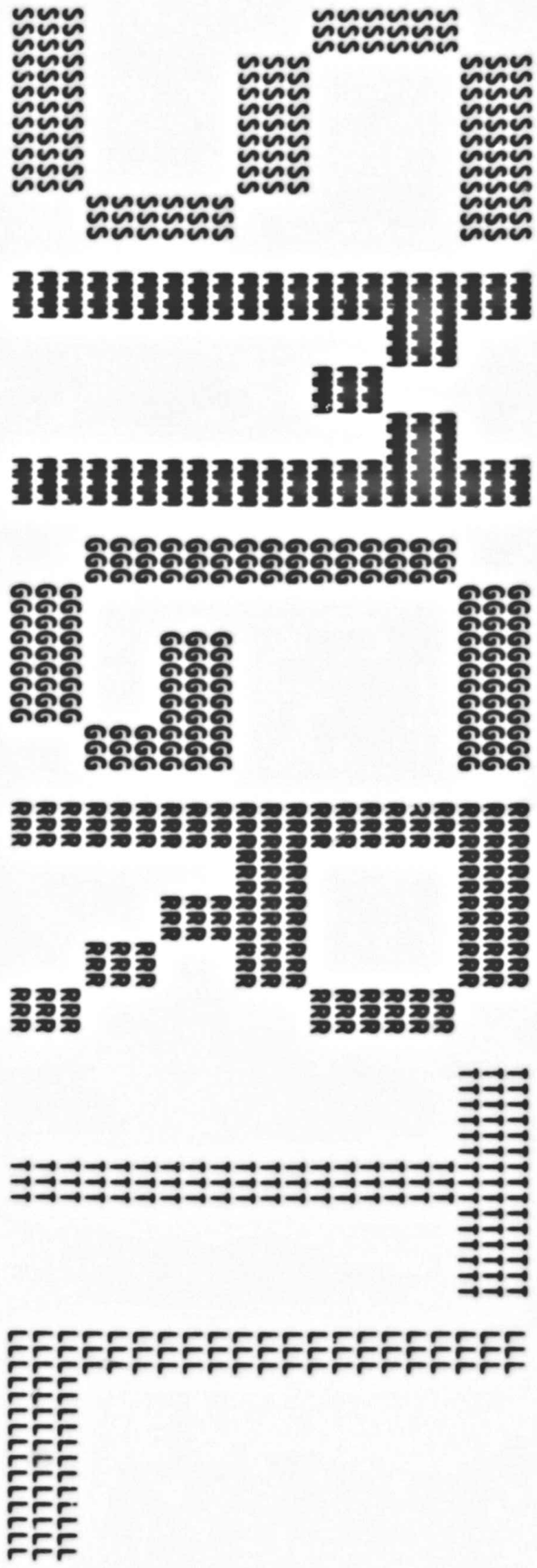

k 7

```
  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 MM  GG        DD      DD      II     SS          OO      OO  UU      UU      TT
SS          MM MM MM MM  GG        DD      DD      II     SS          OO      OO  UU      UU      TT
  SSSSSS    MM    MM MM  GG        DD      DD      II       SSSSSS    OO      OO  UU      UU      TT
  SSSSSS    MM    MM MM  GG        DD      DD      II       SSSSSS    OO      OO  UU      UU      TT
        SS  MM    MM MM  GG  GGGGGG DD      DD      II             SS  OO      OO  UU      UU      TT
        SS  MM    MM MM  GG  GGGGGG DD      DD      II             SS  OO      OO  UU      UU      TT
        SS  MM    MM MM  GG      GG DD      DD      II             SS  OO      OO  UU      UU      TT
        SS  MM    MM MM  GG      GG DD      DD      II             SS  OO      OO  UU      UU      TT
SSSSSSSS    MM    MM MM    GGGGGG   DDDDDDDD     IIIIII     SSSSSSSS    OOOOOO    UUUUUUUUUU      TT
SSSSSSSS    MM    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 (
;001 PLL1073     0003  0                 IDENT = '1-073' ! File: SMGDISOUT.B32 Edit: PLL1073
;    4-1         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
;     8          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$RING_BELL -- add display_id argument.
;  126          0126  1 !           RKR 20-MAY-1983.
;  127          0127  1 ! 1-031   Add logic to BEGIN_DISPLAY_UPDATE and END_DISPLAY_UPDATE to
;  128          0128  1 !           back up and restore DCBs.
;  129          0129  1 !           RKR 23-MAY-1983.
;  130          0130  1 ! 1-032   Fix RING_BELL.  TIMES by reference to STR$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_changed_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$CONTROL_MODE to allow notabs bit.  PLL 28-Jun-1984
;   242          0242  1 ! 1-072 Tighten up some consistency checks on the data structures.  It was
;   243          0243  1 !        possible to get in an endless loop while walking the pasting packet
;   244          0244  1 !        chain.  PLL 18-Jul-1984
;001 PLL1073     0245  1 ! 1-073 Another change to SMG$CONTROL_MODE to look at all unused bits.  PLL
;002 PLL1073     0246  1 !        14-Oct-1984
;   245          0247  1 !--
```

```
;  247     0248  1 %SBTTL 'Declarations'
;  248     0249  1 !
;  249     0250  1 ! SWITCHES:
;  250     0251  1 !
;  251     0252  1
;  252     0253  1 !
;  253     0254  1 ! LINKAGES:
;  254     0255  1 !
;  255     0256  1 LINKAGE
;  256     0257  1     POINT_IN_RECT_LINK = JSB ( REGISTER = 0, REGISTER = 1, REGISTER = 2) :
;  257     0258  1                          NOPRESERVE ( 3)
;  258     0259  1                          NOTUSED (4,5,6,7,8,9,10,11);
;  259     0260  1 !
;  260     0261  1 ! INCLUDE FILES
;  261     0262  1 !
;  262     0263  1
;  263     0264  1 REQUIRE 'RTLIN:SMGPROLOG.REQ';          ! defines psects, macros, tcb,
;  264     0342  1                                         !  wcb, & terminal symbols
;  265     0343  1 !
;  266     0344  1 ! TABLE OF CONTENTS:
;  267     0345  1 !
;  268     0346  1
;  269     0347  1 FORWARD ROUTINE
;  270     0348  1
;  271     0349  1 ! Public entry points:
;  272     0350  1
;  273     0351  1     SMG$BEGIN_DISPLAY_UPDATE,           ! Advance buffering level count for
;  274     0352  1                                         ! a virtual display
;  275     0353  1
;  276     0354  1     SMG$BEGIN_PASTEBOARD_UPDATE,        ! Advance buffering level count for
;  277     0355  1                                         ! a pasteboard
;  278     0356  1
;  279     0357  1     SMG$CONTROL_MODE,                   ! Control operational modes
;  280     0358  1
;  281     0359  1
;  282     0360  1     SMG$END_DISPLAY_UPDATE,             ! Decr. buffering level count
;  283     0361  1                                         ! and flush to screen if it
;  284     0362  1                                         ! reaches zero.
;  285     0363  1
;  286     0364  1     SMG$END_PASTEBOARD_UPDATE,          ! Decr. buffering level count
;  287     0365  1                                         ! for a pasteboard, and flush to
;  288     0366  1                                         ! screen if it reaches 0.
;  289     0367  1
;  290     0368  1     SMG$FIND_CURSOR_DISPLAY,            ! Find the display which
;  291     0369  1                                         ! contains the current physical
;  292     0370  1                                         ! cursor position (if any).
;  293     0371  1
;  294     0372  1     SMG$GET_CHAR_AT_PHYSICAL_CURSOR,    ! Return char at physical cursor
;  295     0373  1                                         ! location
;  296     0374  1
;  297     0375  1     SMG$INVALIDATE_DISPLAY,             ! Mark contents of display as unknown.
;  298     0376  1
;  299     0377  1     SMG$REPAINT_SCREEN,                 ! Repaint the entire screen the
;  300     0378  1                                         ! way our internal database says
;  301     0379  1                                         ! it should look.
;  302     0380  1
;  303     0381  1     SMG$RING_BELL,                      ! Ring bell
```

```
;  304     0382  1
;  305     0383  1 ! Private entry points:
;  306     0384  1
;  307     0385  1     SMG$$BEGIN_PASTEBOARD_UPDATE_R1 : SMG$$BEGIN_PBD_UPDATE$LNK,
;  308     0386  1                                       ! Inner BEGIN_PASTEBOARD_UPDATE
;  309     0387  1
;  310     0388  1     SMG$$CHECK_FOR_OUTPUT_DCB,        ! Check to see if a virtual
;  311     0389  1                                       ! display needs to be flushed to
;  312     0390  1                                       ! screen.
;  313     0391  1
;  314     0392  1     SMG$$CHECK_FOR_OUTPUT_PBCB,       ! Refresh everything on this
;  315     0393  1                                       ! pasteboard -- triggered by
;  316     0394  1                                       ! an unpaste.
;  317     0395  1
;  318     0396  1     SMG$$DRAW_BORDER,                 ! Move border characters into
;  319     0397  1                                       ! WCB text buffer.
;  320     0398  1
;  321     0399  1     SMG$$END_PASTEBOARD_UPDATE_R2 : SMG$$END_PBD_UPDATE$LNK,
;  322     0400  1                                       ! Inner END_PASTEBOARD_UPDATE
;  323     0401  1
;  324     0402  1     SMG$$FILL_WINDOW_BUFFER,          ! Fill the window buffer
;  325     0403  1                                       ! associated with a pasteboard
;  326     0404  1                                       ! with images of the virtual
;  327     0405  1                                       ! displays that map into it.
;  328     0406  1
;  329     0407  1     SMG$$MOVE_TEXT_TO_WINDOW_BUF,     ! move text from DCB buffers to
;  330     0408  1                                       ! WCB window buffers.
;  331     0409  1
;  332     0410  1     SMG$$OCCLUDE,                     ! Check whether two rectangular
;  333     0411  1                                       ! areas occlude each other.
;  334     0412  1
;  335     0413  1     SMG$$POINT_IN_RECT_R3 : POINT_IN_RECT_LINK; ! Returns codes
;  336     0414  1                                       ! indicating where a
;  337     0415  1                                       ! given point is with
;  338     0416  1                                       ! respect to a given
;  339     0417  1                                       ! rectangular area.
;  340     0418  1
;  341     0419  1
;  342     0420  1 !
;  343     0421  1 ! EXTERNAL REFERENCES
;  344     0422  1 !
;  345     0423  1
;  346     0424  1 EXTERNAL
;  347     0425  1     PBD_L_COUNT,          ! No. of pasteboards we currently have
;  348     0426  1
;  349     0427  1     PBD_A_PBCB : VECTOR [PBD_K_MAX_PB, LONG],
;  350     0428  1                           ! Table of addresses of PBCB's
;  351     0429  1
;  352     0430  1     PBD_V_PB_AVAIL : BITVECTOR [PBD_K_MAX_PB];
;  353     0431  1                           ! Bit vector of pasteboard id numbers in use.
;  354     0432  1
;  355     0433  1 EXTERNAL LITERAL
;  356     0434  1
;  357     0435  1     SMG$_BATSTIPRO,       ! Success; but batching is still in progress.
;  358     0436  1     SMG$_BATWASOFF,       ! Success; but batching was already off
;  359     0437  1     SMG$_BATWAS_ON,       ! Success; but batching was already on
;  360     0438  1     SMG$_FATERRLIB,       ! Fatal error in library procedure
```

```
;  361     0439  1      SMG$_INVARG,          ! Invalid argument
;  362     0440  1      SMG$_INVCOL,          ! Invalid column number
;  363     0441  1      SMG$_INVDIS_ID,       ! Invalid virtual display id
;  364     0442  1      SMG$_INVPAS_ID,       ! Invalid pasteboard id
;  365     0443  1      SMG$_INVROW;          ! Invalid row number
;  366     0444  1
;  367     0445  1  EXTERNAL ROUTINE
;  368     0446  1      LIB$ESTABLISH,                ! Used to establish a handler
;  369     0447  1      LIB$FREE_VM,                  ! Deallocate heap storage
;  370     0448  1      LIB$GET_VM,                   ! Allocate heap storage
;  371     0449  1      LIB$SFREE1_DD,                ! Free a dynamic string
;  372     0450  1      LIB$SIG_TO_RET,               ! Handler to turn signals into return
;  373     0451  1                                    ! statuses.
;  374     0452  1      STR$DUPL_CHAR,                ! Make a string of N copies of a byte.
;  375     0453  1      SMG$$DUPL_VIRTUAL_DISPLAY,    ! Make a copy of a DCB with a new
;  376     0454  1                                    ! display id.
;  377     0455  1      SMG$$FLUSH_BUFFER,            ! Flush remaining buffered output
;  378     0456  1      SMG$$INVALIDATE_DISPLAY,      ! Tell SMG that user has
;  379     0457  1                                    ! written into this display on his own.
;  380     0458  1      SMG$$MIN_UPD,                 ! Minimal update output routine
;  381     0459  1      SMG$$OUTPUT,                  ! Output a string to a pasteboard device
;  382     0460  1      SMG$$RECALC_PP_FIELDS,        ! Recalc. PP fields after batching level
;  383     0461  1                                    ! lifted on a DCB
;  384     0462  1      SMG$$UPDATE_PHYSICAL_CURSOR;  ! Update physical cursor position
;  385     0463  1
;  386     0464  1  !<BLF/PAGE>
```

```
;  388          0465  1 %SBTTL 'SMG$BEGIN_DISPLAY_UPDATE - Begin batch of updates to display'
;  389          0466  1 GLOBAL ROUTINE SMG$BEGIN_DISPLAY_UPDATE ( DISPLAY_ID ) =
;  390          0467  1 !++
;  391          0468  1 ! FUNCTIONAL DESCRIPTION:
;  392          0469  1 !
;  393          0470  1 !       Disable outputing this virtual display to the screen until the
;  394          0471  1 !       matching call to SMG$END_DISPLAY_UPDATE is encountered.
;  395          0472  1 !
;  396          0473  1 ! CALLING SEQUENCE:
;  397          0474  1 !
;  398          0475  1 !       ret_status.wlc.v = SMG$BEGIN_DISPLAY_UPDATE ( DISPLAY_ID.rl.r)
;  399          0476  1 !
;  400          0477  1 ! FORMAL PARAMETERS:
;  401          0478  1 !
;  402          0479  1 !       DISPLAY_ID.rl.r         Display id of virtual display.
;  403          0480  1 !
;  404          0481  1 ! IMPLICIT INPUTS:
;  405          0482  1 !
;  406          0483  1 !       NONE
;  407          0484  1 !
;  408          0485  1 ! IMPLICIT OUTPUTS:
;  409          0486  1 !
;  410          0487  1 !       NONE
;  411          0488  1 !
;  412          0489  1 ! COMPLETION STATUS:
;  413          0490  1 !
;  414          0491  1 !       SS$_NORMAL      Normal successful completion, batching has been
;  415          0492  1 !                       initiated
;  416          0493  1 !       SMG$_BATWAS_ON  Success, but note that batching was already on
;  417          0494  1 !       SMG$_INVDIS_ID  Invalid Display Id
;  418          0495  1 !
;  419          0496  1 ! SIDE EFFECTS:
;  420          0497  1 !
;  421          0498  1 !       NONE
;  422          0499  1 !--
;  423          0500  1 !
;  424          0501  2     BEGIN
;  425          0502  2     LOCAL
;  426          0503  2         DCB : REF BLOCK [,BYTE];        ! Addr of display control block
;  427          0504  2
;  428          0505  2 !+
;  429          0506  2 ! Check for right number of arguments.
;  430          0507  2 !-
;  431          0508  2     $SMG$VALIDATE_ARGCOUNT ( 1, 1);
;  432          0509  2
;  433          0510  2     $SMG$GET_DCB (.DISPLAY_ID, DCB);    ! Get DCB address
;  434          0511  2
;  435          0512  2 !+
;  436          0513  2 ! Increment count of number of SMG$END_DISPLAY_UPDATE calls we need to
;  437          0514  2 ! see before we resume outputing from this virtual display.
;  438          0515  2 ! Let the user know what the previous state of batching was by
;  439          0516  2 ! our status return (in case he's interested).
;  440          0517  2 !-
;  441          0518  2     DCB [DCB_L_BATCH_LEVEL] = .DCB [DCB_L_BATCH_LEVEL] + 1;
;  442          0519  2
;  443          0520  2     IF .DCB [DCB_L_BATCH_LEVEL] EQL 1
;  444          0521  2     THEN
```

```
;  445     0522  1          BEGIN   ! Begin of batching operation
;  446     0523  1          LOCAL
;  447     0524  1              STATUS,                         ! Status of subroutine calls
;  448     0525  1              PP : REF BLOCK [,BYTE];         ! Addr of a pasting packet
;  449     0526  1
;  450     0527  1          !+
;  451     0528  1          ! Make a copy of current DCB and leave its address in the
;  452     0529  1          ! backup pointer field of the current DCB.
;  453     0530  1          !-
;  454     0531  1          IF NOT (STATUS = SMG$$DUPL_VIRTUAL_DISPLAY (
;  455     0532  1                                      DCB,                                ! Addr of curr.
;  456     0533  1                                      DCB [DCB_A_BACKUP_DCB])) ! Where new
;  457     0534  1                                                               ! get stored.
;  458     0535  1          THEN
;  459     0536  1              RETURN (.STATUS);
;  460     0537  1
;  461     0538  1          !+
;  462     0539  1          ! Walk chain of all pasteboards that we are pasted to and for
;  463     0540  1          ! each encountered, in the affected pasting packet, set the
;  464     0541  1          ! back pointer to the DCB to be the new DCB we've created.
;  465     0542  1          ! This causes all mapping operations to reference the backed up
;  466     0543  1          ! DCB as the source of images for the screen.  This is the
;  467     0544  1          ! desired action since the current DCB is undergoing changes
;  468     0545  1          ! which should not yet find their way to the screen -- its
;  469     0546  1          ! batched.
;  470     0547  1          !-
;  471     0548  1          PP = .DCB [DCB_A_PP_NEXT];
;  472     0549  1
;  473     0550  1          IF .PP EQL 0
;  474     0551  1          THEN
;  475     0552  1              RETURN (SMG$_FATERRLIB);                ! should never be 0
;  476     0553  1                                                      ! (points to self when empty)
;  477     0554  1
;  478     0555  1          WHILE .PP NEQ DCB [DCB_A_PP_NEXT]           ! While any remain
;  479     0556  1          DO
;  480     0557  2              BEGIN
;  481     0558  2              PP [PP_A_DCB_ADDR] = .DCB [DCB_A_BACKUP_DCB];
;  482     0559  2              PP = .PP [PP_A_NEXT_DCB];   ! Step to next packet
;  483     0560  1              END;
;  484     0561  1          RETURN  SS$_NORMAL;
;  485     0562  1
;  486     0563  1          END     ! Begin of batching operation
;  487     0564  1
;  488     0565  2      ELSE
;  489     0566  2
;  490     0567  2          RETURN  SMG$_BATWAS_ON
;  491     0568  2
;  492     0569  1      END;                    ! End of routine SMG$BEGIN_DISPLAY_UPDATE


;                                                       .TITLE  SMG$DISPLAY_OUTPUT SMG$DISPLAY_OUTPUT - Output
                                                                                   Virtual Displays
                                                       .IDENT  \1-073\

                                                       .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_DD_FIELDS
                                                        .EXTRN  SMG$$UPDATE_PHYSICAL_CURSOR
                                                        .EXTRN  SMG$_WRONUMARG

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

                                0004 00000              .ENTRY  SMG$BEGIN_DISPLAY_UPDATE, Save R2                       ; 0466
                          5E    04   C2 00002          SUBL2   #4, SP
                          01    6C   91 00005          CMPB    (AP), #1                                                ; 0508
                                08   13 00008          BEQL    1$
                       50 00000000G   8F   D0 0000A     MOVL    #SMG$_WRONUMARG, R0
                                04 00011               RET
                          50    04   BC   D0 00012 1$:  MOVL    @DISPLAY_ID, R0                                         ; 0510
                       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$:  MOVL    @DISPLAY_ID, DCB
                          50          6E   D0 0002F     MOVL    DCB, R0                                                 ; 0518
                                1C   A0   D6 00032     INCL    28(R0)
                          01    1C   A0   D1 00035     CMPL    28(R0), #1                                              ; 0520
                                38   12 00039          BNEQ    6$
                                40   A0   9F 0003B     PUSHAB  64(R0)                                                  ; 0533
                                04   AE   9F 0003E     PUSHAB  DCB                                                     ; 0531
                00000000G       00        02   FB 00041     CALLS   #2, SMG$$DUPL_VIRTUAL_DISPLAY                       ; 0533
                                2F        50   E9 00048     BLBC    STATUS, 7$
                                51        6E   D0 0004B     MOVL    DCB, R1                                         ; 0548
                                50   20   A1   D0 0004E     MOVL    32(R1), PP
                                08   12 00052          BNEQ    4$                                                      ; 0550
                       50 00000000G   8F   D0 00054     MOVL    #SMG$_FATERRLIB, R0                                     ; 0552
                                04 0005B               RET
                          52    20   A1   9E 0005C 4$:  MOVAB   32(R1), R2                                             ; 0555
                          52          50   D1 00060     CMPL    PP, R2
                                0A   13 00063          BEQL    5$
                       10 A0    40   A1   D0 00065     MOVL    64(R1), 16(PP)                                          ; 0558
                          50        60   D0 0006A     MOVL    (PP), PP                                                ; 0559
                                ED   11 0006D          BRB     4$                                                      ; 0555
                          50        01   D0 0006F 5$:  MOVL    #1, R0                                                  ; 0567
                                04 00072               RET
                       50 00000000G   8F   D0 00073 6$:  MOVL    #SMG$_BATWAS_ON, R0
                                04 0007A 7$:  RET                                                                       ; 0569

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

```
;  493          0570  1 !<BLF/PAGE>
```

```
;   495          0571  1 %SBTTL 'SMG$BEGIN_PASTEBOARD_UPDATE - Begin batch of updates to pasteboard'
;   496          0572  1 GLOBAL ROUTINE SMG$BEGIN_PASTEBOARD_UPDATE ( PBID ) =
;   497          0573  1 !++
;   498          0574  1 ! FUNCTIONAL DESCRIPTION:
;   499          0575  1 !
;   500          0576  1 !       Disable outputing this pasteboard to the screen until the
;   501          0577  1 !       matching call to SMG$END_PASTEBOARD_UPDATE is encountered.
;   502          0578  1 !
;   503          0579  1 ! CALLING SEQUENCE:
;   504          0580  1 !
;   505          0581  1 !       ret_status.wlc.v = SMG$BEGIN_PASTEBOARD_UPDATE ( PBID.rl.r)
;   506          0582  1 !
;   507          0583  1 ! FORMAL PARAMETERS:
;   508          0584  1 !
;   509          0585  1 !       PBID.rl.r               Pasteboard id.
;   510          0586  1 !
;   511          0587  1 ! IMPLICIT INPUTS:
;   512          0588  1 !
;   513          0589  1 !       NONE
;   514          0590  1 !
;   515          0591  1 ! IMPLICIT OUTPUTS:
;   516          0592  1 !
;   517          0593  1 !       NONE
;   518          0594  1 !
;   519          0595  1 ! COMPLETION STATUS:
;   520          0596  1 !
;   521          0597  1 !       SS$_NORMAL      Normal successful completion, batching has been
;   522          0598  1 !                       initiated
;   523          0599  1 !       SMG$_BATWAS_ON  Success, but note that batching was already on
;   524          0600  1 !       SMG$_INVPAS_ID  Invalid Pasteboard Id
;   525          0601  1 !
;   526          0602  1 ! SIDE EFFECTS:
;   527          0603  1 !
;   528          0604  1 !       NONE
;   529          0605  1 !--
;   530          0606  1
;   531          0607  2     BEGIN
;   532          0608  2     LOCAL
;   533          0609  2         PBCB : REF BLOCK [,BYTE];       ! Addr of pasteboard control block
;   534          0610  2
;   535          0611  2 !+
;   536          0612  2 ! Check for right number of arguments.
;   537          0613  2 !-
;   538          0614  2     $SMG$VALIDATE_ARGCOUNT ( 1, 1);
;   539          0615  2
;   540          0616  2     $SMG$GET_PBCB (.PBID, PBCB);        ! Get PBCB address
;   541          0617  2
;   542          0618  2     RETURN (SMG$$BEGIN_PASTEBOARD_UPDATE_R1 (.PBCB)); ! Do work in inner routine
;   543          0619  2
;   544          0620  1     END;                    ! End of routine SMG$BEGIN_PASTEBOARD_UPDATE


                                    0000 00000          .ENTRY  SMG$BEGIN_PASTEBOARD_UPDATE, Save nothing   ; 0572
                            01   6C  91 00002          CMPB    (AP), #1                                    ; 0614
```

```
                                    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                                               ; 0616
                                        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                         ; 0618
                                        04 00039                  RET                                                             ; 0620

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

;  545          0621  1 !<BLF/PAGE>
```

```
;  547          0622  1 %SBTTL 'SMG$END_DISPLAY_UPDATE - End batch of updates to display'
;  548          0623  1 GLOBAL ROUTINE SMG$END_DISPLAY_UPDATE ( DISPLAY_ID ) =
;  549          0624  1 !++
;  550          0625  1 ! FUNCTIONAL DESCRIPTION:
;  551          0626  1 !
;  552          0627  1 !       Reduce the number of calls to SMG$END_DISPLAY_UPDATE that will
;  553          0628  1 !       be needed before we resume outputing from this display.  If
;  554          0629  1 !       this call makes this count go to zero, flush the display to
;  555          0630  1 !       the screen.
;  556          0631  1 !       If the level is already 0, we return a success status
;  557          0632  1 !       informing caller that the batching level was already 0,
;  558          0633  1 !       but we otherwise allow this.  This gives a user a guaranteed
;  559          0634  1 !       method of ending batching.
;  560          0635  1 !
;  561          0636  1 ! CALLING SEQUENCE:
;  562          0637  1 !
;  563          0638  1 !       ret_status.wlc.v = SMG$END_DISPLAY_UPDATE ( DISPLAY_ID.rl.r)
;  564          0639  1 !
;  565          0640  1 ! FORMAL PARAMETERS:
;  566          0641  1 !
;  567          0642  1 !       DISPLAY_ID.rl.r         Display id of virtual display.
;  568          0643  1 !
;  569          0644  1 ! IMPLICIT INPUTS:
;  570          0645  1 !
;  571          0646  1 !       DCB [DCB_L_BATCH_LEVEL]
;  572          0647  1 !
;  573          0648  1 ! IMPLICIT OUTPUTS:
;  574          0649  1 !
;  575          0650  1 !       DCB [DCB_L_BATCH_LEVEL] gets decremented
;  576          0651  1 !
;  577          0652  1 ! COMPLETION STATUS:
;  578          0653  1 !
;  579          0654  1 !       SS$_NORMAL       Normal successful completion
;  580          0655  1 !       SMG$_BATSTIPRO   Success; but batching is still in progress.
;  581          0656  1 !       SMG$_INVDIS_ID   Invalid Display Id
;  582          0657  1 !       SMG$_BATWASOFF   Success; but batching was already off
;  583          0658  1 !
;  584          0659  1 ! SIDE EFFECTS:
;  585          0660  1 !
;  586          0661  1 !       NONE
;  587          0662  1 !--
;  588          0663  1
;  589          0664  2     BEGIN
;  590          0665  2     LOCAL
;  591          0666  2         STATUS,                             ! Status to return
;  592          0667  2         DCB : REF BLOCK [,BYTE];            ! Addr of display control block
;  593          0668  2
;  594          0669  2 !+
;  595          0670  2 ! Check for right number of arguments.
;  596          0671  2 !-
;  597          0672  2     $SMG$VALIDATE_ARGCOUNT ( 1, 1);
;  598          0673  2
;  599          0674  2     $SMG$GET_DCB (.DISPLAY_ID, DCB);    ! Get DCB address
;  600          0675  2
;  601          0676  2 !+
;  602          0677  2 ! If current count is greater than 1, simply reduce it by one and
;  603          0678  2 ! return to caller.  If it is one, reduce it to zero and cause the
```

