C 12

```
   SSSSSSSSSSSS  MMM         MMM     GGGGGGGGGGGG  RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
   SSSSSSSSSSSS  MMM         MMM     GGGGGGGGGGGG  RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
   SSSSSSSSSSSS  MMM         MMM     GGGGGGGGGGGG  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
```

F 9

FILEID**SMGDISOUT

```
  SSSSSSSS  MM      MM  GGGGGGGG  DDDDDDDD      IIIIII     SSSSSSSS    OOOOOO    UU      UU  TTTTTTTTTT
  SSSSSSSS  MM      MM  GGGGGGGG  DDDDDDDD      IIIIII     SSSSSSSS    OOOOOO    UU      UU  TTTTTTTTTT
SS          MMMM  MMMM GG         DD      DD      II     SS          OO      OO  UU      UU      TT
SS          MMMM  MMMM GG         DD      DD      II     SS          OO      OO  UU      UU      TT
SS          MM  MM  MM GG         DD      DD      II     SS          OO      OO  UU      UU      TT
SS          MM  MM  MM GG         DD      DD      II     SS          OO      OO  UU      UU      TT
  SSSSSS    MM      MM GG         DD      DD      II       SSSSSS    OO      OO  UU      UU      TT
  SSSSSS    MM      MM GG         DD      DD      II       SSSSSS    OO      OO  UU      UU      TT
        SS  MM      MM GG  GGGGGG DD      DD      II             SS  OO      OO  UU      UU      TT
        SS  MM      MM GG  GGGGGG DD      DD      II             SS  OO      OO  UU      UU      TT
        SS  MM      MM GG      GG DD      DD      II             SS  OO      OO  UU      UU      TT       ....
        SS  MM      MM GG      GG DD      DD      II             SS  OO      OO  UU      UU      TT       ....
SSSSSSSS    MM      MM   GGGGGG   DDDDDDDD      IIIIII   SSSSSSSS      OOOOOO    UUUUUUUUUU      TT       ....
SSSSSSSS    MM      MM   GGGGGG   DDDDDDDD      IIIIII   SSSSSSSS      OOOOOO    UUUUUUUUUU      TT       ....

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$DISPLAY_OUTPUT - Output Virtual Displays'
0 MODULE SMG$DISPLAY_OUTPUT (
0               IDENT = '1-072' ! File: SMGDISOUT.B32 Edit: PLL1072
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 output the information contained
1 ! in virtual displays, using the mappings to windows defined via their
1 ! pastings to pasteboards.  Also include in this module are procedures
1 ! to optimize decisions about what needs to be output, and how to output
1 ! in the most optimal fashion.
1 !
1 !       The virtual displays and pasteboards are allocated/deallocated,
1 ! pasted/unpasted, etc. by the procedures in SMG$DISPLAY_LINKS.  The
1 ! contents of the virtual displays themselves are maintained by
1 ! procedures in SMG$DISPLAY_CHANGE.  The routines in module
1 ! SMG$DISPLAY_INPUT support input operations wiht respect to virtual
1 ! displays.
1 !
1 !
1 !
1 ! ENVIRONMENT:  User mode, Shared library routines.
1 !
1 ! AUTHOR: R. Reichert, CREATION DATE: 27-Jan-1983
1 !
1 ! MODIFIED BY:
1 !
1 ! 1-001 - Original.  Skeleton for future code.  RKR 27-Jan-1983
1 ! 1-002 - Further development.  RKR 11-Mar-1983
```

```
1 ! 1-003 - Correct names of data structures and macros.  PLL 15-Mar-1983
1 ! 1-004 - Add SMG$FLUSH_BUFFER, SMG$$FLUSH_BUFFER and SMG$CONTROL_MODE.
1 !         Delete SMG$ENABLE_BUFFER and SMG$DISABLE_BUFFER.
1 !         RKR 17-Mar-1983.
1 ! 1-005 - Minor tweaks to comments.  RKR 18-Mar-1983.
1 ! 1-006 - Change some names.  RKR 25-Mar-1983.
1 ! 1-007 - Make FILL_WINDOW_BUFFER able to handle pastings that don't
1 !         correspond exactly to the window buffer.  RKR 28-Mar-1983.
1 ! 1-008 - More debugging of above.  RKR 29-Mar-1983.
1 ! 1-009 - Add some code dealing with borders.  RKR 4-APR-1983.
1 ! 1-010 - Add code for labeled borders.  RKR 7-APR-1983.
1 ! 1-011 - Separate bits used for borders.  RKR 15-APR-1983.
1 ! 1-012 - Move minimal update into its own module along with buffering
1 !         stuff. RKR 15-APR-1983.
1 ! 1-013 - Update screen buffer when minimal update is enabled.
1 !         Use symbolic names for mode bits.
1 !         STAN 1-May-1983.
1 ! 1-014 - Clear WCB text and attribute buffers at beginning of
1 !         SMG$$FILL_WINDOW_BUFFER.  RKR 2-MAY-1983
1 ! 1-015 - Fix double-dot bug in SMG$CONTROL_MODE.
1 !         Remove call to SMG$$CONSTRUCT_BORDER_CHAR.
1 !         STAN 2-May-1983.
1 ! 1-016 - Flush output when batching ends.
1 !         New status returns for SMG$END_DISPLAY_UPDATE.
1 !         STAN 3-May-1983
1 ! 1-017   Complete overhaul of CHECK_FOR_OUTPUT_DCB, CHECK_FOR_OUTPUT_PBCB,
1 !         and FILL_WINDOW_BUFFER.
1 !         Remove some of the obsolete code resulting from prior edits.
1 !         RKR 4-MAY-1983.
1 ! 1-018   Clean up loose ends from last edit.  Remove
1 !         SMG$$CONSTRUCT_BORDER_CHAR.   RKR 4-MAY-1983
1 ! 1-019   Speed up redrawing of window buffer by making use of
1 !         PP_V_OCCLUDED bit.  RKR 6-MAY-1983.
1 ! 1-020   Admired edit 1-019.
1 !         Set physical (pasteboard) cursor position to the position
1 !         corresponding to the logical cursor position in the
1 !         last virtual display that the user referenced.
1 !         Thus physical cursor position has absolutely nothing
1 !         to do with the order the displays are pasted in.
1 !         STAN 8-May-1983
1 ! 1-021   Write SMG$$INVALIDATE_DISPLAY.
1 !         STAN 9-May-1983
1 ! 1-022   Add optimized move for contiguous source and destination in
1 !         SMG$$FILL_WINDOW_BUFFER and SMG$$CHECK_FOR_OUTPUT_DCB.
1 !         RKR 9-MAY-1983.
1 ! 1-023   Force REPAINT_SCREEN to repaint the screen even though it did
1 !         not think the screen changed.
1 !         STAN 10-May-1983
1 ! 1 -024  Fix Draw Border -- problem with border element overwriting
1 !         cell that has video attributes.
1 !         RKR 13-May-1983.
1 ! 1 -025  Take advantage of sizes available in DCB, WCB and PP.  No
1 !         longer need to multiply number of rows by number of cols.
1 !         RKR 13-MAY-1983.
1 ! 1 -026  Repackage inner-most subroutines to allow usage by input
1 !         support routines.
1 !         RKR 15-MAY-1983.
```

```
1 ! 1-027   Add SMG$BEGIN_PASTEBOARD_UPDATE and SMG$END_PASTEBOARD_UPDATE.
1 !         STAN 16-May-1983.
1 ! 1-028   Change logic of CHECK_FOR_OUTPUT_PBCB with respect to no
1 !         pasting packets.
1 !         RKR 18-MAY-1983.
1 ! 1-029   Fix CHECK_FOR_OUTPUT_PBCB.  Check for no pasting packets is
1 !         wrong.
1 !         RKR 18-MAY-1983.
1 ! 1-030  Delete references to external DD_ structures and counts --
1 !        no longer needed.
1 !        Partial fix up of SMG$RING_BELL -- add display_id argument.
1 !        RKR 20-MAY-1983.
1 ! 1-031  Add logic to BEGIN_DISPLAY_UPDATE and END_DISPLAY_UPDATE to
1 !        back up and restore DCBs.
1 !        RKR 23-MAY-1983.
1 ! 1-032  Fix RING_BELL.  TIMES by reference to STR$DUPL_CHAR.
1 !        RKR 23-MAY-1983.
1 ! 1-033  Add code to calc. what part of WCB buffers got changed and
1 !        leave this info in PBCB for output routines to use.
1 !        RKR 26-MAY-1983.
1 ! 1-034  Allow RING_BELL to return SS$_NORMAL if no error occurs.
1 !        STAN 27-May-1983
1 ! 1-035  Rename PBCB_B_COLS to PBCB_W_WIDTH.
1 !        STAN 1-Jun-1983.
1 ! 1-036  Add some bullet proofing to routine calls that don't check
1 !        status.  Make SMG$$POINT_IN_RECT into SMG$$POINT_IN_RECT_R3.
1 !        RKR 8-JUN-1983.
1 ! 1-037  Make DRAW_BORDER affect the record of what area of the
1 !        window control block buffer was modified.
1 !        RKR 9-JUN-1983.
1 ! 1-038  Make SMG$END_DISPLAY_UPDATE perform the pasting packet
1 !        constant recalculations if DCB_V_PP_MISMATCH indicates that
1 !        the SMG$DISPLAY_LINKS routine couldn't do it at the time the
1 !        change occurred because the display was batched.
1 !        RKR 17-JUN-1983.
1 ! 1-039 In CHECK_FOR_OUTPUT_DCB rearrange inner loop to check for
1 !       pasteboard batching and to use action code to decide whether
1 !       border needs to be (re)drawn.
1 !       In END_DISPLAY_UPDATE, pass action code to CHECK_FOR_OUTPUT_DCB.
1 !       RKR 20-JUN-1983.
1 ! 1-040 Fix error introduced in last edit.  Advancement to next PP must
1 !       must be inside outer loop, but not inside inner loop.
1 !       RKR 21-JUN-1983
1 ! 1-041 Speed up SMG$$MOVE_TEXT_TO_WINDOW_BUF by moving alternate
1 !       character set logic into its own incr loop instead of checking
1 !       for existance of alternate character set each time in main loop.
1 !       RKR 22-JUN-1983.
1 ! 1-042 Further speed up of SMG$$MOVE_TEXT_TO_BUF.
1 !       RKR 22-JUN-1983.
1 ! 1-043 Smarten up SMG$$END_DISPLAY_UPDATE to recognize situations where
1 !       it is necessary to redraw the affected pasteboard buffers from
1 !       the bottom out -- rather than just the display that had its
1 !       batching lifted.
1 !       RKR 24-JUN-1983.
1 ! 1-044 Remove LIB$PUT_SCREEN from the EXTERNAL declaration so that SCRSHR
1 !       will not be pulled into the SMGSHR shareable image.  PLL 1-Jul-1983
1 ! 1-045 Add logic to map the line characteristic vector during mapping
```

```
!	operations.
!	RKR 7-JUL-1983.
! 1-046 Add procedure SMG$FIND_CURSOR_DISPLAY.  Calcs. which virtual
!	display the current physical cursor is in (if any).
!	RKR 27-JUL-1983.
! 1-047 Rewrite SMG$$MOVE_TEXT_TO_WINDOW_BUF to accomodate double-wide
!	and double-wide/double-high.
!	Add optimization to SMG$$CHECK_FOR_OUTPUT_DCB to treat functions
!	that only change the cursor position as a special case.
!	RKR 31-Aug-1983.
! 1-048 STAN 31-Aug-1983. Round right rather tha left when pasting a
!	virtual display to an even boundary.
! 1-049 Fix border positioning for DWDH.
!	Fix places where "first_changed_row" and "last_changed_row"
!	incorrectly computed.  RKR 1-SEP-1983.
! 1-050 Further fixes in SMG$$MOVE_TEXT_TO_WINDOW_BUF wrt DWDH.
!	RKR 6-SEP-1983.
! 1-051 Fix SMG$$DRAW_BORDER to make border labels come out right
!	under DWDH situations.  RKR 7-SEP-1983.
! 1-052 Fix SMG$$DRAW_BORDER to pick up rendition bits for border
!	labels.  RKR 15-SEP-1983.
! 1-053 Allow CLEAR_SCREEN control mode. STAN 24-SEP-1983.
!	Check dummy arguments by name rather than by number.
! 1-054 Fix MOVE_TEXT_TO_WINDOW_BUFFER to remap border elements when
!	there is a transition from normal to DWDH or vice versa.
!	(STAN: Repaint clears line characteristics.)
!	RKR 11-OCT-1983.
! 1-055 Fix DRAW_BORDER for DHDW.  RKR 12-OCT-1983.
! 1-056 Replace critial loop by a SKPC in SMG$$MOVE_TEXT_TO_WINDOW_BUF.
!	RKR 9-NOV-1983.
! 1-057 Changes to SMG$$DRAW_BORDER and SMG$$MOVE_TEXT_TO_WINDOW_BUF
!	to not do special mapping for DW/DWDH if device doesn't
!	support this action.
!	RKR 10-NOV-1983.
! 1-058 Make SMG$$CHECK_FOR_OUTPUT_PBCB initialize WCB [WCB_A_CHAR_SET_BUF]
!	if one exists.  RKR 28-NOV-1983.
! 1-059 Make SMG$BEGIN_DISPLAY_UPDATE call SMG$$DUPL_VIRTUAL_DISPLAY
!	unconditionally.  SMG$$DUPL_VIRTUAL_DISPLAY now has the smarts to
!	decide whether a new DCB needs to be created or whether only current
!	context needs to be preserved in an already-existing backup DCB.
!	RKR 28-NOV-1983.
! 1-060 Introduce inner routine SMG$$BEGIN_PASTEBOARD_UPDATE_R1 and
!	SMG$$END_PASTEBOARD_UPDATE_R2.
!	RKR 2-DEC-1983.
! 1-061 Use new routine SMG$$UPDATE_PHYSICAL_CURSOR to fix physical cursor
!	position in cursor-positioning-only path through
!	SMG$$CHECK_FOR_OUTPUT_DCB.
!	RKR 15-DEC-1983.
! 1-062 Fix SMG$END_DISPLAY_UPDATE.  It was returning garbage if the batching
!	level was already 0.
!	RKR 16-DEC-1983.
! 1-063 Fix SIGNED/UNSIGNED problems in SMG$$POINT_IN_RECT_R3 and in
!	SMG$$OCCLUDE.  Change linkage to SMG$$POINT_IN_RECT_R3.
!	RKR 17-Jan-1984.
! 1-064 Fix SMG$$DRAW_BORDER for cases where virtual display is:
!	a). Wider than pasteboard and pasted at 0 or negative col  and
!	b). Higher than pasteboard and pasted at 0 or negative row
```

```
;   1 !         RKR 23-Jan-1984.
;   1 ! 1-065 Make SMG$$DRAW_BORDER abandon attempt to draw border altogether if none
;   1 !         of the virtual display maps onto the visible part of the pasteboard.
;   1 !         RKR 24-Feb-1984.
;   1 ! 1-066 Add routine SMG$GET_CHAR_AT_PHYSICAL_CURSOR.  RKR  28-FEB-1984.
;   1 ! 1-067 Move INVALIDATE_DISPLAY to file SMGPRVINP.B32. STAN 7-Mar-1984.
;   1 ! 1-068 Update header of CONTROL_MODE to mention the NOTABS bit and CLEAR_SCREEN.
;   1 !         STAN 14-Mar-1984.  Add public entry point SMG$INVALIDATE_DISPLAY.
;   1 ! 1-069 If only one line of a DCB changed, tell that to output, instead
;   1 !         of erroneously telling output that the whole display changed.
;   1 !         STAN 5-May-1984.
;   1 ! 1-070 Make horizontal borders knock down DHDW characteristic. STAN 17-Jun-1984.
;   1 ! 1-071 Fix to SMG$CONTROL_MODE to allow notabs bit.  PLL 28-Jun-1984
;   1 ! 1-072 Tighten up some consistency checks on the data structures.  It was
;   1 !         possible to get in an endless loop while walking the pasting packet
;   1 !         chain.  PLL 18-Jul-1984
;   1 !--
```

```
1 %SBTTL 'Declarations'
1 !
1 ! SWITCHES:
1 !
1
1 !
1 ! LINKAGES:
1 !
1 LINKAGE
1     POINT_IN_RECT_LINK = JSB ( REGISTER = 0, REGISTER = 1, REGISTER = 2) :
1                          NOPRESERVE ( 3)
1                          NOTUSED (4,5,6,7,8,9,10,11);
1 !
1 ! INCLUDE FILES
1 !
1
1 REQUIRE 'RTLIN:SMGPROLOG.REQ';                ! defines psects, macros, tcb,
1                                               !  wcb, & terminal symbols
1 !
1 ! TABLE OF CONTENTS:
1 !
1
1 FORWARD ROUTINE
1
1 ! Public entry points:
1
1     SMG$BEGIN_DISPLAY_UPDATE,                 ! Advance buffering level count for
1                                               ! a virtual display
1
1     SMG$BEGIN_PASTEBOARD_UPDATE,              ! Advance buffering level count for
1                                               ! a pasteboard
1
1     SMG$CONTROL_MODE,                         ! Control operational modes
1
1
1     SMG$END_DISPLAY_UPDATE,                   ! Decr. buffering level count
1                                               ! and flush to screen if it
1                                               ! reaches zero.
1
1     SMG$END_PASTEBOARD_UPDATE,                ! Decr. buffering level count
1                                               ! for a pasteboard, and flush to
1                                               ! screen if it reaches 0.
1
1     SMG$FIND_CURSOR_DISPLAY,                  ! Find the display which
1                                               ! contains the current physical
1                                               ! cursor position (if any).
1
1     SMG$GET_CHAR_AT_PHYSICAL_CURSOR,          ! Return char at physical cursor
1                                               ! location
1
1     SMG$INVALIDATE_DISPLAY,                   ! Mark contents of display as unknown.
1
1     SMG$REPAINT_SCREEN,                       ! Repaint the entire screen the
1                                               ! way our internal database says
1                                               ! it should look.
1
1     SMG$RING_BELL,                            ! Ring bell
```

```
! Private entry points:

    SMG$$BEGIN_PASTEBOARD_UPDATE_R1 : SMG$$BEGIN_PBD_UPDATE$LNK,
                                        ! Inner BEGIN_PASTEBOARD_UPDATE

    SMG$$CHECK_FOR_OUTPUT_DCB,          ! Check to see if a virtual
                                        ! display needs to be flushed to
                                        ! screen.

    SMG$$CHECK_FOR_OUTPUT_PBCB,         ! Refresh everything on this
                                        ! pasteboard -- triggered by
                                        ! an unpaste.

    SMG$$DRAW_BORDER,                   ! Move border characters into
                                        ! WCB text buffer.

    SMG$$END_PASTEBOARD_UPDATE_R2 : SMG$$END_PBD_UPDATE$LNK,
                                        ! Inner END_PASTEBOARD_UPDATE

    SMG$$FILL_WINDOW_BUFFER,            ! Fill the window buffer
                                        ! associated with a pasteboard
                                        ! with images of the virtual
                                        ! displays that map into it.

    SMG$$MOVE_TEXT_TO_WINDOW_BUF,       ! move text from DCB buffers to
                                        ! WCB window buffers.

    SMG$$OCCLUDE,                       ! Check whether two rectangular
                                        ! areas occlude each other.

    SMG$$POINT_IN_RECT_R3 : POINT_IN_RECT_LINK; ! Returns codes
                                                ! indicating where a
                                                ! given point is with
                                                ! respect to a given
                                                ! rectangular area.


!
! EXTERNAL REFERENCES
!

EXTERNAL
    PBD_L_COUNT,            ! No. of pasteboards we currently have

    PBD_A_PBCB : VECTOR [PBD_K_MAX_PB, LONG],
                            ! Table of addresses of PBCB's

    PBD_V_PB_AVAIL : BITVECTOR [PBD_K_MAX_PB];
                            ! Bit vector or pasteboard id numbers in use.

EXTERNAL LITERAL

    SMG$_BATSTIPRO,         ! Success; but batching is still in progress.
    SMG$_BATWASOFF,         ! Success; but batching was already off
    SMG$_BATWAS_ON,         ! Success; but batching was already on
    SMG$_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$_INVROW;              ! Invalid row number
;        1
;        1 EXTERNAL ROUTINE
;        1        LIB$ESTABLISH,                    ! Used to establish a handler
;        1        LIB$FREE_VM,                      ! Deallocate heap storage
;        1        LIB$GET_VM,                       ! Allocate heap storage
;        1        LIB$SFREE1_DD,                    ! Free a dynamic string
;        1        LIB$SIG_TO_RET,                   ! Handler to turn signals into return
;        1                                          ! statuses.
;        1        STR$DUPL_CHAR,                    ! Make a string of N copies of a byte.
;        1        SMG$$DUPL_VIRTUAL_DISPLAY,        ! Make a copy of a DCB with a new
;        1                                          ! display id.
;        1        SMG$$FLUSH_BUFFER,                ! Flush remaining buffered output
;        1        SMG$$INVALIDATE_DISPLAY,          ! Tell SMG that user has
;        1                                          ! written into this display on his own.
;        1        SMG$$MIN_UPD,                     ! Minimal update output routine
;        1        SMG$$OUTPUT,                      ! Output a string to a pasteboard device
;        1        SMG$$RECALC_PP_FIELDS,            ! Recalc. PP fields after batching level
;        1                                          ! lifted on a DCB
;        1        SMG$$UPDATE_PHYSICAL_CURSOR;      ! Update physical cursor position
;        1
;        1 !<BLF/PAGE>
```

```
%SBTTL 'SMG$BEGIN_DISPLAY_UPDATE - Begin batch of updates to display'
GLOBAL ROUTINE SMG$BEGIN_DISPLAY_UPDATE ( DISPLAY_ID ) =
!++
! FUNCTIONAL DESCRIPTION:
!
!       Disable outputing this virtual display to the screen until the
!       matching call to SMG$END_DISPLAY_UPDATE is encountered.
!
! CALLING SEQUENCE:
!
!       ret_status.wlc.v = SMG$BEGIN_DISPLAY_UPDATE ( DISPLAY_ID.rl.r)
!
! FORMAL PARAMETERS:
!
!       DISPLAY_ID.rl.r         Display id of virtual display.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! COMPLETION STATUS:
!
!       SS$_NORMAL      Normal successful completion, batching has been
!                       initiated
!       SMG$_BATWAS_ON  Success, but note that batching was already on
!       SMG$_INVDIS_ID  Invalid Display Id
!
! SIDE EFFECTS:
!
!       NONE
!--

    BEGIN
    LOCAL
        DCB : REF BLOCK [,BYTE];        ! Addr of display control block

!+
! Check for right number of arguments.
!-
    $SMG$VALIDATE_ARGCOUNT ( 1, 1);

    $SMG$GET_DCB (.DISPLAY_ID, DCB);    ! Get DCB address

!+
! Increment count of number of SMG$END_DISPLAY_UPDATE calls we need to
! see before we resume outputing from this virtual display.
! Let the user know what the previous state of batching was by
! our status return (in case he's interested).
!-
    DCB [DCB_L_BATCH_LEVEL] = .DCB [DCB_L_BATCH_LEVEL] + 1;

    IF .DCB [DCB_L_BATCH_LEVEL] EQL 1
    THEN
```

```
;  3          BEGIN   ! Begin of batching operation
;  3          LOCAL
;  3              STATUS,                          ! Status of subroutine calls
;  3              PP : REF BLOCK [,BYTE];          ! Addr of a pasting packet
;  3
;  3          !+
;  3          ! Make a copy of current DCB and leave its address in the
;  3          ! backup pointer field of the current DCB.
;  3          !-
;  4          IF NOT (STATUS = SMG$$DUPL_VIRTUAL_DISPLAY (
;  4                                    DCB,                         ! Addr of curr.
;  4                                    DCB [DCB_A_BACKUP_DCB]))     ! Where new
;  4                                                                 ! get stored.
;  3          THEN
;  3              RETURN (.STATUS);
;  3
;  3          !+
;  3          ! Walk chain of all pasteboards that we are pasted to and for
;  3          ! each encountered, in the affected pasting packet, set the
;  3          ! back pointer to the DCB to be the new DCB we've created.
;  3          ! This causes all mapping operations to reference the backed up
;  3          ! DCB as the source of images for the screen.  This is the
;  3          ! desired action since the current DCB is undergoing changes
;  3          ! which should not yet find their way to the screen -- its
;  3          ! batched.
;  3          !-
;  3          PP = .DCB [DCB_A_PP_NEXT];
;  3
;  3          IF .PP EQL 0
;  3          THEN
;  3              RETURN (SMG$_FATERRLIB);                ! should never be 0
;  3                                                      ! (points to self when empty)
;  3
;  3          WHILE .PP NEQ DCB [DCB_A_PP_NEXT]           ! While any remain
;  3          DO
;  4              BEGIN
;  4              PP [PP_A_DCB_ADDR] = .DCB [DCB_A_BACKUP_DCB];
;  4              PP = .PP [PP_A_NEXT_DCB];   ! Step to next packet
;  3              END;
;  3          RETURN  SS$_NORMAL;
;  3
;  3          END     ! Begin of batching operation
;  3
;  2      ELSE
;  2
;  2          RETURN  SMG$_BATWAS_ON
;  2
;  1      END;                    ! End of routine SMG$BEGIN_DISPLAY_UPDATE

                                          .TITLE  SMG$DISPLAY_OUTPUT SMG$DISPLAY_OUTPUT - Output
;                                                                    Virtual Displays
                                          .IDENT  \1-072\

                                          .EXTRN  PBD_L_COUNT, PBD_A_PBCB
                                          .EXTRN  PBD_V_PB_AVAIL, SMG$_BATSTIPRO
                                          .EXTRN  SMG$_BATWASOFF, SMG$_BATWAS_ON
```

```
                                                    .EXTRN  SMG$_FATERRLIB, SMG$_INVARG
                                                    .EXTRN  SMG$_INVCOL, SMG$_INVDIS_ID
                                                    .EXTRN  SMG$_INVPAS_ID, SMG$_INVROW
                                                    .EXTRN  LIB$ESTABLISH, LIB$FREE_VM
                                                    .EXTRN  LIB$GET_VM, LIB$SFREE1_DD
                                                    .EXTRN  LIB$SIG_TO_RET, STR$DUPL_CHAR
                                                    .EXTRN  SMG$$DUPL_VIRTUAL_DISPLAY
                                                    .EXTRN  SMG$$FLUSH_BUFFER
                                                    .EXTRN  SMG$$INVALIDATE_DISPLAY
                                                    .EXTRN  SMG$$MIN_UPD, SMG$$OUTPUT
                                                    .EXTRN  SMG$$RECALC_PP_FIELDS
                                                    .EXTRN  SMG$$UPDATE_PHYSICAL_CURSOR
                                                    .EXTRN  SMG$_WRONUMARG

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

                                    0004 00000      .ENTRY  SMG$BEGIN_DISPLAY_UPDATE, Save R2              ; 0464
            5E                   04   C2 00002      SUBL2   #4, SP                                         ;
            01                   6C   91 00005      CMPB    (AP), #1                                       ; 0506
                                 08   13 00008      BEQL    1$                                             ;
            50 00000000G         8F   D0 0000A      MOVL    #SMG$_WRONUMARG, R0                            ;
                                      04 00011      RET                                                    ;
            50          04       BC   D0 00012 1$:  MOVL    @DISPLAY_ID, R0                                ; 0508
      04    BC          38       A0   D1 00016      CMPL    56(R0), @DISPLAY_ID                            ;
                                 06   12 0001B      BNEQ    2$                                             ;
            11          44       A0   91 0001D      CMPB    68(R0), #17                                    ;
                                 08   13 00021      BEQL    3$                                             ;
            50 00000000G         8F   D0 00023 2$:  MOVL    #SMG$_INVDIS_ID, R0                            ;
                                      04 0002A      RET                                                    ;
            6E          04       BC   D0 0002B 3$:  MOVL    @DISPLAY_ID, DCB                               ;
            50                   6E   D0 0002F      MOVL    DCB, R0                                        ; 0516
                        1C       A0   D6 00032      INCL    28(R0)                                         ;
            01          1C       A0   D1 00035      CMPL    28(R0), #1                                     ; 0518
                                 38   12 00039      BNEQ    6$                                             ;
                        40       A0   9F 0003B      PUSHAB  64(R0)                                         ; 0531
                        04       AE   9F 0003E      PUSHAB  DCB                                            ; 0529
00000000G   00                   02   FB 00041      CALLS   #2, SMG$$DUPL_VIRTUAL_DISPLAY                  ; 0531
            2F                   50   E9 00048      BLBC    STATUS, 7$                                     ;
            51                   6E   D0 0004B      MOVL    DCB, R1                                        ; 0546
            50          20       A1   D0 0004E      MOVL    32(R1), PP                                     ;
                                 08   12 00052      BNEQ    4$                                             ; 0548
            50 00000000G         8F   D0 00054      MOVL    #SMG$_FATERRLIB, R0                            ; 0550
                                      04 0005B      RET                                                    ;
            52          20       A1   9E 0005C 4$:  MOVAB   32(R1), R2                                     ; 0553
            52                   50   D1 00060      CMPL    PP, R2                                         ;
                                 0A   13 00063      BEQL    5$                                             ;
      10    A0          40       A1   D0 00065      MOVL    64(R1), 16(PP)                                 ; 0556
            50                   60   D0 0006A      MOVL    (PP), PP                                       ; 0557
                                 ED   11 0006D      BRB     4$                                             ; 0553
            50                   01   D0 0006F 5$:  MOVL    #1, R0                                         ; 0565
                                      04 00072      RET                                                    ;
            50 00000000G         8F   D0 00073 6$:  MOVL    #SMG$_BATWAS_ON, R0                            ;
                                      04 0007A 7$:  RET                                                    ; 0567

; Routine Size:  123 bytes,    Routine Base:  _SMG$CODE + 0000
```

```
; 493         0568  1 !<BLF/PAGE>
```

```
;  495      0569  1  %SBTTL 'SMG$BEGIN_PASTEBOARD_UPDATE - Begin batch of updates to pasteboard'
;  496      0570  1  GLOBAL ROUTINE SMG$BEGIN_PASTEBOARD_UPDATE ( PBID ) =
;  497      0571  1  !++
;  498      0572  1  ! FUNCTIONAL DESCRIPTION:
;  499      0573  1  !
;  500      0574  1  !       Disable outputing this pasteboard to the screen until the
;  501      0575  1  !       matching call to SMG$END_PASTEBOARD_UPDATE is encountered.
;  502      0576  1  !
;  503      0577  1  ! CALLING SEQUENCE:
;  504      0578  1  !
;  505      0579  1  !       ret_status.wlc.v = SMG$BEGIN_PASTEBOARD_UPDATE ( PBID.rl.r)
;  506      0580  1  !
;  507      0581  1  ! FORMAL PARAMETERS:
;  508      0582  1  !
;  509      0583  1  !       PBID.rl.r                Pasteboard id.
;  510      0584  1  !
;  511      0585  1  ! IMPLICIT INPUTS:
;  512      0586  1  !
;  513      0587  1  !       NONE
;  514      0588  1  !
;  515      0589  1  ! IMPLICIT OUTPUTS:
;  516      0590  1  !
;  517      0591  1  !       NONE
;  518      0592  1  !
;  519      0593  1  ! COMPLETION STATUS:
;  520      0594  1  !
;  521      0595  1  !       SS$_NORMAL       Normal successful completion, batching has been
;  522      0596  1  !                        initiated
;  523      0597  1  !       SMG$_BATWAS_ON   Success, but note that batching was already on
;  524      0598  1  !       SMG$_INVPAS_ID   Invalid Pasteboard Id
;  525      0599  1  !
;  526      0600  1  ! SIDE EFFECTS:
;  527      0601  1  !
;  528      0602  1  !       NONE
;  529      0603  1  !--
;  530      0604  1
;  531      0605  2      BEGIN
;  532      0606  2      LOCAL
;  533      0607  2      PBCB : REF BLOCK [,BYTE];        ! Addr of pasteboard control block
;  534      0608  2
;  535      0609  2  !+
;  536      0610  2  ! Check for right number of arguments.
