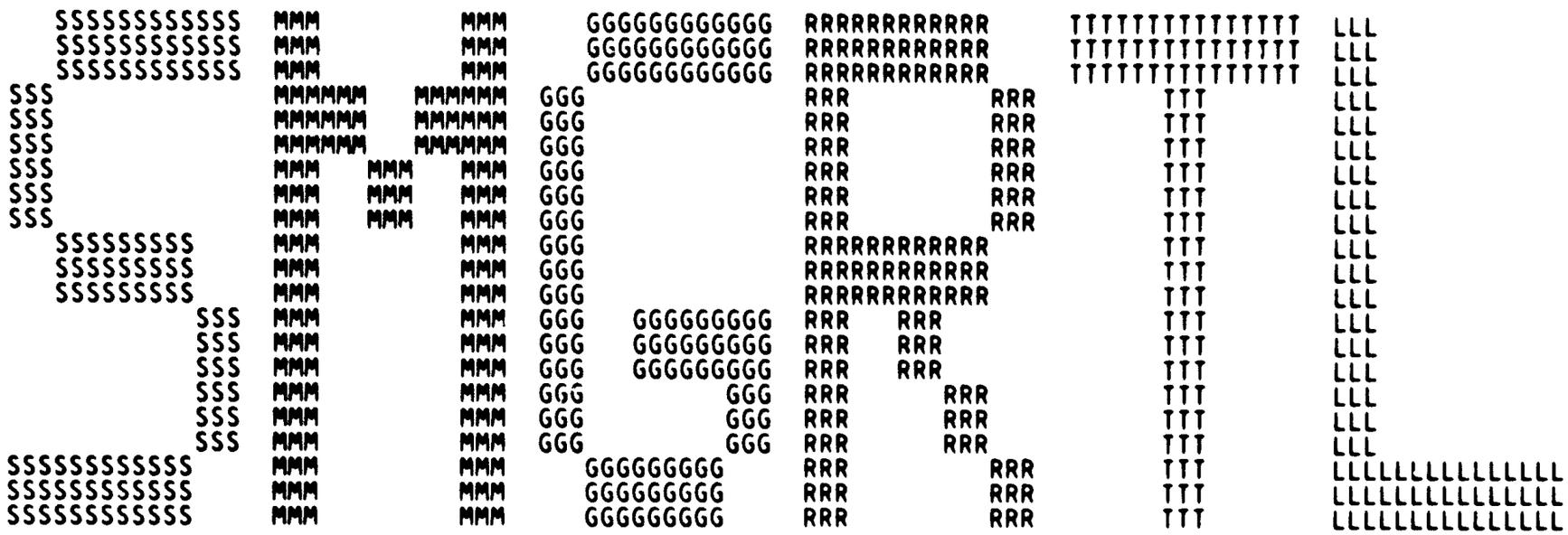

_$2

Val

001
001
001
001
001
001
001
001
001
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF

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
```

```
;  1          0001  0 %TITLE 'SMG$DISPLAY_OUTPUT - Output Virtual Displays'
;  2          0002  0 MODULE SMG$DISPLAY_OUTPUT (
;  3          0003  0                 IDENT = '1-072' ! File: SMGDISOUT.B32 Edit: PLL1072
;  4          0004  0                 ) =
;  5          0005  1 BEGIN
;  6          0006  1 !
;  7          0007  1 !*****************************************************************************
;  8          0008  1 !*                                                                           *
;  9          0009  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
; 10          0010  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
; 11          0011  1 !*  ALL RIGHTS RESERVED.                                                     *
; 12          0012  1 !*                                                                           *
; 13          0013  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
; 14          0014  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
; 15          0015  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
; 16          0016  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
; 17          0017  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
; 18          0018  1 !*  TRANSFERRED.                                                             *
; 19          0019  1 !*                                                                           *
; 20          0020  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
; 21          0021  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
; 22          0022  1 !*  CORPORATION.                                                             *
; 23          0023  1 !*                                                                           *
; 24          0024  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
; 25          0025  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
; 26          0026  1 !*                                                                           *
; 27          0027  1 !*                                                                           *
; 28          0028  1 !*****************************************************************************
; 29          0029  1 !
; 30          0030  1
; 31          0031  1 !++
; 32          0032  1 ! FACILITY:     Screen Management
; 33          0033  1 !
; 34          0034  1 ! ABSTRACT:
; 35          0035  1 !       The procedures in this module output the information contained
; 36          0036  1 ! in virtual displays, using the mappings to windows defined via their
; 37          0037  1 ! pastings to pasteboards.  Also include in this module are procedures
; 38          0038  1 ! to optimize decisions about what needs to be output, and how to output
; 39          0039  1 ! in the most optimal fashion.
; 40          0040  1 !
; 41          0041  1 !       The virtual displays and pasteboards are allocated/deallocated,
; 42          0042  1 ! pasted/unpasted, etc. by the procedures in SMG$DISPLAY_LINKS.  The
; 43          0043  1 ! contents of the virtual displays themselves are maintained by
; 44          0044  1 ! procedures in SMG$DISPLAY_CHANGE.  The routines in module
; 45          0045  1 ! SMG$DISPLAY_INPUT support input operations wiht respect to virtual
; 46          0046  1 ! displays.
; 47          0047  1 !
; 48          0048  1 !
; 49          0049  1 !
; 50          0050  1 ! ENVIRONMENT:  User mode, Shared library routines.
; 51          0051  1 !
; 52          0052  1 ! AUTHOR: R. Reichert, CREATION DATE: 27-Jan-1983
; 53          0053  1 !
; 54          0054  1 ! MODIFIED BY:
; 55          0055  1 !
; 56          0056  1 ! 1-001 - Original.  Skeleton for future code.  RKR 27-Jan-1983
; 57          0057  1 ! 1-002 - Further development.  RKR 11-Mar-1983
```

```
;    58          0058  1 ! 1-003 - Correct names of data structures and macros.  PLL 15-Mar-1983
;    59          0059  1 ! 1-004 - Add SMG$FLUSH_BUFFER, SMG$$FLUSH_BUFFER and SMG$CONTROL_MODE.
;    60          0060  1 !         Delete SMG$ENABLE_BUFFER and SMG$DISABLE_BUFFER.
;    61          0061  1 !         RKR 17-Mar-1983.
;    62          0062  1 ! 1-005 - Minor tweaks to comments.  RKR 18-Mar-1983.
;    63          0063  1 ! 1-006 - Change some names.  RKR 25-Mar-1983.
;    64          0064  1 ! 1-007 - Make FILL_WINDOW_BUFFER able to handle pastings that don't
;    65          0065  1 !         correspond exactly to the window buffer.  RKR 28-Mar-1983.
;    66          0066  1 ! 1-008 - More debugging of above.  RKR 29-Mar-1983.
;    67          0067  1 ! 1-009 - Add some code dealing with borders.  RKR 4-APR-1983.
;    68          0068  1 ! 1-010 - Add code for labeled borders.  RKR 7-APR-1983.
;    69          0069  1 ! 1-011 - Separate bits used for borders.  RKR 15-APR-1983.
;    70          0070  1 ! 1-012 - Move minimal update into its own module along with buffering
;    71          0071  1 !         stuff. RKR 15-APR-1983.
;    72          0072  1 ! 1-013 - Update screen buffer when minimal update is enabled.
;    73          0073  1 !         Use symbolic names for mode bits.
;    74          0074  1 !         STAN 1-May-1983.
;    75          0075  1 ! 1-014 - Clear WCB text and attribute buffers at beginning of
;    76          0076  1 !         SMG$$FILL_WINDOW_BUFFER.  RKR 2-MAY-1983
;    77          0077  1 ! 1-015 - Fix double-dot bug in SMG$CONTROL_MODE.
;    78          0078  1 !         Remove call to SMG$$CONSTRUCT_BORDER_CHAR.
;    79          0079  1 !         STAN 2-May-1983.
;    80          0080  1 ! 1-016 - Flush output when batching ends.
;    81          0081  1 !         New status returns for SMG$END_DISPLAY_UPDATE.
;    82          0082  1 !         STAN 3-May-1983
;    83          0083  1 ! 1-017   Complete overhaul of CHECK_FOR_OUTPUT_DCB, CHECK_FOR_OUTPUT_PBCB,
;    84          0084  1 !         and FILL_WINDOW_BUFFER.
;    85          0085  1 !         Remove some of the obsolete code resulting from prior edits.
;    86          0086  1 !         RKR 4-MAY-1983.
;    87          0087  1 ! 1-018   Clean up loose ends from last edit.  Remove
;    88          0088  1 !         SMG$$CONSTRUCT_BORDER_CHAR.   RKR 4-MAY-1983
;    89          0089  1 ! 1-019   Speed up redrawing of window buffer by making use of
;    90          0090  1 !         PP_V_OCCLUDED bit.  RKR 6-MAY-1983.
;    91          0091  1 ! 1-020   Admired edit 1-019.
;    92          0092  1 !         Set physical (pasteboard) cursor position to the position
;    93          0093  1 !         corresponding to the logical cursor position in the
;    94          0094  1 !         last virtual display that the user referenced.
;    95          0095  1 !         Thus physical cursor position has absolutely nothing
;    96          0096  1 !         to do with the order the displays are pasted in.
;    97          0097  1 !         STAN 8-May-1983
;    98          0098  1 ! 1-021   Write SMG$$INVALIDATE_DISPLAY.
;    99          0099  1 !         STAN 9-May-1983
;   100          0100  1 ! 1-022   Add optimized move for contiguous source and destination in
;   101          0101  1 !         SMG$$FILL_WINDOW_BUFFER and SMG$$CHECK_FOR_OUTPUT_DCB.
;   102          0102  1 !         RKR 9-MAY-1983.
;   103          0103  1 ! 1-023   Force REPAINT_SCREEN to repaint the screen even though it did
;   104          0104  1 !         not think the screen changed.
;   105          0105  1 !         STAN 10-May-1983
;   106          0106  1 ! 1 -024  Fix Draw Border -- problem with border element overwriting
;   107          0107  1 !         cell that has video attributes.
;   108          0108  1 !         RKR 13-May-1983.
;   109          0109  1 ! 1 -025  Take advantage of sizes available in DCB, WCB and PP.  No
;   110          0110  1 !         longer need to multiply number of rows by number of cols.
;   111          0111  1 !         RKR 13-MAY-1983.
;   112          0112  1 ! 1 -026  Repackage inner-most subroutines to allow usage by input
;   113          0113  1 !         support routines.
;   114          0114  1 !         RKR 15-MAY-1983.
```

```
;  115          0115  1 ! 1-027   Add SMG$BEGIN_PASTEBOARD_UPDATE and SMG$END_PASTEBOARD_UPDATE.
;  116          0116  1 !           STAN 16-May-1983.
;  117          0117  1 ! 1-028   Change logic of CHECK_FOR_OUTPUT_PBCB with respect to no
;  118          0118  1 !           pasting packets.
;  119          0119  1 !           RKR 18-MAY-1983.
;  120          0120  1 ! 1-029   Fix CHECK_FOR_OUTPUT_PBCB.  Check for no pasting packets is
;  121          0121  1 !           wrong.
;  122          0122  1 !           RKR 18-MAY-1983.
;  123          0123  1 ! 1-030  Delete references to external DD_ structures and counts --
;  124          0124  1 !           no longer needed.
;  125          0125  1 !           Partial fix up of SMG$RING_BELL -- add display_id argument.
;  126          0126  1 !           RKR 20-MAY-1983.
;  127          0127  1 ! 1-031  Add logic to BEGIN_DISPLAY_UPDATE and END_DISPLAY_UPDATE to
;  128          0128  1 !           back up and restore DCBs.
;  129          0129  1 !           RKR 23-MAY-1983.
;  130          0130  1 ! 1-032  Fix RING_BELL.  TIMES b, reference to STR$DUPL_CHAR.
;  131          0131  1 !           RKR 23-MAY-1983.
;  132          0132  1 ! 1-033  Add code to calc. what part of WCB buffers got changed and
;  133          0133  1 !           leave this info in PBCB for output routines to use.
;  134          0134  1 !           RKR 26-MAY-1983.
;  135          0135  1 ! 1-034  Allow RING_BELL to return SS$_NORMAL if no error occurs.
;  136          0136  1 !           STAN 27-May-1983
;  137          0137  1 ! 1-035  Rename PBCB_B_COLS to PBCB_W_WIDTH.
;  138          0138  1 !           STAN 1-Jun-1983.
;  139          0139  1 ! 1-036  Add some bullet proofing to routine calls that don't check
;  140          0140  1 !           status.  Make SMG$$POINT_IN_RECT into SMG$$POINT_IN_RECT_R3.
;  141          0141  1 !           RKR 8-JUN-1983.
;  142          0142  1 ! 1-037  Make DRAW_BORDER affect the record of what area of the
;  143          0143  1 !           window control block buffer was modified.
;  144          0144  1 !           RKR 9-JUN-1983.
;  145          0145  1 ! 1-038  Make SMG$END_DISPLAY_UPDATE perform the pasting packet
;  146          0146  1 !           constant recalculations if DCB_V_PP_MISMATCH indicates that
;  147          0147  1 !           the SMG$DISPLAY_LINKS routine couldn't do it at the time the
;  148          0148  1 !           change occurred because the display was batched.
;  149          0149  1 !           RKR 17-JUN-1983.
;  150          0150  1 ! 1-039 In CHECK_FOR_OUTPUT_DCB rearrange inner loop to check for
;  151          0151  1 !           pasteboard batching and to use action code to decide whether
;  152          0152  1 !           border needs to be (re)drawn.
;  153          0153  1 !           In END_DISPLAY_UPDATE, pass action code to CHECK_FOR_OUTPUT_DCB.
;  154          0154  1 !           RKR 20-JUN-1983.
;  155          0155  1 ! 1-040 Fix error introduced in last edit.  Advancement to next PP must
;  156          0156  1 !           must be inside outer loop, but not inside inner loop.
;  157          0157  1 !           RKR 21-JUN-1983
;  158          0158  1 ! 1-041 Speed up SMG$$MOVE_TEXT_TO_WINDOW_BUF by moving alternate
;  159          0159  1 !           character set logic into its own incr loop instead of checking
;  160          0160  1 !           for existance of alternate character set each time in main loop.
;  161          0161  1 !           RKR 22-JUN-1983.
;  162          0162  1 ! 1-042 Further speed up of SMG$$MOVE_TEXT_TO_BUF.
;  163          0163  1 !           RKR 22-JUN-1983.
;  164          0164  1 ! 1-043 Smarten up SMG$$END_DISPLAY_UPDATE to recognize situations where
;  165          0165  1 !           it is necessary to redraw the affected pasteboard buffers from
;  166          0166  1 !           the bottom out -- rather than just the display that had its
;  167          0167  1 !           batching lifted.
;  168          0168  1 !           RKR 24-JUN-1983.
;  169          0169  1 ! 1-044 Remove LIB$PUT_SCREEN from the EXTERNAL declaration so that SCRSHR
;  170          0170  1 !           will not be pulled into the SMGSHR shareable image.  PLL 1-Jul-1983
;  171          0171  1 ! 1-045 Add logic to map the line characteristic vector during mapping
```

```
;  172          0172  1 !        operations.
;  173          0173  1 !        RKR 7-JUL-1983.
;  174          0174  1 ! 1-046 Add procedure SMG$FIND_CURSOR_DISPLAY.  Calcs. which virtual
;  175          0175  1 !        display the current physical cursor is in (if any).
;  176          0176  1 !        RKR 27-JUL-1983.
;  177          0177  1 ! 1-047 Rewrite SMG$$MOVE_TEXT_TO_WINDOW_BUF to accomodate double-wide
;  178          0178  1 !        and double-wide/double-high.
;  179          0179  1 !        Add optimization to SMG$$CHECK_FOR_OUTPUT_DCB to treat functions
;  180          0180  1 !        that only change the cursor position as a special case.
;  181          0181  1 !        RKR 31-Aug-1983.
;  182          0182  1 ! 1-048 STAN 31-Aug-1983. Round right rather tha left when pasting a
;  183          0183  1 !        virtual display to an even boundary.
;  184          0184  1 ! 1-049 Fix border positioning for DWDH.
;  185          0185  1 !        Fix places where "first_chang-d_row" and "last_changed_row"
;  186          0186  1 !        incorrectly computed.  RKR 1-SEP-1983.
;  187          0187  1 ! 1-050 Further fixes in SMG$$MOVE_TEXT_TO_WINDOW_BUF wrt DWDH.
;  188          0188  1 !        RKR 6-SEP-1983.
;  189          0189  1 ! 1-051 Fix SMG$$DRAW_BORDER to make border labels come out right
;  190          0190  1 !        under DWDH situations.  RKR 7-SEP-1983.
;  191          0191  1 ! 1-052 Fix SMG$$DRAW_BORDER to pick up rendition bits for border
;  192          0192  1 !        labels.  RKR 15-SEP-1983.
;  193          0193  1 ! 1-053 Allow CLEAR_SCREEN control mode. STAN 24-SEP-1983.
;  194          0194  1 !        Check dummy arguments by name rather than by number.
;  195          0195  1 ! 1-054 Fix MOVE_TEXT_TO_WINDOW_BUFFER to remap border elements when
;  196          0196  1 !        there is a transition from normal to DWDH or vice versa.
;  197          0197  1 !        (STAN: Repaint clears line characteristics.)
;  198          0198  1 !        RKR 11-OCT-1983.
;  199          0199  1 ! 1-055 Fix DRAW_BORDER for DHDW.  RKR 12-OCT-1983.
;  200          0200  1 ! 1-056 Replace critial loop by a SKPC in SMG$$MOVE_TEXT_TO_WINDOW_BUF.
;  201          0201  1 !        RKR 9-NOV-1983.
;  202          0202  1 ! 1-057 Changes to SMG$$DRAW_BORDER and SMG$$MOVE_TEXT_TO_WINDOW_BUF
;  203          0203  1 !        to not do special mapping for DW/DWDH if device doesn't
;  204          0204  1 !        support this action.
;  205          0205  1 !        RKR 10-NOV-1983.
;  206          0206  1 ! 1-058 Make SMG$$CHECK_FOR_OUTPUT_PBCB initialize WCB [WCB_A_CHAR_SET_BUF]
;  207          0207  1 !        if one exists.  RKR 28-NOV-1983.
;  208          0208  1 ! 1-059 Make SMG$BEGIN_DISPLAY_UPDATE call SMG$$DUPL_VIRTUAL_DISPLAY
;  209          0209  1 !        unconditionally.  SMG$$DUPL_VIRTUAL_DISPLAY now has the smarts to
;  210          0210  1 !        decide whether a new DCB needs to be created or whether only current
;  211          0211  1 !        context needs to be preserved in an already-existing backup DCB.
;  212          0212  1 !        RKR 28-NOV-1983.
;  213          0213  1 ! 1-060 Introduce inner routine SMG$$BEGIN_PASTEBOARD_UPDATE_R1 and
;  214          0214  1 !        SMG$$END_PASTEBOARD_UPDATE_R2.
;  215          0215  1 !        RKR 2-DEC-1983.
;  216          0216  1 ! 1-061 Use new routine SMG$$UPDATE_PHYSICAL_CURSOR to fix physical cursor
;  217          0217  1 !        position in cursor-positioning-only path through
;  218          0218  1 !        SMG$$CHECK_FOR_OUTPUT_DCB.
;  219          0219  1 !        RKR 15-DEC-1983.
;  220          0220  1 ! 1-062 Fix SMG$END_DISPLAY_UPDATE.  It was returning garbage if the batching
;  221          0221  1 !        level was already 0.
;  222          0222  1 !        RKR 16-DEC-1983.
;  223          0223  1 ! 1-063 Fix SIGNED/UNSIGNED problems in SMG$$POINT_IN_RECT_R3 and in
;  224          0224  1 !        SMG$$OCCLUDE.  Change linkage to SMG$$POINT_IN_RECT_R3.
;  225          0225  1 !        RKR 17-Jan-1984.
;  226          0226  1 ! 1-064 Fix SMG$$DRAW_BORDER for cases where virtual display is:
;  227          0227  1 !        a). Wider than pasteboard and pasted at 0 or negative col  and
;  228          0228  1 !        b). Higher than pasteboard and pasted at 0 or negative row
```

```
;  229          0229  1 !        RKR 23-Jan-1984.
;  230          0230  1 ! 1-065 Make SMG$$DRAW_BORDER abandon attempt to draw border altogether if none
;  231          0231  1 !        of the virtual display maps onto the visible part of the pasteboard.
;  232          0232  1 !        RKR 24-Feb-1984.
;  233          0233  1 ! 1-066 Add routine SMG$GET_CHAR_AT_PHYSICAL_CURSOR.  RKR  28-FEB-1984.
;  234          0234  1 ! 1-067 Move INVALIDATE_DISPLAY to file SMGPRVINP.B32. STAN 7-Mar-1984.
;  235          0235  1 ! 1-068 Update header of CONTROL_MODE to mention the NOTABS bit and CLEAR_SCREEN.
;  236          0236  1 !        STAN 14-Mar-1984.  Add public entry point SMG$INVALIDATE_DISPLAY.
;  237          0237  1 ! 1-069 If only one line of a DCB changed, tell that to output, instead
;  238          0238  1 !        of erroneously telling output that the whole display changed.
;  239          0239  1 !        STAN 5-May-1984.
;  240          0240  1 ! 1-070 Make horizontal borders knock down DHDW characteristic. STAN 17-Jun-1984.
;  241          0241  1 ! 1-071 Fix to SMG$CONTROL_MODE to allow notabs bit.  PLL 28-Jun-1984
;  242          0242  1 ! 1-072 Tighten up some consistency checks on the data structures.  It was
;  243          0243  1 !        possible to get in an endless loop while walking the pasting packet
;  244          0244  1 !        chain.  PLL 18-Jul-1984
;  245          0245  1 !--
```

```
;  247          0246  1 %SBTTL 'Declarations'
;  248          0247  1 !
;  249          0248  1 ! SWITCHES:
;  250          0249  1 !
;  251          0250  1
;  252          0251  1 !
;  253          0252  1 ! LINKAGES:
;  254          0253  1 !
;  255          0254  1 LINKAGE
;  256          0255  1     POINT_IN_RECT_LINK = JSB ( REGISTER = 0, REGISTER = 1, REGISTER = 2) :
;  257          0256  1                         NOPRESERVE ( 3)
;  258          0257  1                         NOTUSED (4,5,6,7,8,9,10,11);
;  259          0258  1 !
;  260          0259  1 ! INCLUDE FILES
;  261          0260  1 !
;  262          0261  1
;  263          0262  1 REQUIRE 'RTLIN:SMGPROLOG.REQ';          ! defines psects, macros, tcb,
;  264          0340  1                                         !  wcb, & terminal symbols
;  265          0341  1 !
;  266          0342  1 ! TABLE OF CONTENTS:
;  267          0343  1 !
;  268          0344  1
;  269          0345  1 FORWARD ROUTINE
;  270          0346  1
;  271          0347  1 ! Public entry points:
;  272          0348  1
;  273          0349  1     SMG$BEGIN_DISPLAY_UPDATE,           ! Advance buffering level count for
;  274          0350  1                                         ! a virtual display
;  275          0351  1
;  276          0352  1     SMG$BEGIN_PASTEBOARD_UPDATE,        ! Advance buffering level count for
;  277          0353  1                                         ! a pasteboard
;  278          0354  1
;  279          0355  1     SMG$CONTROL_MODE,                   ! Control operational modes
;  280          0356  1
;  281          0357  1
;  282          0358  1     SMG$END_DISPLAY_UPDATE,             ! Decr. buffering level count
;  283          0359  1                                         ! and flush to screen if it
;  284          0360  1                                         ! reaches zero.
;  285          0361  1
;  286          0362  1     SMG$END_PASTEBOARD_UPDATE,          ! Decr. buffering level count
;  287          0363  1                                         ! for a pasteboard, and flush to
;  288          0364  1                                         ! screen if it reaches 0.
;  289          0365  1
;  290          0366  1     SMG$FIND_CURSOR_DISPLAY,            ! Find the display which
;  291          0367  1                                         ! contains the current physical
;  292          0368  1                                         ! cursor position (if any).
;  293          0369  1
;  294          0370  1     SMG$GET_CHAR_AT_PHYSICAL_CURSOR,    ! Return char at physical cursor
;  295          0371  1                                         ! location
;  296          0372  1
;  297          0373  1     SMG$INVALIDATE_DISPLAY,             ! Mark contents of display as unknown.
;  298          0374  1
;  299          0375  1     SMG$REPAINT_SCREEN,                 ! Repaint the entire screen the
;  300          0376  1                                         ! way our internal database says
;  301          0377  1                                         ! it should look.
;  302          0378  1
;  303          0379  1     SMG$RING_BELL,                      ! Ring bell
```