```
;  604          0679  2 ! current contents of this window to be flushed to the screen (if need
;  605          0680  2 ! be).
;  606          0681  2 !-
;  607          0682  2     IF .DCB [DCB_L_BATCH_LEVEL] GTR 1
;  608          0683  2     THEN
;  609          0684  3         BEGIN
;  610          0685  3         DCB [DCB_L_BATCH_LEVEL] = .DCB [DCB_L_BATCH_LEVEL] - 1;
;  611          0686  3         STATUS = SMG$_BATSTIPRO;          ! Ok, but batching is still in
;  612          0687  3         END                               ! progress
;  613          0688  3
;  614          0689  2     ELSE
;  615          0690  2
;  616          0691  3         BEGIN   ! Level is currently 0 or 1
;  617          0692  3         LOCAL
;  618          0693  3             BATCH_STATUS;
;  619          0694  3
;  620          0695  3         BATCH_STATUS = SS$_NORMAL;
;  621          0696  3         IF .DCB [DCB_L_BATCH_LEVEL] EQL 0
;  622          0697  3         THEN
;  623          0698  4             RETURN ( SMG$_BATWASOFF)    ! Ok, but batching was already off
;  624          0699  3         ELSE
;  625          0700  3
;  626          0701  4             BEGIN       ! Reduction from 1 to 0
;  627          0702  4             LOCAL
;  628          0703  4                 PP : REF $PP_DECL;      ! Addr of a pasting packet
;  629          0704  4
;  630          0705  4             DCB [DCB_L_BATCH_LEVEL] = 0;
;  631          0706  4             !+
;  632          0707  4             ! Walk chain of pastboards to which we are pasted.  For each
;  633          0708  4             ! pasting packet involved, set the pointer in the pasting
;  634          0709  4             ! packet back to the original DCB since it is now free of
;  635          0710  4             ! batching and its contents can be mapped to the screen.
;  636          0711  4             ! Note: We do not deallocate the backup DCB.  The assumption
;  637          0712  4             ! is that if the caller batched this virtual display once,
;  638          0713  4             ! he is likely to do it again.  This saves us having to
;  639          0714  4             ! deallocate now only to have to do another
;  640          0715  4             ! SMG$$DUPL_VIRTUAL_DISPLAY when he starts batching again.
;  641          0716  4             !-
;  642          0717  4             PP = .DCB [DCB_A_PP_NEXT];
;  643          0718  4
;  644          0719  4             IF .PP EQL 0
;  645          0720  4             THEN
;  646          0721  4                 RETURN (SMG$_FATERRLIB);        ! should never be 0
;  647          0722  4
;  648          0723  4             WHILE .PP NEQ DCB [DCB_A_PP_NEXT]   ! While any remain...
;  649          0724  4             DO
;  650          0725  5                 BEGIN
;  651          0726  5                 PP [PP_A_DCB_ADDR] = .DCB;      ! To orig. DCB
;  652          0727  5                 PP = .PP [PP_A_NEXT_DCB];       ! To next packet
;  653          0728  4                 END;
;  654          0729  4
;  655          0730  4             !+
;  656          0731  4             ! If DCB_V_PP_MISMATCH is set, a change has been made to
;  657          0732  4             ! this DCB which requires the pasting packet constants to be
;  658          0733  4             ! recalculated.  However, the DCB was "batched" at the time
;  659          0734  4             ! the change took place, so we do the pasting packet update
;  660          0735  4             ! now as part of the unbatching process.
```

```
;  661     0736  4              !-
;  662     0737  4              IF .DCB [DCB_V_PP_MISMATCH]
;  663     0738  4              THEN
;  664     0739  5                  BEGIN   ! Unbatching clean up
;  665     0740  5                  LOCAL
;  666     0741  5                      CURR_PP : REF $PP_DECL; ! Addr of a pasting packet
;  667     0742  5
;  668     0743  6                  IF NOT (STATUS = SMG$$RECALC_PP_FIELDS (.DCB))
;  669     0744  5                  THEN
;  670     0745  5                      RETURN (.STATUS);
;  671     0746  5
;  672     0747  5                  DCB [DCB_V_PP_MISMATCH] = 0 ; ! Knock down flag
;  673     0748  5
;  674     0749  5                  !+
;  675     0750  5                  ! Since DCB_V_PP_MISMATCH was set, the change to the DCB
;  676     0751  5                  ! included dimensional changes.  This means we will have
;  677     0752  5                  ! to remap the whole pasteboard buffer from the bottom
;  678     0753  5                  ! outward -- for each pasteboard that we are pasted to.
;  679     0754  5                  !-
;  680     0755  5                  CURR_PP = .DCB [DCB_A_PP_NEXT];
;  681     0756  5
;  682     0757  5                  IF .CURR_PP EQL 0
;  683     0758  5                  THEN
;  684     0759  5                      RETURN (SMG$_FATERRLIB);    ! should never be 0
;  685     0760  5
;  686     0761  5                  WHILE .CURR_PP NEQ DCB [DCB_A_PP_NEXT]
;  687     0762  5                  DO
;  688     0763  6                      BEGIN       ! For all pasteboards
;  689     0764  6                      LOCAL
;  690     0765  6                          PBCB : REF $PBCB_DECL;  ! Addr of a pasteboard
;  691     0766  6                                                  ! control block
;  692     0767  6
;  693     0768  6                      PBCB = .CURR_PP [PP_A_PBCB_ADDR];
;  694     0769  7                      IF NOT (STATUS = SMG$$CHECK_FOR_OUTPUT_PBCB (.PBCB))
;  695     0770  6                      THEN
;  696     0771  6                          RETURN (.STATUS); ! Quit on first failure
;  697     0772  6
;  698     0773  6                      CURR_PP = .CURR_PP [PP_A_NEXT_DCB]; ! To next packet
;  699     0774  5                      END;        ! For all pasteboards
;  700     0775  5
;  701     0776  5                  RETURN (SS$_NORMAL);
;  702     0777  5                  END     ! Unbatching clean up
;  703     0778  5
;  704     0779  4              ELSE
;  705     0780  4
;  706     0781  5                  BEGIN   ! No cleanup needed
;  707     0782  5                  !+
;  708     0783  5                  ! Call SMG$$CHECK_FOR_OUTPUT_DCB to cause the contents
;  709     0784  5                  ! of this virtual display to be flushed to the screen.
;  710     0785  5                  ! If that fails, then return its status as our status.
;  711     0786  5                  ! If that succeeded then return our previously
;  712     0787  5                  ! calculated status.
;  713     0788  5                  !-
;  714     0789  5                  STATUS = SMG$$CHECK_FOR_OUTPUT_DCB (.DCB,
;  715     0790  5                                                      SMG$C_END_DISPLAY_UPDATE);
;  716     0791  5                  IF .STATUS THEN STATUS=.BATCH_STATUS
;  717     0792  4                  END;    ! No cleanup needed
```

```
;  718          0793  3              END;                ! Reduction from 1 to 0
;  719          0794  2          END;        ! Level is currently 0 or 1
;  720          0795  2
;  721          0796  2      RETURN (.STATUS);
;  722          0797  1      END;                        ! End of routine SMG$END_DISPLAY_UPDATE


                                      001C 00000          .ENTRY  SMG$END_DISPLAY_UPDATE, Save R2,R3,R4      ; 0623
                                01    6C   91 00002          CMPB    (AP), #1                                   ; 0672
                                      08   13 00005          BEQL    1$
                                50 00000000G  8F   D0 00007          MOVL    #SMG$_WRONUMARG, R0
                                      04   0000E          RET
                                50      04    BC   D0 0000F 1$:      MOVL    @DISPLAY_ID, R0                    ; 0674
                             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$:      MOVL    @DISPLAY_ID, DCB
                                01      1C    A2   D1 0002C          CMPL    28(DCB), #1                        ; 0682
                                      0B   15 00030          BLEQ    4$
                                      1C    A2   D7 00032          DECL    28(DCB)                            ; 0685
                                50 00000000G  8F   D0 00035          MOVL    #SMG$_BATSTIPRO, STATUS            ; 0686
                                      04   0003C          RET                                                ; 0682
                                54            01   D0 0003D 4$:      MOVL    #1, BATCH_STATUS                   ; 0695
                                      1C    A2   D5 00040          TSTL    28(DCB)                            ; 0696
                                      08   12 00043          BNEQ    5$
                                50 00000000G  8F   D0 00045          MOVL    #SMG$_BATWASOFF, R0                ; 0698
                                      04   0004C          RET
                                      1C    A2   D4 0004D 5$:      CLRL    28(DCB)                            ; 0705
                                51      20    A2   D0 00050          MOVL    32(DCB), PP                        ; 0717
                                      2D   13 00054          BEQL    8$                                         ; 0719
                                53      20    A2   9E 00056 6$:      MOVAB   32(DCB), R3                        ; 0723
                                53            51   D1 0005A          CMPL    PP, R3
                                      09   13 0005D          BEQL    7$
                             10 A1            52   D0 0005F          MOVL    DCB, 16(PP)                        ; 0726
                                51            61   D0 00063          MOVL    (PP), PP                           ; 0727
                                      EE   11 00066          BRB     6$                                         ; 0723
                          3E 34 A2            03   E1 00068 7$:      BBC     #3, 52(DCB), 11$                   ; 0737
                                      52   DD 0006D          PUSHL   DCB                                        ; 0743
                    00000000G 00              01   FB 0006F          CALLS   #1, SMG$$RECALC_PP_FIELDS
                                41            50   E9 00076          BLBC    STATUS, 12$
                             34 A2            08   8A 00079          BICB2   #8, 52(DCB)                        ; 0747
                                53      20    A2   D0 0007D          MOVL    32(DCB), CURR_PP                   ; 0755
                                      08   12 00081          BNEQ    9$                                         ; 0757
                                50 00000000G  8F   D0 00083 8$:      MOVL    #SMG$_FATERRLIB, R0                ; 0759
                                      04   0008A          RET
                                51      20    A2   9E 0008B 9$:      MOVAB   32(DCB), R1                        ; 0761
                                51            53   D1 0008F          CMPL    CURR_PP, R1
                                      13   13 00092          BEQL    10$
                                51      14    A3   D0 00094          MOVL    20(CURR_PP), PBCB                  ; 0768
                                      51   DD 00098          PUSHL   PBCB                                       ; 0769
                          0000V CF              01   FB 0009A          CALLS   #1, SMG$$CHECK_FOR_OUTPUT_PBCB
```

```
                                 18         50  E9 0009F          BLBC    STATUS, 12$                                  ;
                                 53         63  D0 000A2          MOVL    (CURR_PP), CURR_PP                           ; 0773
                                            E4  11 000A5          BRB     9$                                           ; 0761
                                 50         01  D0 000A7 10$:     MOVL    #1, R0                                       ; 0776
                                                04 000AA          RET                                                  ;
                                            1D  DD 000AB 11$:     PUSHL   #29                                          ; 0789
                                            52  DD 000AD          PUSHL   DCB                                          ;
                           0000V            CF  02 FB 000AF       CALLS   #2, SMG$$CHECK_FOR_OUTPUT_DCB                ;
                                 03         50  E9 000B4          BLBC    STATUS, 12$                                  ; 0791
                                 50         54  D0 000B7          MOVL    BATCH_STATUS, STATUS                         ;
                                                04 000BA 12$:     RET                                                  ; 0797

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

;  723          0798  1 !<BLF/PAGE>
```

```
;   725          0799  1 %SBTTL 'SMG$END_PASTEBOARD_UPDATE - End batch of updates to pasteboard'
;   726          0800  1 GLOBAL ROUTINE SMG$END_PASTEBOARD_UPDATE ( PBID ) =
;   727          0801  1 !++
;   728          0802  1 ! FUNCTIONAL DESCRIPTION:
;   729          0803  1 !
;   730          0804  1 !       Reduce the number of calls to SMG$END_PASTEBOARD_UPDATE that will
;   731          0805  1 !       be needed before we resume outputing from this pasteboard.  If
;   732          0806  1 !       this call makes this count go to zero, flush the pasteboard to
;   733          0807  1 !       the screen.
;   734          0808  1 !       If the level is already 0, we return a success status
;   735          0809  1 !       informing caller that the batching level was already 0,
;   736          0810  1 !       but we otherwise allow this.  This gives a user a guaranteed
;   737          0811  1 !       method of ending batching.
;   738          0812  1 !
;   739          0813  1 ! CALLING SEQUENCE:
;   740          0814  1 !
;   741          0815  1 !       ret_status.wlc.v = SMG$END_PASTEBOARD_UPDATE ( PBID.rl.r)
;   742          0816  1 !
;   743          0817  1 ! FORMAL PARAMETERS:
;   744          0818  1 !
;   745          0819  1 !       PBID.rl.r               Pasteboard id.
;   746          0820  1 !
;   747          0821  1 ! IMPLICIT INPUTS:
;   748          0822  1 !
;   749          0823  1 !       PBCB [PBCB_L_BATCH_LEVEL]
;   750          0824  1 !
;   751          0825  1 ! IMPLICIT OUTPUTS:
;   752          0826  1 !
;   753          0827  1 !       PBCB [PBCB_L_BATCH_LEVEL] gets decremented
;   754          0828  1 !
;   755          0829  1 ! COMPLETION STATUS:
;   756          0830  1 !
;   757          0831  1 !       SS$_NORMAL      Normal successful completion
;   758          0832  1 !       SMG$_BATSTIPRO  Success; but batching is still in progress.
;   759          0833  1 !       SMG$_INVPAS_ID  Invalid Pasteboard Id
;   760          0834  1 !       SMG$_BATWASOFF  Success; but batching was already off
;   761          0835  1 !
;   762          0836  1 ! SIDE EFFECTS:
;   763          0837  1 !
;   764          0838  1 !       NONE
;   765          0839  1 !--
;   766          0840  1
;   767          0841  2     BEGIN
;   768          0842  2     LOCAL
;   769          0843  2         STATUS,                             ! Status to return
;   770          0844  2         PBCB : REF BLOCK [,BYTE];           ! Addr of pasteboard control block
;   771          0845  2
;   772          0846  2 !+
;   773          0847  2 ! Check for right number of arguments.
;   774          0848  2 !-
;   775          0849  2     $SMG$VALIDATE_ARGCOUNT ( 1, 1);
;   776          0850  2
;   777          0851  2     $SMG$GET_PBCB (.PBID, PBCB);            ! Get PBCB address
;   778          0852  2
;   779          0853  2     RETURN (SMG$$END_PASTEBOARD_UPDATE_R2 (.PBCB)); ! Do work in inner routine
;   780          0854  2
;   781          0855  1     END;                    ! End of routine SMG$END_PASTEBOARD_UPDATE
```

```
                                                 0004 00000          .ENTRY  SMG$END_PASTEBOARD_UPDATE, Save R2                  ; 0800
                                        01        6C  91 00002          CMPB    (AP), #T                                            ; 0849
                                                  08  13 00005          BEQL    1$
                                        50 00000000G  8F  D0 00007          MOVL    #SMG$_WRONUMARG, R0
                                                      04 0000E          RET
                                        50        04  BC  D0 0000F 1$:       MOVL    @PBID, R0                                   ; 0851
                                                  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               ; 0853
                                                      04 00039          RET                                                 ; 0855

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

;  782          0856  1 !<BLF/PAGE>
```

```
;  784          0857  1 %SBTTL 'SMG$FIND_CURSOR_DISPLAY - Find which virtual display contains physical cursor'
;  785          0858  1 GLOBAL ROUTINE SMG$FIND_CURSOR_DISPLAY (
;  786          0859  1                                         PASTEBOARD_ID,
;  787          0860  1                                         DISPLAY_ID
;  788          0861  1                                             ) =
;  789          0862  1 !++
;  790          0863  1 ! FUNCTIONAL DESCRIPTION:
;  791          0864  1 !
;  792          0865  1 !       This routine determines which virtual display contains the
;  793          0866  1 !       current physical cursor on the screen.  The pasted virtual
;  794          0867  1 !       displays are searched from the deepest pasted one to the
;  795          0868  1 !       outer-most pasted one.  The outer-most virtual display that
;  796          0869  1 !       encompasses the physical cursor is the one selected and its
;  797          0870  1 !       display id is returned.
;  798          0871  1 !
;  799          0872  1 !       It is possible that no virtual display contains the cursor,
;  800          0873  1 !       in which case a display id of zero (an invalid display id) is
;  801          0874  1 !       returned.
;  802          0875  1 !
;  803          0876  1 ! CALLING SEQUENCE:
;  804          0877  1 !
;  805          0878  1 !       ret_status.wlc.v = SMG$FIND_CURSOR_DISPLAY (
;  806          0879  1 !                                       PASTEBOARD_ID.rl.r,
;  807          0880  1 !                                       DISPLAY_ID.wl.r)
;  808          0881  1 !
;  809          0882  1 ! FORMAL PARAMETERS:
;  810          0883  1 !
;  811          0884  1 !       PASTEBOARD_ID.rl.r      Address of a longword containing the
;  812          0885  1 !                               pasteboard_id for the device for which
;  813          0886  1 !                               the information is desired.
;  814          0887  1 !
;  815          0888  1 !       DISPLAY_ID.wl.r         Address of a longword to receive the
;  816          0889  1 !                               virtual display_id of the virtual
;  817          0890  1 !                               display which contains the cursor.
;  818          0891  1 !                               The outer-most pasted virtual display
;  819          0892  1 !                               that contains the cursor is selected.
;  820          0893  1 !                               If no virtual display can be found,
;  821          0894  1 !                               a display id of 0 ( an invalid display
;  822          0895  1 !                               id) is supplied.
;  823          0896  1 !
;  824          0897  1 !
;  825          0898  1 ! IMPLICIT INPUTS:
;  826          0899  1 !
;  827          0900  1 !       NONE
;  828          0901  1 !
;  829          0902  1 ! IMPLICIT OUTPUTS:
;  830          0903  1 !
;  831          0904  1 !       NONE
;  832          0905  1 !
;  833          0906  1 ! COMPLETION STATUS:
;  834          0907  1 !
;  835          0908  1 !       SS$_NORMAL          Normal successful completion
;  836          0909  1 !       SMG$_INVPAS_ID      Invalid pasteboard id
;  837          0910  1 !       SMG$_WRONUMARG      Wrong number of arguments
;  838          0911  1 !
;  839          0912  1 ! SIDE EFFECTS:
;  840          0913  1 !
```

```
;  841     0914  1 !       NONE
;  842     0915  1 !--
;  843     0916  2     BEGIN
;  844     0917  2     LOCAL
;  845     0918  2         PBCB : REF $PBCB_DECL,  ! Address of pasteboard control block
;  846     0919  2
;  847     0920  2         WCB  : REF $WCB_DECL,   ! Address of window control block
;  848     0921  2
;  849     0922  2         PP;                     ! An address pointing at the PP queue
;  850     0923  2                                 ! header in a pasting packet.  This
;  851     0924  2                                 ! address is not the base of the
;  852     0925  2                                 ! pasting packet.
;  853     0926  2 !+
;  854     0927  2 ! Check for right number of arguments.
;  855     0928  2 !-
;  856     0929  2     $SMG$VALIDATE_ARGCOUNT ( 2, 2);
;  857     0930  2
;  858     0931  2 !+
;  859     0932  2 ! Get address of pasteboard control block and the WCB that goes with it.
;  860     0933  2 !-
;  861     0934  2     $SMG$GET_PBCB ( .PASTEBOARD_ID, PBCB );
;  862     0935  2     WCB = .PBCB [PBCB_A_WCB];
;  863     0936  2
;  864     0937  2 !+
;  865     0938  2 ! Set up an answer of none in case search turns up nothing.
;  866     0939  2 !-
;  867     0940  2     .DISPLAY_ID = 0;
;  868     0941  2
;  869     0942  2 !+
;  870     0943  2 ! Search all the pasting packets for this pasteboard, from the deepest-
;  871     0944  2 ! pasted one to the outer-most pasted one.  For each such pasting
;  872     0945  2 ! packet, see if the physical cursor position lies inside the
;  873     0946  2 ! mapping of the associated virtual display.  For each virtual display
;  874     0947  2 ! found, set up output parameter.  The last one found (if any) will be
;  875     0948  2 ! the one the caller sees.
;  876     0949  2 !-
;  877     0950  2     PP = .PBCB [PBCB_A_PP_PREV]; ! Start with inner-most pasted one
;  878     0951  2
;  879     0952  2     IF .PP EQL 0
;  880     0953  2     THEN
;  881     0954  2         RETURN (SMG$_FATERRLIB);        ! should never be 0
;  882     0955  2
;  883     0956  2     WHILE .PP NEQ PBCB [PBCB_A_PP_NEXT]
;  884     0957  2     DO
;  885     0958  3         BEGIN   ! For all pasting packets
;  886     0959  3         LOCAL
;  887     0960  3             PP_BASE : REF $PP_DECL;     ! Base address of a pasting
;  888     0961  3                                         ! packet
;  889     0962  3
;  890     0963  3         PP_BASE = .PP - PP_PBCB_QUEUE_OFFSET;
;  891     0964  3
;  892     0965  3         IF .PP_BASE [PP_W_ROWS_TO_MOVE] NEQ 0
;  893     0966  3         THEN
;  894     0967  4             BEGIN       ! Projects somewhere on visible pasteboard
;  895     0968  4             LOCAL
;  896     0969  4                 D_PROJ : BLOCK [8,BYTE]; ! Representation of virtual
;  897     0970  4                                          ! display's projection on
```

```
;  898          0971  4                                          ! pasteboard.
;  899          0972  4
;  900          0973  4              D_PROJ [DCB_W_ROW_START] = .PP_BASE [PP_W_FIRST_WCB_ROW];
;  901          0974  4
;  902          0975  4              D_PROJ [DCB_W_NO_ROWS]   = .PP_BASE [PP_W_LAST_WCB_ROW] -
;  903          0976  4                                         .PP_BASE [PP_W_FIRST_WCB_ROW] +1;
;  904          0977  4
;  905          0978  4              D_PROJ [DCB_W_COL_START] = .PP_BASE [PP_W_FIRST_WCB_COL];
;  906          0979  4
;  907          0980  4              D_PROJ [DCB_W_NO_COLS]   = .PP_BASE [PP_W_LAST_WCB_COL] -
;  908          0981  4                                         .PP_BASE [PP_W_FIRST_WCB_COL] +1;
;  909          0982  4
;  910          0983  4              !+
;  911          0984  4              ! Check to see if cursor lies within projection of this
;  912          0985  4              ! virtual display.
;  913          0986  4              !-
;  914          0987  4              IF SMG$$POINT_IN_RECT_R3 (WCB [WCB_W_OLD_CUR_COL],
;  915          0988  4                                        WCB [WCB_W_OLD_CUR_ROW],
;  916          0989  4                                        D_PROJ) EQL 0
;  917          0990  4              THEN
;  918          0991  4                  .DISPLAY_ID = .PP_BASE [PP_A_DCB_ADDR]; !  Save candidate
;  919          0992  4
;  920          0993  3              END;          ! Projects somewhere on visible pasteboard
;  921          0994  3
;  922          0995  3          PP = .PP_BASE [PP_A_PREV_PBCB] ; ! To next outer pasting
;  923          0996  2          END;     ! For all pasting packets
;  924          0997  2
;  925          0998  2      RETURN (SS$_NORMAL);
;  926          0999  1      END;                    ! Routine SMG$FIND_CURSOR_DISPLAY


                                    00FC 00000          .ENTRY  SMG$FIND_CURSOR_DISPLAY, Save R2,R3,R4,R5,- ; 0858
                                                                R6,R7
                                5E          08  C2 00002        SUBL2   #8, SP                              ;
                                02          6C  91 00005        CMPB    (AP), #2                            ; 0929
                                            08  13 00008        BEQL    1$
                                50 00000000G    8F 0000A        MOVL    #SMG$_WRONUMARG, R0
                                                04 00011        RET
                                50      04      BC D? 00012 1$: MOVL    @PASTEBOARD_ID, R0                  ; 0934
                                            11  19 00016        BLSS    2$
                          00000000G 00          50 D1 00018        CMPL    R0, PBD_L_COUNT
                                            08  14 0001F        BGTR    2$
                       08 00000000G 00          50 E0 00021        BBS     R0, PBD_V_PB_AVAIL, 3$
                                50 00000000G    8F D0 00029 2$: MOVL    #SMG$_INVPAS_ID, R0
                                                04 00030        RET
                                56 00000000G0040 D0 00031 3$: MOVL    PBD_A_PBCB[R0], PBCB
                                57      08      A6 D0 00039        MOVL    8(PBCB), WCB                        ; 0935
                                        08      BC D4 0003D        CLRL    @DISPLAY_ID                         ; 0940
                                55      04      A6 D0 00040        MOVL    4(PBCB), PP                         ; 0950
                                            08  12 00044        BNEQ    4$                                  ; 0952
                                50 00000000G    8F D0 00046        MOVL    #SMG$_FATERRLIB, R0                 ; 0954
                                                04 0004D        RET
                                56          55  D1 0004E 4$: CMPL    PP, PBCB                            ; 0956
                                            4F  13 00051        BEQL    6$
```

```
                                54        F8   A5  9E  00053          MOVAB   -8(R5), PP_BASE                    ; 0963
                                          1C   A4  B5  00057          TSTW    28(PP_BASE)                       ; 0965
                                          40   13  0005A              BEQL    5$
                                6E        2F   A4  B0  0005C          MOVW    47(PP_BASE), D_PROJ               ; 0973
                                50        31   A4  3C  00060          MOVZWL  49(PP_BASE), R0                   ; 0976
                                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             ; 0978
                                50        35   A4  3C  00075          MOVZWL  53(PP_BASE), R0                   ; 0981
                                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                        ; 0988
                                51        24   A7  9E  00088          MOVAB   36(WCB), R1                       ; 0987
                                50        26   A7  9E  0008C          MOVAB   38(WCB), R0                       ; 0988
                                     0000V     30  00090              BSBW    SMG$$POINT_IN_RECT_R3             ; 0989
                                               50  D5  00093          TSTL    R0                                ; 0991
                                               05  12  00095          BNEQ    5$
                          08   BC        10   A4  D0  00097          MOVL    16(PP_BASE), @DISPLAY_ID          ; 0995
                                55        0C   A4  D0  0009C  5$:     MOVL    12(PP_BASE), PP                   ; 0956
                                               AC  11  000A0          BRB     4$                                ; 0998
                                50             01  D0  000A2  6$:     MOVL    #1, R0                            ; 0999
                                               04  000A5              RET

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

;  927          1000  1 !<BLF/PAGE>
```

```
;  929          1001  1 %SBTTL 'SMG$CONTROL_MODE - Control overall mode of operation'
;  930          1002  1 GLOBAL ROUTINE SMG$CONTROL_MODE (
;  931          1003  1                                  PASTEBOARD_ID,
;  932          1004  1                                  NEW_MODE_BITS,
;  933          1005  1                                  OLD_MODE_BITS
;  934          1006  1                                ) =
;  935          1007  1 !++
;  936          1008  1 ! FUNCTIONAL DESCRIPTION:
;  937          1009  1 !
;  938          1010  1 !       This routine allows the caller to interrogate various modes
;  939          1011  1 !       of the overall operation of the SMG$ package, and to change
;  940          1012  1 !       those modes.  Each pasteboard (hence device) has its own set
;  941          1013  1 !       of mode settings.
;  942          1014  1 !
;  943          1015  1 !       The modes currently defined are:
;  944          1016  1 !
;  945          1017  1 !               1.  Use of buffers.  Under normal operation, the SMG$
;  946          1018  1 !               package will output to the terminal as soon as it has
;  947          1019  1 !               determined what needs to be written.  That is, no
;  948          1020  1 !               internal buffering will be done.  This mode is the
;  949          1021  1 !               default because it requires no follow-up effort on the
;  950          1022  1 !               part of the user to periodically force the buffer to the
;  951          1023  1 !               terminal.
;  952          1024  1 !               However, greater efficiency of $QIO usage can be
;  953          1025  1 !               achieved by enabling buffering.  In this mode, the SMG$
;  954          1026  1 !               package buffers all of its output for efficient usage of
;  955          1027  1 !               $QIO's.  The caller can at any time call
;  956          1028  1 !               SMG$FLUSH_BUFFER to force to the screen any output which
;  957          1029  1 !               has been queued to the screen, but not yet actually
;  958          1030  1 !               output.  For example, in this mode, each input operation
;  959          1031  1 !               initiated via the SMG$ facility will force a
;  960          1032  1 !               SMG$FLUSH_BUFFER call.
;  961          1033  1 !
;  962          1034  1 !               Default setting is to not provide buffering.
;  963          1035  1 !
;  964          1036  1 !               2.  Minimal screen update.  Unless explicitly told not
;  965          1037  1 !               to, the SMG$ package will try to minimize the number
;  966          1038  1 !               of characters actually sent to a terminal to increase
;  967          1039  1 !               overall system throughput.  It achieves this in part by
;  968          1040  1 !               remembering what is currently on the screen and actually
;  969          1041  1 !               outputing only the specific character sequences which
;  970          1042  1 !               will change the current contents of the screen into the
;  971          1043  1 !               desired contents of the screen.  As a result of this
;  972          1044  1 !               action, the original sequence of changes may not be
;  973          1045  1 !               preserved.  Generally, the screen is repainted so
;  974          1046  1 !               rapidly the this difference is not noticable since the
;  975          1047  1 !               resultant screen appears the way it should.  In rare
;  976          1048  1 !               circumstances, perhaps while debugging a new
;  977          1049  1 !               application, the caller may which to change this
;  978          1050  1 !               behavior to strictly preserve the order of his output.
;  979          1051  1 !
;  980          1052  1 !               The default mode is to perform minimal screen update.
;  981          1053  1 !               There is currently not much difference if you omit this
;  982          1054  1 !               bit because it really doesn't make much sense to repaint
;  983          1055  1 !               the entire screen when you change just one character,
;  984          1056  1 !               but this may change in future releases.  For now,
;  985          1057  1 !               you should always specify this bit.
```