;  537      0611  2  !-
;  538      0612  2      $SMG$VALIDATE_ARGCOUNT ( 1, 1);
;  539      0613  2
;  540      0614  2      $SMG$GET_PBCB (.PBID, PBCB);        ! Get PBCB address
;  541      0615  2
;  542      0616  2      RETURN (SMG$$BEGIN_PASTEBOARD_UPDATE_R1 (.PBCB)); ! Do work in inner routine
;  543      0617  2
;  544      0618  1      END;                  ! End of routine SMG$BEGIN_PASTEBOARD_UPDATE


                              0000 00000       .ENTRY  SMG$BEGIN_PASTEBOARD_UPDATE, Save nothing    ; 0570
               01          6C  91 00002       CMPB    (AP), #1                                     ; 0612
```

```
                              08  13 00005        BEQL    1$                                     ;
             50 00000000G     8F  D0 00007        MOVL    #SMG$_WRONUMARG, R0                    ;
                                  04 0000E        RET                                            ;
             50          04   BC  D0 0000F 1$:    MOVL    @PBID, R0                              ; 0614
                              11  19 00013        BLSS    2$                                     ;
      00000000G  00           50  D1 00015        CMPL    R0, PBD_L_COUNT                        ;
                              08  14 0001C        BGTR    2$                                     ;
   08 00000000G  00           50  E0 0001E        BBS     R0, PBD_V_PB_AVAIL, 3$                 ;
             50 00000000G     8F  D0 00026 2$:    MOVL    #SMG$_INVPAS_ID, R0                    ;
                                  04 0002D        RET                                            ;
             50 00000000G0040     D0 0002E 3$:    MOVL    PBD_A_PBCB[R0], PBCB                   ;
                          0000V   30 00036        BSBW    SMG$$BEGIN_PASTEBOARD_UPDATE_R1        ; 0616
                                  04 00039        RET                                            ; 0618

; Routine Size:  58 bytes,    Routine Base:  _SMG$CODE + 007B


;  545          0619  1 !<BLF/PAGE>
```

```
1 %SBTTL 'SMG$END_DISPLAY_UPDATE - End batch of updates to display'
1 GLOBAL ROUTINE SMG$END_DISPLAY_UPDATE ( DISPLAY_ID ) =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Reduce the number of calls to SMG$END_DISPLAY_UPDATE that will
1 !       be needed before we resume outputing from this display.  If
1 !       this call makes this count go to zero, flush the display to
1 !       the screen.
1 !       If the level is already 0, we return a success status
1 !       informing caller that the batching level was already 0,
1 !       but we otherwise allow this.  This gives a user a guaranteed
1 !       method of ending batching.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       ret_status.wlc.v = SMG$END_DISPLAY_UPDATE ( DISPLAY_ID.rl.r)
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       DISPLAY_ID.rl.r         Display id of virtual display.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       DCB [DCB_L_BATCH_LEVEL]
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       DCB [DCB_L_BATCH_LEVEL] gets decremented
1 !
1 ! COMPLETION STATUS:
1 !
1 !       SS$_NORMAL      Normal successful completion
1 !       SMG$_BATSTIPRO  Success; but batching is still in progress.
1 !       SMG$_INVDIS_ID  Invalid Display Id
1 !       SMG$_BATWASOFF  Success; but batching was already off
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !--
1
2     BEGIN
2     LOCAL
2       STATUS,                         ! Status to return
2       DCB : REF BLOCK [,BYTE];        ! Addr of display control block
2
2 !+
2 ! Check for right number of arguments.
2 !-
2     $SMG$VALIDATE_ARGCOUNT ( 1, 1);
2
2     $SMG$GET_DCB (.DISPLAY_ID, DCB);    ! Get DCB address
2
2 !+
2 ! If current count is greater than 1, simply reduce it by one and
2 ! return to caller.  If it is one, reduce it to zero and cause the
```

```
2	! current contents of this window to be flushed to the screen (if need
2	! be).
2	!-
2	    IF .DCB [DCB_L_BATCH_LEVEL] GTR 1
2	    THEN
3		BEGIN
3		DCB [DCB_L_BATCH_LEVEL] = .DCB [DCB_L_BATCH_LEVEL] - 1;
3		STATUS = SMG$_BATSTIPRO;		! Ok, but batching is still in
3		END					! progress
3
2	    ELSE
2
3		BEGIN	! Level is currently 0 or 1
3		LOCAL
3		    BATCH_STATUS;
3
3		BATCH_STATUS = SS$_NORMAL;
3		IF .DCB [DCB_L_BATCH_LEVEL] EQL 0
3		THEN
4		    RETURN ( SMG$_BATWASOFF)	! Ok, but batching was already off
3		ELSE
3
4		    BEGIN		! Reduction from 1 to 0
4		    LOCAL
4			PP : REF $PP_DECL;	! Addr of a pasting packet
4
4		    DCB [DCB_L_BATCH_LEVEL] = 0;
4		    !+
4		    ! Walk chain of pastboards to which we are pasted.  For each
4		    ! pasting packet involved, set the pointer in the pasting
4		    ! packet back to the original DCB since it is now free of
4		    ! batching and its contents can be mapped to the screen.
4		    ! Note: We do not deallocate the backup DCB.  The assumption
4		    ! is that if the caller batched this virtual display once,
4		    ! he is likely to do it again.  This saves us having to
4		    ! deallocate now only to have to do another
4		    ! SMG$$DUPL_VIRTUAL_DISPLAY when he starts batching again.
4		    !-
4		    PP = .DCB [DCB_A_PP_NEXT];
4
4		    IF .PP EQL 0
4		    THEN
4			RETURN (SMG$_FATERRLIB);	! should never be 0
4
4		    WHILE .PP NEQ DCB [DCB_A_PP_NEXT]	! While any remain...
4		    DO
5			BEGIN
5			PP [PP_A_DCB_ADDR] = .DCB;	! To orig. DCB
5			PP = .PP [PP_A_NEXT_DCB];	! To next packet
4			END;
4
4		    !+
4		    ! If DCB_V_PP_MISMATCH is set, a change has been made to
4		    ! this DCB which requires the pasting packet constants to be
4		    ! recalculated.  However, the DCB was "batched" at the time
4		    ! the change took place, so we do the pasting packet update
4		    ! now as part of the unbatching process.
```

```
!-
IF .DCB [DCB_V_PP_MISMATCH]
THEN
    BEGIN    ! Unbatching clean up
    LOCAL
        CURR_PP : REF $PP_DECL; ! Addr of a pasting packet

    IF NOT (STATUS = SMG$$RECALC_PP_FIELDS (.DCB))
    THEN
        RETURN (.STATUS);

    DCB [DCB_V_PP_MISMATCH] = 0 ; ! Knock down flag

    !+
    ! Since DCB_V_PP_MISMATCH was set, the change to the DCB
    ! included dimensional changes.  This means we will have
    ! to remap the whole pasteboard buffer from the bottom
    ! outward -- for each pasteboard that we are pasted to.
    !-
    CURR_PP = .DCB [DCB_A_PP_NEXT];

    IF .CURR_PP EQL 0
    THEN
        RETURN (SMG$_FATERRLIB);      ! should never be 0

    WHILE .CURR_PP NEQ DCB [DCB_A_PP_NEXT]
    DO
        BEGIN        ! For all pasteboards
        LOCAL
            PBCB : REF $PBCB_DECL;  ! Addr of a pasteboard
                                    ! control block

        PBCB = .CURR_PP [PP_A_PBCB_ADDR];
        IF NOT (STATUS = SMG$$CHECK_FOR_OUTPUT_PBCB (.PBCB))
        THEN
            RETURN (.STATUS); ! Quit on first failure

        CURR_PP = .CURR_PP [PP_A_NEXT_DCB]; ! To next packet
        END;        ! For all pasteboards

    RETURN (SS$_NORMAL);
    END      ! Unbatching clean up

ELSE

    BEGIN    ! No cleanup needed
    !+
    ! Call SMG$$CHECK_FOR_OUTPUT_DCB to cause the contents
    ! of this virtual display to be flushed to the screen.
    ! If that fails, then return its status as our status.
    ! If that succeeded then return our previously
    ! calculated status.
    !-
    STATUS = SMG$$CHECK_FOR_OUTPUT_DCB (.DCB,
                                  SMG$C_END_DISPLAY_UPDATE);
    IF .STATUS THEN STATUS=.BATCH_STATUS
    END;     ! No cleanup needed
```

```
;  718          0791  3             END;            ! Reduction from 1 to 0
;  719          0792  2         END;        ! Level is currently 0 or 1
;  720          0793  2
;  721          0794  2     RETURN (.STATUS);
;  722          0795  1     END;                    ! End of routine SMG$END_DISPLAY_UPDATE


                                     001C 00000          .ENTRY  SMG$END_DISPLAY_UPDATE, Save R2,R3,R4            ; 0621
                   01             6C  91 00002          CMPB    (AP), #1                                         ; 0670
                                  08  13 00005          BEQL    1$                                               ;
                   50 00000000G   8F  D0 00007          MOVL    #SMG$_WRONUMARG, R0                              ;
                                      04 0000E          RET                                                      ;
                   50       04    BC  D0 0000F  1$:     MOVL    @DISPLAY_ID, R0                                  ; 0672
             04    BC       38    A0  D1 00013          CMPL    56(R0), @DISPLAY_ID                              ;
                                  06  12 00018          BNEQ    2$                                               ;
                   11       44    A0  91 0001A          CMPB    68(R0), #17                                      ;
                                  08  13 0001E          BEQL    3$                                               ;
                   50 00000000G   8F  D0 00020  2$:     MOVL    #SMG$_INVDIS_ID, R0                              ;
                                      04 00027          RET                                                      ;
                   52       04    BC  D0 00028  3$:     MOVL    @DISPLAY_ID, DCB                                 ;
                   01       1C    A2  D1 0002C          CMPL    28(DCB), #1                                      ; 0680
                                  0B  15 00030          BLEQ    4$                                               ;
                            1C    A2  D7 00032          DECL    28(DCB)                                          ; 0683
                   50 00000000G   8F  D0 00035          MOVL    #SMG$_BATSTIPRO, STATUS                          ; 0684
                                      04 0003C          RET                                                      ; 0680
                   54             01  D0 0003D  4$:     MOVL    #1, BATCH_STATUS                                 ; 0693
                            1C    A2  D5 00040          TSTL    28(DCB)                                          ; 0694
                                  08  12 00043          BNEQ    5$                                               ;
                   50 00000000G   8F  D0 00045          MOVL    #SMG$_BATWASOFF, R0                              ; 0696
                                      04 0004C          RET                                                      ;
                            1C    A2  D4 0004D  5$:     CLRL    28(DCB)                                          ; 0703
                   51       20    A2  D0 00050          MOVL    32(DCB), PP                                      ; 0715
                                  2D  13 00054          BEQL    8$                                               ; 0717
                   53       20    A2  9E 00056  6$:     MOVAB   32(DCB), R3                                      ; 0721
                   53             51  D1 0005A          CMPL    PP, R3                                           ;
                                  09  13 0005D          BEQL    7$                                               ;
             10    A1             52  D0 0005F          MOVL    DCB, 16(PP)                                      ; 0724
                   51             61  D0 00063          MOVL    (PP), PP                                         ; 0725
                                  EE  11 00066          BRB     6$                                               ; 0721
       3E    34    A2             03  E1 00068  7$:     BBC     #3, 52(DCB), 11$                                 ; 0735
                                  52  DD 0006D          PUSHL   DCB                                              ; 0741
          00000000G  00           01  FB 0006F          CALLS   #1, SMG$$RECALC_PP_FIELDS                        ;
                   41             50  E9 00076          BLBC    STATUS, 12$                                      ;
             34    A2             08  8A 00079          BICB2   #8, 52(DCB)                                      ; 0745
                   53       20    A2  D0 0007D          MOVL    32(DCB), CURR_PP                                 ; 0753
                                  08  12 00081          BNEQ    9$                                               ; 0755
                   50 00000000G   8F  D0 00083  8$:     MOVL    #SMG$_FATERRLIB, R0                              ; 0757
                                      04 0008A          RET                                                      ;
                   51       20    A2  9E 0008B  9$:     MOVAB   32(DCB), R1                                      ; 0759
                   51             53  D1 0008F          CMPL    CURR_PP, R1                                      ;
                                  13  13 00092          BEQL    10$                                              ;
                   51       14    A3  D0 00094          MOVL    20(CURR_PP), PBCB                                ; 0766
                                  51  DD 00098          PUSHL   PBCB                                             ; 0767
              0000V  CF           01  FB 0009A          CALLS   #1, SMG$$CHECK_FOR_OUTPUT_PBCB                   ;
```

```
                 18             50  E9 0009F        BLBC    STATUS, 12$                                ;
                 53             63  D0 000A2        MOVL    (CURR_PP), CURR_PP                         ; 0771
                                E4  11 000A5        BRB     9$                                         ; 0759
                 50             01  D0 000A7 10$:   MOVL    #1, R0                                     ; 0774
                                    04 000AA        RET                                                ;
                                1D  DD 000AB 11$:   PUSHL   #29                                        ; 0787
                                52  DD 000AD        PUSHL   DCB                                        ;
           0000V CF             02  FB 000AF        CALLS   #2, SMG$$CHECK_FOR_OUTPUT_DCB              ;
                 03             50  E9 000B4        BLBC    STATUS, 12$                                ; 0789
                 50             54  D0 000B7        MOVL    BATCH_STATUS, STATUS                       ;
                                    04 000BA 12$:   RET                                                ; 0795

; Routine Size:  187 bytes,    Routine Base:  _SMG$CODE + 00B5

;  723          0796  1 !<BLF/PAGE>
```

```
1 %SBTTL 'SMG$END_PASTEBOARD_UPDATE - End batch of updates to pasteboard'
1 GLOBAL ROUTINE SMG$END_PASTEBOARD_UPDATE ( PBID ) =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Reduce the number of calls to SMG$END_PASTEBOARD_UPDATE that will
1 !       be needed before we resume outputing from this pasteboard.  If
1 !       this call makes this count go to zero, flush the pasteboard to
1 !       the screen.
1 !       If the level is already 0, we return a success status
1 !       informing caller that the batching level was already 0,
1 !       but we otherwise allow this.  This gives a user a guaranteed
1 !       method of ending batching.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       ret_status.wlc.v = SMG$END_PASTEBOARD_UPDATE ( PBID.rl.r)
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       PBID.rl.r                Pasteboard id.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       PBCB [PBCB_L_BATCH_LEVEL]
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       PBCB [PBCB_L_BATCH_LEVEL] gets decremented
1 !
1 ! COMPLETION STATUS:
1 !
1 !       SS$_NORMAL       Normal successful completion
1 !       SMG$_BATSTIPRO   Success; but batching is still in progress.
1 !       SMG$_INVPAS_ID   Invalid Pasteboard Id
1 !       SMG$_BATWASOFF   Success; but batching was already off
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !--
1
2    BEGIN
2    LOCAL
2       STATUS,                          ! Status to return
2       PBCB : REF BLOCK [,BYTE];        ! Addr of pasteboard control block
2
2 !+
2 ! Check for right number of arguments.
2 !-
2    $SMG$VALIDATE_ARGCOUNT ( 1, 1);
2
2    $SMG$GET_PBCB (.PBID, PBCB);          ! Get PBCB address
2
2    RETURN (SMG$$END_PASTEBOARD_UPDATE_R2 (.PBCB)); ! Do work in inner routine
2
1    END;                  ! End of routine SMG$END_PASTEBOARD_UPDATE
```

```
                                  0004 00000          .ENTRY  SMG$END_PASTEBOARD_UPDATE, Save R2                    ; 0798
                  01           6C   91 00002          CMPB    (AP), #T                                              ; 0847
                               08   13 00005          BEQL    1$                                                    ;
                  50 00000000G 8F   D0 00007          MOVL    #SMG$_WRONUMARG, R0                                   ;
                                    04 0000E          RET                                                           ;
                  50        04 BC   D0 0000F 1$:      MOVL    @PBID, R0                                             ; 0849
                               11   19 00013          BLSS    2$                                                    ;
          00000000G 00         50   D1 00015          CMPL    R0, PBD_L_COUNT                                       ;
                               08   14 0001C          BGTR    2$                                                    ;
       08 00000000G 00         50   E0 0001E          BBS     R0, PBD_V_PB_AVAIL, 3$                                ;
                  50 00000000G 8F   D0 00026 2$:      MOVL    #SMG$_INVPAS_ID, R0                                   ;
                                    04 0002D          RET                                                           ;
                  50 00000000G0040  D0 0002E 3$:      MOVL    PBD_A_PBCB[R0], PBCB                                  ;
                             0000V  30 00036          BSBW    SMG$$END_PASTEBOARD_UPDATE_R2                         ; 0851
                                    04 00039          RET                                                           ; 0853

; Routine Size:  58 bytes,    Routine Base:  _SMG$CODE + 0170

;  782          0854  1 !<BLF/PAGE>
```

```
1 %SBTTL 'SMG$FIND_CURSOR_DISPLAY - Find which virtual display contains physical cursor'
1 GLOBAL ROUTINE SMG$FIND_CURSOR_DISPLAY (
1                                                           PASTEBOARD_ID,
1                                                           DISPLAY_ID
1                                                               ) =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine determines which virtual display contains the
1 !       current physical cursor on the screen.  The pasted virtual
1 !       displays are searched from the deepest pasted one to the
1 !       outer-most pasted one.  The outer-most virtual display that
1 !       encompasses the physical cursor is the one selected and its
1 !       display id is returned.
1 !
1 !       It is possible that no virtual display contains the cursor,
1 !       in which case a display id of zero (an invalid display id) is
1 !       returned.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       ret_status.wlc.v = SMG$FIND_CURSOR_DISPLAY (
1 !                                       PASTEBOARD_ID.rl.r,
1 !                                       DISPLAY_ID.wl.r)
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       PASTEBOARD_ID.rl.r      Address of a longword containing the
1 !                               pasteboard_id for the device for which
1 !                               the information is desired.
1 !
1 !       DISPLAY_ID.wl.r         Address of a longword to receive the
1 !                               virtual display_id of the virtual
1 !                               display which contains the cursor.
1 !                               The outer-most pasted virtual display
1 !                               that contains the cursor is selected.
1 !                               If no virtual display can be found,
1 !                               a display id of 0 ( an invalid display
1 !                               id) is supplied.
1 !
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! COMPLETION STATUS:
1 !
1 !       SS$_NORMAL      Normal successful completion
1 !       SMG$_INVPAS_ID  Invalid pasteboard id
1 !       SMG$_WRONUMARG  Wrong number of arguments
1 !
1 ! SIDE EFFECTS:
1 !
```

```
1 !      NONE
1 !--
2     BEGIN
2     LOCAL
2         PBCB : REF $PBCB_DECL,  ! Address of pasteboard control block
2
2         WCB  : REF $WCB_DECL,   ! Address of window control block
2
2         PP;                     ! An address pointing at the PP queue
2                                 ! header in a pasting packet.  This
2                                 ! address is not the base of the
2                                 ! pasting packet.
2 !+
2 ! Check for right number of arguments.
2 !-
2     $SMG$VALIDATE_ARGCOUNT ( 2, 2);
2
2 !+
2 ! Get address of pasteboard control block and the WCB that goes with it.
2 !-
2     $SMG$GET_PBCB ( .PASTEBOARD_ID, PBCB );
2     WCB = .PBCB [PBCB_A_WCB];
2
2 !+
2 ! Set up an answer of none in case search turns up nothing.
2 !-
2     .DISPLAY_ID = 0;
2
2 !+
2 ! Search all the pasting packets for this pasteboard, from the deepest-
2 ! pasted one to the outer-most pasted one.  For each such pasting
2 ! packet, see if the physical cursor position lies inside the
2 ! mapping of the associated virtual display.  For each virtual display
2 ! found, set up output parameter.  The last one found (if any) will be
2 ! the one the caller sees.
2 !-
2     PP = .PBCB [PBCB_A_PP_PREV]; ! Start with inner-most pasted one
2
2     IF .PP EQL 0
2     THEN
2         RETURN (SMG$_FATERRLIB);         ! should never be 0
2
2     WHILE .PP NEQ PBCB [PBCB_A_PP_NEXT]
2     DO
3         BEGIN   ! For all pasting packets
3         LOCAL
3             PP_BASE : REF $PP_DECL;      ! Base address of a pasting
3                                          ! packet
3
3         PP_BASE = .PP - PP_PBCB_QUEUE_OFFSET;
3
3         IF .PP_BASE [PP_W_ROWS_TO_MOVE] NEQ 0
3         THEN
4             BEGIN        ! Projects somewhere on visible pasteboard
4             LOCAL
4                 D_PROJ : BLOCK [8,BYTE]; ! Representation of virtual
4                                          ! display's projection on
```

```
;      4                                                   ! pasteboard.
;      4
;      4                  D_PROJ [DCB_W_ROW_START] = .PP_BASE [PP_W_FIRST_WCB_ROW];
;      4
;      4                  D_PROJ [DCB_W_NO_ROWS]   = .PP_BASE [PP_W_LAST_WCB_ROW] -
;      4                                             .PP_BASE [PP_W_FIRST_WCB_ROW] +1;
;      4
;      4                  D_PROJ [DCB_W_COL_START] = .PP_BASE [PP_W_FIRST_WCB_COL];
;      4
;      4                  D_PROJ [DCB_W_NO_COLS]   = .PP_BASE [PP_W_LAST_WCB_COL] -
;      4                                             .PP_BASE [PP_W_FIRST_WCB_COL] +1;
;      4
;      4                  !+
;      4                  ! Check to see if cursor lies within projection of this
;      4                  ! virtual display.
;      4                  !-
;      4                  IF SMG$$POINT_IN_RECT_R3 (WCB [WCB_W_OLD_CUR_COL],
;      4                                            WCB [WCB_W_OLD_CUR_ROW],
;      4                                            D_PROJ) EQL 0
;      4                  THEN
;      4                      .DISPLAY_ID = .PP_BASE [PP_A_DCB_ADDR]; !  Save candidate
;      4
;      3                  END;          ! Projects somewhere on visible pasteboard
;      3
;      3              PP = .PP_BASE [PP_A_PREV_PBCB] ; ! To next outer pasting
;      2              END;     ! For all pasting packets
;      2
;      2          RETURN (SS$_NORMAL);
;      1          END;                ! Routine SMG$FIND_CURSOR_DISPLAY


                                 00FC 00000        .ENTRY  SMG$FIND_CURSOR_DISPLAY, Save R2,R3,R4,R5,- ; 0856
                                                           R6,R7                                    ;
                   5E         08   C2 00002        SUBL2   #8, SP                                   ;
                   02         6C   91 00005        CMPB    (AP), #2                                 ; 0927
                              08   13 00008        BEQL    1$                                       ;
                   50 00000000G    8F   D0 0000A   MOVL    #SMG$_WRONUMARG, R0                      ;
                              04      00011        RET                                              ;
                   50      04 BC   D0 00012 1$:    MOVL    @PASTEBOARD_ID, R0                       ; 0932
                              11   19 00016        BLSS    2$                                       ;
          00000000G 00        50   D1 00018        CMPL    R0, PBD_L_COUNT                          ;
                              08   14 0001F        BGTR    2$                                       ;
       08 00000000G 00        50   E0 00021        BBS     R0, PBD_V_PB_AVAIL, 3$                   ;
                   50 00000000G 8F D0 00029 2$:    MOVL    #SMG$_INVPAS_ID, R0                      ;
                              04      00030        RET                                              ;
                   56 00000000G0040 D0 00031 3$:   MOVL    PBD_A_PBCB[R0], PBCB                     ;
                   57      08 A6   D0 00039        MOVL    8(PBCB), WCB                             ; 0933
                           08 BC   D4 0003D        CLRL    @DISPLAY_ID                              ; 0938
                   55      04 A6   D0 00040        MOVL    4(PBCB), PP                              ; 0948
                              08   12 00044        BNEQ    4$                                       ; 0950
                   50 00000000G    8F   D0 00046   MOVL    #SMG$_FATERRLIB, R0                      ; 0952
                              04      0004D        RET                                              ;
                   56         55   D1 0004E 4$:    CMPL    PP, PBCB                                 ; 0954
                              4F   13 00051        BEQL    6$                                       ;
```

```
                54     F8   A5   9E 00053          MOVAB   -8(R5), PP_BASE                ; 0961
                       1C   A4   B5 00057          TSTW    28(PP_BASE)                    ; 0963
                            40   13 0005A          BEQL    5$                             ;
                6E     2F   A4   B0 0005C          MOVW    47(PP_BASE), D_PROJ            ; 0971
                50     31   A4   3C 00060          MOVZWL  49(PP_BASE), R0                ; 0974
                51     2F   A4   3C 00064          MOVZWL  47(PP_BASE), R1                ;
                50          51   C2 00068          SUBL2   R1, R0                         ;
   02   AE      50          01   A1 0006B          ADDW3   #1, R0, D_PROJ+2               ;
        04      AE     33   A4   B0 00070          MOVW    51(PP_BASE), D_PROJ+4          ; 0976
                50     35   A4   3C 00075          MOVZWL  53(PP_BASE), R0                ; 0979
                51     33   A4   3C 00079          MOVZWL  51(PP_BASE), R1                ;
                50          51   C2 0007D          SUBL2   R1, R0                         ;
   06   AE      50          01   A1 00080          ADDW3   #1, R0, D_PROJ+6               ;
                52          6E   9E 00085          MOVAB   D_PROJ, R2                     ; 0986
                51     24   A7   9E 00088          MOVAB   36(WCB), R1                    ;
                50     26   A7   9E 0008C          MOVAB   38(WCB), R0                    ; 0985
                          0000V  30 00090          BSBW    SMG$$POINT_IN_RECT_R3          ; 0986
                            50   D5 00093          TSTL    R0                             ; 0987
                            05   12 00095          BNEQ    5$                             ;
        08      BC     10   A4   D0 00097          MOVL    16(PP_BASE), @DISPLAY_ID       ; 0989
                55     0C   A4   D0 0009C 5$:      MOVL    12(PP_BASE), PP                ; 0993
                            AC   11 000A0          BRB     4$                             ; 0954
                50          01   D0 000A2 6$:      MOVL    #1, R0                         ; 0996
                                 04 000A5          RET                                    ; 0997

; Routine Size:  166 bytes,    Routine Base:  _SMG$CODE + 01AA


;  927          0998  1 !<BLF/PAGE>
```

```
%SBTTL 'SMG$CONTROL_MODE - Control overall mode of operation'
GLOBAL ROUTINE SMG$CONTROL_MODE (
                                        PASTEBOARD_ID,
                                        NEW_MODE_BITS,
                                        OLD_MODE_BITS
                                    ) =
!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine allows the caller to interrogate various modes
!       of the overall operation of the SMG$ package, and to change
!       those modes.  Each pasteboard (hence device) has its own set
!       of mode settings.
!
!       The modes currently defined are:
!
!               1.  Use of buffers.  Under normal operation, the SMG$
!               package will output to the terminal as soon as it has
!               determined what needs to be written.  That is, no
!               internal buffering will be done.  This mode is the
!               default because it requires no follow-up effort on the
!               part of the user to periodically force the buffer to the
!               terminal.
!               However, greater efficiency of $QIO usage can be
!               achieved by enabling buffering.  In this mode, the SMG$
!               package buffers all of its output for efficient usage of
!               $QIO's.  The caller can at any time call
!               SMG$FLUSH_BUFFER to force to the screen any output which
!               has been queued to the screen, but not yet actually
!               output.  For example, in this mode, each input operation
!               initiated via the SMG$ facility will force a
!               SMG$FLUSH_BUFFER call.
!
!               Default setting is to not provide buffering.
!
!               2.  Minimal screen update.  Unless explicitly told not
!               to, the SMG$ package will try to minimize the number
!               of characters actually sent to a terminal to increase
!               overall system throughput.  It achieves this in part by
!               remembering what is currently on the screen and actually
!               outputing only the specific character sequences which
!               will change the current contents of the screen into the
!               desired contents of the screen.  As a result of this
!               action, the original sequence of changes may not be
!               preserved.  Generally, the screen is repainted so
!               rapidly the this difference is not noticable since the
!               resultant screen appears the way it should.  In rare
!               circumstances, perhaps while debugging a new
!               application, the caller may which to change this
!               behavior to strictly preserve the order of his output.
!
!               The default mode is to perform minimal screen update.
!               There is currently not much difference if you omit this
!               bit because it really doesn't make much sense to repaint
!               the entire screen when you change just one character,
!               but this may change in future releases.  For now,
!               you should always specify this bit.
```

```
!
!               3. CLEAR_SCREEN. If this bit is set, then when SMG exits
!               normally, it will clear the screen just before returning
!               control to DCL.
!
!               4. NOTABS.  If this bit is set, then SMG will never rely
!               on physical tabs even if the terminal supports them.
!               If not set, then SMG will feel free to use physical
!               tabs for the minimal update procedure.  Such use implicitly
!               assumes that your terminal's tab stops are set to the
!               DEC default locations.  Set this bit if you want to guarantee
!               that your product will run regardless of the tab settings
!               that the user has physically set on his terminal.
!
!       The three parameters are optional.  First, if OLD_MODE_BITS
!       is supplied, it is filled in with the current mode bit settings.
!       Secondly, if the NEW_MODE_BITS parameter is provided, its
!       contents are used to set the current mode(s) of operation.
!
!       Hence this routine can typically be used in three way.
!       a).     To find out current settings:
!               SMG$CONTROL_MODE ( PASTEBOARD_ID, , OLD_MODE_BITS)
!
!       b).     To set the bits with no regard for their current setting:
!               SMG$CONTROL_MODE ( PASTEBOARD_ID, NEW_MODE_BITS)
!
!       c).     To write modular code, saving the current settings,
!               setting them to your desired setting, then restoring
!               original settings before exiting your procedure:
!               SMG$CONTROL_MODE ( PASTEBOARD_ID,
!                                  NEW_MODE_BITS, saved_mode_bits )
!               and before exiting,
!               SMG$CONTROL_MODE ( PASTEBOARD_ID, saved_mode_bits )
!
!
! CALLING SEQUENCE:
!
!       ret_status.wlc.v = SMG$CONTROL_MODE ( PASTEBOARD_ID.rl.r,
!                                             [NEW_MODE_BITS.rl.r],
!                                             [OLD_MODE_BITS.wl.r] )
!
! FORMAL PARAMETERS:
!
!       PASTEBOARD_ID.rl.r      The id of the PASTEBOARD for which the
!                               modes are being set or interrogated.
!
!       NEW_MODE_BITS.rl.r      [Optional].  If supplied, new mode
!                               settings to be employed.
!                               A bit set to 1 forces that mode to be
!                               employed.  A bit set to 0 inhibits that
!                               mode of operation.
!                               Bit SMG$K_BUF_ENABLED controls buffering.
!                               This should normally be set to improve
!                               performance.
!                               Bit SMG$K_MINUPD controls minimal update and
!                               should normally be set.
!                               Bit SMG$K_CLEAR_SCREEN causes SMG to clear
```

```
1 !                                 the screen upon normal exit.
1 !                                 Bit SMG$K_NOTABS prevents SMG from relying
1 !                                 on physical tabs.
1 !                                 All remaining bits must be zero to allow
1 !                                 for expansion in the future.
1 !
1 !     OLD_MODE_BITS.wl.r          [Optional].  If supplied, will be filled
1 !                                 in with mode settings prior to this call.
1 !                                 A bit set to 1 indicates that the mode
1 !                                 was employed.  A bit set to 0 indicates
1 !                                 that the mode was inhibited.
1 !                                 Bit SMG$K_BUF_ENABLED controls buffering.
1 !                                 Bit SMG$K_MINUPD controls minimal update.
1 !                                 Bit SMG$K_CLEAR_SCREEN causes SMG to clear
1 !                                 the screen upon normal exit.
1 !                                 Bit SMG$K_NOTABS prevents use of physical tabs.
1 !                                 All remaining bits will be zero to allow
1 !                                 for expansion in the future.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !     None
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !     None
1 !
1 ! COMPLETION STATUS:
1 !
1 !     SS$_NORMAL        Normal successful completion
1 !     SMG$_INVARG       Invalid arguement.  Bits corresponding to
1 !                       unsupported functions supplied in NEW_MODE_BITS
1 !     SMG$_INVPAS_ID    Invalid pasteboard id.
1 !     SMG$_WRONUMARG    Wrong number of arguments.
1 !
1 ! SIDE EFFECTS:
1 !
1 !     NONE
1 !--
```

```
2        BEGIN
2        BUILTIN
2            NULLPARAMETER;
2
2        LOCAL
2            PBCB : REF BLOCK [,BYTE];          ! Address of associated
2                                               ! pasteboard control block.
