  ;
                               58   11 00069         BRB     8$                                          ; 0449
              10    AE     06   A9   3C 0006B 3$:     MOVZWL  6(DCB), DCB_COLS                            ; 0487
              0C    AE     06   A6   3C 00070         MOVZWL  6(WCB), WCB_COLS                            ; 0488
              08    AE     1C   A7   3C 00075         MOVZWL  28(R7), 8(SP)                               ; 0490
                          04   AE   D4 0007A         CLRL    R                                           ; 0506
                               1A   11 0007D         BRB     5$                                          ;
       14 B648       10 B94A     22   A7   28 0007F 4$:     MOVC3   34(R7), @16(DCB)[FROM_INDEX], @20(WCB)-     ; 0497
                                                                    [TO_INDEX]                                  ;
       18 B648       14 B94A     22   A7   28 00088         MOVC3   34(R7), @20(DCB)[FROM_INDEX], @24(WCB)-     ; 0503
                                                                    [TO_INDEX]                                  ;
                        5A     10   AE   C0 00091         ADDL2   DCB_COLS, FROM_INDEX                        ; 0505
                        58     0C   AE   C0 00095         ADDL2   WCB_COLS, TO_INDEX                          ; 0506
            E0        04   AE     08   AE   F3 00099 5$:     AOBLEQ  8(SP), R, 4$                                ; 0490
                        21          6E   E9 0009F         BLBC    (SP), 8$                                    ; 0512
                        5A     1E   A7   3C 000A2         MOVZWL  30(R7), FROM_INDEX                          ; 0515
                        58     20   A7   3C 000A6         MOVZWL  32(R7), TO_INDEX                            ; 0516
                               59   D4 000AA         CLRL    R                                           ; 0523
                               10   11 000AC         BRB     7$                                          ;
       1C B648        6A4B     22   A7   28 000AE 6$:     MOVC3   34(R7), (FROM_INDEX)[R11], @28(WCB)-        ;
                                                                    [TO_INDEX]                                  ;
                        5A     10   AE   C0 000B6         ADDL2   DCB_COLS, FROM_INDEX                        ; 0525
                        58     0C   AE   C0 000BA         ADDL2   WCB_COLS, TO_INDEX                          ; 0526
            EB          59     08   AE   F3 000BE 7$:     AOBLEQ  8(SP), R, 6$                                ; 0518
                        50          01   D0 000C3 8$:     MOVL    #1, R0                                      ; 0531
                                   04 000C6         RET                                                 ; 0532
```

```
; Routine Size:  199 bytes,    Routine Base:  _SMG$CODE + 004E


;  459          0533  1 !<BLF/PAGE>
```

```
1 %SBTTL 'SMG$$SET_PHYSICAL_CURSOR - Set physical cursor'
1 GLOBAL ROUTINE SMG$$SET_PHYSICAL_CURSOR ( DISPLAY_ID,
1                                           PASTEBOARD_ID,
1                                           OUT_BUFFER,
1                                           OUT_BUFFER_LEN : REF VECTOR [,LONG],
1                                           REMAINING_COLS,
1                                           DESIRED_ROW,
1                                           DESIRED_COL
1                                         ) =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Causes the set physical cursor sequence to be deposited in the
1 !       specified buffer.
1 !
1 !       If the terminal we are about to do input from is one with a
1 !       settable scrolling region, we make sure that the scrolling
1 !       region covers the whole screen.
1 !
1 !       If the desired row and column are not specified, the current
1 !       cursor position within the virtual display are assumed.
1 !       REMAINING_COLS is set to the number of columns that will be
1 !       visible within the interval from the indicated cursor position
1 !       and the first column which is no longer within this display.
1 !       This allows the caller to control inputted strings from running
1 !       outside of the virtual  display.
1 !       If REMAINING_COLS is zero, the desired position corresponds to a
1 !       spot on the screen which is currently occluded by another
1 !       virtual display.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       ret_status.wlc.v = SMG$$SET_PHYSICAL_CURSOR (
1 !                                       DISPLAY_ID.rl.r,
1 !                                       PASTEBOARD_ID.rl.r,
1 !                                       OUT_BUFFER.wl.r,
1 !                                       OUT_BUFFER_LEN.ml.r,
1 !                                       REMAINING_COLS.wl.r,
1 !                                       [,DESIRED_ROW.rl.r]
1 !                                       [,DESIRED_COL.rl.r])
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       DISPLAY_ID.rl.r         Display id of virtual display.
1 !
1 !       PASTEBOARD_ID.rl.r      Pasteboard id.
1 !
1 !       OUT_BUFFER.rl.r         Address of a buffer in
1 !                               which to place the set cursor sequence.
1 !                               This buffer should be at least 15 bytes.
1 !
1 !       OUT_BUFFER_LEN.ml.r     Address of a word which
1 !                               contains the current length of OUT_BUFFER.
1 !                               It will be updated to reflect length of
1 !                               the set cursor sequence added.
1 !
1 !       REMAINING_COLS.wl.r     Returned number of visible columns
```