```
;  986          1058  1 !
;  987          1059  1 !                   3. CLEAR_SCREEN. If this bit is set, then when SMG exits
;  988          1060  1 !                   normally, it will clear the screen just before returning
;  989          1061  1 !                   control to DCL.
;  990          1062  1 !
;  991          1063  1 !                   4. NOTABS.  If this bit is set, then SMG will never rely
;  992          1064  1 !                   on physical tabs even if the terminal supports them.
;  993          1065  1 !                   If not set, then SMG will feel free to use physical
;  994          1066  1 !                   tabs for the minimal update procedure.  Such use implicitly
;  995          1067  1 !                   assumes that your terminal's tab stops are set to the
;  996          1068  1 !                   DEC default locations.  Set this bit if you want to guarantee
;  997          1069  1 !                   that your product will run regardless of the tab settings
;  998          1070  1 !                   that the user has physically set on his terminal.
;  999          1071  1 !
; 1000          1072  1 !           The three parameters are optional.  First, if OLD_MODE_BITS
; 1001          1073  1 !           is supplied, it is filled in with the current mode bit settings.
; 1002          1074  1 !           Secondly, if the NEW_MODE_BITS parameter is provided, its
; 1003          1075  1 !           contents are used to set the current mode(s) of operation.
; 1004          1076  1 !
; 1005          1077  1 !           Hence this routine can typically be used in three way.
; 1006          1078  1 !           a).     To find out current settings:
; 1007          1079  1 !                   SMG$CONTROL_MODE ( PASTEBOARD_ID, , OLD_MODE_BITS)
; 1008          1080  1 !
; 1009          1081  1 !           b).     To set the bits with no regard for their current setting:
; 1010          1082  1 !                   SMG$CONTROL_MODE ( PASTEBOARD_ID, NEW_MODE_BITS)
; 1011          1083  1 !
; 1012          1084  1 !           c).     To write modular code, saving the current settings,
; 1013          1085  1 !                   setting them to your desired setting, then restoring
; 1014          1086  1 !                   original settings before exiting your procedure:
; 1015          1087  1 !                   SMG$CONTROL_MODE ( PASTEBOARD_ID,
; 1016          1088  1 !                                      NEW_MODE_BITS, saved_mode_bits )
; 1017          1089  1 !                   and before exiting,
; 1018          1090  1 !                   SMG$CONTROL_MODE ( PASTEBOARD_ID, saved_mode_bits )
; 1019          1091  1 !
; 1020          1092  1 !
; 1021          1093  1 ! CALLING SEQUENCE:
; 1022          1094  1 !
; 1023          1095  1 !           ret_status.wlc.v = SMG$CONTROL_MODE ( PASTEBOARD_ID.rl.r,
; 1024          1096  1 !                                                 [NEW_MODE_BITS.rl.r],
; 1025          1097  1 !                                                 [OLD_MODE_BITS.wl.r] )
; 1026          1098  1 !
; 1027          1099  1 ! FORMAL PARAMETERS:
; 1028          1100  1 !
; 1029          1101  1 !           PASTEBOARD_ID.rl.r      The id of the PASTEBOARD for which the
; 1030          1102  1 !                                   modes are being set or interrogated.
; 1031          1103  1 !
; 1032          1104  1 !           NEW_MODE_BITS.rl.r      [Optional].  If supplied, new mode
; 1033          1105  1 !                                   settings to be employed.
; 1034          1106  1 !                                   A bit set to 1 forces that mode to be
; 1035          1107  1 !                                   employed.  A bit set to 0 inhibits that
; 1036          1108  1 !                                   mode of operation.
; 1037          1109  1 !                                   Bit SMG$K_BUF_ENABLED controls buffering.
; 1038          1110  1 !                                   This should normally be set to improve
; 1039          1111  1 !                                   performance.
; 1040          1112  1 !                                   Bit SMG$K_MINUPD controls minimal update and
; 1041          1113  1 !                                   should normally be set.
; 1042          1114  1 !                                   Bit SMG$K_CLEAR_SCREEN causes SMG to clear
```

```
;  1043     1115  1 !                                   the screen upon normal exit.
;  1044     1116  1 !                                   Bit SMG$K_NOTABS prevents SMG from relying
;  1045     1117  1 !                                   on physical tabs.
;  1046     1118  1 !                                   All remaining bits must be zero to allow
;  1047     1119  1 !                                   for expansion in the future.
;  1048     1120  1 !
;  1049     1121  1 !       OLD_MODE_BITS.wl.r          [Optional].  If supplied, will be filled
;  1050     1122  1 !                                   in with mode settings prior to this call.
;  1051     1123  1 !                                   A bit set to 1 indicates that the mode
;  1052     1124  1 !                                   was employed.  A bit set to 0 indicates
;  1053     1125  1 !                                   that the mode was inhibited.
;  1054     1126  1 !                                   Bit SMG$K_BUF_ENABLED controls buffering.
;  1055     1127  1 !                                   Bit SMG$K_MINUPD controls minimal update.
;  1056     1128  1 !                                   Bit SMG$K_CLEAR_SCREEN causes SMG to clear
;  1057     1129  1 !                                   the screen upon normal exit.
;  1058     1130  1 !                                   Bit SMG$K_NOTABS prevents use of physical tabs.
;  1059     1131  1 !                                   All remaining bits will be zero to allow
;  1060     1132  1 !                                   for expansion in the future.
;  1061     1133  1 !
;  1062     1134  1 !
;  1063     1135  1 ! IMPLICIT INPUTS:
;  1064     1136  1 !
;  1065     1137  1 !       None
;  1066     1138  1 !
;  1067     1139  1 ! IMPLICIT OUTPUTS:
;  1068     1140  1 !
;  1069     1141  1 !       None
;  1070     1142  1 !
;  1071     1143  1 ! COMPLETION STATUS:
;  1072     1144  1 !
;  1073     1145  1 !       SS$_NORMAL          Normal successful completion
;  1074     1146  1 !       SMG$_INVARG         Invalid arguement.  Bits corresponding to
;  1075     1147  1 !                           unsupported functions supplied in NEW_MODE_BITS
;  1076     1148  1 !       SMG$_INVPAS_ID      Invalid pasteboard id.
;  1077     1149  1 !       SMG$_WRONUMARG      Wrong number of arguments.
;  1078     1150  1 !
;  1079     1151  1 ! SIDE EFFECTS:
;  1080     1152  1 !
;  1081     1153  1 !       NONE
;  1082     1154  1 !--
```

```
;  1084          1155  2     BEGIN
;  1085          1156  2     BUILTIN
;  1086          1157  2         NULLPARAMETER;
;  1087          1158  2
;  1088          1159  2     LOCAL
;  1089          1160  2         PBCB : REF BLOCK [,BYTE];       ! Address of associated
;  1090          1161  2                                         ! pasteboard control block.
;  1091          1162  2
;  1092          1163  2     $SMG$GET_PBCB ( .PASTEBOARD_ID, PBCB) ; ! Get address of PBCB
;  1093          1164  2
;  1094          1165  2
;  1095          1166  2     !+
;  1096          1167  2     ! If caller requested current settings, return them.
;  1097          1168  2     !-
;  1098          1169  2         IF NOT NULLPARAMETER (OLD_MODE_BITS)
;  1099          1170  2         THEN
;  1100          1171  2             .OLD_MODE_BITS = .PBCB [PBCB_L_MODE_SETTINGS];
;  1101          1172  2
;  1102          1173  2     !+
;  1103          1174  2     ! If caller provided new settings, process them.
;  1104          1175  2     !-
;  1105          1176  2         IF NOT NULLPARAMETER (NEW_MODE_BITS)
;  1106          1177  2         THEN
;  1107          1178  3             BEGIN   ! Caller provided new settings
;  1108          1179  3
;  1109          1180  3             BIND    NEW_MODE = .NEW_MODE_BITS       : BITVECTOR[32];
;  1110          1181  3
;  1111          1182  3             !+
;  1112          1183  3             ! First make sure that he didn't supply extraneous bits
;  1113          1184  3             !-
;001 PLL1073     1185  3             IF .(.NEW_MODE_BITS)<4,28> NEQ 0
;  1115-1        1186  3             THEN
;  1116          1187  3                 RETURN (SMG$_INVARG);       ! Unsupported bits provided
;  1117          1188  3
;  1118          1189  3             !+
;  1119          1190  3             ! If he's switching from buffered to non-buffered, flush out
;  1120          1191  3             ! current contents of buffers so we are caught up.
;  1121          1192  3             !-
;  1122          1193  3             IF .PBCB [PBCB_V_BUF_ENABLED]  AND      ! If currently enabled
;  1123          1194  3                NOT .NEW_MODE[SMG$K_BUF_ENABLED]     ! and new disabled
;  1124          1195  3             THEN
;  1125          1196  3                 SMG$$FLUSH_BUFFER ( .PBCB );
;  1126          1197  3
;  1127          1198  3             !+
;  1128          1199  3             ! If he's switching from no minimal update to minimal update,
;  1129          1200  3             ! then ensure that the screen buffer is up-to-date.
;  1130          1201  3             !-
;  1131          1202  3             IF NOT .PBCB [PBCB_V_MINUPD]  AND       ! If currently disabled
;  1132          1203  3                .NEW_MODE[SMG$K_MINUPD]              ! and new enabled
;  1133          1204  3             THEN
;  1134          1205  4                 BEGIN       ! update screen buffer
;  1135          1206  4
;  1136          1207  4                 BIND WCB = .PBCB [PBCB_A_WCB]       : BLOCK [,BYTE];
;  1137          1208  4
;  1138          1209  4                 !+
;  1139          1210  4                 ! Move the current text and attributes buffers
;  1140          1211  4                 ! to the corresponding screen buffers.
```

```
;  1141          1212  4              !-
;  1142          1213  4
;  1143          1214  4              CH$MOVE(.WCB [WCB_L_BUFSIZE], .WCB [WCB_A_TEXT_BUF],
;  1144          1215  4                      .WCB [WCB_A_SCR_TEXT_BUF]);
;  1145          1216  4
;  1146          1217  4              CH$MOVE(.WCB [WCB_L_BUFSIZE], .WCB [WCB_A_ATTR_BUF],
;  1147          1218  4                      .WCB [WCB_A_SCR_ATTR_BUF]);
;  1148          1219  4
;  1149          1220  4              !+
;  1150          1221  4              ! Move the text line characteristics vector to the screen
;  1151          1222  4              ! text line characteristics vector.
;  1152          1223  4              !-
;  1153          1224  4              CH$MOVE(.WCB [WCB_W_NO_ROWS] +1, .WCB [WCB_A_LINE_CHAR],
;  1154          1225  4                      .WCB [WCB_A_SCR_LINE_CHAR]);
;  1155          1226  4
;  1156          1227  3              END;          ! update screen buffer
;  1157          1228  3
;  1158          1229  3          !+
;  1159          1230  3          ! Finally, reset modes based on callers wishes.
;  1160          1231  3          !-
;  1161          1232  3          PBCB [PBCB_L_MODE_SETTINGS] = ..NEW_MODE_BITS;
;  1162          1233  3
;  1163          1234  2          END;     ! Caller provided new settings
;  1164          1235  2
;  1165          1236  2      RETURN (SS$_NORMAL);
;  1166          1237  2
;  1167          1238  1      END;                          ! Routine SMG$CONTROL_MODE
```

```
                                    00FC 00000          .ENTRY  SMG$CONTROL_MODE, Save P2,R3,R4,R5,R6,R7    ; 1002
                          50    04    BC   D0 00002       MOVL    @PASTEBOARD_ID, R0                          ; 1163
                                      11   19 00006       BLSS    1$                                          ;
                 00000000G   00       50   D1 00008       CMPL    R0, PBD_L_COUNT                             ;
                                      08   14 0000F       BGTR    1$                                          ;
              08 00000000G   00       50   E0 00011       BBS     R0, PBD_V_PB_AVAIL, 2$                      ;
                          50 00000000G 8F   D0 00019 1$:    MOVL    #SMG$_INVPAS_ID, R0                         ;
                                      04   00020         RET                                                 ;
                          57 00000000G0040  D0 00021 2$:    MOVL    PBD_A_PBCB[R0], PBCB                        ;
                          03            6C  91 00029       CMPB    (AP), #3                                    ; 1169
                                        0A  1F 0002C       BLSSU   3$                                          ;
                                C     AC    D5 0002E       TSTL    12(AP)                                      ;
                                      05   13 00031       BEQL    3$                                          ;
                       0C   BC        0C   A7 D0 00033    MOVL    12(PBCB), @OLD_MODE_BITS                    ; 1171
                          02            6C  91 00038 3$:    CMPB    (AP), #2                                    ; 1176
                                        53  1F 0003B       BLSSU   7$                                          ;
                                C     08    D5 0003D       TSTL    8(AP)                                       ;
                                      4E   13 00040       BEQL    7$                                          ;
00          08    BC                1C        04  ED 00042    CMPZV   #4, #28, @NEW_MODE_BITS, #0             ; 1185
                                      08   13 00048       BEQL    4$                                          ;
                          50 00000000G  8F  D0 0004A       MOVL    #SMG$_INVARG, R0                            ; 1187
                                      04   00051         RET                                                 ;
                          CD        0C    A7  E9 00052 4$:    BLBC    12(PBCB), 5$                            ; 1193
                          09        08    BC  E8 00056       BLBS    @NEW_MODE_BITS, 5$                      ; 1194
                                      57   DD 0005A       PUSHL   PBCB                                        ; 1196
```

```
                    00000000G  00          01  FB  0005C                CALLS   #1, SMG$$FLUSH_BUFFER                    ;
                            23      0C  A7  01  E0  00063  5$:           BBS     #1, 12(PBCB), 6$                         ; 1202
                            1E      08  BC  01  E1  00068                BBC     #1, @NEW_MODE_BITS, 6$                   ; 1203
                                        56  08  A7  D0  0006D            MOVL    8(PBCB), R6                              ; 1207
                        14  B6      08  B6  28  A6  28  00071            MOVC3   40(R6), @8(R6), @20(R6)                  ; 1215
                        18  B6      0C  B6  28  A6  28  00078            MOVC3   40(R6), @12(R6), @24(R6)                 ; 1218
                                        50  02  A6  3C  0007F            MOVZWL  2(R6), R0                                ; 1224
                                            50  D6  00083                INCL    R0                                       ;
                        30  B6      2C  B6      50  28  00085            MOVC3   R0, @44(R6), @48(R6)                     ; 1225
                                    0C  A7  08  BC  D0  00088  6$:       MOVL    @NEW_MODE_BITS, 12(PBCB)                 ; 1232
                                        50      01  D0  00090  7$:       MOVL    #1, R0                                   ; 1236
                                                    04  00093            RET                                              ; 1238

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

```
;1169     1239  1 %SBTTL 'SMG$INVALIDATE_DISPLAY - Mark display as being privately used'
;1170     1240  1 GLOBAL ROUTINE SMG$INVALIDATE_DISPLAY ( DISPLAY_ID) =
;1171     1241  1
;1172     1242  1 !++
;1173     1243  1 ! FUNCTIONAL DESCRIPTION:
;1174     1244  1 !
;1175     1245  1 !       This routine is called when ever a change has been completed
;1176     1246  1 !       to a given virtual display and the user had previously
;1177     1247  1 !       written into that display on his own, without using
;1178     1248  1 !       SMG routines.
;1179     1249  1 !
;1180     1250  1 !       The virtual display must not be occluded.
;1181     1251  1 !
;1182     1252  1 !       Each pasteboard to which this display is pasted is isolated and
;1183     1253  1 !       its window image must be redrawn.
;1184     1254  1 !
;1185     1255  1 ! CALLING SEQUENCE:
;1186     1256  1 !
;1187     1257  1 !       ret_status.wlc.v = SMG$INVALIDATE_DISPLAY ( DISPLAY_ID.rl.r)
;1188     1258  1 !
;1189     1259  1 ! FORMAL PARAMETERS:
;1190     1260  1 !
;1191     1261  1 !       DISPLAY_ID.rl.r         Display ID of virtual display.
;1192     1262  1 !
;1193     1263  1 ! IMPLICIT INPUTS:
;1194     1264  1 !
;1195     1265  1 !       NONE
;1196     1266  1 !
;1197     1267  1 ! IMPLICIT OUTPUTS:
;1198     1268  1 !
;1199     1269  1 !       NONE
;1200     1270  1 !
;1201     1271  1 ! COMPLETION STATUS:
;1202     1272  1 !
;1203     1273  1 !       SS$_NORMAL      Normal successful completion
;1204     1274  1 !
;1205     1275  1 ! SIDE EFFECTS:
;1206     1276  1 !
;1207     1277  1 !       NONE
;1208     1278  1 !--
```

```
;  1210          1279  2 BEGIN
;  1211          1280  2
;  1212          1281  2 EXTERNAL ROUTINE
;  1213          1282  2
;  1214          1283  2         SMG$$INVALIDATE_DISPLAY;
;  1215          1284  2
;  1216          1285  2 LOCAL
;  1217          1286  2
;  1218          1287  2         STATUS; ! Status of subroutine calls
;  1219          1288  2
;  1220          1289  2 !+
;  1221          1290  2 ! Check for right number of arguments.
;  1222          1291  2 !-
;  1223          1292  2
;  1224          1293  2 $SMG$VALIDATE_ARGCOUNT ( 1, 1);
;  1225          1294  2
;  1226          1295  2 !+
;  1227          1296  2 ! Call the internal routine.
;  1228          1297  2 !-
;  1229          1298  2
;  1230          1299  2 RETURN  SMG$$INVALIDATE_DISPLAY(.DISPLAY_ID)
;  1231          1300  2
;  1232          1301  1 END;              ! End of routine SMG$INVALIDATE_DISPLAY


                                  0000 00000          .ENTRY  SMG$INVALIDATE_DISPLAY, Save nothing          ; 1240
                           01        6C   91 00002     CMPB    (AP), #1                                      ; 1293
                                     08   13 00005     BEQL    1$
                           50 00000000G  8F   D0 00007     MOVL    #SMG$_WRONUMARG, R0
                                         04 0000E     RET
                              04         AC   DD 0000F 1$:  PUSHL   DISPLAY_ID                            ; 1299
                  00000000G   00         01   FB 00012     CALLS   #1, SMG$$INVALIDATE_DISPLAY
                                         04 00019     RET                                                   ; 1301

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

```
;  1234     1302  1 %SBTTL 'SMG$GET_CHAR_AT_PHYSICAL_CURSOR - Get character at physical cursor'
;  1235     1303  1 GLOBAL ROUTINE SMG$GET_CHAR_AT_PHYSICAL_CURSOR (
;  1236     1304  1                                                 PASTEBOARD_ID,
;  1237     1305  1                                                 CHARACTER : REF VECTOR [,BYTE]
;  1238     1306  1                                                 ) =
;  1239     1307  1
;  1240     1308  1 !++
;  1241     1309  1 ! FUNCTIONAL DESCRIPTION
;  1242     1310  1 !
;  1243     1311  1 ! Returns the character that occupies the position on the screen corresponding
;  1244     1312  1 ! to the current physical cursor position.
;  1245     1313  1 !
;  1246     1314  1 ! Note:  If the SMG$ facility has not written to that location on the screen,
;  1247     1315  1 ! then it doesn't know its contents.  A character encoded as X'FF' will be
;  1248     1316  1 ! returned in that case.
;  1249     1317  1 !
;  1250     1318  1 ! If character returned has a value less than X'20', then it is not an actual
;  1251     1319  1 ! printable character, but an internal terminal-independent code of what should
;  1252     1320  1 ! be displayed at that position, e.g. an element of the line-drawing character
;  1253     1321  1 ! set.  Do not attempt to use this code for subsequent output operations.
;  1254     1322  1 !
;  1255     1323  1 !
;  1256     1324  1 ! CALLING SEQUENCE
;  1257     1325  1 !
;  1258     1326  1 !       status.wlc.v = SMG$GET_CHAR_AT_PHYSICAL_CURSOR (
;  1259     1327  1 !                                  PASTEBOARD_ID.rl.r,
;  1260     1328  1 !                                  CHARACTER.wbu.r)
;  1261     1329  1 !
;  1262     1330  1 ! FORMAL PARAMETERS
;  1263     1331  1 !
;  1264     1332  1 !       PASTEBOARD_ID.rl.r      Address of a longword containing the
;  1265     1333  1 !                               pasteboard_id for which the information is
;  1266     1334  1 !                               desired.
;  1267     1335  1 !
;  1268     1336  1 !       CHARACTER.wbu.r         Address of a byte to receive the result.
;  1269     1337  1 !
;  1270     1338  1 ! IMPLICIT INPUTS:
;  1271     1339  1 !
;  1272     1340  1 !       NONE
;  1273     1341  1 !
;  1274     1342  1 ! IMPLCIT OUTPUTS:
;  1275     1343  1 !
;  1276     1344  1 !       NONE
;  1277     1345  1 !
;  1278     1346  1 ! COMPLETION STATUS:
;  1279     1347  1 !
;  1280     1348  1 !       SS$NORMAL       Normal successful completion
;  1281     1349  1 !       SMG$_INVPAS_ID  Invalid pasteboard id
;  1282     1350  1 !       SMG$_WRONUMARG  Wrong number of arguments
;  1283     1351  1 !
;  1284     1352  1 ! SIDE EFFECTS:
;  1285     1353  1 !
;  1286     1354  1 !       NONE
;  1287     1355  1 !--
;  1288     1356  2     BEGIN
;  1289     1357  2     LOCAL
;  1290     1358  2         PBCB : REF $PBCB_DECL,          ! Address of pasteboard control block
```

```
;  1291     1359  2          WCB  : REF $WCB_DECL,                ! Address of window control block
;  1292     1360  2          PTR  : REF VECTOR [,BYTE];           ! Pointer to text buffer of what is
;  1293     1361  2                                               ! currently on the screen.
;  1294     1362  2
;  1295     1363  2      !+
;  1296     1364  2      ! Check for right number of arguments.
;  1297     1365  2      !-
;  1298     1366  2          $SMG$VALIDATE_ARGCOUNT (2, 2);
;  1299     1367  2
;  1300     1368  2      !+
;  1301     1369  2      ! Get address of pasteboard control block and of window control block.
;  1302     1370  2      !-
;  1303     1371  2          $SMG$GET_PBCB (.PASTEBOARD_ID, PBCB);
;  1304     1372  2          WCB = .PBCB [PBCB_A_WCB];
;  1305     1373  2          PTR = .WCB [WCB_A_SCR_TEXT_BUF];
;  1306     1374  2
;  1307     1375  2      !+
;  1308     1376  2      ! Fetch character at position corresponding to current actual physical cursor
;  1309     1377  2      ! position on the screen.
;  1310     1378  2      !-
;  1311     1379  2          CHARACTER [0] = .PTR [
;  1312     1380  2                              (.WCB [WCB_W_OLD_CUR_ROW] -1) * .WCB [WCB_W_NO_COLS] +
;  1313     1381  2                              (.WCB [WCB_W_OLD_CUR_COL] -1) ];
;  1314     1382  2
;  1315     1383  2          RETURN (SS$_NORMAL);
;  1316     1384  1          END;        ! Routine SMG$GET_CHAR_AT_PHYSICAL_CURSOR



                                 000C 00000          .ENTRY  SMG$GET_CHAR_AT_PHYSICAL_CURSOR, Save R2,R3 ;  1303
                          02      6C 91 00002        CMPB    (AP), #2                                    ;  1366
                                  08 13 00005        BEQL    1$
                    50 00000000G  8F DD 00007        MOVL    #SMG$_WRONUMARG, R0
                                     04 0000E        RET
                    50        04  BC DD 0000F  1$:   MOVL    @PASTEBOARD_ID, R0                          ;  1371
                                  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 DD 00026  2$:   MOVL    #SMG$_INVPAS_ID, R0
                                     04 0002D        RET
                    50 00000000G0040 DD 0002E  3$:   MOVL    PBD_A_PBCB[R0], PBCB
                    50        08  A0 DD 00036        MOVL    8(PBCB), WCB                                ;  1372
                    52        14  A0 DD 0003A        MOVL    20(WCB), PTR                                ;  1373
                    51        24  A0 32 0003E        CVTWL   36(WCB), R1                                 ;  1380
                                  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                                 ;  1381
                    51            50 C0 0004F        ADDL2   R0, R1
                 08 BC        FF A142 90 00052        MOVB    -1(R1)[PTR], @CHARACTER                     ;  1379
                    50            01 DD 00058        MOVL    #1, R0                                      ;  1383
                                     04 0005B        RET                                                 ;  1384

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

```
;  1318     1385  1 %SBTTL 'SMG$REPAINT_SCREEN - Repaint current screen'
;  1319     1386  1 GLOBAL ROUTINE SMG$REPAINT_SCREEN (
;  1320     1387  1                                   PASTEBOARD_ID
;  1321     1388  1                                     )=
;  1322     1389  1 !++
;  1323     1390  1 ! FUNCTIONAL DESCRIPTION:
;  1324     1391  1 !
;  1325     1392  1 !       This procedure will repaint the current screen based on its
;  1326     1393  1 !       internal knowledge of what the screen should look like.
;  1327     1394  1 !       It is intended to be called when it is known or suspected that
;  1328     1395  1 !       some outside agent has disrupted the screen so that it no longer
;  1329     1396  1 !       matches the internal knowledge of what should be on the screen.
;  1330     1397  1 !       If pasteboard batching is in effect, the repaint occurs
;  1331     1398  1 !       from the screen image; otherwise it occurs from the text image.
;  1332     1399  1 !
;  1333     1400  1 ! CALLING SEQUENCE:
;  1334     1401  1 !
;  1335     1402  1 !       ret_status.wlc.v = SMG$REPAINT_SCREEN ( PASTEBOARD_ID.rl.r )
;  1336     1403  1 !
;  1337     1404  1 ! FORMAL PARAMETERS:
;  1338     1405  1 !
;  1339     1406  1 !       PASTEBOARD_ID.rl.r      Id of pasteboard associated with
;  1340     1407  1 !                               physical screen to be repainted.
;  1341     1408  1 !
;  1342     1409  1 ! IMPLICIT INPUTS:
;  1343     1410  1 !
;  1344     1411  1 !       Current internal representation of what should be on the screen.
;  1345     1412  1 !
;  1346     1413  1 ! IMPLICIT OUTPUTS:
;  1347     1414  1 !
;  1348     1415  1 !       NONE
;  1349     1416  1 !
;  1350     1417  1 ! COMPLETION STATUS:
;  1351     1418  1 !
;  1352     1419  1 !       SS$_NORMAL      Normal successful completion
;  1353     1420  1 !       SMG$_INVPAS_ID  Invalid pasteboard control block
;  1354     1421  1 !
;  1355     1422  1 ! SIDE EFFECTS:
;  1356     1423  1 !
;  1357     1424  1 !       NONE
;  1358     1425  1 !--
;  1359     1426  2     BEGIN
;  1360     1427  2     LOCAL
;  1361     1428  2         WCB     : REF BLOCK [,BYTE],    ! Window control block
;  1362     1429  2         PBCB    : REF BLOCK [,BYTE];    ! Address of pasteboard control
;  1363     1430  2                                         ! block
;  1364     1431  2
;  1365     1432  2     $SMG$GET_PBCB ( .PASTEBOARD_ID, PBCB); ! Get address of pasteboard
;  1366     1433  2                                         ! control block.
;  1367     1434  2
;  1368     1435  2     !+
;  1369     1436  2     ! Get address of window control block.
;  1370     1437  2     !-
;  1371     1438  2
;  1372     1439  2     WCB = .PBCB [PBCB_A_WCB];
;  1373     1440  2
;  1374     1441  2     !+
```

```
;  1375          1442  2     ! Zero out the screen image, thereby forcing output to
;  1376          1443  2     ! redraw the whole screen.
;  1377          1444  2     !-
;  1378          1445  2
;  1379          1446  2     ! ***************** We have to add code here to make
;  1380          1447  2     ! the repaint occur from the window buffer if batching is in effect.
;  1381          1448  2     ! ****************************
;  1382          1449  2
;  1383          1450  2     CH$FILL(0,.WCB [WCB_L_BUFSIZE],.WCB [WCB_A_SCR_TEXT_BUF]);
;  1384          1451  2
;  1385          1452  2     !+
;  1386          1453  2     ! Clear the line characteristics vector.
;  1387          1454  2     !-
;  1388          1455  2
;  1389          1456  2     CH$FILL(0,.WCB [WCB_W_NO_COLS]+1,.WCB [WCB_A_SCR_LINE_CHAR]);
;  1390          1457  2
;  1391          1458  2 !+
;  1392          1459  2 ! Tell the output routines that we've changed the whole WCB buffer.
;  1393          1460  2 !-
;  1394          1461  2     PBCB [PBCB_W_FIRST_CHANGED_ROW] = 1;
;  1395          1462  2     PBCB [PBCB_W_LAST_CHANGED_ROW]  = .PBCB [PBCB_B_ROWS];
;  1396          1463  2     PBCB [PBCB_W_FIRST_CHANGED_COL] = 1;
;  1397          1464  2     PBCB [PBCB_W_LAST_CHANGED_COL]  = .PBCB [PBCB_W_WIDTH];
;  1398          1465  2
;  1399          1466  2     RETURN (SMG$$MIN_UPD (.PBCB));
;  1400          1467  1     END;                    ! End of procedure SMG$REPAINT_SCREEN



                                                00FC 00000          .ENTRY  SMG$REPAINT_SCREEN, Save R2,R3,R4,R5,R6,R7  ;  1386
                                        50      04   BC   D0 00002  MOVL   @PASTEBOARD_ID, R0                          ;  1432
                                                     11   19 00006  BLSS   1$
                                    00000000G   00       50   D1 00008  CMPL   R0, PBD_L_COUNT
                                                         08   14 0000F  BGTR   1$
                                08  00000000G   00       50   E0 00011  BBS    R0, PBD_V_PB_AVAIL, 2$
                                                50 00000000G   8F   D0 00019  1$:  MOVL   #SMG$_INVPAS_ID, R0
                                                               04   00020  RET
                                                57 00000000G0040   D0 00021  2$:  MOVL   PBD_A_PBCB[R0], PBCB
                                                56       08    A7   D0 00029  MOVL   8(PBCB), WCB                       ;  1439
    28  A6                          00          6E             00   2C 0002D  MOVC5  #0, (SP), #0, 40(WCB), @20(WCB)    ;  1450
                                                           14   B6   00033
                                                50       06    A6   3C 00035  MOVZWL 6(WCB), R0                       ;  1456
                                                               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)                     ;  1461
                                        00AA    C7       5F    A7   9B 00047  MOVZBW 95(PBCB), 170(PBCB)              ;  1462
                                        00AC    C7             01   B0 0004D  MOVW   #1, 172(PBCB)                    ;  1463
                                        00AE    C7       5A    A7   B0 00052  MOVW   90(PBCB), 174(PBCB)              ;  1464
                                                               57   DD 00058  PUSHL  PBCB                             ;  1466
                                    00000000G   00             01   FB 0005A  CALLS  #1, SMG$$MIN_UPD
                                                               04   00061  RET                                       ;  1467

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