2
2        $SMG$GET_PBCB ( .PASTEBOARD_ID, PBCB) ; ! Get address of PBCB
2
2
2    !+
2    ! If caller requested current settings, return them.
2    !-
2        IF NOT NULLPARAMETER (OLD_MODE_BITS)
2        THEN
2            .OLD_MODE_BITS = .PBCB [PBCB_L_MODE_SETTINGS];
2
2    !+
2    ! If caller provided new settings, process them.
2    !-
2        IF NOT NULLPARAMETER (NEW_MODE_BITS)
2        THEN
3            BEGIN   ! Caller provided new settings
3
3            BIND    NEW_MODE = .NEW_MODE_BITS        : BITVECTOR[32];
3
3            !+
3            ! First make sure that he didn't supply extraneous bits
3            !-
3            IF .(.NEW_MODE_BITS)<4,29> NEQ 0
3            THEN
3                RETURN (SMG$_INVARG);         ! Unsupported bits provided
3
3            !+
3            ! If he's switching from buffered to non-buffered, flush out
3            ! current contents of buffers so we are caught up.
3            !-
3            IF .PBCB [PBCB_V_BUF_ENABLED]  AND        ! If currently enabled
3                NOT .NEW_MODE[SMG$K_BUF_ENABLED]      ! and new disabled
3            THEN
3                SMG$$FLUSH_BUFFER ( .PBCB );
3
3            !+
3            ! If he's switching from no minimal update to minimal update,
3            ! then ensure that the screen buffer is up-to-date.
3            !-
3            IF NOT .PBCB [PBCB_V_MINUPD]  AND         ! If currently disabled
3                .NEW_MODE[SMG$K_MINUPD]               ! and new enabled
3            THEN
4                BEGIN        ! update screen buffer
4
4                BIND WCB = .PBCB [PBCB_A_WCB]         : BLOCK [,BYTE];
4
4                !+
4                ! Move the current text and attributes buffers
4                ! to the corresponding screen buffers.
```

```
; 1141          1210 4                 !-
; 1142          1211 4
; 1143          1212 4                 CH$MOVE(.WCB [WCB_L_BUFSIZE], .WCB [WCB_A_TEXT_BUF],
; 1144          1213 4                         .WCB [WCB_A_SCR_TEXT_BUF]);
; 1145          1214 4
; 1146          1215 4                 CH$MOVE(.WCB [WCB_L_BUFSIZE], .WCB [WCB_A_ATTR_BUF],
; 1147          1216 4                         .WCB [WCB_A_SCR_ATTR_BUF]);
; 1148          1217 4
; 1149          1218 4                 !+
; 1150          1219 4                 ! Move the text line characteristics vector to the screen
; 1151          1220 4                 ! text line characteristics vector.
; 1152          1221 4                 !-
; 1153          1222 4                 CH$MOVE(.WCB [WCB_W_NO_ROWS] +1, .WCB [WCB_A_LINE_CHAR],
; 1154          1223 4                         .WCB [WCB_A_SCR_LINE_CHAR]);
; 1155          1224 4
; 1156          1225 3                 END;          ! update screen buffer
; 1157          1226 3
; 1158          1227 3             !+
; 1159          1228 3             ! Finally, reset modes based on callers wishes.
; 1160          1229 3             !-
; 1161          1230 3             PBCB [PBCB_L_MODE_SETTINGS] = ..NEW_MODE_BITS;
; 1162          1231 3
; 1163          1232 2             END;     ! Caller provided new settings
; 1164          1233 2
; 1165          1234 2         RETURN (SS$_NORMAL);
; 1166          1235 2
; 1167          1236 1         END;                            ! Routine SMG$CONTROL_MODE



                                       00FC 00000          .ENTRY  SMG$CONTROL_MODE, Save R2,R3,R4,R5,R6,R7         ; 1000
                      50          04  BC  D0 00002          MOVL    @PASTEBOARD_ID, R0                               ; 1161
                                      11  19 00006          BLSS    1$                                               ;
                00000000G  00             50  D1 00008          CMPL    R0, PBD_L_COUNT                                  ;
                                      08  14 0000F          BGTR    1$                                               ;
             08 00000000G  00             50  E0 00011          BBS     R0, PBD_V_PB_AVAIL, 2$                           ;
                      50 00000000G       8F  D0 00019 1$:      MOVL    #SMG$_INVPAS_ID, R0                              ;
                                          04 00020          RET                                                      ;
                      57 00000000G0040        D0 00021 2$:      MOVL    PBD_A_PBCB[R0], PBCB                             ;
                      03                  6C  91 00029          CMPB    (AP), #3                                         ; 1167
                                          0A  1F 0002C          BLSSU   3$                                               ;
                                  0C      AC  D5 0002E          TSTL    12(AP)                                           ;
                                          05  13 00031          BEQL    3$                                               ;
                  0C  BC          0C      A7  D0 00033          MOVL    12(PBCB), @OLD_MODE_BITS                         ; 1169
                      02                  6C  91 00038 3$:      CMPB    (AP), #2                                         ; 1174
                                          53  1F 0003B          BLSSU   7$                                               ;
                                  08      AC  D5 0003D          TSTL    8(AP)                                            ;
                                          4E  13 00040          BEQL    7$                                               ;
      50      08  BC          1D          04  EF 00042          EXTZV   #4, #29, @NEW_MODE_BITS, R0                      ; 1183
                                          08  13 00048          BEQL    4$                                               ;
                      50 00000000G       8F  D0 0004A          MOVL    #SMG$_INVARG, R0                                 ; 1185
                                          04 00051          RET                                                      ;
                      0D          0C      A7  E9 00052 4$:      BLBC    12(PBCB), 5$                                     ; 1191
                      09          08      BC  E8 00056          BLBS    @NEW_MODE_BITS, 5$                               ; 1192
                                          57  DD 0005A          PUSHL   PBCB                                             ; 1194
```

```
                     0000000G   00          01  FB 0005C       CALLS   #1, SMG$$FLUSH_BUFFER                      ;
              23           0C   A7          01  E0 00063 5$:   BBS     #1, 12(PBCB), 6$                           ; 1200
              1E           08   BC          01  E1 00068       BBC     #1, @NEW_MODE_BITS, 6$                     ; 1201
                                56      08  A7  D0 0006D       MOVL    8(PBCB), R6                                ; 1205
      14      B6           08   B6      28  A6  28 00071       MOVC3   40(R6), @8(R6), @20(R6)                    ; 1213
      18      B6           0C   B6      28  A6  28 00078       MOVC3   40(R6), @12(R6), @24(R6)                   ; 1216
                                50      02  A6  3C 0007F       MOVZWL  2(R6), R0                                  ; 1222
                                            50  D6 00083       INCL    R0                                         ;
      30      B6           2C   B6          50  28 00085       MOVC3   R0, @44(R6), @48(R6)                       ; 1223
                           0C   A7      08  BC  D0 0008B 6$:   MOVL    @NEW_MODE_BITS, 12(PBCB)                   ; 1230
                                50          01  D0 00090 7$:   MOVL    #1, R0                                     ; 1234
                                            04 00093           RET                                                ; 1236

; Routine Size:  148 bytes,    Routine Base:  _SMG$CODE + 0250
```

```
%SBTTL 'SMG$INVALIDATE_DISPLAY - Mark display as being privately used'
GLOBAL ROUTINE SMG$INVALIDATE_DISPLAY ( DISPLAY_ID) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine is called when ever a change has been completed
!       to a given virtual display and the user had previously
!       written into that display on his own, without using
!       SMG routines.
!
!       The virtual display must not be occluded.
!
!       Each pasteboard to which this display is pasted is isolated and
!       its window image must be redrawn.
!
! CALLING SEQUENCE:
!
!       ret_status.wlc.v = SMG$INVALIDATE_DISPLAY ( DISPLAY_ID.rl.r)
!
! FORMAL PARAMETERS:
!
!       DISPLAY_ID.rl.r         Display ID of virtual display.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! COMPLETION STATUS:
!
!       SS$_NORMAL      Normal successful completion
!
! SIDE EFFECTS:
!
!       NONE
!--
```

```
; 1210          1277  2 BEGIN
; 1211          1278  2
; 1212          1279  2 EXTERNAL ROUTINE
; 1213          1280  2
; 1214          1281  2         SMG$$INVALIDATE_DISPLAY;
; 1215          1282  2
; 1216          1283  2 LOCAL
; 1217          1284  2
; 1218          1285  2         STATUS; ! Status of subroutine calls
; 1219          1286  2
; 1220          1287  2 !+
; 1221          1288  2 ! Check for right number of arguments.
; 1222          1289  2 !-
; 1223          1290  2
; 1224          1291  2 $SMG$VALIDATE_ARGCOUNT ( 1, 1);
; 1225          1292  2
; 1226          1293  2 !+
; 1227          1294  2 ! Call the internal routine.
; 1228          1295  2 !-
; 1229          1296  2
; 1230          1297  2 RETURN  SMG$$INVALIDATE_DISPLAY(.DISPLAY_ID)
; 1231          1298  2
; 1232          1299  1 END;           ! End of routine SMG$INVALIDATE_DISPLAY


                                   0000 00000          .ENTRY  SMG$INVALIDATE_DISPLAY, Save nothing      ; 1238
                   01          6C  91 00002            CMPB    (AP), #1                                  ; 1291
                               08  13 00005            BEQL    1$                                        ;
                50 00000000G   8F  D0 00007            MOVL    #SMG$_WRONUMARG, R0                       ;
                                   04 0000E            RET                                               ;
                         04    AC  DD 0000F  1$:       PUSHL   DISPLAY_ID                                ; 1297
          00000000G  00        01  FB 00012            CALLS   #1, SMG$$INVALIDATE_DISPLAY               ;
                                   04 00019            RET                                               ; 1299

; Routine Size:  26 bytes,    Routine Base:  _SMG$CODE + 02E4
```

```
%SBTTL 'SMG$GET_CHAR_AT_PHYSICAL_CURSOR - Get character at physical cursor'
GLOBAL ROUTINE SMG$GET_CHAR_AT_PHYSICAL_CURSOR (
                                        PASTEBOARD_ID,
                                        CHARACTER : REF VECTOR [,BYTE]
                                        ) =

!++
! FUNCTIONAL DESCRIPTION
!
! Returns the character that occupies the position on the screen corresponding
! to the current physical cursor position.
!
! Note:  If the SMG$ facility has not written to that location on the screen,
! then it doesn't know its contents.  A character encoded as X'FF' will be
! returned in that case.
!
! If character returned has a value less than X'20', then it is not an actual
! printable character, but an internal terminal-independent code of what should
! be displayed at that position, e.g. an element of the line-drawing character
! set.  Do not attempt to use this code for subsequent output operations.
!
!
! CALLING SEQUENCE
!
!       status.wlc.v = SMG$GET_CHAR_AT_PHYSICAL_CURSOR (
!                               PASTEBOARD_ID.rl.r,
!                               CHARACTER.wbu.r)
!
! FORMAL PARAMETERS
!
!       PASTEBOARD_ID.rl.r      Address of a longword containing the
!                               pasteboard_id for which the information is
!                               desired.
!
!       CHARACTER.wbu.r         Address of a byte to receive the result.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLCIT OUTPUTS:
!
!       NONE
!
! COMPLETION STATUS:
!
!       SS$NORMAL       Normal successful completion
!       SMG$_INVPAS_ID  Invalid pasteboard id
!       SMG$_WRONUMARG  Wrong number of arguments
!
! SIDE EFFECTS:
!
!       NONE
!--
    BEGIN
    LOCAL
        PBCB : REF $PBCB_DECL,          ! Address of pasteboard control block
```

```
; 1291          1357  2          WCB  : REF $WCB_DECL,              ! Address of window control block
; 1292          1358  2          PTR  : REF VECTOR [,BYTE];         ! Pointer to text buffer of what is
; 1293          1359  2                                             ! currently on the screen.
; 1294          1360  2
; 1295          1361  2  !+
; 1296          1362  2  ! Check for right number of arguments.
; 1297          1363  2  !-
; 1298          1364  2      $SMG$VALIDATE_ARGCOUNT (2, 2);
; 1299          1365  2
; 1300          1366  2  !+
; 1301          1367  2  ! Get address of pasteboard control block and of window control block.
; 1302          1368  2  !-
; 1303          1369  2      $SMG$GET_PBCB (.PASTEBOARD_ID, PBCB);
; 1304          1370  2      WCB = .PBCB [PBCB_A_WCB];
; 1305          1371  2      PTR = .WCB [WCB_A_SCR_TEXT_BUF];
; 1306          1372  2
; 1307          1373  2  !+
; 1308          1374  2  ! Fetch character at position corresponding to current actual physical cursor
; 1309          1375  2  ! position on the screen.
; 1310          1376  2  !-
; 1311          1377  2      CHARACTER [0] = .PTR [
; 1312          1378  2                      (.WCB [WCB_W_OLD_CUR_ROW] -1) * .WCB [WCB_W_NO_COLS] +
; 1313          1379  2                      (.WCB [WCB_W_OLD_CUR_COL] -1) ];
; 1314          1380  2
; 1315          1381  2      RETURN (SS$_NORMAL);
; 1316          1382  1      END;            ! Routine SMG$GET_CHAR_AT_PHYSICAL_CURSOR


                                   000C 00000         .ENTRY  SMG$GET_CHAR_AT_PHYSICAL_CURSOR, Save R2,R3  ; 1301
                   02          6C    91 00002         CMPB    (AP), #2                                     ; 1364
                               08    13 00005         BEQL    1$                                           ;
                   50 00000000G 8F   D0 00007         MOVL    #SMG$_WRONUMARG, R0                          ;
                               04       0000E         RET                                                  ;
                   50       04  BC   D0 0000F  1$:    MOVL    @PASTEBOARD_ID, R0                           ; 1369
                               11    19 00013         BLSS    2$                                           ;
           00000000G 00        50    D1 00015         CMPL    R0, PBD_L_COUNT                              ;
                               08    14 0001C         BGTR    2$                                           ;
        08 00000000G 00        50    E0 0001E         BBS     R0, PBD_V_PB_AVAIL, 3$                       ;
                   50 00000000G 8F   D0 00026  2$:    MOVL    #SMG$_INVPAS_ID, R0                          ;
                               04       0002D         RET                                                  ;
                   50 00000000G0040  D0 0002E  3$:    MOVL    PBD_A_PBCB[R0], PBCB                         ;
                   50       08  A0   D0 00036         MOVL    8(PBCB), WCB                                 ; 1370
                   52       14  A0   D0 0003A         MOVL    20(WCB), PTR                                 ; 1371
                   51       24  A0   32 0003E         CVTWL   36(WCB), R1                                  ; 1378
                               51    D7 00042         DECL    R1                                           ;
                   53       06  A0   3C 00044         MOVZWL  6(WCB), R3                                   ;
                   51          53    C4 00048         MULL2   R3, R1                                       ;
                   50       26  A0   32 0004B         CVTWL   38(WCB), R0                                  ; 1379
                   51          50    C0 0004F         ADDL2   R0, R1                                       ;
              08   BC       FF A142  90 00052         MOVB    -1(R1)[PTR], @CHARACTER                      ; 1377
                   50          01    D0 00058         MOVL    #1, R0                                       ; 1381
                               04       0005B         RET                                                  ; 1382

; Routine Size:  92 bytes,    Routine Base:  _SMG$CODE + 02FE
```

```
%SBTTL 'SMG$REPAINT_SCREEN - Repaint current screen'
GLOBAL ROUTINE SMG$REPAINT_SCREEN (
                                PASTEBOARD_ID
                                )=
!++
! FUNCTIONAL DESCRIPTION:
!
!       This procedure will repaint the current screen based on its
!       internal knowledge of what the screen should look like.
!       It is intended to be called when it is known or suspected that
!       some outside agent has disrupted the screen so that it no longer
!       matches the internal knowledge of what should be on the screen.
!       If pasteboard batching is in effect, the repaint occurs
!       from the screen image; otherwise it occurs from the text image.
!
! CALLING SEQUENCE:
!
!       ret_status.wlc.v = SMG$REPAINT_SCREEN ( PASTEBOARD_ID.rl.r )
!
! FORMAL PARAMETERS:
!
!       PASTEBOARD_ID.rl.r        Id of pasteboard associated with
!                                 physical screen to be repainted.
!
! IMPLICIT INPUTS:
!
!       Current internal representation of what should be on the screen.
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! COMPLETION STATUS:
!
!       SS$_NORMAL       Normal successful completion
!       SMG$_INVPAS_ID  Invalid pasteboard control block
!
! SIDE EFFECTS:
!
!       NONE
!--
    BEGIN
    LOCAL
        WCB     : REF BLOCK [,BYTE],     ! Window control block
        PBCB    : REF BLOCK [,BYTE];     ! Address of pasteboard control
                                         ! block

    $SMG$GET_PBCB ( .PASTEBOARD_ID, PBCB); ! Get address of pasteboard
                                           ! control block.

    !+
    ! Get address of window control block.
    !-

    WCB = .PBCB [PBCB_A_WCB];

    !+
```

```
; 1375      1440  2       ! Zero out the screen image, thereby forcing output to
; 1376      1441  2       ! redraw the whole screen.
; 1377      1442  2       !-
; 1378      1443  2
; 1379      1444  2       ! **************** We have to add code here to make
; 1380      1445  2       ! the repaint occur from the window buffer if batching is in effect.
; 1381      1446  2       ! ******************************
; 1382      1447  2
; 1383      1448  2       CH$FILL(0,.WCB [WCB_L_BUFSIZE],.WCB [WCB_A_SCR_TEXT_BUF]);
; 1384      1449  2
; 1385      1450  2       !+
; 1386      1451  2       ! Clear the line characteristics vector.
; 1387      1452  2       !-
; 1388      1453  2
; 1389      1454  2       CH$FILL(0,.WCB [WCB_W_NO_COLS]+1,.WCB [WCB_A_SCR_LINE_CHAR]);
; 1390      1455  2
; 1391      1456  2   !+
; 1392      1457  2   ! Tell the output routines that we've changed the whole WCB buffer.
; 1393      1458  2   !-
; 1394      1459  2       PBCB [PBCB_W_FIRST_CHANGED_ROW] = 1;
; 1395      1460  2       PBCB [PBCB_W_LAST_CHANGED_ROW]  = .PBCB [PBCB_B_ROWS];
; 1396      1461  2       PBCB [PBCB_W_FIRST_CHANGED_COL] = 1;
; 1397      1462  2       PBCB [PBCB_W_LAST_CHANGED_COL]  = .PBCB [PBCB_W_WIDTH];
; 1398      1463  2
; 1399      1464  2       RETURN (SMG$$MIN_UPD (.PBCB));
; 1400      1465  1       END;                    ! End of procedure SMG$REPAINT_SCREEN


                                          00FC 00000       .ENTRY  SMG$REPAINT_SCREEN, Save R2,R3,R4,R5,R6,R7   ; 1384
                       50        04   BC   D0 00002        MOVL    @PASTEBOARD_ID, R0                           ; 1430
                                      11   19 00006        BLSS    1$                                           ;
               00000000G 00           50   D1 00008        CMPL    R0, PBD_L_COUNT                              ;
                                      08   14 0000F        BGTR    1$                                           ;
            08 00000000G 00           50   E0 00011        BBS     R0, PBD_V_PB_AVAIL, 2$                       ;
                       50 00000000G   8F   D0 00019 1$:    MOVL    #SMG$_INVPAS_ID, R0                          ;
                                           04 00020        RET                                                  ;
                       57 00000000G0040    D0 00021 2$:    MOVL    PBD_A_PBCB[R0], PBCB                         ;
                       56        08   A7   D0 00029        MOVL    8(PBCB), WCB                                 ; 1437
    28  A6          00           6E   00   2C 0002D        MOVC5   #0, (SP), #0, 40(WCB), @20(WCB)              ; 1448
                                 14   B6      00033                                                         ;
                       50        06   A6   3C 00035        MOVZWL  6(WCB), R0                                   ; 1454
                                      50   D6 00039        INCL    R0                                           ;
        50          00           6E   00   2C 0003B        MOVC5   #0, (SP), #0, R0, @48(WCB)                   ;
                                 30   B6      00040                                                         ;
                  00A8  C7            01   B0 00042        MOVW    #1, 168(PBCB)                                ; 1459
                  00AA  C7       5F   A7   9B 00047        MOVZBW  95(PBCB), 170(PBCB)                          ; 1460
                  00AC  C7            01   B0 0004D        MOVW    #1, 172(PBCB)                                ; 1461
                  00AE  C7       5A   A7   B0 00052        MOVW    90(PBCB), 174(PBCB)                          ; 1462
                                      57   DD 00058        PUSHL   PBCB                                         ; 1464
               00000000G 00           01   FB 0005A        CALLS   #1, SMG$$MIN_UPD                             ;
                                           04 00061        RET                                                  ; 1465

; Routine Size:  98 bytes,    Routine Base:  _SMG$CODE + 035A
```

```
; 1401          1466  1 !<BLF/PAGE>
```

```
%SBTTL 'SMG$RING_BELL - Ring Bell'
GLOBAL ROUTINE SMG$RING_BELL (
                                DISPLAY_ID,
                                TIMES
                            ) =
!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine rings the bell, on each physical terminal to which
!       the given virtual display is pasted, the number of times
!       specified.  If TIMES omitted, 1 is used.
!
! CALLING SEQUENCE:
!
!       ret_status.wlc.v = SMG$RING_BELL (
!                                       DISPLAY_ID.rl.r,
!                                       [TIMES.rl.r] )
!
! FORMAL PARAMETERS:
!
!       DISPLAY_ID.rl.r The display id of the virtual display.
!
!       TIMES.rl.r      Optional.  The number of times to ring the bell.
!                       If not specified, the bell is sounded once.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! COMPLETION STATUS:
!
!       SS$_NORMAL      Normal successful completion
!       Statuses returned by SMG$$OUTPUT or
!       statuses or signal values returned by STR$DUPL_CHAR.
!
! SIDE EFFECTS:
!
!       NONE
!--

    BEGIN
    BUILTIN
        NULLPARAMETER ;

    BIND
        BELL_CHAR = UPLIT (BYTE (BELL));

    LOCAL
        DCB : REF BLOCK [,BYTE],        ! Address of display control
                                        ! block

        PP  : REF BLOCK [,BYTE],        ! Address of a pasting packet

```

```
    STATUS;                             ! Status to return to caller

$SMG$VALIDATE_ARGCOUNT (1, 2);

$SMG$GET_DCB (.DISPLAY_ID, DCB);        ! Get addr of virtual display
                                        ! control block.

STATUS=SS$_NORMAL;

IF NULLPARAMETER (TIMES)
THEN
    BEGIN   ! No arg supplied, output 1 bell
    !+
    ! Chase the chain of pasteboards to which this virtual display
    ! is pasted.  For each pasteboard located, output the bell.
    !-
    PP = .DCB [DCB_A_PP_NEXT];

    IF .PP EQL 0
    THEN
        RETURN (SMG$_FATERRLIB);                ! should never be 0

    WHILE .PP NEQ DCB [DCB_A_PP_NEXT]           ! While any remain...
    DO
        BEGIN
        STATUS=SMG$$OUTPUT ( .PP [PP_A_PBCB_ADDR],
                             1,
                             BELL_CHAR);

        PP = .PP [PP_A_NEXT_DCB];   ! Step to next packet
        END;
    END     ! No arg supplied, output 1 bell

ELSE

    BEGIN   ! Multiple bells
    LOCAL
        DESCR : BLOCK [8, BYTE] ;   ! A local descriptor

    !+
    ! Must build a string of TIMES bell characters.  Use dynamic
    ! string.
    !-
    DESCR [DSC$W_LENGTH] = 0 ;
    DESCR [DSC$B_CLASS]  = DSC$K_CLASS_D ;
    DESCR [DSC$B_DTYPE]  = DSC$K_DTYPE_T ;
    DESCR [DSC$A_POINTER] = 0 ;

    LIB$ESTABLISH ( LIB$SIG_TO_RET ) ; ! Establish handler in case
                                       ! STR$DUPL_CHAR signals out
                                       ! from under us.

    IF NOT (STATUS = STR$DUPL_CHAR ( DESCR, .TIMES, BELL_CHAR))
    THEN
        RETURN (.STATUS );

    !+
```

```
; 1517        1581 3          ! Chase the chain of pasteboards to which this virtual display
; 1518        1582 3          ! is pasted.  For each pasteboard located, output the bell(s).
; 1519        1583 3          !-
; 1520        1584 3          PP = .DCB [DCB_A_PP_NEXT];
; 1521        1585 3
; 1522        1586 3          IF .PP EQL 0
; 1523        1587 3          THEN
; 1524        1588 3              RETURN (SMG$_FATERRLIB);                ! should never be 0
; 1525        1589 3
; 1526        1590 3          WHILE .PP NEQ DCB [DCB_A_PP_NEXT]       ! While any remain...
; 1527        1591 3          DO
; 1528        1592 4              BEGIN
; 1529        1593 4              STATUS=SMG$$OUTPUT ( .PP [PP_A_PBCB_ADDR],
; 1530        1594 4                                   .DESCR [DSC$W_LENGTH],
; 1531        1595 4                                   .DESCR [DSC$A_POINTER]);
; 1532        1596 4
; 1533        1597 4              PP = .PP [PP_A_NEXT_DCB];   ! Step to next packet
; 1534        1598 3              END;
; 1535        1599 3
; 1536        1600 3          LIB$SFREE1_DD ( DESCR ) ;       ! Return dynamic string
; 1537        1601 2          END ;   ! Multiple bells
; 1538        1602 2
; 1539        1603 2      RETURN (.STATUS);
; 1540        1604 1      END;                                    ! End of routine SMG$RING_BELL


                                 07  003BC P.AAA:  .BYTE    7                                   ;

                                                   BELL_CHAR=          P.AAA


                              003C  00000          .ENTRY   SMG$RING_BELL, Save R2,R3,R4,R5     ; 1468
                55 00000000G  00 9E  00002          MOVAB    SMG$$OUTPUT, R5                     ;
                5E            08 C2  00009          SUBL2    #8, SP                              ;
     50         6C            01 83  0000C          SUBB3    #1, (AP), DIFF                      ; 1526
                01            50 91  00010          CMPB     DIFF, #1                            ;
                              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                     ; 1528
          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                                          ;
                53      04    BC D0  00036 3$:      MOVL     @DISPLAY_ID, DCB                    ;
                54            01 D0  0003A          MOVL     #1, STATUS                          ; 1531
                02            6C 91  0003D          CMPB     (AP), #2                            ; 1533
                              05 1F  00040          BLSSU    4$                                  ;
                        08    AC D5  00042          TSTL     8(AP)                               ;
                              22 12  00045          BNEQ     6$                                  ;
                52      20    A3 D0  00047 4$:      MOVL     32(DCB), PP                         ; 1540
                              50 13  0004B          BEQL     7$                                  ; 1542
                50      20    A3 9E  0004D 5$:      MOVAB    32(DCB), R0                         ; 1546
                50            52 D1  00051          CMPL     PP, R0                              ;
```

```
                    76  13 00054       BEQL    10$                          ;
             A6     AF  9F 00056       PUSHAB  BELL_CHAR                    ; 1549
             01     DD  00059          PUSHL   #1                           ;
             14     A2  DD 0005B       PUSHL   20(PP)                       ;
      65            03  FB 0005E       CALLS   #3, SMG$$OUTPUT              ;
      54            50  D0 00061       MOVL    R0, STATUS                   ;
      52            62  D0 00064       MOVL    (PP), PP                     ; 1553
                    E4  11 00067       BRB     5$                           ; 1546
      6E 020E0000   8F  D0 00069 6$:   MOVL    #34471936, DESCR             ; 1567
             04     AE  D4 00070       CLRL    DESCR+4                      ; 1570
      00000000G     00  9F 00073       PUSHAB  LIB$SIG_TO_RET               ; 1572
00000000G  00       01  FB 00079       CALLS   #1, LIB$ESTABLISH            ;
           FF7B     CF  9F 00080       PUSHAB  BELL_CHAR                    ; 1576
             08     AC  DD 00084       PUSHL   TIMES                        ;
             08     AE  9F 00087       PUSHAB  DESCR                        ;
00000000G  00       03  FB 0008A       CALLS   #3, STR$DUPL_CHAR            ;
      54            50  D0 00091       MOVL    R0, STATUS                   ;
      35            54  E9 00094       BLBC    STATUS, 10$                  ;
      52     20     A3  D0 00097       MOVL    32(DCB), PP                  ; 1584
                    08  12 0009B       BNEQ    8$                           ; 1586
      50 00000000G  8F  D0 0009D 7$:   MOVL    #SMG$_FATERRLIB, R0          ; 1588
                        04 000A4       RET                                  ;
      50     20     A3  9E 000A5 8$:   MOVAB   32(DCB), R0                  ; 1590
      50            52  D1 000A9       CMPL    PP, R0                       ;
                    15  13 000AC       BEQL    9$                           ;
             04     AE  DD 000AE       PUSHL   DESCR+4                      ; 1595
      7E     04     AE  3C 000B1       MOVZWL  DESCR, -(SP)                 ; 1594
             14     A2  DD 000B5       PUSHL   20(PP)                       ; 1593
      65            03  FB 000B8       CALLS   #3, SMG$$OUTPUT              ;
      54            50  D0 000BB       MOVL    R0, STATUS                   ;
      52            62  D0 000BE       MOVL    (PP), PP                     ; 1597
                    E2  11 000C1       BRB     8$                           ; 1590
                    5E  DD 000C3 9$:   PUSHL   SP                           ; 1600
00000000G  00       01  FB 000C5       CALLS   #1, LIB$SFREE1_DD            ;
      50            54  D0 000CC 10$:  MOVL    STATUS, R0                   ; 1603
                        04 000CF       RET                                  ; 1604

; Routine Size:  208 bytes,    Routine Base:  _SMG$CODE + 03BD

; 1541         1605  1 !<BLF/PAGE>
```

```
%SBTTL 'SMG$$BEGIN_PASTEBOARD_UPDATE_R1 - Begin batch of updates to pasteboard'
GLOBAL ROUTINE SMG$$BEGIN_PASTEBOARD_UPDATE_R1 (PBCB: REF $PBCB_DECL)
                                : SMG$$BEGIN_PBD_UPDATE$LNK =
!++
! FUNCTIONAL DESCRIPTION:
!
!       Inner routine to support BEGIN_PASTEBOARD_UPDATE action.
!       Disable outputing this pasteboard to the screen until the
!       matching call to SMG$$END_PASTEBOARD_UPDATE_R2 is encountered.
!
! CALLING SEQUENCE:
!
!       ret_status.wlc.v = SMG$$BEGIN_PASTEBOARD_UPDATE_R1 ( PBCB.rab.r)
!
! FORMAL PARAMETERS:
!
!       PBCB.rab.r              Address of pasteboard control block
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! COMPLETION STATUS:
!
!       SS$_NORMAL        Normal successful completion, batching has been
!                         initiated
!       SMG$_BATWAS_ON    Success, but note that batching was already on
!
! SIDE EFFECTS:
!
!       NONE
!--

    BEGIN

!+
! Increment count of number of SMG$$END_PASTEBOARD_UPDATE_R2 calls we need to
! see before we resume outputing from this pasteboard.
! Let the user know what the previous state of batching was by
! our status return (in case he's interested).
!-
    PBCB [PBCB_L_BATCH_LEVEL] = .PBCB [PBCB_L_BATCH_LEVEL] + 1;

    IF .PBCB [PBCB_L_BATCH_LEVEL] EQL 1
    THEN
        RETURN  SS$_NORMAL
    ELSE
        RETURN  SMG$_BATWAS_ON

    END;                ! End of routine SMG$$BEGIN_PASTEBOARD_UPDATE_R1
```

```
                     00A4    C0  D6 00000 SMG$$BEGIN_PASTEBOARD_UPDATE_R1::
                                                  INCL    164(PBCB)                                 ; 1651
             01      00A4    C0  D1 00004         CMPL    164(PBCB), #1                             ; 1653
                             04  12 00009         BNEQ    1$                                        ;
             50              01  D0 0000B         MOVL    #1, R0                                    ; 1657
                                 05 0000E         RSB                                               ;
             50 00000000G    8F  D0 0000F 1$:     MOVL    #SMG$_BATWAS_ON, R0                       ;
                                 05 00016         RSB                                               ; 1659

; Routine Size:  23 bytes,    Routine Base:  _SMG$CODE + 048D

; 1597          1660  1 !<BLF/PAGE>
```

```
1 %SBTTL 'SMG$$END_PASTEBOARD_UPDATE_R2 - End batch of updates to pasteboard'
1 GLOBAL ROUTINE SMG$$END_PASTEBOARD_UPDATE_R2 ( PBCB: REF $PBCB_DECL)
1                              : SMG$$END_PBD_UPDATE$LNK =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Inner routine to support BEGIN_PASTEBOARD_UPDATE action.