```
1 !                                           available on the physical screen at
1 !                                           this point which are entirely within
1 !                                           the specified virtual display and are
1 !                                           not occluded by another virtual display.
1 !
1 !         [,DESIRED_ROW.rl.r]               Desired row number within the virtual
1 !                                           display to which the physical cursor
1 !                                           should be postioned.  If omitted, the
1 !                                           current cursor row in this virtual
1 !                                           display is assumed.
1 !
1 !         [,DESIRED_COL.rl.r]               Desired column number within the virtual
1 !                                           display to which the physical cursor
1 !                                           should be postioned.  If omitted, the
1 !                                           current cursor column in this virtual
1 !                                           display is assumed.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !         NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !         NONE
1 !
1 ! COMPLETION STATUS:
1 !
1 !         SS$_NORMAL        Normal successful completion
1 !         SMG$_INVDIS_ID    Invalid Display Id
1 !         SMG$_INVPAS_ID    Invalid Pasteboard Id
1 !         SMG$_INVROW       Invalid row specified
1 !         SMG$_INVCOL       Invalid column specified
1 !
1 ! SIDE EFFECTS:
1 !
1 !         NONE
1 !--
1
2     BEGIN
2     BUILTIN
2         NULLPARAMETER;
2
2     LOCAL
2         STATUS,                           ! Status of subroutine calls
2         ROW,                              ! Row of interest -- in virtual display
2         COL,                              ! Column of interest -- in virtual disp.
2         DCB  : REF $DCB_DECL,             ! Addr of display control block
2         PBCB : REF $PBCB_DECL,            ! Addr of pasteboard control
2                                           ! block.
2         WCB  : REF $WCB_DECL,             ! Addr of window control block
2         PP   : REF $PP_DECL;              ! Addr of pasting packet
2
2
2     $SMG$VALIDATE_ARGCOUNT (5, 7);        ! Test for right no. of args
2
2     $SMG$GET_DCB (.DISPLAY_ID, DCB);                      ! Get DCB addr.
2     $SMG$GET_PBCB ( .PASTEBOARD_ID, PBCB) ;               ! Get PBCB addr.
```

```
2       WCB = .PBCB [PBCB_A_WCB];
2
2   !+
2   ! If we are reading from a VT100 (i.e., a device with a settable
2   ! scrolling region), we force the physical scrolling region to be the
2   ! whole screen.  If we didn't do this, the user might be trying to input
2   ! into the last line of a virtual display which might coincide with the
2   ! bottom line of a scrolling region.  As he types the <CR> to terminate
2   ! his input, the virtual display will scroll out from under us because
2   ! of the <CR><LF> echoed by the terminal driver.
2   ! This way, the only way he can get into trouble is if he tries to
2   ! input into the last physical line of the screen.
2   !-
2
2   !**** SHOULD CHANGE WITH TERMTABLE SUPPORT
2
2       IF .?BCB [PBCB_B_DEVTYPE] EQL VT100
2       THEN
3           BEGIN   ! Scroll region check
3           IF .PBCB [PBCB_W_TOP_SCROLL_LINE] NEQ 1             OR
3              .PECB [PBCB_W_BOT_SCROLL_LINE] NEQ .PBCB [PBCB_B_ROWS]
3           THEN
4               BEGIN       ! Set scroll to whole screen
4               LOCAL
4                   STATUS;
5               IF NOT (STATUS = SMG$$FORCE_SCROLL_REG (
5                                                   .PBCB,
5                                                   1,
5                                                   .PBCB [PBCB_B_ROWS]))
4               THEN
4                   RETURN .STATUS;
4               !+
4               ! Flush out this transaction separately.  We don't what it
4               ! to become part of the cursor setting sequence which our
4               ! caller may want to output a second time.
4               !-
4               IF .PBCB [PBCB_V_BUF_ENABLED]
4               THEN
4                   SMG$$FLUSH_BUFFER ( .PBCB);
3               END;        ! Set scroll to whole screen
2           END;    ! Scroll region check
2
2   !+
2   ! If this display is not pasted to this pasteboard, quit -- but flush
2   ! the buffer on the way out anyway.
2   !-
3       IF NOT (STATUS = SMG$$LOCATE_PP (.DCB, .PBCB, PP)) ! Get PP addr.
2       THEN
3           BEGIN
3           IF .PBCB [PBCB_V_BUF_ENABLED]
3           THEN
3               SMG$$FLUSH_BUFFER ( .PBCB);
3           RETURN (.STATUS);
2           END;
2
3       ROW = (IF NOT NULLPARAMETER (DESIRED_ROW) THEN ..DESIRED_ROW
2                                                 ELSE .DCB [DCB_W_CURSOR_ROW]);
```