```
;  304     0380  1
;  305     0381  1 ! Private entry points:
;  306     0382  1
;  307     0383  1     SMG$$BEGIN_PASTEBOARD_UPDATE_R1 : SMG$$BEGIN_PBD_UPDATE$LNK,
;  308     0384  1                                       ! Inner BEGIN_PASTEBOARD_UPDATE
;  309     0385  1
;  310     0386  1     SMG$$CHECK_FOR_OUTPUT_DCB,        ! Check to see if a virtual
;  311     0387  1                                       ! display needs to be flushed to
;  312     0388  1                                       ! screen.
;  313     0389  1
;  314     0390  1     SMG$$CHECK_FOR_OUTPUT_PBCB,       ! Refresh everything on this
;  315     0391  1                                       ! pasteboard -- triggered by
;  316     0392  1                                       ! an unpaste.
;  317     0393  1
;  318     0394  1     SMG$$DRAW_BORDER,                 ! Move border characters into
;  319     0395  1                                       ! WCB text buffer.
;  320     0396  1
;  321     0397  1     SMG$$END_PASTEBOARD_UPDATE_R2 : SMG$$END_PBD_UPDATE$LNK,
;  322     0398  1                                       ! Inner END_PASTEBOARD_UPDATE
;  323     0399  1
;  324     0400  1     SMG$$FILL_WINDOW_BUFFER,          ! Fill the window buffer
;  325     0401  1                                       ! associated with a pasteboard
;  326     0402  1                                       ! with images of the virtual
;  327     0403  1                                       ! displays that map into it.
;  328     0404  1
;  329     0405  1     SMG$$MOVE_TEXT_TO_WINDOW_BUF,     ! move text from DCB buffers to
;  330     0406  1                                       ! WCB window buffers.
;  331     0407  1
;  332     0408  1     SMG$$OCCLUDE,                     ! Check whether two rectangular
;  333     0409  1                                       ! areas occlude each other.
;  334     0410  1
;  335     0411  1     SMG$$POINT_IN_RECT_R3 : POINT_IN_RECT_LINK; ! Returns codes
;  336     0412  1                                       ! indicating where a
;  337     0413  1                                       ! given point is with
;  338     0414  1                                       ! respect to a given
;  339     0415  1                                       ! rectangular area.
;  340     0416  1
;  341     0417  1
;  342     0418  1 !
;  343     0419  1 ! EXTERNAL REFERENCES
;  344     0420  1 !
;  345     0421  1
;  346     0422  1 EXTERNAL
;  347     0423  1     PBD_L_COUNT,          ! No. of pasteboards we currently have
;  348     0424  1
;  349     0425  1     PBD_A_PBCB : VECTOR [PBD_K_MAX_PB, LONG],
;  350     0426  1                           ! Table of addresses of PBCB's
;  351     0427  1
;  352     0428  1     PBD_V_PB_AVAIL : BITVECTOR [PBD_K_MAX_PB];
;  353     0429  1                           ! Bit vector or pasteboard id numbers in use.
;  354     0430  1
;  355     0431  1 EXTERNAL LITERAL
;  356     0432  1
;  357     0433  1     SMG$_BATSTIPRO,       ! Success; but batching is still in progress.
;  358     0434  1     SMG$_BATWASOFF,       ! Success; but batching was already off
;  359     0435  1     SMG$_BATWAS_ON,       ! Success; but batching was already on
;  360     0436  1     SMG$_FATERRLIB,       ! Fatal error in library procedure
```

```
;  361     0437  1      SMG$_INVARG,          ! Invalid argument
;  362     0438  1      SMG$_INVCOL,          ! Invalid column number
;  363     0439  1      SMG$_INVDIS_ID,       ! Invalid virtual display id
;  364     0440  1      SMG$_INVPAS_ID,       ! Invalid pasteboard id
;  365     0441  1      SMG$_INVROW;          ! Invalid row number
;  366     0442  1
;  367     0443  1  EXTERNAL ROUTINE
;  368     0444  1      LIB$ESTABLISH,                ! Used to establish a handler
;  369     0445  1      LIB$FREE_VM,                  ! Deallocate heap storage
;  370     0446  1      LIB$GET_VM,                   ! Allocate heap storage
;  371     0447  1      LIB$SFREE1_DD,                ! Free a dynamic string
;  372     0448  1      LIB$SIG_TO_RET,               ! Handler to turn signals into return
;  373     0449  1                                    ! statuses.
;  374     0450  1      STR$DUPL_CHAR,                ! Make a string of N copies of a byte.
;  375     0451  1      SMG$$DUPL_VIRTUAL_DISPLAY,    ! Make a copy of a DCB with a new
;  376     0452  1                                    ! display id.
;  377     0453  1      SMG$$FLUSH_BUFFER,            ! Flush remaining buffered output
;  378     0454  1      SMG$$INVALIDATE_DISPLAY,      ! Tell SMG that user has
;  379     0455  1                                    ! written into this display on his own.
;  380     0456  1      SMG$$MIN_UPD,                 ! Minimal update output routine
;  381     0457  1      SMG$$OUTPUT,                  ! Output a string to a pasteboard device
;  382     0458  1      SMG$$RECALC_PP_FIELDS,        ! Recalc. PP fields after batching level
;  383     0459  1                                    ! lifted on a DCB
;  384     0460  1      SMG$$UPDATE_PHYSICAL_CURSOR;  ! Update physical cursor position
;  385     0461  1
;  386     0462  1  !<BLF/PAGE>
```

```
;  388          0463  1 %SBTTL 'SMG$BEGIN_DISPLAY_UPDATE - Begin batch of updates to display'
;  389          0464  1 GLOBAL ROUTINE SMG$BEGIN_DISPLAY_UPDATE ( DISPLAY_ID ) =
;  390          0465  1 !++
;  391          0466  1 ! FUNCTIONAL DESCRIPTION:
;  392          0467  1 !
;  393          0468  1 !       Disable outputing this virtual display to the screen until the
;  394          0469  1 !       matching call to SMG$END_DISPLAY_UPDATE is encountered.
;  395          0470  1 !
;  396          0471  1 ! CALLING SEQUENCE:
;  397          0472  1 !
;  398          0473  1 !       ret_status.wlc.v = SMG$BEGIN_DISPLAY_UPDATE ( DISPLAY_ID.rl.r)
;  399          0474  1 !
;  400          0475  1 ! FORMAL PARAMETERS:
;  401          0476  1 !
;  402          0477  1 !       DISPLAY_ID.rl.r         Display id of virtual display.
;  403          0478  1 !
;  404          0479  1 ! IMPLICIT INPUTS:
;  405          0480  1 !
;  406          0481  1 !       NONE
;  407          0482  1 !
;  408          0483  1 ! IMPLICIT OUTPUTS:
;  409          0484  1 !
;  410          0485  1 !       NONE
;  411          0486  1 !
;  412          0487  1 ! COMPLETION STATUS:
;  413          0488  1 !
;  414          0489  1 !       SS$_NORMAL      Normal successful completion, batching has been
;  415          0490  1 !                       initiated
;  416          0491  1 !       SMG$_BATWAS_ON  Success, but note that batching was already on
;  417          0492  1 !       SMG$_INVDIS_ID  Invalid Display Id
;  418          0493  1 !
;  419          0494  1 ! SIDE EFFECTS:
;  420          0495  1 !
;  421          0496  1 !       NONE
;  422          0497  1 !--
;  423          0498  1
;  424          0499  2     BEGIN
;  425          0500  2     LOCAL
;  426          0501  2         DCB : REF BLOCK [,BYTE];        ! Addr of display control block
;  427          0502  2
;  428          0503  2 !+
;  429          0504  2 ! Check for right number of arguments.
;  430          0505  2 !-
;  431          0506  2     $SMG$VALIDATE_ARGCOUNT ( 1, 1);
;  432          0507  2
;  433          0508  2     $SMG$GET_DCB (.DISPLAY_ID, DCB);    ! Get DCB address
;  434          0509  2
;  435          0510  2 !+
;  436          0511  2 ! Increment count of number of SMG$END_DISPLAY_UPDATE calls we need to
;  437          0512  2 ! see before we resume outputing from this virtual display.
;  438          0513  2 ! Let the user know what the previous state of batching was by
;  439          0514  2 ! our status return (in case he's interested).
;  440          0515  2 !-
;  441          0516  2     DCB [DCB_L_BATCH_LEVEL] = .DCB [DCB_L_BATCH_LEVEL] + 1;
;  442          0517  2
;  443          0518  2     IF .DCB [DCB_L_BATCH_LEVEL] EQL 1
;  444          0519  2     THEN
```

```
;  445          0520  3          BEGIN   ! Begin of batching operation
;  446          0521  3          LOCAL
;  447          0522  3              STATUS,                         ! Status of subroutine calls
;  448          0523  3              PP : REF BLOCK [,BYTE];         ! Addr of a pasting packet
;  449          0524  3
;  450          0525  3          !+
;  451          0526  3          ! Make a copy of current DCB and leave its address in the
;  452          0527  3          ! backup pointer field of the current DCB.
;  453          0528  3          !-
;  454          0529  4          IF NOT (STATUS = SMG$$DUPL_VIRTUAL_DISPLAY (
;  455          0530  4                                  DCB,                    ! Addr of curr.
;  456          0531  4                                  DCB [DCB_A_BACKUP_DCB])) ! Where new
;  457          0532  4                                                           ! get stored.
;  458          0533  3          THEN
;  459          0534  3              RETURN (.STATUS);
;  460          0535  3
;  461          0536  3          !+
;  462          0537  3          ! Walk chain of all pasteboards that we are pasted to and for
;  463          0538  3          ! each encountered, in the affected pasting packet, set the
;  464          0539  3          ! back pointer to the DCB to be the new DCB we've created.
;  465          0540  3          ! This causes all mapping operations to reference the backed up
;  466          0541  3          ! DCB as the source of images for the screen.  This is the
;  467          0542  3          ! desired action since the current DCB is undergoing changes
;  468          0543  3          ! which should not yet find their way to the screen -- its
;  469          0544  3          ! batched.
;  470          0545  3          !-
;  471          0546  3          PP = .DCB [DCB_A_PP_NEXT];
;  472          0547  3
;  473          0548  3          IF .PP EQL 0
;  474          0549  3          THEN
;  475          0550  3              RETURN (SMG$_FATERRLIB);            ! should never be 0
;  476          0551  3                                                  ! (points to self when empty)
;  477          0552  3
;  478          0553  3          WHILE .PP NEQ DCB [DCB_A_PP_NEXT]       ! While any remain
;  479          0554  3          DO
;  480          0555  4              BEGIN
;  481          0556  4              PP [PP_A_DCB_ADDR] = .DCB [DCB_A_BACKUP_DCB];
;  482          0557  4              PP = .PP [PP_A_NEXT_DCB];   ! Step to next packet
;  483          0558  3              END;
;  484          0559  3          RETURN  SS$_NORMAL;
;  485          0560  3
;  486          0561  3              END     ! Begin of batching operation
;  487          0562  3
;  488          0563  2      ELSE
;  489          0564  2
;  490          0565  2          RETURN  SMG$_BATWAS_ON
;  491          0566  2
;  492          0567  1      END;                ! End of routine SMG$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      2C      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>
```

```
;   547          0620  1 %SBTTL 'SMG$END_DISPLAY_UPDATE - End batch of updates to display'
;   548          0621  1 GLOBAL ROUTINE SMG$END_DISPLAY_UPDATE ( DISPLAY_ID ) =
;   549          0622  1 !++
;   550          0623  1 ! FUNCTIONAL DESCRIPTION:
;   551          0624  1 !
;   552          0625  1 !       Reduce the number of calls to SMG$END_DISPLAY_UPDATE that will
;   553          0626  1 !       be needed before we resume outputing from this display.  If
;   554          0627  1 !       this call makes this count go to zero, flush the display to
;   555          0628  1 !       the screen.
;   556          0629  1 !       If the level is already 0, we return a success status
;   557          0630  1 !       informing caller that the batching level was already 0,
;   558          0631  1 !       but we otherwise allow this.  This gives a user a guaranteed
;   559          0632  1 !       method of ending batching.
;   560          0633  1 !
;   561          0634  1 ! CALLING SEQUENCE:
;   562          0635  1 !
;   563          0636  1 !       ret_status.wlc.v = SMG$END_DISPLAY_UPDATE ( DISPLAY_ID.rl.r)
;   564          0637  1 !
;   565          0638  1 ! FORMAL PARAMETERS:
;   566          0639  1 !
;   567          0640  1 !       DISPLAY_ID.rl.r         Display id of virtual display.
;   568          0641  1 !
;   569          0642  1 ! IMPLICIT INPUTS:
;   570          0643  1 !
;   571          0644  1 !       DCB [DCB_L_BATCH_LEVEL]
;   572          0645  1 !
;   573          0646  1 ! IMPLICIT OUTPUTS:
;   574          0647  1 !
;   575          0648  1 !       DCB [DCB_L_BATCH_LEVEL] gets decremented
;   576          0649  1 !
;   577          0650  1 ! COMPLETION STATUS:
;   578          0651  1 !
;   579          0652  1 !       SS$_NORMAL      Normal successful completion
;   580          0653  1 !       SMG$_BATSTIPRO  Success; but batching is still in progress.
;   581          0654  1 !       SMG$_INVDIS_ID  Invalid Display Id
;   582          0655  1 !       SMG$_BATWASOFF  Success; but batching was already off
;   583          0656  1 !
;   584          0657  1 ! SIDE EFFECTS:
;   585          0658  1 !
;   586          0659  1 !       NONE
;   587          0660  1 !--
;   588          0661  1
;   589          0662  2     BEGIN
;   590          0663  2     LOCAL
;   591          0664  2         STATUS,                             ! Status to return
;   592          0665  2         DCB : REF BLOCK [,BYTE];            ! Addr of display control block
;   593          0666  2
;   594          0667  2 !+
;   595          0668  2 ! Check for right number of arguments.
;   596          0669  2 !-
;   597          0670  2     $SMG$VALIDATE_ARGCOUNT ( 1, 1);
;   598          0671  2
;   599          0672  2     $SMG$GET_DCB (.DISPLAY_ID, DCB);    ! Get DCB address
;   600          0673  2
;   601          0674  2 !+
;   602          0675  2 ! If current count is greater than 1, simply reduce it by one and
;   603          0676  2 ! return to caller.  If it is one, reduce it to zero and cause the
```

```
;  604          0677  2 ! current contents of this window to be flushed to the screen (if need
;  605          0678  2 ! be).
;  606          0679  2 !-
;  607          0680  2     IF .DCB [DCB_L_BATCH_LEVEL] GTR 1
;  608          0681  2     THEN
;  609          0682  3         BEGIN
;  610          0683  3         DCB [DCB_L_BATCH_LEVEL] = .DCB [DCB_L_BATCH_LEVEL] - 1;
;  611          0684  3         STATUS = SMG$_BATSTIPRO;          ! Ok, but batching is still in
;  612          0685  3         END                               ! progress
;  613          0686  3
;  614          0687  2     ELSE
;  615          0688  2
;  616          0689  3         BEGIN   ! Level is currently 0 or 1
;  617          0690  3         LOCAL
;  618          0691  3             BATCH_STATUS;
;  619          0692  3
;  620          0693  3         BATCH_STATUS = SS$_NORMAL;
;  621          0694  3         IF .DCB [DCB_L_BATCH_LEVEL] EQL 0
;  622          0695  3         THEN
;  623          0696  4             RETURN ( SMG$_BATWASOFF)     ! Ok, but batching was already off
;  624          0697  3         ELSE
;  625          0698  3
;  626          0699  4             BEGIN       ! Reduction from 1 to 0
;  627          0700  4             LOCAL
;  628          0701  4                 PP : REF $PP_DECL;       ! Addr of a pasting packet
;  629          0702  4
;  630          0703  4             DCB [DCB_L_BATCH_LEVEL] = 0;
;  631          0704  4             !+
;  632          0705  4             ! Walk chain of pastboards to which we are pasted.  For each
;  633          0706  4             ! pasting packet involved, set the pointer in the pasting
;  634          0707  4             ! packet back to the original DCB since it is now free of
;  635          0708  4             ! batching and its contents can be mapped to the screen.
;  636          0709  4             ! Note: We do not deallocate the backup DCB.  The assumption
;  637          0710  4             ! is that if the caller batched this virtual display once,
;  638          0711  4             ! he is likely to do it again.  This saves us having to
;  639          0712  4             ! deallocate now only to have to do another
;  640          0713  4             ! SMG$$DUPL_VIRTUAL_DISPLAY when he starts batching again.
;  641          0714  4             !-
;  642          0715  4             PP = .DCB [DCB_A_PP_NEXT];
;  643          0716  4
;  644          0717  4             IF .PP EQL 0
;  645          0718  4             THEN
;  646          0719  4                 RETURN (SMG$_FATERRLIB);         ! should never be 0
;  647          0720  4
;  648          0721  4             WHILE .PP NEQ DCB [DCB_A_PP_NEXT]   ! While any remain...
;  649          0722  4             DO
;  650          0723  5                 BEGIN
;  651          0724  5                 PP [PP_A_DCB_ADDR] = .DCB;       ! To orig. DCB
;  652          0725  5                 PP = .PP [PP_A_NEXT_DCB];        ! To next packet
;  653          0726  4                 END;
;  654          0727  4
;  655          0728  4             !+
;  656          0729  4             ! If DCB_V_PP_MISMATCH is set, a change has been made to
;  657          0730  4             ! this DCB which requires the pasting packet constants to be
;  658          0731  4             ! recalculated.  However, the DCB was "batched" at the time
;  659          0732  4             ! the change took place, so we do the pasting packet update
;  660          0733  4             ! now as part of the unbatching process.
```

```
;  661          0734  4               !-
;  662          0735  4               IF .DCB [DCB_V_PP_MISMATCH]
;  663          0736  4               THEN
;  664          0737  5                   BEGIN   ! Unbatching clean up
;  665          0738  5                   LOCAL
;  666          0739  5                       CURR_PP : REF $PP_DECL; ! Addr of a pasting packet
;  667          0740  5
;  668          0741  6                   IF NOT (STATUS = SMG$$RECALC_PP_FIELDS (.DCB))
;  669          0742  5                   THEN
;  670          0743  5                       RETURN (.STATUS);
;  671          0744  5
;  672          0745  5                   DCB [DCB_V_PP_MISMATCH] = 0 ; ! Knock down flag
;  673          0746  5
;  674          0747  5                   !+
;  675          0748  5                   ! Since DCB_V_PP_MISMATCH was set, the change to the DCB
;  676          0749  5                   ! included dimensional changes.  This means we will have
;  677          0750  5                   ! to remap the whole pasteboard buffer from the bottom
;  678          0751  5                   ! outward -- for each pasteboard that we are pasted to.
;  679          0752  5                   !-
;  680          0753  5                   CURR_PP = .DCB [DCB_A_PP_NEXT];
;  681          0754  5
;  682          0755  5                   IF .CURR_PP EQL 0
;  683          0756  5                   THEN
;  684          0757  5                       RETURN (SMG$_FATERRLIB);    ! should never be 0
;  685          0758  5
;  686          0759  5                   WHILE .CURR_PP NEQ DCB [DCB_A_PP_NEXT]
;  687          0760  5                   DO
;  688          0761  6                       BEGIN       ! For all pasteboards
;  689          0762  6                       LOCAL
;  690          0763  6                           PBCB : REF $PBCB_DECL;  ! Addr of a pasteboard
;  691          0764  6                                                   ! control block
;  692          0765  6
;  693          0766  6                       PBCB = .CURR_PP [PP_A_PBCB_ADDR];
;  694          0767  7                       IF NOT (STATUS = SMG$$CHECK_FOR_OUTPUT_PBCB (.PBCB))
;  695          0768  6                       THEN
;  696          0769  6                           RETURN (.STATUS); ! Quit on first failure
;  697          0770  6
;  698          0771  6                       CURR_PP = .CURR_PP [PP_A_NEXT_DCB]; ! To next packet
;  699          0772  5                       END;        ! For all pasteboards
;  700          0773  5
;  701          0774  5                   RETURN (SS$_NORMAL);
;  702          0775  5                   END     ! Unbatching clean up
;  703          0776  5
;  704          0777  4               ELSE
;  705          0778  4
;  706          0779  5                   BEGIN   ! No cleanup needed
;  707          0780  5                   !+
;  708          0781  5                   ! Call SMG$$CHECK_FOR_OUTPUT_DCB to cause the contents
;  709          0782  5                   ! of this virtual display to be flushed to the screen.
;  710          0783  5                   ! If that fails, then return its status as our status.
;  711          0784  5                   ! If that succeeded then return our previously
;  712          0785  5                   ! calculated status.
;  713          0786  5                   !-
;  714          0787  5                   STATUS = SMG$$CHECK_FOR_OUTPUT_DCB (.DCB,
;  715          0788  5                                                       SMG$C_END_DISPLAY_UPDATE);
;  716          0789  5                   IF .STATUS THEN STATUS=.BATCH_STATUS
;  717          0790  4                   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$END_DIS^ AY_UPDATE, Save R2,R3,R4      ; 0621
                                    01          6C  91 00002        CMPB    (AP), #T                                   ; 0670
                                                08  13 00005        BEQL    1$                                         ;
                                    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>
```

```
;  725     0797  1 %SBTTL 'SMG$END_PASTEBOARD_UPDATE - End batch of updates to pasteboard'
;  726     0798  1 GLOBAL ROUTINE SMG$END_PASTEBOARD_UPDATE ( PBID ) =
;  727     0799  1 !++
;  728     0800  1 ! FUNCTIONAL DESCRIPTION:
;  729     0801  1 !
;  730     0802  1 !       Reduce the number of calls to SMG$END_PASTEBOARD_UPDATE that will
;  731     0803  1 !       be needed before we resume outputing from this pasteboard.  If
;  732     0804  1 !       this call makes this count go to zero, flush the pasteboard to
;  733     0805  1 !       the screen.
;  734     0806  1 !       If the level is already 0, we return a success status
;  735     0807  1 !       informing caller that the batching level was already 0,
;  736     0808  1 !       but we otherwise allow this.  This gives a user a guaranteed
;  737     0809  1 !       method of ending batching.
;  738     0810  1 !
;  739     0811  1 ! CALLING SEQUENCE:
;  740     0812  1 !
;  741     0813  1 !       ret_status.wlc.v = SMG$END_PASTEBOARD_UPDATE ( PBID.rl.r)
;  742     0814  1 !
;  743     0815  1 ! FORMAL PARAMETERS:
;  744     0816  1 !
;  745     0817  1 !       PBID.rl.r               Pasteboard id.
;  746     0818  1 !
;  747     0819  1 ! IMPLICIT INPUTS:
;  748     0820  1 !
;  749     0821  1 !       PBCB [PBCB_L_BATCH_LEVEL]
;  750     0822  1 !
;  751     0823  1 ! IMPLICIT OUTPUTS:
;  752     0824  1 !
;  753     0825  1 !       PBCB [PBCB_L_BATCH_LEVEL] gets decremented
;  754     0826  1 !
;  755     0827  1 ! COMPLETION STATUS:
;  756     0828  1 !
;  757     0829  1 !       SS$_NORMAL      Normal successful completion
;  758     0830  1 !       SMG$_BATSTIPRO  Success; but batching is still in progress.
;  759     0831  1 !       SMG$_INVPAS_ID  Invalid Pasteboard Id
;  760     0832  1 !       SMG$_BATWASOFF  Success; but batching was already off
;  761     0833  1 !
;  762     0834  1 ! SIDE EFFECTS:
;  763     0835  1 !
;  764     0836  1 !       NONE
;  765     0837  1 !--
;  766     0838  1
;  767     0839  2     BEGIN
;  768     0840  2     LOCAL
;  769     0841  2         STATUS,                         ! Status to return
;  770     0842  2         PBCB : REF BLOCK [,BYTE];       ! Addr of pasteboard control block
;  771     0843  2
;  772     0844  2 !+
;  773     0845  2 ! Check for right number of arguments.
;  774     0846  2 !-
;  775     0847  2     $SMG$VALIDATE_ARGCOUNT ( 1, 1);
;  776     0848  2
;  777     0849  2     $SMG$GET_PBCB (.PBID, PBCB);        ! Get PBCB address
;  778     0850  2
;  779     0851  2     RETURN (SMG$$END_PASTEBOARD_UPDATE_R2 (.PBCB)); ! Do work in inner routine
;  780     0852  2
;  781     0853  1     END;                    ! End of routine SMG$END_PASTEBOARD_UPDATE
```

```
                                          0004 00000          .ENTRY  SMG$END_PASTEBOARD_UPDATE, Save R2          ; 0798
                                01            6C  91 00002          CMPB    (AP), #1                            ; 0847
                                              08  13 00005          BEQL    1$                                  ;
                                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>
```