1 !       Reduce the number of calls to SMG$$END_PASTEBOARD_UPDATE_R2 that will
1 !       be needed before we resume outputing from this pasteboard.  If
1 !       this call makes this count go to zero, flush the pasteboard to
1 !       the screen.
1 !       If the level is already 0, we return a success status
1 !       informing caller that the batching level was already 0,
1 !       but we otherwise allow this.  This gives a user a guaranteed
1 !       method of ending batching.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       ret_status.wlc.v = SMG$$END_PASTEBOARD_UPDATE_R2 ( PBCB.rab.r)
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       PBCB.rab.r              Address of pasteboard control block
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       PBCB [PBCB_L_BATCH_LEVEL]
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       PBCB [PBCB_L_BATCH_LEVEL] gets decremented
1 !
1 ! COMPLETION STATUS:
1 !
1 !       SS$_NORMAL        Normal successful completion
1 !       SMG$_BATSTIPRO    Success; but batching is still in progress.
1 !       SMG$_BATWASOFF    Success; but batching was already off
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !--
1
2     BEGIN
2     LOCAL
2       STATUS;                         ! Status to return
2
2 !+
2 ! If current count is greater than 1, simply reduce it by one and
2 ! return to caller.  If it is one, reduce it to zero and cause the
2 ! current contents of this pasteboard to be flushed to the screen (if need
2 ! be).
2 !-
2     IF .PBCB [PBCB_L_BATCH_LEVEL] GTRU 1
2     THEN
3       BEGIN
3       PBCB [PBCB_L_BATCH_LEVEL] = .PBCB [PBCB_L_BATCH_LEVEL] - 1;
```

```
; 1656        1718 3            STATUS = SMG$_BATSTIPRO          ! Ok, but batching is still in progress
; 1657        1719 3            END
; 1658        1720 2        ELSE
; 1659        1721 3            BEGIN   ! Being reduced to zero
; 1660        1722 3            LOCAL BATCH_STATUS;
; 1661        1723 3            BATCH_STATUS = SS$_NORMAL;
; 1662        1724 3            IF .PBCB [PBCB_L_BATCH_LEVEL] EQL 0
; 1663        1725 3            THEN
; 1664        1726 3                BATCH_STATUS = SMG$_BATWASOFF        ! Ok, but batching was already off
; 1665        1727 3            ELSE
; 1666        1728 3                PBCB [PBCB_L_BATCH_LEVEL] = 0;
; 1667        1729 3            !+
; 1668        1730 3            ! Call SMG$$CHECK_FOR_OUTPUT_PBCB to cause the
; 1669        1731 3            ! contents of this pasteboard to be
; 1670        1732 3            ! flushed to the screen.
; 1671        1733 3            ! It may not actually get there if buffering is in effect.
; 1672        1734 3            ! If that fails, then return its status as our status.
; 1673        1735 3            ! If that succeeded then return our previously calculated status.
; 1674        1736 3            !-
; 1675        1737 3            STATUS = SMG$$CHECK_FOR_OUTPUT_PBCB (.PBCB);
; 1676        1738 3            IF .STATUS THEN STATUS=.BATCH_STATUS
; 1677        1739 2            END;     ! Being reduced to zero
; 1678        1740 2
; 1679        1741 2        RETURN      .STATUS
; 1680        1742 1        END;                ! End of routine SMG$$END_PASTEBOARD_UPDATE_R2


                 5E              04  C2 00000 SMG$$END_PASTEBOARD_UPDATE_R2::
                                                        SUBL2   #4, SP                              ; 1662
                 51       00A4   C0  9E 00003           MOVAB   164(PBCB), R1                       ; 1714
                 01              61  D1 00008           CMPL    (R1), #1                            ;
                                 0B  1B 0000B           BLEQU   1$                                  ;
                                 61  D7 0000D           DECL    (R1)                                ; 1717
                 52 00000000G    8F  D0 0000F           MOVL    #SMG$_BATSTIPRO, STATUS             ; 1718
                                 22  11 00016           BRB     4$                                  ;
                 6E              01  D0 00018 1$:       MOVL    #1, BATCH_STATUS                    ; 1723
                                 61  D5 0001B           TSTL    (R1)                                ; 1724
                                 09  12 0001D           BNEQ    2$                                  ;
                 6E 00000000G    8F  D0 0001F           MOVL    #SMG$_BATWASOFF, BATCH_STATUS       ; 1726
                                 02  11 00026           BRB     3$                                  ;
                                 61  D4 00028 2$:       CLRL    (R1)                                ; 1728
                                 50  DD 0002A 3$:       PUSHL   PBCB                                ; 1737
            0000V  CF            01  FB 0002C           CALLS   #1, SMG$$CHECK_FOR_OUTPUT_PBCB      ;
                   52            50  D0 00031           MOVL    R0, STATUS                          ;
                   03            52  E9 00034           BLBC    STATUS, 4$                          ; 1738
                   52            6E  D0 00037           MOVL    BATCH_STATUS, STATUS                ;
                   50            52  D0 0003A 4$:       MOVL    STATUS, R0                          ; 1741
                   5E            04  C0 0003D           ADDL2   #4, SP                              ; 1742
                                     05 00040           RSB                                         ;

; Routine Size:  65 bytes,    Routine Base:  _SMG$CODE + 04A4


; 1681        1743  1 !<BLF/PAGE>
```

```
%SBTTL 'SMG$$CHECK_FOR_OUTPUT_DCB - Check to see if output needed'
GLOBAL ROUTINE SMG$$CHECK_FOR_OUTPUT_DCB (
                                        DCB : REF BLOCK [,BYTE],
                                        REQUEST_CODE,
                                        REQUEST_ARG
                                   ) =
!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine is called when ever a change has been completed
!       to a given virtual display.  Changes consist of:
!               1. Altering the text in a virtual display
!               2. Altering the attributes of the text in a virtual
!                  display.
!               3. Altering the cursor position within a display.
!               4. A virtual display's buffering level going to zero.
!               5. A virtual display has just been pasted to a
!                  pasteboard.
!       The criteria for outputing consist of:
!               1. Virtual display buffering level is zero
!               2. The virtual display is pasted to at least one
!                  pasteboard.
!
!       Each pasteboard to which this display is pasted is isolated and
!       its window image must be redrawn.
!       The new physical pasteboard cursor position gets set.
!
! CALLING SEQUENCE:
!
!       ret_status.wlc.v = SMG$$CHECK_FOR_OUTPUT_DCB (
!                                               DCB.rab.r
!                                               [,REQEST_CODE.rl.v])
!
! FORMAL PARAMETERS:
!
!       DCB.rab.r       Address of virtual display control block for
!                       affected display.
!
!       [REQUEST_CODE.rl.v] Optional.  If supplied, it is a code telling
!                       the nature of the request that provoked this
!                       call to SMG$$CHECK_FOR_OUTPUT_DCB.
!                       Among other things, it tells us when it is
!                       necessary to redraw borders.
!
!       [REQUEST_ARG.rl.v] Optional. If supplied, then it tells us the
!                       unique line in the DCB that has changed.
!                       If specified as 0, then the whole DCB has changed.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       WCB[WCB_W_CURR_CUR_ROW]         gets set to place corresponding
!                                       to logical cursor position in
!                                       this virtual display.
```

```
1 !
1 !        WCB[WCB_W_CURR_CUR_COL]          gets set to place corresponding
1 !                                         to logical cursor position in
1 !                                         this virtual display.
1 !
1 ! COMPLETION STATUS:
1 !
1 !        SS$_NORMAL      Normal successful completion
1 !
1 ! SIDE EFFECTS:
1 !
1 !        NONE
1 !--
1
2     BEGIN
2     BUILTIN
2        ACTUALCOUNT;
2
2     LOCAL
2        CURR_PP : REF BLOCK [,BYTE],      ! Addr of pasting packet under
2                                          ! inspection
2
2        STATUS; ! Status of subroutine calls
2
2 !+
2 ! If current buffer level not equal to zero, this is not the time to
2 ! flush this display to the screen.
2 !-
2     IF .DCB [DCB_L_BATCH_LEVEL] NEQ 0
2     THEN
2        RETURN (SS$_NORMAL);
2
2     CURR_PP = .DCB [DCB_A_PP_NEXT];      ! Start of chain of pasting
2                                          ! packets to which this display
2                                          ! is pasted.
2     IF .CURR_PP EQL 0
2     THEN
2        RETURN (SMG$_FATERRLIB);          ! should never be 0
2
2 !+
2 ! Deal with each pasteboard that this virtual display is pasted to...
2 !-
2     WHILE .CURR_PP NEQ DCB [DCB_A_PP_NEXT]
2     DO
3        BEGIN   ! Overall loop
3
3        LOCAL
3           PBCB : REF BLOCK [,BYTE],   ! Addr. of pasteboard control
3           WCB  : REF BLOCK [,BYTE];   ! Addr. of window control block
3
3        PBCB = .CURR_PP [PP_A_PBCB_ADDR]; ! Select pasteboard and WCB
3
3        IF .PBCB [PBCB_L_BATCH_LEVEL] EQL 0
3        THEN
4           BEGIN       ! Force output
4           LOCAL
4              OPT_FLAG;       ! Flag
```

```
WCB  = .PBCB [PBCB_A_WCB]; ! whose window needs rebuilding.

!+
! Set the cursor position in the pasteboard to correspond
! to it's place on this virtual display.
!-
WCB [WCB_W_CURR_CUR_ROW] = .DCB [DCB_W_CURSOR_ROW]-1
                         + .CURR_PP [PP_Q_ROW];

WCB [WCB_W_CURR_CUR_COL] = .DCB [DCB_W_CURSOR_COL]-1
                         + .CURR_PP [PP_Q_COL];

!+
! If REQUEST_CODE is present and it is a type that just
! involves cursor motion, deal with it as a special case to
! avoid remapping the virtual display since contents did
! not change.
!-
OPT_FLAG = 0;
IF ACTUALCOUNT () GEQ 2      ! REQUEST_CODE present
THEN
    BEGIN   ! Try optimization
    IF .REQUEST_CODE EQL SMG$C_SET_CURSOR_ABS OR
       .REQUEST_CODE EQL SMG$C_SET_CURSOR_REL
    THEN
        OPT_FLAG = 1;
    END;    ! Try optimization

IF .OPT_FLAG
THEN
    BEGIN   ! Opt. possible
    !+
    ! Force cursor to desired spot in optimal fashion
    !-
    SMG$$UPDATE_PHYSICAL_CURSOR ( .PBCB);
    END     ! Opt. possible

ELSE

    BEGIN   ! Remapping necessary
    !+
    ! Check to see if we can get by with just redrawing the
    ! changed virtual display, or whether we need to redraw
    ! the changed one and all higher-pasted virtual
    ! displays.  We can do the former if this pasting has a
    ! bit that says it is not occluded.
    !-
    IF NOT .CURR_PP [PP_V_OCCLUDED]
    THEN
        BEGIN       ! Redraw just this virtual display
        IF NOT (STATUS = SMG$$MOVE_TEXT_TO_WINDOW_BUF (.CURR_PP))
        THEN
            RETURN (.STATUS);

        !+
        ! If something actually got redrawn, tell the output
```

```
                ! routines we've changed the whole virtual display.
                ! If just one line has changed, however, then tell
                ! the output routines which line changed.
                !-
                IF .CURR_PP [PP_W_ROWS_TO_MOVE] NEQ 0
                THEN
                    BEGIN
                    PBCB [PBCB_W_FIRST_CHANGED_ROW] =
                            .CURR_PP [PP_W_FIRST_WCB_ROW];

                    PBCB [PBCB_W_LAST_CHANGED_ROW]  =
                            .CURR_PP [PP_W_LAST_WCB_ROW];

                    PBCB [PBCB_W_FIRST_CHANGED_COL] =
                            .CURR_PP [PP_W_FIRST_WCB_COL];

                    PBCB [PBCB_W_LAST_CHANGED_COL]  =
                            .CURR_PP [PP_W_LAST_WCB_COL];
                    END;

                IF ACTUALCOUNT () GEQ 3 ! If REQUEST_ARG supplied
                THEN
                    IF .REQUEST_CODE EQL SMG$C_PUT_CHARS AND
                       .REQUEST_ARG  NEQ 0
                    THEN
                        BEGIN       ! Just one row changed
                        PBCB [PBCB_W_FIRST_CHANGED_ROW] =
                            .CURR_PP [PP_W_ROW]+.REQUEST_ARG-1;

                        PBCB [PBCB_W_LAST_CHANGED_ROW]  =
                            .CURR_PP [PP_W_ROW]+.REQUEST_ARG-1;

                        END;        ! Just one row changed

                !+
                ! If REQUEST_CODE is present and it is of a type
                ! that could have added a border, we must redraw the
                ! border at this time.
                !-
                IF ACTUALCOUNT () GEQ 2 ! If REQUEST_CODE supplied
                THEN
                    IF .REQUEST_CODE EQL SMG$C_LABEL_BORDER
                    THEN
                        IF .DCB [DCB_V_BORDERED]
                        THEN
                            IF NOT (STATUS = SMG$$DRAW_BORDER (
                                            .DCB, .CURR_PP, .WCB))
                            THEN
                                RETURN (.STATUS);

                SMG$$MIN_UPD (.PBCB);

                END         ! Redraw just this virtual display
            ELSE
                BEGIN       ! Must redraw it and outer ones
```

```
; 1911        1972  6                     !+
; 1912        1973  6                     ! Go rebuild window buffers for this pasteboard
; 1913        1974  6                     ! starting with current pasting packet, and output.
; 1914        1975  6                     !-
; 1915        1976  7                     IF NOT (STATUS = SMG$$FILL_WINDOW_BUFFER ( .CURR_PP))
; 1916        1977  6                     THEN
; 1917        1978  6                         RETURN (.STATUS);
; 1918        1979  6
; 1919        1980  5                     END;            ! Must redraw it and outer ones
; 1920        1981  4                 END;        ! Remapping necessary
; 1921        1982  3             END;            ! Force output
; 1922        1983  3
; 1923        1984  3     CURR_PP = .CURR_PP [PP_A_NEXT_DCB];         ! Walk the DCB side of
; 1924        1985  3                                                 ! the chain from front
; 1925        1986  3                                                 ! to back.
; 1926        1987  2     END;      ! Overall loop
; 1927        1988  2
; 1928        1989  2   RETURN (SS$_NORMAL);
; 1929        1990  1   END;                        ! End of routine SMG$$CHECK_FOR_OUTPUT_DCB


                                       007C 00000        .ENTRY  SMG$$CHECK_FOR_OUTPUT_DCB, Save R2,R3,R4,-   ; 1745
                                                                 R5,R6
                   55        04   AC   D0 00002          MOVL    DCB, R5                                     ; 1829
                             1C   A5   D5 00006          TSTL    28(R5)                                      ;
                                  03   13 00009          BEQL    2$                                          ;
                                00E9   31 0000B 1$:      BRW     14$                                         ;
                   52        20   A5   D0 0000E 2$:      MOVL    32(R5), CURR_PP                             ; 1833
                                  08   12 00012          BNEQ    3$                                          ; 1836
                   50 00000000G   8F   D0 00014          MOVL    #SMG$_FATERRLIB, R0                         ; 1838
                                       04 0001B          RET                                                 ;
                   50        20   A5   9E 0001C 3$:      MOVAB   32(R5), R0                                  ; 1843
                   50             52   D1 00020          CMPL    CURR_PP, R0                                 ;
                                  E6   13 00023          BEQL    1$                                          ;
                   54        14   A2   D0 00025          MOVL    20(CURR_PP), PBCB                           ; 1851
                           00A4   C4   D5 00029          TSTL    164(PBCB)                                   ; 1853
                                  46   12 0002D          BNEQ    6$                                          ;
                   53        08   A4   D0 0002F          MOVL    8(PBCB), WCB                                ; 1859
                   50        28   A5   3C 00033          MOVZWL  40(R5), R0                                  ; 1866
                   51        18   A2   32 00037          CVTWL   24(CURR_PP), R1                             ;
                   50             51   C0 0003B          ADDL2   R1, R0                                      ;
      20    A3     50             01   A3 0003E          SUBW3   #1, R0, 32(WCB)                             ;
                   50        2A   A5   3C 00043          MOVZWL  42(R5), R0                                  ; 1869
                   51        1A   A2   32 00047          CVTWL   26(CURR_PP), R1                             ;
                   50             51   C0 0004B          ADDL2   R1, R0                                      ;
      22    A3     50             01   A3 0004E          SUBW3   #1, R0, 34(WCB)                             ;
                                  50   D4 00053          CLRL    OPT_FLAG                                    ; 1877
                   02             6C   91 00055          CMPB    (AP), #2                                    ; 1878
                                  0F   1F 00058          BLSSU   5$                                          ;
                   16        08   AC   D1 0005A          CMPL    REQUEST_CODE, #22                           ; 1881
                                  06   13 0005E          BEQL    4$                                          ;
                   17        08   AC   D1 00060          CMPL    REQUEST_CODE, #23                           ; 1882
                                  03   12 00064          BNEQ    5$                                          ;
                   50             01   D0 00066 4$:      MOVL    #1, OPT_FLAG                                ; 1884
```

```
                      0B              50  E9 00069 5$:    BLBC    OPT_FLAG, 7$                              ; 1887
                                      54  DD 0006C        PUSHL   PBCB                                      ; 1893
            00000000G 00              01  FB 0006E        CALLS   #1, SMG$$UPDATE_PHYSICAL_CURSOR            ;
                                      7A  11 00075 6$:    BRB     13$                                       ; 1887
                      65       2A     A2  E8 00077 7$:    BLBS    42(CURR_PP), 11$                          ; 1906
                                      52  DD 0007B        PUSHL   CURR_PP                                   ; 1909
                0000V CF              01  FB 0007D        CALLS   #1, SMG$$MOVE_TEXT_TO_WINDOW_BUF          ;
                      56              50  D0 00082        MOVL    R0, STATUS                                ;
                      65              56  E9 00085        BLBC    STATUS, 12$                               ;
                               1C     A2  B5 00088        TSTW    28(CURR_PP)                               ; 1919
                                      06  13 0008B        BEQL    8$                                        ;
                 00A8 C4       2F     A2  7D 0008D        MOVQ    47(CURR_PP), 168(PBCB)                    ; 1923
                      03              6C  91 00093 8$:    CMPB    (AP), #3                                  ; 1935
                                      1F  1F 00096        BLSSU   9$                                        ;
                      11       08     AC  D1 00098        CMPL    REQUEST_CODE, #17                         ; 1937
                                      19  12 0009C        BNEQ    9$                                        ;
                               0C     AC  D5 0009E        TSTL    REQUEST_ARG                               ; 1938
                                      14  13 000A1        BEQL    9$                                        ;
                      50       18     A2  32 000A3        CVTWL   24(CURR_PP), R0                           ; 1942
                      50       0C     AC  C0 000A7        ADDL2   REQUEST_ARG, R0                           ;
                                      50  D7 000AB        DECL    R0                                        ;
                 00A8 C4              50  B0 000AD        MOVW    R0, 168(PBCB)                             ;
                 00AA C4              50  B0 000B2        MOVW    R0, 170(PBCB)                             ; 1945
                      02              6C  91 000B7 9$:    CMPB    (AP), #2                                  ; 1954
                                      19  1F 000BA        BLSSU   10$                                       ;
                      1C       08     AC  D1 000BC        CMPL    REQUEST_CODE, #28                         ; 1956
                                      13  12 000C0        BNEQ    10$                                       ;
                      0F       2F     A5  E9 000C2        BLBC    47(R5), 10$                               ; 1958
                                      0C  BB 000C6        PUSHR   #^M<R2,R3>                                ; 1961
                                      55  DD 000C8        PUSHL   R5                                        ;
                0000V CF              03  FB 000CA        CALLS   #3, SMG$$DRAW_BORDER                      ;
                      56              50  D0 000CF        MOVL    R0, STATUS                                ;
                      18              56  E9 000D2        BLBC    STATUS, 12$                               ; 1960
                                      54  DD 000D5 10$:   PUSHL   PBCB                                      ; 1965
            00000000G 00              01  FB 000D7        CALLS   #1, SMG$$MIN_UPD                          ;
                                      11  11 000DE        BRB     13$                                       ; 1906
                                      52  DD 000E0 11$:   PUSHL   CURR_PP                                   ; 1976
                0000V CF              01  FB 000E2        CALLS   #1, SMG$$FILL_WINDOW_BUFFER               ;
                      56              50  D0 000E7        MOVL    R0, STATUS                                ;
                      04              56  E8 000EA        BLBS    STATUS, 13$                               ;
                      50              56  D0 000ED 12$:   MOVL    STATUS, R0                                ; 1978
                                          04 000F0        RET                                               ;
                      52              62  D0 000F1 13$:   MOVL    (CURR_PP), CURR_PP                        ; 1984
                                    FF25  31 000F4        BRW     3$                                        ; 1843
                      50              01  D0 000F7 14$:   MOVL    #1, R0                                    ; 1989
                                          04 000FA        RET                                               ; 1990

; Routine Size:  251 bytes,    Routine Base:  _SMG$CODE + 04E5


; 1930         1991  1 !<BLF/PAGE>
```

```
1 %SBTTL 'SMG$$CHECK_FOR_OUTPUT_PBCB - Check to see if output needed'
1 GLOBAL ROUTINE SMG$$CHECK_FOR_OUTPUT_PBCB (
1                                   PBCB : REF BLOCK [,BYTE]
1                                   ) =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine is called when a change has been made to a
1 !       particular pasteboard that requires the entire screen image to
1 !       be recomposed, e.g. as a result of an UNPASTE operation.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       ret_status.wlc.v = SMG$$CHECK_FOR_OUTPUT_PBCB ( PBCB.rab.r)
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       PBCB.rab.r      Address of pasteboard control block for
1 !                       affected display.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       PBCB [PBCB_L_BATCH_LEVEL]
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! COMPLETION STATUS:
1 !
1 !       SMG$_BATWAS_ON  Ok, but batching was on so nothing happened
1 !       SS$_NORMAL      Normal successful completion
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !--
1
2     BEGIN
2     LOCAL
2         WCB : REF BLOCK [,BYTE],        ! Address of window control
2                                         ! block
2         STATUS;                         ! Status of subroutine calls
2
2     WCB = .PBCB [PBCB_A_WCB];
2
2 !+
2 ! Do nothing if batching is in effect.
2 !-
2
2     IF .PBCB [PBCB_L_BATCH_LEVEL] NEQ 0
2     THEN    RETURN SMG$_BATWAS_ON;
2
2 !+
2 ! Clear text buffer to blanks and attribute buffer to zeroes before we
2 ! start.  Clear alternate character set buffer to zeroes if it exists.
2 !-
```

```
; 1989      2049 2      CH$FILL (%C' ', .WCB [WCB_L_BUFSIZE], .WCB [WCB_A_TEXT_BUF]);
; 1990      2050 2
; 1991      2051 2      CH$FILL (0,     .WCB [WCB_L_BUFSIZE], .WCB [WCB_A_ATTR_BUF]);
; 1992      2052 2
; 1993      2053 2      IF .WCB [WCB_A_CHAR_SET_BUF] NEQ 0
; 1994      2054 2      THEN
; 1995      2055 2          CH$FILL (0, .WCB [WCB_L_BUFSIZE], .WCB [WCB_A_CHAR_SET_BUF]);
; 1996      2056 2
; 1997      2057 2  !+
; 1998      2058 2  ! Clear text line characteristics vector to zero.
; 1999      2059 2  !-
; 2000      2060 2      CH$FILL (0, .WCB [WCB_W_NO_ROWS] +1, .WCB [WCB_A_LINE_CHAR]);
; 2001      2061 2
; 2002      2062 2  !+
; 2003      2063 2  ! Tell the output routines that we've changed the whole WCB buffer.
; 2004      2064 2  !-
; 2005      2065 2      PBCB [PBCB_W_FIRST_CHANGED_ROW] = 1;
; 2006      2066 2      PBCB [PBCB_W_LAST_CHANGED_ROW]  = .PBCB [PBCB_B_ROWS];
; 2007      2067 2      PBCB [PBCB_W_FIRST_CHANGED_COL] = 1;
; 2008      2068 2      PBCB [PBCB_W_LAST_CHANGED_COL]  = .PBCB [PBCB_W_WIDTH];
; 2009      2069 2
; 2010      2070 2  !+
; 2011      2071 2  ! Go rebuild window buffers for this pasteboard and output.
; 2012      2072 2  ! Pass SMG$$FILL_WINDOW_BUFFER the address of the first pasting packet
; 2013      2073 2  ! if any exist, else just output now-blank screen.
; 2014      2074 2  !-
; 2015      2075 2      IF .PBCB [PBCB_A_PP_NEXT] EQL PBCB [PBCB_A_PP_NEXT]
; 2016      2076 2      THEN
; 2017      2077 3          BEGIN   ! Nothing pasted -- output blanked buffer
; 2018      2078 3          RETURN (SMG$$MIN_UPD (.PBCB));
; 2019      2079 3          END     ! Nothing pasted -- output blanked buffer
; 2020      2080 3
; 2021      2081 2      ELSE
; 2022      2082 2
; 2023      2083 3          RETURN (SMG$$FILL_WINDOW_BUFFER (
; 2024      2084 2                          .PBCB [PBCB_A_PP_PREV] - PP_PBCB_QUEUE_OFFSET) );
; 2025      2085 2
; 2026      2086 1      END;                ! End of routine SMG$$CHECK_FOR_OUTPUT_PBCB


                                     00FC 00000         .ENTRY  SMG$$CHECK_FOR_OUTPUT_PBCB, Save R2,R3,R4,-  ; 1993
                                                                R5,R6,R7                                     ;
                  57         04   AC  D0 00002          MOVL    PBCB, R7                                     ; 2036
                  56         08   A7  D0 00006          MOVL    8(R7), WCB                                   ;
                           00A4   C7  D5 0000A          TSTL    164(R7)                                      ; 2042
                             08   13     0000E          BEQL    1$                                           ;
                  50 00000000G    8F  D0 00010          MOVL    #SMG$_BATWAS_ON, R0                          ; 2043
                                      04 00017          RET                                                  ;
  28   A6      20         6E          00 2C 00018 1$:   MOVC5   #0, (SP), #32, 40(WCB), @8(WCB)              ; 2049
                             08   B6     0001E                                                               ;
  28   A6      00         6E          00 2C 00020       MOVC5   #0, (SP), #0, 40(WCB), @12(WCB)              ; 2051
                             0C   B6     00026                                                               ;
                             10   A6  D5 00028          TSTL    16(WCB)                                      ; 2053
                             08   13     0002B          BEQL    2$                                           ;
```

```
     28  A6            00              6E          00  2C 0002D        MOVC5   #0, (SP), #0, 40(WCB), a16(WCB)          ; 2055
                                               10  B6     00033                                                         ;
                                       50      02  A6  3C 00035 2$:    MOVZWL  2(WCB), R0                               ; 2060
                                                   50  D6 00039        INCL    R0                                       ;
         50            00              6E          00  2C 0003B        MOVC5   #0, (SP), #0, R0, a44(WCB)               ;
                                               2C  B6     00040                                                         ;
                            00A8   C7              01  B0 00042        MOVW    #1, 168(R7)                              ; 2065
                            00AA   C7          5F  A7  9B 00047        MOVZBW  95(R7), 170(R7)                          ; 2066
                            00AC   C7              01  B0 0004D        MOVW    #1, 172(R7)                              ; 2067
                            00AE   C7          5A  A7  B0 00052        MOVW    90(R7), 174(R7)                          ; 2068
                                   57              67  D1 00058        CMPL    (R7), R7                                 ; 2075
                                                   0A  12 0005B        BNEQ    3$                                       ;
                                                   57  DD 0005D        PUSHL   R7                                       ; 2078
                       00000000G   00              01  FB 0005F        CALLS   #1, SMG$$MIN_UPD                         ;
                                                       04 00066        RET                                              ; 2083
                       7E      04  A7              08  C3 00067 3$:    SUBL3   #8, 4(R7), -(SP)                         ; 2084
                            0000V  CF              01  FB 0006C        CALLS   #1, SMG$$FILL_WINDOW_BUFFER              ;
                                                       04 00071        RET                                              ; 2086

; Routine Size:  114 bytes,    Routine Base:  _SMG$CODE + 05E0

; 2027          2087  1 !<BLF/PAGE>
```

```
%SBTTL 'SMG$$DRAW_BORDER - Move border characters into buffer'
GLOBAL ROUTINE SMG$$DRAW_BORDER (
        DCB : REF BLOCK [,BYTE],        ! Display control block address
        PP  : REF BLOCK [,BYTE],        ! Pasting packet address
        WCB : REF BLOCK [,BYTE]         ! Window control block address
                                ) =
!++
! FUNCTIONAL DESCRIPTION:
!
!       Draw the border characters into the text buffer of the WCB.
!       The cell in the text buffer has bits set to indicate which
!       elements of a border character will be needed in this cell
!       position.  When the first such bit is set in the text cell,
!       the corresponding cell in the attribute array is flag with
!       a bit which signals that the text array cell no longer contains
!       a printable character, but an encoding of the border character.
!
! CALLING SEQUENCE:
!
!       ret_status.wlc.v = SMG$$DRAW_BORDER (   DCB.rab.r,
!                                               PP.rab.r,
!                                               WCB.rab.r)
!
! FORMAL PARAMETERS:
!
!       DCB.rab.r               ! Display control block address
!       PP.rab.r                ! Pasting packet address
!       WCB.rab.r               ! Window control block address
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! COMPLETION STATUS:
!
!       SS$_NORMAL      Normal successful completion
!
! SIDE EFFECTS:
!
!       NONE
!--

    BEGIN

!+
! Macro $MARKIT checks a "text" character to see if it is already a
! border element in which case it OR's in new border contribution.
! If it is not yet flagged as being a border element it stores the
! border contribution and marks attribute byte to record this is a
! border element.
!-
MACRO
    $MARKIT ( OFFSET, ELEMENT ) =
```

```
        BEGIN
        IF .(PTRA [.OFFSET])<ATTR_V_BORD_ELEM, 1>
        THEN
            PTRT [.OFFSET] = .PTRT [.OFFSET] OR ELEMENT
        ELSE
            BEGIN
            PTRT [.OFFSET] = ELEMENT;
            PTRA [.OFFSET] = ATTR_M_BORD_ELEM ;
            END;
        END %;

    !+
    ! Macro $CALC_LEFT_INDEX computes the byte offset into the WCB text and
    ! attribute buffers where a left border element needs to be placed for
    ! a double-wide line to make it come out in the correct column.
    !-
    MACRO
        $CALC_LEFT_INDEX =
        BEGIN
            SINDEX = (.WLN -1) * .WCB [WCB_W_NO_COLS] +
                               (.PP [PP_W_FIRST_WCB_COL] / 2) -1;
        END %;

    !+
    ! Macro $CALC_RIGHT_INDEX computes the byte offset into the WCB text and
    ! attribute buffers where a right border element needs to be placed for
    ! a double-wide line to make it come out in the correct column.
    !-
    MACRO
        $CALC_RIGHT_INDEX =
        BEGIN
            SINDEX = (.WLN -1) * .WCB [WCB_W_NO_COLS] +
                     ((.PP [PP_W_FIRST_WCB_COL] + .PP [PP_W_MOVE_LENGTH] +1) / 2) -1;
        END %;

        LOCAL
            SPOS ,
            ! Position where the first (or only) border element will be
            ! written.  This is a displacement from the beginning of the
            ! window buffer.  Before using this position, we must always
            ! check to see if it points into the confines of the buffer.
            ! This basically tells us if that portion of the border is
            ! visible considering how the virtual display is pasted to the
            ! pasteboard.

            LDES : REF BLOCK [,BYTE],           ! Address of the dynamic string
                                                ! descriptor in the DCB which
                                                ! records the border label text
                                                ! string.