```
2
3       COL = (IF NOT NULLPARAMETER (DESIRED_COL) THEN ..DESIRED_COL
2                                                   ELSE .DCB [DCB_W_CURSOR_COL]);
2
2       $SMG$VALIDATE_ROW_COL ( .ROW, .COL);
2
2   !+
2   ! Perform remaining column calculation.
2   !-
2       IF .PP [PP_V_OCCLUDED]
2       THEN
3           BEGIN   ! Remaining number of col. calculation. -occluded case.
3           LOCAL
3               TEMP : BLOCK [8,BYTE],       ! Temporary representation of
3                                            ! area of input line in pasteboard
3                                            ! coordinates.
3               CURR_PP : REF $PP_DECL;      ! Addr of a pasting packet
3           !+
3           ! Identify a rectangular area that consists of the part of the
3           ! line bounded by ROW and COL and the right end of the same
3           ! line.  This represents the maximum allowable input area.
3           !-
3           TEMP [DCB_W_RCW_START] = .ROW + .PP [PP_W_ROW] -1;
3           TEMP [DCB_W_NO_ROWS]   = 1;
3           TEMP [DCB_W_COL_START] = .COL + .PP [PP_W_COL] - 1;
3           TEMP [DCB_W_NO_COLS]   = .PP [PP_W_MOVE_LENGTH] -
3                                    ( .COL + .PP [PP_W_COL]) + 2;
3
3           !+
3           ! Check the rectangle isolated above against the projections of
3           ! all virtual displays on this same pasteboard which may occlude
3           ! parts or all of this input rectangle.  The most resticted
3           ! length of the input line is what we return to our caller.
3           ! We start with the one pasted next in the chain (if any).
3           !-
3           CURR_PP = .PP [PP_A_PREV_PBCB];
3           WHILE .CURR_PP NEQ PBCB [PBCB_A_PP_NEXT] ! While more displays
3           DO                                       ! remain...
4               BEGIN       ! Overall loop
4               LOCAL
4                   PP_BASE : REF $PP_DECL;     ! Base address of current
4                                               ! pasting packet
4
4               PP_BASE = .CURR_PP - PP_PBCB_QUEUE_OFFSET;
4                                       ! Since the queue headers for this part
4                                       ! of the chain are not at relative 0 in
4                                       ! the pasting packet.
4
4               IF .PP_BASE [PP_W_MOVE_LENGTH] NEQ 0
4               THEN
5                   BEGIN   ! Display visible
5                   LOCAL
5                       OVERLAP : BLOCK [8,BYTE],   ! Representation of
5                                                   ! overlap between input
5                                                   ! line and a
5                                                   ! higher-pasted virtual
5                                                   ! display.
```

```
    TEMP1   : BLOCK [8,BYTE], ! Representation of
                            ! current virtual display
                            ! in pasteboard
                            ! coordinates.
    NEW_DCB : REF $DCB_DECL;! Addr of display control
                            ! .lock currently involved.

NEW_DCB = .PP_BASE [PP_A_DCB_ADDR];

TEMP1 [DCB_W_ROW_START] = .PP_BASE [PP_W_ROW];
TEMP1 [DCB_W_NO_ROWS]   = .NEW_DCB [ DCB_W_NO_ROWS];
TEMP1 [DCB_W_COL_START] = .PP_BASE [PP_W_COL];
TEMP1 [DCB_W_NO_COLS]   = .NEW_DCB [DCB_W_NO_COLS];

!+
! If the virtual display we're looking at is bordered,
! it will have a bigger footprint in the pasteboard
! buffer than its dimensions alone.  Adjust for the
! increased size of the foot print.
!-
IF .NEW_DCB [DCB_V_BORDERED]
THEN
    BEGIN
    TEMP1 [DCB_W_ROW_START] = .TEMP1 [DCB_W_ROW_START] - 1;
    TEMP1 [DCB_W_NO_ROWS]   = .TEMP1 [DCB_W_NO_ROWS] + 2;
    TEMP1 [DCB_W_COL_START] = .TEMP1 [DCB_W_COL_START] - 1;
    TEMP1 [DCB_W_NO_COLS]   = .TEMP1 [DCB_W_NO_COLS] + 2;
    END;

IF SMG$$OCCLUDE ( TEMP, ! The input line
                  TEMP1,! The higher-pasted display
                  OVERLAP) ! Overlapping region (if any)
THEN
    BEGIN       ! Overlap
    IF .TEMP [DCB_W_COL_START] GEQ
       .OVERLAP [DCB_W_COL_START]         AND
       .TEMP [DCB_W_COL_START] LEQ
       .OVERLAP [DCB_W_COL_START] + .OVERLAP [DCB_W_NO_COLS] - 1
    THEN
        BEGIN   ! Requested start pos. occluded
        .REMAINING_COLS = 0;
        IF .PBCB [PBCB_V_BUF_ENABLED]
        THEN
            SMG$$FLUSH_BUFFER ( .PBCB);
        RETURN (SS$_NORMAL);      ! *** Should this be a
                                  ! failure status ??? ***
        END             ! Requested start pos. occluded
    ELSE
        BEGIN   ! Tail end of input pos occluded
        !+
        ! Truncate length of input line down to the part
        ! that is not occluded.
        !-
        TEMP [DCB_W_NO_COLS] = .TEMP1 [DCB_W_COL_START]
                              - .TEMP [DCB_W_COL_START];
        IF .TEMP [DCB_W_NO_COLS] LEQ 0
        THEN
```