```
;  784          0855  1 %SBTTL 'SMG$FIND_CURSOR_DISPLAY - Find which virtual display contains physical cursor'
;  785          0856  1 GLOBAL ROUTINE SMG$FIND_CURSOR_DISPLAY (
;  786          0857  1                                                 PASTEBOARD_ID,
;  787          0858  1                                                 DISPLAY_ID
;  788          0859  1                                                     ) =
;  789          0860  1 !++
;  790          0861  1 ! FUNCTIONAL DESCRIPTION:
;  791          0862  1 !
;  792          0863  1 !       This routine determines which virtual display contains the
;  793          0864  1 !       current physical cursor on the screen.  The pasted virtual
;  794          0865  1 !       displays are searched from the deepest pasted one to the
;  795          0866  1 !       outer-most pasted one.  The outer-most virtual display that
;  796          0867  1 !       encompasses the physical cursor is the one selected and its
;  797          0868  1 !       display id is returned.
;  798          0869  1 !
;  799          0870  1 !       It is possible that no virtual display contains the cursor,
;  800          0871  1 !       in which case a display id of zero (an invalid display id) is
;  801          0872  1 !       returned.
;  802          0873  1 !
;  803          0874  1 ! CALLING SEQUENCE:
;  804          0875  1 !
;  805          0876  1 !       ret_status.wlc.v = SMG$FIND_CURSOR_DISPLAY (
;  806          0877  1 !                                           PASTEBOARD_ID.rl.r,
;  807          0878  1 !                                           DISPLAY_ID.wl.r)
;  808          0879  1 !
;  809          0880  1 ! FORMAL PARAMETERS:
;  810          0881  1 !
;  811          0882  1 !       PASTEBOARD_ID.rl.r      Address of a longword containing the
;  812          0883  1 !                               pasteboard_id for the device for which
;  813          0884  1 !                               the information is desired.
;  814          0885  1 !
;  815          0886  1 !       DISPLAY_ID.wl.r         Address of a longword to receive the
;  816          0887  1 !                               virtual display_id of the virtual
;  817          0888  1 !                               display which contains the cursor.
;  818          0889  1 !                               The outer-most pasted virtual display
;  819          0890  1 !                               that contains the cursor is selected.
;  820          0891  1 !                               If no virtual display can be found,
;  821          0892  1 !                               a display id of 0 ( an invalid display
;  822          0893  1 !                               id) is supplied.
;  823          0894  1 !
;  824          0895  1 !
;  825          0896  1 ! IMPLICIT INPUTS:
;  826          0897  1 !
;  827          0898  1 !       NONE
;  828          0899  1 !
;  829          0900  1 ! IMPLICIT OUTPUTS:
;  830          0901  1 !
;  831          0902  1 !       NONE
;  832          0903  1 !
;  833          0904  1 ! COMPLETION STATUS:
;  834          0905  1 !
;  835          0906  1 !       SS$_NORMAL          Normal successful completion
;  836          0907  1 !       SMG$_INVPAS_ID      Invalid pasteboard id
;  837          0908  1 !       SMG$_WRONUMARG      Wrong number of arguments
;  838          0909  1 !
;  839          0910  1 ! SIDE EFFECTS:
;  840          0911  1 !
```

```
;  841          0912  1 !       NONE
;  842          0913  1 !--
;  843          0914  2     BEGIN
;  844          0915  2     LOCAL
;  845          0916  2         PBCB : REF $PBCB_DECL,  ! Address of pasteboard control block
;  846          0917  2
;  847          0918  2         WCB  : REF $WCB_DECL,   ! Address of window control block
;  848          0919  2
;  849          0920  2         PP;                     ! An address pointing at the PP queue
;  850          0921  2                                 ! header in a pasting packet.  This
;  851          0922  2                                 ! address is not the base of the
;  852          0923  2                                 ! pasting packet.
;  853          0924  2 !+
;  854          0925  2 ! Check for right number of arguments.
;  855          0926  2 !-
;  856          0927  2     $SMG$VALIDATE_ARGCOUNT ( 2, 2);
;  857          0928  2
;  858          0929  2 !+
;  859          0930  2 ! Get address of pasteboard control block and the WCB that goes with it.
;  860          0931  2 !-
;  861          0932  2     $SMG$GET_PBCB ( .PASTEBOARD_ID, PBCB );
;  862          0933  2     WCB = .PBCB [PBCB_A_WCB];
;  863          0934  2
;  864          0935  2 !+
;  865          0936  2 ! Set up an answer of none in case search turns up nothing.
;  866          0937  2 !-
;  867          0938  2     .DISPLAY_ID = 0;
;  868          0939  2
;  869          0940  2 !+
;  870          0941  2 ! Search all the pasting packets for this pasteboard, from the deepest-
;  871          0942  2 ! pasted one to the outer-most pasted one.  For each such pasting
;  872          0943  2 ! packet, see if the physical cursor position lies inside the
;  873          0944  2 ! mapping of the associated virtual display.  For each virtual display
;  874          0945  2 ! found, set up output parameter.  The last one found (if any) will be
;  875          0946  2 ! the one the caller sees.
;  876          0947  2 !-
;  877          0948  2     PP = .PBCB [PBCB_A_PP_PREV]; ! Start with inner-most pasted one
;  878          0949  2
;  879          0950  2     IF .PP EQL 0
;  880          0951  2     THEN
;  881          0952  2         RETURN (SMG$_FATERRLIB);        ! should never be 0
;  882          0953  2
;  883          0954  2     WHILE .PP NEQ PBCB [PBCB_A_PP_NEXT]
;  884          0955  2     DO
;  885          0956  3         BEGIN   ! For all pasting packets
;  886          0957  3         LOCAL
;  887          0958  3             PP_BASE : REF $PP_DECL;     ! Base address of a pasting
;  888          0959  3                                         ! packet
;  889          0960  3
;  890          0961  3         PP_BASE = .PP - PP_PBCB_QUEUE_OFFSET;
;  891          0962  3
;  892          0963  3         IF .PP_BASE [PP_W_ROWS_TO_MOVE] NEQ 0
;  893          0964  3         THEN
;  894          0965  4             BEGIN       ! Projects somewhere on visible pasteboard
;  895          0966  4             LOCAL
;  896          0967  4                 D_PROJ   BLOCK [8,BYTE]; ! Representation of virtual
;  897          0968  4                                          ! display's projection on
```

```
;  898          0969  4                                          ! pasteboard.
;  899          0970  4
;  900          0971  4              D_PROJ [DCB_W_ROW_START] = .PP_BASE [PP_W_FIRST_WCB_ROW];
;  901          0972  4
;  902          0973  4              D_PROJ [DCB_W_NO_ROWS]   = .PP_BASE [PP_W_LAST_WCB_ROW] -
;  903          0974  4                                         .PP_BASE [PP_W_FIRST_WCB_ROW] +1;
;  904          0975  4
;  905          0976  4              D_PROJ [DCB_W_COL_START] = .PP_BASE [PP_W_FIRST_WCB_COL];
;  906          0977  4
;  907          0978  4              D_PROJ [DCB_W_NO_COLS]   = .PP_BASE [PP_W_LAST_WCB_COL] -
;  908          0979  4                                         .PP_BASE [PP_W_FIRST_WCB_COL] +1;
;  909          0980  4
;  910          0981  4              !+
;  911          0982  4              ! Check to see if cursor lies within projection of this
;  912          0983  4              ! virtual display.
;  913          0984  4              !-
;  914          0985  4              IF SMG$$POINT_IN_RECT_R3 (WCB [WCB_W_OLD_CUR_COL],
;  915          0986  4                                        WCB [WCB_W_OLD_CUR_ROW],
;  916          0987  4                                        D_PROJ) EQL 0
;  917          0988  4              THEN
;  918          0989  4                  .DISPLAY_ID = .PP_BASE [PP_A_DCB_ADDR]; !  Save candidate
;  919          0990  4
;  920          0991  3              END;        ! Projects somewhere on visible pasteboard
;  921          0992  3
;  922          0993  3          PP = .PP_BASE [PP_A_PREV_PBCB] ; ! To next outer pasting
;  923          0994  2          END;    ! For all pasting packets
;  924          0995  2
;  925          0996  2      RETURN (SS$_NORMAL);
;  926          0997  1      END;                ! Routine SMG$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       GTR     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 00000000G00040 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>
```

```
;  929          0999  1 %SBTTL 'SMG$CONTROL_MODE - Control overall mode of operation'
;  930          1000  1 GLOBAL ROUTINE SMG$CONTROL_MODE (
;  931          1001  1                                  PASTEBOARD_ID,
;  932          1002  1                                  NEW_MODE_BITS,
;  933          1003  1                                  OLD_MODE_BITS
;  934          1004  1                                ) =
;  935          1005  1 !++
;  936          1006  1 ! FUNCTIONAL DESCRIPTION:
;  937          1007  1 !
;  938          1008  1 !       This routine allows the caller to interrogate various modes
;  939          1009  1 !       of the overall operation of the SMG$ package, and to change
;  940          1010  1 !       those modes.  Each pasteboard (hence device) has its own set
;  941          1011  1 !       of mode settings.
;  942          1012  1 !
;  943          1013  1 !       The modes currently defined are:
;  944          1014  1 !
;  945          1015  1 !               1.  Use of buffers.  Under normal operation, the SMG$
;  946          1016  1 !               package will output to the terminal as soon as it has
;  947          1017  1 !               determined what needs to be written.  That is, no
;  948          1018  1 !               internal buffering will be done.  This mode is the
;  949          1019  1 !               default because it requires no follow-up effort on the
;  950          1020  1 !               part of the user to periodically force the buffer to the
;  951          1021  1 !               terminal.
;  952          1022  1 !               However  greater efficiency of $QIO usage can be
;  953          1023  1 !               achieved by enabling buffering.  In this mode, the SMG$
;  954          1024  1 !               package buffers all of its output for efficient usage of
;  955          1025  1 !               $QIO's.  The caller can at any time call
;  956          1026  1 !               SMG$FLUSH_BUFFER to force to the screen any output which
;  957          1027  1 !               has been queued to the screen, but not yet actually
;  958          1028  1 !               output.  For example, in this mode, each input operation
;  959          1029  1 !               initiated via the SMG$ facility will force a
;  960          1030  1 !               SMG$FLUSH_BUFFER call.
;  961          1031  1 !
;  962          1032  1 !               Default setting is to not provide buffering.
;  963          1033  1 !
;  964          1034  1 !               2.  Minimal screen update.  Unless explicitly told not
;  965          1035  1 !               to, the SMG$ package will try to minimize the number
;  966          1036  1 !               of characters actually sent to a terminal to increase
;  967          1037  1 !               overall system throughput.  It achieves this in part by
;  968          1038  1 !               remembering what is currently on the screen and actually
;  969          1039  1 !               outputing only the specific character sequences which
;  970          1040  1 !               will change the current contents of the screen into the
;  971          1041  1 !               desired contents of the screen.  As a result of this
;  972          1042  1 !               action, the original sequence of changes may not be
;  973          1043  1 !               preserved.  Generally, the screen is repainted so
;  974          1044  1 !               rapidly the this difference is not noticable since the
;  975          1045  1 !               resultant screen appears the way it should.  In rare
;  976          1046  1 !               circumstances, perhaps while debugging a new
;  977          1047  1 !               application, the caller may which to change this
;  978          1048  1 !               behavior to strictly preserve the order of his output.
;  979          1049  1 !
;  980          1050  1 !               The default mode is to perform minimal screen update.
;  981          1051  1 !               There is currently not much difference if you omit this
;  982          1052  1 !               bit because it really doesn't make much sense to repaint
;  983          1053  1 !               the entire screen when you change just one character,
;  984          1054  1 !               but this may change in future releases.  For now,
;  985          1055  1 !               you should always specify this bit.
```

```
;  986          1056  1 !
;  987          1057  1 !                   3. CLEAR_SCREEN. If this bit is set, then when SMG exits
;  988          1058  1 !                   normally, it will clear the screen just before returning
;  989          1059  1 !                   control to DCL.
;  990          1060  1 !
;  991          1061  1 !                   4. NOTABS.  If this bit is set, then SMG will never rely
;  992          1062  1 !                   on physical tabs even if the terminal supports them.
;  993          1063  1 !                   If not set, then SMG will feel free to use physical
;  994          1064  1 !                   tabs for the minimal update procedure.  Such use implicitly
;  995          1065  1 !                   assumes that your terminal's tab stops are set to the
;  996          1066  1 !                   DEC default locations.  Set this bit if you want to guarantee
;  997          1067  1 !                   that your product will run regardless of the tab settings
;  998          1068  1 !                   that the user has physically set on his terminal.
;  999          1069  1 !
; 1000          1070  1 !           The three parameters are optional.  First, if OLD_MODE_BITS
; 1001          1071  1 !           is supplied, it is filled in with the current mode bit settings.
; 1002          1072  1 !           Secondly, if the NEW_MODE_BITS parameter is provided, its
; 1003          1073  1 !           contents are used to set the current mode(s) of operation.
; 1004          1074  1 !
; 1005          1075  1 !           Hence this routine can typically be used in three way.
; 1006          1076  1 !           a).     To find out current settings:
; 1007          1077  1 !                   SMG$CONTROL_MODE ( PASTEBOARD_ID, , OLD_MODE_BITS)
; 1008          1078  1 !
; 1009          1079  1 !           b).     To set the bits with no regard for their current setting:
; 1010          1080  1 !                   SMG$CONTROL_MODE ( PASTEBOARD_ID, NEW_MODE_BITS)
; 1011          1081  1 !
; 1012          1082  1 !           c).     To write modular code, saving the current settings,
; 1013          1083  1 !                   setting them to your desired setting, then restoring
; 1014          1084  1 !                   original settings before exiting your procedure:
; 1015          1085  1 !                   SMG$CONTROL_MODE ( PASTEBOARD_ID,
; 1016          1086  1 !                                      NEW_MODE_BITS, saved_mode_bits )
; 1017          1087  1 !                   and before exiting,
; 1018          1088  1 !                   SMG$CONTROL_MODE ( PASTEBOARD_ID, saved_mode_bits )
; 1019          1089  1 !
; 1020          1090  1
; 1021          1091  1 ! CALLING SEQUENCE:
; 1022          1092  1 !
; 1023          1093  1 !           ret_status.wlc.v = SMG$CONTROL_MODE ( PASTEBOARD_ID.rl.r,
; 1024          1094  1 !                                                 [NEW_MODE_BITS.rl.r],
; 1025          1095  1 !                                                 [OLD_MODE_BITS.wl.r] )
; 1026          1096  1 !
; 1027          1097  1 ! FORMAL PARAMETERS:
; 1028          1098  1 !
; 1029          1099  1 !           PASTEBOARD_ID.rl.r      The id of the PASTEBOARD for which the
; 1030          1100  1 !                                   modes are being set or interrogated.
; 1031          1101  1 !
; 1032          1102  1 !           NEW_MODE_BITS.rl.r      [Optional].  If supplied, new mode
; 1033          1103  1 !                                   settings to be employed.
; 1034          1104  1 !                                   A bit set to 1 forces that mode to be
; 1035          1105  1 !                                   employed.  A bit set to 0 inhibits that
; 1036          1106  1 !                                   mode of operation.
; 1037          1107  1 !                                   Bit SMG$K_BUF_ENABLED controls buffering.
; 1038          1108  1 !                                   This should normally be set to improve
; 1039          1109  1 !                                   performance.
; 1040          1110  1 !                                   Bit SMG$K_MINUPD controls minimal update and
; 1041          1111  1 !                                   should normally be set.
; 1042          1112  1 !                                   Bit SMG$K_CLEAR_SCREEN causes SMG to clear
```

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

```
; 1084          1153  2     BEGIN
; 1085          1154  2     BUILTIN
; 1086          1155  2         NULLPARAMETER;
; 1087          1156  2
; 1088          1157  2     LOCAL
; 1089          1158  2         PBCB : REF BLOCK [,BYTE];       ! Address of associated
; 1090          1159  2                                         ! pasteboard control block.
; 1091          1160  2
; 1092          1161  2     $SMG$GET_PBCB ( .PASTEBOARD_ID, PBCB) ; ! Get address of PBCB
; 1093          1162  2
; 1094          1163  2
; 1095          1164  2     !+
; 1096          1165  2     ! If caller requested current settings, return them.
; 1097          1166  2     !-
; 1098          1167  2         IF NOT NULLPARAMETER (OLD_MODE_BITS)
; 1099          1168  2         THEN
; 1100          1169  2             .OLD_MODE_BITS = .PBCB [PBCB_L_MODE_SETTINGS];
; 1101          1170  2
; 1102          1171  2     !+
; 1103          1172  2     ! If caller provided new settings, process them.
; 1104          1173  2     !-
; 1105          1174  2         IF NOT NULLPARAMETER (NEW_MODE_BITS)
; 1106          1175  2         THEN
; 1107          1176  3             BEGIN   ! Caller provided new settings
; 1108          1177  3
; 1109          1178  3             BIND    NEW_MODE = .NEW_MODE_BITS       : BITVECTOR[32];
; 1110          1179  3
; 1111          1180  3             !+
; 1112          1181  3             ! First make sure that he didn't supply extraneous bits
; 1113          1182  3             !-
; 1114          1183  3             IF .(.NEW_MODE_BITS)<4,29> NEQ 0
; 1115          1184  3             THEN
; 1116          1185  3                 RETURN (SMG$_INVARG);       ! Unsupported bits provided
; 1117          1186  3
; 1118          1187  3             !+
; 1119          1188  3             ! If he's switching from buffered to non-buffered, flush out
; 1120          1189  3             ! current contents of buffers so we are caught up.
; 1121          1190  3             !-
; 1122          1191  3             IF .PBCB [PBCB_V_BUF_ENABLED]  AND      ! If currently enabled
; 1123          1192  3                NOT .NEW_MODE[SMG$K_BUF_ENABLED]     ! and new disabled
; 1124          1193  3             THEN
; 1125          1194  3                 SMG$$FLUSH_BUFFER ( .PBCB ),
; 1126          1195  3
; 1127          1196  3             !+
; 1128          1197  3             ! If he's switching from no minimal update to minimal update,
; 1129          1198  3             ! then ensure that the screen buffer is up-to-date.
; 1130          1199  3             !-
; 1131          1200  3             IF NOT .PBCB [PBCB_V_MINUPD]  AND       ! If currently disabled
; 1132          1201  3                .NEW_MODE[SMG$K_MINUPD]              ! and new enabled
; 1133          1202  3             THEN
; 1134          1203  4                 BEGIN       ! update screen buffer
; 1135          1204  4
; 1136          1205  4                 BIND WCB = .PBCB [PBCB_A_WCB]       : BLOCK [,BYTE];
; 1137          1206  4
; 1138          1207  4                 !+
; 1139          1208  4                 ! Move the current text and attributes buffers
; 1140          1209  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$:      MOVL    PBD_A_PBCB[R0], PBCB                        ;
                                C?                        6C   91 00029       CMPB    (AP), #3                            ; 1167
                                                          0A   1F 0002C       BLSSU   3$                                  ;
                                                    0C    AC   D5 0002E       TSTL    12(AP)                              ;
                                                          05   13 00031       BEQL    3$                                  ;
                                                0C  BC        0C   A7 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
```

```
                    00000000G  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
```

```
;  1169          1237  1 %SBTTL 'SMG$INVALIDATE_DISPLAY - Mark display as being privately used'
;  1170          1238  1 GLOBAL ROUTINE SMG$INVALIDATE_DISPLAY ( DISPLAY_ID) =
;  1171          1239  1
;  1172          1240  1 !++
;  1173          1241  1 ! FUNCTIONAL DESCRIPTION:
;  1174          1242  1 !
;  1175          1243  1 !       This routine is called when ever a change has been completed
;  1176          1244  1 !       to a given virtual display and the user had previously
;  1177          1245  1 !       written into that display on his own, without using
;  1178          1246  1 !       SMG routines.
;  1179          1247  1 !
;  1180          1248  1 !       The virtual display must not be occluded.                    ;
;  1181          1249  1 !
;  1182          1250  1 !       Each pasteboard to which this display is pasted is isolated and
;  1183          1251  1 !       its window image must be redrawn.                            ;
;  1184          1252  1 !
;  1185          1253  1 ! CALLING SEQUENCE:
;  1186          1254  1 !
;  1187          1255  1 !       ret_status.wlc.v = SMG$INVALIDATE_DISPLAY ( DISPLAY_ID.rl.r)
;  1188          1256  1 !
;  1189          1257  1 ! FORMAL PARAMETERS:
;  1190          1258  1 !
;  1191          1259  1 !       DISPLAY_ID.rl.r         Display ID of virtual display.
;  1192          1260  1 !
;  1193          1261  1 ! IMPLICIT INPUTS:
;  1194          1262  1 !
;  1195          1263  1 !       NONE
;  1196          1264  1 !
;  1197          1265  1 ! IMPLICIT OUTPUTS:
;  1198          1266  1 !
;  1199          1267  1 !       NONE
;  1200          1268  1 !
;  1201          1269  1 ! COMPLETION STATUS:
;  1202          1270  1 !
;  1203          1271  1 !       SS$_NORMAL      Normal successful completion
;  1204          1272  1 !
;  1205          1273  1 ! SIDE EFFECTS:
;  1206          1274  1 !
;  1207          1275  1 !       NONE
;  1208          1276  1 !--
```

```
;  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
```

```
;  1234     1300  1 %SBTTL 'SMG$GET_CHAR_AT_PHYSICAL_CURSOR - Get character at physical cursor'
;  1235     1301  1 GLOBAL ROUTINE SMG$GET_CHAR_AT_PHYSICAL_CURSOR (
;  1236     1302  1                                                 PASTEBOARD_ID,
;  1237     1303  1                                                 CHARACTER : REF VECTOR [,BYTE]
;  1238     1304  1                                                 ) =
;  1239     1305  1
;  1240     1306  1 !++
;  1241     1307  1 ! FUNCTIONAL DESCRIPTION
;  1242     1308  1 !
;  1243     1309  1 ! Returns the character that occupies the position on the screen corresponding
;  1244     1310  1 ! to the current physical cursor position.
;  1245     1311  1 !
;  1246     1312  1 ! Note:  If the SMG$ facility has not written to that location on the screen,
;  1247     1313  1 ! then it doesn't know its contents.  A character encoded as X'FF' will be
;  1248     1314  1 ! returned in that case.
;  1249     1315  1 !
;  1250     1316  1 ! If character returned has a value less than X'20', then it is not an actual
;  1251     1317  1 ! printable character, but an internal terminal-independent code of what should
;  1252     1318  1 ! be displayed at that position, e.g. an element of the line-drawing character
;  1253     1319  1 ! set.  Do not attempt to use this code for subsequent output operations.
;  1254     1320  1 !
;  1255     1321  1 !
;  1256     1322  1 ! CALLING SEQUENCE
;  1257     1323  1 !
;  1258     1324  1 !       status.wlc.v = SMG$GET_CHAR_AT_PHYSICAL_CURSOR (
;  1259     1325  1 !                                  PASTEBOARD_ID.rl.r,
;  1260     1326  1 !                                  CHARACTER.wbu.r)
;  1261     1327  1 !
;  1262     1328  1 ! FORMAL PARAMETERS
;  1263     1329  1 !
;  1264     1330  1 !       PASTEBOARD_ID.rl.r      Address of a longword containing the
;  1265     1331  1 !                               pasteboard_id for which the information is
;  1266     1332  1 !                               desired.
;  1267     1333  1 !
;  1268     1334  1 !       CHARACTER.wbu.r         Address of a byte to receive the result.
;  1269     1335  1 !
;  1270     1336  1 ! IMPLICIT INPUTS:
;  1271     1337  1 !
;  1272     1338  1 !       NONE
;  1273     1339  1 !
;  1274     1340  1 ! IMPLCIT OUTPUTS:
;  1275     1341  1 !
;  1276     1342  1 !       NONE
;  1277     1343  1 !
;  1278     1344  1 ! COMPLETION STATUS:
;  1279     1345  1 !
;  1280     1346  1 !       SS$NORMAL       Normal successful completion
;  1281     1347  1 !       SMG$_INVPAS_ID  Invalid pasteboard id
;  1282     1348  1 !       SMG$_WRONUMARG  Wrong number of arguments
;  1283     1349  1 !
;  1284     1350  1 ! SIDE EFFECTS:
;  1285     1351  1 !
;  1286     1352  1 !       NONE
;  1287     1353  1 !--
;  1288     1354  2     BEGIN
;  1289     1355  2     LOCAL
;  1290     1356  2         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$_WRONUMARG, R0                         ;
                                         04   04 0000E         RET                                         ;
                              50      04   BC   D0 0000F 1$:     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   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   04 0005B         RET                                         ; 1382