            PTRT : REF VECTOR [,BYTE],          ! pointer to top of WCB TEXT buf
            PTRA : REF VECTOR [,BYTE],          ! pointer to top of WCB ATTR buf
            UPPER_ROW,          ! Row number of top border
            LOWER_ROW,          ! Row number of bottom border
            LEFT_COL,           ! Col number of left border
            RIGHT_COL,          ! Col number of right border
            WCB_LCV : REF VECTOR [,BYTE],   ! Line control block
```

```
2               PBCB : REF BLOCK [,BYTE];               ! Addr. of pasteboard control
2                                                       ! block
2
2
2           PTRT = .WCB [WCB_A_TEXT_BUF];
2           PTRA = .WCB [WCB_A_ATTR_BUF];
2           PBCB = .PP [PP_A_PBCB_ADDR];
2           WCB_LCV = .WCB [WCB_A_LINE_CHAR];
2
2       !+
2       ! If no part of the virtual display projects to the visible part of the
2       ! pasteboard, abandon attempt to draw border.  This is strictly speaking not
2       ! the correct solution since it ignores the case where a virtual display is
2       ! pasted such that it is not visible, but its border should be.  Trying to
2       ! make this pathelogical case work will require some coordinated work on the
2       ! part of SMG$$CALC_PASTE_TRANS.  (See corresponding note there.)
2       !-
2           IF .PP [PP_W_ROWS_TO_MOVE] EQL 0  OR
2              .PP [PP_W_MOVE_LENGTH]  EQL 0
2           THEN
2               RETURN (SS$_NORMAL);
2
2       !+
2       ! Compute the row and column numbers where the borders fall.  Note these
2       ! rows and columns may not map into buffer and need to be validated
2       ! before use.
2       !-
2           UPPER_ROW = .PP [PP_W_ROW] - 1;
2           LOWER_ROW = MAX ( .PP [PP_W_ROW], 1) + .PP [PP_W_ROWS_TO_MOVE];
2           LEFT_COL  = .PP [PP_W_COL] - 1;
2           RIGHT_COL = MAX ( .PP [PP_W_COL], 1) + .PP [PP_W_MOVE_LENGTH];
2
2       !+
2       ! If the line above where the virtual display is pasted exists in the
2       ! window buffer, draw the the top border.
2       !-
2           IF .UPPER_ROW GEQ 1
2           THEN
3               BEGIN   ! Top border
3
3                   !+
3                   ! If WCB is special, blank it and force line to normal.
3                   !-
3
3                   IF (.WCB_LCV[.UPPER_ROW] NEQ 0) AND .PBCB[PBCB_V_WIDE]
3                   THEN
4                       BEGIN   ! Force line normal
4                       CH$FILL(%C' ',.WCB[WCB_W_NO_COLS],
4                               PTRT[ (.UPPER_ROW-1)*.WCB[WCB_W_NO_COLS] ]);
4                       CH$FILL(%C' ',.WCB[WCB_W_NO_COLS],
4                               PTRA[ (.UPPER_ROW-1)*.WCB[WCB_W_NO_COLS] ]);
4                       WCB_LCV[.UPPER_ROW]=0
3                       END;    ! Force line normal
3
3               SPOS = .PP [PP_W_TO_INDEX] - .WCB [WCB_W_NO_COLS] ;
3               INCR I FROM 1 TO .PP [PP_W_MOVE_LENGTH]
3               DO
```

```
                BEGIN        ! Top line loop
                $MARKIT (SPOS, BORD_M_HORIZ );
                (PTRT [.SPOS])<BORD_V_DOWN,1> = 0; ! Knock down ''down'' seg.
                SPOS = .SPOS + 1;
                END;         ! Top line loop

            IF .UPPER_ROW LSS .PBCB [PBCB_W_FIRST_CHANGED_ROW]
            THEN
                PBCB [PBCB_W_FIRST_CHANGED_ROW] = .UPPER_ROW;

            END;     ! Top border

!+
! If the line below where the virtual display is pasted exists in the
! window buffer, draw the the bottom border.
!-
        IF .LOWER_ROW LEQ .WCB [WCB_W_NO_ROWS]
        THEN
            BEGIN   ! Bottom border

                !+
                ! If WCB is special, blank it and force line to normal.
                !-

                IF (.WCB_LCV[.LOWER_ROW] NEQ 0) AND .PBCB[PBCB_V_WIDE]
                THEN
                    BEGIN   ! Force line normal
                    CH$FILL(%C' ',.WCB[WCB_W_NO_COLS],
                            PTRT[ (.LOWER_ROW-1)*.WCB[WCB_W_NO_COLS] ]);
                    CH$FILL(%C' ',.WCB[WCB_W_NO_COLS],
                            PTRA[ (.LOWER_ROW-1)*.WCB[WCB_W_NO_COLS] ]);
                    WCB_LCV[.LOWER_ROW]=0
                    END;    ! Force line normal

            SPOS = .PP [PP_W_TO_INDEX] + (.PP  [PP_W_ROWS_TO_MOVE] *
                                            .WCB [WCB_W_NO_COLS] ) ;
            INCR I FROM 1 TO .PP [PP_W_MOVE_LENGTH]
            DO
                BEGIN        ! Bottom line loop
                $MARKIT (SPOS, BORD_M_HORIZ );
                (PTRT [.SPOS])<BORD_V_UP,1> = 0;   ! Knock down ''up'' seg.
                SPOS = .SPOS + 1;
                END;         ! Bottom line loop

            IF .LOWER_ROW GTR .PBCB [PBCB_W_LAST_CHANGED_ROW]
            THEN
                PBCB [PBCB_W_LAST_CHANGED_ROW] = .LOWER_ROW;

            END;     ! Bottom border

!+
! If the column to the left of where the virtual display is pasted
! exists in the window buffer, draw the left border.
!-
        IF .LEFT_COL GEQ 1
        THEN
            BEGIN   ! Left border
```

```
        LOCAL
            WLN,            ! Line number in WCB buffers
            LCV : REF VECTOR [,BYTE]; ! Addr of line characteristics
                                      ! vector in WCB
        WLN = .PP [PP_W_FIRST_WCB_ROW];
        LCV = .WCB [WCB_A_LINE_CHAR];
        SPOS = .PP [PP_W_TO_INDEX] - 1;
        INCR I FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
        DO
            BEGIN           ! Left column loop
            IF .LCV [.WLN] EQL 0 OR                 ! If normal or
               (NOT .PBCB [PBCB_V_WIDE])            ! DWDH not supported
            THEN
                BEGIN   ! Normal case
                $MARKIT (SPOS, BORD_M_VERT ) ;
                (PTRT [.SPOS])<BORD_V_RIGHT,1> = 0; ! Knock down "right" seg.
                END     ! Normal case
            ELSE
                BEGIN   ! Double-wide or Double-wide/Double-high case
                LOCAL
                    SINDEX;     ! Special index for DWDH
                $CALC_LEFT_INDEX ;
                $MARKIT (SINDEX, BORD_M_VERT);
                (PTRT [.SINDEX])<BORD_V_RIGHT,1> = 0; ! Delete "right"
                END;    ! Double-wide or Double-wide/Double-high case
            SPOS = .SPOS + .WCB [WCB_W_NO_COLS] ;
            WLN = .WLN + 1;
            END;            ! Left column loop

        IF .LEFT_COL LSS .PBCB [PBCB_W_FIRST_CHANGED_COL]
        THEN
            PBCB [PBCB_W_FIRST_CHANGED_COL] = .LEFT_COL;

        END;    ! Left border

!+
! If the column to the right of where the virtual display is pasted
! exists in the window buffer, draw the right border.
!-
    IF .RIGHT_COL LEQ .WCB [WCB_W_NO_COLS]
    THEN
        BEGIN   ! Right border
        LOCAL
            WLN,            ! Line number in WCB buffers
            LCV : REF VECTOR [,BYTE]; ! Addr of line characteristics
                                      ! vector in WCB
        WLN = .PP [PP_W_FIRST_WCB_ROW];
        LCV = .WCB [WCB_A_LINE_CHAR];
        SPOS = .PP [PP_W_TO_INDEX] + .PP [PP_W_MOVE_LENGTH];
        INCR I FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
        DO
            BEGIN           ! Right column loop
            IF .LCV [.WLN] EQL 0 OR                 ! If normal or
               (NOT .PBCB [PBCB_V_WIDE])            ! DWDH not supported
            THEN
                BEGIN   ! Normal case
                $MARKIT (SPOS, BORD_M_VERT ) ;
```

```
                (PTRT [.SPOS])<BORD_V_LEFT,1> = 0; ! Knock down "left" seg.
                END     ! Normal case
            ELSE
                BEGIN   ! Double-wide or Double-wide/Double-high case
                LOCAL
                    SINDEX;     ! Special index for DWDH
                $CALC_RIGHT_INDEX ;
                $MARKIT (SINDEX, BORD_M_VERT);
                (PTRT [.SINDEX])<BORD_V_LEFT,1> = 0; ! Delete "left"
                END;    ! Double-wide or Double-wide/Double-high case
            SPOS = .SPOS + .WCB [WCB_W_NO_COLS] ;
            WLN = .WLN + 1;
            END;        ! Right column loop

        IF .RIGHT_COL GTR .PBCB [PBCB_W_LAST_CHANGED_COL]
        THEN
            PBCB [PBCB_W_LAST_CHANGED_COL] = .RIGHT_COL;

        END;    ! Right border

!+
! Handle upper left-hand corner if position exists in buffer.
!-
    IF .UPPER_ROW GEQ 1         AND     ! Row exists
       .LEFT_COL  GEQ 1                 ! Col exists
    THEN
        BEGIN   ! Upper left-hand corner
        SPOS = .PP [PP_W_TO_INDEX] - .WCB [WCB_W_NO_COLS] - 1;
        $MARKIT (SPOS, BORD_M_ULCORN ) ;
        END;    ! Upper left-hand corner

!+
! Handle upper right-hand corner if position exists in buffer.
!-
    IF .UPPER_ROW GEQ 1         AND             ! Row exists
       .RIGHT_COL LEQ .WCB [WCB_W_NO_COLS]      ! Col exists
    THEN
        BEGIN   ! Upper right-hand corner
        SPOS = .PP [PP_W_TO_INDEX] - .WCB [WCB_W_NO_COLS] +
                    .PP [PP_W_MOVE_LENGTH] ;
        $MARKIT (SPOS, BORD_M_URCORN ) ;
        END;    ! Upper right-hand corner

!+
! Handle lower left-hand corner if position exists in buffer.
!-
    IF .LOWER_ROW LEQ .WCB [WCB_W_NO_ROWS]  AND ! Row exists
       .LEFT_COL  GEQ 1                         ! Col exists
    THEN
        BEGIN   ! Lower left-hand corner
        SPOS = .PP [PP_W_TO_INDEX] + (.PP [PP_W_ROWS_TO_MOVE] *
                                .WCB [WCB_W_NO_COLS]) - 1;
        $MARKIT (SPOS, BORD_M_LLCORN ) ;
        END;    ! Lower left-hand corner

!+
! Handle lower right-hand corner if position exists in buffer.
```

```
!-
    IF .LOWER_ROW LEQ .WCB [WCB_W_NO_ROWS]  AND ! Row exists
       .RIGHT_COL LEQ .WCB [WCB_W_NO_COLS]      ! Col exists
    THEN
        BEGIN   ! Lower right-hand corner
        SPOS = .PP [PP_W_TO_INDEX] +
                (.PP [PP_W_ROWS_TO_MOVE] * .WCB [WCB_W_NO_COLS]) +
                .PP [PP_Q_MOVE_LENGTH] ;
        $MARKIT (SPOS, BORD_M_LRCORN ) ;
        END;    ! Lower right-hand corner

!+
! Check to see if this border is labeled and if so, move the label text
! into the approriate border area.
! All the fields in the PP accessed below have been previously computed
! at the time the virtual display was pasted to the pasteboard so
! that they need not be recomputed each time through here.  The lengths
! used in the various CH$MOVE's and CH$FILL's have been so computed
! that if no parts of the label fit, the lengths used are zero.
!-
    LDES = DCB [DCB_Q_LABEL_DESC];
    IF .LDES [DSC$W_LENGTH] NEQ 0                   ! If label exists...
    THEN
        BEGIN   ! Overall border label processing
        CASE .DCB [DCB_B_LABEL_POS] FROM SMG$K_TOP TO SMG$K_RIGHT OF
        SET
            [SMG$K_TOP]:
                BEGIN    ! Label in top row
                !+
                ! If row where label lands is not on the pasteboard, no
                ! movement will be necessary.
                !-
                IF .UPPER_ROW GEQ 1     ! If row above present...
                THEN
                    BEGIN       ! Top present

                    CH$MOVE (
                        .PP [PP_W_LABEL_BYTES_TO_MOVE],  ! Length
                        .LDES [DSC$A_POINTER] + .PP [PP_W_SRC_LABEL_OFF],  ! Source
                        PTRT [.PP [PP_W_DST_LABEL_OFF]]);  ! Dest

                    CH$FILL (
                        .DCB [DCB_B_LABEL_REND],          ! Fill
                        .PP [PP_W_LABEL_BYTES_TO_MOVE],   ! Number
                        PTRA [.PP [PP_W_DST_LABEL_OFF]]);  ! Dest

                    END;        ! Top present
                END;     ! Label in top row

            [SMG$K_BOTTOM]:
                BEGIN    ! Label in bottom row
                !+
                ! If row where label lands is not on the pasteboard, no
                ! movement will be necessary.
                !-
                IF .LOWER_ROW LEQ .WCB [WCB_W_NO_ROWS] ! If row below present...
                THEN
```

```
                    BEGIN           ! Bottom present

                    CH$MOVE (
                        .PP [PP_W_LABEL_BYTES_TO_MOVE],   ! Length
                        .LDES [DSC$A_POINTER] + .PP [PP_W_SRC_LABEL_OFF],   ! Source
                        PTRT [.PP [PP_W_DST_LABEL_OFF]]);   ! Dest

                    CH$FILL (
                        .DCB [DCB_B_LABEL_REND],          ! Fill
                        .PP [PP_W_LABEL_BYTES_TO_MOVE],   ! Number
                        PTRA [.PP [PP_W_DST_LABEL_OFF]]);  ! Dest

                    END;            ! Bottom present
                END;    ! Label in bottom row

            [SMG$K_LEFT]:
                BEGIN   ! Label in left column
                !+
                ! If column where label lands is not on the pasteboard,
                ! no movement will be necessary.
                !-
                IF .LEFT_COL GEQ 1          ! If column to left present...
                THEN
                    BEGIN           ! Left present
                    LOCAL
                        WLN,                        ! Line # in WCB buffer
                        LCV : REF VECTOR [,BYTE],   ! Addr of line
                                                    ! characteristics vect.
                                                    ! in WCB
                        I;          ! Local index
                    WLN = (.PP [PP_W_DST_LABEL_OFF] /
                          .WCB [WCB_W_NO_COLS]) + 1;
                    LCV = .WCB [WCB_A_LINE_CHAR];
                    SPOS = .PP [PP_W_DST_LABEL_OFF];
                    I = 0;
                    UNTIL .I EQL .PP [PP_W_LABEL_BYTES_TO_MOVE]
                    DO
                        BEGIN   ! Character by character movement
                        IF .LCV [.WLN] EQL 0 OR            ! If normal or
                           (NOT .PBCB [PBCB_V_WIDE])       ! DWDH not supported
                        THEN
                            BEGIN ! Normal case
                            PTRT [.SPOS] = .(.LDES [DSC$A_POINTER] +
                                        .PP [PP_W_SRC_LABEL_OFF] + .I);
                            PTRA [.SPOS] = .DCB [DCB_B_LABEL_REND];
                            END    ! Normal case
                        ELSE
                            BEGIN ! Special case
                            LOCAL
                                SINDEX;
                            $CALC_LEFT_INDEX ;
                            PTRT [.SINDEX] = .(.LDES [DSC$A_POINTER] +
                                        .PP [PP_W_SRC_LABEL_OFF] + .I);
                            PTRA [.SINDEX] = .DCB [DCB_B_LABEL_REND];
                            END;  ! Special case

                        SPOS = .SPOS + .WCB [WCB_W_NO_COLS];
```

```
6                       I = .I+1;
6                       WLN = .WLN + 1;
5                       END;        ! Character by character movement
4                   END;            ! Left present
3               END;    ! Label in left column
3
3           [SMG$K_RIGHT]:
4               BEGIN   ! Label in right column
4               !+
4               ! If column where label lands is not on the pasteboard,
4               ! no movement will be necessary.
4               !-
4               IF .RIGHT_COL LEQ .WCB [WCB_W_NO_COLS] ! If column to right present...
4               THEN
5                   BEGIN       ! Right present
5                   LOCAL
5                       WLN,                        ! Line # in WCB buffer
5                       LCV : REF VECTOR [,BYTE],   ! Addr of line
5                                                   ! characteristics vect.
5                                                   ! in WCB
5                       I;      ! Local index
5                   I = 0;
6                   WLN = (.PP [PP_W_DST_LABEL_OFF] /
5                           .WCB [WCB_W_NO_COLS]) + 1;
5                   LCV = .WCB [WCB_A_LINE_CHAR];
5                   SPOS = .PP [PP_W_DST_LABEL_OFF];
5                   UNTIL .I EQL .PP [PP_W_LABEL_BYTES_TO_MOVE]
5                   DO
6                       BEGIN   ! Character by character movement
6                       IF .LCV [.WLN] EQL 0 OR             ! If normal or
7                          (NOT .PBCB [PBCB_V_WIDE])        ! DWDH not supported
6                       THEN
7                           BEGIN       ! Normal case
8                           PTRT [.SPOS] = .(.LDES [DSC$A_POINTER] +
7                                   .PP [PP_W_SRC_LABEL_OFF] + .I);
7                           PTRA [.SPOS] = .DCB [DCB_B_LABEL_REND];
7                           END         ! Normal case
6                       ELSE
7                           BEGIN       ! Special case
7                           LOCAL
7                               SINDEX;
7                           $CALC_RIGHT_INDEX ;
8                           PTRT [.SINDEX] = .(.LDES [DSC$A_POINTER] +
7                                   .PP [PP_W_SRC_LABEL_OFF] + .I);
7                           PTRA [.SINDEX] = .DCB [DCB_B_LABEL_REND];
6                           END;        ! Special case
6                       SPOS = .SPOS + .WCB [WCB_W_NO_COLS];
6                       I = .I+1;
6                       WLN = .WLN + 1;
5                       END;        ! Character by character movement
4                   END;            ! Right present
3               END;    ! Label in right column
3
3           [OUTRANGE]:
3               RETURN (SMG$_FATERRLIB);
3       TES;
3
```

```
; 2542          2601  2         END;        ! Overall border label processing
; 2543          2602  2      RETURN (SS$_NORMAL);
; 2544          2603  1      END;                    ! End of routine SMG$$DRAW_BORDER

                                          0FFC 00000        .ENTRY   SMG$$DRAW_BORDER, Save R2,R3,R4,R5,R6,R7,-      ; 2089
                                                                     R8,R9,R10,R11                                   ;
                           5E               28  C2 00002        SUBL2    #40, SP                                         ;
              50     0C    AC               08  C1 00005        ADDL3    #8, WCB, R0                                     ; 2206
                           59               60  D0 0000A        MOVL     (R0), PTRT                                      ;
              50     0C    AC               0C  C1 0000D        ADDL3    #12, WCB, R0                                    ; 2207
                           5A               60  D0 00012        MOVL     (R0), PTRA                                      ;
                           5B        08     AC  D0 00015        MOVL     PP, R11                                         ; 2208
                     10    AE        14     AB  D0 00019        MOVL     20(R11), PBCB                                   ;
              50     0C    AC               2C  C1 0001E        ADDL3    #44, WCB, R0                                    ; 2209
                     0C    AE               60  D0 00023        MOVL     (R0), 12(SP)                                    ;
                     08    AE        0C     AE  D0 00027        MOVL     12(SP), WCB_LCV                                 ;
                     04    AE        1C     AB  3C 0002C        MOVZWL   28(R11), 4(SP)                                  ; 2219
                                            03  13 00031        BEQL     1$                                              ;
                                     22     AB  B5 00033        TSTW     34(R11)                                         ; 2220
                                            03  12 00036 1$:    BNEQ     2$                                              ;
                                          04C0  31 00038        BRW      64$                                             ;
                           57        18     AB  32 0003B 2$:    CVTWL    24(R11), UPPER_ROW                              ; 2229
                                            57  D7 0003F        DECL     UPPER_ROW                                       ;
                           52        18     AB  32 00041        CVTWL    24(R11), R2                                     ; 2230
                                            03  14 00045        BGTR     3$                                              ;
                           52               01  D0 00047        MOVL     #1, R2                                          ;
              58           52        04     AE  C1 0004A 3$:    ADDL3    4(SP), R2, LOWER_ROW                            ;
                     18    AE        1A     AB  32 0004F        CVTWL    26(R11), LEFT_COL                               ; 2231
                                     18     AE  D7 00054        DECL     LEFT_COL                                        ;
                           52        1A     AB  32 00057        CVTWL    26(R11), R2                                     ; 2232
                                            03  14 0005B        BGTR     4$                                              ;
                           52               01  D0 0005D        MOVL     #1, R2                                          ;
                     14    AE        22     AB  3C 00060 4$:    MOVZWL   34(R11), 20(SP)                                 ;
                     1C    AE        14   BE42  9E 00065        MOVAB    @20(SP)[R2], RIGHT_COL                          ;
                                     24     AE  D4 0006B        CLRL     36(SP)                                          ; 2238
                                            57  D5 0006E        TSTL     UPPER_ROW                                       ;
                                            03  14 00070        BGTR     5$                                              ;
                                          0097  31 00072        BRW      11$                                             ;
                                     24     AE  D6 00075 5$:    INCL     36(SP)                                          ;
                           50        08     AE  D0 00078        MOVL     WCB_LCV, R0                                     ; 2246
                                          6740  95 0007C        TSTB     (UPPER_ROW)[R0]                                 ;
                                            3E  13 0007F        BEQL     6$                                              ;
              51     10    AE  000000FA     8F  C1 00081        ADDL3    #250, PBCB, R1                                  ;
                           32               61  E9 0008A        BLBC     (R1), 6$                                        ;
                           56        FF     A7  9E 0008D        MOVAB    -1(R7), R6                                      ; 2250
              51     0C    AC               06  C1 00091        ADDL3    #6, WCB, R1                                     ;
                           50               61  3C 00096        MOVZWL   (R1), R0                                        ;
                           56               50  C4 00099        MULL2    R0, R6                                          ;
         20   AE     0C    AC               06  C1 0009C        ADDL3    #6, WCB, 32(SP)                                 ;
 20  BE       20           6E               00  2C 000A2        MOVC5    #0, (SP), #32, @32(SP), (R6)[PTRT]              ;
                                          6649     000A8                                                                 ;
         20   AE     0C    AC               06  C1 000AA        ADDL3    #6, WCB, 32(SP)                                 ; 2252
 20  BE       20           6E               00  2C 000B0        MOVC5    #0, (SP), #32, @32(SP), (R6)[PTRA]              ;
```

```
                                    664A       000B6                                                   ;
                      50        08    AE    D0 000B8         MOVL    WCB_LCV, R0                       ; 2253
                                    6740    94 000BC         CLRB    (UPPER_ROW)[R0]                   ;
                      50        20    AB    3C 000BF  6$:    MOVZWL  32(R11), R0                       ; 2256
            51    0C  AC              06    C1 000C3         ADDL3   #6, WCB, R1                       ;
                      56              61    3C 000C8         MOVZWL  (R1), SPOS                        ;
            56        50              56    C3 000CB         SUBL3   SPOS, R0, SPOS                    ;
                                      50    D4 000CF         CLRL    I                                 ; 2260
                                      18    11 000D1         BRB     10$                               ;
                                    664A    95 000D3  7$:    TSTB    (SPOS)[PTRA]                      ;
                                      06    18 000D6         BGEQ    8$                                ;
                    6649              05    88 000D8         BISB2   #5, (SPOS)[PTRT]                  ;
                                      09    11 000DC         BRB     9$                                ;
                    6649              05    90 000DE  8$:    MOVB    #5, (SPOS)[PTRT]                  ;
                    664A        80    8F    90 000E2         MOVB    #-128, (SPOS)[PTRA]               ;
                    8649              08    8A 000E7  9$:    BICB2   #8, (SPOS)+[PTRT]                 ; 2261
            E3        50        14    AE    F3 000EB  10$:   AOBLEQ  20(SP), I, 7$                     ; 2257
            50        10    AE 000000A8    8F    C1 000F0         ADDL3   #168, PBCB, R0                    ; 2265
      57    60        10              00    EC 000F9         CMPV    #0, #16, (R0), UPPER_ROW          ;
                                      0C    15 000FE         BLEQ    11$                               ;
            50        10    AE 000000A8    8F    C1 00100         ADDL3   #168, PBCB, R0                    ; 2267
                      60              57    B0 00109         MOVW    UPPER_ROW, (R0)                   ;
                                20    AE    D4 0010C  11$:   CLRL    32(SP)                            ; 2275
            50    0C  AC              02    C1 0010F         ADDL3   #2, WCB, R0                       ;
      58    60        10              00    ED 00114         CMPZV   #0, #16, (R0), LOWER_ROW          ;
                                      03    18 00119         BGEQ    12$                               ;
                                    0098    31 0011B         BRW     18$                               ;
                                20    AE    D6 0011E  12$:   INCL    32(SP)                            ;
                      50        08    AE    D0 00121         MOVL    WCB_LCV, R0                       ; 2283
                                    6840    95 00125         TSTB    (LOWER_ROW)[R0]                   ;
                                      3C    13 00128         BEQL    13$                               ;
            51        10    AE 000000FA    8F    C1 0012A         ADDL3   #250, PBCB, R1                    ;
                      30              61    E9 00133         BLBC    (R1), 13$                         ;
                      57        FF    A8    9E 00136         MOVAB   -1(R8), R7                        ; 2287
            51    0C  AC              06    C1 0013A         ADDL3   #6, WCB, R1                       ;
                      50              61    3C 0013F         MOVZWL  (R1), R0                          ;
                      57              50    C4 00142         MULL2   R0, R7                            ;
            6E    0C  AC              06    C1 00145         ADDL3   #6, WCB, (SP)                     ;
00    BE    20        6E              00    2C 0014A         MOVC5   #0, (SP), #32, @0(SP), (R7)[PTRT] ;
                                    6749       00150                                                   ;
            6E    0C  AC              06    C1 00152         ADDL3   #6, WCB, (SP)                     ; 2289
00    BE    20        6E              00    2C 00157         MOVC5   #0, (SP), #32, @0(SP), (R7)[PTRA] ;
                                    674A       0015D                                                   ;
                      50        08    AE    D0 0015F         MOVL    WCB_LCV, R0                       ; 2290
                                    6840    94 00163         CLRB    (LOWER_ROW)[R0]                   ;
            50    0C  AC              06    C1 00166  13$:   ADDL3   #6, WCB, R0                       ; 2294
                      51              60    3C 0016B         MOVZWL  (R0), R1                          ;
                      51        04    AE    C4 0016E         MULL2   4(SP), R1                         ;
                      56        20    AB    3C 00172         MOVZWL  32(R11), SPOS                     ; 2293
                      56              51    C0 00176         ADDL2   R1, SPOS                          ;
                                      50    D4 00179         CLRL    I                                 ; 2298
                                      18    11 0017B         BRB     17$                               ;
                                    664A    95 0017D  14$:   TSTB    (SPOS)[PTRA]                      ;
                                      06    18 00180         BGEQ    15$                               ;
                    6649              05    88 00182         BISB2   #5, (SPOS)[PTRT]                  ;
                                      09    11 00186         BRB     16$                               ;
                    6649              05    90 00188  15$:   MOVB    #5, (SPOS)[PTRT]                  ;
```

```
                         664A      80      8F   90 0018C        MOVB    #-128, (SPOS)[PTRA]                  ;
                         8649              02   8A 00191 16$:   BICB2   #2, (SPOS)+[PTRT]                    ; 2299
                   E3      50      14      AE   F3 00195 17$:   AOBLEQ  20(SP), I, 14$                       ; 2295
                           50  10  AE 000000AA  8F   C1 0019A   ADDL3   #170, PBCB, R0                       ; 2303
           58              60      10           00   EC 001A3   CMPV    #0, #16, (R0), LOWER_ROW             ;
                                                0C   18 001A8   BGEQ    18$                                  ;
                           50  10  AE 000000AA  8F   C1 001AA   ADDL3   #170, PBCB, R0                       ; 2305
                                   60           58   B0 001B3   MOVW    LOWER_ROW, (R0)                      ;
                                                55   D4 001B6 18$:  CLRL    R5                               ; 2313
                                   18      AE   D5 001B8        TSTL    LEFT_COL                             ;
                                                03   14 001BB   BGTR    19$                                  ;
                                              00A2   31 001BD   BRW     29$                                  ;
                                                55   D6 001C0 19$:  INCL    R5                               ;
                                   50      2F      AB   3C 001C2    MOVZWL  47(R11), WLN                 ; 2320
                                   53      0C      AE   D0 001C6    MOVL    12(SP), LCV                  ; 2321
                                   56      20      AB   3C 001CA    MOVZWL  32(R11), SPOS                ; 2322
                                                56   D7 001CE   DECL    SPOS                                 ;
                                                54   D4 001D0   CLRL    I                                    ; 2341
                                                6B   11 001D2   BRB     28$                                  ;
                                              6043   95 001D4 20$:  TSTB    (WLN)[LCV]                       ; 2326
                                                0C   13 001D7   BEQL    21$                                  ;
                           51  10  AE 000000FA  8F   C1 001D9   ADDL3   #250, PBCB, R1                       ; 2327
                                   1A           61   E8 001E2   BLBS    (R1), 24$                            ;
                                              664A   95 001E5 21$:  TSTB    (SPOS)[PTRA]                     ; 2330
                                                06   18 001E8   BGEQ    22$                                  ;
                         6649                   0A   88 001EA   BISB2   #10, (SPOS)[PTRT]                    ;
                                                09   11 001EE   BRB     23$                                  ;
                         6649                   0A   90 001F0 22$:  MOVB    #10, (SPOS)[PTRT]                ;
                         664A      80           8F   90 001F4   MOVB    #-128, (SPOS)[PTRA]                  ;
                         6649                   01   8A 001F9 23$:  BICB2   #1, (SPOS)[PTRT]                 ; 2331
                                                33   11 001FD   BRB     27$                                  ; 2326
                           51      FF      A0   9E 001FF 24$:   MOVAB   -1(R0), R1                           ; 2336
                   57  0C      AC           06   C1 00203       ADDL3   #6, WCB, R7                          ;
                           52                   67   3C 00208   MOVZWL  (R7), R2                             ;
                           51                   52   C4 0020B   MULL2   R2, R1                               ;
                           52      33      AB   3C 0020E        MOVZWL  51(R11), R2                          ;
                           52                   02   C6 00212   DIVL2   #2, R2                               ;
                           51      FF A241      9E 00215        MOVAB   -1(R2)[R1], SINDEX                   ;
                                              614A   95 0021A   TSTB    (SINDEX)[PTRA]                       ; 2338
                                                06   18 0021D   BGEQ    25$                                  ;
                         6149                   0A   88 0021F   BISB2   #10, (SINDEX)[PTRT]                  ;
                                                09   11 00223   BRB     26$                                  ;
                         6149                   0A   90 00225 25$:  MOVB    #10, (SINDEX)[PTRT]              ;
                         614A      80           8F   90 00229   MOVB    #-128, (SINDEX)[PTRA]                ;
                         6149                   01   8A 0022E 26$:  BICB2   #1, (SINDEX)[PTRT]               ; 2339
                   52  0C      AC           06   C1 00232 27$:  ADDL3   #6, WCB, R2                          ; 2341
                           51                   62   3C 00237   MOVZWL  (R2), R1                             ;
                           56                   51   C0 0023A   ADDL2   R1, SPOS                             ;
                                                50   D6 0023D   INCL    WLN                                  ; 2342
                   90      54      04      AE   F3 0023F 28$:   AOBLEQ  4(SP), I, 20$                        ; 2323
                           50  10  AE 000000AC  8F   C1 00244   ADDL3   #172, PBCB, R0                       ; 2345