```
;  746          0819  8                           BEGIN
;  747          0820  8                           .REMAINING_COLS = 0;
;  748          0821  8                           IF .PBCB [PBCB_V_BUF_ENABLED]
;  749          0822  8                           THEN
;  750          0823  8                               SMG$$FLUSH_BUFFER ( .PBCB);
;  751          0824  8                           RETURN (SS$_NORMAL); ! *** Should this be a
;  752          0825  8                                                ! failure status ??? ***
;  753          0826  7                           END;
;  754          0827  6                       END;     ! Tail end of input pos occluded
;  755          0828  5                   END;         ! Overlap
;  756          0829  4               END;     ! Display visible
;  757          0830  4
;  758          0831  4           !+
;  759          0832  4           ! Walk the chain backwards, from the current packet back to
;  760          0833  4           ! the head of the chain -- since the most recently pasted
;  761          0834  4           ! displays are at the head of the chain.
;  762          0835  4           !-
;  763          0836  4           CURR_PP = .PP_BASE [PP_A_PREV_PBCB];
;  764          0837  3           END;         ! Overall loop
;  765          0838  3
;  766          0839  3       !+
;  767          0840  3       ! If we fall out of the bottom of the loop, the requested row
;  768          0841  3       ! and column is not occluded, and some non-zero portion of the
;  769          0842  3       ! remainder of the row is visible as well.  Return its length to
;  770          0843  3       ! caller.
;  771          0844  3       !-
;  772          0845  3       .REMAINING_COLS = .TEMP [DCB_W_NO_COLS];
;  773          0846  3
;  774          0847  3       END     ! Remaining number of col. calculation. - occluded case
;  775          0848  3
;  776          0849  2   ELSE
;  777          0850  3       BEGIN   ! Not occluded case
;  778          0851  3       .REMAINING_COLS = .PP [PP_W_MOVE_LENGTH] -  .COL  + 1;
;  779          0852  2       END;    ! Not occluded case
;  780          0853  2
;  781          0854  2 !+
;  782          0855  2 ! All that remains to to set the cursor where requested -- both in the
;  783          0856  2 ! virtual display and on the physical screen.
;  784          0857  2 !-
;  785          0858  2   DCB [DCB_W_CURSOR_ROW] = .ROW;
;  786          0859  2   DCB [DCB_W_CURSOR_COL] = .COL;
;  787          0860  2   WCB [WCB_W_CURR_CUR_ROW] = .ROW + .PP [PP_W_ROW] - 1;
;  788          0861  2   WCB [WCB_W_CURR_CUR_COL] = .COL + .PP [PP_W_COL] - 1;
;  789          0862  2
;  790          0863  2 !+
;  791          0864  2 ! Don't output the sequence to the terminal.  Store the set cursor sequence
;  792          0865  2 ! in the specified buffer (if it and its length provided), and let the
;  793          0866  2 ! caller output the sequence if desired.
;  794          0867  2 ! (SMG$INPUT will store this sequence in its QIO buffer.)
;  795          0868  2 !-
;  796          0869  2
;  797        P 0870  2   $SMG$GET_TERM_DATA(SET_CURSOR_ABS,
;  798        P 0871  2                           .WCB [WCB_W_CURR_CUR_ROW],
;  799          0872  2                           .WCB [WCB_W_CURR_CUR_COL]);
;  800          0873  2
;  801          0874  2   ! Move it to the OUT_BUFFER
;  802          0875  2
;  803          0876  2     CH$MOVE(.PBCB[PBCB_L_CAP_LENGTH],.PBCB[PBCB_A_CAP_BUFFER],.OUT_BUFFER);
;  804          0877  2     OUT_BUFFER_LEN [0] = .PBCB[PBCB_L_CAP_LENGTH];
;  805          0878  2
;  806          0879  2     RETURN (SS$_NORMAL);
;  807          0880  1     END:                    ! End of routine SMG$$SET_PHYSICAL_CURSOR