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

```
;1318          1383  1 %SBTTL 'SMG$REPAINT_SCREEN - Repaint current screen'
;1319          1384  1 GLOBAL ROUTINE SMG$REPAINT_SCREEN (
;1320          1385  1                                   PASTEBOARD_ID
;1321          1386  1                                     )=
;1322          1387  1 !++
;1323          1388  1 ! FUNCTIONAL DESCRIPTION:
;1324          1389  1 !
;1325          1390  1 !       This procedure will repaint the current screen based on its
;1326          1391  1 !       internal knowledge of what the screen should look like.
;1327          1392  1 !       It is intended to be called when it is known or suspected that
;1328          1393  1 !       some outside agent has disrupted the screen so that it no longer
;1329          1394  1 !       matches the internal knowledge of what should be on the screen.
;1330          1395  1 !       If pasteboard batching is in effect, the repaint occurs
;1331          1396  1 !       from the screen image; otherwise it occurs from the text image.
;1332          1397  1 !
;1333          1398  1 ! CALLING SEQUENCE:
;1334          1399  1 !
;1335          1400  1 !       ret_status.wlc.v = SMG$REPAINT_SCREEN ( PASTEBOARD_ID.rl.r )
;1336          1401  1 !
;1337          1402  1 ! FORMAL PARAMETERS:
;1338          1403  1 !
;1339          1404  1 !       PASTEBOARD_ID.rl.r      Id of pasteboard associated with
;1340          1405  1 !                               physical screen to be repainted.
;1341          1406  1 !
;1342          1407  1 ! IMPLICIT INPUTS:
;1343          1408  1 !
;1344          1409  1 !       Current internal representation of what should be on the screen.
;1345          1410  1 !
;1346          1411  1 ! IMPLICIT OUTPUTS:
;1347          1412  1 !
;1348          1413  1 !       NONE
;1349          1414  1 !
;1350          1415  1 ! COMPLETION STATUS:
;1351          1416  1 !
;1352          1417  1 !       SS$_NORMAL      Normal successful completion
;1353          1418  1 !       SMG$_INVPAS_ID  Invalid pasteboard control block
;1354          1419  1 !
;1355          1420  1 ! SIDE EFFECTS:
;1356          1421  1 !
;1357          1422  1 !       NONE
;1358          1423  1 !--
;1359          1424  2     BEGIN
;1360          1425  2     LOCAL
;1361          1426  2         WCB     : REF BLOCK [,BYTE],    ! Window control block
;1362          1427  2         PBCB    : REF BLOCK [,BYTE];    ! Address of pasteboard control
;1363          1428  2                                         ! block
;1364          1429  2
;1365          1430  2     $SMG$GET_PBCB ( .PASTEBOARD_ID, PBCB); ! Get address of pasteboard
;1366          1431  2                                          ! control block.
;1367          1432  2
;1368          1433  2     !+
;1369          1434  2     ! Get address of window control block.
;1370          1435  2     !-
;1371          1436  2
;1372          1437  2     WCB = .PBCB [PBCB_A_WCB];
;1373          1438  2
;1374          1439  2     !+
```

```
;  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            5C E0 00011      BBS    R0, PBD_V_PB_AVAIL, 2$              ;
                                        50  00000000G   8F    DO 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>
```

```
;  1403          1467  1 %SBTTL 'SMG$RING_BELL - Ring Bell'
;  1404          1468  1 GLOBAL ROUTINE SMG$RING_BELL (
;  1405          1469  1                               DISPLAY_ID,
;  1406          1470  1                               TIMES
;  1407          1471  1                             ) =
;  1408          1472  1 !++
;  1409          1473  1 ! FUNCTIONAL DESCRIPTION:
;  1410          1474  1 !
;  1411          1475  1 !       This routine rings the bell, on each physical terminal to which
;  1412          1476  1 !       the given virtual display is pasted, the number of times
;  1413          1477  1 !       specified.  If TIMES omitted, 1 is used.
;  1414          1478  1 !
;  1415          1479  1 ! CALLING SEQUENCE:
;  1416          1480  1 !
;  1417          1481  1 !       ret_status.wlc.v = SMG$RING_BELL (
;  1418          1482  1 !                                         DISPLAY_ID.rl.r,
;  1419          1483  1 !                                         [TIMES.rl.r] )
;  1420          1484  1 !
;  1421          1485  1 ! FORMAL PARAMETERS:
;  1422          1486  1 !
;  1423          1487  1 !       DISPLAY_ID.rl.r The display id of the virtual display.
;  1424          1488  1 !
;  1425          1489  1 !       TIMES.rl.r      Optional.  The number of times to ring the bell.
;  1426          1490  1 !                       If not specified, the bell is sounded once.
;  1427          1491  1 !
;  1428          1492  1 ! IMPLICIT INPUTS:
;  1429          1493  1 !
;  1430          1494  1 !       NONE
;  1431          1495  1 !
;  1432          1496  1 ! IMPLICIT OUTPUTS:
;  1433          1497  1 !
;  1434          1498  1 !       NONE
;  1435          1499  1 !
;  1436          1500  1 ! COMPLETION STATUS:
;  1437          1501  1 !
;  1438          1502  1 !       SS$_NORMAL      Normal successful completion
;  1439          1503  1 !       Statuses returned by SMG$$OUTPUT or
;  1440          1504  1 !       statuses or signal values returned by STR$DUPL_CHAR.
;  1441          1505  1 !
;  1442          1506  1 ! SIDE EFFECTS:
;  1443          1507  1 !
;  1444          1508  1 !       NONE
;  1445          1509  1 !--
;  1446          1510  1
;  1447          1511  2     BEGIN
;  1448          1512  2     BUILTIN
;  1449          1513  2         NULLPARAMETER ;
;  1450          1514  2                                                                    ;
;  1451          1515  2     BIND
;  1452          1516  2         BELL_CHAR = UPLIT (BYTE (BELL));
;  1453          1517  2                                                                    ;
;  1454          1518  2     LOCAL
;  1455          1519  2         DCB : REF BLOCK [,BYTE],        ! Address of display control
;  1456          1520  2                                         ! block
;  1457          1521  2
;  1458          1522  2         PP  : REF BLOCK [,BYTE],        ! Address of a pasting packet
;  1459          1523  2
```

```
;  1460          1524  2          STATUS;                                 ! Status to return to caller
;  1461          1525  2
;  1462          1526  2      $SMG$VALIDATE_ARGCOUNT (1, 2);
;  1463          1527  2
;  1464          1528  2      $SMG$GET_DCB (.DISPLAY_ID, DCB);            ! Get addr of virtual display
;  1465          1529  2                                                  ! control block.
;  1466          1530  2
;  1467          1531  2      STATUS=SS$_NORMAL;
;  1468          1532  2
;  1469          1533  2      IF NULLPARAMETER (TIMES)
;  1470          1534  2      THEN
;  1471          1535  3          BEGIN   ! No arg supplied, output 1 bell
;  1472          1536  3          !+
;  1473          1537  3          ! Chase the chain of pasteboards to which this virtual display
;  1474          1538  3          ! is pasted.  For each pasteboard located, output the bell.
;  1475          1539  3          !-
;  1476          1540  3          PP = .DCB [DCB_A_PP_NEXT];
;  1477          1541  3
;  1478          1542  3          IF .PP EQL 0
;  1479          1543  3          THEN
;  1480          1544  3              RETURN (SMG$_FATERRLIB);            ! should never be 0
;  1481          1545  3
;  1482          1546  3          WHILE .PP NEQ DCB [DCB_A_PP_NEXT]       ! While any remain...
;  1483          1547  3          DO
;  1484          1548  4              BEGIN
;  1485          1549  4              STATUS=SMG$$OUTPUT ( .PP [PP_A_PBCB_ADDR],
;  1486          1550  4                                   1,
;  1487          1551  4                                   BELL_CHAR);
;  1488          1552  4
;  1489          1553  4              PP = .PP [PP_A_NEXT_DCB];   ! Step to next packet
;  1490          1554  3              END;
;  1491          1555  3          END     ! No arg supplied, output 1 bell
;  1492          1556  3
;  1493          1557  2      ELSE
;  1494          1558  2
;  1495          1559  3          BEGIN   ! Multiple bells
;  1496          1560  3          LOCAL
;  1497          1561  3              DESCR : BLOCK [8, BYTE] ;   ! A local descriptor
;  1498          1562  3
;  1499          1563  3          !+
;  1500          1564  3          ! Must build a string of TIMES bell characters.  Use dynamic
;  1501          1565  3          ! string.
;  1502          1566  3          !-
;  1503          1567  3          DESCR [DSC$W_LENGTH] = 0 ;
;  1504          1568  3          DESCR [DSC$B_CLASS]  = DSC$K_CLASS_D ;
;  1505          1569  3          DESCR [DSC$B_DTYPE]  = DSC$K_DTYPE_T ;
;  1506          1570  3          DESCR [DSC$A_POINTER] = 0 ;
;  1507          1571  3
;  1508          1572  3          LIB$ESTABLISH ( LIB$SIG_TO_RET ) ; ! Establish handler in case
;  1509          1573  3                                             ! STR$DUPL_CHAR signals out
;  1510          1574  3                                             ! from under us.
;  1511          1575  3
;  1512          1576  4          IF NOT (STATUS = STR$DUPL_CHAR ( DESCR, .TIMES, BELL_CHAR))
;  1513          1577  3          THEN
;  1514          1578  3              RETURN (.STATUS );
;  1515          1579  3
;  1516          1580  3          !+
```

```
;  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  00A4              RET                                                     ;
                                50      20  A3  9E 00A5 8$:     MOVAB   32(DCB), R0                                     ; 1590
                                50          52  D1 00A9          CMPL    PP, R0                                          ;
                                            15  13 00AC          BEQL    9$                                              ;
                                    04      AE  DD 00AE          PUSHL   DESCR+4                                         ; 1595
                                7E  04      AE  3C 00B1          MOVZWL  DESCR, -(SP)                                    ; 1594
                                    14      A2  DD 00B5          PUSHL   20(PP)                                          ; 1593
                                65          03  FB 00B8          CALLS   #3, SMG$$OUTPUT                                 ;
                                54          50  D0 00BB          MOVL    R0, STATUS                                      ;
                                52          62  D0 00BE          MOVL    (PP), PP                                        ; 1597
                                            E2  11 00C1          BRB     8$                                              ; 1590
                                            5E  DD 00C3 9$:      PUSHL   SP                                              ; 1600
                        00000000G   00          01  FB 00C5          CALLS   #1, LIB$SFREE1_DD                               ;
                                50          54  D0 00CC 10$:     MOVL    STATUS, R0                                      ; 1603
                                            04  00CF              RET                                                     ; 1604

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

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

```
;  1543          1606  1 %SBTTL 'SMG$$BEGIN_PASTEBOARD_UPDATE_R1 - Begin batch of updates to pasteboard'
;  1544          1607  1 GLOBAL ROUTINE SMG$$BEGIN_PASTEBOARD_UPDATE_R1 (PBCB: REF $PBCB_DECL)
;  1545          1608  1                                 : SMG$$BEGIN_PBD_UPDATE$LNK =
;  1546          1609  1 !++
;  1547          1610  1 ! FUNCTIONAL DESCRIPTION:
;  1548          1611  1 !
;  1549          1612  1 !       Inner routine to support BEGIN_PASTEBOARD_UPDATE action.
;  1550          1613  1 !       Disable outputing this pasteboard to the screen until the
;  1551          1614  1 !       matching call to SMG$$END_PASTEBOARD_UPDATE_R2 is encountered.
;  1552          1615  1 !
;  1553          1616  1 ! CALLING SEQUENCE:
;  1554          1617  1 !
;  1555          1618  1 !       ret_status.wlc.v = SMG$$BEGIN_PASTEBOARD_UPDATE_R1 ( PBCB.rab.r)
;  1556          1619  1 !
;  1557          1620  1 ! FORMAL PARAMETERS:
;  1558          1621  1 !
;  1559          1622  1 !       PBCB.rab.r              Address of pasteboard control block
;  1560          1623  1 !
;  1561          1624  1 ! IMPLICIT INPUTS:
;  1562          1625  1 !
;  1563          1626  1 !       NONE
;  1564          1627  1 !
;  1565          1628  1 ! IMPLICIT OUTPUTS:
;  1566          1629  1 !
;  1567          1630  1 !       NONE
;  1568          1631  1 !
;  1569          1632  1 ! COMPLETION STATUS:
;  1570          1633  1 !
;  1571          1634  1 !       SS$_NORMAL      Normal successful completion, batching has been
;  1572          1635  1 !                       initiated
;  1573          1636  1 !       SMG$_BATWAS_ON  Success, but note that batching was already on
;  1574          1637  1 !
;  1575          1638  1 ! SIDE EFFECTS:
;  1576          1639  1 !
;  1577          1640  1 !       NONE
;  1578          1641  1 !--
;  1579          1642  1
;  1580          1643  2     BEGIN
;  1581          1644  2
;  1582          1645  2 !+
;  1583          1646  2 ! Increment count of number of SMG$$END_PASTEBOARD_UPDATE_R2 calls we need to
;  1584          1647  2 ! see before we resume outputing from this pasteboard.
;  1585          1648  2 ! Let the user know what the previous state of batching was by
;  1586          1649  2 ! our status return (in case he's interested).
;  1587          1650  2 !-
;  1588          1651  2     PBCB [PBCB_L_BATCH_LEVEL] = .PBCB [PBCB_L_BATCH_LEVEL] + 1,
;  1589          1652  2
;  1590          1653  2     IF .PBCB [PBCB_L_BATCH_LEVEL] EQL 1
;  1591          1654  2     THEN
;  1592          1655  2         RETURN  SS$_NORMAL
;  1593          1656  2     ELSE
;  1594          1657  2         RETURN  SMG$_BATWAS_ON
;  1595          1658  2
;  1596          1659  1     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>
```

```
;  1599          1661  1 %SBTTL 'SMG$$END_PASTEBOARD_UPDATE_R2 - End batch of updates to pasteboard'
;  1600          1662  1 GLOBAL ROUTINE SMG$$END_PASTEBOARD_UPDATE_R2 ( PBCB: REF $PBCB_DECL)
;  1601          1663  1                         : SMG$$END_PBD_UPDATE$LNK =
;  1602          1664  1 !++
;  1603          1665  1 ! FUNCTIONAL DESCRIPTION:
;  1604          1666  1 !
;  1605          1667  1 !       Inner routine to support BEGIN_PASTEBOARD_UPDATE action.
;  1606          1668  1 !       Reduce the number of calls to SMG$$END_PASTEBOARD_UPDATE_R2 that will
;  1607          1669  1 !       be needed before we resume outputing from this pasteboard.  If
;  1608          1670  1 !       this call makes this count go to zero, flush the pasteboard to
;  1609          1671  1 !       the screen.
;  1610          1672  1 !       If the level is already 0, we return a success status
;  1611          1673  1 !       informing caller that the batching level was already 0,
;  1612          1674  1 !       but we otherwise allow this.  This gives a user a guaranteed
;  1613          1675  1 !       method of ending batching.
;  1614          1676  1 !
;  1615          1677  1 ! CALLING SEQUENCE:
;  1616          1678  1 !
;  1617          1679  1 !       ret_status.wlc.v = SMG$$END_PASTEBOARD_UPDATE_R2 ( PBCB.rab.r)
;  1618          1680  1 !
;  1619          1681  1 ! FORMAL PARAMETERS:
;  1620          1682  1 !
;  1621          1683  1 !       PBCB.rab.r              Address of pasteboard control block
;  1622          1684  1 !
;  1623          1685  1 ! IMPLICIT INPUTS:
;  1624          1686  1 !
;  1625          1687  1 !       PBCB [PBCB_L_BATCH_LEVEL]
;  1626          1688  1 !
;  1627          1689  1 ! IMPLICIT OUTPUTS:
;  1628          1690  1 !
;  1629          1691  1 !       PBCB [PBCB_L_BATCH_LEVEL] gets decremented
;  1630          1692  1 !
;  1631          1693  1 ! COMPLETION STATUS:
;  1632          1694  1 !
;  1633          1695  1 !       SS$_NORMAL      Normal successful completion
;  1634          1696  1 !       SMG$_BATSTIPRO  Success; but batching is still in progress.
;  1635          1697  1 !       SMG$_BATWASOFF  Success; but batching was already off
;  1636          1698  1 !
;  1637          1699  1 ! SIDE EFFECTS:
;  1638          1700  1 !
;  1639          1701  1 !       NONE
;  1640          1702  1 !--
;  1641          1703  1
;  1642          1704  2     BEGIN
;  1643          1705  2     LOCAL
;  1644          1706  2         STATUS;                         ! Status to return
;  1645          1707  2
;  1646          1708  2 !+
;  1647          1709  2 ! If current count is greater than 1, simply reduce it by one and
;  1648          1710  2 ! return to caller.  If it is one, reduce it to zero and cause the
;  1649          1711  2 ! current contents of this pasteboard to be flushed to the screen (if need
;  1650          1712  2 ! be).
;  1651          1713  2 !-
;  1652          1714  2     IF .PBCB [PBCB_L_BATCH_LEVEL] GTRU 1
;  1653          1715  2     THEN
;  1654          1716  3         BEGIN
;  1655          1717  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>
```

```
;  1683          1744  1 %SBTTL 'SMG$$CHECK_FOR_OUTPUT_DCB - Check to see if output needed'
;  1684          1745  1 GLOBAL ROUTINE SMG$$CHECK_FOR_OUTPUT_DCB (
;  1685          1746  1                                           DCB : REF BLOCK [,BYTE],
;  1686          1747  1                                           REQUEST_CODE,
;  1687          1748  1                                           REQUEST_ARG
;  1688          1749  1                                         ) =
;  1689          1750  1 !++
;  1690          1751  1 ! FUNCTIONAL DESCRIPTION:
;  1691          1752  1 !
;  1692          1753  1 !       This routine is called when ever a change has been completed
;  1693          1754  1 !       to a given virtual display.  Changes consist of:
;  1694          1755  1 !               1. Altering the text in a virtual display
;  1695          1756  1 !               2. Altering the attributes of the text in a virtual
;  1696          1757  1 !                  display.
;  1697          1758  1 !               3. Altering the cursor position within a display.
;  1698          1759  1 !               4. A virtual display's buffering level going to zero.
;  1699          1760  1 !               5. A virtual display has just been pasted to a
;  1700          1761  1 !                  pasteboard.
;  1701          1762  1 !       The criteria for outputing consist of:
;  1702          1763  1 !               1. Virtual display buffering level is zero
;  1703          1764  1 !               2. The virtual display is pasted to at least one
;  1704          1765  1 !                  pasteboard.
;  1705          1766  1 !
;  1706          1767  1 !       Each pasteboard to which this display is pasted is isolated and
;  1707          1768  1 !       its window image must be redrawn.
;  1708          1769  1 !       The new physical pasteboard cursor position gets set.
;  1709          1770  1 !
;  1710          1771  1 ! CALLING SEQUENCE:
;  1711          1772  1 !
;  1712          1773  1 !       ret_status.wlc.v = SMG$$CHECK_FOR_OUTPUT_DCB (
;  1713          1774  1 !                                                     DCB.rab.r
;  1714          1775  1 !                                                     [,REQEST_CODE.rl.v])
;  1715          1776  1 !
;  1716          1777  1 ! FORMAL PARAMETERS:
;  1717          1778  1 !
;  1718          1779  1 !       DCB.rab.r       Address of virtual display control block for
;  1719          1780  1 !                       affected display.
;  1720          1781  1 !
;  1721          1782  1 !       [REQUEST_CODE.rl.v] Optional.  If supplied, it is a code telling
;  1722          1783  1 !                       the rature of the request that provoked this
;  1723          1784  1 !                       call to SMG$$CHECK_FOR_OUTPUT_DCB.
;  1724          1785  1 !                       Among other things, it tells us when it is
;  1725          1786  1 !                       necessary to redraw borders.
;  1726          1787  1 !
;  1727          1788  1 !       [REQUEST_ARG.rl.v] Optional. If supplied, then it tells us the
;  1728          1789  1 !                       unique line in the DCB that has changed.
;  1729          1790  1 !                       If specified as 0, then the whole DCB has changed.
;  1730          1791  1 !
;  1731          1792  1 ! IMPLICIT INPUTS:
;  1732          1793  1 !
;  1733          1794  1 !       NONE
;  1734          1795  1 !
;  1735          1796  1 ! IMPLICIT OUTPUTS:
;  1736          1797  1 !
;  1737          1798  1 !       WCB[WCB_W_CURR_CUR_ROW]         gets set to place corresponding
;  1738          1799  1 !                                       to logical cursor position in
;  1739          1800  1 !                                       this virtual display.
```

```
;  1740          1801  1 !
;  1741          1802  1 !       WCB[WCB_W_CURR_CUR_COL]          gets set to place corresponding
;  1742          1803  1 !                                        to logical cursor position in
;  1743          1804  1 !                                        this virtual display.
;  1744          1805  1 !
;  1745          1806  1 ! COMPLETION STATUS:
;  1746          1807  1 !
;  1747          1808  1 !       SS$_NORMAL      Normal successful completion
;  1748          1809  1 !
;  1749          1810  1 ! SIDE EFFECTS:
;  1750          1811  1 !
;  1751          1812  1 !       NONE
;  1752          1813  1 !--
;  1753          1814  1
;  1754          1815  2     BEGIN
;  1755          1816  2     BUILTIN
;  1756          1817  2         ACTUALCOUNT;
;  1757          1818  2
;  1758          1819  2     LOCAL
;  1759          1820  2         CURR_PP : REF BLOCK [,BYTE],     ! Addr of pasting packet under
;  1760          1821  2                                          ! inspection
;  1761          1822  2
;  1762          1823  2         STATUS; ! Status of subroutine calls
;  1763          1824  2
;  1764          1825  2 !+
;  1765          1826  2 ! If current buffer level not equal to zero, this is not the time to
;  1766          1827  2 ! flush this display to the screen.
;  1767          1828  2 !-
;  1768          1829  2     IF .DCB [DCB_L_BATCH_LEVEL] NEQ 0
;  1769          1830  2     THEN
;  1770          1831  2         RETURN (SS$_NORMAL);
;  1771          1832  2
;  1772          1833  2     CURR_PP = .DCB [DCB_A_PP_NEXT];      ! Start of chain of pasting
;  1773          1834  2                                          ! packets to which this display
;  1774          1835  2                                          ! is pasted.
;  1775          1836  2     IF .CURR_PP EQL 0
;  1776          1837  2     THEN
;  1777          1838  2         RETURN (SMG$_FATERRLIB);         ! should never be 0
;  1778          1839  2
;  1779          1840  2 !+
;  1780          1841  2 ! Deal with each pasteboard that this virtual display is pasted to...
;  1781          1842  2 !-
;  1782          1843  2     WHILE .CURR_PP NEQ DCB [DCB_A_PP_NEXT]
;  1783          1844  2     DO
;  1784          1845  3         BEGIN   ! Overall loop
;  1785          1846  3
;  1786          1847  3         LOCAL
;  1787          1848  3             PBCB : REF BLOCK [,BYTE],    ! Addr. of pasteboard control
;  1788          1849  3             WCB  : REF BLOCK [,BYTE];    ! Addr. of window control block
;  1789          1850  3
;  1790          1851  3         PBCB = .CURR_PP [PP_A_PBCB_ADDR]; ! Select pasteboard and WCB
;  1791          1852  3
;  1792          1853  3         IF .PBCB [PBCB_L_BATCH_LEVEL] EQL 0
;  1793          1854  3         THEN
;  1794          1855  4             BEGIN       ! Force output
;  1795          1856  4             LOCAL
;  1796          1857  4                 OPT_FLAG;       ! Flag
```

```
;  1797          1856  4
;  1798          1859  4              WCB  = .PBCB [PBCB_A_WCB]; ! whose window needs rebuilding.
;  1799          1860  4
;  1800          1861  4              !+
;  1801          1862  4              ! Set the cursor position in the pasteboard to correspond
;  1802          1863  4              ! to it's place on this virtual display.
;  1803          1864  4              !-
;  1804          1865  4              WCB [WCB_W_CURR_CUR_ROW] = .DCB [DCB_W_CURSOR_ROW]-1
;  1805          1866  4                                       + .CURR_PP [PP_W_ROW];
;  1806          1867  4
;  1807          1868  4              WCB [WCB_W_CURR_CUR_COL] = .DCB [DCB_W_CURSOR_COL]-1
;  1808          1869  4                                       + .CURR_PP [PP_W_COL];
;  1809          1870  4
;  1810          1871  4              !+
;  1811          1872  4              ! If REQUEST_CODE is present and it is a type that just
;  1812          1873  4              ! involves cursor motion, deal with it as a special case to
;  1813          1874  4              ! avoid remapping the virtual display since contents did
;  1814          1875  4              ! not change.
;  1815          1876  4              !-
;  1816          1877  4              OPT_FLAG = 0;
;  1817          1878  4              IF ACTUALCOUNT () GEQ 2     ! REQUEST_CODE present
;  1818          1879  4              THEN
;  1819          1880  5                  BEGIN   ! Try optimization
;  1820          1881  5                  IF .REQUEST_CODE EQL SMG$C_SET_CURSOR_ABS OR
;  1821          1882  5                     .REQUEST_CODE EQL SMG$C_SET_CURSOR_REL
;  1822          1883  5                  THEN
;  1823          1884  5                     OPT_FLAG = 1;
;  1824          1885  4                  END;    ! Try optimization
;  1825          1886  4
;  1826          1887  4              IF .OPT_FLAG
;  1827          1888  4              THEN
;  1828          1889  5                  BEGIN   ! Opt. possible
;  1829          1890  5                  !+
;  1830          1891  5                  ! Force cursor to desired spot in optimal fashion
;  1831          1892  5                  !-
;  1832          1893  5                  SMG$$UPDATE_PHYSICAL_CURSOR ( .PBCB);
;  1833          1894  5                  END     ! Opt. possible
;  1834          1895  5
;  1835          1896  4              ELSE
;  1836          1897  4
;  1837          1898  5                  BEGIN   ! Remapping necessary
;  1838          1899  5                  !+
;  1839          1900  5                  ! Check to see if we can get by with just redrawing the
;  1840          1901  5                  ! changed virtual display, or whether we need to redraw
;  1841          1902  5                  ! the changed one and all higher-pasted virtual
;  1842          1903  5                  ! displays.  We can do the former if this pasting has a
;  1843          1904  5                  ! bit that says it is not occluded.
;  1844          1905  5                  !-
;  1845          1906  5                  IF NOT .CURR_PP [PP_V_OCCLUDED]
;  1846          1907  5                  THEN
;  1847          1908  6                      BEGIN       ! Redraw just this virtual display
;  1848          1909  7                      IF NOT (STATUS = SMG$$MOVE_TEXT_TO_WINDOW_BUF (.CURR_PP))
;  1849          1910  6                      THEN
;  1850          1911  6                          RETURN (.STATUS);
;  1851          1912  6
;  1852          1913  6                      !+
;  1853          1914  6                      ! If something actually got redrawn, tell the output
```