   18      AE              60      10           00   EC 0024D   CMPV    #0, #16, (R0), LEFT_COL              ;
                                                0D   15 00253   BLEQ    29$                                  ;
                           50  10  AE 000000AC  8F   C1 00255   ADDL3   #172, PBCB, R0                       ; 2347
                                   60      18   AE   B0 0025E   MOVW    LEFT_COL, (R0)                       ;
                   50  0C      AC           06   C1 00262 29$:  ADDL3   #6, WCB, R0                          ; 2355
                           18      AE           60   3C 00267   MOVZWL  (R0), 24(SP)                         ;
```

```
                         52 D4 0026B        CLRL    R2                                 ;
          18  AE     1C  AE D1 0026D        CMPL    RIGHT_COL, 24(SP)                  ;
                         03 15 00272        BLEQ    30$                                ;
                       009E 31 00274        BRW     40$                                ;
                         52 D6 00277 30$:   INCL    R2                                 ;
          50         2F  AB 3C 00279        MOVZWL  47(R11), WLN                       ; 2362
          53         0C  AE D0 0027D        MOVL    12(SP), LCV                        ; 2363
          56         20  AB 3C 00281        MOVZWL  32(R11), SPOS                      ; 2364
          56         14  AE C0 00285        ADDL2   20(SP), SPOS                       ;
                         54 D4 00289        CLRL    I                                  ; 2372
                         65 11 0028B        BRB     39$                                ;
                       6043 95 0028D 31$:   TSTB    (WLN)[LCV]                         ; 2368
                         0C 13 00290        BEQL    32$                                ;
     51   10  AE 000000FA 8F C1 00292       ADDL3   #250, PBCB, R1                     ; 2369
              1A         61 E8 0029B        BLBS    (R1), 35$                          ;
                       664A 95 0029E 32$:   TSTB    (SPOS)[PTRA]                       ; 2372
                         06 18 002A1        BGEQ    33$                                ;
        6649             0A 88 002A3        BISB2   #10, (SPOS)[PTRT]                  ;
                         09 11 002A7        BRB     34$                                ;
        6649             0A 90 002A9 33$:   MOVB    #10, (SPOS)[PTRT]                  ;
        664A         80  8F 90 002AD        MOVB    #-128, (SPOS)[PTRA]                ;
        6649             04 8A 002B2 34$:   BICB2   #4, (SPOS)[PTRT]                   ; 2373
                         34 11 002B6        BRB     38$                                ; 2368
          57         FF  A0 9E 002B8 35$:   MOVAB   -1(R0), R7                         ; 2378
          57         18  AE C4 002BC        MULL2   24(SP), R7                         ;
          51         33  AB 3C 002C0        MOVZWL  51(R11), R1                        ;
     58   14         AE  01 C1 002C4        ADDL3   #1, 20(SP), R8                     ;
          51             58 C0 002C9        ADDL2   R8, R1                             ;
          51             02 C6 002CC        DIVL2   #2, R1                             ;
          51         FF A147 9E 002CF       MOVAB   -1(R1)[R7], SINDEX                 ;
                       614A 95 002D4        TSTB    (SINDEX)[PTRA]                     ; 2380
                         06 18 002D7        BGEQ    36$                                ;
        6149             0A 88 002D9        BISB2   #10, (SINDEX)[PTRT]                ;
                         09 11 002DD        BRB     37$                                ;
        6149             0A 90 002DF 36$:   MOVB    #10, (SINDEX)[PTRT]                ;
        614A         80  8F 90 002E3        MOVB    #-128, (SINDEX)[PTRA]              ;
        6149             04 8A 002E8 37$:   BICB2   #4, (SINDEX)[PTRT]                 ; 2381
          56         18  AE C0 002EC 38$:   ADDL2   24(SP), SPOS                       ; 2383
                         50 D6 002F0        INCL    WLN                                ; 2384
     96   54         04  AE F3 002F2 39$:   AOBLEQ  4(SP), I, 31$                      ; 2365
     50   10  AE 000000AE 8F C1 002F7       ADDL3   #174, PBCB, R0                     ; 2387
1C  AE    60         10  00 EC 00300        CMPV    #0, #16, (R0), RIGHT_COL           ;
                         0D 18 00306        BGEQ    40$                                ;
     50   10  AE 000000AE 8F C1 00308       ADDL3   #174, PBCB, R0                     ; 2389
          60         1C  AE B0 00311        MOVW    RIGHT_COL, (R0)                    ;
          4B         24  AE E9 00315 40$:   BLBC    36(SP), 44$                        ; 2396
          20             55 E9 00319        BLBC    R5, 42$                            ; 2397
          51         20  AB 3C 0031C        MOVZWL  32(R11), R1                        ; 2400
          51         18  AE C2 00320        SUBL2   24(SP), R1                         ;
          56         FF  A1 9E 00324        MOVAB   -1(R1), SPOS                       ;
                       664A 95 00328        TSTB    (SPOS)[PTRA]                       ; 2401
                         06 18 0032B        BGEQ    41$                                ;
        6649             09 88 0032D        BISB2   #9, (SPOS)[PTRT]                   ;
                         09 11 00331        BRB     42$                                ;
        6649             09 90 00333 41$:   MOVB    #9, (SPOS)[PTRT]                   ;
        664A         80  8F 90 00337        MOVB    #-128, (SPOS)[PTRA]                ;
          24         24  AE E9 0033C 42$:   BLBC    36(SP), 44$                        ; 2407
```

```
                          21          52  E9 00340          BLBC    R2, 44$                                ; 2408
                          51     20   AB  3C 00343          MOVZWL  32(R11), R1                            ; 2411
                          51     18   AE  C2 00347          SUBL2   24(SP), R1
            56            51     14   AE  C1 0034B          ADDL3   20(SP), R1, SPOS                       ; 2412
                                    664A  95 00350          TSTB    (SPOS)[PTRA]                           ; 2413
                                      06  18 00353          BGEQ    43$                                    ;
                        6649          0C  88 00355          BISB2   #12, (SPOS)[PTRT]                      ;
                                      09  11 00359          BRB     44$                                    ;
                        6649          0C  90 0035B 43$:     MOVB    #12, (SPOS)[PTRT]                      ;
                        664A     80   8F  90 0035F          MOVB    #-128, (SPOS)[PTRA]                    ;
                          53     20   AE  E9 00364 44$:     BLBC    32(SP), 48$                            ; 2419
                          23          55  E9 00368          BLBC    R5, 46$                                ; 2420
                          50     20   AB  3C 0036B          MOVZWL  32(R11), R0                            ; 2423
            51      04    AE     18   AE  C5 0036F          MULL3   24(SP), 4(SP), R1                      ; 2424
                          56     FF A140  9E 00375          MOVAB   -1(R1)[R0], SPOS                       ;
                                    664A  95 0037A          TSTB    (SPOS)[PTRA]                           ; 2425
                                      06  18 0037D          BGEQ    45$                                    ;
                        6649          03  88 0037F          BISB2   #3, (SPOS)[PTRT]                       ;
                                      09  11 00383          BRB     46$                                    ;
                        6649          03  90 00385 45$:     MOVB    #3, (SPOS)[PTRT]                       ;
                        664A     80   8F  90 00389          MOVB    #-128, (SPOS)[PTRA]                    ;
                          29     20   AE  E9 0038E 46$:     BLBC    32(SP), 48$                            ; 2431
                          26          52  E9 00392          BLBC    R2, 48$                                ; 2432
                          50     20   AB  3C 00395          MOVZWL  32(R11), R0                            ; 2435
            51      04    AE     18   AE  C5 00399          MULL3   24(SP), 4(SP), R1                      ; 2436
                          50          51  C0 0039F          ADDL2   R1, R0                                 ; 2435
            56            50     14   AE  C1 003A2          ADDL3   20(SP), R0, SPOS                       ; 2437
                                    664A  95 003A7          TSTB    (SPOS)[PTRA]                           ; 2438
                                      06  18 003AA          BGEQ    47$                                    ;
                        6649          06  88 003AC          BISB2   #6, (SPOS)[PTRT]                       ;
                                      09  11 003B0          BRB     48$                                    ;
                        6649          06  90 003B2 47$:     MOVB    #6, (SPOS)[PTRT]                       ;
                        664A     80   8F  90 003B6          MOVB    #-128, (SPOS)[PTRA]                    ;
                          58     04   AC  D0 003BB 48$:     MOVL    DCB, R8                                ; 2450
                          57     08   A8  9E 003BF          MOVAB   8(R8), LDES                            ;
                                      67  B5 003C3          TSTW    (LDES)                                 ; 2451
                                      3D  13 003C5          BEQL    53$                                    ;
            03            00     31   A8  8F 003C7          CASEB   49(R8), #0, #3                         ; 2454
    00B1    003B        0016        0010     003CC 49$:     .WORD   50$-49$,-                              ;
                                                                    51$-49$,-                              ;
                                                                    54$-49$,-                              ;
                                                                    59$-49$                                ;
                          50 00000000G    8F  D0 003D4          MOVL    #SMG$_FATERRLIB, R0                    ; 2598
                                          04 003DB          RET                                            ;
                          24     24   AE  E9 003DC 50$:     BLBC    36(SP), 53$                            ;
                                      04  11 003E0          BRB     52$                                    ; 2468
                          1E     20   AE  E9 003E2 51$:     BLBC    32(SP), 53$                            ; 2485
                          51     26   AB  3C 003E6 52$:     MOVZWL  38(R11), R1                            ; 2491
                          51     04   A7  C0 003EA          ADDL2   4(LDES), R1                            ;
                    24    AE     28   AB  3C 003EE          MOVZWL  40(R11), 36(SP)                        ; 2492
         24 BE49          61     24   AB  28 003F3          MOVC3   36(R11), (R1), @36(SP)[PTRT]           ;
  24  AB   33  A8         6E          00  2C 003FA          MOVC5   #0, (SP), 51(R8), 36(R11), @36(SP)[PTRA] ; 2497
                               24 BE4A      00401                                                  ;
                                    00F4  31 00404 53$:     BRW     64$                                    ; 2454
                          FA          55  E9 00407 54$:     BLBC    R5, 53$                                ; 2508
                          52     28   AB  3C 0040A          MOVZWL  40(R11), R2                            ; 2518
                          52     18   AE  C6 0040E          DIVL2   24(SP), R2                             ;
```

```
                        50      01      A2    9E 00412          MOVAB   1(R2), WLN                              ;
                        53      0C      AE    D0 00416          MOVL    12(SP), LCV                             ; 2519
                        56      28      AB    3C 0041A          MOVZWL  40(R11), SPOS                           ; 2520
                                              54 D4 0041E       CLRL    I                                       ; 2521
 54      24      AB     10              00    ED 00420 55$:     CMPZV   #0, #16, 36(R11), I                     ; 2522
                                        DC    13 00426          BEQL    53$                                     ;
                                      6043    95 00428          TSTB    (WLN)[LCV]                              ; 2525
                                        0C    13 0042B          BEQL    56$                                     ;
                 51      10      AE 000000FA  8F C1 0042D       ADDL3   #250, PBCB, R1                          ; 2526
                        14              61    E8 00436          BLBS    (R1), 57$                               ;
                        51      26      AB    3C 00439 56$:     MOVZWL  38(R11), R1                             ; 2530
                        51      04      A7    C0 0043D          ADDL2   4(LDES), R1                             ;
                      6649    6441            90 00441          MOVB    (I)[R1], (SPOS)[PTRT]                   ; 2529
                      664A      33      A8    90 00446          MOVB    51(R8), (SPOS)[PTRA]                    ; 2531
                        26                    11 0044B          BRB     58$                                     ; 2525
                        51      FF      A0    9E 0044D 57$:     MOVAB   -1(R0), R1                              ; 2536
                        51      18      AE    C4 00451          MULL2   24(SP), R1                              ;
                        52      33      AB    3C 00455          MOVZWL  51(R11), R2                             ;
                        52      02            C6 00459          DIVL2   #2, R2                                  ;
                        52      FF    A241    9E 0045C          MOVAB   -1(R2)[R1], SINDEX                      ;
                        51      26      AB    3C 00461          MOVZWL  38(R11), R1                             ; 2539
                        51      04      A7    C0 00465          ADDL2   4(LDES), R1                             ;
                      6249    6441            90 00469          MOVB    (I)[R1], (SINDEX)[PTRT]                 ; 2538
                      624A      33      A8    90 0046E          MOVB    51(R8), (SINDEX)[PTRA]                  ; 2540
                        56      18      AE    C0 00473 58$:     ADDL2   24(SP), SPOS                            ; 2543
                                        54    D6 00477          INCL    I                                       ; 2544
                                        50    D6 00479          INCL    WLN                                     ; 2545
                                        A3    11 0047B          BRB     55$                                     ; 2522
                        7B              52    E9 0047D 59$:     BLBC    R2, 64$                                 ; 2556
                                        54    D4 00480          CLRL    I                                       ; 2565
                        52      28      AB    3C 00482          MOVZWL  40(R11), R2                             ; 2567
                        52      18      AE    C6 00486          DIVL2   24(SP), R2                              ;
                        50      01      A2    9E 0048A          MOVAB   1(R2), WLN                              ;
                        53      0C      AE    D0 0048E          MOVL    12(SP), LCV                             ; 2568
                        56      28      AB    3C 00492          MOVZWL  40(R11), SPOS                           ; 2569
 54      24      AB     10              00    ED 00496 60$:     CMPZV   #0, #16, 36(R11), I                     ; 2570
                                        5D    13 0049C          BEQL    64$                                     ;
                                      6043    95 0049E          TSTB    (WLN)[LCV]                              ; 2573
                                        0C    13 004A1          BEQL    61$                                     ;
                 51      10      AE 000000FA  8F C1 004A3       ADDL3   #250, PBCB, R1                          ; 2574
                        14              61    E8 004AC          BLBS    (R1), 62$                               ;
                        51      26      AB    3C 004AF 61$:     MOVZWL  38(R11), R1                             ; 2578
                        51      04      A7    C0 004B3          ADDL2   4(LDES), R1                             ;
                      6649    6441            90 004B7          MOVB    (I)[R1], (SPOS)[PTRT]                   ; 2577
                      664A      33      A8    90 004BC          MOVB    51(R8), (SPOS)[PTRA]                    ; 2579
                        2E                    11 004C1          BRB     63$                                     ; 2573
                        51      FF      A0    9E 004C3 62$:     MOVAB   -1(R0), R1                              ; 2584
                        51      18      AE    C4 004C7          MULL2   24(SP), R1                              ;
                        52      33      AB    3C 004CB          MOVZWL  51(R11), R2                             ;
                 55      14      AE     01    C1 004CF          ADDL3   #1, 20(SP), R5                          ;
                        52              55    C0 004D4          ADDL2   R5, R2                                  ;
                        52              02    C6 004D7          DIVL2   #2, R2                                  ;
                        52      FF    A241    9E 004DA          MOVAB   -1(R2)[R1], SINDEX                      ;
                        51      26      AB    3C 004DF          MOVZWL  38(R11), R1                             ; 2587
                        51      04      A7    C0 004E3          ADDL2   4(LDES), R1                             ;
                      6249    6441            90 004E7          MOVB    (I)[R1], (SINDEX)[PTRT]                 ; 2586
                      624A      33      A8    90 004EC          MOVB    51(R8), (SINDEX)[PTRA]                  ; 2588
```

```
                    56      18   AE  C0 004F1 63$:    ADDL2   24(SP), SPOS                    ; 2590
                                 54  D6 004F5         INCL    I                               ; 2591
                                 50  D6 004F7         INCL    WLN                             ; 2592
                                 9B  11 004F9         BRB     60$                             ; 2570
                    50           01  D0 004FB 64$:    MOVL    #1, R0                          ; 2602
                                     04 004FE         RET                                     ; 2603

; Routine Size:  1279 bytes,    Routine Base:  _SMG$CODE + 0652


; 2545          2604  1 !<BLF/PAGE>
```

```
%SBTTL 'SMG$$FILL_WINDOW_BUFFER - Fill window with virtual displays'
GLOBAL ROUTINE SMG$$FILL_WINDOW_BUFFER (
                                        PP : REF BLOCK [,BYTE]
                                        ) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This procedure rebuilds the portions of the window buffer that
!       need to be changed because some virtual display has changed.
!       Rebuilding takes place from the given pasting packet down the
!       chain to the most-recently pasted pasting packet.
!
! CALLING SEQUENCE:
!
!       ret_status.wlc.v = SMG$$FILL_WINDOW_BUFFER ( PP.rab.r)
!
! FORMAL PARAMETERS:
!
!
!       PP.rab.r                Address of pasting packet which
!                               determines the first display to be
!                               repainted.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! COMPLETION STATUS:
!
!       SS$_NORMAL      Normal successful completion
!
! SIDE EFFECTS:
!
!       NONE
!--

    BEGIN
    LOCAL
        STATUS,                         ! Status of subr. calls
        PBCB : REF BLOCK [,BYTE],       ! Address of pasteboard control
                                        ! block

        WCB : REF BLOCK [,BYTE],        ! Addr of current WCB

        CURR_PP : REF BLOCK [,BYTE];    ! Addr of 2 longwords that form
                                        ! queue header in PP currently
                                        ! under inspection.

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

!+
! Change packet address to address of queue header.
```

```
2 !-
2       CURR_PP = .PP + PP_PBCB_QUEUE_OFFSET; ! Start with specified packet
2
2 !+
2 ! Loop for all pasting packets starting with this one to the last-pasted
2 ! one...
2 !-
2       WHILE .CURR_PP NEQ PBCB [PBCB_A_PP_NEXT]
2       DO
3           BEGIN   ! For all displays that need to be rewritten
3           LOCAL
3               PP_BASE : REF BLOCK [,BYTE],        ! Base address of the PP
3               DCB : REF BLOCK [,BYTE]; ! Current virtual display that
3                                       ! needs to be repainted.
3
3           PP_BASE = .CURR_PP - PP_PBCB_QUEUE_OFFSET;
3                                   ! Since the queue headers for this part
3                                   ! of the chain are not at relative 0 in
3                                   ! the pasting packet.
3           DCB = .PP_BASE [PP_A_DCB_ADDR]; ! DCB addr of this pairing
3
4           IF NOT (STATUS = SMG$$MOVE_TEXT_TO_WINDOW_BUF (.PP_BASE))
3           THEN
3               RETURN (.STATUS);
3
3           !+
3           ! If bordered, move the border characters into the text buffer
3           ! and set the bit in the attribute bytes to indicate a border
3           ! element.
3           !-
3           IF .DCB [DCB_V_BORDERED]
3           THEN
4               IF NOT (STATUS = SMG$$DRAW_BORDER ( .DCB, .PP_BASE, .WCB))
3               THEN
3                   RETURN (.STATUS);
3
3           !+
3           ! Update changed area fields in the PBCB based on latest
3           ! contribution.
3           !-
3           IF .PP_BASE [PP_W_ROWS_TO_MOVE] NEQ 0
3           THEN
4               BEGIN
4               IF .PP_BASE [PP_W_FIRST_WCB_ROW] LSS .PBCB [PBCB_W_FIRST_CHANGED_ROW]
4               THEN
4                   PBCB [PBCB_W_FIRST_CHANGED_ROW] = .PP_BASE [PP_W_FIRST_WCB_ROW];
4
4               IF .PP_BASE [PP_W_LAST_WCB_ROW] GTR .PBCB [PBCB_W_LAST_CHANGED_ROW]
4               THEN
4                   PBCB [PBCB_W_LAST_CHANGED_ROW] = .PP_BASE [PP_W_LAST_WCB_ROW];
4
4               IF .PP_BASE [PP_W_FIRST_WCB_COL] LSS .PBCB [PBCB_W_FIRST_CHANGED_COL]
4               THEN
4                   PBCB [PBCB_W_FIRST_CHANGED_COL] = .PP_BASE [PP_W_FIRST_WCB_COL];
4
4               IF .PP_BASE [PP_W_LAST_WCB_COL] GTR .PBCB [PBCB_W_LAST_CHANGED_COL]
4               THEN
```

```
4               PBCB [PBCB_W_LAST_CHANGED_COL] = .PP_BASE [PP_W_LAST_WCB_COL];
4
3           END;
3
3       !+
3       ! Walk this chain backwards, from the packet we started with
3       ! back to the head of the chain -- since the most recently
3       ! pasted displays are at the head of the chain.
3       !-
3       CURR_PP = .PP_BASE [PP_A_PREV_PBCB];
2       END;    ! For all displays that need to be rewritten
2
2
2   RETURN ( SMG$$MIN_UPD ( .PBCB ));      ! Cause window buffer to be
2                                          ! output
2
1   END;                    ! End of routine SMG$$FILL_WINDOW_BUFFER


                                     007C 00000        .ENTRY  SMG$$FILL_WINDOW_BUFFER, Save R2,R3,R4,R5,-  ; 2606
                                                               R6                                     ;
                     52       04  AC  D0 00002         MOVL    PP, R2                                 ; 2657
                     54       14  A2  D0 00006         MOVL    20(R2), PBCB                           ;
                     56       08  A4  D0 0000A         MOVL    8(PBCB), WCB                           ; 2658
                     53       08  A2  9E 0000E         MOVAB   8(R2), CURR_PP                         ; 2663
                     54           53  D1 00012 1$:     CMPL    CURR_PP, PBCB                          ; 2669
                                  7B  13 00015         BEQL    7$                                     ;
                     52       F8  A3  9E 00017         MOVAB   -8(R3), PP_BASE                        ; 2677
                     55       10  A2  D0 0001B         MOVL    16(PP_BASE), DCB                       ; 2681
                                  52  DD 0001F         PUSHL   PP_BASE                                ; 2683
               0000V CF           01  FB 00021         CALLS   #1, SMG$$MOVE_TEXT_TO_WINDOW_BUF       ;
                     72           50  E9 00026         BLBC    STATUS, 8$                             ;
                     0E       2F  A5  E9 00029         BLBC    47(DCB), 2$                            ; 2692
                            0044  8F  BB 0002D         PUSHR   #^M<R2,R6>                             ; 2694
                                  55  DD 00031         PUSHL   DCB                                    ;
               FAC9  CF           03  FB 00033         CALLS   #3, SMG$$DRAW_BORDER                   ;
                     60           50  E9 00038         BLBC    STATUS, 8$                             ;
                              1C  A2  B5 0003B 2$:     TSTW    28(PP_BASE)                            ; 2702
                                  4C  13 0003E         BEQL    6$                                     ;
                     51     00A8  C4  32 00040         CVTWL   168(PBCB), R1                          ; 2705
51     2F    A2      10           00  ED 00045         CMPZV   #0, #16, 47(PP_BASE), R1               ;
                     06           18  0004B            BGEQ    3$                                     ;
               00A8  C4       2F  A2  B0 0004D         MOVW    47(PP_BASE), 168(PBCB)                 ; 2707
                     51     00AA  C4  32 00053 3$:     CVTWL   170(PBCB), R1                          ; 2709
51     31    A2      10           00  ED 00058         CMPZV   #0, #16, 49(PP_BASE), R1               ;
                     06           15  0005E            BLEQ    4$                                     ;
               00AA  C4       31  A2  B0 00060         MOVW    49(PP_BASE), 170(PBCB)                 ; 2711
                     51     00AC  C4  32 00066 4$:     CVTWL   172(PBCB), R1                          ; 2713
51     33    A2      10           00  ED 0006B         CMPZV   #0, #16, 51(PP_BASE), R1               ;
                     06           18  00071            BGEQ    5$                                     ;
               00AC  C4       33  A2  B0 00073         MOVW    51(PP_BASE), 172(PBCB)                 ; 2715
                     51     00AE  C4  32 00079 5$:     CVTWL   174(PBCB), R1                          ; 2717
51     35    A2      10           00  ED 0007E         CMPZV   #0, #16, 53(PP_BASE), R1               ;
                     06           15  00084            BLEQ    6$                                     ;
```

```
                00AE   C4       35   A2  B0 00086       MOVW    53(PP_BASE), 174(PBCB)         ; 2719
                       53       0C   A2  D0 0008C 6$:   MOVL    12(PP_BASE), CURR_PP           ; 2728
                                     80  11 00090       BRB     1$                             ; 2669
                                     54  DD 00092 7$:   PUSHL   PBCB                           ; 2732
           00000000G   00            01  FB 00094       CALLS   #1, SMG$$MIN_UPD               ;
                                         04 0009B 8$:   RET                                    ; 2735

; Routine Size:  156 bytes,    Routine Base:  _SMG$CODE + 0B51

; 2678          2736  1 !<BLF/PAGE>
```

SM
1-
;
;

```
1 %SBTTL 'SMG$$MOVE_TEXT_TO_WINDOW_BUF - Move text from display buf. to window buf.'
1 GLOBAL ROUTINE SMG$$MOVE_TEXT_TO_WINDOW_BUF (
1                                       PP : REF $PP_DECL
1                                       ) =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine moves text from the buffer located at
1 !       .DCB [DCB_A_TEXT_BUF] into the window text buffer.
1 !       Array of bytes at .DCB [ DCB_A_ATTR_BUF ] describe the
1 !       rendition this text must assume and is moved into the
1 !       associated window attribute buffer.
1 !       Similarly, if the alternate character set buffer at
1 !       .DCB [DCB_A_CHAR_SET_BUF] exists, it must be mapped into the
1 !       window alternate character set buffer.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       ret_status.wlc.v = SMG$$MOVE_TEXT_TO_WINDOW_BUF ( PP.rab.r)
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       PP.rab.r                Address of pasting packet.
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 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !--
1
2   BEGIN
2
2   BUILTIN
2       SKPC;
2
2   LOCAL
2       DCB  : REF BLOCK [,BYTE],       ! Addr of virtual display
2                                       ! control block.
2       PBCB : REF BLOCK [,BYTE],       ! Addr of pasteboard control
2                                       ! block
2       WCB  : REF BLOCK [,BYTE],       ! Addr of window control block
2       IN_LINE_CHAR  : REF VECTOR [,BYTE],     ! Addr. of line char.
2                                               ! vector off DCB
2       OUT_LINE_CHAR : REF VECTOR [,BYTE],     ! Addr. of line char.
2                                               ! vector off WCB
2
```

```
        WCB_COLS,           ! Extracted .WCB [WCB_W_NO_COLS]
        DCB_COLS,           ! Extracted .DCB [DCB_W_NO_COLS]
        FROM_INDEX,         ! Pointer into DCB buffers
        TO_INDEX,           ! Pointer into WCB buffers for normal mapping
        SPEC_TO_INDEX,      ! Pointer into WCB buffers for special mapping
        RE_BORDER : INITIAL (0);            ! Flag to indicate that the
                                            ! border elements need to be
                                            ! remapped because some line
                                            ! changed from normal to DWDH or
                                            ! vice versa.

MACRO
    $MAP_LINE_NORMAL =
        CH$MOVE ( .PP [PP_W_MOVE_LENGTH],                   ! Length
                  .DCB [DCB_A_TEXT_BUF] + .FROM_INDEX,      ! Source
                  .WCB [WCB_A_TEXT_BUF] + .TO_INDEX);       ! Destination
    %,

    $MAP_LINE_SPECIAL =
        CH$MOVE ( (.PP [PP_W_MOVE_LENGTH]-1)/2,                 ! Length
                  .DCB [DCB_A_TEXT_BUF] + .FROM_INDEX,          ! Source
                  .WCB [WCB_A_TEXT_BUF] + .SPEC_TO_INDEX); ! Destination
    %,

    $MAP_ATTR_NORMAL =
        CH$MOVE ( .PP [PP_W_MOVE_LENGTH],                   ! Length
                  .DCB [DCB_A_ATTR_BUF] + .FROM_INDEX,      ! Source
                  .WCB [WCB_A_ATTR_BUF] + .TO_INDEX);       ! Destination
    %,

    $MAP_ATTR_SPECIAL =
        CH$MOVE ( (.PP [PP_W_MOVE_LENGTH]-1)/2,                 ! Length
                  .DCB [DCB_A_ATTR_BUF] + .FROM_INDEX,          ! Source
                  .WCB [WCB_A_ATTR_BUF] + .SPEC_TO_INDEX); ! Destination
    %,

    $MAP_ALT_CHAR_NORMAL =
        IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
        THEN
        CH$MOVE ( .PP [PP_W_MOVE_LENGTH],                       ! Length
                  .DCB [DCB_A_CHAR_SET_BUF] + .FROM_INDEX,      ! Source
                  .WCB [WCB_A_CHAR_SET_BUF] + .TO_INDEX);       ! Dest.
    %,

    $MAP_ALT_CHAR_SPECIAL =
        IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
        THEN
        CH$MOVE ( (.PP [PP_W_MOVE_LENGTH]-1)/2,                     ! Length
                  .DCB [DCB_A_CHAR_SET_BUF] + .FROM_INDEX,          ! Source
                  .WCB [WCB_A_CHAR_SET_BUF] + .SPEC_TO_INDEX);      ! Dest.
    %,

    $CLEAR_TEXT_LINE =
        CH$FILL ( %C' ',                                    ! Fill char
                  .WCB [WCB_W_NO_COLS],                     ! # of bytes
                  .WCB [WCB_A_TEXT_BUF] + .WLP);            ! Destination
    %,
```

```
	$CLEAR_ATTR_LINE =
	    CH$FILL ( 0,				! Fill char
			.WCB [WCB_W_NO_COLS],		! # of bytes
			.WCB [WCB_A_ATTR_BUF] + .WLP);	! Destination
	%,

	$MAP_ALL_NORMAL =
	    BEGIN
	    $MAP_LINE_NORMAL;
	    $MAP_ATTR_NORMAL;
	    $MAP_ALT_CHAR_NORMAL;
	    END;
	%,

	$CLEAR_ALT_CHAR_LINE =
	    IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
	    THEN
	    CH$FILL ( 0,
			.WCB [WCB_W_NO_COLS],
			.WCB [WCB_A_CHAR_SET_BUF] + .WLP);
	%,

	$MAP_ALL_SPECIAL =
	    BEGIN
	    $MAP_LINE_SPECIAL;
	    $MAP_ATTR_SPECIAL;
	    $MAP_ALT_CHAR_SPECIAL;
	    END;
	%,

	$CLEAR_ALL =
	    BEGIN
	    $CLEAR_TEXT_LINE;
	    $CLEAR_ATTR_LINE;
	    $CLEAR_ALT_CHAR_LINE;
	    RE_BORDER = 1; ! Set flag to rebuild border
	    END;
	%;

	DCB  = .PP [PP_A_DCB_ADDR];
	PBCB = .PP [PP_A_PBCB_ADDR];
	WCB  = .PBCB [PBCB_A_WCB];
	IN_LINE_CHAR = .DCB [DCB_A_LINE_CHAR];
	OUT_LINE_CHAR= .WCB [WCB_A_LINE_CHAR];
	WCB_COLS = .WCB [WCB_W_NO_COLS];
	DCB_COLS = .DCB [DCB_W_NO_COLS];

    !+
    ! Before diverging on two copying paths, check to see if we are going
    ! to get involved with alternate character set buffers.  If one exists
    ! in the DCB but does not yet exist in the WCB, we have to allocate
    ! one for the WCB and initialize it.
    !-
	IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0  AND
	   .WCB [WCB_A_CHAR_SET_BUF] EQL 0
	THEN
```

```
        BEGIN   ! Alloc. and init. window alternate char. set buffer
        LOCAL
            STATUS;     ! Status of LIB$GET_VM call

        IF NOT (STATUS = LIB$GET_VM ( WCB [WCB_L_BUFSIZE],
                                      WCB [WCB_A_CHAR_SET_BUF]))
        THEN
            RETURN (.STATUS);

        CH$FILL ( 0, .WCB [WCB_L_BUFSIZE], .WCB [WCB_A_CHAR_SET_BUF]);
        END;    ! Alloc. and init. window alternate char. set buffer

    !+
    ! At this point we need to decide whether we are dealing with either
    ! a virtual display or a window control block that includes either
    ! Double-Wide (DW) and/or Double-Wide/Double-High (DWDH) lines.
    ! We can tell whether either is non-standard by looking at the zeroth
    ! byte of its line characteristics vector.
    ! If we are dealing with either non-standard, we need to deal with them
    ! on a line by line basis.
    ! For each line to be mapped we implement the logic summarized in the
    ! following table:
    !
    !   DCB Line          WCB Line          Mapping
    !   Char. Vect        Char. Vector      Action
    !   ----------        ------------      ------
    !
    !   Not DW or DWDH    Not DW or DWDH    Map normally
    !   (i.e., normal)    (i.e., Normal)
    !
    !