                                                                 .EXTRN  SMG$GET_TERM_DATA

                                               07FC 00000        .ENTRY  SMG$$SET_PHYSICAL_CURSOR, Save R2,R3,R4,R5,-; 0535
                                                                         R6,R7,R8,R9,R10
                        5A 00000000G  00  9E 00002               MOVAB   SMG$$FLUSH_BUFFER, R10                       ;
                        5E            20  C2 00009               SUBL2   #32, SP                                      ;
              50        6C            05  83 0000C               SUBB3   #5, (AP), DIFF                               ; 0644
                        02            50  91 00010               CMPB    DIFF, #2                                     ;
                                      08  1B 00013               BLEQU   1$                                           ;
                        50 00000000G  8F  D0 00015               MOVL    #SMG$_WRONUMARG, R0                          ;
                                          04 0001C               RET                                                  ;
                        50       04   BC  D0 0001D 1$:           MOVL    @DISPLAY_ID, R0                              ; 0646
                   04   BC       38   A0  D1 00021               CMPL    56(R0), @DISPLAY_ID                          ;
                                      06  12 00026               BNEQ    2$                                           ;
                        11       44   A0  91 00028               CMPB    68(R0), #17                                  ;
                                      08  13 0002C               BEQL    3$                                           ;
                        50 00000000G  8F  D0 0002E 2$:           MOVL    #SMG$_INVDIS_ID, R0                          ;
                                          04 00035               RET                                                  ;
                        59       04   BC  D0 00036 3$:           MOVL    @DISPLAY_ID, DCB                             ;
                        50       08   BC  D0 0003A               MOVL    @PASTEBOARD_ID, R0                           ; 0647
                                      11  19 0003E               BLSS    4$                                           ;
             00000000G  00            50  D1 00040               CMPL    R0, PBD_L_COUNT                              ;
                                      08  14 00047               BGTR    4$                                           ;
          08 00000000G  00            50  E0 00049               BBS     R0, PBD_V_PB_AVAIL, 5$                       ;
                        50 00000000G  8F  D0 00051 4$:           MOVL    #SMG$_INVPAS_ID, R0                          ;
                                          04 00058               RET                                                  ;
                        54 00000000G0040  D0 00059 5$:           MOVL    PBD_A_PBCB[R0], PBCB                         ;
                        58       08   A4  D0 00061               MOVL    8(PBCB), WCB                                 ; 0648
                        03       10   A4  91 00065               CMPB    16(PBCB), #3                                 ; 0664
                                      2E  12 00069               BNEQ    8$                                           ;
                        01     00F4   C4  B1 0006B               CMPW    244(PBCB), #1                                ; 0667
                                      0B  12 00070               BNEQ    6$                                           ;
                        50       5F   A4  9A 00072               MOVZBL  95(PBCB), R0                                 ; 0668
                 00F6   C4            50  B1 00076               CMPW    R0, 246(PBCB)                                ;
                                      1C  13 0007B               BEQL    8$                                           ;
                        7E       5F   A4  9A 0007D 6$:           MOVZBL  95(PBCB), -(SP)                              ; 0676
                                      01  DD 00081               PUSHL   #1                                           ; 0673
                                      54  DD 00083               PUSHL   PBCB                                         ; 0674
             00000000G  00            03  FB 00085               CALLS   #3, SMG$$FORCE_SCROLL_REG                    ;
                        01            50  E8 0008C               BLBS    STATUS, 7$                                   ; 0673
                                          04 0008F               RET                                                  ;
                        05       0C   A4  E9 00090 7$:           BLBC    12(PBCB), 8$                                 ; 0684
                                      54  DD 00094               PUSHL   PBCB                                         ; 0686
                        6A            C1  FB 00096               CALLS   #1, SMG$$FLUSH_BUFFER                        ;
                                 04   AE  9F 00099 8$:           PUSHAB  PP                                           ; 0694
                                      54  DD 0009C               PUSHL   PBCB                                         ;
                                      59  DD 0009E               PUSHL   DCB                                          ;
             00000000G  00            03  FB 000A0               CALLS   #3, SMG$$LOCATE_PP                           ;
                        52            50  D0 000A7               MOVL    R0, STATUS                                   ;
```

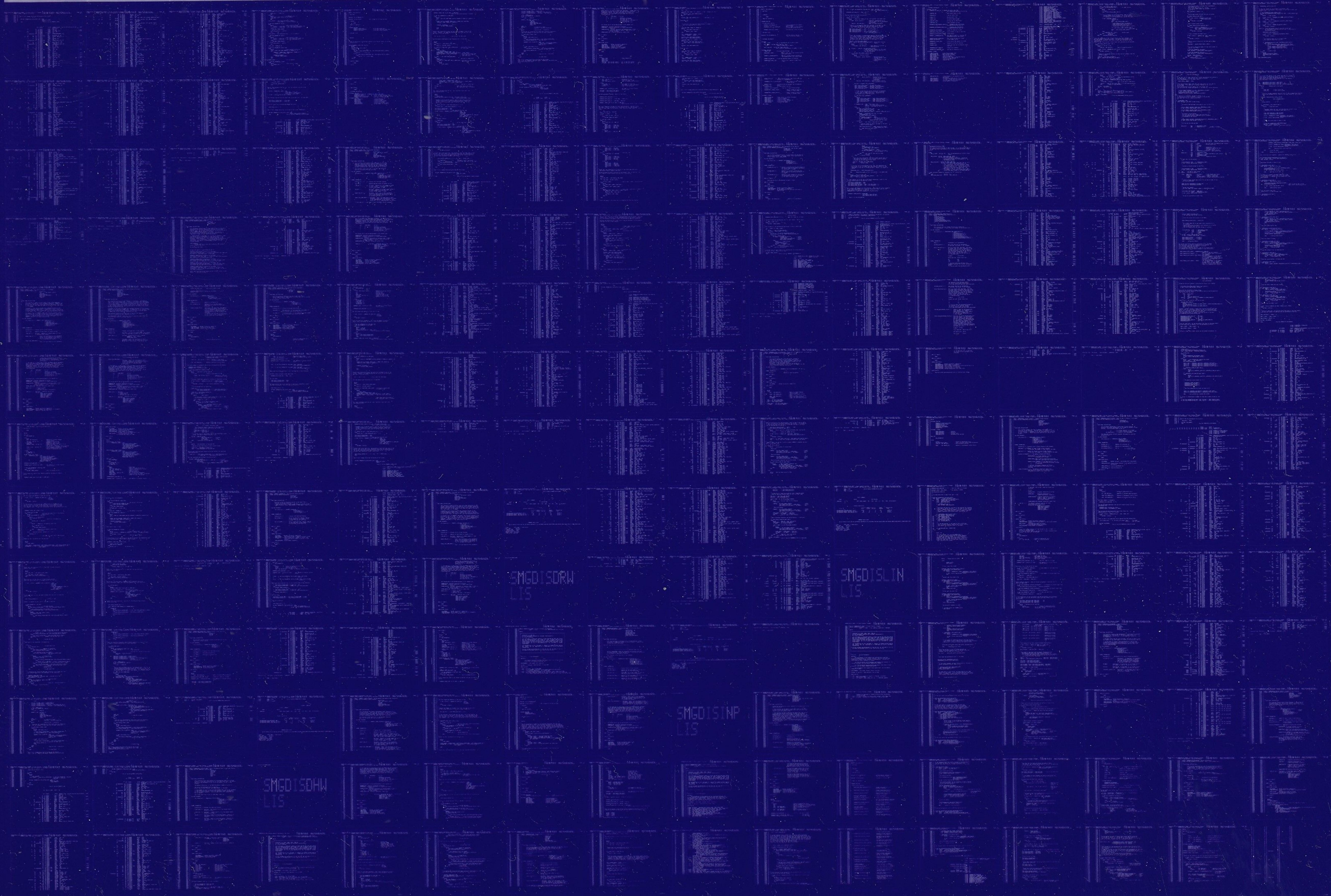
```
                             0D           52  E8 000AA        BLBS    STATUS, 10$                        ;
                             05      0C   A4  E9 000AD        BLBC    12(PBCB), 9$                       ; 0697