```
; 1401          1468  1 !<BLF/PAGE>
```

```
;  1403     1469  1 %SBTTL 'SMG$RING_BELL - Ring Bell'
;  1404     1470  1 GLOBAL ROUTINE SMG$RING_BELL (
;  1405     1471  1                                 DISPLAY_ID,
;  1406     1472  1                                 TIMES
;  1407     1473  1                               ) =
;  1408     1474  1 !++
;  1409     1475  1 ! FUNCTIONAL DESCRIPTION:
;  1410     1476  1 !
;  1411     1477  1 !       This routine rings the bell, on each physical terminal to which
;  1412     1478  1 !       the given virtual display is pasted, the number of times
;  1413     1479  1 !       specified.  If TIMES omitted, 1 is used.
;  1414     1480  1 !
;  1415     1481  1 ! CALLING SEQUENCE:
;  1416     1482  1 !
;  1417     1483  1 !       ret_status.wlc.v = SMG$RING_BELL (
;  1418     1484  1 !                                         DISPLAY_ID.rl.r,
;  1419     1485  1 !                                         [TIMES.rl.r] )
;  1420     1486  1 !
;  1421     1487  1 ! FORMAL PARAMETERS:
;  1422     1488  1 !
;  1423     1489  1 !       DISPLAY_ID.rl.r The display id of the virtual display.
;  1424     1490  1 !
;  1425     1491  1 !       TIMES.rl.r      Optional.  The number of times to ring the bell.
;  1426     1492  1 !                       If not specified, the bell is sounded once.
;  1427     1493  1 !
;  1428     1494  1 ! IMPLICIT INPUTS:
;  1429     1495  1 !
;  1430     1496  1 !       NONE
;  1431     1497  1 !
;  1432     1498  1 ! IMPLICIT OUTPUTS:
;  1433     1499  1 !
;  1434     1500  1 !       NONE
;  1435     1501  1 !
;  1436     1502  1 ! COMPLETION STATUS:
;  1437     1503  1 !
;  1438     1504  1 !       SS$_NORMAL      Normal successful completion
;  1439     1505  1 !       Statuses returned by SMG$$OUTPUT or
;  1440     1506  1 !       statuses or signal values returned by STR$DUPL_CHAR.
;  1441     1507  1 !
;  1442     1508  1 ! SIDE EFFECTS:
;  1443     1509  1 !
;  1444     1510  1 !       NONE
;  1445     1511  1 !--
;  1446     1512  1
;  1447     1513  2     BEGIN
;  1448     1514  2     BUILTIN
;  1449     1515  2         NULLPARAMETER ;
;  1450     1516  2
;  1451     1517  2     BIND
;  1452     1518  2         BELL_CHAR = UPLIT (BYTE (BELL));
;  1453     1519  2
;  1454     1520  2     LOCAL
;  1455     1521  2         DCB : REF BLOCK [,BYTE],        ! Address of display control
;  1456     1522  2                                         ! block
;  1457     1523  2
;  1458     1524  2         PP  : REF BLOCK [,BYTE],        ! Address of a pasting packet
;  1459     1525  2
```

```
;  1460     1526  2          STATUS;                         ! Status to return to caller
;  1461     1527  2
;  1462     1528  2      $SMG$VALIDATE_ARGCOUNT (1, 2);
;  1463     1529  2
;  1464     1530  2      $SMG$GET_DCB (.DISPLAY_ID, DCB);    ! Get addr of virtual display
;  1465     1531  2                                          ! control block.
;  1466     1532  2
;  1467     1533  2      STATUS=SS$_NORMAL;
;  1468     1534  2
;  1469     1535  2      IF NULLPARAMETER (TIMES)
;  1470     1536  2      THEN
;  1471     1537  3          BEGIN   ! No arg supplied, output 1 bell
;  1472     1538  3          !+
;  1473     1539  3          ! Chase the chain of pasteboards to which this virtual display
;  1474     1540  3          ! is pasted.  For each pasteboard located, output the bell.
;  1475     1541  3          !-
;  1476     1542  3          PP = .DCB [DCB_A_PP_NEXT];
;  1477     1543  3
;  1478     1544  3          IF .PP EQL 0
;  1479     1545  3          THEN
;  1480     1546  3              RETURN (SMG$_FATERRLIB);            ! should never be 0
;  1481     1547  3
;  1482     1548  3          WHILE .PP NEQ DCB [DCB_A_PP_NEXT]       ! While any remain...
;  1483     1549  3          DO
;  1484     1550  4              BEGIN
;  1485     1551  4              STATUS=SMG$$OUTPUT ( .PP [PP_A_PBCB_ADDR],
;  1486     1552  4                                   1,
;  1487     1553  4                                   BELL_CHAR);
;  1488     1554  4
;  1489     1555  4              PP = .PP [PP_A_NEXT_DCB];   ! Step to next packet
;  1490     1556  3              END;
;  1491     1557  3          END     ! No arg supplied, output 1 bell
;  1492     1558  3
;  1493     1559  2      ELSE
;  1494     1560  2
;  1495     1561  3          BEGIN   ! Multiple bells
;  1496     1562  3          LOCAL
;  1497     1563  3              DESCR : BLOCK [8, BYTE] ;   ! A local descriptor
;  1498     1564  3
;  1499     1565  3          !+
;  1500     1566  3          ! Must build a string of TIMES bell characters.  Use dynamic
;  1501     1567  3          ! string.
;  1502     1568  3          !-
;  1503     1569  3          DESCR [DSC$W_LENGTH] = 0 ;
;  1504     1570  3          DESCR [DSC$B_CLASS]  = DSC$K_CLASS_D ;
;  1505     1571  3          DESCR [DSC$B_DTYPE]  = DSC$K_DTYPE_T ;
;  1506     1572  3          DESCR [DSC$A_POINTER] = 0 ;
;  1507     1573  3
;  1508     1574  3          LIB$ESTABLISH ( LIB$SIG_TO_RET ) ; ! Establish handler in case
;  1509     1575  3                                             ! STR$DUPL_CHAR signals out
;  1510     1576  3                                             ! from under us.
;  1511     1577  3
;  1512     1578  4          IF NOT (STATUS = STR$DUPL_CHAR ( DESCR, .TIMES, BELL_CHAR))
;  1513     1579  3          THEN
;  1514     1580  3              RETURN (.STATUS );
;  1515     1581  3
;  1516     1582  3          !+
```

```
;  1517          1583  3          ! Chase the chain of pasteboards to which this virtual display
;  1518          1584  3          ! is pasted.  For each pasteboard located, output the bell(s).
;  1519          1585  3          !-
;  1520          1586  3          PP = .DCB [DCB_A_PP_NEXT];
;  1521          1587  3
;  1522          1588  3          IF .PP EQL 0
;  1523          1589  3          THEN
;  1524          1590  3              RETURN (SMG$_FATERRLIB);            ! should never be 0
;  1525          1591  3
;  1526          1592  3          WHILE .PP NEQ DCB [DCB_A_PP_NEXT]       ! While any remain...
;  1527          1593  3          DO
;  1528          1594  4              BEGIN
;  1529          1595  4              STATUS=SMG$$OUTPUT ( .PP [PP_A_PBCB_ADDR],
;  1530          1596  4                                   .DESCR [DSC$W_LENGTH],
;  1531          1597  4                                   .DESCR [DSC$A_POINTER]);
;  1532          1598  4
;  1533          1599  4              PP = .PP [PP_A_NEXT_DCB];   ! Step to next packet
;  1534          1600  3              END;
;  1535          1601  3
;  1536          1602  3          LIB$SFREE1_DD ( DESCR ) ;       ! Return dynamic string
;  1537          1603  2          END ;   ! Multiple bells
;  1538          1604  2
;  1539          1605  2      RETURN (.STATUS);
;  1540          1606  1      END;                                    ! End of routine SMG$RING_BELL


                                    07  003BC P.AAA:  .BYTE   7                                                  ;

                                              BELL_CHAR=        P.AAA


                                        003C 00000          .ENTRY  SMG$RING_BELL, Save R2,R3,R4,R5             ; 1470
                 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                              ; 1528
                 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                             ; 1530
           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$:      MOVL    @DISPLAY_ID, DCB                            ;
                 54             01    D0 0003A          MOVL    #1, STATUS                                  ; 1533
                 02             6C    91 0003D          CMPB    (AP), #2                                    ; 1535
                                05    1F 00040          BLSSU   4$                                          ;
                          08    AC    D5 00042          TSTL    8(AP)                                       ;
                                22    12 00045          BNEQ    6$                                          ;
                 52       20    A3    D0 00047 4$:      MOVL    32(DCB), PP                                 ; 1542
                                50    13 0004B          BEQL    7$                                          ; 1544
                 50       20    A3    9E 0004D 5$:      MOVAB   32(DCB), R0                                 ; 1548
                 50             52    D1 00051          CMPL    PP, R0                                      ;
```

```
                                    76  13 00054          BEQL    10$                                             ;
                                A6  AF  9F 00056          PUSHAB  BELL_CHAR                                       ; 1551
                                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                                        ; 1555
                                    E4  11 00067          BRB     5$                                              ; 1548
                            6E  020E0000    8F  D0 00069  6$:     MOVL    #34471936, DESCR                        ; 1569
                                04  AE  D4 00070          CLRL    DESCR+4                                         ; 1572
                                00000000G   00  9F 00073  PUSHAB  LIB$SIG_TO_RET                                  ; 1574
                00000000G   00              01  FB 00079  CALLS   #1, LIB$ESTABLISH                               ;
                                FF7B    CF  9F 00080      PUSHAB  BELL_CHAR                                       ; 1578
                                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                                     ; 1586
                                08  12 0009B             BNEQ    8$                                              ; 1588
                            50  00000000G   8F  D0 0009D  7$:     MOVL    #SMG$_FATERRLIB, R0                     ; 1590
                                    04 000A4              RET                                                     ;
                            50  20  A3  9E 000A5  8$:     MOVAB   32(DCB), R0                                     ; 1592
                            50      52  D1 000A9          CMPL    PP, R0                                          ;
                                    15  13 000AC          BEQL    9$                                              ;
                                04  AE  DD 000AE          PUSHL   DESCR+4                                         ; 1597
                            7E      04  AE  3C 000B1      MOVZWL  DESCR, -(SP)                                    ; 1596
                                14  A2  DD 000B5          PUSHL   20(PP)                                          ; 1595
                            65      03  FB 000B8          CALLS   #3, SMG$$OUTPUT                                 ;
                            54      50  D0 000BB          MOVL    R0, STATUS                                      ;
                            52      62  D0 000BE          MOVL    (PP), PP                                        ; 1599
                                    E2  11 000C1          BRB     8$                                              ; 1592
                                    5E  DD 000C3  9$:     PUSHL   SP                                              ; 1602
                00000000G   00          01  FB 000C5     CALLS   #1, LIB$SFREE1_DD                               ;
                            50          54  D0 000CC  10$:     MOVL    STATUS, R0                                  ; 1605
                                        04 000CF              RET                                                 ; 1606

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

; 1541          1607  1 !<BLF/PAGE>
```

```
;  1543          1608  1 %SBTTL 'SMG$$BEGIN_PASTEBOARD_UPDATE_R1 - Begin batch of updates to pasteboard'
;  1544          1609  1 GLOBAL ROUTINE SMG$$BEGIN_PASTEBOARD_UPDATE_R1 (PBCB: REF $PBCB_DECL)
;  1545          1610  1                                 : SMG$$BEGIN_PBD_UPDATE$LNK =
;  1546          1611  1 !++
;  1547          1612  1 ! FUNCTIONAL DESCRIPTION:
;  1548          1613  1 !
;  1549          1614  1 !       Inner routine to support BEGIN_PASTEBOARD_UPDATE action.
;  1550          1615  1 !       Disable outputing this pasteboard to the screen until the
;  1551          1616  1 !       matching call to SMG$$END_PASTEBOARD_UPDATE_R2 is encountered.
;  1552          1617  1 !
;  1553          1618  1 ! CALLING SEQUENCE:
;  1554          1619  1 !
;  1555          1620  1 !       ret_status.wlc.v = SMG$$BEGIN_PASTEBOARD_UPDATE_R1 ( PBCB.rab.r)
;  1556          1621  1 !
;  1557          1622  1 ! FORMAL PARAMETERS:
;  1558          1623  1 !
;  1559          1624  1 !       PBCB.rab.r              Address of pasteboard control block
;  1560          1625  1 !
;  1561          1626  1 ! IMPLICIT INPUTS:
;  1562          1627  1 !
;  1563          1628  1 !       NONE
;  1564          1629  1 !
;  1565          1630  1 ! IMPLICIT OUTPUTS:
;  1566          1631  1 !
;  1567          1632  1 !       NONE
;  1568          1633  1 !
;  1569          1634  1 ! COMPLETION STATUS:
;  1570          1635  1 !
;  1571          1636  1 !       SS$_NORMAL      Normal successful completion, batching has been
;  1572          1637  1 !                       initiated
;  1573          1638  1 !       SMG$_BATWAS_ON  Success, but note that batching was already on
;  1574          1639  1 !
;  1575          1640  1 ! SIDE EFFECTS:
;  1576          1641  1 !
;  1577          1642  1 !       NONE
;  1578          1643  1 !--
;  1579          1644  1
;  1580          1645  2     BEGIN
;  1581          1646  2
;  1582          1647  2 !+
;  1583          1648  2 ! Increment count of number of SMG$$END_PASTEBOARD_UPDATE_R2 calls we need to
;  1584          1649  2 ! see before we resume outputting from this pasteboard.
;  1585          1650  2 ! Let the user know what the previous state of batching was by
;  1586          1651  2 ! our status return (in case he's interested).
;  1587          1652  2 !-
;  1588          1653  2     PBCB [PBCB_L_BATCH_LEVEL] = .PBCB [PBCB_L_BATCH_LEVEL] + 1;
;  1589          1654  2
;  1590          1655  2     IF .PBCB [PBCB_L_BATCH_LEVEL] EQL 1
;  1591          1656  2     THEN
;  1592          1657  2         RETURN  SS$_NORMAL
;  1593          1658  2     ELSE
;  1594          1659  2         RETURN  SMG$_BATWAS_ON
;  1595          1660  2
;  1596          1661  1     END;                    ! End of routine SMG$$BEGIN_PASTEBOARD_UPDATE_R1
```

```
                                    00A4    C0  D6 00000 SMG$$BEGIN_PASTEBOARD_UPDATE_R1::
                                                      INCL    164(PBCB)                          ; 1653
                                01  00A4    C0  D1 00004                                       ;
                                            04  12 00009      BNEQ    1$                         ; 1655
                                50              01  D0 0000B      MOVL    #1, R0                 ;
                                                05  0000E      RSB                                ; 1659
                                50 00000000G    8F  D0 0000F 1$:    MOVL    #SMG$_BATWAS_ON, R0
                                                05  00016      RSB                                ; 1661

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

; 1597          1662  1 !<BLF/PAGE>
```

```
;  1599          1663  1 %SBTTL 'SMG$$END_PASTEBOARD_UPDATE_R2 - End batch of updates to pasteboard'
;  1600          1664  1 GLOBAL ROUTINE SMG$$END_PASTEBOARD_UPDATE_R2 ( PBCB: REF $PBCB_DECL)
;  1601          1665  1                       : SMG$$END_PBD_UPDATE$LNK =
;  1602          1666  1 !++
;  1603          1667  1 ! FUNCTIONAL DESCRIPTION:
;  1604          1668  1 !
;  1605          1669  1 !       Inner routine to support BEGIN_PASTEBOARD_UPDATE action.
;  1606          1670  1 !       Reduce the number of calls to SMG$$END_PASTEBOARD_UPDATE_R2 that will
;  1607          1671  1 !       be needed before we resume outputting from this pasteboard.  If
;  1608          1672  1 !       this call makes this count go to zero, flush the pasteboard to
;  1609          1673  1 !       the screen.
;  1610          1674  1 !       If the level is already 0, we return a success status
;  1611          1675  1 !       informing caller that the batching level was already 0,
;  1612          1676  1 !       but we otherwise allow this.  This gives a user a guaranteed
;  1613          1677  1 !       method of ending batching.
;  1614          1678  1 !
;  1615          1679  1 ! CALLING SEQUENCE:
;  1616          1680  1 !
;  1617          1681  1 !       ret_status.wlc.v = SMG$$END_PASTEBOARD_UPDATE_R2 ( PBCB.rab.r)
;  1618          1682  1 !
;  1619          1683  1 ! FORMAL PARAMETERS:
;  1620          1684  1 !
;  1621          1685  1 !       PBCB.rab.r              Address of pasteboard control block
;  1622          1686  1 !
;  1623          1687  1 ! IMPLICIT INPUTS:
;  1624          1688  1 !
;  1625          1689  1 !       PBCB [PBCB_L_BATCH_LEVEL]
;  1626          1690  1 !
;  1627          1691  1 ! IMPLICIT OUTPUTS:
;  1628          1692  1 !
;  1629          1693  1 !       PBCB [PBCB_L_BATCH_LEVEL] gets decremented
;  1630          1694  1 !
;  1631          1695  1 ! COMPLETION STATUS:
;  1632          1696  1 !
;  1633          1697  1 !       SS$_NORMAL      Normal successful completion
;  1634          1698  1 !       SMG$_BATSTIPRO  Success; but batching is still in progress.
;  1635          1699  1 !       SMG$_BATWASOFF  Success; but batching was already off
;  1636          1700  1 !
;  1637          1701  1 ! SIDE EFFECTS:
;  1638          1702  1 !
;  1639          1703  1 !       NONE
;  1640          1704  1 !--
;  1641          1705  1
;  1642          1706  2     BEGIN
;  1643          1707  2     LOCAL
;  1644          1708  2         STATUS;                             ! Status to return
;  1645          1709  2
;  1646          1710  2 !+
;  1647          1711  2 ! If current count is greater than 1, simply reduce it by one and
;  1648          1712  2 ! return to caller.  If it is one, reduce it to zero and cause the
;  1649          1713  2 ! current contents of this pasteboard to be flushed to the screen (if need
;  1650          1714  2 ! be).
;  1651          1715  2 !-
;  1652          1716  2     IF .PBCB [PBCB_L_BATCH_LEVEL] GTRU 1
;  1653          1717  2     THEN
;  1654          1718  3         BEGIN
;  1655          1719  3         PBCB [PBCB_L_BATCH_LEVEL] = .PBCB [PBCB_L_BATCH_LEVEL] - 1;
```

```
;  1656          1720  3          STATUS = SMG$_BATSTIPRO         ! Ok, but batching is still in progress
;  1657          1721  3          END
;  1658          1722  2      ELSE
;  1659          1723  3          BEGIN    ! Being reduced to zero
;  1660          1724  3          LOCAL BATCH_STATUS;
;  1661          1725  3          BATCH_STATUS = SS$_NORMAL;
;  1662          1726  3          IF .PBCB [PBCB_L_BATCH_LEVEL] EQL 0
;  1663          1727  3          THEN
;  1664          1728  3              BATCH_STATUS = SMG$_BATWASOFF       ! Ok, but batching was already off
;  1665          1729  3          ELSE
;  1666          1730  3              PBCB [PBCB_L_BATCH_LEVEL] = 0;
;  1667          1731  3          !+
;  1668          1732  3          ! Call SMG$$CHECK_FOR_OUTPUT_PBCB to cause the
;  1669          1733  3          ! contents of this pasteboard to be
;  1670          1734  3          ! flushed to the screen.
;  1671          1735  3          ! It may not actually get there if buffering is in effect.
;  1672          1736  3          ! If that fails, then return its status as our status.
;  1673          1737  3          ! If that succeeded then return our previously calculated status.
;  1674          1738  3          !-
;  1675          1739  3          STATUS = SMG$$CHECK_FOR_OUTPUT_PBCB (.PBCB);
;  1676          1740  3          IF .STATUS THEN STATUS=.BATCH_STATUS
;  1677          1741  3          END;     ! Being reduced to zero
;  1678          1742  3
;  1679          1743  2      RETURN      .STATUS
;  1680          1744  1      END;                    ! End of routine SMG$$END_PASTEBOARD_UPDATE_R2


                                5E            04   C2 00000 SMG$$END_PASTEBOARD_UPDATE_R2::
                                                             SUBL2   #4, SP                          ; 1664
                                51      00A4  C0   9E 00003         MOVAB   164(PBCB), R1           ; 1716
                                01            61   D1 00008         CMPL    (R1), #1
                                              0B   1B 0000B         BLEQU   1$
                                              61   D7 0000D         DECL    (R1)                    ; 1719
                                52 00000000G  8F   D0 0000F         MOVL    #SMG$_BATSTIPRO, STATUS ; 1720
                                              22   11 00016         BRB     4$
                                6E            01   D0 00018 1$:    MOVL    #1, BATCH_STATUS        ; 1725
                                              61   D5 0001B         TSTL    (R1)                    ; 1726
                                              09   12 0001D         BNEQ    2$
                                6E 00000000G  8F   D0 0001F         MOVL    #SMG$_BATWASOFF, BATCH_STATUS ; 1728
                                              02   11 00026         BRB     3$
                                              61   D4 00028 2$:    CLRL    (R1)                    ; 1730
                                              50   DD 0002A 3$:    PUSHL   PBCB                    ; 1738
                      0000V     CF                 01   FB 0002C   CALLS   #1, SMG$$CHECK_FOR_OUTPUT_PBCB
                                52                 50   D0 00031   MOVL    R0, STATUS
                                03                 52   E9 00034   BLBC    STATUS, 4$              ; 1740
                                52                 6E   D0 00037   MOVL    BATCH_STATUS, STATUS
                                50                 52   D0 0003A 4$:    MOVL    STATUS, R0          ; 1743
                                5E                 04   C0 0003D   ADDL2   #4, SP                  ; 1744
                                                        05   00040   RSB

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

;  1681          1745  1 !<BLF/PAGE>
```

```
;  1683          1746  1 %SBTTL 'SMG$$CHECK_FOR_OUTPUT_DCB - Check to see if output needed'
;  1684          1747  1 GLOBAL ROUTINE SMG$$CHECK_FOR_OUTPUT_DCB (
;  1685          1748  1                                           DCB : REF BLOCK [,BYTE],
;  1686          1749  1                                           REQUEST_CODE,
;  1687          1750  1                                           REQUEST_ARG
;  1688          1751  1                                         ) =
;  1689          1752  1 !++
;  1690          1753  1 ! FUNCTIONAL DESCRIPTION:
;  1691          1754  1 !
;  1692          1755  1 !       This routine is called when ever a change has been completed
;  1693          1756  1 !       to a given virtual display.  Changes consist of:
;  1694          1757  1 !               1. Altering the text in a virtual display
;  1695          1758  1 !               2. Altering the attributes of the text in a virtual
;  1696          1759  1 !                  display.
;  1697          1760  1 !               3. Altering the cursor position within a display.
;  1698          1761  1 !               4. A virtual display's buffering level going to zero.
;  1699          1762  1 !               5. A virtual display has just been pasted to a
;  1700          1763  1 !                  pasteboard.
;  1701          1764  1 !       The criteria for outputing consist of:
;  1702          1765  1 !               1. Virtual display buffering level is zero
;  1703          1766  1 !               2. The virtual display is pasted to at least one
;  1704          1767  1 !                  pasteboard.
;  1705          1768  1 !
;  1706          1769  1 !       Each pasteboard to which this display is pasted is isolated and
;  1707          1770  1 !       its window image must be redrawn.
;  1708          1771  1 !       The new physical pasteboard cursor position gets set.
;  1709          1772  1 !
;  1710          1773  1 ! CALLING SEQUENCE:
;  1711          1774  1 !
;  1712          1775  1 !       ret_status.wlc.v = SMG$$CHECK_FOR_OUTPUT_DCB (
;  1713          1776  1 !                                                     DCB.rab.r
;  1714          1777  1 !                                                     [,REQEST_CODE.rl.v])
;  1715          1778  1 !
;  1716          1779  1 ! FORMAL PARAMETERS:
;  1717          1780  1 !
;  1718          1781  1 !       DCB.rab.r       Address of virtual display control block for
;  1719          1782  1 !                       affected display.
;  1720          1783  1 !
;  1721          1784  1 !       [REQUEST_CODE.rl.v] Optional.  If supplied, it is a code telling
;  1722          1785  1 !                       the nature of the request that provoked this
;  1723          1786  1 !                       call to SMG$$CHECK_FOR_OUTPUT_DCB.
;  1724          1787  1 !                       Among other things, it tells us when it is
;  1725          1788  1 !                       necessary to redraw borders.
;  1726          1789  1 !
;  1727          1790  1 !       [REQUEST_ARG.rl.v] Optional. If supplied, then it tells us the
;  1728          1791  1 !                       unique line in the DCB that has changed.
;  1729          1792  1 !                       If specified as 0, then the whole DCB has changed.
;  1730          1793  1 !
;  1731          1794  1 ! IMPLICIT INPUTS:
;  1732          1795  1 !
;  1733          1796  1 !       NONE
;  1734          1797  1 !
;  1735          1798  1 ! IMPLICIT OUTPUTS:
;  1736          1799  1 !
;  1737          1800  1 !       WCB[WCB_W_CURR_CUR_ROW]         gets set to place corresponding
;  1738          1801  1 !                                       to logical cursor position in
;  1739          1802  1 !                                       this virtual display.
```

```
;  1740          1803  1 !
;  1741          1804  1 !       WCB[WCB_W_CURR_CUR_COL]          gets set to place corresponding
;  1742          1805  1 !                                        to logical cursor position in
;  1743          1806  1 !                                        this virtual display.
;  1744          1807  1 !
;  1745          1808  1 ! COMPLETION STATUS:
;  1746          1809  1 !
;  1747          1810  1 !       SS$_NORMAL       Normal successful completion
;  1748          1811  1 !
;  1749          1812  1 ! SIDE EFFECTS:
;  1750          1813  1 !
;  1751          1814  1 !       NONE
;  1752          1815  1 !--
;  1753          1816  1
;  1754          1817  2     BEGIN
;  1755          1818  2     BUILTIN
;  1756          1819  2         ACTUALCOUNT;
;  1757          1820  2
;  1758          1821  2     LOCAL
;  1759          1822  2         CURR_PP : REF BLOCK [,BYTE],     ! Addr of pasting packet under
;  1760          1823  2                                          ! inspection
;  1761          1824  2
;  1762          1825  2         STATUS; ! Status of subroutine calls
;  1763          1826  2
;  1764          1827  2 !+
;  1765          1828  2 ! If current buffer level not equal to zero, this is not the time to
;  1766          1829  2 ! flush this display to the screen.
;  1767          1830  2 !-
;  1768          1831  2     IF .DCB [DCB_L_BATCH_LEVEL] NEQ 0
;  1769          1832  2     THEN
;  1770          1833  2         RETURN (SS$_NORMAL);
;  1771          1834  2
;  1772          1835  2     CURR_PP = .DCB [DCB_A_PP_NEXT];      ! Start of chain of pasting
;  1773          1836  2                                          ! packets to which this display
;  1774          1837  2                                          ! is pasted.
;  1775          1838  2     IF .CURR_PP EQL 0
;  1776          1839  2     THEN
;  1777          1840  2         RETURN (SMG$_FATERRLIB);         ! should never be 0
;  1778          1841  2
;  1779          1842  2 !+
;  1780          1843  2 ! Deal with each pasteboard that this virtual display is pasted to...
;  1781          1844  2 !-
;  1782          1845  2     WHILE .CURR_PP NEQ DCB [DCB_A_PP_NEXT]
;  1783          1846  2     DO
;  1784          1847  3         BEGIN   ! Overall loop
;  1785          1848  3
;  1786          1849  3         LOCAL
;  1787          1850  3             PBCB : REF BLOCK [,BYTE],    ! Addr. of pasteboard control
;  1788          1851  3             WCB  : REF BLOCK [,BYTE];    ! Addr. of window control block
;  1789          1852  3
;  1790          1853  3         PBCB = .CURR_PP [PP_A_PBCB_ADDR]; ! Select pasteboard and WCB
;  1791          1854  3
;  1792          1855  3         IF .PBCB [PBCB_L_BATCH_LEVEL] EQL 0
;  1793          1856  3         THEN
;  1794          1857  4             BEGIN       ! Force output
;  1795          1858  4             LOCAL
;  1796          1859  4                 OPT_FLAG;       ! Flag
```