```
; 1854          1915  6                   ! routines we've changed the whole virtual display.
; 1855          1916  6                   ! If just one line has changed, however, then tell
; 1856          1917  6                   ! the output routines which line changed.
; 1857          1918  6                   !-
; 1858          1919  6                   IF .CURR_PP [PP_W_ROWS_TO_MOVE] NEQ 0
; 1859          1920  6                   THEN
; 1860          1921  7                       BEGIN
; 1861          1922  7                       PBCB [PBCB_W_FIRST_CHANGED_ROW] =
; 1862          1923  7                               .CURR_PP [PP_W_FIRST_WCB_ROW];
; 1863          1924  7
; 1864          1925  7                       PBCB [PBCB_W_LAST_CHANGED_ROW]  =
; 1865          1926  7                               .CURR_PP [PP_W_LAST_WCB_ROW];
; 1866          1927  7
; 1867          1928  7                       PBCB [PBCB_W_FIRST_CHANGED_COL] =
; 1868          1929  7                               .CURR_PP [PP_W_FIRST_WCB_COL];
; 1869          1930  7
; 1870          1931  7                       PBCB [PBCB_W_LAST_CHANGED_COL]  =
; 1871          1932  7                               .CURR_PP [PP_W_LAST_WCB_COL];
; 1872          1933  6                       END;
; 1873          1934  6
; 1874          1935  6                   IF ACTUALCOUNT () GEQ 3 ! If REQUEST_ARG supplied
; 1875          1936  6                   THEN
; 1876          1937  6                       IF .REQUEST_CODE EQL SMG$C_PUT_CHARS AND
; 1877          1938  6                          .REQUEST_ARG  NEQ 0
; 1878          1939  6                       THEN
; 1879          1940  7                           BEGIN       ! Just one row changed
; 1880          1941  7                           PBCB [PBCB_W_FIRST_CHANGED_ROW] =
; 1881          1942  7                               .CURR_PP [PP_W_ROW]+.REQUEST_ARG-1;
; 1882          1943  7
; 1883          1944  7                           PBCB [PBCB_W_LAST_CHANGED_ROW]  =
; 1884          1945  7                               .CURR_PP [PP_W_ROW]+.REQUEST_ARG-1;
; 1885          1946  7
; 1886          1947  6                           END;        ! Just one row changed
; 1887          1948  6
; 1888          1949  6                   !+
; 1889          1950  6                   ! If REQUEST_CODE is present and it is of a type
; 1890          1951  6                   ! that could have added a border, we must redraw the
; 1891          1952  6                   ! border at this time.
; 1892          1953  6                   !-
; 1893          1954  6                   IF ACTUALCOUNT () GEQ 2 ! If REQUEST_CODE supplied
; 1894          1955  6                   THEN
; 1895          1956  6                       IF .REQUEST_CODE EQL SMG$C_LABEL_BORDER
; 1896          1957  6                       THEN
; 1897          1958  6                           IF .DCB [DCB_V_BORDERED]
; 1898          1959  6                           THEN
; 1899          1960  7                               IF NOT (STATUS = SMG$$DRAW_BORDER (
; 1900          1961  7                                                .DCB, .CURR_PP, .WCB))
; 1901          1962  6                               THEN
; 1902          1963  6                                   RETURN (.STATUS);
; 1903          1964  6
; 1904          1965  6                       SMG$$MIN_UPD (.PBCB);
; 1905          1966  6
; 1906          1967  6                       END         ! Redraw just this virtual display
; 1907          1968  6
; 1908          1969  5               ELSE
; 1909          1970  5
; 1910          1971  6                   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>
```

```
;  1932          1992  1 %SBTTL 'SMG$$CHECK_FOR_OUTPUT_PBCB - Check to see if output needed'
;  1933          1993  1 GLOBAL ROUTINE SMG$$CHECK_FOR_OUTPUT_PBCB (
;  1934          1994  1                                           PBCB : REF BLOCK [,BYTE]
;  1935          1995  1                                         ) =
;  1936          1996  1 !++
;  1937          1997  1 ! FUNCTIONAL DESCRIPTION:
;  1938          1998  1 !
;  1939          1999  1 !       This routine is called when a change has been made to a
;  1940          2000  1 !       particular pasteboard that requires the entire screen image to
;  1941          2001  1 !       be recomposed, e.g. as a result of an UNPASTE operation.
;  1942          2002  1 !
;  1943          2003  1 ! CALLING SEQUENCE:
;  1944          2004  1 !
;  1945          2005  1 !       ret_status.wlc.v = SMG$$CHECK_FOR_OUTPUT_PBCB ( PBCB.rab.r)
;  1946          2006  1 !
;  1947          2007  1 ! FORMAL PARAMETERS:
;  1948          2008  1 !
;  1949          2009  1 !       PBCB.rab.r      Address of pasteboard control block for
;  1950          2010  1 !                       affected display.
;  1951          2011  1 !
;  1952          2012  1 ! IMPLICIT INPUTS:
;  1953          2013  1 !
;  1954          2014  1 !       PBCB [PBCB_L_BATCH_LEVEL]
;  1955          2015  1 !
;  1956          2016  1 ! IMPLICIT OUTPUTS:
;  1957          2017  1 !
;  1958          2018  1 !       NONE
;  1959          2019  1 !
;  1960          2020  1 ! COMPLETION STATUS:
;  1961          2021  1 !
;  1962          2022  1 !       SMG$_BATWAS_ON  Ok, but batching was on so nothing happened
;  1963          2023  1 !       SS$_NORMAL      Normal successful completion
;  1964          2024  1 !
;  1965          2025  1 ! SIDE EFFECTS:
;  1966          2026  1 !
;  1967          2027  1 !       NONE
;  1968          2028  1 !--
;  1969          2029  1
;  1970          2030  2     BEGIN
;  1971          2031  2     LOCAL
;  1972          2032  2         WCB : REF BLOCK [,BYTE],        ! Address of window control
;  1973          2033  2                                         ! block
;  1974          2034  2         STATUS;                         ! Status of subroutine calls
;  1975          2035  2
;  1976          2036  2     WCB = .PBCB [PBCB_A_WCB];
;  1977          2037  2
;  1978          2038  2 !+
;  1979          2039  2 ! Do nothing if batching is in effect.
;  1980          2040  2 !-
;  1981          2041  2
;  1982          2042  2     IF .PBCB [PBCB_L_BATCH_LEVEL] NEQ 0
;  1983          2043  2     THEN    RETURN SMG$_BATWAS_ON;
;  1984          2044  2
;  1985          2045  2 !+
;  1986          2046  2 ! Clear text buffer to blanks and attribute buffer to zeroes before we
;  1987          2047  2 ! start.  Clear alternate character set buffer to zeroes if it exists.
;  1988          2048  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$                                  ;
```

```
                                                                J 13
      28    A6            00            6E            00  2C 0002D          MOVC5   #0, (SP), #0, 40(WCB), @16(WCB)          ; 2055
                                                      10  B( 00033                                                          ;
                                          50        02  A6  3C 00035 2$:      MOVZWL  2(WCB), R0                           ; 2060
                                                        50  D6 00039          INCL    R0                                   ;
            50            00            6E            00  2C 0003B          MOVC5   #0, (SP), #0, R0, @44(WCB)           ;
                                                      2C  B6 00040                                                         ;
                                  00A8    C7            01  B0 00042          MOVW    #1, 168(R7)                          ; 2065
                                  00AA    C7        5F  A7  9B 00047          MOVZBW  95(R7), 170(R7)                      ; 2066
                                  00AC    C7            01  B0 0004D          MOVW    #1, 172(R7)                          ; 2067
                                  00AE    C7        5A  A7  B0 00052          MOVW    90(R7), 174(R7)                      ; 2068
                                          57            67  D1 00058          CMPL    (R7), R7                             ; 2075
                                                        0A  12 0005B          BNEQ    3$                                   ;
                                                        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>
```

```
;  2029          2088  1 %SBTTL 'SMG$$DRAW_BORDER - Move border characters into buffer'
;  2030          2089  1 GLOBAL ROUTINE SMG$$DRAW_BORDER (
;  2031          2090  1         DCB : REF BLOCK [,BYTE],        ! Display control block address
;  2032          2091  1         PP  : REF BLOCK [,BYTE],        ! Pasting packet address
;  2033          2092  1         WCB : REF BLOCK [,BYTE]         ! Window control block address
;  2034          2093  1                                 ) =
;  2035          2094  1 !++
;  2036          2095  1 ! FUNCTIONAL DESCRIPTION:
;  2037          2096  1 !
;  2038          2097  1 !       Draw the border characters into the text buffer of the WCB.
;  2039          2098  1 !       The cell in the text buffer has bits set to indicate which
;  2040          2099  1 !       elements of a border character will be needed in this cell
;  2041          2100  1 !       position.  When the first such bit is set in the text cell,
;  2042          2101  1 !       the corresponding cell in the attribute array is flag with
;  2043          2102  1 !       a bit which signals that the text array cell no longer contains
;  2044          2103  1 !       a printable character, but an encoding of the border charac*er.
;  2045          2104  1 !
;  2046          2105  1 ! CALLING SEQUENCE:
;  2047          2106  1 !
;  2048          2107  1 !       ret_status.wlc.v = SMG$$DRAW_BORDER (   DCB.rab.r,
;  2049          2108  1 !                                               PP.rab.r,
;  2050          2109  1 !                                               WCB.rab.r)
;  2051          2110  1 !
;  2052          2111  1 ! FORMAL PARAMETERS:
;  2053          2112  1 !
;  2054          2113  1 !       DCB.rab.r               ! Display control block address
;  2055          2114  1 !       PP.rab.r                ! Pasting packet address
;  2056          2115  1 !       WCB.rab.r               ! Window control block address
;  2057          2116  1 !
;  2058          2117  1 ! IMPLICIT INPUTS:
;  2059          2118  1 !
;  2060          2119  1 !       NONE
;  2061          2120  1 !
;  2062          2121  1 ! IMPLICIT OUTPUTS:
;  2063          2122  1 !
;  2064          2123  1 !       NONE
;  2065          2124  1 !
;  2066          2125  1 ! COMPLETION STATUS:
;  2067          2126  1 !
;  2068          2127  1 !       SS$_NORMAL      Normal successful completion
;  2069          2128  1 !
;  2070          2129  1 ! SIDE EFFECTS:
;  2071          2130  1 !
;  2072          2131  1 !       NONE
;  2073          2132  1 !--
;  2074          2133  1
;  2075          2134  2     BEGIN
;  2076          2135  2
;  2077          2136  2 !+
;  2078          2137  2 ! Macro $MARKIT checks a "text" character to see if it is already a
;  2079          2138  2 ! border element in which case it OR's in new border contribution.
;  2080          2139  2 ! If it is not yet flagged as being a border element it stores the
;  2081          2140  2 ! border contribution and marks attribute byte to record this is a
;  2082          2141  2 ! border element.
;  2083          2142  2 !-
;  2084          2143  2 MACRO
;  2085        M 2144  2   $MARKIT ( OFFSET, ELEMENT ) =
```

```
; 2086        M 2145  2     BEGIN
; 2087        M 2146  2     IF .(PTRA [.OFFSET])<ATTR_V_BORD_ELEM, 1>
; 2088        M 2147  2     THEN
; 2089        M 2148  2         PTRT [.OFFSET] = .PTRT [.OFFSET] OR ELEMENT
; 2090        M 2149  2     ELSE
; 2091        M 2150  2         BEGIN
; 2092        M 2151  2         PTRT [.OFFSET] = ELEMENT;
; 2093        M 2152  2         PTRA [.OFFSET] = ATTR_M_BORD_ELEM ;
; 2094        M 2153  2         END;
; 2095          2154  2     END %;
; 2096          2155  2
; 2097          2156  2 !+
; 2098          2157  2 ! Macro $CALC_LEFT_INDEX computes the byte offset into the WCB text and
; 2099          2158  2 ! attribute buffers where a left border element needs to be placed for
; 2100          2159  2 ! a double-wide line to make it come out in the correct column.
; 2101          2160  2 !-
; 2102          2161  2 MACRO
; 2103        M 2162  2     $CALC_LEFT_INDEX =
; 2104        M 2163  2     BEGIN
; 2105        M 2164  2         SINDEX = (.WLN -1) * .WCB [WCB_W_NO_COLS] +
; 2106        M 2165  2                             (.PP [PP_W_FIRST_WCB_COL] / 2) -1;
; 2107          2166  2     END %;
; 2108          2167  2
; 2109          2168  2 !+
; 2110          2169  2 ! Macro $CALC_RIGHT_INDEX computes the byte offset into the WCB text and
; 2111          2170  2 ! attribute buffers where a right border element needs to be placed for
; 2112          2171  2 ! a double-wide line to make it come out in the correct column.
; 2113          2172  2 !-
; 2114          2173  2 MACRO
; 2115        M 2174  2     $CALC_RIGHT_INDEX =
; 2116        M 2175  2     BEGIN
; 2117        M 2176  2         SINDEX = (.WLN -1) * .WCB [WCB_W_NO_COLS] +
; 2118        M 2177  2                  ((.PP [PP_W_FIRST_WCB_COL] + .PP [PP_W_MOVE_LENGTH] +1) / 2) -1;
; 2119          2178  2     END %;
; 2120          2179  2
; 2121          2180  2     LOCAL
; 2122          2181  2         SPOS,
; 2123          2182  2         ! Position where the first (or only) border element will be
; 2124          2183  2         ! written.  This is a displacement from the beginning of the
; 2125          2184  2         ! window buffer.  Before using this position, we must always
; 2126          2185  2         ! check to see if it points into the confines of the buffer.
; 2127          2186  2         ! This basically tells us if that portion of the border is
; 2128          2187  2         ! visible considering how the virtual display is pasted to the
; 2129          2188  2         ! pasteboard.
; 2130          2189  2
; 2131          2190  2         LDES : REF BLOCK [,BYTE],       ! Address of the dynamic string
; 2132          2191  2                                         ! descriptor in the DCB which
; 2133          2192  2                                         ! records the border label text
; 2134          2193  2                                         ! string.
; 2135          2194  2
; 2136          2195  2         PTRT : REF VECTOR [,BYTE],      ! pointer to top of WCB TEXT buf
; 2137          2196  2         PTRA : REF VECTOR [,BYTE],      ! pointer to top of WCB ATTR buf
; 2138          2197  2         UPPER_ROW,      ! Row number of top border
; 2139          2198  2         LOWER_ROW,      ! Row number of bottom border
; 2140          2199  2         LEFT_COL,       ! Col number of left border
; 2141          2200  2         RIGHT_COL,      ! Col number of right border
; 2142          2201  2         WCB_LCV : REF VECTOR [,BYTE],   ! Line control block
```

```
; 2143          2202  2          PBCB : REF BLOCK [,BYTE];       ! Addr. of pasteboard control
; 2144          2203  2                                          ! block
; 2145          2204  2
; 2146          2205  2
; 2147          2206  2      PTRT = .WCB [WCB_A_TEXT_BUF];
; 2148          2207  2      PTRA = .WCB [WCB_A_ATTR_BUF];
; 2149          2208  2      PBCB = .PP [PP_A_PBCB_ADDR];
; 2150          2209  2      WCB_LCV = .WCB [WCB_A_LINE_CHAR];
; 2151          2210  2
; 2152          2211  2  !+
; 2153          2212  2  ! If no part of the virtual display projects to the visible part of the
; 2154          2213  2  ! pasteboard, abandon attempt to draw border.  This is strictly speaking not
; 2155          2214  2  ! the correct solution since it ignores the case where a virtual display is
; 2156          2215  2  ! pasted such that it is not visible, but its border should be.  Trying to
; 2157          2216  2  ! make this pathelogical case work will require some coordinated work on the
; 2158          2217  2  ! part of SMG$$CALC_PASTE_TRANS.  (See corresponding note there.)
; 2159          2218  2  !-
; 2160          2219  2      IF .PP [PP_W_ROWS_TO_MOVE] EQL 0  OR
; 2161          2220  2         .PP [PP_W_MOVE_LENGTH]  EQL 0
; 2162          2221  2      THEN
; 2163          2222  2          RETURN (SS$_NORMAL);
; 2164          2223  2
; 2165          2224  2  !+
; 2166          2225  2  ! Compute the row and column numbers where the borders fall.  Note these
; 2167          2226  2  ! rows and columns may not map into buffer and need to be validated
; 2168          2227  2  ! before use.
; 2169          2228  2  !-
; 2170          2229  2      UPPER_ROW = .PP [PP_W_ROW] - 1;
; 2171          2230  2      LOWER_ROW = MAX ( .PP [PP_W_ROW], 1) + .PP [PP_W_ROWS_TO_MOVE];
; 2172          2231  2      LEFT_COL  = .PP [PP_W_COL] - 1;
; 2173          2232  2      RIGHT_COL = MAX ( .PP [PP_W_COL], 1) + .PP [PP_W_MOVE_LENGTH];
; 2174          2233  2
; 2175          2234  2  !+
; 2176          2235  2  ! If the line above where the virtual display is pasted exists in the
; 2177          2236  2  ! window buffer, draw the the top border.
; 2178          2237  2  !-
; 2179          2238  2      IF .UPPER_ROW GEQ 1
; 2180          2239  2      THEN
; 2181          2240  3          BEGIN   ! Top border
; 2182          2241  3
; 2183          2242  3              !+
; 2184          2243  3              ! If WCB is special, blank it and force line to normal.
; 2185          2244  3              !-
; 2186          2245  3
; 2187          2246  3              IF (.WCB_LCV[.UPPER_ROW] NEQ 0) AND .PBCB[PBCB_V_WIDE]
; 2188          2247  3              THEN
; 2189          2248  4                  BEGIN   ! Force line normal
; 2190          2249  4                  CH$FILL(%C' ',.WCB[WCB_W_NO_COLS]
; 2191          2250  4                          PTRT[ (.UPPER_ROW-1)*.WCB[WCB_W_NO_COLS] ]);
; 2192          2251  4                  CH$FILL(%C' ',.WCB[WCB_W_NO_COLS]
; 2193          2252  4                          PTRA[ (.UPPER_ROW-1)*.WCB[WCB_W_NO_COLS] ]);
; 2194          2253  4                  WCB_LCV[.UPPER_ROW]=0
; 2195          2254  3                  END;    ! Force line normal
; 2196          2255  3
; 2197          2256  3          SPOS = .PP [PP_W_TO_INDEX] - .WCB [WCB_W_NO_COLS] ;
; 2198          2257  3          INCR I FROM 1 TO .PP [PP_W_MOVE_LENGTH]
; 2199          2258  3          DO
```

```
;  2200          2259  4              BEGIN       ! Top line loop
;  2201          2260  4              $MARKIT (SPOS, BORD_M_HORIZ );
;  2202          2261  4              (PTRT [.SPOS])<BORD_V_DOWN,1> = 0; ! Knock down "down" seg.
;  2203          2262  4              SPOS = .SPOS + 1;
;  2204          2263  3              END;        ! Top line loop
;  2205          2264  3
;  2206          2265  3          IF .UPPER_ROW LSS .PBCB [PBCB_W_FIRST_CHANGED_ROW]
;  2207          2266  3          THEN
;  2208          2267  3              PBCB [PBCB_W_FIRST_CHANGED_ROW] = .UPPER_ROW;
;  2209          2268  3
;  2210          2269  2          END;     ! Top border
;  2211          2270  2
;  2212          2271  2      !+
;  2213          2272  2      ! If the line below where the virtual display is pasted exists in the
;  2214          2273  2      ! window buffer, draw the the bottom border.
;  2215          2274  2      !-
;  2216          2275  2          IF .LOWER_ROW LEQ .WCB [WCB_W_NO_ROWS]
;  2217          2276  2          THEN
;  2218          2277  3              BEGIN   ! Bottom border
;  2219          2278  3
;  2220          2279  3                  !+
;  2221          2280  3                  ! If WCB is special, blank it and force line to normal.
;  2222          2281  3                  !-
;  2223          2282  3
;  2224          2283  3                  IF (.WCB_LCV[.LOWER_ROW] NEQ 0) AND .PBCB[PBCB_V_WIDE]
;  2225          2284  3                  THEN
;  2226          2285  4                      BEGIN   ! Force line normal
;  2227          2286  4                      CH$FILL(%C' ',.WCB[WCB_W_NO_COLS],
;  2228          2287  4                              PTRT[ (.LOWER_ROW-1)*.WCB[WCB_W_NO_COLS] ]);
;  2229          2288  4                      CH$FILL(%C' ',.WCB[WCB_W_NO_COLS],
;  2230          2289  4                              PTRA[ (.LOWER_ROW-1)*.WCB[WCB_W_NO_COLS] ]);
;  2231          2290  4                      WCB_LCV[.LOWER_ROW]=0
;  2232          2291  3                      END;    ! Force line normal
;  2233          2292  3
;  2234          2293  4          SPOS = .PP [PP_W_TO_INDEX] + (.PP  [PP_W_ROWS_TO_MOVE] *
;  2235          2294  3                                        .WCB [WCB_W_NO_COLS] ) ;
;  2236          2295  3          INCR I FROM 1 TO .PP [PP_W_MOVE_LENGTH]
;  2237          2296  3          DO
;  2238          2297  4              BEGIN       ! Bottom line loop
;  2239          2298  4              $MARKIT (SPOS, BORD_M_HORIZ );
;  2240          2299  4              (PTRT [.SPOS])<BORD_V_UP,1> = 0;    ! Knock down "up" seg.
;  2241          2300  4              SPOS = .SPOS + 1;
;  2242          2301  3              END;        ! Bottom line loop
;  2243          2302  3
;  2244          2303  3          IF .LOWER_ROW GTR .PBCB [PBCB_W_LAST_CHANGED_ROW]
;  2245          2304  3          THEN
;  2246          2305  3              PBCB [PBCB_W_LAST_CHANGED_ROW] = .LOWER_ROW;
;  2247          2306  3
;  2248          2307  2          END;     ! Bottom border
;  2249          2308  2
;  2250          2309  2      !+
;  2251          2310  2      ! If the column to the left of where the virtual display is pasted
;  2252          2311  2      ! exists in the window buffer, draw the left border.
;  2253          2312  2      !-
;  2254          2313  2          IF .LEFT_COL GEQ 1
;  2255          2314  2          THEN
;  2256          2315  3              BEGIN   ! Left border
```

```
;  2257          2316  3          LOCAL
;  2258          2317  3              WLN,            ! Line number in WCB buffers
;  2259          2318  3              LCV : REF VECTOR [,BYTE]; ! Addr of line characteristics
;  2260          2319  3                                       ! vector in WCB
;  2261          2320  3          WLN = .PP [PP_W_FIRST_WCB_ROW];
;  2262          2321  3          LCV = .WCB [WCB_A_LINE_CHAR];
;  2263          2322  3          SPOS = .PP [PP_W_TO_INDEX] - 1;
;  2264          2323  3          INCR I FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
;  2265          2324  3          DO
;  2266          2325  4              BEGIN       ! Left column loop
;  2267          2326  4              IF .LCV [.WLN] EQL 0 OR             ! If normal or
;  2268          2327  5                 (NOT .PBCB [PBCB_V_WIDE])        ! DWDH not supported
;  2269          2328  4              THEN
;  2270          2329  5                  BEGIN   ! Normal case
;  2271          2330  5                  $MARKIT (SPOS, BORD_M_VERT ) ;
;  2272          2331  5                  (PTRT [.SPOS])<BORD_V_RIGHT,1> = 0; ! Knock down "right" seg.
;  2273          2332  5                  END     ! Normal case
;  2274          2333  4              ELSE
;  2275          2334  5                  BEGIN   ! Double-wide or Double-wide/Double-high case
;  2276          2335  5                  LOCAL
;  2277          2336  5                      SINDEX;     ! Special index for DWDH
;  2278          2337  5                  $CALC_LEFT_INDEX ;
;  2279          2338  5                  $MARKIT (SINDEX, BORD_M_VERT);
;  2280          2339  5                  (PTRT [.SINDEX])<BORD_V_RIGHT,1> = 0; ! Delete "right"
;  2281          2340  4                  END;    ! Double-wide or Double-wide/Double-high case
;  2282          2341  4              SPOS = .SPOS + .WCB [WCB_W_NO_COLS] ;
;  2283          2342  4              WLN = .WLN + 1;
;  2284          2343  3              END;        ! Left column loop
;  2285          2344  3
;  2286          2345  3          IF .LEFT_COL LSS .PBCB [PBCB_W_FIRST_CHANGED_COL]
;  2287          2346  3          THEN
;  2288          2347  3              PBCB [PBCB_W_FIRST_CHANGED_COL] = .LEFT_COL;
;  2289          2348  3
;  2290          2349  2          END;    ! Left border
;  2291          2350  2
;  2292          2351  2 !+
;  2293          2352  2 ! If the column to the right of where the virtual display is pasted
;  2294          2353  2 ! exists in the window buffer, draw the right border.
;  2295          2354  2 !-
;  2296          2355  2      IF .RIGHT_COL LEQ .WCB [WCB_W_NO_COLS]
;  2297          2356  2      THEN
;  2298          2357  3          BEGIN   ! Right border
;  2299          2358  3          LOCAL
;  2300          2359  3              WLN,            ! Line number in WCB buffers
;  2301          2360  3              LCV : REF VECTOR [,BYTE]; ! Addr of line characteristics
;  2302          2361  3                                       ! vector in WCB
;  2303          2362  3          WLN = .PP [PP_W_FIRST_WCB_ROW];
;  2304          2363  3          LCV = .WCB [WCB_A_LINE_CHAR];
;  2305          2364  3          SPOS = .PP [PP_W_TO_INDEX] + .PP [PP_W_MOVE_LENGTH];
;  2306          2365  3          INCR I FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
;  2307          2366  3          DO
;  2308          2367  4              BEGIN       ! Right column loop
;  2309          2368  4              IF .LCV [.WLN] EQL 0 OR             ! If normal or
;  2310          2369  5                 (NOT .PBCB [PBCB_V_WIDE])        ! DWDH not supported
;  2311          2370  4              THEN
;  2312          2371  5                  BEGIN   ! Normal case
;  2313          2372  5                  $MARKIT (SPOS, BORD_M_VERT ) ;
```