    !   Not DW or DWDH    DW or DWDH        If DW supported
    !   (i.e., normal)                      then
    !                                           Blank entire WCB text line
    !                                           map normally
    !                                       else
    !                                           map normally
    !
    !
    !   DW or DWDH        Not DW or DWDH    If DW supported
    !                     (i.e., normal)    then
    !                                           Blank entire WCB text line
    !                                           use special mapping
    !                                       else
    !                                           map normally
    !
    !
    !   DW or DWDH        DW or DWDH        If DW supported
    !                                       then
    !                                           Use special mapping
    !                                       else
    !                                           Map normally
    !
    !
    !-
    FROM_INDEX = .PP [PP_W_FROM_INDEX];
    TO_INDEX   = .PP [PP_W_TO_INDEX];
```

```
    IF .IN_LINE_CHAR  [0] EQL 0 AND        ! DCB normal
       .OUT_LINE_CHAR [0] EQL 0            ! WCB normal
    THEN
        BEGIN   ! Both are normal -- no special action needed
        !+
        ! Check to see if we can do it with a single CH$MOVE or whether
        ! we must do it a row at a time.
        !-
        IF .PP [PP_V_CONTIG]
        THEN
            BEGIN       ! Can be done in single move

            ! Move text
            CH$MOVE ( .PP [PP_L_MOVE_SIZE],                 ! length
                  .DCB [DCB_A_TEXT_BUF] + .FROM_INDEX,  ! source
                  .WCB [WCB_A_TEXT_BUF] + .TO_INDEX);   ! dest.

            ! Move attributes
            CH$MOVE ( .PP [PP_L_MOVE_SIZE],                 ! length
                  .DCB [DCB_A_ATTR_BUF] + .FROM_INDEX,  ! source
                  .WCB [WCB_A_ATTR_BUF] + .TO_INDEX);   ! dest.

            ! Move alternate character set buffer pieces, if necessary
            IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
            THEN
                BEGIN   ! Map alternate character set
                CH$MOVE ( .PP [PP_L_MOVE_SIZE],                      ! Length
                      .DCB [DCB_A_CHAR_SET_BUF] + .FROM_INDEX, ! Source
                      .WCB [WCB_A_CHAR_SET_BUF] + .TO_INDEX);  ! Dest.
                END;    ! Map alternate character set
            END         ! Can be done in single move

        ELSE

            BEGIN       ! Must be done row at a time

            INCR R FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
            DO
                BEGIN   ! For all rows in this display
                $MAP_LINE_NORMAL;
                $MAP_ATTR_NORMAL;

                FROM_INDEX = .FROM_INDEX + .DCB_COLS;
                TO_INDEX   = .TO_INDEX   + .WCB_COLS;
                END;    ! For all rows in this display

            !+
            ! Move alternate character set buffer pieces, if necessary
            !
            IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
            THEN
                BEGIN   ! Map alternate character set buffer
                FROM_INDEX = .PP [PP_W_FROM_INDEX];
                TO_INDEX   = .PP [PP_W_TO_INDEX];

                INCR R FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
                DO
```

```
                        BEGIN
                        $MAP_ALT_CHAR_NORMAL;

                        FROM_INDEX = .FROM_INDEX + .DCB_COLS;
                        TO_INDEX   = .TO_INDEX   + .WCB_COLS;
                        END;

                    END;        ! Map alternate character set buffer

                END;            ! Must be done row at a time
            END     ! Both are normal -- no special action needed

        ELSE
            BEGIN   ! One or the other contains special characteristics
            LOCAL
                DLN,            ! Virtual display line number
                WLN,            ! WCB line number
                WLP;            ! Byte offset corresponding to WLN

            !+
            ! Must deal with this mapping on a row by row basis and on
            ! each row inspect the line characteristics vector entry of
            ! both the DCB line characteristics vector and the WCB line
            ! characteristics vector in order to determine how to map this
            ! particular line.
            !
            ! Initialize the special index we will be using.
            !-
            SPEC_TO_INDEX = (.PP [PP_W_FIRST_WCB_ROW] -1) *
                            .WCB [WCB_W_NO_COLS] +
                            ((.PP [PP_W_FIRST_WCB_COL] +2) / 2) -1;

            DLN = (.FROM_INDEX / .DCB_COLS) +1;
            WLN = .PP [PP_W_FIRST_WCB_ROW];
            WLP = (.WLN - 1) * .WCB_COLS;

            INCR R FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
            DO
                BEGIN           ! For each row
                IF .IN_LINE_CHAR  [.DLN] EQL 0 AND  ! DCB normal
                   .OUT_LINE_CHAR [.WLN] EQL 0      ! WCB normal
                THEN
                    BEGIN   ! Normal line
                    $MAP_ALL_NORMAL;
                    END     ! Normal line
                ELSE
                    BEGIN   ! Special line
                    IF .IN_LINE_CHAR  [.DLN] EQL 0 AND      ! DCB normal
                       .OUT_LINE_CHAR [.WLN] NEQ 0          ! WCB special
                    THEN
                        BEGIN           ! WCB has special, DCB normal
                        IF .PBCB [PBCB_V_WIDE]      ! If DW supported
                        THEN
                            BEGIN ! Special mapping
                            $CLEAR_ALL;
                            $MAP_ALL_NORMAL ;
                            END   ! Special mapping
```

```
                        ELSE
                            BEGIN ! Normal mapping
                            $MAP_ALL_NORMAL;
                            END;  ! Normal mapping
                        END           ! WCB has special, DCB normal
                    ELSE
                        BEGIN
                        IF .IN_LINE_CHAR  [.DLN] NEQ 0 AND  ! DCB special
                           .OUT_LINE_CHAR [.WLN] EQL 0      ! WCB normal
                        THEN
                            BEGIN   ! DCB special, WCB normal
                            IF .PBCB [PBCB_V_WIDE]  ! If DW supported
                            THEN
                                BEGIN       ! Map special
                                $CLEAR_ALL;
                                $MAP_ALL_SPECIAL;
                                END         ! Map special
                            ELSE
                                BEGIN       ! Map normal
                                $MAP_ALL_NORMAL;
                                END;        ! Map normal
                            END       ! DCB special, WCB normal
                        ELSE
                            BEGIN   ! Both have special
                            IF .PBCB [PBCB_V_WIDE]  ! If DW supported
                            THEN
                                BEGIN       ! Map special
                                $MAP_ALL_SPECIAL;
                                END         ! Map special
                            ELSE
                                BEGIN       ! Map normal
                                $MAP_ALL_NORMAL;
                                END;        ! Map normal
                            END;      ! Both have special
                        END;
                    END;    ! Special line
                !+
                ! Advance indices for next row
                !-
                FROM_INDEX    = .FROM_INDEX    + .DCB_COLS;
                TO_INDEX      = .TO_INDEX      + .WCB_COLS;
                SPEC_TO_INDEX = .SPEC_TO_INDEX + .WCB_COLS;
                DLN = .DLN + 1;
                WLN = .WLN + 1;
                WLP = .WLP + .WCB_COLS;
                END;        ! For each row

            END;     ! One or the other contains special characteristics

        !+
        ! If this device supports DW or DWDH then we must update WCB's line
        ! characteristics vector based on the rows we just modified.
        ! All further actions are necessary only if DW or DWDH is supported by
        ! this device.
        !-
            IF .PBCB [PBCB_V_WIDE]
            THEN
```

```
3              BEGIN   ! Special supported
3              IF .PP [PP_W_ROWS_TO_MOVE] NEQ 0        ! Something got mapped
3              THEN
4                  BEGIN
4                  IF .PP [PP_W_ROW] GTR 0
4                  THEN
4                      ! Start movement from 1st display row
4                      CH$MOVE ( .PP [PP_W_ROWS_TO_MOVE],
4                                IN_LINE_CHAR [1],
4                                OUT_LINE_CHAR [ .PP [PP_W_ROW]])
4                  ELSE
4                      ! Start movement from nth display row
4                      CH$MOVE ( .PP [PP_W_ROWS_TO_MOVE],
4                                IN_LINE_CHAR [.DCB [DCB_W_NO_ROWS] -
4                                          .PP [PP_W_ROWS_TO_MOVE] +1],
4                                OUT_LINE_CHAR [1]);
4
3                  END;
3
3              !+
3              ! If we are dealing with a bordered display and we have
3              ! made the transition from a normal line to a DHDW line, or
3              ! vice versa, it will be necessary to remap the border elements.
3              !-
3              IF .DCB [DCB_V_BORDERED]  AND
3                 .RE_BORDER EQL 1
3              THEN
4                  BEGIN
4                  LOCAL
4                      STATUS ; ! Status of subroutine call
5                  IF NOT (STATUS = SMG$$DRAW_BORDER ( .DCB, .PP, .WCB))
4                  THEN
4                      RETURN (.STATUS);
3                  END;
3
3              !+
3              ! Check through current WCB line characteristics vector.  If
3              ! any of the elements is non-zero, set OUT_LINE_CHAR [0] to the
3              ! 1st encountered non-zero value.
3              !-
3              IF .PP [PP_W_ROWS_TO_MOVE] GTR 0
3              THEN
4                  BEGIN        ! Some mapping took place
4                  LOCAL
4                      BYTES_REMAINING,                  ! Output from SKPC
4                      FOUND_BYTE : REF VECTOR [,BYTE]; ! Output from SKPC
4
4                  OUT_LINE_CHAR [0] = 0;       ! Assume none will be found
4                  SKPC (%REF(0), WCB [WCB_W_NO_ROWS], OUT_LINE_CHAR [0];
4                        BYTES_REMAINING, FOUND_BYTE);
4
4                  IF .BYTES_REMAINING NEQ 0   ! If non-zero found
4                  THEN
4                      OUT_LINE_CHAR [0] = .FOUND_BYTE [0];
3                  END;         ! Some mapping took place
2              END;     ! Special supported
2
```

```
; 3136          3193  2
; 3137          3194  2         RETURN (SS$_NORMAL);
; 3138          3195  1         END;                    ! End of routine SMG$$MOVE_TEXT_TO_WINDOW_BUF


                                                   0FFC 00000           .ENTRY  SMG$$MOVE_TEXT_TO_WINDOW_BUF, Save R2,R3,-  ; 2738
                                                                                R4,R5,R6,R7,R8,R9,R10,R11                   ;
                               5E           30  C2 00002           SUBL2   #48, SP                                     ;
                                            7E  D4 00005           CLRL    RE_BORDER                                   ; 2778
                  50      04   AC           10  C1 00007           ADDL3   #16, PP, R0                                 ; 2891
                               57           60  D0 0000C           MOVL    (R0), DCB                                   ;
                  50      04   AC           14  C1 0000F           ADDL3   #20, PP, R0                                 ; 2892
                                            60  DD 00014           PUSHL   (R0)                                        ;
                  50           6E           08  C1 00016           ADDL3   #8, PBCB, R0                                ; 2893
                               5B           60  D0 0001A           MOVL    (R0), WCB                                   ;
                                       4C   A7  DD 0001D           PUSHL   76(DCB)                                     ; 2894
                                       2C   AB  DD 00020           PUSHL   44(WCB)                                     ; 2895
                               7E      06   AB  3C 00023           MOVZWL  6(WCB), WCB_COLS                            ; 2896
                               7E      06   A7  3C 00027           MOVZWL  6(DCB), DCB_COLS                            ; 2897
                                       18   A7  9F 0002B           PUSHAB  24(DCB)                                     ; 2905
                                       48   AE  D4 0002E           CLRL    72(SP)                                      ;
                                       00   BE  D5 00031           TSTL    @0(SP)                                      ;
                                       21   13  00034           BEQL    2$                                          ;
                                       48   AE  D6 00036           INCL    72(SP)                                      ;
                                       10   AB  D5 00039           TSTL    16(WCB)                                     ; 2906
                                       19   12  0003C           BNEQ    2$                                          ;
                                       10   AB  9F 0003E           PUSHAB  16(WCB)                                     ; 2913
                                       28   AB  9F 00041           PUSHAB  40(WCB)                                     ; 2912
                          00000000G    00       02  FB 00044           CALLS   #2, LIB$GET_VM                              ; 2913
                                       01       50  E8 0004B           BLBS    STATUS, 1$                                  ;
                                                04  0004E           RET                                                 ;
      28   AB         00           6E           00  2C 0004F 1$:       MOVC5   #0, (SP), #0, 40(WCB), @16(WCB)             ; 2917
                                       10   BB     00055                                                               ;
                  50      04   AC           1E  C1 00057 2$:       ADDL3   #30, PP, R0                                 ; 2963
                               56           60  3C 0005C           MOVZWL  (R0), FROM_INDEX                            ;
                  50      04   AC           20  C1 0005F           ADDL3   #32, PP, R0                                 ; 2964
                               5A           60  3C 00064           MOVZWL  (R0), TO_INDEX                              ;
                          30   AE      10   A7  9E 00067           MOVAB   16(DCB), 48(SP)                             ; 2979
                          28   AE      08   AB  9E 0006C           MOVAB   8(WCB), 40(SP)                              ; 2980
                          2C   AE      14   A7  9E 00071           MOVAB   20(DCB), 44(SP)                             ; 2984
                          24   AE      0C   AB  9E 00076           MOVAB   12(WCB), 36(SP)                             ; 2985
                                       10   BE  95 0007B           TSTB    @IN_LINE_CHAR                               ; 2965
                                            03  12 0007E           BNEQ    3$                                          ;
                                       0C   BE  95 00080           TSTB    @OUT_LINE_CHAR                              ; 2966
                                            03  13 00083 3$:       BEQL    4$                                          ;
                                          00B9  31 00085           BRW     13$                                         ;
                  7E      04   AC           2A  C1 00088 4$:       ADDL3   #42, PP, -(SP)                              ; 2973
                  3C           9E           01  E1 0008D           BBC     #1, @(SP)+, 5$                              ;
                               58      28   BE  D0 00091           MOVL    @40(SP), R8                                 ; 2980
                               59      30   BE  D0 00095           MOVL    @48(SP), R9                                 ;
                  7E      04   AC           2B  C1 00099           ADDL3   #43, PP, -(SP)                              ;
                684A         6946           9E  28 0009E           MOVC3   @(SP)+, (R9)[FROM_INDEX], (R8)[TO_INDEX]    ;
                               58      24   BE  D0 000A4           MOVL    @36(SP), R8                                 ; 2985
                               59      2C   BE  D0 000A8           MOVL    @44(SP), R9                                 ;
```

```
      7E     04    AC          2B  C1 000AC       ADDL3   #43, PP, -(SP)                                        ;
    684A         6946          9E  28 000B1       MOVC3   @(SP)+, (R9)[FROM_INDEX], (R8)[TO_INDEX]              ;
                   4C     48   AE  E9 000B7       BLBC    72(SP), 8$                                            ; 2988
                   58     00   BE  D0 000BB       MOVL    @0(SP), R8                                            ; 2993
      59     04    AC          2B  C1 000BF       ADDL3   #43, PP, R9                                           ;
 10 BB4A         6846          69  28 000C4       MOVC3   (R9), (R8)[FROM_INDEX], @16(WCB)[TO_INDEX]            ;
                               71  11 000CB       BRB     12$                                                   ; 2973
      50     04    AC          1C  C1 000CD 5$:   ADDL3   #28, PP, R0                                           ; 3001
                   59          60  3C 000D2       MOVZWL  (R0), R9                                              ;
                               58  D4 000D5       CLRL    R                                                     ; 3008
                               2A  11 000D7       BRB     7$                                                    ;
                          28   BE  DD 000D9 6$:   PUSHL   @40(SP)                                               ; 3003
                          34   BE  DD 000DC       PUSHL   @52(SP)                                               ;
      7E     04    AC          22  C1 000DF       ADDL3   #34, PP, -(SP)                                        ;
    9E4A         9E46          9E  28 000E4       MOVC3   @(SP)+, @(SP)+[FROM_INDEX], @(SP)+-                   ;
                                                          [TO_INDEX]
                          24   BE  DD 000EA       PUSHL   @36(SP)                                               ; 3004
                          30   BE  DD 000ED       PUSHL   @48(SP)                                               ;
      7E     04    AC          22  C1 000F0       ADDL3   #34, PP, -(SP)                                        ;
    9E4A         9E46          9E  28 000F5       MOVC3   @(SP)+, @(SP)+[FROM_INDEX], @(SP)+-                   ;
                                                          [TO_INDEX]
                   56     04   AE  C0 000FB       ADDL2   DCB_COLS, FROM_INDEX                                  ; 3007
                   5A     08   AE  C0 000FF       ADDL2   WCB_COLS, TO_INDEX                                    ; 3008
      D2           58          59  F3 00103 7$:   AOBLEQ  R9, R, 6$                                             ; 3001
                   33     48   AE  E9 00107 8$:   BLBC    72(SP), 12$                                           ; 3014
      50     04    AC          1E  C1 0010B       ADDL3   #30, PP, R0                                           ; 3017
                   56          60  3C 00110       MOVZWL  (R0), FROM_INDEX                                      ;
      50     04    AC          20  C1 00113       ADDL3   #32, PP, R0                                           ; 3018
                   5A          60  3C 00118       MOVZWL  (R0), TO_INDEX                                        ;
                               58  D4 0011B       CLRL    R                                                     ; 3020
                               1B  11 0011D       BRB     11$                                                   ;
                   0F     48   AE  E9 0011F 9$:   BLBC    72(SP), 10$                                           ; 3022
                          00   BE  DD 00123       PUSHL   @0(SP)                                                ;
      7E     04    AC          22  C1 00126       ADDL3   #34, PP, -(SP)                                        ;
 10 BB4A         9E46          9E  28 0012B       MOVC3   @(SP)+, @(SP)+[FROM_INDEX], @16(WCB)-                 ;
                                                          [TO_INDEX]
                   56     04   AE  C0 00132 10$:  ADDL2   DCB_COLS, FROM_INDEX                                  ; 3025
                   5A     08   AE  C0 00136       ADDL2   WCB_COLS, TO_INDEX                                    ; 3026
      E1           58          59  F3 0013A 11$:  AOBLEQ  R9, R, 9$                                             ; 3020
                             020E  31 0013E 12$:  BRW     27$                                                   ; 2965
      51     04    AC          2F  C1 00141 13$:  ADDL3   #47, PP, R1                                           ; 3050
                   50          61  3C 00146       MOVZWL  (R1), R0                                              ;
                               50  D7 00149       DECL    R0                                                    ;
             3C    AE     06   AB  3C 0014B       MOVZWL  6(WCB), 60(SP)                                        ; 3051
                   50     3C   AE  C4 00150       MULL2   60(SP), R0                                            ;
      52     04    AC          33  C1 00154       ADDL3   #51, PP, R2                                           ; 3052
                   51          62  3C 00159       MOVZWL  (R2), R1                                              ;
                   51          02  C0 0015C       ADDL2   #2, R1                                                ;
                   51          02  C6 0015F       DIVL2   #2, R1                                                ;
             20    AE   FF A140  9E 00162       MOVAB   -1(R1)[R0], SPEC_TO_INDEX                             ;
      59           56     04   AE  C7 00168       DIVL3   DCB_COLS, FROM_INDEX, R9                              ; 3054
                               59  D6 0016D       INCL    DLN                                                   ;
      50     04    AC          2F  C1 0016F       ADDL3   #47, PP, R0                                           ; 3055
                   58          60  3C 00174       MOVZWL  (R0), WLN                                             ;
                   50     FF   A8  9E 00177       MOVAB   -1(R8), R0                                            ; 3056
   1C  AE          50     08   AE  C5 0017B       MULL3   WCB_COLS, R0, WLP                                     ;
      50     04    AC          1C  C1 00181       ADDL3   #28, PP, R0                                           ; 3058
```

```
                          48   AE           60  3C 00186        MOVZWL  (R0), 72(SP)                          ;
                50        04   AC           22  C1 0018A        ADDL3   #34, PP, R0                           ; 3064
                          34   AE           60  9E 0018F        MOVAB   (R0), 52(SP)                          ;
                                      44    AE  D4 00193        CLRL    R                                     ;
                                          01AE  31 00196        BRW     26$                                   ;
                                            50  D4 00199 14$:   CLRL    R0                                    ; 3061
                               51     10    AE  D0 0019B        MOVL    IN_LINE_CHAR, R1                      ;
                                          6941  95 0019F        TSTB    (D[N)[R1]                             ;
                                            0E  12 001A2        BNEQ    16$                                   ;
                                            50  D6 001A4        INCL    R0                                    ;
                               51     0C    AE  D0 001A6        MOVL    OUT_LINE_CHAR, R1                     ; 3062
                                          6841  95 001AA        TSTB    (WLN)[R1]                             ;
                                            03  12 001AD        BNEQ    16$                                   ;
                                          0155  31 001AF 15$:   BRW     23$                                   ;
                51        14   AE 000000FA  8F  C1 001B2 16$:   ADDL3   #250, PBCB, R1                        ; 3073
                          40   AE           61  9E 001BB        MOVAB   (R1), 64(SP)                          ;
                               6D           50  E9 001BF        BLBC    R0, 19$                               ;
                               50     0C    AE  D0 001C2        MOVL    OUT_LINE_CHAR, R0                     ; 3070
                                          6840  95 001C6        TSTB    (WLN)[R0]                             ;
                                            64  13 001C9        BEQL    19$                                   ;
                               E0     40    BE  E9 001CB        BLBC    @64(SP), 15$                          ; 3073
            38   AE       28   BE     1C    AE  C1 001CF        ADDL3   WLP, @40(SP), 56(SP)                  ; 3075
  3C   AE        20            6E           00  2C 001D6        MOVC5   #0, (SP), #32, 60(SP), @56(SP)        ;
                                      38    BE     001DC                                                      ;
            40   AE       24   BE     1C    AE  C1 001DE        ADDL3   WLP, @36(SP), 64(SP)                  ;
  3C   AE        00            6E           00  2C 001E5        MOVC5   #0, (SP), #0, 60(SP), @64(SP)         ;
                                      40    BE     001EB                                                      ;
                                      40    AE  D4 001ED        CLRL    64(SP)                                ;
                                      00    BE  D5 001F0        TSTL    @0(SP)                                ;
                                            12  13 001F3        BEQL    17$                                   ;
                                      40    AE  D6 001F5        INCL    64(SP)                                ;
            38   AE       10   AB     1C    AE  C1 001F8        ADDL3   WLP, 16(R11), 56(SP)                  ;
  3C   AE        00            6E           00  2C 001FF        MOVC5   #0, (SP), #0, 60(SP), @56(SP)         ;
                                      38    BE     00205                                                      ;
                          18   AE           01  D0 00207 17$:   MOVL    #1, RE_BORDER                         ;
                                      28    BE  DD 0020B        PUSHL   @40(SP)                               ; 3076
                                      34    BE  DD 0020E        PUSHL   @52(SP)                               ;
               9E4A          9E46     3C    BE  28 00211        MOVC3   @60(SP), @(SP)+[FROM_INDEX], @(SP)+-  ;
                                                                        [TO_INDEX]                            ;
                                      24    BE  DD 00218        PUSHL   @36(SP)                               ;
                                      30    BE  DD 0021B        PUSHL   @48(SP)                               ;
               9E4A          9E46     3C    BE  28 0021E        MOVC3   @60(SP), @(SP)+[FROM_INDEX], @(SP)+-  ;
                                                                        [TO_INDEX]                            ;
                               03     40    AE  E8 00225        BLBS    64(SP), 18$                           ;
                                          0105  31 00229        BRW     25$                                   ;
                                          00F7  31 0022C 18$:   BRW     24$                                   ;
                               50     10    AE  D0 0022F 19$:   MOVL    IN_LINE_CHAR, R0                      ; 3086
                                          6940  95 00233        TSTB    (D[N)[R0]                             ;
                                            03  12 00236        BNEQ    20$                                   ;
                                          0088  31 00238        BRW     22$                                   ;
                               50     0C    AE  D0 0023B 20$:   MOVL    OUT_LINE_CHAR, R0                     ; 3087
                                          6840  95 0023F        TSTB    (WLN)[R0]                             ;
                                            7F  12 00242        BNEQ    22$                                   ;
                               7B     40    BE  E9 00244        BLBC    @64(SP), 22$                          ; 3090
            40   AE       28   BE     1C    AE  C1 00248        ADDL3   WLP, @40(SP), 64(SP)                  ; 3092
  3C   AE        20            6E           00  2C 0024F        MOVC5   #0, (SP), #32, 60(SP), @64(SP)        ;
                                      40    BE     00255                                                      ;
```

```
              40   AE        24   BE     1C   AE  C1 00257          ADDL3   WLP, @36(SP), 64(SP)                          :
     3C   AE       00             6E          00  2C 0025E          MOVC5   #0, (SP), #0, 60(SP), @64(SP)                 :
                                         40   BE     00264                                                                 :
                                         40   AE  D4 00266          CLRL    64(SP)                                        :
                                         00   BE  D5 00269          TSTL    @0(SP)                                        :
                                              12  13 0026C          BEQL    21$                                           :
                                         40   AE  D6 0026E          INCL    64(SP)                                        :
              38   AE        10   AB     1C   AE  C1 00271          ADDL3   WLP, 16(R11), 56(SP)                          :
     3C   AE       00             6E          00  2C 00278          MOVC5   #0, (SP), #0, 60(SP), @56(SP)                 :
                                         38   BE     0027E                                                                 :
                             18   AE          01  D0 00280 21$:     MOVL    #1, RE_BORDER                                 :
                             38   AE     34   BE  3C 00284          MOVZWL  @52(SP), 56(SP)                               : 3093
                                         38   AE  D7 00289          DECL    56(SP)                                        :
                             38   AE          02  C6 0028C          DIVL2   #2, 56(SP)                                    :
                   7E        28   BE     20   AE  C1 00290          ADDL3   SPEC_TO_INDEX, @40(SP), -(SP)                 :
                                         34   BE  DD 00296          PUSHL   @52(SP)                                       :
                   9E           9E46     40   AE  28 00299          MOVC3   64(SP), @(SP)+[FROM_INDEX], @(SP)+            :
                   7E        24   BE     20   AE  C1 0029F          ADDL3   SPEC_TO_INDEX, @36(SP), -(SP)                 :
                                         30   BE  DD 002A5          PUSHL   @48(SP)                                       :
                   9E           9E46     40   AE  28 002A8          MOVC3   64(SP), @(SP)+[FROM_INDEX], @(SP)+            :
                                  7F     40   AE  E9 002AE          BLBC    64(SP), 25$                                   :
                   7E        10   AB     20   AE  C1 002B2          ADDL3   SPEC_TO_INDEX, 16(R11), -(SP)                 :
                                         04   BE  DD 002B8          PUSHL   @4(SP)                                        :
                   9E           9E46     40   AE  28 002BB          MOVC3   64(SP), @(SP)+[FROM_INDEX], @(SP)+            :
                                              6E  11 002C1          BRB     25$                                           : 3090
                                  40     40   BE  E9 002C3 22$:     BLBC    @64(SP), 23$                                  : 3103
                             40   AE     34   BE  3C 002C7          MOVZWL  @52(SP), 64(SP)                               : 3105
                                         40   AE  D7 002CC          DECL    64(SP)                                        :
                             40   AE          02  C6 002CF          DIVL2   #2, 64(SP)                                    :
                   7E        28   BE     20   AE  C1 002D3          ADDL3   SPEC_TO_INDEX, @40(SP), -(SP)                 :
                                         34   BE  DD 002D9          PUSHL   @52(SP)                                       :
                   9E           9E46     48   AE  28 002DC          MOVC3   72(SP), @(SP)+[FROM_INDEX], @(SP)+            :
                   7E        24   BE     20   AE  C1 002E2          ADDL3   SPEC_TO_INDEX, @36(SP), -(SP)                 :
                                         30   BE  DD 002E8          PUSHL   @48(SP)                                       :
                   9E           9E46     48   AE  28 002EB          MOVC3   72(SP), @(SP)+[FROM_INDEX], @(SP)+            :
                                         00   BE  D5 002F1          TSTL    @0(SP)                                        :
                                              3B  13 002F4          BEQL    25$                                           :
                   7E        10   AB     20   AE  C1 002F6          ADDL3   SPEC_TO_INDEX, 16(R11), -(SP)                 :
                                         04   BE  DD 002FC          PUSHL   @4(SP)                                        :
                   9E           9E46     48   AE  28 002FF          MOVC3   72(SP), @(SP)+[FROM_INDEX], @(SP)+            :
                                              2A  11 00305          BRB     25$                                           : 3103
                                         28   BE  DD 00307 23$:     PUSHL   @40(SP)                                       : 3109
                                         34   BE  DD 0030A          PUSHL   @52(SP)                                       :
                 9E4A           9E46     3C   BE  28 0030D          MOVC3   @60(SP), @(SP)+[FROM_INDEX], @(SP)+-          :
                                                                            [TO_INDEX]                                    :
                                         24   BE  DD 00314          PUSHL   @36(SP)                                       :
                                         30   BE  DD 00317          PUSHL   @48(SP)                                       :
                 9E4A           9E46     3C   BE  28 0031A          MOVC3   @60(SP), @(SP)+[FROM_INDEX], @(SP)+-          :
                                                                            [TO_INDEX]                                    :
                                         00   BE  D5 00321          TSTL    @0(SP)                                        :
                                              0B  13 00324          BEQL    25$                                           :
                                         00   BE  DD 00326 24$:     PUSHL   @0(SP)                                        :
              10 BB4A           9E46     38   BE  28 00329          MOVC3   @56(SP), @(SP)+[FROM_INDEX], @16(WCB)-        :
                                                                            [TO_INDEX]                                    :
                                  56     04   AE  C0 00331 25$:     ADDL2   DCB_COLS, FROM_INDEX                          : 3118
                                  5A     08   AE  C0 00335          ADDL2   WCB_COLS, TO_INDEX                            : 3119
                             20   AE     08   AE  C0 00339          ADDL2   WCB_COLS, SPEC_TO_INDEX                       : 3120
```

```
                                          59  D6 0033E          INCL    DLN                                    ; 3121
                                          58  D6 00340          INCL    WLN                                    ; 3122
                         1C     AE    08  AE  C0 00342          ADDL2   WCB_COLS, WLP                          ; 3123
      FE4A         44    AE     01    48  AE  F1 00347 26$:     ACBL    72(SP), #1, R, 14$                     ; 3058
                   58    14     AE 000000FA 8F C1 0034F 27$:    ADDL3   #250, PBCB, R8                         ; 3134
                                6D        68  E9 00358          BLBC    (R8), 31$                              ;
                   50    04     AC        1C  C1 0035B          ADDL3   #28, PP, R0                            ; 3137
                                56        60  3C 00360          MOVZWL  (R0), R6                               ;
                                          30  13 00363          BEQL    29$                                    ;
                   51    04     AC        18  C1 00365          ADDL3   #24, PP, R1                            ; 3140
                                50        61  32 0036A          CVTWL   (R1), R0                               ;
                                          10  15 0036D          BLEQ    28$                                    ;
                                59    0C  AE  D0 0036F          MOVL    OUT_LINE_CHAR, R9                      ; 3145
                   5A    10     AE        01  C1 00373          ADDL3   #1, IN_LINE_CHAR, R10                  ;
                 6049           6A        56  28 00378          MOVC3   R6, (R10), (R0)[R9]                    ;
                                          16  11 0037D          BRB     29$                                    ;
                                50    02  A7  3C 0037F 28$:     MOVZWL  2(DCB), R0                             ; 3150
                                50        56  C2 00383          SUBL2   R6, R0                                 ;
                   59    0C     AE        01  C1 00386          ADDL3   #1, OUT_LINE_CHAR, R9                  ; 3151
                                5A    10  AE  D0 0038B          MOVL    IN_LINE_CHAR, R10                      ;
                   69    01   A04A        56  28 0038F          MOVC3   R6, 1(R0)[R10], (R9)                   ;
                                15    2F  A7  E9 00395 29$:     BLBC    47(DCB), 30$                           ; 3160
                                01    18  AE  D1 00399          CMPL    RE_BORDER, #1                          ; 3161
                                          0F  12 0039D          BNEQ    30$                                    ;
                                          5B  DD 0039F          PUSHL   WCB                                    ; 3166
                                      04  AC  DD 003A1          PUSHL   PP                                     ;
                                          57  DD 003A4          PUSHL   DCB                                    ;
                        F6BA    CF        03  FB 003A6          CALLS   #3, SMG$$DRAW_BORDER                   ;
                                1D        50  E9 003AB          BLBC    STATUS, 32$                            ;
                   50    04     AC        1C  C1 003AE 30$:     ADDL3   #28, PP, R0                            ; 3176
                                          60  B5 003B3          TSTW    (R0)                                   ;
                                          11  13 003B5          BEQL    31$                                    ;
                                      0C  BE  94 003B7          CLRB    @OUT_LINE_CHAR                         ; 3183
             0C    BE    02     AB        00  3B 003BA          SKPC    #0, 2(WCB), @OUT_LINE_CHAR             ; 3184
                                          50  D5 003C0          TSTL    BYTES_REMAINING                        ; 3187
                                          04  13 003C2          BEQL    31$                                    ;
                         0C     BE        61  90 003C4          MOVB    (FOUND_BYTE), @OUT_LINE_CHAR           ; 3189
                                50        01  D0 003C8 31$:     MOVL    #1, R0                                 ; 3194
                                              04 003CB 32$:     RET                                            ; 3195

; Routine Size:  972 bytes,    Routine Base:  _SMG$CODE + 0BED


; 3139          3196  1 !<BLF/PAGE>
```

```
1 %SBTTL 'SMGOCCLUDE - Check for occlusion'
1 GLOBAL ROUTINE SMG$$OCCLUDE (    LOWER : REF VECTOR [,WORD,SIGNED],
1                                  UPPER : REF VECTOR [,WORD,SIGNED],
1                                  REWRITE : REF VECTOR [,WORD,SIGNED]
1                              ) =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine inspects the relative positioning of two areas --
1 !       a LOWER area and an UPPER area.  It determines whether the
1 !       UPPER area occludes any portion of the LOWER area.  If occlusion
1 !       is found, a rectangular area which is the occluded part is
1 !       isolated and its coordinates are returned as the area which
1 !       must be rewritten.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       CODE.wl.v = SMG$$OCCLUDE (
1 !                                   LOWER.rw.r,
1 !                                   UPPER.rw.r,
1 !                                   REWRITE.waw.r
1 !                                  )
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       LOWER.raw.r     Descripion of a rectangular area represent
1 !                       a "lower" area.!