```
;  1797          1860  4
;  1798          1861  4              WCB  = .PBCB [PBCB_A_WCB]; ! whose window needs rebuilding.
;  1799          1862  4
;  1800          1863  4              !+
;  1801          1864  4              ! Set the cursor position in the pasteboard to correspond
;  1802          1865  4              ! to it's place on this virtual display.
;  1803          1866  4              !-
;  1804          1867  4              WCB [WCB_W_CURR_CUR_ROW] = .DCB [DCB_W_CURSOR_ROW]-1
;  1805          1868  4                                       + .CURR_PP [PP_W_ROW];
;  1806          1869  4
;  1807          1870  4              WCB [WCB_W_CURR_CUR_COL] = .DCB [DCB_W_CURSOR_COL]-1
;  1808          1871  4                                       + .CURR_PP [PP_W_COL];
;  1809          1872  4
;  1810          1873  4              !+
;  1811          1874  4              ! If REQUEST_CODE is present and it is a type that just
;  1812          1875  4              ! involves cursor motion, deal with it as a special case to
;  1813          1876  4              ! avoid remapping the virtual display since contents did
;  1814          1877  4              ! not change.
;  1815          1878  4              !-
;  1816          1879  4              OPT_FLAG = 0;
;  1817          1880  4              IF ACTUALCOUNT () GEQ 2     ! REQUEST_CODE present
;  1818          1881  4              THEN
;  1819          1882  5                  BEGIN   ! Try optimization
;  1820          1883  5                  IF .REQUEST_CODE EQL SMG$C_SET_CURSOR_ABS OR
;  1821          1884  5                     .REQUEST_CODE EQL SMG$C_SET_CURSOR_REL
;  1822          1885  5                  THEN
;  1823          1886  5                     OPT_FLAG = 1;
;  1824          1887  4                  END;    ! Try optimization
;  1825          1888  4
;  1826          1889  4              IF .OPT_FLAG
;  1827          1890  4              THEN
;  1828          1891  5                  BEGIN   ! Opt. possible
;  1829          1892  5                  !+
;  1830          1893  5                  ! Force cursor to desired spot in optimal fashion
;  1831          1894  5                  !-
;  1832          1895  5                  SMG$$UPDATE_PHYSICAL_CURSOR ( .PBCB);
;  1833          1896  5                  END     ! Opt. possible
;  1834          1897  5
;  1835          1898  4              ELSE
;  1836          1899  4
;  1837          1900  5                  BEGIN   ! Remapping necessary
;  1838          1901  5                  !+
;  1839          1902  5                  ! Check to see if we can get by with just redrawing the
;  1840          1903  5                  ! changed virtual display, or whether we need to redraw
;  1841          1904  5                  ! the changed one and all higher-pasted virtual
;  1842          1905  5                  ! displays.  We can do the former if this pasting has a
;  1843          1906  5                  ! bit that says it is not occluded.
;  1844          1907  5                  !-
;  1845          1908  5                  IF NOT .CURR_PP [PP_V_OCCLUDED]
;  1846          1909  5                  THEN
;  1847          1910  6                      BEGIN       ! Redraw just this virtual display
;  1848          1911  7                      IF NOT (STATUS = SMG$$MOVE_TEXT_TO_WINDOW_BUF (.CURR_PP))
;  1849          1912  6                      THEN
;  1850          1913  6                          RETURN (.STATUS);
;  1851          1914  6
;  1852          1915  6                      !+
;  1853          1916  6                      ! If something actually got redrawn, tell the output
```

```
;  1854     1917  6                    ! routines we've changed the whole virtual display.
;  1855     1918  6                    ! If just one line has changed, however, then tell
;  1856     1919  6                    ! the output routines which line changed.
;  1857     1920  6                    !-
;  1858     1921  6                    IF .CURR_PP [PP_W_ROWS_TO_MOVE] NEQ 0
;  1859     1922  6                    THEN
;  1860     1923  7                        BEGIN
;  1861     1924  7                        PBCB [PBCB_W_FIRST_CHANGED_ROW] =
;  1862     1925  7                                .CURR_PP [PP_W_FIRST_WCB_ROW];
;  1863     1926  7
;  1864     1927  7                        PBCB [PBCB_W_LAST_CHANGED_ROW]  =
;  1865     1928  7                                .CURR_PP [PP_W_LAST_WCB_ROW];
;  1866     1929  7
;  1867     1930  7                        PBCB [PBCB_W_FIRST_CHANGED_COL] =
;  1868     1931  7                                .CURR_PP [PP_W_FIRST_WCB_COL];
;  1869     1932  7
;  1870     1933  7                        PBCB [PBCB_W_LAST_CHANGED_COL]  =
;  1871     1934  7                                .CURR_PP [PP_W_LAST_WCB_COL];
;  1872     1935  6                        END;
;  1873     1936  6
;  1874     1937  6                    IF ACTUALCOUNT () GEQ 3 ! If REQUEST_ARG supplied
;  1875     1938  6                    THEN
;  1876     1939  6                        IF .REQUEST_CODE EQL SMG$C_PUT_CHARS AND
;  1877     1940  6                           .REQUEST_ARG  NEQ 0
;  1878     1941  6                        THEN
;  1879     1942  7                            BEGIN       ! Just one row changed
;  1880     1943  7                            PBCB [PBCB_W_FIRST_CHANGED_ROW] =
;  1881     1944  7                                .CURR_PP [PP_W_ROW]+.REQUEST_ARG-1;
;  1882     1945  7
;  1883     1946  7                            PBCB [PBCB_W_LAST_CHANGED_ROW]  =
;  1884     1947  7                                .CURR_PP [PP_W_ROW]+.REQUEST_ARG-1;
;  1885     1948  7
;  1886     1949  6                            END;        ! Just one row changed
;  1887     1950  6
;  1888     1951  6                    !+
;  1889     1952  6                    ! If REQUEST_CODE is present and it is of a type
;  1890     1953  6                    ! that could have added a border, we must redraw the
;  1891     1954  6                    ! border at this time.
;  1892     1955  6                    !-
;  1893     1956  6                    IF ACTUALCOUNT () GEQ 2 ! If REQUEST_CODE supplied
;  1894     1957  6                    THEN
;  1895     1958  6                        IF .REQUEST_CODE EQL SMG$C_LABEL_BORDER
;  1896     1959  6                        THEN
;  1897     1960  6                            IF .DCB [DCB_V_BORDERED]
;  1898     1961  6                            THEN
;  1899     1962  7                                IF NOT (STATUS = SMG$$DRAW_BORDER (
;  1900     1963  7                                                    .DCB, .CURR_PP, .WCB))
;  1901     1964  6                                THEN
;  1902     1965  6                                    RETURN (.STATUS);
;  1903     1966  6
;  1904     1967  6                    SMG$$MIN_UPD (.PBCB);
;  1905     1968  6
;  1906     1969  6                    END         ! Redraw just this virtual display
;  1907     1970  6
;  1908     1971  5                ELSE
;  1909     1972  5
;  1910     1973  6                    BEGIN       ! Must redraw it and outer ones
```

```
;  1911     1974  6                    !+
;  1912     1975  6                    ! Go rebuild window buffers for this pasteboard
;  1913     1976  6                    ! starting with current pasting packet, and output.
;  1914     1977  6                    !-
;  1915     1978  7                    IF NOT (STATUS = SMG$$FILL_WINDOW_BUFFER ( .CURR_PP))
;  1916     1979  6                    THEN
;  1917     1980  6                        RETURN (.STATUS);
;  1918     1981  6
;  1919     1982  5                    END;        ! Must redraw it and outer ones
;  1920     1983  4                END;    ! Remapping necessary
;  1921     1984  3            END;        ! Force output
;  1922     1985  3
;  1923     1986  3        CURR_PP = .CURR_PP [PP_A_NEXT_DCB];     ! Walk the DCB side of
;  1924     1987  3                                                ! the chain from front
;  1925     1988  3                                                ! to back.
;  1926     1989  2        END;    ! Overall loop
;  1927     1990  2
;  1928     1991  2    RETURN (SS$_NORMAL);
;  1929     1992  1    END;                ! End of routine SMG$$CHECK_FOR_OUTPUT_DCB



                                 007C 00000          .ENTRY  SMG$$CHECK_FOR_OUTPUT_DCB, Save R2,R3,R4,-  ; 1747
                                                             R5,R6
                           55        04   AC  D0 00002      MOVL    DCB, R5                             ; 1831
                                     1C   A5  D5 00006      TSTL    28(R5)
                                          03  13 00009      BEQL    2$
                                        00E9  31 0000B 1$:   BRW     14$
                                52        20  A5  D0 0000E 2$:    MOVL    32(R5), CURR_PP             ; 1835
                                          08  12 00012      BNEQ    3$                                  ; 1838
                                50 00000000G  8F  D0 00014      MOVL    #SMG$_FATERRLIB, R0             ; 1840
                                              04  0001B      RET
                                50        20  A5  9E 0001C 3$:    MOVAB   32(R5), R0                  ; 1845
                                50            52  D1 00020      CMPL    CURR_PP, R0
                                              E6  13 00023      BEQL    1$
                                54        14  A2  D0 00025      MOVL    20(CURR_PP), PBCB               ; 1853
                                        00A4  C4  D5 00029      TSTL    164(PBCB)                       ; 1855
                                              46  12 0002D      BNEQ    6$
                                53        08  A4  D0 00C2F      MOVL    8(PBCB), WCB                    ; 1861
                                50        28  A5  3C 00033      MOVZWL  40(R5), R0                      ; 1868
                                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                      ; 1871
                                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                        ; 1879
                                02            6C  91 00055      CMPB    (AP), #2                        ; 1880
                                              0F  1F 00058      BLSSU   5$
                                16        08  AC  D1 0005A      CMPL    REQUEST_CODE, #22               ; 1883
                                              06  13 0005E      BEQL    4$
                                17        08  AC  D1 00060      CMPL    REQUEST_CODE, #23               ; 1884
                                              03  12 00064      BNEQ    5$
                                50            01  D0 00068 4$:    MOVL    #1, OPT_FLAG                    ; 1886
```

```
                          0B            50  E9 00069 5$:      BLBC    OPT_FLAG, 7$                                    ; 1889
                                        54  DD 0006C          PUSHL   PBCB                                            ; 1895
                00000000G   00          01  FB 0006E          CALLS   #1, SMG$$UPDATE_PHYSICAL_CURSOR
                                        7A  11 00075 6$:      BRB     13$                                             ; 1889
                          65      2A    A2  E8 00077 7$:      BLBS    42(CURR_PP), 11$                                ; 1908
                                        52  DD 0007B          PUSHL   CURR_PP                                         ; 1911
                    0000V CF            01  FB 0007D          CALLS   #1, SMG$$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)                                     ; 1921
                                        06  13 0008B          BEQL    8$
                    00A8  C4      2F    A2  7D 0008D          MOVQ    47(CURR_PP), 168(PBCB)                          ; 1925
                          03            6C  91 00093 8$:      CMPB    (AP), #3                                        ; 1937
                                        1F  1F 00096          BLSSU   9$
                          11      08    AC  D1 00098          CMPL    REQUEST_CODE, #17                               ; 1939
                                        19  12 0009C          BNEQ    9$
                                  0C    AC  D5 0009E          TSTL    REQUEST_ARG                                     ; 1940
                                        14  13 000A1          BEQL    9$
                          50      18    A2  32 000A3          CVTWL   24(CURR_PP), R0                                 ; 1944
                          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)                                   ; 1947
                          02            6C  91 000B7 9$:      CMPB    (AP), #2                                        ; 1956
                                        19  1F 000BA          BLSSU   10$
                          1C      08    AC  D1 000BC          CMPL    REQUEST_CODE, #28                               ; 1958
                                        13  12 000C0          BNEQ    10$
                          0F      2F    A5  E9 000C2          BLBC    47(R5), 10$                                     ; 1960
                                        0C  BB 000C6          PUSHR   #^M<R2,R3>                                      ; 1963
                                        55  DD 000C8          PUSHL   R5
                        0000V CF        03  FB 000CA          CALLS   #3, SMG$$DRAW_BORDER
                          56            50  D0 000CF          MOVL    R0, STATUS
                          18            56  E9 000D2          BLBC    STATUS, 12$                                     ; 1962
                                        54  DD 000D5 10$:      PUSHL   PBCB                                            ; 1967
                00000000G   00          01  FB 000D7          CALLS   #1, SMG$$MIN_UPD
                                        11  11 000DE          BRB     13$                                             ; 1908
                                        52  DD 000E0 11$:      PUSHL   CURR_PP                                         ; 1978
                        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$:      MOVL    STATUS, R0                                      ; 1980
                                        04  000F0             RET
                          52          62    D0 000F1 13$:      MOVL    (CURR_PP), CURR_PP                              ; 1986
                                    FF25    31 000F4          BRW     3$                                              ; 1845
                          50            01  D0 000F7 14$:      MOVL    #1, R0                                          ; 1991
                                        04  000FA             RET                                                     ; 1992

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

; 1930          1993  1 !<BLF/PAGE>
```

```
;  1932     1994  1 %SBTTL 'SMG$$CHECK_FOR_OUTPUT_PBCB - Check to see if output needed'
;  1933     1995  1 GLOBAL ROUTINE SMG$$CHECK_FOR_OUTPUT_PBCB (
;  1934     1996  1                                           PBCB : REF BLOCK [,BYTE]
;  1935     1997  1                                         ) =
;  1936     1998  1 !++
;  1937     1999  1 ! FUNCTIONAL DESCRIPTION:
;  1938     2000  1 !
;  1939     2001  1 !       This routine is called when a change has been made to a
;  1940     2002  1 !       particular pasteboard that requires the entire screen image to
;  1941     2003  1 !       be recomposed, e.g. as a result of an UNPASTE operation.
;  1942     2004  1 !
;  1943     2005  1 ! CALLING SEQUENCE:
;  1944     2006  1 !
;  1945     2007  1 !       ret_status.wlc.v = SMG$$CHECK_FOR_OUTPUT_PBCB ( PBCB.rab.r)
;  1946     2008  1 !
;  1947     2009  1 ! FORMAL PARAMETERS:
;  1948     2010  1 !
;  1949     2011  1 !       PBCB.rab.r      Address of pasteboard control block for
;  1950     2012  1 !                       affected display.
;  1951     2013  1 !
;  1952     2014  1 ! IMPLICIT INPUTS:
;  1953     2015  1 !
;  1954     2016  1 !       PBCB [PBCB_L_BATCH_LEVEL]
;  1955     2017  1 !
;  1956     2018  1 ! IMPLICIT OUTPUTS:
;  1957     2019  1 !
;  1958     2020  1 !       NONE
;  1959     2021  1 !
;  1960     2022  1 ! COMPLETION STATUS:
;  1961     2023  1 !
;  1962     2024  1 !       SMG$_BATWAS_ON  Ok, but batching was on so nothing happened
;  1963     2025  1 !       SS$_NORMAL      Normal successful completion
;  1964     2026  1 !
;  1965     2027  1 ! SIDE EFFECTS:
;  1966     2028  1 !
;  1967     2029  1 !       NONE
;  1968     2030  1 !--
;  1969     2031  1
;  1970     2032  2     BEGIN
;  1971     2033  2     LOCAL
;  1972     2034  2         WCB : REF BLOCK [,BYTE],        ! Address of window control
;  1973     2035  2                                         ! block
;  1974     2036  2         STATUS;                         ! Status of subroutine calls
;  1975     2037  2
;  1976     2038  2     WCB = .PBCB [PBCB_A_WCB];
;  1977     2039  2
;  1978     2040  2 !+
;  1979     2041  2 ! Do nothing if batching is in effect.
;  1980     2042  2 !-
;  1981     2043  2
;  1982     2044  2     IF .PBCB [PBCB_L_BATCH_LEVEL] NEQ 0
;  1983     2045  2     THEN    RETURN SMG$_BATWAS_ON;
;  1984     2046  2
;  1985     2047  2 !+
;  1986     2048  2 ! Clear text buffer to blanks and attribute buffer to zeroes before we
;  1987     2049  2 ! start.  Clear alternate character set buffer to zeroes if it exists.
;  1988     2050  2 !-
```

```
;  1989          2051  2     CH$FILL (%C' ', .WCB [WCB_L_BUFSIZE], .WCB [WCB_A_TEXT_BUF]);
;  1990          2052  2
;  1991          2053  2     CH$FILL (0,     .WCB [WCB_L_BUFSIZE], .WCB [WCB_A_ATTR_BUF]);
;  1992          2054  2
;  1993          2055  2     IF .WCB [WCB_A_CHAR_SET_BUF] NEQ 0
;  1994          2056  2     THEN
;  1995          2057  2         CH$FILL (0, .WCB [WCB_L_BUFSIZE], .WCB [WCB_A_CHAR_SET_BUF]);
;  1996          2058  2
;  1997          2059  2 !+
;  1998          2060  2 ! Clear text line characteristics vector to zero.
;  1999          2061  2 !-
;  2000          2062  2     CH$FILL (0, .WCB [WCB_W_NO_ROWS] +1, .WCB [WCB_A_LINE_CHAR]);
;  2001          2063  2
;  2002          2064  2 !+
;  2003          2065  2 ! Tell the output routines that we've changed the whole WCB buffer.
;  2004          2066  2 !-
;  2005          2067  2     PBCB [PBCB_W_FIRST_CHANGED_ROW] = 1;
;  2006          2068  2     PBCB [PBCB_W_LAST_CHANGED_ROW]  = .PBCB [PBCB_B_ROWS];
;  2007          2069  2     PBCB [PBCB_W_FIRST_CHANGED_COL] = 1;
;  2008          2070  2     PBCB [PBCB_W_LAST_CHANGED_COL]  = .PBCB [PBCB_W_WIDTH];
;  2009          2071  2
;  2010          2072  2 !+
;  2011          2073  2 ! Go rebuild window buffers for this pasteboard and output.
;  2012          2074  2 ! Pass SMG$$FILL_WINDOW_BUFFER the address of the first pasting packet
;  2013          2075  2 ! if any exist, else just output now-blank screen.
;  2014          2076  2 !-
;  2015          2077  2     IF .PBCB [PBCB_A_PP_NEXT] EQL PBCB [PBCB_A_PP_NEXT]
;  2016          2078  2     THEN
;  2017          2079  3         BEGIN   ! Nothing pasted -- output blanked buffer
;  2018          2080  3         RETURN (SMG$$MIN_UPD (.PBCB));
;  2019          2081  3         END     ! Nothing pasted -- output blanked buffer
;  2020          2082  3
;  2021          2083  2     ELSE
;  2022          2084  2
;  2023          2085  3         RETURN (SMG$$FILL_WINDOW_BUFFER (
;  2024          2086  2                         .PBCB [PBCB_A_PP_PREV] - PP_PBCB_QUEUE_OFFSET) );
;  2025          2087  2
;  2026          2088  1     END;                ! End of routine SMG$$CHECK_FOR_OUTPUT_PBCB


                                        00FC 00000          .ENTRY  SMG$$CHECK_FOR_OUTPUT_PBCB, Save R2,R3,R4,- ;  1995
                                                                    R5,R6,R7
                                    57    04   AC  D0 00002        MOVL    PBCB, R7                                ;  2038
                                    56    08   A7  D0 00006        MOVL    8(R7), WCB
                                        00A4   C7  D5 0000A        TSTL    164(R7)                                 ;  2044
                                               08  13 0000E        BEQL    1$
                                    50 00000000G   8F  D0 00010    MOVL    #SMG$_BATWAS_ON, R0                     ;  2045
                                                   04  00017    1$: RET
    28  A6          20            6E             00  2C 00018    MOVC5   #0, (SP), #32, 40(WCB), @8(WCB)         ;  2051
                                            08     B6  0001E
    28  A6          00            6E             00  2C 00020    MOVC5   #0, (SP), #0, 40(WCB), @12(WCB)         ;  2053
                                            0C     B6  00026
                                        10     A6  D5 00028        TSTL    16(WCB)                                 ;  2055
                                               08  13 0002B        BEQL    2$
```

```
      28   A6            00            6E            00  2C 0002D          MOVC5   #0, (SP), #0, 40(WCB), a16(WCB)          ; 2057
                                                    10  B6 00033
                                       50         02  A6  3C 00035 2$:     MOVZWL  2(WCB), R0                            ; 2062
                                                      50  D6 00039          INCL    R0                                    ;
           50            00            6E            00  2C 0003B          MOVC5   #0, (SP), #0, R0, a44(WCB)            ;
                                                    2C  B6 00040                                                        ;
                                 00A8    C7            01  B0 00042          MOVW    #1, 168(R7)                           ; 2067
                                 00AA    C7         5F  A7  9B 00047          MOVZBW  95(R7), 170(R7)                       ; 2068
                                 00AC    C7            01  B0 0004D          MOVW    #1, 172(R7)                           ; 2069
                                 00AE    C7         5A  A7  B0 00052          MOVW    90(R7), 174(R7)                       ; 2070
                                         57            67  D1 00058          CMPL    (R7), R7                              ; 2077
                                                       0A  12 0005B          BNEQ    3$                                    ;
                                                       57  DD 0005D          PUSHL   R7                                    ; 2080
                              00000000G  00            01  FB 0005F          CALLS   #1, SMG$$MIN_UPD                      ;
                                                       04  00066          RET                                           ; 2085
                           7E      04    A7            08  C3 00067 3$:     SUBL3   #8, 4(R7), -(SP)                      ; 2086
                                 0000V   CF            01  FB 0006C          CALLS   #1, SMG$$FILL_WINDOW_BUFFER           ;
                                                       04  00071          RET                                           ; 2088

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

; 2027          2089  1 !<BLF/PAGE>
```

```
;  2029          2090  1 %SBTTL 'SMG$$DRAW_BORDER - Move border characters into buffer'
;  2030          2091  1 GLOBAL ROUTINE SMG$$DRAW_BORDER (
;  2031          2092  1         DCB : REF BLOCK [,BYTE],        ! Display control block address
;  2032          2093  1         PP  : REF BLOCK [,BYTE],        ! Pasting packet address
;  2033          2094  1         WCB : REF BLOCK [,BYTE]         ! Window control block address
;  2034          2095  1                                 ) =
;  2035          2096  1 !++
;  2036          2097  1 ! FUNCTIONAL DESCRIPTION:
;  2037          2098  1 !
;  2038          2099  1 !       Draw the border characters into the text buffer of the WCB.
;  2039          2100  1 !       The cell in the text buffer has bits set to indicate which
;  2040          2101  1 !       elements of a border character will be needed in this cell
;  2041          2102  1 !       position.  When the first such bit is set in the text cell,
;  2042          2103  1 !       the corresponding cell in the attribute array is flag with
;  2043          2104  1 !       a bit which signals that the text array cell no longer contains
;  2044          2105  1 !       a printable character, but an encoding of the border character.
;  2045          2106  1 !
;  2046          2107  1 ! CALLING SEQUENCE:
;  2047          2108  1 !
;  2048          2109  1 !       ret_status.wlc.v = SMG$$DRAW_BORDER (   DCB.rab.r,
;  2049          2110  1 !                                               PP.rab.r,
;  2050          2111  1 !                                               WCB.rab.r)
;  2051          2112  1 !
;  2052          2113  1 ! FORMAL PARAMETERS:
;  2053          2114  1 !
;  2054          2115  1 !       DCB.rab.r               ! Display control block address
;  2055          2116  1 !       PP.rab.r                ! Pasting packet address
;  2056          2117  1 !       WCB.rab.r               ! Window control block address
;  2057          2118  1 !
;  2058          2119  1 ! IMPLICIT INPUTS:
;  2059          2120  1 !
;  2060          2121  1 !       NONE
;  2061          2122  1 !
;  2062          2123  1 ! IMPLICIT OUTPUTS:
;  2063          2124  1 !
;  2064          2125  1 !       NONE
;  2065          2126  1 !
;  2066          2127  1 ! COMPLETION STATUS:
;  2067          2128  1 !
;  2068          2129  1 !       SS$_NORMAL      Normal successful completion
;  2069          2130  1 !
;  2070          2131  1 ! SIDE EFFECTS:
;  2071          2132  1 !
;  2072          2133  1 !       NONE
;  2073          2134  1 !--
;  2074          2135  1
;  2075          2136  2     BEGIN
;  2076          2137  2
;  2077          2138  2 !+
;  2078          2139  2 ! Macro $MARKIT checks a "text" character to see if it is already a
;  2079          2140  2 ! border element in which case it OR's in new border contribution.
;  2080          2141  2 ! If it is not yet flagged as being a border element it stores the
;  2081          2142  2 ! border contribution and marks attribute byte to record this is a
;  2082          2143  2 ! border element.
;  2083          2144  2 !-
;  2084          2145  2 MACRO
;  2085        M 2146  2   $MARKIT ( OFFSET, ELEMENT ) =
```

```
;  2086        M 2147  2      BEGIN
;  2087        M 2148  2      IF .(PTRA [.OFFSET])<ATTR_V_BORD_ELEM, 1>
;  2088        M 2149  2      THEN
;  2089        M 2150  2          PTRT [.OFFSET] = .PTRT [.OFFSET] OR ELEMENT
;  2090        M 2151  2      ELSE
;  2091        M 2152  2          BEGIN
;  2092        M 2153  2          PTRT [.OFFSET] = ELEMENT;
;  2093        M 2154  2          PTRA [.OFFSET] = ATTR_M_BORD_ELEM ;
;  2094        M 2155  2          END;
;  2095          2156  2      END %;
;  2096          2157  2
;  2097          2158  2  !+
;  2098          2159  2  ! Macro $CALC_LEFT_INDEX computes the byte offset into the WCB text and
;  2099          2160  2  ! attribute buffers where a left border element needs to be placed for
;  2100          2161  2  ! a double-wide line to make it come out in the correct column.
;  2101          2162  2  !-
;  2102          2163  2  MACRO
;  2103        M 2164  2      $CALC_LEFT_INDEX =
;  2104        M 2165  2      BEGIN
;  2105        M 2166  2          SINDEX = (.WLN -1) * .WCB [WCB_W_NO_COLS] +
;  2106        M 2167  2                              (.PP [PP_W_FIRST_WCB_COL] / 2) -1;
;  2107          2168  2      END %;
;  2108          2169  2
;  2109          2170  2  !+
;  2110          2171  2  ! Macro $CALC_RIGHT_INDEX computes the byte offset into the WCB text and
;  2111          2172  2  ! attribute buffers where a right border element needs to be placed for
;  2112          2173  2  ! a double-wide line to make it come out in the correct column.
;  2113          2174  2  !-
;  2114          2175  2  MACRO
;  2115        M 2176  2      $CALC_RIGHT_INDEX =
;  2116        M 2177  2      BEGIN
;  2117        M 2178  2          SINDEX = (.WLN -1) * .WCB [WCB_W_NO_COLS] +
;  2118        M 2179  2                   ((.PP [PP_W_FIRST_WCB_COL] + .PP [PP_W_MOVE_LENGTH] +1) / 2) -1;
;  2119        M 2180  2      END %;
;  2120          2181  2
;  2121          2182  2      LOCAL
;  2122          2183  2          SPOS,
;  2123          2184  2          ! Position where the first (or only) border element will be
;  2124          2185  2          ! written.  This is a displacement from the beginning of the
;  2125          2186  2          ! window buffer.  Before using this position, we must always
;  2126          2187  2          ! check to see if it points into the confines of the buffer.
;  2127          2188  2          ! This basically tells us if that portion of the border is
;  2128          2189  2          ! visible considering how the virtual display is pasted to the
;  2129          2190  2          ! pasteboard.
;  2130          2191  2
;  2131          2192  2          LDES : REF BLOCK [,BYTE],          ! Address of the dynamic string
;  2132          2193  2                                             ! descriptor in the DCB which
;  2133          2194  2                                             ! records the border label text
;  2134          2195  2                                             ! string.
;  2135          2196  2
;  2136          2197  2          PTRT : REF VECTOR [,BYTE],         ! pointer to top of WCB TEXT buf
;  2137          2198  2          PTRA : REF VECTOR [,BYTE],         ! pointer to top of WCB ATTR buf
;  2138          2199  2          UPPER_ROW,          ! Row number of top border
;  2139          2200  2          LOWER_ROW,          ! Row number of bottom border
;  2140          2201  2          LEFT_COL,           ! Col number of left border
;  2141          2202  2          RIGHT_COL,          ! Col number of right border
;  2142          2203  2          WCB_LCV : REF VECTOR [,BYTE],      ! Line control block
```

```
;  2143          2204  2          PBCB : REF BLOCK [,BYTE];       ! Addr. of pasteboard control
;  2144          2205  2                                          ! block
;  2145          2206  2
;  2146          2207  2
;  2147          2208  2      PTRT = .WCB [WCB_A_TEXT_BUF];
;  2148          2209  2      PTRA = .WCB [WCB_A_ATTR_BUF];
;  2149          2210  2      PBCB = .PP [PP_A_PBCB_ADDR];
;  2150          2211  2      WCB_LCV = .WCB [WCB_A_LINE_CHAR];
;  2151          2212  2
;  2152          2213  2  !+
;  2153          2214  2  ! If no part of the virtual display projects to the visible part of the
;  2154          2215  2  ! pasteboard, abandon attempt to draw border.  This is strictly speaking not
;  2155          2216  2  ! the correct solution since it ignores the case where a virtual display is
;  2156          2217  2  ! pasted such that it is not visible, but its border should be.  Trying to
;  2157          2218  2  ! make this pathelogical case work will require some coordinated work on the
;  2158          2219  2  ! part of SMG$$CALC_PASTE_TRANS.  (See corresponding note there.)
;  2159          2220  2  !-
;  2160          2221  2      IF .PP [PP_W_ROWS_TO_MOVE] EQL 0  OR
;  2161          2222  2         .PP [PP_W_MOVE_LENGTH]  EQL 0
;  2162          2223  2      THEN
;  2163          2224  2          RETURN (SS$_NORMAL);
;  2164          2225  2
;  2165          2226  2  !+
;  2166          2227  2  ! Compute the row and column numbers where the borders fall.  Note these
;  2167          2228  2  ! rows and columns may not map into buffer and need to be validated
;  2168          2229  2  ! before use.
;  2169          2230  2  !-
;  2170          2231  2      UPPER_ROW = .PP [PP_W_ROW] - 1;
;  2171          2232  2      LOWER_ROW = MAX ( .PP [PP_W_ROW], 1) + .PP [PP_W_ROWS_TO_MOVE];
;  2172          2233  2      LEFT_COL  = .PP [PP_W_COL] - 1;
;  2173          2234  2      RIGHT_COL = MAX ( .PP [PP_W_COL], 1) + .PP [PP_W_MOVE_LENGTH];
;  2174          2235  2
;  2175          2236  2  !+
;  2176          2237  2  ! If the line above where the virtual display is pasted exists in the
;  2177          2238  2  ! window buffer, draw the the top border.
;  2178          2239  2  !-
;  2179          2240  2      IF .UPPER_ROW GEQ 1
;  2180          2241  2      THEN
;  2181          2242  2          BEGIN   ! Top border
;  2182          2243  2
;  2183          2244  2              !+
;  2184          2245  2              ! If WCB is special, blank it and force line to normal.
;  2185          2246  2              !-
;  2186          2247  2
;  2187          2248  2              IF (.WCB_LCV[.UPPER_ROW] NEQ 0) AND .PBCB[PBCB_V_WIDE]
;  2188          2249  2              THEN
;  2189          2250  2                  BEGIN   ! Force line normal
;  2190          2251  2                  CH$FILL(%C' ',.WCB[WCB_W_NO_COLS],
;  2191          2252  2                          PTRT[ (.UPPER_ROW-1)*.WCB[WCB_W_NO_COLS] ]);
;  2192          2253  2                  CH$FILL(%C' ',.WCB[WCB_W_NO_COLS],
;  2193          2254  2                          PTRA[ (.UPPER_ROW-1)*.WCB[WCB_W_NO_COLS] ]);
;  2194          2255  2                  WCB_LCV[.UPPER_ROW]=0
;  2195          2256  2                  END;    ! Force line normal
;  2196          2257  2
;  2197          2258  2          SPOS = .PP [PP_W_TO_INDEX] - .WCB [WCB_W_NO_COLS] ;
;  2198          2259  2          INCR I FROM 1 TO .PP [PP_W_MOVE_LENGTH]
;  2199          2260  2          DO
```