                                          54  DD 000B1        PUSHL   PBCB                               ; 0699
                             6A           01  FB 000B3        CALLS   #1, SMG$$FLUSH_BUFFER              ;
                             50           52  D0 000B6 9$:    MOVL    STATUS, R0                         ; 0700
                                              04 000B9        RET                                        ;
                             06           6C  91 000BA 10$:   CMPB    (AP), #6                           ; 0703
                                          0B  1F 000BD        BLSSU   11$                                ;
                                     18   AC  D5 000BF        TSTL    24(AP)                             ;
                                          06  13 000C2        BEQL    11$                                ;
                             56      18   BC  D0 000C4        MOVL    @DESIRED_ROW, ROW                  ;
                                          04  11 000C8        BRB     12$                                ;
                             56      28   A9  3C 000CA 11$:   MOVZWL  40(DCB), ROW                       ; 0704
                             07           6C  91 000CE 12$:   CMPB    (AP), #7                           ; 0706
                                          0B  1F 000D1        BLSSU   13$                                ;
                                     1C   AC  D5 000D3        TSTL    28(AP)                             ;
                                          06  13 000D6        BEQL    13$                                ;
                             57      1C   BC  D0 000D8        MOVL    @DESIRED_COL, COL                  ;
                                          04  11 000DC        BRB     14$                                ;
                             57      2A   A9  3C 000DE 13$:   MOVZWL  42(DCB), COL                       ; 0707
                                          56  D5 000E2 14$:   TSTL    ROW                                ; 0709
                                          08  15 000E4        BLEQ    15$                                ;
        56      02    A9     10           00  ED 000E6        CMPZV   #0, #16, 2(DCB), ROW               ;
                                          08  18 000EC        BGEQ    16$                                ;
                             50 00000000G 8F  D0 000EE 15$:   MOVL    #SMG$_INVROW, R0                   ;
                                              04 000F5        RET                                        ;
                                          57  D5 000F6 16$:   TSTL    COL                                ;
                                          08  15 000F8        BLEQ    17$                                ;
        57      06    A9     10           00  ED 000FA        CMPZV   #0, #16, 6(DCB), COL               ;
                                          08  18 00100        BGEQ    18$                                ;
                             50 00000000G 8F  D0 00102 17$:   MOVL    #SMG$_INVCOL, R0                   ;
                                              04 00109        RET                                        ;
                             55      04   AE  D0 0010A 18$:   MOVL    PP, R5                             ; 0714
                             03      2A   A5  E8 0010E        BLBS    42(R5), 19$                        ;
                                        00C7  31 00112        BRW     28$                                ;
                             50      18   A5  32 00115 19$:   CVTWL   24(R5), R0                         ; 0727
                             51      FF A046  9E 00119        MOVAB   -1(R0)[ROW], R1                    ;
                       18    AE           51  B0 0011E        MOVW    R1, TEMP                           ;
                       1A    AE           01  B0 00122        MOVW    #1, TEMP+2                         ; 0728
                             50      1A   A5  32 00126        CVTWL   26(R5), R0                         ; 0729
                             50           57  C0 0012A        ADDL2   COL, R0                            ;
                1C    AE     50           01  A3 0012D        SUBW3   #1, R0, TEMP+4                     ;
                             51      22   A5  3C 00132        MOVZWL  34(R5), R1                         ; 0731
                      50     51           50  C3 00136        SUBL3   R0, R1, R0                         ;
                1E    AE     50           02  A1 0013A        ADDW3   #2, R0, TEMP+6                     ;
                             53      0C   A5  D0 0013F        MOVL    12(R5), CURR_PP                    ; 0740
                             54           53  D1 00143 20$:   CMPL    CURR_PP, PBCB                      ; 0741
                                          03  12 00146        BNEQ    21$                                ;
                                        008A  31 00148        BRW     27$                                ;
                             52      F8   A3  9E 0014B 21$:   MOVAB   -8(R3), PP_BASE                    ; 0748
                                     22   A2  B5 0014F        TSTW    34(PP_BASE)                        ; 0753
                                          7A  13 00152        BEQL    26$                                ;
                             50      10   A2  D0 00154        MOVL    16(PP_BASE), NEW_DCB               ; 0769
                       08    AE      18   A2  B0 00158        MOVW    24(PP_BASE), TEMP1                 ; 0771
                       0A    AE      02   A0  B0 0015D        MOVW    2(NEW_DCB), TEMP1+2                ; 0772
                       0C    AE      1A   A2  B0 00162        MOVW    26(PP_BASE), TEMP1+4               ; 0773
                       0E    AE      06   A0  B0 00167        MOVW    6(NEW_DCB), TEMP1+6                ; 0774
```