1 !       UPPER.raw.r     Description of a rectangular area representing
1 !                       an "upper" area.
1 !
1 !       REWRITE.waw.r   Description of a rectangular area which
1 !                       represents the area of the lower which is
1 !                       occluded by the upper ( if any).
1 !
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE
1 !
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !--
1
2     BEGIN
2     LITERAL
2       ROW_START = 0,
2       NUM_ROW   = 1,
2       COL_START = 2,
2       NUM_COL   = 3;
```

```
2
2        LOCAL
2            ANS,
2            STATUS,
2            LOWER_BOTTOM_ROW: SIGNED,
2            LOWER_RIGHT_COL: SIGNED,
2            UPPER_BOTTOM_ROW: SIGNED,
2            UPPER_RIGHT_COL: SIGNED;
2
2    MACRO
2            LOWER_ROW_START             = LOWER [ROW_START]%,
2            LOWER_NO_ROW                = LOWER [NUM_ROW]%,
2            LOWER_COL_START             = LOWER [COL_START]%,
2            LOWER_NO_COL                = LOWER [NUM_COL]%,
2
2            UPPER_ROW_START             = UPPER [ROW_START]%,
2            UPPER_NO_ROW                = UPPER [NUM_ROW]%,
2            UPPER_COL_START             = UPPER [COL_START]%,
2            UPPER_NO_COL                = UPPER [NUM_COL]%,
2
2            REWRITE_ROW_START           = REWRITE [ROW_START]%,
2            REWRITE_NO_ROW              = REWRITE [NUM_ROW]%,
2            REWRITE_COL_START           = REWRITE [COL_START]%,
2            REWRITE_NO_COL              = REWRITE [NUM_COL]%;
2
2    !+
2    ! Calculate the other bounds of the specified rectangles.
2    !-
2        LOWER_BOTTOM_ROW = .LOWER_ROW_START + .LOWER_NO_ROW - 1;
2        LOWER_RIGHT_COL =  .LOWER_COL_START + .LOWER_NO_COL - 1;
2        UPPER_BOTTOM_ROW = .UPPER_ROW_START + .UPPER_NO_ROW - 1;
2        UPPER_RIGHT_COL  = .UPPER_COL_START + .UPPER_NO_COL - 1;
2
2    !+
2    ! Assume no occlusion occurs until it proves otherwise.
2    !-
2        STATUS = 0;
2
2    !+
2    ! Check for relative position of upper left-hand corner of UPPER
2    ! rectangle with respect to LOWER rectangle.
2    !-
2        ANS = SMG$$POINT_IN_RECT_R3 (UPPER_COL_START,        ! X1
2                                     UPPER_ROW_START,        ! Y1
2                                     .LOWER);
2
2        CASE .ANS FROM 0 TO 10 OF
2        SET
3            [0]:BEGIN
3                REWRITE_ROW_START = .UPPER_ROW_START;
3                REWRITE_COL_START = .UPPER_COL_START;
3                !+
3                ! Check relative position of lower right-hand corner of
3                ! UPPER rectangle with respect to LOWER rectangle.
3                !-
3                ANS = SMG$$POINT_IN_RECT_R3 (UPPER_RIGHT_COL,        ! X1
3                                             UPPER_BOTTOM_ROW,       ! Y1
```

```
3                                       .LOWER);
3
3                   CASE .ANS FROM 0 TO 6 OF
3                   SET
4                       [0]:BEGIN
4                           REWRITE_NO_ROW = .UPPER_NO_ROW;
4                           REWRITE_NO_COL = .UPPER_NO_COL;
3                           END;
3
4                       [2]:BEGIN
4                           REWRITE_NO_ROW = .UPPER_NO_ROW;
4                           REWRITE_NO_COL = .LOWER_RIGHT_COL -.UPPER_COL_START+1;
3                           END;
3
4                       [4]:BEGIN
4                           REWRITE_NO_ROW = .LOWER_BOTTOM_ROW -.UPPER_ROW_START+1;
4                           REWRITE_NO_COL = .UPPER_NO_COL;
3                           END;
3
4                       [6]:BEGIN
4                           REWRITE_NO_ROW = .LOWER_BOTTOM_ROW -.UPPER_ROW_START+1;
4                           REWRITE_NO_COL = .LOWER_RIGHT_COL - .UPPER_COL_START+1;
3                           END;
3                       [1,3,5]: RETURN (SMG$_FATERRLIB);
3
3                   TES;
2                   END;
2
3               [1]:BEGIN
3                   REWRITE_ROW_START = .UPPER_ROW_START;
3                   REWRITE_COL_START = .LOWER_COL_START;
3                   !+
3                   ! Check relative position of lower right-hand corner of
3                   ! UPPER rectangle with respect to lower rectangle.
3                   !-
3                   ANS = SMG$$POINT_IN_RECT_R3 (UPPER_RIGHT_COL,        ! X1
3                                                UPPER_BOTTOM_ROW,       ! Y1
3                                                .LOWER);
3
3                   CASE .ANS FROM 0 TO 6 OF
3                   SET
4                       [0]:BEGIN
4                           REWRITE_NO_ROW = .UPPER_NO_ROW;
4                           REWRITE_NO_COL = .UPPER_RIGHT_COL - .LOWER_COL_START +1;
3                           END;
3
3                       [1]:    RETURN (.STATUS);
3
4                       [2]:BEGIN
4                           REWRITE_NO_ROW = .UPPER_NO_ROW;
4                           REWRITE_NO_COL = .LOWER_NO_COL;
3                           END;
3
4                       [4]:BEGIN
4                           REWRITE_NO_ROW = .LOWER_BOTTOM_ROW -.UPPER_ROW_START+1;
4                           REWRITE_NO_COL = .UPPER_RIGHT_COL -.LOWER_COL_START+1;
3                           END;
```

```

                [5]:    RETURN (.STATUS);

                [6]:BEGIN
                    REWRITE_NO_ROW = .LOWER_BOTTOM_ROW -.UPPER_ROW_START+1;
                    REWRITE_NO_COL = .LOWER_NO_COL;
                    END;
                [3]: RETURN (SMG$_FATERRLIB);

            TES;
            END;

        [2]:    RETURN (.STATUS);

        [4]:    RETURN (.STATUS);

        [5]:    RETURN (.STATUS);

        [6]:    RETURN (.STATUS);

        [8]:BEGIN
            REWRITE_ROW_START = .LOWER_ROW_START;
            REWRITE_COL_START = .UPPER_COL_START;
            !+
            ! Check relative position of lower right-hand corner of
            ! UPPER rectangle with respect to LOWER rectangle.
            !-
            ANS = SMG$$POINT_IN_RECT_R3 (UPPER_RIGHT_COL,      ! X1
                                         UPPER_BOTTOM_ROW,     ! Y1
                                         .LOWER);

            CASE .ANS FROM 0 TO 10 OF
            SET
                [0]:BEGIN
                    REWRITE_NO_ROW = .UPPER_BOTTOM_ROW -.LOWER_ROW_START+1;
                    REWRITE_NO_COL = .UPPER_NO_COL;
                    END;

                [2]:BEGIN
                    REWRITE_NO_ROW = .UPPER_BOTTOM_ROW -.LOWER_ROW_START+1;
                    REWRITE_NO_COL = .LOWER_RIGHT_COL - .UPPER_COL_START+1;
                    END;

                [4]:BEGIN
                    REWRITE_NO_ROW = .LOWER_NO_ROW;
                    REWRITE_NO_COL = .UPPER_NO_COL;
                    END;

                [6]:BEGIN
                    REWRITE_NO_ROW = .LOWER_NO_ROW;
                    REWRITE_NO_COL = .LOWER_RIGHT_COL -.UPPER_COL_START+1;
                    END;
                [8]:    RETURN (.STATUS);

                [10]:   RETURN (.STATUS);

                [1,3,5,7,9]: RETURN (SMG$_FATERRLIB);
```

```
                TES;
                END;

        [9]:BEGIN
            REWRITE_ROW_START = .LOWER_ROW_START;
            REWRITE_COL_START = .LOWER_COL_START;
            !+
            ! Check relative position of lower right-hand corner of
            ! UPPER rectangle with respect to LOWER rectangle.
            !-
            ANS = SMG$$POINT_IN_RECT_R3 (UPPER_RIGHT_COL,        ! X1
                                         UPPER_BOTTOM_ROW,       ! Y1
                                         .LOWER);

            CASE .ANS FROM 0 TO 10 OF
            SET
                [0]:BEGIN
                    REWRITE_NO_ROW = .UPPER_BOTTOM_ROW -.LOWER_ROW_START+1;
                    REWRITE_NO_COL = .UPPER_RIGHT_COL -.LOWER_COL_START+1;
                    END;

                [1]:    RETURN (.STATUS);

                [2]:BEGIN
                    REWRITE_NO_ROW = .UPPER_BOTTOM_ROW -.LOWER_ROW_START+1;
                    REWRITE_NO_COL = .LOWER_NO_COL;
                    END;

                [4]:BEGIN
                    REWRITE_NO_ROW = .LOWER_NO_ROW;
                    REWRITE_NO_COL = .UPPER_RIGHT_COL -.LOWER_COL_START+1;
                    END;

                [5]:    RETURN (.STATUS);

                [6]:BEGIN
                    REWRITE_NO_ROW = .LOWER_NO_ROW;;
                    REWRITE_NO_COL = .LOWER_NO_COL;
                    END;

                [8]:    RETURN (.STATUS);

                [9]:    RETURN (.STATUS);

                [10]:   RETURN (.STATUS);

                [3,7]: RETURN (SMG$_FATERRLIB);

            TES;
            END;

        [10]:   RETURN (.STATUS);

        [3,7]: RETURN (SMG$_FATERRLIB);

    TES;
```

```
; 3426        3482  2
; 3427        3483  2  !+
; 3428        3484  2  ! If we reach here, some occlusion has been detected and the output
; 3429        3485  2  ! arguments have been calculated.  All that remains is to return the
; 3430        3486  2  ! right status reporting existance of occlusion.
; 3431        3487  2  !-
; 3432        3488  2      RETURN (1);
; 3433        3489  2
; 3434        3490  1      END;                    ! End of routine SMG$$OCCLUDE


                                         0FFC 00000         .ENTRY  SMG$$OCCLUDE, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 3198
                                                                    R10,R11                                  ;
                         5E          08    C2 00002         SUBL2   #8, SP                                   ;
                         52    04    AC    D0 00005         MOVL    LOWER, R2                                ; 3282
                               02    A2    9F 00009         PUSHAB  2(R2)                                    ;
                         50          62    32 0000C         CVTWL   (R2), R0                                 ;
                         51    00    BE    32 0000F         CVTWL   @0(SP), R1                               ;
                         50          51    C0 00013         ADDL2   R1, R0                                   ;
                               FF    A0    9F 00016         PUSHAB  -1(R0)                                   ;
                         5A    04    A2    9E 00019         MOVAB   4(R2), R10                               ; 3283
                         5B    06    A2    9E 0001D         MOVAB   6(R2), R11                               ;
                         50          6A    32 00021         CVTWL   (R10), R0                                ;
                         51          6B    32 00024         CVTWL   (R11), R1                                ;
                         50          51    C0 00027         ADDL2   R1, R0                                   ;
                         54    FF    A0    9E 0002A         MOVAB   -1(R0), LOWER_RIGHT_COL                  ;
                         56    08    AC    D0 0002E         MOVL    UPPER, R6                                ; 3284
                         59    02    A6    9E 00032         MOVAB   2(R6), R9                                ;
                         51          66    32 00036         CVTWL   (R6), R1                                 ;
                         50          69    32 00039         CVTWL   (R9), R0                                 ;
                         51          50    C0 0003C         ADDL2   R0, R1                                   ;
                   08    AE    FF    A1    9E 0003F         MOVAB   -1(R1), UPPER_BOTTOM_ROW                 ;
                         58    06    A6    9E 00044         MOVAB   6(R6), R8                                ; 3285
                         51    04    A6    32 00048         CVTWL   4(R6), R1                                ;
                         50          68    32 0004C         CVTWL   (R8), R0                                 ;
                         51          50    C0 0004F         ADDL2   R0, R1                                   ;
                   0C    AE    FF    A1    9E 00052         MOVAB   -1(R1), UPPER_RIGHT_COL                  ;
                                     57    D4 00057         CLRL    STATUS                                   ; 3290
                         50    04    A6    9E 00059         MOVAB   4(R6), R0                                ; 3296
                         51          56    D0 0005D         MOVL    R6, R1                                   ;
                                  0000V    30 00060         BSBW    SMG$$POINT_IN_RECT_R3                    ;
                         53          50    D0 00063         MOVL    R0, ANS                                  ;
              0A         00          53    CF 00066         CASEL   ANS, #0, #10                             ; 3300
  01B7        01B3     006F        0016       0006A  1$:    .WORD   2$-1$,-                                  ;
  01B7        01B3     01B3        01B3       00072                 9$-1$,-                                  ;
              01B3     0143        00D3       0007A                 38$-1$,-                                 ;
                                                                    39$-1$,-                                 ;
                                                                    38$-1$,-                                 ;
                                                                    38$-1$,-                                 ;
                                                                    38$-1$,-                                 ;
                                                                    39$-1$,-                                 ;
                                                                    18$-1$,-                                 ;
                                                                    29$-1$,-                                 ;
                                                                    38$-1$                                   ;
```

```
                        55    0C   AC  D0 00080 2$:   MOVL    REWRITE, R5                   ; 3302
                        65         66  B0 00084       MOVW    (R6), (R5)                    ; 3303
                  04    A5    04   A6  B0 00087       MOVW    4(R6), 4(R5)                  ; 3304
                        51    08   AE  9E 0008C       MOVAB   UPPER_BOTTOM_ROW, R1          ; 3309
                        50    0C   AE  9E 00090       MOVAB   UPPER_RIGHT_COL, R0           ;
                           0000V   30 00094       BSBW    SMG$$POINT_IN_RECT_R3         ;
                        53         50  D0 00097       MOVL    R0, ANS                       ;
            06          00         53  CF 0009A       CASEL   ANS, #0, #6                   ; 3313
 0183      0015       0183  000E      0009E 3$:   .WORD   4$-3$,-                       ;
           0025       0183  001B      000A6               39$-3$,-                      ;
                                                          5$-3$,-                       ;
                                                          39$-3$,-                      ;
                                                          6$-3$,-                       ;
                                                          39$-3$,-                      ;
                                                          7$-3$                         ;
                  02    A5         69  B0 000AC 4$:   MOVW    (R9), 2(R5)                   ; 3316
                              00E1 31 000B0       BRW     24$                           ; 3317
                  02    A5         69  B0 000B3 5$:   MOVW    (R9), 2(R5)                   ; 3321
                                   16  11 000B7       BRB     8$                            ; 3322
                        50         66  32 000B9 6$:   CVTWL   (R6), R0                      ; 3326
            50          6E         50  C3 000BC       SUBL3   R0, LOWER_BOTTOM_ROW, R0      ;
                              00B6 31 000C0       BRW     21$                           ;
                        50         66  32 000C3 7$:   CVTWL   (R6), R0                      ; 3331
            50          6E         50  C3 000C6       SUBL3   R0, LOWER_BOTTOM_ROW, R0      ;
       02   A5          50         01  A1 000CA       ADDW3   #1, R0, 2(R5)                 ;
                        50    04   A6  32 000CF 8$:   CVTWL   4(R6), R0                     ; 3332
            50          54         50  C3 000D3       SUBL3   R0, LOWER_RIGHT_COL, R0       ;
                                   4B  11 000D7       BRB     15$                           ;
                        55    0C   AC  D0 000D9 9$:   MOVL    REWRITE, R5                   ; 3339
                        65         66  B0 000DD       MOVW    (R6), (R5)                    ; 3340
                  04    A5         6A  B0 000E0       MOVW    (R10), 4(R5)                  ; 3341
                        51    08   AE  9E 000E4       MOVAB   UPPER_BOTTOM_ROW, R1          ; 3346
                        50    0C   AE  9E 000E8       MOVAB   UPPER_RIGHT_COL, R0           ;
                           0000V   30 000EC       BSBW    SMG$$POINT_IN_RECT_R3         ;
                        53         50  D0 000EF       MOVL    R0, ANS                       ;
            06          00         53  CF 000F2       CASEL   ANS, #0, #6                   ; 3350
 012B      0014       0127  000E      000F6 10$:  .WORD   11$-10$,-                     ;
           0035       0127  001A      000FE               38$-10$,-                     ;
                                                          12$-10$,-                     ;
                                                          39$-10$,-                     ;
                                                          13$-10$,-                     ;
                                                          38$-10$,-                     ;
                                                          16$-10$                       ;
                  02    A5         69  B0 00104 11$:  MOVW    (R9), 2(R5)                   ; 3353
                                   12  11 00108       BRB     14$                           ; 3354
                  02    A5         69  B0 0010A 12$:  MOVW    (R9), 2(R5)                   ; 3360
                                   27  11 0010E       BRB     17$                           ; 3361
                        50         66  32 00110 13$:  CVTWL   (R6), R0                      ; 3365
            50          6E         50  C3 00113       SUBL3   R0, LOWER_BOTTOM_ROW, R0      ;
       02   A5          50         01  A1 00117       ADDW3   #1, R0, 2(R5)                 ;
                        50         6A  32 0011C 14$:  CVTWL   (R10), R0                     ; 3366
            50    0C    AE         50  C3 0011F       SUBL3   R0, UPPER_RIGHT_COL, R0       ;
       06   A5          50         01  A1 00124 15$:  ADDW3   #1, R0, 6(R5)                 ;
                                   6D  11 00129       BRB     25$                           ; 3350
                        50         66  32 0012B 16$:  CVTWL   (R6), R0                      ; 3372
            50          6E         50  C3 0012E       SUBL3   R0, LOWER_BOTTOM_ROW, R0      ;
       02   A5          50         01  A1 00132       ADDW3   #1, R0, 2(R5)                 ;
```

```
                    06   A5            6B  B0 00137 17$:   MOVW    (R11), 6(R5)                     ; 3373
                                       6E  11 0013B        BRB     28$                              ; 3350
                         55     0C     AC  D0 0013D 18$:   MOVL    REWRITE, R5                      ; 3388
                         65            62  B0 00141        MOVW    (R2), (R5)                       ; 3389
                    04   A5     04     A6  B0 00144        MOVW    4(R6), 4(R5)                     ; 3390
                         51     08     AE  9E 00149        MOVAB   UPPER_BOTTOM_ROW, R1             ; 3395
                         50     0C     AE  9E 0014D        MOVAB   UPPER_RIGHT_COL, R0              ;
                                    0000V  30 00151        BSBW    SMG$$POINT_IN_RECT_R3            ;
                         53            50  D0 00154        MOVL    R0, ANS                          ;
              0A         00            53  CF 00157        CASEL   ANS, #0, #10                     ; 3399
 00C6       0025       00C6          0016     0015B 19$:   .WORD   20$-19$,-                        ;
 00C6       003F       00C6          0034     00163                39$-19$,-                        ;
            00C2       00C6          00C2     0016B                22$-19$,-                        ;
                                                                   39$-19$,-                        ;
                                                                   23$-19$,-                        ;
                                                                   39$-19$,-                        ;
                                                                   26$-19$,-                        ;
                                                                   39$-19$,-                        ;
                                                                   38$-19$,-                        ;
                                                                   39$-19$,-                        ;
                                                                   38$-19$                          ;
                         50            62  32 00171 20$:   CVTWL   (R2), R0                         ; 3402
              50         08     AE     50  C3 00174        SUBL3   R0, UPPER_BOTTOM_ROW, R0         ;
         02   A5         50            01  A1 00179 21$:   ADDW3   #1, R0, 2(R5)                    ;
                                       14  11 0017E        BRB     24$                              ; 3403
                         50            62  32 00180 22$:   CVTWL   (R2), R0                         ; 3407
              50         08     AE     50  C3 00183        SUBL3   R0, UPPER_BOTTOM_ROW, R0         ;
         02   A5         50            01  A1 00188        ADDW3   #1, R0, 2(R5)                    ;
                                       10  11 0018D        BRB     27$                              ; 3408
                    02   A5     04     BE  B0 0018F 23$:   MOVW    @4(SP), 2(R5)                    ; 3412
                    06   A5            68  B0 00194 24$:   MOVW    (R8), 6(R5)                      ; 3413
                                       76  11 00198 25$:   BRB     35$                              ; 3399
                    02   A5     04     BE  B0 0019A 26$:   MOVW    @4(SP), 2(R5)                    ; 3417
                         50     04     A6  32 0019F 27$:   CVTWL   4(R6), R0                        ; 3418
                         54            50  C2 001A3        SUBL2   R0, R4                           ;
         06   A5         54            01  A1 001A6        ADDW3   #1, R4, 6(R5)                    ;
                                       7C  11 001AB 28$:   BRB     40$                              ; 3399
                         54     0C     AC  D0 001AD 29$:   MOVL    REWRITE, R4                      ; 3429
                         64            62  B0 001B1        MOVW    (R2), (R4)                       ; 3430
                    04   A4            6A  B0 001B4        MOVW    (R10), 4(R4)                     ; 3431
                         51     08     AE  9E 001B8        MOVAB   UPPER_BOTTOM_ROW, R1             ; 3436
                         50     0C     AE  9E 001BC        MOVAB   UPPER_RIGHT_COL, R0              ;
                                    0000V  30 001C0        BSBW    SMG$$POINT_IN_RECT_R3            ;
                         53            50  D0 001C3        MOVL    R0, ANS                          ;
              0A         00            53  CF 001C6        CASEL   ANS, #0, #10                     ; 3440
 0057       0025       0053          0016     001CA 30$:   .WORD   31$-30$,-                        ;
 0057       0048       0053          0034     001D2                38$-30$,-                        ;
            0053       0053          0053     001DA                32$-30$,-                        ;
                                                                   39$-30$,-                        ;
                                                                   33$-30$,-                        ;
                                                                   38$-30$,-                        ;
                                                                   36$-30$,-                        ;
                                                                   39$-30$,-                        ;
                                                                   38$-30$,-                        ;
                                                                   38$-30$,-                        ;
                                                                   38$-30$                          ;
                         50            62  32 001E0 31$:   CVTWL   (R2), R0                         ; 3443
```

```
          50       08   AE            50  C3 001E3          SUBL3   R0, UPPER_BOTTOM_ROW, R0              ;
     02   A4            50            01  A1 001E8          ADDW3   #1, R0, 2(R4)                         ;
                                      14  11 001ED          BRB     34$                                   ; 3444
                        50            62  32 001EF 32$:     CVTWL   (R2), R0                              ; 3450
          50       08   AE            50  C3 001F2          SUBL3   R0, UPPER_BOTTOM_ROW, R0              ;
     02   A4            50            01  A1 001F7          ADDW3   #1, R0, 2(R4)                         ;
                                      19  11 001FC          BRB     37$                                   ; 3451
                   02   A4       04   BE  B0 001FE 33$:     MOVW    @4(SP), 2(R4)                         ; 3455
                        50            6A  32 00203 34$:     CVTWL   (R10), R0                             ; 3456
          50       0C   AE            50  C3 00206          SUBL3   R0, UPPER_RIGHT_COL, R0               ;
     06   A4            50            01  A1 0020B          ADDW3   #1, R0, 6(R4)                         ;
                                      17  11 00210 35$:     BRB     40$                                   ; 3440
                   02   A4       04   BE  B0 00212 36$:     MOVW    @4(SP), 2(R4)                         ; 3462
                   06   A4            6B  B0 00217 37$:     MOVW    (R11), 6(R4)                          ; 3463
                                      0C  11 0021B          BRB     40$                                   ; 3440
                        50            57  D0 0021D 38$:     MOVL    STATUS, R0                            ; 3477
                                      04     00220          RET                                           ;
                        50 00000000G  8F  D0 00221 39$:     MOVL    #SMG$_FATERRLIB, R0                   ; 3479
                                      04     00228          RET                                           ;
                        50            01  D0 00229 40$:     MOVL    #1, R0                                ; 3488
                                      04     0022C          RET                                           ; 3490

; Routine Size:  557 bytes,    Routine Base:  _SMG$CODE + 0FB9

; 3435          3491   1 !<BLF/PAGE>
```

```
%SBTTL 'SMG$$POINT_IN_RECT_R3 - Point inside rectangle'
GLOBAL ROUTINE SMG$$POINT_IN_RECT_R3 (  ! Point in rectangle
                X1 : REF VECTOR [,WORD,SIGNED], ! X coordinate of point
                Y1 : REF VECTOR [,WORD,SIGNED], ! Y coordinate of point
                RECT : REF VECTOR [,WORD,SIGNED]
                                ) : POINT_IN_RECT_LINK  =
!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine returns a code indicating the relative placement
! of a point with respect to a horizontally-oriented window (that is,
! one of the sides of the rectangle is parallel to the x-axis.
! (See diagram below).
!
!       This is the inner-most function of the Cohen-Sutherland clipping
! algorithm.
!
!       These codes have two significant properties:
!
!       1.  If the CODE of two points are each zero, both points are
!           within the window and no clipping needs to be done.
!
!       2.  If the logical AND of the CODEs of two points is non-zero,
!           the points are both outside of the window and are so
!           positioned that the line joining them does not intersect
!           the window. (I.e., it is a line segment which need not
!           be even partially displayed.
!
!
!                         !                      !
!       1001              !        1000          !        1010
!                         !                      !
!  -----------------------+----------------------+----------- RECT [DCB_W_ROW_START]
!                         !                      !          ^
!                         !       Window         !          !  RECT [DCB_W_NO_ROWS]
!                         !                      !          !
!       0001              !        0000          !     0010 !
!                         !                      !          V
!  -----------------------+----------------------+-----------
!                         !                      !
!       0101              !        0100          !     0110
!                         !                      !
!                         !<-RECT [DCB_W_NO_COLS]>!
!                         !                      !
!       RECT [DCB_W_COL_START]
!
!
! CALLING SEQUENCE:
!
!       CODE.wl.v = SMG$$POINT_IN_RECT_R3 (X1.rw.r, Y1.rw.r, RECT.rab.r)
!
! FORMAL PARAMETERS:
!
!       X1.rw.r         X coordinate of point
!       Y1.rw.r         Y coordinate of point
!       RECT.rab.r      Discription of the rectangular area.
!
```

```
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE
1 !
1 !       One of the 9 codes indicated in the diagram above.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !--
1
2     BEGIN
2
2     LITERAL
2         ROW_START = 0,
2         NUM_ROW   = 1,
2         COL_START = 2,
2         NUM_COL   = 3;
2
3     RETURN (
4         (
4         IF .X1[0] LSS .RECT [COL_START]
4         THEN                          %B'0001'     ! to left
4         ELSE
5             BEGIN
6             IF .X1[0] GEQ (.RECT [COL_START] + .RECT [NUM_COL])
5             THEN                      %B'0010'     ! to right
5             ELSE                      0            ! neither to right or left
5             END
4         )
3         +
4         (
5         IF .Y1[0] GEQ (.RECT [ROW_START] + .RECT [NUM_ROW])
4         THEN                          %B'0100'     ! below bottom
4         ELSE
5             BEGIN
5             IF .Y1[0] LSS .RECT [ROW_START]
5             THEN                      %B'1000'     ! above top
5             ELSE                      0            ! Neither above or below
5             END
2         ) ) ;
2
2
1     END;                    ! End of routine SMG$$POINT_IN_RECT_R3


                        5E            04  C2 00000 SMG$$POINT_IN_RECT_R3::
```

```
                                                SUBL2   #4, SP                  ; 3493
              04   A2        60  B1 00003       CMPW    (X1), 4(RECT)           ; 3577
                             05  18 00007       BGEQ    1$                      ;
                   53        01  D0 00009       MOVL    #1, R3                  ;
                             1C  11 0000C       BRB     4$                      ;
                   53   04   A2  32 0000E 1$:   CVTWL   4(RECT), R3             ; 3581
                   6E   06   A2  32 00012       CVTWL   6(RECT), (SP)           ;
                   53        6E  C0 00016       ADDL2   (SP), R3                ;
  53        60     10        00  EC 00019       CMPV    #0, #16, (X1), R3       ;
                             05  19 0001E       BLSS    2$                      ;
                   50        02  D0 00020       MOVL    #2, R0                  ;
                             02  11 00023       BRB     3$                      ;
                             50  D4 00025 2$:   CLRL    R0                      ;
                   53        50  D0 00027 3$:   MOVL    R0, R3                  ; 3580
                   50        62  32 0002A 4$:   CVTWL   (RECT), R0              ; 3588
                   6E   02   A2  32 0002D       CVTWL   2(RECT), (SP)           ;
                   50        6E  C0 00031       ADDL2   (SP), R0                ;
  50        61     10        00  EC 00034       CMPV    #0, #16, (Y1), R0       ;
                             05  19 00039       BLSS    5$                      ;
                   50        04  D0 0003B       MOVL    #4, R0                  ;
                             0C  11 0003E       BRB     7$                      ;
                   62        61  B1 00040 5$:   CMPW    (Y1), (RECT)            ; 3592
                             05  18 00043       BGEQ    6$                      ;
                   50        08  D0 00045       MOVL    #8, R0                  ;
                             02  11 00048       BRB     7$                      ;
                             50  D4 0004A 6$:   CLRL    R0                      ;
                   53        50  C0 0004C 7$:   ADDL2   R0, R3                  ; 3587
                   50        53  D0 0004F       MOVL    R3, R0                  ; 3575
                   5E        04  C0 00052       ADDL2   #4, SP                  ; 3599
                                 05 00055       RSB                             ;

; Routine Size:  86 bytes,    Routine Base:  _SMG$CODE + 11E6

; 3545         3600  1 !<BLF/PAGE>
```

```
; 3547          3601  1 END                    ! End of module SMG$DISPLAY_OUTPUT
; 3548          3602  1
; 3549          3603  0 ELUDOM



;                                PSECT SUMMARY
;
;       Name                          Bytes                        Attributes
;
;   _SMG$CODE                           4668  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        16         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        99        21              38        00:00.4



;                                COMMAND QUALIFIERS

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

; Size:           4667 code + 1 data bytes
; Run Time:          01:42.7
; Elapsed Time:      04:56.9
; Lines/CPU Min:      2104
; Lexemes/CPU-Min:  19506
; Memory Used:   537 pages
; Compilation Complete
```

0357 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SMGDISOUT
LIS

0358 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SMGKEYUTI
LIS

SMGKEYPAD
LIS

SMGINPUT
LIS

SMGLIB
LIS