```
;  2200          2261  4              BEGIN       ! Top line loop
;  2201          2262  4              $MARKIT (SPOS, BORD_M_HORIZ );
;  2202          2263  4              (PTRT [.SPOS])<BORD_V_DOWN,1> = 0; ! Knock down "down" seg.
;  2203          2264  4              SPOS = .SPOS + 1;
;  2204          2265  4              END;        ! Top line loop
;  2205          2266
;  2206          2267                 IF .UPPER_ROW LSS .PBCB [PBCB_W_FIRST_CHANGED_ROW]
;  2207          2268                 THEN
;  2208          2269                     PBCB [PBCB_W_FIRST_CHANGED_ROW] = .UPPER_ROW;
;  2209          2270
;  2210          2271                 END;    ! Top border
;  2211          2272
;  2212          2273         !+
;  2213          2274         ! If the line below where the virtual display is pasted exists in the
;  2214          2275         ! window buffer, draw the the bottom border.
;  2215          2276         !-
;  2216          2277             IF .LOWER_ROW LEQ .WCB [WCB_W_NO_ROWS]
;  2217          2278             THEN
;  2218          2279                 BEGIN   ! Bottom border
;  2219          2280
;  2220          2281                     !+
;  2221          2282                     ! If WCB is special, blank it and force line to normal.
;  2222          2283                     !-
;  2223          2284
;  2224          2285                     IF (.WCB_LCV[.LOWER_ROW] NEQ 0) AND .PBCB[PBCB_V_WIDE]
;  2225          2286                     THEN
;  2226          2287                         BEGIN   ! Force line normal
;  2227          2288                         CH$FILL(%C' ',.WCB[WCB_W_NO_COLS],
;  2228          2289                                 PTRT[ (.LOWER_ROW-1)*.WCB[WCB_W_NO_COLS] ]);
;  2229          2290                         CH$FILL(%C' ',.WCB[WCB_W_NO_COLS],
;  2230          2291                                 PTRA[ (.LOWER_ROW-1)*.WCB[WCB_W_NO_COLS] ]);
;  2231          2292                         WCB_LCV[.LOWER_ROW]=0
;  2232          2293                         END;    ! Force line normal
;  2233          2294
;  2234          2295                 SPOS = .PP [PP_W_TO_INDEX] + (.PP  [PP_W_ROWS_TO_MOVE] *
;  2235          2296                                               .WCB [WCB_W_NO_COLS] ) ;
;  2236          2297                 INCR I FROM 1 TO .PP [PP_W_MOVE_LENGTH]
;  2237          2298                 DO
;  2238          2299                     BEGIN       ! Bottom line loop
;  2239          2300                     $MARKIT (SPOS, BORD_M_HORIZ );
;  2240          2301                     (PTRT [.SPOS])<BORD_V_UP,1> = 0;    ! Knock down "up" seg.
;  2241          2302                     SPOS = .SPOS + 1;
;  2242          2303                     END;        ! Bottom line loop
;  2243          2304
;  2244          2305                 IF .LOWER_ROW GTR .PBCB [PBCB_W_LAST_CHANGED_ROW]
;  2245          2306                 THEN
;  2246          2307                     PBCB [PBCB_W_LAST_CHANGED_ROW] = .LOWER_ROW;
;  2247          2308
;  2248          2309                 END;    ! Bottom border
;  2249          2310
;  2250          2311         !+
;  2251          2312         ! If the column to the left of where the virtual display is pasted
;  2252          2313         ! exists in the window buffer, draw the left border.
;  2253          2314         !-
;  2254          2315             IF .LEFT_COL GEQ 1
;  2255          2316             THEN
;  2256          2317                 BEGIN   ! Left border
```

```
;  2257          2318  3          LOCAL
;  2258          2319  3              WLN,            ! Line number in WCB buffers
;  2259          2320  3              LCV : REF VECTOR [,BYTE]; ! Addr of line characteristics
;  2260          2321  3                                       ! vector in WCB
;  2261          2322  3          WLN = .PP [PP_W_FIRST_WCB_ROW];
;  2262          2323  3          LCV = .WCB [WCB_A_LINE_CHAR];
;  2263          2324  3          SPOS = .PP [PP_W_TO_INDEX] - 1;
;  2264          2325  3          INCR I FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
;  2265          2326  3          DO
;  2266          2327  4              BEGIN       ! Left column loop
;  2267          2328  4              IF .LCV [.WLN] EQL 0 OR             ! If normal or
;  2268          2329  5                 (NOT .PBCB [PBCB_V_WIDE])        ! DWDH not supported
;  2269          2330  4              THEN
;  2270          2331  5                  BEGIN   ! Normal case
;  2271          2332  5                  $MARKIT (SPOS, BORD_M_VERT ) ;
;  2272          2333  5                  (PTRT [.SPOS])<BORD_V_RIGHT,1> = 0; ! Knock down "right" seg.
;  2273          2334  5                  END     ! Normal case
;  2274          2335  4              ELSE
;  2275          2336  5                  BEGIN   ! Double-wide or Double-wide/Double-high case
;  2276          2337  5                  LOCAL
;  2277          2338  5                      SINDEX;     ! Special index for DWDH
;  2278          2339  5                  $CALC_LEFT_INDEX ;
;  2279          2340  5                  $MARKIT (SINDEX, BORD_M_VERT);
;  2280          2341  5                  (PTRT [.SINDEX])<BORD_V_RIGHT,1> = 0; ! Delete "right"
;  2281          2342  4                  END;    ! Double-wide or Double-wide/Double-high case
;  2282          2343  4              SPOS = .SPOS + .WCB [WCB_W_NO_COLS] ;
;  2283          2344  4              WLN = .WLN + 1;
;  2284          2345  3              END;        ! Left column loop
;  2285          2346  3
;  2286          2347  3          IF .LEFT_COL LSS .PBCB [PBCB_W_FIRST_CHANGED_COL]
;  2287          2348  3          THEN
;  2288          2349  3              PBCB [PBCB_W_FIRST_CHANGED_COL] = .LEFT_COL;
;  2289          2350  3
;  2290          2351  2          END;    ! Left border
;  2291          2352  2
;  2292          2353  2      !+
;  2293          2354  2      ! If the column to the right of where the virtual display is pasted
;  2294          2355  2      ! exists in the window buffer, draw the right border.
;  2295          2356  2      !-
;  2296          2357  2          IF .RIGHT_COL LEQ .WCB [WCB_W_NO_COLS]
;  2297          2358  2          THEN
;  2298          2359  3              BEGIN   ! Right border
;  2299          2360  3              LOCAL
;  2300          2361  3                  WLN,            ! Line number in WCB buffers
;  2301          2362  3                  LCV : REF VECTOR [,BYTE]; ! Addr of line characteristics
;  2302          2363  3                                           ! vector in WCB
;  2303          2364  3              WLN = .PP [PP_W_FIRST_WCB_ROW];
;  2304          2365  3              LCV = .WCB [WCB_A_LINE_CHAR];
;  2305          2366  3              SPOS = .PP [PP_W_TO_INDEX] + .PP [PP_W_MOVE_LENGTH];
;  2306          2367  3              INCR I FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
;  2307          2368  3              DO
;  2308          2369  4                  BEGIN       ! Right column loop
;  2309          2370  4                  IF .LCV [.WLN] EQL 0 OR             ! If normal or
;  2310          2371  5                     (NOT .PBCB [PBCB_V_WIDE])        ! DWDH not supported
;  2311          2372  4                  THEN
;  2312          2373  5                      BEGIN   ! Normal case
;  2313          2374  5                      $MARKIT (SPOS, BORD_M_VERT ) ;
```

```
;  2314     2375  5                (PTRT [.SPOS])<BORD_V_LEFT,1> = 0; ! Knock down "left" seg.
;  2315     2376  5                END     ! Normal case
;  2316     2377  4            ELSE
;  2317     2378  3                BEGIN   ! Double-wide or Double-wide/Double-high case
;  2318     2379  5                LOCAL
;  2319     2380  5                    SINDEX;     ! Special index for DWDH
;  2320     2381  5                $CALC_RIGHT_INDEX ;
;  2321     2382  5                $MARKIT (SINDEX, BORD_M_VERT);
;  2322     2383  5                (PTRT [.SINDEX])<BORD_V_LEFT,1> = 0; ! Delete "left"
;  2323     2384  4                END;    ! Double-wide or Double-wide/Double-high case
;  2324     2385  4            SPOS = .SPOS + .WCB [WCB_W_NO_COLS] ;
;  2325     2386  4            WLN = .WLN + 1;
;  2326     2387  3            END;        ! Right column loop
;  2327     2388  3
;  2328     2389  3        IF .RIGHT_COL GTR .PBCB [PBCB_W_LAST_CHANGED_COL]
;  2329     2390  3        THEN
;  2330     2391  3            PBCB [PBCB_W_LAST_CHANGED_COL] = .RIGHT_COL;
;  2331     2392  3
;  2332     2393  2        END;    ! Right border
;  2333     2394  2
;  2334     2395  2    !+
;  2335     2396  2    ! Handle upper left-hand corner if position exists in buffer.
;  2336     2397  2    !-
;  2337     2398  2        IF .UPPER_ROW GEQ 1         AND     ! Row exists
;  2338     2399  2           .LEFT_COL  GEQ 1                 ! Col exists
;  2339     2400  2        THEN
;  2340     2401  3            BEGIN   ! Upper left-hand corner
;  2341     2402  3            SPOS = .PP [PP_W_TO_INDEX] - .WCB [WCB_W_NO_COLS] - 1;
;  2342     2403  3            $MARKIT (SPOS, BORD_M_ULCORN ) ;
;  2343     2404  2            END;    ! Upper left-hand corner
;  2344     2405  2
;  2345     2406  2    !+
;  2346     2407  2    ! Handle upper right-hand corner if position exists in buffer.
;  2347     2408  2    !-
;  2348     2409  2        IF .UPPER_ROW GEQ 1         AND             ! Row exists
;  2349     2410  2           .RIGHT_COL LEQ .WCB [WCB_W_NO_COLS]      ! Col exists
;  2350     2411  2        THEN
;  2351     2412  3            BEGIN   ! Upper right-hand corner
;  2352     2413  3            SPOS = .PP [PP_W_TO_INDEX] - .WCB [WCB_W_NO_COLS] +
;  2353     2414  3                       .PP [PP_W_MOVE_LENGTH] ;
;  2354     2415  3            $MARKIT (SPOS, BORD_M_URCORN ) ;
;  2355     2416  2            END;    ! Upper right-hand corner
;  2356     2417  2
;  2357     2418  2    !+
;  2358     2419  2    ! Handle lower left-hand corner if position exists in buffer.
;  2359     2420  2    !-
;  2360     2421  2        IF .LOWER_ROW LEQ .WCB [WCB_W_NO_ROWS]  AND ! Row exists
;  2361     2422  2           .LEFT_COL  GEQ 1                         ! Col exists
;  2362     2423  2        THEN
;  2363     2424  3            BEGIN   ! Lower left-hand corner
;  2364     2425  4            SPOS = .PP [PP_W_TO_INDEX] + (.PP [PP_W_ROWS_TO_MOVE] *
;  2365     2426  3                                          .WCB [WCB_W_NO_COLS]) - 1;
;  2366     2427  3            $MARKIT (SPOS, BORD_M_LLCORN ) ;
;  2367     2428  2            END;    ! Lower left-hand corner
;  2368     2429  2
;  2369     2430  2    !+
;  2370     2431  2    ! Handle lower right-hand corner if position exists in buffer.
```

```
;  2371          2432  2 !-
;  2372          2433  2     IF .LOWER_ROW LEQ .WCB [WCB_W_NO_ROWS]  AND ! Row exists
;  2373          2434  2        .RIGHT_COL LEQ .WCB [WCB_W_NO_COLS]      ! Col exists
;  2374          2435  2     THEN
;  2375          2436  3         BEGIN   ! Lower right-hand corner
;  2376          2437  3         SPOS = .PP [PP_W_TO_INDEX] +
;  2377          2438  3                   (.PP [PP_W_ROWS_TO_MOVE] * .WCB [WCB_W_NO_COLS]) +
;  2378          2439  3                   .PP [PP_W_MOVE_LENGTH] ;
;  2379          2440  3         $MARKIT (SPOS, BORD_M_LRCORN ) ;
;  2380          2441  2         END;    ! Lower right-hand corner
;  2381          2442  2
;  2382          2443  2 !+
;  2383          2444  2 ! Check to see if this border is labeled and if so, move the label text
;  2384          2445  2 ! into the approriate border area.
;  2385          2446  2 ! All the fields in the PP accessed below have been previously computed
;  2386          2447  2 ! at the time the virtual display was pasted to the pasteboard so
;  2387          2448  2 ! that they need not be recomputed each time through here.  The lengths
;  2388          2449  2 ! used in the various CH$MOVE's and CH$FILL's have been so computed
;  2389          2450  2 ! that if no parts of the label fit, the lengths used are zero.
;  2390          2451  2 !-
;  2391          2452  2     LDES = DCB [DCB_Q_LABEL_DESC];
;  2392          2453  2     IF .LDES [DSC$W_LENGTH] NEQ 0                ! If label exists...
;  2393          2454  2     THEN
;  2394          2455  3         BEGIN   ! Overall border label processing
;  2395          2456  3         CASE .DCB [DCB_B_LABEL_POS] FROM SMG$K_TOP TO SMG$K_RIGHT OF
;  2396          2457  3         SET
;  2397          2458  3             [SMG$K_TOP]:
;  2398          2459  4                 BEGIN   ! Label in top row
;  2399          2460  4                 !+
;  2400          2461  4                 ! If row where label lands is not on the pasteboard, no
;  2401          2462  4                 ! movement will be necessary.
;  2402          2463  4                 !-
;  2403          2464  4                 IF .UPPER_ROW GEQ 1     ! If row above present...
;  2404          2465  4                 THEN
;  2405          2466  5                     BEGIN       ! Top present
;  2406          2467  5
;  2407          2468  5                     CH$MOVE (
;  2408          2469  5                        .PP [PP_W_LABEL_BYTES_TO_MOVE],  ! Length
;  2409          2470  5                        .LDES [DSC$A_POINTER] + .PP [PP_W_SRC_LABEL_OFF],  ! Source
;  2410          2471  5                        PTRT [.PP [PP_W_DST_LABEL_OFF]]);  ! Dest
;  2411          2472  5
;  2412          2473  5                     CH$FILL (
;  2413          2474  5                        .DCB [DCB_B_LABEL_REND],          ! Fill
;  2414          2475  5                        .PP [PP_W_LABEL_BYTES_TO_MOVE],   ! Number
;  2415          2476  5                        PTRA [.PP [PP_W_DST_LABEL_OFF]]);  ! Dest
;  2416          2477  5
;  2417          2478  4                     END;        ! Top present
;  2418          2479  3                 END;    ! Label in top row
;  2419          2480  3
;  2420          2481  3             [SMG$K_BOTTOM]:
;  2421          2482  4                 BEGIN   ! Label in bottom row
;  2422          2483  4                 !+
;  2423          2484  4                 ! If row where label lands is not on the pasteboard, no
;  2424          2485  4                 ! movement will be necessary.
;  2425          2486  4                 !-
;  2426          2487  4                 IF .LOWER_ROW LEQ .WCB [WCB_W_NO_ROWS] ! If row below present...
;  2427          2488  4                 THEN
```

```
;  2428     2489  5                    BEGIN       ! Bottom present
;  2429     2490  5
;  2430     2491  5                    CH$MOVE (
;  2431     2492  5                        .PP [PP_W_LABEL_BYTES_TO_MOVE],  ! Length
;  2432     2493  5                        .LDES [DSC$A_POINTER] + .PP [PP_W_SRC_LABEL_OFF],  ! Source
;  2433     2494  5                        PTRT [.PP [PP_W_DST_LABEL_OFF]]);  ! Dest
;  2434     2495  5
;  2435     2496  5                    CH$FILL (
;  2436     2497  5                        .DCB [DCB_B_LABEL_REND],          ! Fill
;  2437     2498  5                        .PP [PP_W_LABEL_BYTES_TO_MOVE],   ! Number
;  2438     2499  5                        PTRA [.PP [PP_W_DST_LABEL_OFF]]);  ! Dest
;  2439     2500  5
;  2440     2501  4                    END;        ! Bottom present
;  2441     2502  3                END;    ! Label in bottom row
;  2442     2503  3
;  2443     2504  3            [SMG$K_LEFT]:
;  2444     2505  4                BEGIN   ! Label in left column
;  2445     2506  4                !+
;  2446     2507  4                ! If column where label lands is not on the pasteboard,
;  2447     2508  4                ! no movement will be necessary.
;  2448     2509  4                !-
;  2449     2510  4                IF .LEFT_COL GEQ 1      ! If column to left present...
;  2450     2511  4                THEN
;  2451     2512  5                    BEGIN       ! Left present
;  2452     2513  5                    LOCAL
;  2453     2514  5                        WLN,                              ! Line # in WCB buffer
;  2454     2515  5                        LCV : REF VECTOR [,BYTE],         ! Addr of line
;  2455     2516  5                                                          ! characteristics vect.
;  2456     2517  5                                                          ! in WCB
;  2457     2518  5                    I;      ! Local index
;  2458     2519  6            WLN = (.PP [PP_W_DST_LABEL_OFF] /
;  2459     2520  5                   .WCB [WCB_W_NO_COLS]) + 1;
;  2460     2521  5            LCV = .WCB [WCB_A_LINE_CHAR];
;  2461     2522  5            SPOS = .PP [PP_W_DST_LABEL_OFF];
;  2462     2523  5            I = 0;
;  2463     2524  5            UNTIL .I EQL .PP [PP_W_LABEL_BYTES_TO_MOVE]
;  2464     2525  5            DO
;  2465     2526  6                BEGIN   ! Character by character movement
;  2466     2527  6                IF .LCV [.WLN] EQL 0 OR          ! If normal or
;  2467     2528  7                   (NOT .PBCB [PBCB_V_WIDE])     ! DWDH not supported
;  2468     2529  6                THEN
;  2469     2530  7                    BEGIN ! Normal case
;  2470     2531  8                    PTRT [.SPOS] = .(.LDES [DSC$A_POINTER] +
;  2471     2532  7                                     .PP [PP_W_SRC_LABEL_OFF] + .I);
;  2472     2533  7                    PTRA [.SPOS] = .DCB [DCB_B_LABEL_REND];
;  2473     2534  7                    END   ! Normal case
;  2474     2535  6                ELSE
;  2475     2536  7                    BEGIN ! Special case
;  2476     2537  7                    LOCAL
;  2477     2538  7                        SINDEX;
;  2478     2539  7                    $CALC_LEFT_INDEX ;
;  2479     2540  8                    PTRT [.SINDEX] = .(.LDES [DSC$A_POINTER] +
;  2480     2541  7                                       .PP [PP_W_SRC_LABEL_OFF] + .I);
;  2481     2542  7                    PTRA [.SINDEX] = .DCB [DCB_B_LABEL_REND];
;  2482     2543  6                    END;  ! Special case
;  2483     2544  6
;  2484     2545  6                SPOS = .SPOS + .WCB [WCB_W_NO_COLS];
```

```
;  2485     2546  6                        I = .I+1;
;  2486     2547  6                        WLN = .WLN + 1;
;  2487     2548  5                        END;    ! Character by character movement
;  2488     2549  4                    END;        ! Left present
;  2489     2550  3                END;    ! Label in left column
;  2490     2551  3
;  2491     2552  3            [SMG$K_RIGHT]:
;  2492     2553  4                BEGIN   ! Label in right column
;  2493     2554  4                !+
;  2494     2555  4                ! If column where label lands is not on the pasteboard,
;  2495     2556  4                ! no movement will be necessary.
;  2496     2557  4                !-
;  2497     2558  4                IF .RIGHT_COL LEQ .WCB [WCB_W_NO_COLS] ! If column to right present...
;  2498     2559  4                THEN
;  2499     2560  5                    BEGIN       ! Right present
;  2500     2561  5                    LOCAL
;  2501     2562  5                        WLN,                        ! Line # in WCB buffer
;  2502     2563  5                        LCV : REF VECTOR [,BYTE],   ! Addr of line
;  2503     2564  5                                                    ! characteristics vect.
;  2504     2565  5                                                    ! in WCB
;  2505     2566  5                        I;      ! Local index
;  2506     2567  5                    I = 0;
;  2507     2568  6                    WLN = (.PP [PP_W_DST_LABEL_OFF] /
;  2508     2569  5                           .WCB [WCB_W_NO_COLS]) + 1;
;  2509     2570  5                    LCV = .WCB [WCB_A_LINE_CHAR];
;  2510     2571  5                    SPOS = .PP [PP_W_DST_LABEL_OFF];
;  2511     2572  5                    UNTIL .I EQL .PP [PP_W_LABEL_BYTES_TO_MOVE]
;  2512     2573  5                    DO
;  2513     2574  6                        BEGIN   ! Character by character movement
;  2514     2575  6                        IF .LCV [.WLN] EQL 0 OR         ! If normal or
;  2515     2576  7                           (NOT .PBCB [PBCB_V_WIDE])    ! DWDH not supported
;  2516     2577  6                        THEN
;  2517     2578  7                            BEGIN       ! Normal case
;  2518     2579  8                            PTRT [.SPOS] = .(.LDES [DSC$A_POINTER] +
;  2519     2580  7                                            .PP [PP_W_SRC_LABEL_OFF] + .I);
;  2520     2581  7                            PTRA [.SPOS] = .DCB [DCB_B_LABEL_REND];
;  2521     2582  7                            END         ! Normal case
;  2522     2583  6                        ELSE
;  2523     2584  7                            BEGIN       ! Special case
;  2524     2585  7                            LOCAL
;  2525     2586  7                                SINDEX;
;  2526     2587  7                            $CALC_RIGHT_INDEX ;
;  2527     2588  8                            PTRT [.SINDEX] = .(.LDES [DSC$A_POINTER] +
;  2528     2589  7                                            .PP [PP_W_SRC_LABEL_OFF] + .I);
;  2529     2590  7                            PTRA [.SINDEX] = .DCB [DCB_B_LABEL_REND];
;  2530     2591  6                            END;        ! Special case
;  2531     2592  6                        SPOS = .SPOS + .WCB [WCB_W_NO_COLS];
;  2532     2593  6                        I = .I+1;
;  2533     2594  6                        WLN = .WLN + 1;
;  2534     2595  5                        END;    ! Character by character movement
;  2535     2596  4                    END;        ! Right present
;  2536     2597  3                END;    ! Label in right column
;  2537     2598  3
;  2538     2599  3            [OUTRANGE]:
;  2539     2600  3                RETURN (SMG$_FATERRLIB);
;  2540     2601  3        TES;
;  2541     2602  3
```

```
: 2542          2603  2          END;      ! Overall border label processing
: 2543          2604  2      RETURN (SS$_NORMAL);
: 2544          2605  1      END;                    ! End of routine SMG$$DRAW_BORDER


                                        0FFC 00000          .ENTRY  SMG$$DRAW_BORDER, Save R2,R3,R4,R5,R6,R7,-      ; 2091
                                                                    R8,R9,R10,R11
                                5E          28  C2 00002    SUBL2   #40, SP
                50      0C      AC          08  C1 00005    ADDL3   #8, WCB, R0                                     ; 2208
                                59          60  D0 0000A    MOVL    (R0), PTRT
                50      0C      AC          0C  C1 0000D    ADDL3   #12, WCB, R0                                    ; 2209
                                5A          60  D0 00012    MOVL    (R0), PTRA
                                5B    08    AC  D0 00015    MOVL    PP, R11                                         ; 2210
                        10      AE    14    AB  D0 00019    MOVL    20(R11), PBCB
                50      0C      AC          2C  C1 0001E    ADDL3   #44, WCB, R0                                    ; 2211
                        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)                                  ; 2221
                                            03  13 00031    BEQL    1$
                                      22    AB  B5 00033    TSTW    34(R11)                                         ; 2222
                                            03  12 00036 1$: BNEQ    2$
                                        04C0    31 00038    BRW     64$
                                5A          57  18 AB 32 0003B 2$: CVTWL   24(R11), UPPER_ROW                      ; 2231
                                            57  D7 0003F    DECL    UPPER_ROW
                                52          18  AB 32 00041 CVTWL   24(R11), R2                                    ; 2232
                                            03  14 00045    BGTR    3$
                                52          01  D0 00047    MOVL    #1, R2
                        58      52          04  AE C1 0004A 3$: ADDL3   4(SP), R2, LOWER_ROW
                18      AE              1A  AB 32 0004F    CVTWL   26(R11), LEFT_COL                               ; 2233
                                            18  AE D7 00054    DECL    LEFT_COL
                                52          1A  AB 32 00057 CVTWL   26(R11), R2                                    ; 2234
                                            03  14 0005B    BGTR    4$
                                52          01  D0 0005D    MOVL    #1, R2
                14      AE              22  AB 3C 00060 4$: MOVZWL  34(R11), 20(SP)
                1C      AE              14  BE42 9E 00065   MOVAB   @20(SP)[R2], RIGHT_COL
                                        24  AE D4 0006B    CLRL    36(SP)                                          ; 2240
                                            57  D5 0006E    TSTL    UPPER_ROW
                                            03  14 00070    BGTR    5$
                                        0097    31 00072    BRW     11$
                                        24  AE D6 00075 5$: INCL    36(SP)
                                50          08  AE D0 00078 MOVL    WCB_LCV, R0                                     ; 2248
                                        6740    95 0007C    TSTB    (UPPER_ROW)[R0]
                                            3E  13 0007F    BEQL    6$
                        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                                      ; 2252
                        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)                                 ; 2254
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                                 ; 2255
                                                6740  94 000BC          CLRB    (UPPER_ROW)[R0]
                                        50      20  AB  3C 000BF 6$:      MOVZWL  32(R11), R0                                 ; 2258
                                51      0C      AC      06 C1 000C3          ADDL3   #6, WCB, R1
                                        56          61  3C 000C8          MOVZWL  (R1), SPOS
                                56      50          56  C3 000CB          SUBL3   SPOS, R0, SPOS
                                                    50  D4 000CF          CLRL    I                                           ; 2262
                                                    18  11 000D1          BRB     10$
                                                  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]                           ; 2263
                                E3          50      14  AE  F3 000EB 10$:     AOBLEQ  20(SP), I, 7$                       ; 2259
                                50      10  AE  000000A8    8F  C1 000F0          ADDL3   #168, PBCB, R0                      ; 2267
                        57      60          10              00  EC 000F9          CMPV    #0, #16, (R0), UPPER_ROW
                                                    0C  15 000FE          BLEQ    11$
                                50      10  AE  000000A8    8F  C1 00100          ADDL3   #168, PBCB, R0                      ; 2269
                                            60              57  B0 00109          MOVW    UPPER_ROW, (R0)
                                                    20  AE  D4 0010C 11$:     CLRL    32(SP)                              ; 2277
                                50      0C      AC          02  C1 0010F          ADDL3   #2, WCB, R0
                        58      60          10              00  ED 00114          CMPZV   #0, #16, (R0), LOWER_ROW
                                                        03  18 00119          BGEQ    12$
                                                  0098  31 0011B          BRW     18$
                                                    20  AE  D6 0011E 12$:     INCL    32(SP)
                                        50      08  AE  D0 00121          MOVL    WCB_LCV, R0                                 ; 2285
                                                6840  95 00125          TSTB    (LOWER_ROW)[R0]
                                                    3C  13 00128          BEQL    13$
                                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                          ; 2289
                                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)                       ; 2291
                        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                                 ; 2292
                                                6840  94 00163          CLRB    (LOWER_ROW)[R0]
                                50      0C      AC          06  C1 00166 13$:     ADDL3   #6, WCB, R0                         ; 2296
                                            51              60  3C 0016B          MOVZWL  (R0), R1
                                            51      04  AE  C4 0016E          MULL2   4(SP), R1
                                            56      20  AB  3C 00172          MOVZWL  32(R11), SPOS                       ; 2295
                                            56              51  C0 00176          ADDL2   R1, SPOS
                                                            50  D4 00179          CLRL    I                                   ; 2300
                                                            18  11 0017B          BRB     17$
                                                  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$:          BICB2   #2, (SPOS)+[PTRT]                    2301
                          E3           50    14   AE  F3 00195 17$:          AOBLEQ  20(SP), I, 14$                       2297
                          50     10    AE 000000AA  8F  C1 0019A             ADDL3   #170, PBCB, R0                       2305
                      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                       2307
                                 60                 58  B0 001B3             MOVW    LOWER_ROW, (R0)
                                                    55  D4 001B6 18$:          CLRL    R5                                   2315
                                       18    AE     D5  001B8                TSTL    LEFT_COL
                                                    03  14 001BB             BGTR    19$
                                                00A2    31 001BD             BRW     29$
                                                    55  D6 001C0 19$:          INCL    R5
                                       50    2F     AB  3C 001C2             MOVZWL  47(R11), WLN                         2322
                                       53    0C     AE  D0 001C6             MOVL    12(SP), LCV                          2323
                                       56    20     AB  3C 001CA             MOVZWL  32(R11), SPOS                        2324
                                                    56  D7 001CE             DECL    SPOS
                                                    54  D4 001D0             CLRL    I                                    2343
                                                    6B  11 001D2             BRB     28$
                                                6043    95 001D4 20$:          TSTB    (WLN)[LCV]                           2328
                                                    0C  13 001D7             BEQL    21$
                          51     10    AE 000000FA  8F  C1 001D9             ADDL3   #250, PBCB, R1                       2329
                                 1A                 61  E8 001E2             BLBS    (R1), 24$
                                                664A    95 001E5 21$:          TSTB    (SPOS)[PTRA]                         2332
                                                    06  18 001E8             BGEQ    22$
                                 6649               0A  88 001EA             BISB2   #10, (SPOS)[PTRT]
                                                    09  11 001EE             BRB     23$
                                 6649               0A  90 001F0 22$:          MOVB    #10, (SPOS)[PTRT]
                                 664A        80     8F  90 001F4             MOVB    #-128, (SPOS)[PTRA]
                                 6649               01  8A 001F9 23$:          BICB2   #1, (SPOS)[PTRT]                     2333
                                                    33  11 001FD             BRB     27$                                  2328
                                   51        FF     A0  9E 001FF 24$:          MOVAB   -1(R0), R1                           2338
                          57       0C     AC          06  C1 00203             ADDL3   #6, WCB, R7
                                          52           67  3C 00208             MOVZWL  (R7), R2
                                          51           52  C4 0020B             MULL2   R2, R1
                                          52     33    AB  3C 0020E             MOVZWL  51(R11), R2
                                          52           02  C6 00212             DIVL2   #2, R2
                                          51     FF A241   9E 00215             MOVAB   -1(R2)[R1], SINDEX
                                                614A    95 0021A             TSTB    (SINDEX)[PTRA]                       2340
                                                    06  18 0021D             BGEQ    25$
                                 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]                   2341
                          52       0C     AC          06  C1 00232 27$:          ADDL3   #6, WCB, R2                          2343
                                          51           62  3C 00237             MOVZWL  (R2), R1
                                          56           51  C0 0023A             ADDL2   R1, SPOS
                                                       50  D6 0023D             INCL    WLN                                  2344
                          90           54    04   AE  F3 0023F 28$:          AOBLEQ  4(SP), I, 20$                        2325
                          50     10    AE 000000AC  8F  C1 00244             ADDL3   #172, PBCB, R0                       2347
                  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                       2349
                                 60           18    AE  B0 0025E             MOVW    LEFT_COL, (R0)
                          50     0C    AC           06  C1 00262 29$:          ADDL3   #6, WCB, R0                          2357
                                 18    AE           60  3C 00267             MOVZWL  (R0), 24(SP)
```