```
                                    0E       2F   A0  E9 0016C        BLBC     47(NEW_DCB), 22$                    ; 0782
                                             08   AE  B7 00170        DECW     TEMP1                               ; 0785
                               0A   AE            02  A0 00173        ADDW2    #2, TEMP1+2                         ; 0786
                                             0C   AE  B7 00177        DECW     TEMP1+4                             ; 0787
                               0E   AE            02  A0 0017A        ADDW2    #2, TEMP1+6                         ; 0788
                                             10   AE  9F 0017E 22$:   PUSHAB   OVERLAP                             ; 0791
                                             0C   AE  9F 00181        PUSHAB   TEMP1                               ;
                                             20   AE  9F 00184        PUSHAB   TEMP                                ;
                       00000000G    00            03  FB 00187        CALLS    #3, SMG$$OCCLUDE                    ;
                                    3D            50  E9 0018E        BLBC     R0, 26$                             ;
                               14   AE       1C   AE  B1 00191        CMPW     TEMP+4, OVERLAP+4                   ; 0797
                                                  1E  1F 00196        BLSSU    23$                                 ;
                                    50       14   AE  3C 00198        MOVZWL   OVERLAP+4, R0                       ; 0799
                                    51       16   AE  3C 0019C        MOVZWL   OVERLAP+6, R1                       ;
                                    50            51  C0 001A0        ADDL2    R1, R0                              ;
                                                  50  D7 001A3        DECL     R0                                  ;
      50       1C   AE              10            00  ED 001A5        CMPZV    #0, #16, TEMP+4, R0                 ;
                                                  09  14 001AB        BGTR     23$                                 ;
                                             14   BC  D4 001AD        CLRL     @REMAINING_COLS                     ; 0802
                                    12       0C   A4  E8 001B0        BLBS     12(PBCB), 24$                       ; 0803
                                                  15  11 001B4        BRB      25$                                 ; 0806
               1E   AE         0C   AE       1C   AE  A3 001B6 23$:   SUBW3    TEMP+4, TEMP1+4, TEMP+6             ; 0816
                                                  0F  12 001BD        BNEQ     26$                                 ; 0817
                                             14   BC  D4 001BF        CLRL     @REMAINING_COLS                     ; 0820
                                    05       0C   A4  E9 001C2        BLBC     12(PBCB), 25$                       ; 0821
                                                  54  DD 001C6 24$:   PUSHL    PBCB                                ; 0823
                                    6A            01  FB 001C8        CALLS    #1, SMG$$FLUSH_BUFFER               ;
                                          0088    31 001CB 25$:       BRW      32$                                 ; 0824
                                    53       0C   A2  D0 001CE 26$:   MOVL     12(PP_BASE), CURR_PP                ; 0836
                                          FF6E    31 001D2            BRW      20$                                 ; 0741
                               14   BC       1E   AE  3C 001D5 27$:   MOVZWL   TEMP+6, @REMAINING_COLS             ; 0845
                                                  0C  11 001DA        BRB      29$                                 ; 0714
                                    50       22   A5  3C 001DC 28$:   MOVZWL   34(R5), R0                          ; 0851
                                    50            57  C2 001E0        SUBL2    COL, R0                             ;
                               14   BC       01   A0  9E 001E3        MOVAB    1(R0), @REMAINING_COLS              ;
                               28   A9            56  B0 001E8 29$:   MOVW     ROW, 40(DCB)                        ; 0858
                               2A   A9            57  B0 001EC        MOVW     COL, 42(DCB)                        ; 0859
                                    50       18   A5  32 001F0        CVTWL    24(R5), R0                          ; 0860
                                    51       FF A046  9E 001F4        MOVAB    -1(R0)[ROW], R1                     ;
                               20   A8            51  B0 001F9        MOVW     R1, 32(WCB)                         ;
                                    50       1A   A5  32 001FD        CVTWL    26(R5), R0                          ; 0861
                                    51       FF A047  9E 00201        MOVAB    -1(R0)[COL], R1                     ;
                               22   A8            51  B0 00206        MOVW     R1, 34(WCB)                         ;
                                    56     0108   C4  9E 0020A        MOVAB    264(PBCB), R6                       ; 0872
                                           00FC   C4  D5 0020F        TSTL     252(PBCB)                           ;
                                                  04  12 00213        BNEQ     30$                                 ;
                                                  66  D4 00215        CLRL     (R6)                                ;
                                                  32  11 00217        BRB      31$                                 ;
                               14   AE            02  D0 00219 30$:   MOVL     #2, INPUT_ARGS                      ;
                               18   AE       20   A8  32 0021D        CVTWL    32(WCB), INPUT_ARGS+4               ;
                               1C   AE       22   A8  32 00222        CVTWL    34(WCB), INPUT_ARGS+8               ;
                                             14   AE  9F 00227        PUSHAB   INPUT_ARGS                          ;
                                           0104   C4  DD 0022A        PUSHL    260(PBCB)                           ;
                                                  56  DD 0022E        PUSHL    R6                                  ;
                                           0100   C4  9F 00230        PUSHAB   256(PBCB)                           ;
                               10   AE     023A   8F  3C 00234        MOVZWL   #570, 16(SP)                        ;
                                             10   AE  9F 0023A        PUSHAB   16(SP)                              ;
```