```
; 2314          2373  5                 (PTRT [.SPOS])<BORD_V_LEFT,1> = 0; ! Knock down "left" seg.
; 2315          2374  5                 END     ! Normal case
; 2316          2375  4             ELSE
; 2317          2376  5                 BEGIN   ! Double-wide or Double-wide/Double-high case
; 2318          2377  5                 LOCAL
; 2319          2378  5                     SINDEX;     ! Special index for DWDH
; 2320          2379  5                 $CALC_RIGHT_INDEX ;
; 2321          2380  5                 $MARKIT (SINDEX, BORD_M_VERT);
; 2322          2381  5                 (PTRT [.SINDEX])<BORD_V_LEFT,1> = 0; ! Delete "left"
; 2323          2382  4                 END;    ! Double-wide or Double-wide/Double-high case
; 2324          2383  4             SPOS = .SPOS + .WCB [WCB_W_NO_COLS] ;
; 2325          2384  4             WLN = .WLN + 1;
; 2326          2385  3             END;        ! Right column loop
; 2327          2386  3
; 2328          2387  3         IF .RIGHT_COL GTR .PBCB [PBCB_W_LAST_CHANGED_COL]
; 2329          2388  3         THEN
; 2330          2389  3             PBCB [PBCB_W_LAST_CHANGED_COL] = .RIGHT_COL;
; 2331          2390  3
; 2332          2391  2         END;    ! Right border
; 2333          2392  2
; 2334          2393  2 !+
; 2335          2394  2 ! Handle upper left-hand corner if position exists in buffer.
; 2336          2395  2 !-
; 2337          2396  2     IF .UPPER_ROW GEQ 1         AND     ! Row exists
; 2338          2397  2        .LEFT_COL  GEQ 1                 ! Col exists
; 2339          2398  2     THEN
; 2340          2399  3         BEGIN   ! Upper left-hand corner
; 2341          2400  3         SPOS = .PP [PP_W_TO_INDEX] - .WCB [WCB_W_NO_COLS] - 1;
; 2342          2401  3         $MARKIT (SPOS, BORD_M_ULCORN ) ;
; 2343          2402  2         END;    ! Upper left-hand corner
; 2344          2403  2
; 2345          2404  2 !+
; 2346          2405  2 ! Handle upper right-hand corner if position exists in buffer.
; 2347          2406  2 !-
; 2348          2407  2     IF .UPPER_ROW GEQ 1         AND             ! Row exists
; 2349          2408  2        .RIGHT_COL LEQ .WCB [WCB_W_NO_COLS]      ! Col exists
; 2350          2409  2     THEN
; 2351          2410  3         BEGIN   ! Upper right-hand corner
; 2352          2411  3         SPOS = .PP [PP_W_TO_INDEX] - .WCB [WCB_W_NO_COLS] +
; 2353          2412  3                  .PP [PP_W_MOVE_LENGTH] ;
; 2354          2413  3         $MARKIT (SPOS, BORD_M_URCORN ) ;
; 2355          2414  2         END;    ! Upper right-hand corner
; 2356          2415  2
; 2357          2416  2 !+
; 2358          2417  2 ! Handle lower left-hand corner if position exists in buffer.
; 2359          2418  2 !-
; 2360          2419  2     IF .LOWER_ROW LEQ .WCB [WCB_W_NO_ROWS]  AND ! Row exists
; 2361          2420  2        .LEFT_COL  GEQ 1                         ! Col exists
; 2362          2421  2     THEN
; 2363          2422  3         BEGIN   ! Lower left-hand corner
; 2364          2423  4         SPOS = .PP [PP_W_TO_INDEX] + (.PP [PP_W_ROWS_TO_MOVE] *
; 2365          2424  3                                       .WCB [WCB_W_NO_COLS]) - 1;
; 2366          2425  3         $MARKIT (SPOS, BORD_M_LLCORN ) ;
; 2367          2426  2         END;    ! Lower left-hand corner
; 2368          2427  2
; 2369          2428  2 !+
; 2370          2429  2 ! Handle lower right-hand corner if position exists in buffer.
```

```
;  2371          2430  2 !-
;  2372          2431  2     IF .LOWER_ROW LEQ .WCB [WCB_W_NO_ROWS]  AND ! Row exists
;  2373          2432  2        .RIGHT_COL LEQ .WCB [WCB_W_NO_COLS]      ! Col exists
;  2374          2433  2     THEN
;  2375          2434  3         BEGIN   ! Lower right-hand corner
;  2376          2435  3         SPOS = .PP [PP_W_TO_INDEX] +
;  2377          2436  3                     (.PP [PP_W_ROWS_TO_MOVE] * .WCB [WCB_W_NO_COLS]) +
;  2378          2437  3                     .PP [PP_W_MOVE_LENGTH] ;
;  2379          2438  3         $MARKIT (SPOS, BORD_M_LRCORN ) ;
;  2380          2439  2         END;    ! Lower right-hand corner
;  2381          2440  2
;  2382          2441  2 !+
;  2383          2442  2 ! Check to see if this border is labeled and if so, move the label text
;  2384          2443  2 ! into the approriate border area.
;  2385          2444  2 ! All the fields in the PP accessed below have been previously computed
;  2386          2445  2 ! at the time the virtual display was pasted to the pasteboard so
;  2387          2446  2 ! that they need not be recomputed each time through here.  The lengths
;  2388          2447  2 ! used in the various CH$MOVE's and CH$FILL's have been so computed
;  2389          2448  2 ! that if no parts of the label fit, the lengths used are zero.
;  2390          2449  2 !-
;  2391          2450  2     LDES = DCB [DCB_Q_LABEL_DESC];
;  2392          2451  2     IF .LDES [DSC$W_LENGTH] NEQ 0                ! If label exists...
;  2393          2452  2     THEN
;  2394          2453  3         BEGIN   ! Overall border label processing
;  2395          2454  3         CASE .DCB [DCB_B_LABEL_POS] FROM SMG$K_TOP TO SMG$K_RIGHT OF
;  2396          2455  3         SET
;  2397          2456  3             [SMG$K_TOP]:
;  2398          2457  4                 BEGIN   ! Label in top row
;  2399          2458  4                 !+
;  2400          2459  4                 ! If row where label lands is not on the pasteboard, no
;  2401          2460  4                 ! movement will be necessary.
;  2402          2461  4                 !-
;  2403          2462  4                 IF .UPPER_ROW GEQ 1     ! If row above present...
;  2404          2463  4                 THEN
;  2405          2464  5                     BEGIN       ! Top present
;  2406          2465  5
;  2407          2466  5                     CH$MOVE (
;  2408          2467  5                         .PP [PP_W_LABEL_BYTES_TO_MOVE],  ! Length
;  2409          2468  5                         .LDES [DSC$A_POINTER] + .PP [PP_W_SRC_LABEL_OFF],  ! Source
;  2410          2469  5                         PTRT [.PP [PP_W_DST_LABEL_OFF]]);  ! Dest
;  2411          2470  5
;  2412          2471  5                     CH$FILL (
;  2413          2472  5                         .DCB [DCB_B_LABEL_REND],             ! Fill
;  2414          2473  5                         .PP [PP_W_LABEL_BYTES_TO_MOVE],      ! Number
;  2415          2474  5                         PTRA [.PP [PP_W_DST_LABEL_OFF]]);  ! Dest
;  2416          2475  5
;  2417          2476  4                     END;        ! Top present
;  2418          2477  3                 END;    ! Label in top row
;  2419          2478  3
;  2420          2479  3             [SMG$K_BOTTOM]:
;  2421          2480  4                 BEGIN   ! Label in bottom row
;  2422          2481  4                 !+
;  2423          2482  4                 ! If row where label lands is not on the pasteboard, no
;  2424          2483  4                 ! movement will be necessary.
;  2425          2484  4                 !-
;  2426          2485  4                 IF .LOWER_ROW LEQ .WCB [WCB_W_NO_ROWS] ! If row below present...
;  2427          2486  4                 THEN
```

```
;  2428          2487  5                      BEGIN       ! Bottom present
;  2429          2488  5
;  2430          2489  5                      CH$MOVE (
;  2431          2490  5                          .PP [PP_W_LABEL_BYTES_TO_MOVE],  ! Length
;  2432          2491  5                          .LDES [DSC$A_POINTER] + .PP [PP_W_SRC_LABEL_OFF],  ! Source
;  2433          2492  5                          PTRT [.PP [PP_W_DST_LABEL_OFF]]);  ! Dest
;  2434          2493  5
;  2435          2494  5                      CH$FILL (
;  2436          2495  5                          .DCB [DCB_B_LABEL_REND],            ! Fill
;  2437          2496  5                          .PP [PP_W_LABEL_BYTES_TO_MOVE],     ! Number
;  2438          2497  5                          PTRA [.PP [PP_W_DST_LABEL_OFF]]);  ! Dest
;  2439          2498  5
;  2440          2499  4                      END;        ! Bottom present
;  2441          2500  3                  END;    ! Label in bottom row
;  2442          2501  3
;  2443          2502  3              [SMG$K_LEFT]:
;  2444          2503  4                  BEGIN   ! Label in left column
;  2445          2504  4                  !+
;  2446          2505  4                  ! If column where label lands is not on the pasteboard,
;  2447          2506  4                  ! no movement will be necessary.
;  2448          2507  4                  !-
;  2449          2508  4                  IF .LEFT_COL GEQ 1      ! If column to left present...
;  2450          2509  4                  THEN
;  2451          2510  5                      BEGIN       ! Left present
;  2452          2511  5                      LOCAL
;  2453          2512  5                          WLN,                            ! Line # in WCB buffer
;  2454          2513  5                          LCV : REF VECTOR [,BYTE],       ! Addr of line
;  2455          2514  5                                                          ! characteristics vect.
;  2456          2515  5                                                          ! in WCB
;  2457          2516  5                      I;      ! Local index
;  2458          2517  6                  WLN = (.PP [PP_W_DST_LABEL_OFF] /
;  2459          2518  5                         .WCB [WCB_W_NO_COLS]) + 1;
;  2460          2519  5                  LCV = .WCB [WCB_A_LINE_CHAR];
;  2461          2520  5                  SPOS = .PP [PP_W_DST_LABEL_OFF];
;  2462          2521  5                  I = 0;
;  2463          2522  5                  UNTIL .I EQL .PP [PP_W_LABEL_BYTES_TO_MOVE]
;  2464          2523  5                  DO
;  2465          2524  6                      BEGIN   ! Character by character movement
;  2466          2525  6                      IF .LCV [.WLN] EQL 0 OR         ! If normal or
;  2467          2526  7                         (NOT .PBCB [PBCB_V_WIDE])    ! DWDH not supported
;  2468          2527  6                      THEN
;  2469          2528  7                          BEGIN ! Normal case
;  2470          2529  8                          PTRT [.SPOS] = .(.LDES [DSC$A_POINTER] +
;  2471          2530  7                                          .PP [PP_W_SRC_LABEL_OFF] + .I);
;  2472          2531  7                          PTRA [.SPOS] = .DCB [DCB_B_LABEL_REND];
;  2473          2532  7                          END   ! Normal case
;  2474          2533  6                      ELSE
;  2475          2534  7                          BEGIN ! Special case
;  2476          2535  7                          LOCAL
;  2477          2536  7                              SINDEX;
;  2478          2537  7                          $CALC_LEFT_INDEX ;
;  2479          2538  8                          PTRT [.SINDEX] = .(.LDES [DSC$A_POINTER] +
;  2480          2539  7                                          .PP [PP_W_SRC_LABEL_OFF] + .I);
;  2481          2540  7                          PTRA [.SINDEX] = .DCB [DCB_B_LABEL_REND];
;  2482          2541  6                          END;  ! Special case
;  2483          2542  6
;  2484          2543  6                      SPOS = .SPOS + .WCB [WCB_W_NO_COLS];
```

```
;  2485          2544  6                          I = .I+1;
;  2486          2545  6                          WLN = .WLN + 1;
;  2487          2546  5                          END;    ! Character by character movement
;  2488          2547  4                      END;        ! Left present
;  2489          2548  3                  END;    ! Label in left column
;  2490          2549  3
;  2491          2550  3              [SMG$K_RIGHT]:
;  2492          2551  4                  BEGIN   ! Label in right column
;  2493          2552  4                  !+
;  2494          2553  4                  ! If column where label lands is not on the pasteboard,
;  2495          2554  4                  ! no movement will be necessary.
;  2496          2555  4                  !-
;  2497          2556  4                  IF .RIGHT_COL LEQ .WCB [WCB_W_NO_COLS] ! If column to right present...
;  2498          2557  4                  THEN
;  2499          2558  5                      BEGIN       ! Right present
;  2500          2559  5                      LOCAL
;  2501          2560  5                          WLN,                        ! Line # in WCB buffer
;  2502          2561  5                          LCV : REF VECTOR [,BYTE],   ! Addr of line
;  2503          2562  5                                                      ! characteristics vect.
;  2504          2563  5                                                      ! in WCB
;  2505          2564  5                          I;      ! Local index
;  2506          2565  5                      I = 0;
;  2507          2566  6                      WLN = (.PP [PP_W_DST_LABEL_OFF] /
;  2508          2567  5                             .WCB [WCB_W_NO_COLS]) + 1;
;  2509          2568  5                      LCV = .WCB [WCB_A_LINE_CHAR];
;  2510          2569  5                      SPOS = .PP [PP_W_DST_LABEL_OFF];
;  2511          2570  5                      UNTIL .I EQL .PP [PP_W_LABEL_BYTES_TO_MOVE]
;  2512          2571  5                      DO
;  2513          2572  6                          BEGIN   ! Character by character movement
;  2514          2573  6                          IF .LCV [.WLN] EQL 0 OR         ! If normal or
;  2515          2574  7                             (NOT .PBCB [PBCB_V_WIDE])    ! DWDH not supported
;  2516          2575  6                          THEN
;  2517          2576  7                              BEGIN       ! Normal case
;  2518          2577  8                              PTRT [.SPOS] = .(.LDES [DSC$A_POINTER] +
;  2519          2578  7                                              .PP [PP_W_SRC_LABEL_OFF] + .I);
;  2520          2579  7                              PTRA [.SPOS] = .DCB [DCB_B_LABEL_REND];
;  2521          2580  7                              END         ! Normal case
;  2522          2581  6                          ELSE
;  2523          2582  7                              BEGIN       ! Special case
;  2524          2583  7                              LOCAL
;  2525          2584  7                                  SINDEX;
;  2526          2585  7                              $CALC_RIGHT_INDEX ;
;  2527          2586  8                              PTRT [.SINDEX] = .(.LDES [DSC$A_POINTER] +
;  2528          2587  7                                              .PP [PP_W_SRC_LABEL_OFF] + .I);
;  2529          2588  7                              PTRA [.SINDEX] = .DCB [DCB_B_LABEL_REND];
;  2530          2589  6                              END;        ! Special case
;  2531          2590  6                          SPOS = .SPOS + .WCB [WCB_W_NO_COLS];
;  2532          2591  6                          I = .I+1;
;  2533          2592  6                          WLN = .WLN + 1;
;  2534          2593  5                          END;    ! Character by character movement
;  2535          2594  4                      END;        ! Right present
;  2536          2595  3                  END;    ! Label in right column
;  2537          2596  3
;  2538          2597  3              [OUTRANGE]:
;  2539          2598  3                  RETURN (SMG$_FATERRLIB);
;  2540          2599  3          TES;
;  2541          2600  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     6$
                                        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        60  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$:     TSTB    (SPOS)[PTRA]                         ;2330
                                                  06  18 001E8          BGEQ    22$
                                8649              0A  88 001EA          BISB2   #10, (SPOS)[PTRT]
                                                  09  11 001EE          BRB     23$
                                8649              0A  90 001F0 22$:     MOVB    #10, (SPOS)[PTRT]
                                664A        80    8F  90 001F4          MOVB    #-128, (SPOS)[PTRA]
                                8649              01  8A 001F9 23$:     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$                        ;2343
                    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)                         ;
```

```
                                                J 14
                                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                    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$:     MOVZWL  38(R11), R1                                     ; 2491
                              51          04  A7 C0 003EA          ADDL2   4(LDES), R1                                     ;
                         24   AE          28  AB 3C 003EE          MOVZWL  40(R11), 36(SP)                                 ; 2492
                   24 BE49    61          24  AB 28 003F3          MOVC3   36(R11), (R1), @36(SP)[PTRT]                    ;
             24 AB      33 A8              6E      00 2C 003FA      MOVC5   #0, (SP), 51(R8), 36(R11), @36(SP)[PTRA]        ; 2497
                                                24 BE4A    00401
                                                  C^F4 31 00404 53$:     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$:     MOVAB   -1(R0), R1                              ; 26
                                51        18    AE  C4 00451          MULL2   24(SP), R1                              ;
                                52        33    AB  3C 00455          MOVZWL  51(R11), R2                             ;
                                52              02  C6 00459          DIVL2   #2, R2                                  ;
                                52        FF A241  9E 0045C          MOVAB   -1(R2)[R1], SINDEX                      ;
                                51        26    AB  3C 00461          MOVZWL  38(R11), R1                             ; 2539
                                51        04    A7  C0 00465          ADDL2   4(LDES), R1                             ;
                              6249              6441  90 00469          MOVB    (I)[R1], (SINDEX)[PTRT]                 ; 2538
                              624A        33    A8  90 0046E          MOVB    51(R8), (SINDEX)[PTRA]                  ; 2540
                                56        18    AE  C0 00473 58$:     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  CO 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  DO 004FB 64$:     MOVL    #1, RO                          ; 2602
                                                    04 004FE         RET                                     ; 2603