```
                                   52  D4 0026B        CLRL    R2                                   ;
              18   AE        1C    AE  D1 0026D        CMPL    RIGHT_COL, 24(SP)                    ;
                                   03  15 00272        BLEQ    30$                                  ;
                                 009E  31 00274        BRW     40$                                  ;
                                   52  D6 00277 30$:   INCL    R2                                   ;
                    50        2F   AB  3C 00279        MOVZWL  47(R11), WLN                         ; 2364
                    53        0C   AE  D0 0027D        MOVL    12(SP), LCV                          ; 2365
                    56        20   AB  3C 00281        MOVZWL  32(R11), SPOS                        ; 2366
                    56        14   AE  C0 00285        ADDL2   20(SP), SPOS                         ;
                                   54  D4 00289        CLRL    I                                    ; 2374
                                   65  11 0028B        BRB     39$                                  ;
                                 6043  95 0028D 31$:   TSTB    (WLN)[LCV]                           ; 2370
                                   0C  13 00290        BEQL    32$                                  ;
         51         10   AE 000000FA  8F  C1 00292     ADDL3   #250, PBCB, R1                       ; 2371
                    1A             61  E8 0029B        BLBS    (R1), 35$                            ;
                                 664A  95 0029E 32$:   TSTB    (SPOS)[PTRA]                         ; 2374
                                   06  18 002A1        BGEQ    33$                                  ;
                  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]                     ; 2375
                                   34  11 002B6        BRB     38$                                  ; 2370
                    57        FF   A0  9E 002B8 35$:   MOVAB   -1(R0), R7                           ; 2380
                    57        18   AE  C4 002BC        MULL2   24(SP), R7                           ;
                    51        33   AB  3C 002C0        MOVZWL  51(R11), R1                          ;
         58         14   AE        01  C1 002C4        ADDL3   #1, 20(SP), R8                       ;
                    51             58  C0 002C9        ADDL2   R8, R1                               ;
                    51             02  C6 002CC        DIVL2   #2, R1                               ;
                    51     FF A147  9E 002CF           MOVAB   -1(R1)[R7], SINDEX                   ;
                                 614A  95 002D4        TSTB    (SINDEX)[PTRA]                       ; 2382
                                   06  18 002D7        BGEQ    36$                                  ;
                  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]                   ; 2383
                    56        18   AE  C0 002EC 38$:   ADDL2   24(SP), SPOS                         ; 2385
                                   50  D6 002F0        INCL    WLN                                  ; 2386
         96         54        04   AE  F3 002F2 39$:   AOBLEQ  4(SP), I, 31$                        ; 2367
         50         10   AE 000000AE  8F  C1 002F7     ADDL3   #174, PBCB, R0                       ; 2389
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                       ; 2391
                    60        1C   AE  B0 00311        MOVW    RIGHT_COL, (R0)                      ;
                    48        24   AE  E9 00315 40$:   BLBC    36(SP), 44$                          ; 2398
                    20             55  E9 00319        BLBC    R5, 42$                              ; 2399
                    51        20   AB  3C 0031C        MOVZWL  32(R11), R1                          ; 2402
                    51        18   AE  C2 00320        SUBL2   24(SP), R1                           ;
                    56        FF   A1  9E 00324        MOVAB   -1(R1), SPOS                         ;
                                 664A  95 00328        TSTB    (SPOS)[PTRA]                         ; 2403
                                   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$                          ; 2409
```

```
                    21          52  E9 00340       BLBC    R2, 44$                                     ; 2410
                    51    20    AB  3C 00343       MOVZWL  32(R11), R1                                 ; 2413
                    51    18    AE  C2 00347       SUBL2   24(SP), R1                                  ;
          56        51    14    AE  C1 0034B       ADDL3   20(SP), R1, SPOS                            ; 2414
                              664A  95 00350       TSTB    (SPOS)[PTRA]                                ; 2415
                                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$                                 ; 2421
                    23          55  E9 00368       BLBC    R5, 46$                                     ; 2422
                    50    20    AB  3C 0036B       MOVZWL  32(R11), R0                                 ; 2425
          51    04  AE    18    AE  C5 0036F       MULL3   24(SP), 4(SP), R1                           ; 2426
                    56    FF  A140  9E 00375       MOVAB   -1(R1)[R0], SPOS                            ;
                              664A  95 0037A       TSTB    (SPOS)[PTRA]                                ; 2427
                                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$                                 ; 2433
                    26          52  E9 00392       BLBC    R2, 48$                                     ; 2434
                    50    20    AB  3C 00395       MOVZWL  32(R11), R0                                 ; 2437
          51    04  AE    18    AE  C5 00399       MULL3   24(SP), 4(SP), R1                           ; 2438
                    50          51  C0 0039F       ADDL2   R1, R0                                      ; 2437
          56        50    14    AE  C1 003A2       ADDL3   20(SP), R0, SPOS                            ; 2439
                              664A  95 003A7       TSTB    (SPOS)[PTRA]                                ; 2440
                                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                                     ; 2452
                    57    08    A8  9E 003BF       MOVAB   8(R8), LDES                                 ;
                                67  B5 003C3       TSTW    (LDES)                                      ; 2453
                                3D  13 003C5       BEQL    53$                                         ;
          03        00    31    A8  8F 003C7       CASEB   49(R8), #0, #3                              ; 2456
    00B1  003B    0016        0010     003CC 49$:  .WORD   50$-49$,-                                   ;
                                                           51$-49$,-                                   ;
                                                           54$-49$,-                                   ;
                                                           59$-49$                                     ;
          50 00000000G          8F  D0 003D4       MOVL    #SMG$_FATERRLIB, R0                         ; 2600
                                   04 003DB       RET                                                 ;
                    24    24    AE  E9 003DC 50$:  BLBC    36(SP), 53$                                 ;
                                04  11 003E0       BRB     52$                                         ; 2470
                    1E    20    AE  E9 003E2 51$:  BLBC    32(SP), 53$                                 ; 2487
                    51    26    AB  3C 003E6 52$:  MOVZWL  38(R11), R1                                 ; 2493
                    51    04    A7  C0 003EA       ADDL2   4(LDES), R1                                 ;
              24    AE    28    AB  3C 003EE       MOVZWL  40(R11), 36(SP)                             ; 2494
        24 BE49     61    24    AB  28 003F3       MOVC3   36(R11), (R1), @36(SP)[PTRT]                ;
 24  AB  33  A8     6E          00  2C 003FA       MOVC5   #0, (SP), 51(R8), 36(R11), @36(SP)[PTRA]    ; 2499
                          24  BE4A     00401                                                       ;
                              00F4  31 00404 53$:  BRW     64$                                         ; 2456
                    FA          55  E9 00407 54$:  BLBC    R5, 53$                                     ; 2510
                    52    28    AB  3C 0040A       MOVZWL  40(R11), R2                                 ; 2520
                    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                               ; 2521
                                56       28    AB  3C 0041A          MOVZWL  40(R11), SPOS                             ; 2522
                                         54    D4  0041E             CLRL    I                                         ; 2523
     54        24    AB           10             00  ED 00420 55$:    CMPZV   #0, #16, 36(R11), I                       ; 2524
                                         DC    13  00426             BEQL    53$                                       ;
                                               6043 95 00428         TSTB    (WLN)[LCV]                                ; 2527
                                         0C    13  0042B             BEQL    56$                                       ;
                    51        10    AE 000000FA    8F C1 0042D       ADDL3   #250, PBCB, R1                            ; 2528
                                14             61  E8 00436         BLBS    (R1), 57$                                 ;
                                51       26    AB  3C 00439 56$:    MOVZWL  38(R11), R1                               ; 2532
                                51       04    A7  C0 0043D         ADDL2   4(LDES), R1                               ;
                              6649             6441 90 00441         MOVB    (I)[R1], (SPOS)[PTRT]                     ; 2531
                              664A       33    A8  90 00446         MOVB    51(R8), (SPOS)[PTRA]                      ; 2533
                                         26    11  0044B             BRB     58$                                       ; 2527
                                51       FF    A0  9E 0044D 57$:    MOVAB   -1(R0), R1                                ; 2538
                                51       18    AE  C4 00451         MULL2   24(SP), R1                                ;
                                52       33    AB  3C 00455         MOVZWL  51(R11), R2                               ;
                                52             02  C6 00459         DIVL2   #2, R2                                    ;
                                52       FF  A241  9E 0045C         MOVAB   -1(R2)[R1], SINDEX                        ;
                                51       26    AB  3C 00461         MOVZWL  38(R11), R1                               ; 2541
                                51       04    A7  C0 00465         ADDL2   4(LDES), R1                               ;
                              6249             6441 90 00469         MOVB    (I)[R1], (SINDEX)[PTRT]                   ; 2540
                              624A       33    A8  90 0046E         MOVB    51(R8), (SINDEX)[PTRA]                    ; 2542
                                56       18    AE  C0 00473 58$:    ADDL2   24(SP), SPOS                              ; 2545
                                         54    D6  00477             INCL    I                                         ; 2546
                                         50    D6  00479             INCL    WLN                                       ; 2547
                                         A3    11  0047B             BRB     55$                                       ; 2524
                                7B             52  E9 0047D 59$:    BLBC    R2, 64$                                   ; 2558
                                         54    D4  00480             CLRL    I                                         ; 2567
                                52       28    AB  3C 00482         MOVZWL  40(R11), R2                               ; 2569
                                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                               ; 2570
                                56       28    AB  3C 00492         MOVZWL  40(R11), SPOS                             ; 2571
     54        24    AB           10             00  ED 00496 60$:    CMPZV   #0, #16, 36(R11), I                       ; 2572
                                         5D    13  0049C             BEQL    64$                                       ;
                                               6043 95 0049E         TSTB    (WLN)[LCV]                                ; 2575
                                         0C    13  004A1             BEQL    61$                                       ;
                    51        10    AE 000000FA    8F C1 004A3       ADDL3   #250, PBCB, R1                            ; 2576
                                14             61  E8 004AC         BLBS    (R1), 62$                                 ;
                                51       26    AB  3C 004AF 61$:    MOVZWL  38(R11), R1                               ; 2580
                                51       04    A7  C0 004B3         ADDL2   4(LDES), R1                               ;
                              6649             6441 90 004B7         MOVB    (I)[R1], (SPOS)[PTRT]                     ; 2579
                              664A       33    A8  90 004BC         MOVB    51(R8), (SPOS)[PTRA]                      ; 2581
                                         2E    11  004C1             BRB     63$                                       ; 2575
                                51       FF    A0  9E 004C3 62$:    MOVAB   -1(R0), R1                                ; 2586
                                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                               ; 2589
                                51       04    A7  C0 004E3         ADDL2   4(LDES), R1                               ;
                              6249             6441 90 004E7         MOVB    (I)[R1], (SINDEX)[PTRT]                   ; 2588
                              624A       33    A8  90 004EC         MOVB    51(R8), (SINDEX)[PTRA]                    ; 2590
```

```
                                56       18   AE  C0 004F1 63$:    ADDL2   24(SP), SPOS                          ; 2592
                                              54  D6 004F5         INCL    I                                      ; 2593
                                              50  D6 004F7         INCL    WLN                                    ; 2594
                                              9B  11 004F9         BRB     60$                                    ; 2572
                                50            01  D0 004FB 64$:    MOVL    #1, R0                                 ; 2604
                                                  04 004FE         RET                                            ; 2605

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

; 2545          2606  1 !<BLF/PAGE>
```

```
;  2547          2607  1 %SBTTL 'SMG$$FILL_WINDOW_BUFFER - Fill window with virtual displays'
;  2548          2608  1 GLOBAL ROUTINE SMG$$FILL_WINDOW_BUFFER (
;  2549          2609  1                                       PP : REF BLOCK [,BYTE]
;  2550          2610  1                                       ) =
;  2551          2611  1 !++
;  2552          2612  1 ! FUNCTIONAL DESCRIPTION:
;  2553          2613  1 !
;  2554          2614  1 !       This procedure rebuilds the portions of the window buffer that
;  2555          2615  1 !       need to be changed because some virtual display has changed.
;  2556          2616  1 !       Rebuilding takes place from the given pasting packet down the
;  2557          2617  1 !       chain to the most-recently pasted pasting packet.
;  2558          2618  1 !
;  2559          2619  1 ! CALLING SEQUENCE:
;  2560          2620  1 !
;  2561          2621  1 !       ret_status.wlc.v = SMG$$FILL_WINDOW_BUFFER ( PP.rab.r)
;  2562          2622  1 !
;  2563          2623  1 ! FORMAL PARAMETERS:
;  2564          2624  1 !
;  2565          2625  1 !
;  2566          2626  1 !       PP.rab.r                Address of pasting packet which
;  2567          2627  1 !                               determines the first display to be
;  2568          2628  1 !                               repainted.
;  2569          2629  1 !
;  2570          2630  1 ! IMPLICIT INPUTS:
;  2571          2631  1 !
;  2572          2632  1 !       NONE
;  2573          2633  1 !
;  2574          2634  1 ! IMPLICIT OUTPUTS:
;  2575          2635  1 !
;  2576          2636  1 !       NONE
;  2577          2637  1 !
;  2578          2638  1 ! COMPLETION STATUS:
;  2579          2639  1 !
;  2580          2640  1 !       SS$_NORMAL      Normal successful completion
;  2581          2641  1 !
;  2582          2642  1 ! SIDE EFFECTS:
;  2583          2643  1 !
;  2584          2644  1 !       NONE
;  2585          2645  1 !--
;  2586          2646  1
;  2587          2647  2     BEGIN
;  2588          2648  2     LOCAL
;  2589          2649  2         STATUS,                         ! Status of subr. calls
;  2590          2650  2         PBCB : REF BLOCK [,BYTE],       ! Address of pasteboard control
;  2591          2651  2                                         ! block
;  2592          2652  2
;  2593          2653  2         WCB : REF BLOCK [,BYTE],        ! Addr of current WCB
;  2594          2654  2
;  2595          2655  2         CURR_PP : REF BLOCK [,BYTE];    ! Addr of 2 longwords that form
;  2596          2656  2                                         ! queue header in PP currently
;  2597          2657  2                                         ! under inspection.
;  2598          2658  2
;  2599          2659  2     PBCB = .PP   [PP_A_PBCB_ADDR];
;  2600          2660  2     WCB  = .PBCB [PBCB_A_WCB];
;  2601          2661  2
;  2602          2662  2 !+
;  2603          2663  2 ! Change packet address to address of queue header.
```

```
;  2604          2664  2 !-
;  2605          2665  2     CURR_PP = .PP + PP_PBCB_QUEUE_OFFSET; ! Start with specified packet
;  2606          2666  2
;  2607          2667  2 !+
;  2608          2668  2 ! Loop for all pasting packets starting with this one to the last-pasted
;  2609          2669  2 ! one...
;  2610          2670  2 !-
;  2611          2671  2     WHILE .CURR_PP NEQ PBCB [PBCB_A_PP_NEXT]
;  2612          2672  2     DO
;  2613          2673  3         BEGIN   ! For all displays that need to be rewritten
;  2614          2674  3         LOCAL
;  2615          2675  3             PP_BASE : REF BLOCK [,BYTE],        ! Base address of the PP
;  2616          2676  3             DCB : REF BLOCK [,BYTE]; ! Current virtual display that
;  2617          2677  3                                      ! needs to be repainted.
;  2618          2678  3
;  2619          2679  3         PP_BASE = .CURR_PP - PP_PBCB_QUEUE_OFFSET;
;  2620          2680  3                                 ! Since the queue headers for this part
;  2621          2681  3                                 ! of the chain are not at relative 0 in
;  2622          2682  3                                 ! the pasting packet.
;  2623          2683  3         DCB = .PP_BASE [PP_A_DCB_ADDR]; ! DCB addr of this pairing
;  2624          2684  3
;  2625          2685  4         IF NOT (STATUS = SMG$$MOVE_TEXT_TO_WINDOW_BUF (.PP_BASE))
;  2626          2686  3         THEN
;  2627          2687  3             RETURN (.STATUS);
;  2628          2688  3
;  2629          2689  3         !+
;  2630          2690  3         ! If bordered, move the border characters into the text buffer
;  2631          2691  3         ! and set the bit in the attribute bytes to indicate a border
;  2632          2692  3         ! element.
;  2633          2693  3         !-
;  2634          2694  3         IF .DCB [DCB_V_BORDERED]
;  2635          2695  3         THEN
;  2636          2696  4             IF NOT (STATUS = SMG$$DRAW_BORDER ( .DCB, .PP_BASE, .WCB))
;  2637          2697  3             THEN
;  2638          2698  3                 RETURN (.STATUS);
;  2639          2699  3
;  2640          2700  3         !+
;  2641          2701  3         ! Update changed area fields in the PBCB based on latest
;  2642          2702  3         ! contribution.
;  2643          2703  3         !-
;  2644          2704  3         IF .PP_BASE [PP_W_ROWS_TO_MOVE] NEQ 0
;  2645          2705  3         THEN
;  2646          2706  4             BEGIN
;  2647          2707  4             IF .PP_BASE [PP_W_FIRST_WCB_ROW] LSS .PBCB [PBCB_W_FIRST_CHANGED_ROW]
;  2648          2708  4             THEN
;  2649          2709  4                 PBCB [PBCB_W_FIRST_CHANGED_ROW] = .PP_BASE [PP_W_FIRST_WCB_ROW];
;  2650          2710  4
;  2651          2711  4             IF .PP_BASE [PP_W_LAST_WCB_ROW] GTR .PBCB [PBCB_W_LAST_CHANGED_ROW]
;  2652          2712  4             THEN
;  2653          2713  4                 PBCB [PBCB_W_LAST_CHANGED_ROW] = .PP_BASE [PP_W_LAST_WCB_ROW];
;  2654          2714  4
;  2655          2715  4             IF .PP_BASE [PP_W_FIRST_WCB_COL] LSS .PBCB [PBCB_W_FIRST_CHANGED_COL]
;  2656          2716  4             THEN
;  2657          2717  4                 PBCB [PBCB_W_FIRST_CHANGED_COL] = .PP_BASE [PP_W_FIRST_WCB_COL];
;  2658          2718  4
;  2659          2719  4             IF .PP_BASE [PP_W_LAST_WCB_COL] GTR .PBCB [PBCB_W_LAST_CHANGED_COL]
;  2660          2720  4             THEN
```

```
;  2661          2721  4                 PBCB [PBCB_W_LAST_CHANGED_COL] = .PP_BASE [PP_W_LAST_WCB_COL];
;  2662          2722  4
;  2663          2723  3             END;
;  2664          2724  3
;  2665          2725  3         !+
;  2666          2726  3         ! Walk this chain backwards, from the packet we started with
;  2667          2727  3         ! back to the head of the chain -- since the most recently
;  2668          2728  3         ! pasted displays are at the head of the chain.
;  2669          2729  3         !-
;  2670          2730  3         CURR_PP = .PP_BASE [PP_A_PREV_PBCB];
;  2671          2731  2         END;    ! For all displays that need to be rewritten
;  2672          2732  2
;  2673          2733  2
;  2674          2734  2     RETURN ( SMG$$MIN_UPD ( .PBCB ));   ! Cause window buffer to be
;  2675          2735  2                                         ! output
;  2676          2736  2
;  2677          2737  1     END;                  ! End of routine SMG$$FILL_WINDOW_BUFFER



                                    007C 00000          .ENTRY  SMG$$FILL_WINDOW_BUFFER, Save R2,R3,R4,R5,- ; 2608
                                                                R6                                          :
                              52    04   AC  D0 00002          MOVL    PP, R2                                 ; 2659
                              54    14   A2  D0 00006          MOVL    20(R2), PBCB                           :
                              56    08   A4  D0 0000A          MOVL    8(PBCB), WCB                           ; 2660
                              53    08   A2  9E C000E          MOVAB   8(R2), CURR_PP                         ; 2665
                              54         53  D1 00012 1$:     CMPL    CURR_PP, PBCB                          ; 2671
                                         7B  13 00015          BEQL    7$                                     :
                              52    F8   A3  9E 00017          MOVAB   -8(R3), PP_BASE                        ; 2679
                              55    10   A2  D0 0001B          MOVL    16(PP_BASE), DCB                       ; 2683
                                         52  DD 0001F          PUSHL   PP_BASE                                ; 2685
                         0000V  CF           01  FB 00021          CALLS   #1, SMG$$MOVE_TEXT_TO_WINDOW_BUF
                                72           50  E9 00026          BLBC    STATUS, 8$
                                0E     2F    A5  E9 00029          BLBC    47(DCB), 2$                            ; 2694
                                       0044  8F  BB 0002D          PUSHR   #^M<R2,R6>                             ; 2696
                                         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$:      TSTW    28(PP_BASE)                            ; 2704
                                         4C  13 0003E          BEQL    6$
                                51     00A8  C4  32 00040          CVTWL   168(PBCB), R1                          ; 2707
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)                 ; 2709
                                51     00AA  C4  32 00053 3$:      CVTWL   170(PBCB), R1                          ; 2711
51          31    A2                 10           00  ED 00058          CMPZV   #0, #16, 49(PP_BASE), R1
                                         06  15 0005E          BLEQ    4$
                         00AA    C4     31    A2  B0 00060          MOVW    49(PP_BASE), 170(PBCB)                 ; 2713
                                51     00AC  C4  32 00066 4$:      CVTWL   172(PBCB), R1                          ; 2715
51          33    A2                 10           00  ED 0006B          CMPZV   #0, #16, 51(PP_BASE), R1
                                         06  18 00071          BGEQ    5$
                         00AC    C4     33    A2  B0 00073          MOVW    51(PP_BASE), 172(PBCB)                 ; 2717
                                51     00AE  C4  32 00079 5$:      CVTWL   174(PBCB), R1                          ; 2719
51          35    A2                 10           00  ED 0007E          CMPZV   #0, #16, 53(PP_BASE), R1
                                         06  15 00084          BLEQ    6$
```

```
                    00AE   C4      35   A2  B0 00086          MOVW    53(PP_BASE), 174(PBCB)          ; 2721
                           53      0C   A2  D0 0008C 6$:      MOVL    12(PP_BASE), CURR_PP            ; 2730
                                        80  11 00090          BRB     1$                              ; 2671
                                        54  DD 00092 7$:      PUSHL   PBCB                            ; 2734
                00000000G  00           01  FB 00094          CALLS   #1, SMG$$MIN_UPD                ;
                                            04 0009B 8$:      RET                                     ; 2737

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

; 2678          2738  1 !<BLF/PAGE>
```

```
;  2680          2739  1 %SBTTL 'SMG$$MOVE_TEXT_TO_WINDOW_BUF - Move text from display buf. to window buf.'
;  2681          2740  1 GLOBAL ROUTINE SMG$$MOVE_TEXT_TO_WINDOW_BUF (
;  2682          2741  1                                       PP : REF $PP_DECL
;  2683          2742  1                                           ) =
;  2684          2743  1 !++
;  2685          2744  1 ! FUNCTIONAL DESCRIPTION:
;  2686          2745  1 !
;  2687          2746  1 !       This routine moves text from the buffer located at
;  2688          2747  1 !       .DCB [DCB_A_TEXT_BUF] into the window text buffer.
;  2689          2748  1 !       Array of bytes at .DCB [ DCB_A_ATTR_BUF ] describe the
;  2690          2749  1 !       rendition this text must assume and is moved into the
;  2691          2750  1 !       associated window attribute buffer.
;  2692          2751  1 !       Similarly, if the alternate character set buffer at
;  2693          2752  1 !       .DCB [DCB_A_CHAR_SET_BUF] exists, it must be mapped into the
;  2694          2753  1 !       window alternate character set buffer.
;  2695          2754  1 !
;  2696          2755  1 ! CALLING SEQUENCE:
;  2697          2756  1 !
;  2698          2757  1 !       ret_status.wlc.v = SMG$$MOVE_TEXT_TO_WINDOW_BUF ( PP.rab.r)
;  2699          2758  1 !
;  2700          2759  1 ! FORMAL PARAMETERS:
;  2701          2760  1 !
;  2702          2761  1 !       PP.rab.r                Address of pasting packet.
;  2703          2762  1 !
;  2704          2763  1 ! IMPLICIT INPUTS:
;  2705          2764  1 !
;  2706          2765  1 !       NONE
;  2707          2766  1 !
;  2708          2767  1 ! IMPLICIT OUTPUTS:
;  2709          2768  1 !
;  2710          2769  1 !       NONE
;  2711          2770  1 !
;  2712          2771  1 ! COMPLETION STATUS:
;  2713          2772  1 !
;  2714          2773  1 !       SS$_NORMAL      Normal successful completion
;  2715          2774  1 !
;  2716          2775  1 ! SIDE EFFECTS:
;  2717          2776  1 !
;  2718          2777  1 !       NONE
;  2719          2778  1 !--
;  2720          2779  1
;  2721          2780  2     BEGIN
;  2722          2781  2
;  2723          2782  2     BUILTIN
;  2724          2783  2         SKPC;
;  2725          2784  2
;  2726          2785  2     LOCAL
;  2727          2786  2         DCB  : REF BLOCK [,BYTE],       ! Addr of virtual display
;  2728          2787  2                                         ! control block.
;  2729          2788  2         PBCB : REF BLOCK [,BYTE],       ! Addr of pasteboard control
;  2730          2789  2                                         ! block
;  2731          2790  2         WCB  : REF BLOCK [,BYTE],       ! Addr of window control block
;  2732          2791  2         IN_LINE_CHAR  : REF VECTOR [,BYTE],     ! Addr. of line char.
;  2733          2792  2                                                 ! vector off DCB
;  2734          2793  2         OUT_LINE_CHAR : REF VECTOR [,BYTE],     ! Addr. of line char.
;  2735          2794  2                                                 ! vector off WCB
;  2736          2795  2
```