```
                                  00FC    C4  9F 00230           PUSHAB  252(PBCB)                                  ;
                    00000000G  00         06  FB 00241           CALLS   #6, SMG$GET_TERM_DATA                      ;
                               0E         50  E9 00248           BLBC    STATUS, 33$
          0C    BC      0104   D4         66  28 0024B 31$:      MOVC3   (R6), @260(PBCB), @OUT_BUFFER              ; 0876
                          10   BC         66  D0 00252           MOVL    (R6), @OUT_BUFFER_LEN                      ; 0877
                               50         01  D0 00256 32$:      MOVL    #1, R0                                     ; 0879
                                              04 00259 33$:      RET                                                ; 0880

; Routine Size:  602 bytes,    Routine Base:  _SMG$CODE + 0115

;  808          0881  1 !<BLF/PAGE>
;  810          0882  1 END                           . End of module SMG$$DISPLAY_INPUT
;  811          0883  1
;  812          0884  0 ELUDOM



;                                          PSECT SUMMARY
;
;       Name                         Bytes                        Attributes
;
;   _SMG$CODE                             879  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         4         0          581         00:01.0
;   _$255$DUA28:[SMGRTL.OBJ]RTLLIB.L32;1           36         0         0            8         00:00.1
;   _$255$DUA28:[SMGRTL.OBJ]SMGLIB.L32;1          469        61        13           38         00:00.4



;                                          COMMAND QUALIFIERS

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

; Size:          879 code + 0 data bytes
; Run Time:         00:21.6
; Elapsed Time:     01:14.5
; Lines/CPU Min:     2453
; Lexemes/CPU-Min: 19076
; Memory Used:  231 pages
; Compilation Complete
```

0356 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



SMGDISDRW
LIS

SMGDISLIN
LIS

SMGDISINP
LIS

SMGDISDAW
LIS