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

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

```
;  2547          2605  1 %SBTTL 'SMG$$FILL_WINDOW_BUFFER - Fill window with virtual displays'
;  2548          2606  1 GLOBAL ROUTINE SMG$$FILL_WINDOW_BUFFER (
;  2549          2607  1                                       PP : REF BLOCK [,BYTE]
;  2550          2608  1                                           ) =
;  2551          2609  1 !++
;  2552          2610  1 ! FUNCTIONAL DESCRIPTION:
;  2553          2611  1 !
;  2554          2612  1 !       This procedure rebuilds the portions of the window buffer that
;  2555          2613  1 !       need to be changed because some virtual display has changed.
;  2556          2614  1 !       Rebuilding takes place from the given pasting packet down the
;  2557          2615  1 !       chain to the most-recently pasted pasting packet.
;  2558          2616  1 !
;  2559          2617  1 ! CALLING SEQUENCE:
;  2560          2618  1 !
;  2561          2619  1 !       ret_status.wlc.v = SMG$$FILL_WINDOW_BUFFER ( PP.rab.r)
;  2562          2620  1 !
;  2563          2621  1 ! FORMAL PARAMETERS:
;  2564          2622  1 !
;  2565          2623  1 !
;  2566          2624  1 !       PP.rab.r                Address of pasting packet which
;  2567          2625  1 !                               determines the first display to be
;  2568          2626  1 !                               repainted.
;  2569          2627  1 !
;  2570          2628  1 ! IMPLICIT INPUTS:
;  2571          2629  1 !
;  2572          2630  1 !       NONE
;  2573          2631  1 !
;  2574          2632  1 ! IMPLICIT OUTPUTS:
;  2575          2633  1 !
;  2576          2634  1 !       NONE
;  2577          2635  1 !
;  2578          2636  1 ! COMPLETION STATUS:
;  2579          2637  1 !
;  2580          2638  1 !       SS$_NORMAL      Normal successful completion
;  2581          2639  1 !
;  2582          2640  1 ! SIDE EFFECTS:
;  2583          2641  1 !
;  2584          2642  1 !       NONE
;  2585          2643  1 !--
;  2586          2644  1
;  2587          2645  2     BEGIN
;  2588          2646  2     LOCAL
;  2589          2647  2         STATUS,                             ! Status of subr. calls
;  2590          2648  2         PBCB : REF BLOCK [,BYTE],           ! Address of pasteboard control
;  2591          2649  2                                             ! block
;  2592          2650  2
;  2593          2651  2         WCB : REF BLOCK [,BYTE],            ! Addr of current WCB
;  2594          2652  2
;  2595          2653  2         CURR_PP : REF BLOCK [,BYTE];        ! Addr of 2 longwords that form
;  2596          2654  2                                             ! queue header in PP currently
;  2597          2655  2                                             ! under inspection.
;  2598          2656  2
;  2599          2657  2     PBCB = .PP   [PP_A_PBCB_ADDR];
;  2600          2658  2     WCB  = .PBCB [PBCB_A_WCB];
;  2601          2659  2
;  2602          2660  2 !+
;  2603          2661  2 ! Change packet address to address of queue header.
```

```
; 2604          2662  2 !-
; 2605          2663  2      CUPR_PP = .PP + PP_PBCB_QUEUE_OFFSET; ! Start with specified packet
; 2606          2664  2
; 2607          2665  2 !+
; 2608          2666  2 ! Loop for all pasting packets starting with this one to the last-pasted
; 2609          2667  2 ! one...
; 2610          2668  2 !-
; 2611          2669  2      WHILE .CURR_PP NEQ PBCB [PBCB_A_PP_NEXT]
; 2612          2670  2      DO
; 2613          2671  3          BEGIN   ! For all displays that need to be rewritten
; 2614          2672  3          LOCAL
; 2615          2673  3              PP_BASE : REF BLOCK [,BYTE],        ! Base address of the PP
; 2616          2674  3              DCB : REF BLOCK [,BYTE]; ! Current virtual display that
; 2617          2675  3                                       ! needs to be repainted.
; 2618          2676  3
; 2619          2677  3          PP_BASE = .CURR_PP - PP_PBCB_QUEUE_OFFSET;
; 2620          2678  3                                  ! Since the queue headers for this part
; 2621          2679  3                                  ! of the chain are not at relative 0 in
; 2622          2680  3                                  ! the pasting packet.
; 2623          2681  3          DCB = .PP_BASE [PP_A_DCB_ADDR]; ! DCB addr of this pairing
; 2624          2682  3
; 2625          2683  4          IF NOT (STATUS = SMG$$MOVE_TEXT_TO_WINDOW_BUF (.PP_BASE))
; 2626          2684  3          THEN
; 2627          2685  3              RETURN (.STATUS);
; 2628          2686  3
; 2629          2687  3          !+
; 2630          2688  3          ! If bordered, move the border characters into the text buffer
; 2631          2689  3          ! and set the bit in the attribute bytes to indicate a border
; 2632          2690  3          ! element.
; 2633          2691  3          !-
; 2634          2692  3          IF .DCB [DCB_V_BORDERED]
; 2635          2693  3          THEN
; 2636          2694  4              IF NOT (STATUS = SMG$$DRAW_BORDER ( .DCB, .PP_BASE, .WCB))
; 2637          2695  3              THEN
; 2638          2696  3                  RETURN (.STATUS);
; 2639          2697  3
; 2640          2698  3          !+
; 2641          2699  3          ! Update changed area fields in the PBCB based on latest
; 2642          2700  3          ! contribution.
; 2643          2701  3          !-
; 2644          2702  3          IF .PP_BASE [PP_W_ROWS_TO_MOVE] NEQ 0
; 2645          2703  3          THEN
; 2646          2704  4              BEGIN
; 2647          2705  4              IF .PP_BASE [PP_W_FIRST_WCB_ROW] LSS .PBCB [PBCB_W_FIRST_CHANGED_ROW]
; 2648          2706  4              THEN
; 2649          2707  4                  PBCB [PBCB_W_FIRST_CHANGED_ROW] = .PP_BASE [PP_W_FIRST_WCB_ROW];
; 2650          2708  4
; 2651          2709  4              IF .PP_BASE [PP_W_LAST_WCB_ROW] GTR .PBCB [PBCB_W_LAST_CHANGED_ROW]
; 2652          2710  4              THEN
; 2653          2711  4                  PBCB [PBCB_W_LAST_CHANGED_ROW] = .PP_BASE [PP_W_LAST_WCB_ROW];
; 2654          2712  4
; 2655          2713  4              IF .PP_BASE [PP_W_FIRST_WCB_COL] LSS .PBCB [PBCB_W_FIRST_CHANGED_COL]
; 2656          2714  4              THEN
; 2657          2715  4                  PBCB [PBCB_W_FIRST_CHANGED_COL] = .PP_BASE [PP_W_FIRST_WCB_COL];
; 2658          2716  4
; 2659          2717  4              IF .PP_BASE [PP_W_LAST_WCB_COL] GTR .PBCB [PBCB_W_LAST_CHANGED_COL]
; 2660          2718  4              THEN
```

```
;  2661     2719  4                  PBCB [PBCB_W_LAST_CHANGED_COL] = .PP_BASE [PP_W_LAST_WCB_COL];
;  2662     2720  4
;  2663     2721  3              END;
;  2664     2722  3
;  2665     2723  3          !+
;  2666     2724  3          ! Walk this chain backwards, from the packet we started with
;  2667     2725  3          ! back to the head of the chain -- since the most recently
;  2668     2726  3          ! pasted displays are at the head of the chain.
;  2669     2727  3          !-
;  2670     2728  3          CURR_PP = .PP_BASE [PP_A_PREV_PBCB];
;  2671     2729  2          END;    ! For all displays that need to be rewritten
;  2672     2730  2
;  2673     2731  2
;  2674     2732  2      RETURN ( SMG$$MIN_UPD ( .PBCB ));    ! Cause window buffer to be
;  2675     2733  2                                           ! output
;  2676     2734  2
;  2677     2735  1      END;                  ! End of routine SMG$$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$:      CVTWL   172(PBCB), R1                       ; 2713
                51    33    A2          10  00  ED 00068          CMPZV   #0, #16, 51(PP_BASE), R1            ;
                                         06  18 00071          BGEQ    5$                                  ;
                      00AC  C4      33   A2  B0 00073          MOVW    51(PP_BASE), 172(PBCB)              ; 2715
                            51      00AE C4  32 00079 5$:      CVTWL   174(PBCB), R1                       ; 2717
                51    35    A2          10  00  ED 0007E          CMPZV   #0, #16, 53(PP_BASE), R1            ;
                                         06  15 00084          BLEQ    6$                                  ;
```

```
                                                D 15

                          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>
```

```
;  2680          2737  1 %SBTTL 'SMG$$MOVE_TEXT_TO_WINDOW_BUF - Move text from display buf. to window buf.'
;  2681          2738  1 GLOBAL ROUTINE SMG$$MOVE_TEXT_TO_WINDOW_BUF (
;  2682          2739  1                                           PP : REF $PP_DECL
;  2683          2740  1                                                ) =
;  2684          2741  1 !++
;  2685          2742  1 ! FUNCTIONAL DESCRIPTION:
;  2686          2743  1 !
;  2687          2744  1 !       This routine moves text from the buffer located at
;  2688          2745  1 !       .DCB [DCB_A_TEXT_BUF] into the window text buffer.
;  2689          2746  1 !       Array of bytes at .DCB [ DCB_A_ATTR_BUF ] describe the
;  2690          2747  1 !       rendition this text must assume and is moved into the
;  2691          2748  1 !       associated window attribute buffer.
;  2692          2749  1 !       Similarly, if the alternate character set buffer at
;  2693          2750  1 !       .DCB [DCB_A_CHAR_SET_BUF] exists, it must be mapped into the
;  2694          2751  1 !       window alternate character set buffer.
;  2695          2752  1 !
;  2696          2753  1 ! CALLING SEQUENCE:
;  2697          2754  1 !
;  2698          2755  1 !       ret_status.wlc.v = SMG$$MOVE_TEXT_TO_WINDOW_BUF ( PP.rab.r)
;  2699          2756  1 !
;  2700          2757  1 ! FORMAL PARAMETERS:
;  2701          2758  1 !
;  2702          2759  1 !       PP.rab.r                Address of pasting packet.
;  2703          2760  1 !
;  2704          2761  1 ! IMPLICIT INPUTS:
;  2705          2762  1 !
;  2706          2763  1 !       NONE
;  2707          2764  1 !
;  2708          2765  1 ! IMPLICIT OUTPUTS:
;  2709          2766  1 !
;  2710          2767  1 !       NONE
;  2711          2768  1 !
;  2712          2769  1 ! COMPLETION STATUS:
;  2713          2770  1 !
;  2714          2771  1 !       SS$_NORMAL      Normal successful completion
;  2715          2772  1 !
;  2716          2773  1 ! SIDE EFFECTS:
;  2717          2774  1 !
;  2718          2775  1 !       NONE
;  2719          2776  1 !--
;  2720          2777  1
;  2721          2778  2     BEGIN
;  2722          2779  2
;  2723          2780  2     BUILTIN
;  2724          2781  2         SKPC;
;  2725          2782  2
;  2726          2783  2     LOCAL
;  2727          2784  2         DCB  : REF BLOCK [,BYTE],       ! Addr of virtual display
;  2728          2785  2                                         ! control block.
;  2729          2786  2         PBCB : REF BLOCK [,BYTE],       ! Addr of pasteboard control
;  2730          2787  2                                         ! block
;  2731          2788  2         WCB  : REF BLOCK [,BYTE],       ! Addr of window control block
;  2732          2789  2         IN_LINE_CHAR  : REF VECTOR [,BYTE],     ! Addr. of line char.
;  2733          2790  2                                                 ! vector off DCB
;  2734          2791  2         OUT_LINE_CHAR : REF VECTOR [,BYTE],     ! Addr. of line char.
;  2735          2792  2                                                 ! vector off WCB
;  2736          2793  2
```

```
;  2737          2794  2          WCB_COLS,       ! Extracted .WCB [WCB_W_NO_COLS]
;  2738          2795  2          DCB_COLS,       ! Extracted .DCB [DCB_W_NO_COLS]
;  2739          2796  2          FROM_INDEX,     ! Pointer into DCB buffers
;  2740          2797  2          TO_INDEX,       ! Pointer into WCB buffers for normal mapping
;  2741          2798  2          SPEC_TO_INDEX,  ! Pointer into WCB buffers for special mapping
;  2742          2799  2          RE_BORDER : INITIAL (0);        ! Flag to indicate that the
;  2743          2800  2                                          ! border elements need to be
;  2744          2801  2                                          ! remapped because some line
;  2745          2802  2                                          ! changed from normal to DWDH or
;  2746          2803  2                                          ! vice versa.
;  2747          2804  2
;  2748          2805  2  MACRO
;  2749        M 2806  2      $MAP_LINE_NORMAL =
;  2750        M 2807  2          CH$MOVE ( .PP [PP_W_MOVE_LENGTH],               ! Length
;  2751        M 2808  2                    .DCB [DCB_A_TEXT_BUF] + .FROM_INDEX,  ! Source
;  2752        M 2809  2                    .WCB [WCB_A_TEXT_BUF] + .TO_INDEX);   ! Destination
;  2753          2810  2      %,
;  2754          2811  2
;  2755        M 2812  2      $MAP_LINE_SPECIAL =
;  2756        M 2813  2          CH$MOVE ( (.PP [PP_W_MOVE_LENGTH]-1)/2,         ! Length
;  2757        M 2814  2                    .DCB [DCB_A_TEXT_BUF] + .FROM_INDEX,  ! Source
;  2758        M 2815  2                    .WCB [WCB_A_TEXT_BUF] + .SPEC_TO_INDEX); ! Destination
;  2759          2816  2      %,
;  2760          2817  2
;  2761        M 2818  2      $MAP_ATTR_NORMAL =
;  2762        M 2819  2          CH$MOVE ( .PP [PP_W_MOVE_LENGTH],               ! Length
;  2763        M 2820  2                    .DCB [DCB_A_ATTR_BUF] + .FROM_INDEX,  ! Source
;  2764        M 2821  2                    .WCB [WCB_A_ATTR_BUF] + .TO_INDEX);   ! Destination
;  2765          2822  2      %,
;  2766          2823  2
;  2767        M 2824  2      $MAP_ATTR_SPECIAL =
;  2768        M 2825  2          CH$MOVE ( (.PP [PP_W_MOVE_LENGTH]-1)/2,         ! Length
;  2769        M 2826  2                    .DCB [DCB_A_ATTR_BUF] + .FROM_INDEX,  ! Source
;  2770        M 2827  2                    .WCB [WCB_A_ATTR_BUF] + .SPEC_TO_INDEX); ! Destination
;  2771          2828  2      %,
;  2772          2829  2
;  2773        M 2830  2      $MAP_ALT_CHAR_NORMAL =
;  2774        M 2831  2          IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
;  2775        M 2832  2          THEN
;  2776        M 2833  2          CH$MOVE ( .PP [PP_W_MOVE_LENGTH],               ! Length
;  2777        M 2834  2                    .DCB [DCB_A_CHAR_SET_BUF] + .FROM_INDEX, ! Source
;  2778        M 2835  2                    .WCB [WCB_A_CHAR_SET_BUF] + .TO_INDEX);  ! Dest.
;  2779          2836  2      %,
;  2780          2837  2
;  2781        M 2838  2      $MAP_ALT_CHAR_SPECIAL =
;  2782        M 2839  2          IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
;  2783        M 2840  2          THEN
;  2784        M 2841  2          CH$MOVE ( (.PP [PP_W_MOVE_LENGTH]-1)/2,         ! Length
;  2785        M 2842  2                    .DCB [DCB_A_CHAR_SET_BUF] + .FROM_INDEX,  ! Source
;  2786        M 2843  2                    .WCB [WCB_A_CHAR_SET_BUF] + .SPEC_TO_INDEX);  ! Dest.
;  2787          2844  2      %,
;  2788          2845  2
;  2789        M 2846  2      $CLEAR_TEXT_LINE =
;  2790        M 2847  2          CH$FILL ( %C' ',                                ! Fill char
;  2791        M 2848  2                    .WCB [WCB_W_NO_COLS],                 ! # of bytes
;  2792        M 2849  2                    .WCB [WCB_A_TEXT_BUF] + .WLP);        ! Destination
;  2793          2850  2      %,
```

```
;  2794          2851  2
;  2795        M 2852  2      $CLEAR_ATTR_LINE =
;  2796        M 2853  2          CH$FILL ( 0,                                    ! Fill char
;  2797        M 2854  2                    .WCB [WCB_W_NO_COLS],                 ! # of bytes
;  2798        M 2855  2                    .WCB [WCB_A_ATTR_BUF] + .WLP);        ! Destination
;  2799          2856  2      %,
;  2800          2857  2
;  2801        M 2858  2      $MAP_ALL_NORMAL =
;  2802        M 2859  2          BEGIN
;  2803        M 2860  2          $MAP_LINE_NORMAL;
;  2804        M 2861  2          $MAP_ATTR_NORMAL;
;  2805        M 2862  2          $MAP_ALT_CHAR_NORMAL;
;  2806        M 2863  2          END;
;  2807          2864  2      %,
;  2808          2865  2
;  2809        M 2866  2      $CLEAR_ALT_CHAR_LINE =
;  2810        M 2867  2          IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
;  2811        M 2868  2          THEN
;  2812        M 2869  2          CH$FILL ( 0,
;  2813        M 2870  2                    .WCB [WCB_W_NO_COLS],
;  2814        M 2871  2                    .WCB [WCB_A_CHAR_SET_BUF] + .WLP);
;  2815          2872  2      %,
;  2816          2873  2
;  2817        M 2874  2      $MAP_ALL_SPECIAL =
;  2818        M 2875  2          BEGIN
;  2819        M 2876  2          $MAP_LINE_SPECIAL;
;  2820        M 2877  2          $MAP_ATTR_SPECIAL;
;  2821        M 2878  2          $MAP_ALT_CHAR_SPECIAL;
;  2822        M 2879  2          END;
;  2823          2880  2      %,
;  2824          2881  2
;  2825        M 2882  2      $CLEAR_ALL =
;  2826        M 2883  2          BEGIN
;  2827        M 2884  2          $CLEAR_TEXT_LINE;
;  2828        M 2885  2          $CLEAR_ATTR_LINE;
;  2829        M 2886  2          $CLEAR_ALT_CHAR_LINE;
;  2830        M 2887  2          RE_BORDER = 1; ! Set flag to rebuild border
;  2831        M 2888  2          END;
;  2832          2889  2      %;
;  2833          2890  2
;  2834          2891  2      DCB  = .PP [PP_A_DCB_ADDR];
;  2835          2892  2      PBCB = .PP [PP_A_PBCB_ADDR];
;  2836          2893  2      WCB  = .PBCB [PBCB_A_WCB];
;  2837          2894  2      IN_LINE_CHAR = .DCB [DCB_A_LINE_CHAR];
;  2838          2895  2      OUT_LINE_CHAR= .WCB [WCB_A_LINE_CHAR];
;  2839          2896  2      WCB_COLS = .WCB [WCB_W_NO_COLS];
;  2840          2897  2      DCB_COLS = .DCB [DCB_W_NO_COLS];
;  2841          2898  2
;  2842          2899  2  !+
;  2843          2900  2  ! Before diverging on two copying paths, check to see if we are going
;  2844          2901  2  ! to get involved with alternate character set buffers.  If one exists
;  2845          2902  2  ! in the DCB but does not yet exist in the WCB, we have to allocate
;  2846          2903  2  ! one for the WCB and initialize it.
;  2847          2904  2  !-
;  2848          2905  2      IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0  AND
;  2849          2906  2         .WCB [WCB_A_CHAR_SET_BUF] EQL 0
;  2850          2907  2      THEN
```

```
: 2851          2908  3          BEGIN   ! Alloc. and init. window alternate char. set buffer
: 2852          2909  3          LOCAL
: 2853          2910  3              STATUS;     ! Status of LIB$GET_VM call
: 2854          2911  3
: 2855          2912  4          IF NOT (STATUS = LIB$GET_VM ( WCB [WCB_L_BUFSIZE],
: 2856          2913  4                                        WCB [WCB_A_CHAR_SET_BUF]))
: 2857          2914  3          THEN
: 2858          2915  3              RETURN (.STATUS);
: 2859          2916  3
: 2860          2917  3          CH$FILL ( 0, .WCB [WCB_L_BUFSIZE], .WCB [WCB_A_CHAR_SET_BUF]);
: 2861          2918  2          END;    ! Alloc. and init. window alternate char. set buffer
: 2862          2919  2
: 2863          2920  2      !+
: 2864          2921  2      ! At this point we need to decide whether we are dealing with either
: 2865          2922  2      ! a virtual display or a window control block that includes either
: 2866          2923  2      ! Double-Wide (DW) and/or Double-Wide/Double-High (DWDH) lines.
: 2867          2924  2      ! We can tell whether either is non-standard by looking at the zeroth
: 2868          2925  2      ! byte of its line characteristics vector.
: 2869          2926  2      ! If we are dealing with either non-standard, we need to deal with them
: 2870          2927  2      ! on a line by line basis.
: 2871          2928  2      ! For each line to be mapped we implement the logic summarized in the
: 2872          2929  2      ! following table:
: 2873          2930  2      !
: 2874          2931  2      !     DCB Line            WCB Line         Mapping
: 2875          2932  2      !     Char. Vect          Char. Vector     Action
: 2876          2933  2      !     ----------          ------------     ------
: 2877          2934  2      !
: 2878          2935  2      !     Not DW or DWDH      Not DW or DWDH   Map normally
: 2879          2936  2      !     (i.e., normal)      (i.e., Normal)
: 2880          2937  2      !
: 2881          2938  2      !
: 2882          2939  2      !     Not DW or DWDH      DW or DWDH       If DW supported
: 2883          2940  2      !     (i.e., normal)                       then
: 2884          2941  2      !                                              Blank entire WCB text line
: 2885          2942  2      !                                              map normally
: 2886          2943  2      !                                            else
: 2887          2944  2      !                                              map normally
: 2888          2945  2      !
: 2889          2946  2      !
: 2890          2947  2      !     DW or DWDH          Not DW or DWDH   If DW supported
: 2891          2948  2      !                         (i.e., normal)   then
: 2892          2949  2      !                                              Blank entire WCB text line
: 2893          2950  2      !                                              use special mapping
: 2894          2951  2      !                                            else
: 2895          2952  2      !                                              map normally
: 2896          2953  2      !
: 2897          2954  2      !
: 2898          2955  2      !     DW or DWDH          DW or DWDH       If DW supported
: 2899          2956  2      !                                            then
: 2900          2957  2      !                                              Use special mapping
: 2901          2958  2      !                                            else
: 2902          2959  2      !                                              Map normally
: 2903          2960  2      !
: 2904          2961  2      !
: 2905          2962  2      !-
: 2906          2963  2          FROM_INDEX = .PP [PP_W_FROM_INDEX];
: 2907          2964  2          TO_INDEX   = .PP [PP_W_TO_INDEX];
```

```
;  2908          2965  2     IF .IN_LINE_CHAR  [0] EQL 0 AND     ! DCB normal
;  2909          2966  2        .OUT_LINE_CHAR [0] EQL 0         ! WCB normal
;  2910          2967  2     THEN
;  2911          2968  3         BEGIN   ! Both are normal -- no special action needed
;  2912          2969  3         !+
;  2913          2970  3         ! Check to see if we can do it with a single CH$MOVE or whether
;  2914          2971  3         ! we must do it a row at a time.
;  2915          2972  3         !-
;  2916          2973  3         IF .PP [PP_V_CONTIG]
;  2917          2974  3         THEN
;  2918          2975  4             BEGIN       ! Can be done in single move
;  2919          2976  4
;  2920          2977  4             ! Move text
;  2921          2978  4             CH$MOVE ( .PP [PP_L_MOVE_SIZE],             ! length
;  2922          2979  4                   .DCB [DCB_A_TEXT_BUF] + .FROM_INDEX, ! source
;  2923          2980  4                   .WCB [WCB_A_TEXT_BUF] + .TO_INDEX);  ! dest.
;  2924          2981  4
;  2925          2982  4             ! Move attributes
;  2926          2983  4             CH$MOVE ( .PP [PP_L_MOVE_SIZE],             ! length
;  2927          2984  4                   .DCB [DCB_A_ATTR_BUF] + .FROM_INDEX,  ! source
;  2928          2985  4                   .WCB [WCB_A_ATTR_BUF] + .TO_INDEX);   ! dest.
;  2929          2986  4
;  2930          2987  4             ! Move alternate character set buffer pieces, if necessary
;  2931          2988  4             IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
;  2932          2989  4             THEN
;  2933          2990  5                 BEGIN   ! Map alternate character set
;  2934          2991  5                 CH$MOVE ( .PP [PP_L_MOVE_SIZE],             ! Length
;  2935          2992  5                       .DCB [DCB_A_CHAR_SET_BUF] + .FROM_INDEX, ! Source
;  2936          2993  5                       .WCB [WCB_A_CHAR_SET_BUF] + .TO_INDEX);  ! Dest.
;  2937          2994  4                 END;    ! Map alternate character set
;  2938          2995  4             END         ! Can be done in single move
;  2939          2996  4
;  2940          2997  3         ELSE
;  2941          2998  3
;  2942          2999  4             BEGIN       ! Must be done row at a time
;  2943          3000  4
;  2944          3001  4             INCR R FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
;  2945          3002  4             DO
;  2946          3003  5                 BEGIN   ! For all rows in this display
;  2947          3004  5                 $MAP_LINE_NORMAL;
;  2948          3005  5                 $MAP_ATTR_NORMAL;
;  2949          3006  5
;  2950          3007  5                 FROM_INDEX = .FROM_INDEX + .DCB_COLS;
;  2951          3008  5                 TO_INDEX   = .TO_INDEX   + .WCB_COLS;
;  2952          3009  4                 END;    ! For all rows in this display
;  2953          3010  4
;  2954          3011  4             !+
;  2955          3012  4             ! Move alternate character set buffer pieces, if necessary
;  2956          3013  4             !
;  2957          3014  4             IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
;  2958          3015  4             THEN
;  2959          3016  5                 BEGIN   ! Map alternate character set buffer
;  2960          3017  5                 FROM_INDEX = .PP [PP_W_FROM_INDEX];
;  2961          3018  5                 TO_INDEX   = .PP [PP_W_TO_INDEX];
;  2962          3019  5
;  2963          3020  5                 INCR R FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
;  2964          3021  5                 DO
```

```
;  2965          3022  6                    BEGIN
;  2966          3023  6                    $MAP_ALT_CHAR_NORMAL;
;  2967          3024  6
;  2968          3025  6                    FROM_INDEX = .FROM_INDEX + .DCB_COLS;
;  2969          3026  6                    TO_INDEX   = .TO_INDEX   + .WCB_COLS;
;  2970          3027  5                    END;
;  2971          3028  5
;  2972          3029  4                END;     ! Map alternate character set buffer
;  2973          3030  4
;  2974          3031  3            END;         ! Must be done row at a time
;  2975          3032  3        END      ! Both are normal -- no special action needed
;  2976          3033  3
;  2977          3034  2    ELSE
;  2978          3035  3        BEGIN    ! One or the otner contains special characteristics
;  2979          3036  3        LOCAL
;  2980          3037  3            DLN,         ! Virtual display line number
;  2981          3038  3            WLN,         ! WCB line number
;  2982          3039  3            WLP;         ! Byte offset corresponding to WLN
;  2983          3040  3
;  2984          3041  3        !+
;  2985          3042  3        ! Must deal with this mapping on a row by row basis and on
;  2986          3043  3        ! each row inspect the line characteristics vector entry of
;  2987          3044  3        ! both the DCB line characteristics vector and the WCB line
;  2988          3045  3        ! characteristics vector in order to determine how to map this
;  2989          3046  3        ! particular line.
;  2990          3047  3        !
;  2991          3048  3        ! Initialize the special index we will be using.
;  2992          3049  3        !-
;  2993          3050  3        SPEC_TO_INDEX = (.PP [PP_W_FIRST_WCB_ROW] -1) *
;  2994          3051  3                        .WCB [WCB_W_NO_COLS] +
;  2995          3052  3                        ((.PP [PP_W_FIRST_WCB_COL] +2) / 2) -1;
;  2996          3053  3
;  2997          3054  3        DLN = (.FROM_INDEX / .DCB_COLS) +1;
;  2998          3055  3        WLN = .PP [PP_W_FIRST_WCB_ROW];
;  2999          3056  3        WLP = (.WLN - 1) * .WCB_COLS;
;  3000          3057  3
;  3001          3058  3        INCR R FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
;  3002          3059  3        DO
;  3003          3060  4            BEGIN        ! For each row
;  3004          3061  4            IF .IN_LINE_CHAR  [.DLN] EQL 0 AND  ! DCB normal
;  3005          3062  4               .OUT_LINE_CHAR [.WLN] EQL 0      ! WCB normal
;  3006          3063  4            THEN
;  3007          3064  5                BEGIN    ! Normal line
;  3008          3065  5                $MAP_ALL_NORMAL;
;  3009          3066  5                END      ! Normal line
;  3010          3067  4            ELSE
;  3011          3068  5                BEGIN    ! Special line
;  3012          3069  5                IF .IN_LINE_CHAR  [.DLN] EQL 0 AND      ! DCB normal
;  3013          3070  5                   .OUT_LINE_CHAR [.WLN] NEQ 0          ! WCB special
;  3014          3071  5                THEN
;  3015          3072  6                    BEGIN        ! WCB has special, DCB normal
;  3016          3073  6                    IF .PBCB [PBCB_V_WIDE]       ! If DW supported
;  3017          3074  6                    THEN
;  3018          3075  7                        BEGIN ! Special mapping
;  3019          3076  7                        $CLEAR_ALL;
;  3020          3077  7                        $MAP_ALL_NORMAL ;
;  3021          3078  7                        END   ! Special mapping
```

```
;  3022          3079  6                    ELSE
;  3023          3080  7                        BEGIN ! Normal mapping
;  3024          3081  7                        $MAP_ALL_NORMAL;
;  3025          3082  6                        END;  ! Normal mapping
;  3026          3083  6                    END           ! WCB has special, DCB normal
;  3027          3084  5                ELSE
;  3028          3085  6                    BEGIN
;  3029          3086  6                    IF .IN_LINE_CHAR  [.DLN] NEQ 0 AND  ! DCB special
;  3030          3087  6                       .OUT_LINE_CHAR [.WLN] EQL 0      ! WCB normal
;  3031          3088  6                    THEN
;  3032          3089  7                        BEGIN   ! DCB special, WCB normal
;  3033          3090  7                        IF .PBCB [PBCB_V_WIDE]  ! If DW supported
;  3034          3091  7                        THEN
;  3035          3092  8                            BEGIN       ! Map special
;  3036          3093  8                            $CLEAR_ALL;
;  3037          3094  8                            $MAP_ALL_SPECIAL;
;  3038          3095  8                            END         ! Map special
;  3039          3096  7                        ELSE
;  3040          3097  8                            BEGIN       ! Map normal
;  3041          3098  8                            $MAP_ALL_NORMAL;
;  3042          3099  7                            END;        ! Map normal
;  3043          3100  7                        END     ! DCB special, WCB normal
;  3044          3101  6                    ELSE
;  3045          3102  7                        BEGIN   ! Both have special
;  3046          3103  7                        IF .PBCB [PBCB_V_WIDE]  ! If DW supported
;  3047          3104  7                        THEN
;  3048          3105  8                            BEGIN       ! Map special
;  3049          3106  8                            $MAP_ALL_SPECIAL;
;  3050          3107  8                            END         ! Map special
;  3051          3108  7                        ELSE
;  3052          3109  8                            BEGIN       ! Map normal
;  3053          3110  8                            $MAP_ALL_NORMAL;
;  3054          3111  7                            END;        ! Map normal
;  3055          3112  6                        END;    ! Both have special
;  3056          3113  5                    END;
;  3057          3114  4                END;    ! Special line
;  3058          3115  4            !+
;  3059          3116  4            ! Advance indices for next row
;  3060          3117  4            !-
;  3061          3118  4            FROM_INDEX    = .FROM_INDEX    + .DCB_COLS;
;  3062          3119  4            TO_INDEX      = .TO_INDEX      + .WCB_COLS;
;  3063          3120  4            SPEC_TO_INDEX = .SPEC_TO_INDEX + .WCB_COLS;
;  3064          3121  4            DLN = .DLN + 1;
;  3065          3122  4            WLN = .WLN + 1;
;  3066          3123  4            WLP = .WLP + .WCB_COLS;
;  3067          3124  3            END;        ! For each row
;  3068          3125  3
;  3069          3126  2        END;    ! One or the other contains special characteristics
;  3070          3127  2
;  3071          3128  2 !+
;  3072          3129  2 ! If this device supports DW or DWDH then we must update WCB's line
;  3073          3130  2 ! characteristics vector based on the rows we just modified.
;  3074          3131  2 ! All further actions are necessary only if DW or DWDH is supported by
;  3075          3132  2 ! this device.
;  3076          3133  2 !-
;  3077          3134  2     IF .PBCB [PBCB_V_WIDE]
;  3078          3135  2     THEN
```

```
; 3079          3136  3          BEGIN   ! Special supported
; 3080          3137  3          IF .PP [PP_W_ROWS_TO_MOVE] NEQ 0        ! Something got mapped
; 3081          3138  3          THEN
; 3082          3139  4              BEGIN
; 3083          3140  4              IF .PP [PP_W_ROW] GTR 0
; 3084          3141  4              THEN
; 3085          3142  4                  ! Start movement from 1st display row
; 3086          3143  4                  CH$MOVE ( .PP [PP_W_ROWS_TO_MOVE],
; 3087          3144  4                            IN_LINE_CHAR [1],
; 3088          3145  4                            OUT_LINE_CHAR [ .PP [PP_W_ROW]])
; 3089          3146  4              ELSE
; 3090          3147  4                  ! Start movement from nth display row
; 3091          3148  4                  CH$MOVE ( .PP [PP_W_ROWS_TO_MOVE],
; 3092          3149  4                            IN_LINE_CHAR [.DCB [DCB_W_NO_ROWS] -
; 3093          3150  4                                          .PP [PP_W_ROWS_TO_MOVE] +1],
; 3094          3151  4                            OUT_LINE_CHAR [1]);
; 3095          3152  4
; 3096          3153  3              END;
; 3097          3154  3
; 3098          3155  3          !+
; 3099          3156  3          ! If we are dealing with a bordered display and we have
; 3100          3157  3          ! made the transition from a normal line to a DHDW line, or
; 3101          3158  3          ! vice versa, it will be necessary to remap the border elements.
; 3102          3159  3          !-
; 3103          3160  3          IF .DCB [DCB_V_BORDERED]  AND
; 3104          3161  3             .RE_BORDER EQL 1
; 3105          3162  3          THEN
; 3106          3163  4              BEGIN
; 3107          3164  4              LOCAL
; 3108          3165  4                  STATUS ; ! Status of subroutine call
; 3109          3166  5              IF NOT (STATUS = SMG$$DRAW_BORDER ( .DCB, .PP, .WCB))
; 3110          3167  4              THEN
; 3111          3168  4                  RETURN (.STATUS);
; 3112          3169  3              END;
; 3113          3170  3
; 3114          3171  3          !+
; 3115          3172  3          ! Check through current WCB line characteristics vector.  If
; 3116          3173  3          ! any of the elements is non-zero, set OUT_LINE_CHAR [0] to the
; 3117          3174  3          ! 1st encountered non-zero value.
; 3118          3175  3          !-
; 3119          3176  3          IF .PP [PP_W_ROWS_TO_MOVE] GTR 0
; 3120          3177  3          THEN
; 3121          3178  4              BEGIN       ! Some mapping took place
; 3122          3179  4              LOCAL
; 3123          3180  4                  BYTES_REMAINING,                ! Output from SKPC
; 3124          3181  4                  FOUND_BYTE : REF VECTOR [,BYTE]; ! Output from SKPC
; 3125          3182  4
; 3126          3183  4              OUT_LINE_CHAR [0] = 0;      ! Assume none will be found
; 3127          3184  4              SKPC (%REF(0), WCB [WCB_W_NO_ROWS], OUT_LINE_CHAR [0];
; 3128          3185  4                    BYTES_REMAINING, FOUND_BYTE);
; 3129          3186  4
; 3130          3187  4              IF .BYTES_REMAINING NEQ 0   ! If non-zero found
; 3131          3188  4              THEN
; 3132          3189  4                  OUT_LINE_CHAR [0] = .FOUND_BYTE [0];
; 3133          3190  3              END;        ! Some mapping took place
; 3134          3191  2          END;    ! Special supported
; 3135          3192  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$:     ADDL3   #42, PP, -(SP)                    ; 2973
                                  3C          9E        01  E1 0008D  BBC     #1, @(SP)+, 5$                                  ;
                                                  28    BE  D0 00091  MOVL    @40(SP), R8                                     ; 2980
                                                  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  3E 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$:      CLRL    R0                                              ; 3061
                                     51         10    AE  D0 0019B          MOVL    IN_LINE_CHAR, R1                                ;
                                                   69'1 95 0019F          TSTB    (D[N)[R1]                                       ;
                                                     0E 12 001A2          BNEQ    16$                                             ;
                                                     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  3C 0037F 28$:   MOVZWL  2(DCB), R0              ; 3150
                                50              50          56  C2 00383          SUBL2   R6, R0                  ;
                                59          0C  AE              01  C1 00386          ADDL3   #1, OUT_LINE_CHAR, R9   ; 3151
                                                5A          10  AE  D0 0038B          MOVL    IN_LINE_CHAR, R10       ;
                                69          01  A04A            56  28 0038F          MOVC3   R6, 1(R0)[R10], (R9)    ;
                                                15          2F  A7  E9 00395 29$:   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>
```

```
;  3141          3197  1 %SBTTL 'SMGOCCLUDE - Check for occlusion'
;  3142          3198  1 GLOBAL ROUTINE SMG$$OCCLUDE (   LOWER : REF VECTOR [,WORD,SIGNED],
;  3143          3199  1                                 UPPER : REF VECTOR [,WORD,SIGNED],
;  3144          3200  1                                 REWRITE : REF VECTOR [,WORD,SIGNED]
;  3145          3201  1                               ) =
;  3146          3202  1 !++
;  3147          3203  1 ! FUNCTIONAL DESCRIPTION:
;  3148          3204  1 !
;  3149          3205  1 !       This routine inspects the relative positioning of two areas --
;  3150          3206  1 !       a LOWER area and an UPPER area.  It determines whether the
;  3151          3207  1 !       UPPER area occludes any portion of the LOWER area.  If occlusion
;  3152          3208  1 !       is found, a rectangular area which is the occluded part is
;  3153          3209  1 !       isolated and its coordinates are returned as the area which
;  3154          3210  1 !       must be rewritten.
;  3155          3211  1 !
;  3156          3212  1 ! CALLING SEQUENCE:
;  3157          3213  1 !
;  3158          3214  1 !       CODE.wl.v = SMG$$OCCLUDE (
;  3159          3215  1 !                                  LOWER.rw.r,
;  3160          3216  1 !                                  UPPER.rw.r,
;  3161          3217  1 !                                  REWRITE.waw.r
;  3162          3218  1 !                                 )
;  3163          3219  1 !
;  3164          3220  1 ! FORMAL PARAMETERS:
;  3165          3221  1 !
;  3166          3222  1 !       LOWER.raw.r     Description of a rectangular area represent
;  3167          3223  1 !                       a "lower" area.!
;  3168          3224  1 !       UPPER.raw.r     Description of a rectangular area representing
;  3169          3225  1 !                       an "upper" area.
;  3170          3226  1 !
;  3171          3227  1 !       REWRITE.waw.r   Description of a rectangular area which
;  3172          3228  1 !                       represents the area of the lower which is
;  3173          3229  1 !                       occluded by the upper ( if any).
;  3174          3230  1 !
;  3175          3231  1 !
;  3176          3232  1 ! IMPLICIT INPUTS:
;  3177          3233  1 !
;  3178          3234  1 !       NONE
;  3179          3235  1 !
;  3180          3236  1 ! IMPLICIT OUTPUTS:
;  3181          3237  1 !
;  3182          3238  1 !       NONE
;  3183          3239  1 !
;  3184          3240  1 ! ROUTINE VALUE
;  3185          3241  1 !
;  3186          3242  1 !
;  3187          3243  1 ! SIDE EFFECTS:
;  3188          3244  1 !
;  3189          3245  1 !       NONE
;  3190          3246  1 !--
;  3191          3247  1
;  3192          3248  2     BEGIN
;  3193          3249  2     LITERAL
;  3194          3250  2          ROW_START = 0,
;  3195          3251  2          NUM_ROW   = 1,
;  3196          3252  2          COL_START = 2,
;  3197          3253  2          NUM_COL   = 3;
```

```
;3198          3254  2
;3199          3255  2      LOCAL
;3200          3256  2          ANS,
;3201          3257  2          STATUS,
;3202          3258  2          LOWER_BOTTOM_ROW: SIGNED,
;3203          3259  2          LOWER_RIGHT_COL: SIGNED,
;3204          3260  2          UPPER_BOTTOM_ROW: SIGNED,
;3205          3261  2          UPPER_RIGHT_COL: SIGNED;
;3206          3262  2
;3207          3263  2  MACRO
;3208          3264  2          LOWER_ROW_START         = LOWER [ROW_START]%,
;3209          3265  2          LOWER_NO_ROW            = LOWER [NUM_ROW]%,
;3210          3266  2          LOWER_COL_START         = LOWER [COL_START]%,
;3211          3267  2          LOWER_NO_COL            = LOWER [NUM_COL]%,
;3212          3268  2
;3213          3269  2          UPPER_ROW_START         = UPPER [ROW_START]%,
;3214          3270  2          UPPER_NO_ROW            = UPPER [NUM_ROW]%,
;3215          3271  2          UPPER_COL_START         = UPPER [COL_START]%,
;3216          3272  2          UPPER_NO_COL            = UPPER [NUM_COL]%,
;3217          3273  2
;3218          3274  2          REWRITE_ROW_START       = REWRITE [ROW_START]%,
;3219          3275  2          REWRITE_NO_ROW          = REWRITE [NUM_ROW]%,
;3220          3276  2          REWRITE_COL_START       = REWRITE [COL_START]%,
;3221          3277  2          REWRITE_NO_COL          = REWRITE [NUM_COL]%;
;3222          3278  2
;3223          3279  2  !+
;3224          3280  2  ! Calculate the other bounds of the specified rectangles.
;3225          3281  2  !-
;3226          3282  2      LOWER_BOTTOM_ROW = .LOWER_ROW_START + .LOWER_NO_ROW - 1;
;3227          3283  2      LOWER_RIGHT_COL =  .LOWER_COL_START + .LOWER_NO_COL - 1;
;3228          3284  2      UPPER_BOTTOM_ROW = .UPPER_ROW_START + .UPPER_NO_ROW - 1;
;3229          3285  2      UPPER_RIGHT_COL  = .UPPER_COL_START + .UPPER_NO_COL - 1;
;3230          3286  2
;3231          3287  2  !+
;3232          3288  2  ! Assume no occlusion occurs until it proves otherwise.
;3233          3289  2  !-
;3234          3290  2      STATUS = 0;
;3235          3291  2
;3236          3292  2  !+
;3237          3293  2  ! Check for relative position of upper left-hand corner of UPPER
;3238          3294  2  ! rectangle with respect to LOWER rectangle.
;3239          3295  2  !-
;3240          3296  2      ANS = SMG$$POINT_IN_RECT_R3 (UPPER_COL_START,       ! X1
;3241          3297  2                                   UPPER_ROW_START,       ! Y1
;3242          3298  2                                   .LOWER);
;3243          3299  2
;3244          3300  2      CASE .ANS FROM 0 TO 10 OF
;3245          3301  2      SET
;3246          3302  3          [0]:BEGIN
;3247          3303  3              REWRITE_ROW_START = .UPPER_ROW_START;
;3248          3304  3              REWRITE_COL_START = .UPPER_COL_START;
;3249          3305  3              !+
;3250          3306  3              ! Check relative position of lower right-hand corner of
;3251          3307  3              ! UPPER rectangle with respect to LOWER rectangle.
;3252          3308  3              !-
;3253          3309  3              ANS = SMG$$POINT_IN_RECT_R3 (UPPER_RIGHT_COL,       ! X1
;3254          3310  3                                          UPPER_BOTTOM_ROW,       ! Y1
```

```
;  3255          3311  3                                                .LOWER);
;  3256          3312  3
;  3257          3313  3                CASE .ANS FROM 0 TO 6 OF
;  3258          3314  3                SET
;  3259          3315  4                    [0]:BEGIN
;  3260          3316  4                        REWRITE_NO_ROW = .UPPER_NO_ROW;
;  3261          3317  4                        REWRITE_NO_COL = .UPPER_NO_COL;
;  3262          3318  3                        END;
;  3263          3319  3
;  3264          3320  4                    [2]:BEGIN
;  3265          3321  4                        REWRITE_NO_ROW = .UPPER_NO_ROW;
;  3266          3322  4                        REWRITE_NO_COL = .LOWER_RIGHT_COL -.UPPER_COL_START+1;
;  3267          3323  3                        END;
;  3268          3324  3
;  3269          3325  4                    [4]:BEGIN
;  3270          3326  4                        REWRITE_NO_ROW = .LOWER_BOTTOM_ROW -.UPPER_ROW_START+1;
;  3271          3327  4                        REWRITE_NO_COL = .UPPER_NO_COL;
;  3272          3328  3                        END;
;  3273          3329  3
;  3274          3330  4                    [6]:BEGIN
;  3275          3331  4                        REWRITE_NO_ROW = .LOWER_BOTTOM_ROW -.UPPER_ROW_START+1;
;  3276          3332  4                        REWRITE_NO_COL = .LOWER_RIGHT_COL - .UPPER_COL_START+1;
;  3277          3333  3                        END;
;  3278          3334  3                    [1,3,5]: RETURN (SMG$_FATERRLIB);
;  3279          3335  3
;  3280          3336  3                TES;
;  3281          3337  2                END;
;  3282          3338  2
;  3283          3339  3            [1]:BEGIN
;  3284          3340  3                REWRITE_ROW_START = .UPPER_ROW_START;
;  3285          3341  3                REWRITE_COL_START = .LOWER_COL_START;
;  3286          3342  3                !+
;  3287          3343  3                ! Check relative position of lower right-hand corner of
;  3288          3344  3                ! UPPER rectangle with respect to lower rectangle.
;  3289          3345  3                !-
;  3290          3346  3                ANS = SMG$$POINT_IN_RECT_R3 (UPPER_RIGHT_COL,       ! X1
;  3291          3347  3                                            UPPER_BOTTOM_ROW,       ! Y1
;  3292          3348  3                                            .LOWER);
;  3293          3349  3
;  3294          3350  3                CASE .ANS FROM 0 TO 6 OF
;  3295          3351  3                SET
;  3296          3352  4                    [0]:BEGIN
;  3297          3353  4                        REWRITE_NO_ROW = .UPPER_NO_ROW;
;  3298          3354  4                        REWRITE_NO_COL = .UPPER_RIGHT_COL - .LOWER_COL_START +1;
;  3299          3355  3                        END;
;  3300          3356  3
;  3301          3357  3                    [1]:    RETURN (.STATUS);
;  3302          3358  3
;  3303          3359  4                    [2]:BEGIN
;  3304          3360  4                        REWRITE_NO_ROW = .UPPER_NO_ROW;
;  3305          3361  4                        REWRITE_NO_COL = .LOWER_NO_COL;
;  3306          3362  3                        END;
;  3307          3363  3
;  3308          3364  4                    [4]:BEGIN
;  3309          3365  4                        REWRITE_NO_ROW = .LOWER_BOTTOM_ROW -.UPPER_ROW_START+1;
;  3310          3366  4                        REWRITE_NO_COL = .UPPER_RIGHT_COL -.LOWER_COL_START+1;
;  3311          3367  3                        END;
```

```
;  3312          3368  3
;  3313          3369  3                    [5]:    RETURN (.STATUS);
;  3314          3370  3
;  3315          3371  4                    [6]:BEGIN
;  3316          3372  4                        REWRITE_NO_ROW = .LOWER_BOTTOM_ROW -.UPPER_ROW_START+1;
;  3317          3373  4                        REWRITE_NO_COL = .LOWER_NO_COL;
;  3318          3374  3                        END;
;  3319          3375  3                    [3]: RETURN (SMG$_FATERRLIB);
;  3320          3376  3
;  3321          3377  3                TES;
;  3322          3378  2                END;
;  3323          3379  2
;  3324          3380  2            [2]:    RETURN (.STATUS);
;  3325          3381  2
;  3326          3382  2            [4]:    RETURN (.STATUS);
;  3327          3383  2
;  3328          3384  2            [5]:    RETURN (.STATUS);
;  3329          3385  2
;  3330          3386  2            [6]:    RETURN (.STATUS);
;  3331          3387  2
;  3332          3388  3            [8]:BEGIN
;  3333          3389  3                REWRITE_ROW_START = .LOWER_ROW_START;
;  3334          3390  3                REWRITE_COL_START = .UPPER_COL_START;
;  3335          3391  3                !+
;  3336          3392  3                ! Check relative position of lower right-hand corner of
;  3337          3393  3                ! UPPER rectangle with respect to LOWER rectangle.
;  3338          3394  3                !-
;  3339          3395  3                ANS = SMG$$POINT_IN_RECT_R3 (UPPER_RIGHT_COL,       ! X1
;  3340          3396  3                                            UPPER_BOTTOM_ROW,       ! Y1
;  3341          3397  3                                            .LOWER);
;  3342          3398  3
;  3343          3399  3                CASE .ANS FROM 0 TO 10 OF
;  3344          3400  3                SET
;  3345          3401  4                    [0]:BEGIN
;  3346          3402  4                        REWRITE_NO_ROW = .UPPER_BOTTOM_ROW -.LOWER_ROW_START+1;
;  3347          3403  4                        REWRITE_NO_COL = .UPPER_NO_COL;
;  3348          3404  3                        END;
;  3349          3405  3
;  3350          3406  4                    [2]:BEGIN
;  3351          3407  4                        REWRITE_NO_ROW = .UPPER_BOTTOM_ROW -.LOWER_ROW_START+1;
;  3352          3408  4                        REWRITE_NO_COL = .LOWER_RIGHT_COL - .UPPER_COL_START+1;
;  3353          3409  3                        END;
;  3354          3410  3
;  3355          3411  4                    [4]:BEGIN
;  3356          3412  4                        REWRITE_NO_ROW = .LOWER_NO_ROW;
;  3357          3413  4                        REWRITE_NO_COL = .UPPER_NO_COL;
;  3358          3414  3                        END;
;  3359          3415  3
;  3360          3416  4                    [6]:BEGIN
;  3361          3417  4                        REWRITE_NO_ROW = .LOWER_NO_ROW;
;  3362          3418  4                        REWRITE_NO_COL = .LOWER_RIGHT_COL -.UPPER_COL_START+1;
;  3363          3419  3                        END;
;  3364          3420  3                    [8]:    RETURN (.STATUS);
;  3365          3421  3
;  3366          3422  3                    [10]:   RETURN (.STATUS);
;  3367          3423  3
;  3368          3424  3                    [1,3,5,7,9]: RETURN (SMG$_FATERRLIB);
```

```
;  3369          3425  3              TES;
;  3370          3426  3              END;
;  3371          3427  2
;  3372          3428  2
;  3373          3429  3          [9]:BEGIN
;  3374          3430  3              REWRITE_ROW_START = .LOWER_ROW_START;
;  3375          3431  3              REWRITE_COL_START = .LOWER_COL_START;
;  3376          3432  3              !+
;  3377          3433  3              ! Check relative position of lower right-hand corner of
;  3378          3434  3              ! UPPER rectangle with respect to LOWER rectangle.
;  3379          3435  3              !-
;  3380          3436  3              ANS = SMG$$POINT_IN_RECT_R3 (UPPER_RIGHT_COL,       ! X1
;  3381          3437  3                                          UPPER_BOTTOM_ROW,       ! Y1
;  3382          3438  3                                          .LOWER);
;  3383          3439  3
;  3384          3440  3              CASE .ANS FROM 0 TO 10 OF
;  3385          3441  3              SET
;  3386          3442  4                  [0]:BEGIN
;  3387          3443  4                      REWRITE_NO_ROW = .UPPER_BOTTOM_ROW -.LOWER_ROW_START+1;
;  3388          3444  4                      REWRITE_NO_COL = .UPPER_RIGHT_COL -.LOWER_COL_START+1;
;  3389          3445  3                      END;
;  3390          3446  3
;  3391          3447  3                  [1]:    RETURN (.STATUS);
;  3392          3448  3
;  3393          3449  4                  [2]:BEGIN
;  3394          3450  4                      REWRITE_NO_ROW = .UPPER_BOTTOM_ROW -.LOWER_ROW_START+1;
;  3395          3451  4                      REWRITE_NO_COL = .LOWER_NO_COL;
;  3396          3452  3                      END;
;  3397          3453  3
;  3398          3454  4                  [4]:BEGIN
;  3399          3455  4                      REWRITE_NO_ROW = .LOWER_NO_ROW;
;  3400          3456  4                      REWRITE_NO_COL = .UPPER_RIGHT_COL -.LOWER_COL_START+1;
;  3401          3457  3                      END.
;  3402          3458  3
;  3403          3459  3                  [5]:    RETURN (.STATUS);
;  3404          3460  3
;  3405          3461  4                  [6]:BEGIN
;  3406          3462  4                      REWRITE_NO_ROW = .LOWER_NO_ROW;;
;  3407          3463  4                      REWRITE_NO_COL = .LOWER_NO_COL;
;  3408          3464  3                      END;
;  3409          3465  3
;  3410          3466  3                  [8]:    RETURN (.STATUS);
;  3411          3467  3
;  3412          3468  3                  [9]:    RETURN (.STATUS);
;  3413          3469  3
;  3414          3470  3                  [10]:   RETURN (.STATUS);
;  3415          3471  3
;  3416          3472  3                  [3,7]: RETURN (SMG$_FATERRLIB);
;  3417          3473  3
;  3418          3474  3              TES;
;  3419          3475  2              END;
;  3420          3476  2
;  3421          3477  2          [10]:   RETURN (.STATUS);
;  3422          3478  2
;  3423          3479  2          [3,7]: RETURN (SMG$_FATERRLIB);
;  3424          3480  2
;  3425          3481  2      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            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                   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                   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                    53   CF 00157           CASEL   ANS, #0, #10                                ; 3399
                0025            00C6            0016      0015B 19$:     .WORD   20$-19$,-                                   ;
                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                        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                    53   CF 001C6           CASEL   ANS, #0, #10                                ; 3440
                0025            0053            0016      001CA 30$      .WORD   31$-30$,-                                   ;
                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>
```

```
;  3437          3492  1 %SBTTL 'SMG$$POINT_IN_RECT_R3 - Point inside rectangle'
;  3438          3493  1 GLOBAL ROUTINE SMG$$POINT_IN_RECT_R3 (  ! Point in rectangle
;  3439          3494  1                     X1 : REF VECTOR [,WORD,SIGNED], ! X coordinate of point
;  3440          3495  1                     Y1 : REF VECTOR [,WORD,SIGNED], ! Y coordinate of point
;  3441          3496  1                     RECT : REF VECTOR [,WORD,SIGNED]
;  3442          3497  1                                     ) : POINT_IN_RECT_LINK  =
;  3443          3498  1 !++
;  3444          3499  1 ! FUNCTIONAL DESCRIPTION:
;  3445          3500  1 !
;  3446          3501  1 !       This routine returns a code indicating the relative placement
;  3447          3502  1 ! of a point with respect to a horizontally-oriented window (that is,
;  3448          3503  1 ! one of the sides of the rectangle is parallel to the x-axis.
;  3449          3504  1 ! (See diagram below).
;  3450          3505  1 !
;  3451          3506  1 !       This is the inner-most function of the Cohen-Sutherland clipping
;  3452          3507  1 ! algorithm.
;  3453          3508  1 !
;  3454          3509  1 !       These codes have two significant properties:
;  3455          3510  1 !
;  3456          3511  1 !       1.  If the CODE of two points are each zero, both points are
;  3457          3512  1 !           within the window and no clipping needs to be done.
;  3458          3513  1 !
;  3459          3514  1 !       2.  If the logical AND of the CODEs of two points is non-zero,
;  3460          3515  1 !           the points are both outside of the window and are so
;  3461          3516  1 !           positioned that the line joining them does not intersect
;  3462          3517  1 !           the window. (I.e., it is a line segment which need not
;  3463          3518  1 !           be even partially displayed.
;  3464          3519  1 !
;  3465          3520  1 !
;  3466          3521  1 !       1001        |       1000        |       1010
;  3467          3522  1 !                   |                   |
;  3468          3523  1 !   ----------------+-------------------+------------ RECT [DCB_W_ROW_START]
;  3469          3524  1 !                   |                   |           ^
;  3470          3525  1 !                   |      Window       |           :
;  3471          3526  1 !                   |                   |           : RECT [DCB_W_NO_ROWS]
;  3472          3527  1 !       0001        |       0000        |       0010 :
;  3473          3528  1 !                   |                   |           V
;  3474          3529  1 !   ----------------+-------------------+------------
;  3475          3530  1 !                   |                   |
;  3476          3531  1 !       0101        |       0100        |       0110
;  3477          3532  1 !                   |                   |
;  3478          3533  1 !                   |<-RECT [DCB_W_NO_COLS]>|
;  3479          3534  1 !                   |                   |
;  3480          3535  1 !       RECT [DCB_W_COL_START]
;  3481          3536  1 !
;  3482          3537  1 !
;  3483          3538  1 ! CALLING SEQUENCE:
;  3484          3539  1 !
;  3485          3540  1 !       CODE.wl.v = SMG$$POINT_IN_RECT_R3 (X1.rw.r, Y1.rw.r, RECT.rab.r)
;  3486          3541  1 !
;  3487          3542  1 ! FORMAL PARAMETERS:
;  3488          3543  1 !
;  3489          3544  1 !       X1.rw.r         X coordinate of point
;  3490          3545  1 !       Y1.rw.r         Y coordinate of point
;  3491          3546  1 !       RECT.rab.r      Discription of the rectangular area.
;  3492          3547  1 !
;  3493          3548  1 !
```

```
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
```

SMGDISOUT
LIS

SMGKEYUTI
LIS

SMGKEYPAD
LIS

SMGINPUT
LIS

SMGLIB
LIS