```
;  2737          2796  2          WCB_COLS,       ! Extracted .WCB [WCB_W_NO_COLS]
;  2738          2797  2          DCB_COLS,       ! Extracted .DCB [DCB_W_NO_COLS]
;  2739          2798  2          FROM_INDEX,     ! Pointer into DCB buffers
;  2740          2799  2          TO_INDEX,       ! Pointer into WCB buffers for normal mapping
;  2741          2800  2          SPEC_TO_INDEX,  ! Pointer into WCB buffers for special mapping
;  2742          2801  2          RE_BORDER : INITIAL (0);        ! Flag to indicate that the
;  2743          2802  2                                          ! border elements need to be
;  2744          2803  2                                          ! remapped because some line
;  2745          2804  2                                          ! changed from normal to DWDH or
;  2746          2805  2                                          ! vice versa.
;  2747          2806  2
;  2748          2807  2      MACRO
;  2749        M 2808  2          $MAP_LINE_NORMAL =
;  2750        M 2809  2              CH$MOVE ( .PP [PP_W_MOVE_LENGTH],                   ! Length
;  2751        M 2810  2                        .DCB [DCB_A_TEXT_BUF] + .FROM_INDEX,      ! Source
;  2752        M 2811  2                        .WCB [WCB_A_TEXT_BUF] + .TO_INDEX);       ! Destination
;  2753          2812  2          %,
;  2754          2813  2
;  2755        M 2814  2          $MAP_LINE_SPECIAL =
;  2756        M 2815  2              CH$MOVE ( (.PP [PP_W_MOVE_LENGTH]-1)/2,             ! Length
;  2757        M 2816  2                        .DCB [DCB_A_TEXT_BUF] + .FROM_INDEX,      ! Source
;  2758        M 2817  2                        .WCB [WCB_A_TEXT_BUF] + .SPEC_TO_INDEX); ! Destination
;  2759          2818  2          %,
;  2760          2819  2
;  2761        M 2820  2          $MAP_ATTR_NORMAL =
;  2762        M 2821  2              CH$MOVE ( .PP [PP_W_MOVE_LENGTH],                   ! Length
;  2763        M 2822  2                        .DCB [DCB_A_ATTR_BUF] + .FROM_INDEX,      ! Source
;  2764        M 2823  2                        .WCB [WCB_A_ATTR_BUF] + .TO_INDEX);       ! Destination
;  2765          2824  2          %,
;  2766          2825  2
;  2767        M 2826  2          $MAP_ATTR_SPECIAL =
;  2768        M 2827  2              CH$MOVE ( (.PP [PP_W_MOVE_LENGTH]-1)/2,             ! Length
;  2769        M 2828  2                        .DCB [DCB_A_ATTR_BUF] + .FROM_INDEX,      ! Source
;  2770        M 2829  2                        .WCB [WCB_A_ATTR_BUF] + .SPEC_TO_INDEX); ! Destination
;  2771          2830  2          %,
;  2772          2831  2
;  2773        M 2832  2          $MAP_ALT_CHAR_NORMAL =
;  2774        M 2833  2              IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
;  2775        M 2834  2              THEN
;  2776        M 2835  2              CH$MOVE ( .PP [PP_W_MOVE_LENGTH],                   ! Length
;  2777        M 2836  2                        .DCB [DCB_A_CHAR_SET_BUF] + .FROM_INDEX, ! Source
;  2778        M 2837  2                        .WCB [WCB_A_CHAR_SET_BUF] + .TO_INDEX);  ! Dest.
;  2779          2838  2          %,
;  2780          2839  2
;  2781        M 2840  2          $MAP_ALT_CHAR_SPECIAL =
;  2782        M 2841  2              IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
;  2783        M 2842  2              THEN
;  2784        M 2843  2              CH$MOVE ( (.PP [PP_W_MOVE_LENGTH]-1)/2,             ! Length
;  2785        M 2844  2                        .DCB [DCB_A_CHAR_SET_BUF] + .FROM_INDEX,  ! Source
;  2786        M 2845  2                        .WCB [WCB_A_CHAR_SET_BUF] + .SPEC_TO_INDEX);  ! Dest.
;  2787          2846  2          %,
;  2788          2847  2
;  2789        M 2848  2          $CLEAR_TEXT_LINE =
;  2790        M 2849  2              CH$FILL ( %C' ',                                    ! Fill char
;  2791        M 2850  2                        .WCB [WCB_W_NO_COLS],                     ! # of bytes
;  2792        M 2851  2                        .WCB [WCB_A_TEXT_BUF] + .WLP);            ! Destination
;  2793          2852  2          %,
```

```
;  2794          2853  2
;  2795        M 2854  2       $CLEAR_ATTR_LINE =
;  2796        M 2855  2           CH$FILL ( 0,                                    ! Fill char
;  2797        M 2856  2                     .WCB [WCB_W_NO_COLS],                 ! # of bytes
;  2798        M 2857  2                     .WCB [WCB_A_ATTR_BUF] + .WLP);        ! Destination
;  2799          2858  2       %,
;  2800          2859  2
;  2801        M 2860  2       $MAP_ALL_NORMAL =
;  2802        M 2861  2           BEGIN
;  2803        M 2862  2           $MAP_LINE_NORMAL;
;  2804        M 2863  2           $MAP_ATTR_NORMAL;
;  2805        M 2864  2           $MAP_ALT_CHAR_NORMAL;
;  2806        M 2865  2           END;
;  2807          2866  2       %,
;  2808          2867  2
;  2809        M 2868  2       $CLEAR_ALT_CHAR_LINE =
;  2810        M 2869  2           IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
;  2811        M 2870  2           THEN
;  2812        M 2871  2           CH$FILL ( 0,
;  2813        M 2872  2                     .WCB [WCB_W_NO_COLS],
;  2814        M 2873  2                     .WCB [WCB_A_CHAR_SET_BUF] + .WLP);
;  2815          2874  2       %,
;  2816          2875  2
;  2817        M 2876  2       $MAP_ALL_SPECIAL =
;  2818        M 2877  2           BEGIN
;  2819        M 2878  2           $MAP_LINE_SPECIAL;
;  2820        M 2879  2           $MAP_ATTR_SPECIAL;
;  2821        M 2880  2           $MAP_ALT_CHAR_SPECIAL;
;  2822        M 2881  2           END;
;  2823          2882  2       %,
;  2824          2883  2
;  2825        M 2884  2       $CLEAR_ALL =
;  2826        M 2885  2           BEGIN
;  2827        M 2886  2           $CLEAR_TEXT_LINE;
;  2828        M 2887  2           $CLEAR_ATTR_LINE;
;  2829        M 2888  2           $CLEAR_ALT_CHAR_LINE;
;  2830        M 2889  2           RE_BORDER = 1; ! Set flag to rebuild border
;  2831        M 2890  2           END;
;  2832          2891  2       %;
;  2833          2892  2
;  2834          2893  2       DCB  = .PP [PP_A_DCB_ADDR];
;  2835          2894  2       PBCB = .PP [PP_A_PBCB_ADDR];
;  2836          2895  2       WCB  = .PBCB [PBCB_A_WCB];
;  2837          2896  2       IN_LINE_CHAR = .DCB [DCB_A_LINE_CHAR];
;  2838          2897  2       OUT_LINE_CHAR= .WCB [WCB_A_LINE_CHAR];
;  2839          2898  2       WCB_COLS = .WCB [WCB_W_NO_COLS];
;  2840          2899  2       DCB_COLS = .DCB [DCB_W_NO_COLS];
;  2841          2900  2
;  2842          2901  2   !+
;  2843          2902  2   ! Before diverging on two copying paths, check to see if we are going
;  2844          2903  2   ! to get involved with alternate character set buffers.  If one exists
;  2845          2904  2   ! in the DCB but does not yet exist in the WCB, we have to allocate
;  2846          2905  2   ! one for the WCB and initialize it.
;  2847          2906  2   !-
;  2848          2907  2       IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0  AND
;  2849          2908  2          .WCB [WCB_A_CHAR_SET_BUF] EQL 0
;  2850          2909  2       THEN
```

```
;  2851     2910  3          BEGIN   ! Alloc. and init. window alternate char. set buffer
;  2852     2911  3          LOCAL
;  2853     2912  3              STATUS;     ! Status of LIB$GET_VM call
;  2854     2913  3
;  2855     2914  4          IF NOT (STATUS = LIB$GET_VM ( WCB [WCB_L_BUFSIZE],
;  2856     2915  4                                        WCB [WCB_A_CHAR_SET_BUF]))
;  2857     2916  3          THEN
;  2858     2917  3              RETURN (.STATUS);
;  2859     2918  3
;  2860     2919  3          CH$FILL ( 0, .WCB [WCB_L_BUFSIZE], .WCB [WCB_A_CHAR_SET_BUF]);
;  2861     2920  2          END;    ! Alloc. and init. window alternate char. set buffer
;  2862     2921  2
;  2863     2922  2      !+
;  2864     2923  2      ! At this point we need to decide whether we are dealing with either
;  2865     2924  2      ! a virtual display or a window control block that includes either
;  2866     2925  2      ! Double-Wide (DW) and/or Double-Wide/Double-High (DWDH) lines.
;  2867     2926  2      ! We can tell whether either is non-standard by looking at the zeroth
;  2868     2927  2      ! byte of its line characteristics vector.
;  2869     2928  2      ! If we are dealing with either non-standard, we need to deal with them
;  2870     2929  2      ! on a line by line basis.
;  2871     2930  2      ! For each line to be mapped we implement the logic summarized in the
;  2872     2931  2      ! following table:
;  2873     2932  2      !
;  2874     2933  2      !    DCB Line           WCB Line         Mapping
;  2875     2934  2      !    Char. Vect         Char. Vector     Action
;  2876     2935  2      !    ----------         ------------     ------
;  2877     2936  2      !
;  2878     2937  2      !    Not DW or DWDH     Not DW or DWDH   Map normally
;  2879     2938  2      !    (i.e., normal)     (i.e., Normal)
;  2880     2939  2      !
;  2881     2940  2      !    Not DW or DWDH     DW or DWDH       If DW supported
;  2882     2941  2      !    (i.e., normal)                      then
;  2883     2942  2      !                                            Blank entire WCB text line
;  2884     2943  2      !                                            map normally
;  2885     2944  2      !                                        else
;  2886     2945  2      !                                            map normally
;  2887     2946  2      !
;  2888     2947  2      !
;  2889     2948  2      !    DW or DWDH         Not DW or DWDH   If DW supported
;  2890     2949  2      !                       (i.e., normal)   then
;  2891     2950  2      !                                            Blank entire WCB text line
;  2892     2951  2      !                                            use special mapping
;  2893     2952  2      !                                        else
;  2894     2953  2      !                                            map normally
;  2895     2954  2      !
;  2896     2955  2      !
;  2897     2956  2      !    DW or DWDH         DW or DWDH       If DW supported
;  2898     2957  2      !                                        then
;  2899     2958  2      !                                            Use special mapping
;  2900     2959  2      !                                        else
;  2901     2960  2      !                                            Map normally
;  2902     2961  2      !
;  2903     2962  2      !
;  2904     2963  2      !
;  2905     2964  2      !-
;  2906     2965  2          FROM_INDEX = .PP [PP_W_FROM_INDEX];
;  2907     2966  2          TO_INDEX   = .PP [PP_W_TO_INDEX];
```

```
;  2908          2967  2     IF .IN_LINE_CHAR  [0] EQL 0 AND     ! DCB normal
;  2909          2968  2        .OUT_LINE_CHAR [0] EQL 0         ! WCB normal
;  2910          2969  2     THEN
;  2911          2970  3         BEGIN   ! Both are normal -- no special action needed
;  2912          2971  3         !+
;  2913          2972  3         ! Check to see if we can do it with a single CH$MOVE or whether
;  2914          2973  3         ! we must do it a row at a time.
;  2915          2974  3         !-
;  2916          2975  3         IF .PP [PP_V_CONTIG]
;  2917          2976  3         THEN
;  2918          2977  4             BEGIN       ! Can be done in single move
;  2919          2978  4
;  2920          2979  4             ! Move text
;  2921          2980  4             CH$MOVE ( .PP [PP_L_MOVE_SIZE],              ! length
;  2922          2981  4                   .DCB [DCB_A_TEXT_BUF] + .FROM_INDEX,   ! source
;  2923          2982  4                   .WCB [WCB_A_TEXT_BUF] + .TO_INDEX);    ! dest.
;  2924          2983  4
;  2925          2984  4             ! Move attributes
;  2926          2985  4             CH$MOVE ( .PP [PP_L_MOVE_SIZE],              ! length
;  2927          2986  4                   .DCB [DCB_A_ATTR_BUF] + .FROM_INDEX,   ! source
;  2928          2987  4                   .WCB [WCB_A_ATTR_BUF] + .TO_INDEX);    ! dest.
;  2929          2988  4
;  2930          2989  4             ! Move alternate character set buffer pieces, if necessary
;  2931          2990  4             IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
;  2932          2991  4             THEN
;  2933          2992  5                 BEGIN   ! Map alternate character set
;  2934          2993  5                 CH$MOVE ( .PP [PP_L_MOVE_SIZE],              ! Length
;  2935          2994  5                       .DCB [DCB_A_CHAR_SET_BUF] + .FROM_INDEX, ! Source
;  2936          2995  5                       .WCB [WCB_A_CHAR_SET_BUF] + .TO_INDEX);  ! Dest.
;  2937          2996  4                 END;    ! Map alternate character set
;  2938          2997  4             END         ! Can be done in single move
;  2939          2998  4
;  2940          2999  3         ELSE
;  2941          3000  3
;  2942          3001  4             BEGIN       ! Must be done row at a time
;  2943          3002  4
;  2944          3003  4             INCR R FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
;  2945          3004  4             DO
;  2946          3005  5                 BEGIN   ! For all rows in this display
;  2947          3006  5                 $MAP_LINE_NORMAL;
;  2948          3007  5                 $MAP_ATTR_NORMAL;
;  2949          3008  5
;  2950          3009  5                 FROM_INDEX = .FROM_INDEX + .DCB_COLS;
;  2951          3010  5                 TO_INDEX   = .TO_INDEX   + .WCB_COLS;
;  2952          3011  4                 END;    ! For all rows in this display
;  2953          3012  4
;  2954          3013  4             !+
;  2955          3014  4             ! Move alternate character set buffer pieces, if necessary
;  2956          3015  4             !
;  2957          3016  4             IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
;  2958          3017  4             THEN
;  2959          3018  5                 BEGIN   ! Map alternate character set buffer
;  2960          3019  5                 FROM_INDEX = .PP [PP_W_FROM_INDEX];
;  2961          3020  5                 TO_INDEX   = .PP [PP_W_TO_INDEX];
;  2962          3021  5
;  2963          3022  5                 INCR R FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
;  2964          3023  5                 DO
```

```
; 2965          3024  6                       BEGIN
; 2966          3025  6                       $MAP_ALT_CHAR_NORMAL;
; 2967          3026  6
; 2968          3027  6                       FROM_INDEX = .FROM_INDEX + .DCB_COLS;
; 2969          3028  6                       TO_INDEX   = .TO_INDEX   + .WCB_COLS;
; 2970          3029  6                       END;
; 2971          3030  5
; 2972          3031  4                   END;    ! Map alternate character set buffer
; 2973          3032  4
; 2974          3033  3               END;        ! Must be done row at a time
; 2975          3034  2           END     ! Both are normal -- no special action needed
; 2976          3035  2
; 2977          3036  2       ELSE
; 2978          3037  2           BEGIN   ! One or the other contains special characteristics
; 2979          3038  2           LOCAL
; 2980          3039  2               DLN,        ! Virtual display line number
; 2981          3040  2               WLN,        ! WCB line number
; 2982          3041  2               WLP;        ! Byte offset corresponding to WLN
; 2983          3042  2
; 2984          3043  2               !+
; 2985          3044  2               ! Must deal with this mapping on a row by row basis and on
; 2986          3045  2               ! each row inspect the line characteristics vector entry of
; 2987          3046  2               ! both the DCB line characteristics vector and the WCB line
; 2988          3047  2               ! characteristics vector in order to determine how to map this
; 2989          3048  2               ! particular line.
; 2990          3049  2               !
; 2991          3050  2               ! Initialize the special index we will be using.
; 2992          3051  2               !-
; 2993          3052  2               SPEC_TO_INDEX = (.PP [PP_W_FIRST_WCB_ROW] -1) *
; 2994          3053  2                               .WCB [WCB_W_NO_COLS] +
; 2995          3054  2                               ((.PP [PP_W_FIRST_WCB_COL] +2) / 2) -1;
; 2996          3055  2
; 2997          3056  2               DLN = (.FROM_INDEX / .DCB_COLS) +1;
; 2998          3057  2               WLN = .PP [PP_W_FIRST_WCB_ROW];
; 2999          3058  2               WLP = (.WLN - 1) * .WCB_COLS;
; 3000          3059  2
; 3001          3060  2               INCR R FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
; 3002          3061  2               DO
; 3003          3062  3                   BEGIN       ! For each row
; 3004          3063  3                   IF .IN_LINE_CHAR  [.DLN] EQL 0 AND  ! DCB normal
; 3005          3064  3                      .OUT_LINE_CHAR [.WLN] EQL 0      ! WCB normal
; 3006          3065  3                   THEN
; 3007          3066  4                       BEGIN   ! Normal line
; 3008          3067  4                       $MAP_ALL_NORMAL;
; 3009          3068  4                       END     ! Normal line
; 3010          3069  3                   ELSE
; 3011          3070  4                       BEGIN   ! Special line
; 3012          3071  4                       IF .IN_LINE_CHAR  [.DLN] EQL 0 AND      ! DCB normal
; 3013          3072  4                          .OUT_LINE_CHAR [.WLN] NEQ 0          ! WCB special
; 3014          3073  4                       THEN
; 3015          3074  5                           BEGIN       ! WCB has special, DCB normal
; 3016          3075  5                           IF .PBCB [PBCB_V_WIDE]      ! If DW supported
; 3017          3076  5                           THEN
; 3018          3077  6                               BEGIN ! Special mapping
; 3019          3078  6                               $CLEAR_ALL;
; 3020          3079  6                               $MAP_ALL_NORMAL ;
; 3021          3080  6                               END   ! Special mapping
```

```
;  3022     3081  6                    ELSE
;  3023     3082  7                        BEGIN ! Normal mapping
;  3024     3083  7                        $MAP_ALL_NORMAL;
;  3025     3084  6                        END;  ! Normal mapping
;  3026     3085  6                    END           ! WCB has special, DCB normal
;  3027     3086  5                ELSE
;  3028     3087  3                    BEGIN
;  3029     3088  6                    IF .IN_LINE_CHAR  [.DLN] NEQ 0 AND  ! DCB special
;  3030     3089  6                       .OUT_LINE_CHAR [.WLN] EQL 0      ! WCB normal
;  3031     3090  6                    THEN
;  3032     3091  7                        BEGIN   ! DCB special, WCB normal
;  3033     3092  7                        IF .PBCB [PBCB_V_WIDE]  ! If DW supported
;  3034     3093  7                        THEN
;  3035     3094  8                            BEGIN       ! Map special
;  3036     3095  8                            $CLEAR_ALL;
;  3037     3096  8                            $MAP_ALL_SPECIAL;
;  3038     3097  8                            END         ! Map special
;  3039     3098  7                        ELSE
;  3040     3099  8                            BEGIN       ! Map normal
;  3041     3100  8                            $MAP_ALL_NORMAL;
;  3042     3101  7                            END;        ! Map normal
;  3043     3102  7                        END     ! DCB special, WCB normal
;  3044     3103  6                    ELSE
;  3045     3104  7                        BEGIN   ! Both have special
;  3046     3105  7                        IF .PBCB [PBCB_V_WIDE]  ! If DW supported
;  3047     3106  7                        THEN
;  3048     3107  8                            BEGIN       ! Map special
;  3049     3108  8                            $MAP_ALL_SPECIAL;
;  3050     3109  8                            END         ! Map special
;  3051     3110  7                        ELSE
;  3052     3111  8                            BEGIN       ! Map normal
;  3053     3112  8                            $MAP_ALL_NORMAL;
;  3054     3113  7                            END;        ! Map normal
;  3055     3114  6                        END;    ! Both have special
;  3056     3115  5                    END;
;  3057     3116  4                END;    ! Special line
;  3058     3117  4            !+
;  3059     3118  4            ! Advance indices for next row
;  3060     3119  4            !-
;  3061     3120  4            FROM_INDEX     = .FROM_INDEX     + .DCB_COLS;
;  3062     3121  4            TO_INDEX       = .TO_INDEX       + .WCB_COLS;
;  3063     3122  4            SPEC_TO_INDEX = .SPEC_TO_INDEX + .WCB_COLS;
;  3064     3123  4            DLN = .DLN + 1;
;  3065     3124  4            WLN = .WLN + 1;
;  3066     3125  4            WLP = .WLP + .WCB_COLS;
;  3067     3126  3            END;        ! For each row
;  3068     3127  3
;  3069     3128  2        END;    ! One or the other contains special characteristics
;  3070     3129  2
;  3071     3130  2    !+
;  3072     3131  2    ! If this device supports DW or DWDH then we must update WCB's line
;  3073     3132  2    ! characteristics vector based on the rows we just modified.
;  3074     3133  2    ! All further actions are necessary only if DW or DWDH is supported by
;  3075     3134  2    ! this device.
;  3076     3135  2    !-
;  3077     3136  2        IF .PBCB [PBCB_V_WIDE]
;  3078     3137  2        THEN
```

```
;  3079     3138  3          BEGIN   ! Special supported
;  3080     3139  3          IF .PP [PP_W_ROWS_TO_MOVE] NEQ 0        ! Something got mapped
;  3081     3140  3          THEN
;  3082     3141  4              BEGIN
;  3083     3142  4              IF .PP [PP_W_ROW] GTR 0
;  3084     3143  4              THEN
;  3085     3144  4                  ! Start movement from 1st display row
;  3086     3145  4                  CH$MOVE ( .PP [PP_W_ROWS_TO_MOVE],
;  3087     3146  4                            IN_LINE_CHAR [1],
;  3088     3147  4                            OUT_LINE_CHAR [ .PP [PP_W_ROW]])
;  3089     3148  4              ELSE
;  3090     3149  4                  ! Start movement from nth display row
;  3091     3150  4                  CH$MOVE ( .PP [PP_W_ROWS_TO_MOVE],
;  3092     3151  4                            IN_LINE_CHAR [.DCB [DCB_W_NO_ROWS] -
;  3093     3152  4                                          .PP [PP_W_ROWS_TO_MOVE] +1],
;  3094     3153  4                            OUT_LINE_CHAR [1]);
;  3095     3154  4
;  3096     3155  4              END;
;  3097     3156
;  3098     3157          !+
;  3099     3158          ! If we are dealing with a bordered display and we have
;  3100     3159          ! made the transition from a normal line to a DHDW line, or
;  3101     3160          ! vice versa, it will be necessary to remap the border elements.
;  3102     3161          !-
;  3103     3162          IF .DCB [DCB_V_BORDERED]  AND
;  3104     3163             .RE_BORDER EQL 1
;  3105     3164          THEN
;  3106     3165  4           BEGIN
;  3107     3166  4           LOCAL
;  3108     3167  4               STATUS ; ! Status of subroutine call
;  3109     3168  5           IF NOT (STATUS = SMG$$DRAW_BORDER ( .DCB, .PP, .WCB))
;  3110     3169  4           THEN
;  3111     3170  4               RETURN (.STATUS);
;  3112     3171  4           END;
;  3113     3172
;  3114     3173          !+
;  3115     3174          ! Check through current WCB line characteristics vector.  If
;  3116     3175          ! any of the elements is non-zero, set OUT_LINE_CHAR [0] to the
;  3117     3176          ! 1st encountered non-zero value.
;  3118     3177          !-
;  3119     3178          IF .PP [PP_W_ROWS_TO_MOVE] GTR 0
;  3120     3179          THEN
;  3121     3180  4           BEGIN       ! Some mapping took place
;  3122     3181  4           LOCAL
;  3123     3182  4               BYTES_REMAINING,                    ! Output from SKPC
;  3124     3183  4               FOUND_BYTE : REF VECTOR [,BYTE]; ! Output from SKPC
;  3125     3184  4
;  3126     3185  4           OUT_LINE_CHAR [0] = 0;      ! Assume none will be found
;  3127     3186  4           SKPC (%REF(0), WCB [WCB_W_NO_ROWS], OUT_LINE_CHAR [0];
;  3128     3187  4                 BYTES_REMAINING, FOUND_BYTE);
;  3129     3188  4
;  3130     3189  4           IF .BYTES_REMAINING NEQ 0   ! If non-zero found
;  3131     3190  4           THEN
;  3132     3191  4               OUT_LINE_CHAR [0] = .FOUND_BYTE [0];
;  3133     3192  3           END;        ! Some mapping took place
;  3134     3193  2      END;    ! Special supported
;  3135     3194  2
```

```
;  3136          3195  2
;  3137          3196  2      RETURN (SS$_NORMAL);
;  3138          3197  1      END;                    ! End of routine SMG$$MOVE_TEXT_TO_WINDOW_BUF


                                                 0FFC 00000          .ENTRY  SMG$$MOVE_TEXT_TO_WINDOW_BUF, Save R2,R3,-      ; 2740
                                                                             R4,R5,R6,R7,R8,R9,R10,R11                           ;
                                      5E              30  C2 00002   SUBL2   #48, SP                                         ;
                                                      7E  D4 00005   CLRL    RE_BORDER                                       ; 2780
                          50      04  AC              10  C1 00007   ADDL3   #16, PP, R0                                     ; 2893
                                      57              60  D0 0000C   MOVL    (R0), DCB                                       ;
                          50      04  AC              14  C1 0000F   ADDL3   #20, PP, R0                                     ; 2894
                                                      60  DD 00014   PUSHL   (R0)                                            ;
                          50          6E              08  C1 00016   ADDL3   #8, PBCB, R0                                    ; 2895
                                      5B              60  D0 0001A   MOVL    (R0), WCB                                       ;
                                              4C      A7  DD 0001D   PUSHL   76(DCB)                                         ; 2896
                                              2C      AB  DD 00020   PUSHL   44(WCB)                                         ; 2897
                                      7E      06      AB  3C 00023   MOVZWL  6(WCB), WCB_COLS                               ; 2898
                                      7E      06      A7  3C 00027   MOVZWL  6(DCB), DCB_COLS                               ; 2899
                                              18      A7  9F 0002B   PUSHAB  24(DCB)                                         ; 2907
                                              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)                                         ; 2908
                                                      19  12 0003C   BNEQ    2$                                              ;
                                              10      AB  9F 0003E   PUSHAB  16(WCB)                                         ; 2915
                                              28      AB  9F 00041   PUSHAB  40(WCB)                                         ; 2914
                                  00000000G       00      02  FB 00044   CALLS   #2, LIB$GET_VM                             ; 2915
                                                  01      50  E8 0004B   BLBS    STATUS, 1$                                      ;
                                                          04  0004E   RET                                                    ;
28  AB                      00                  6E          00  2C 0004F 1$:    MOVC5   #0, (SP), #0, 40(WCB), @16(WCB)         ; 2919
                                                          10  BB 00055                                                        ;
                          50          04      AC          1E  C1 00057 2$:    ADDL3   #30, PP, R0                             ; 2965
                                      56              60  3C 0005C   MOVZWL  (R0), FROM_INDEX                                ;
                          50          04      AC          20  C1 0005F   ADDL3   #32, PP, R0                                 ; 2966
                                      5A              60  3C 00064   MOVZWL  (R0), TO_INDEX                                  ;
                                      30      AE      10  A7  9E 00067   MOVAB   16(DCB), 48(SP)                             ; 2981
                                      28      AE      08  AB  9E 0006C   MOVAB   8(WCB), 40(SP)                              ; 2982
                                      2C      AE      14  A7  9E 00071   MOVAB   20(DCB), 44(SP)                            ; 2986
                                      24      AE      0C  AB  9E 00076   MOVAB   12(WCB), 36(SP)                            ; 2987
                                                      10  BF  95 0007B   TSTB    @IN_LINE_CHAR                               ; 2967
                                                          03  12 0007E   BNEQ    3$                                          ;
                                                      0C  BF  95 00080   TSTB    @OUT_LINE_CHAR                              ; 2968
                                                          03  13 00083 3$:     BEQL    4$                                      ;
                                                  00B9        31  00085   BRW     13$                                         ;
                                      7E      04      AC          2A  C1 00088 4$:    ADDL3   #42, PP, -(SP)                  ; 2975
                                      3C              9E          01  E1 0008D   BBC     #1, @(SP)+, 5$                          ;
                                                      58  28  BE  D0 00091   MOVL    @40(SP), R8                             ; 2982
                                                      59  30  BE  D0 00095   MOVL    @48(SP), R9                             ;
                                      7E      04      AC          2B  C1 00099   ADDL3   #43, PP, -(SP)                      ;
                                  684A            6946            9E  28 0009E   MOVC3   @(SP)+, (R9)[FROM_INDEX], (R8)[TO_INDEX]  ;
                                                      58  24  BE  D0 000A4   MOVL    @36(SP), R8                             ; 2987
                                                      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$                                  ;2990
                  58          00  BE  D0  000BB                   MOVL    @0(SP), R8                                  ;2995
                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$                                         ;2975
                50  04  AC                  1C  C1  000CD  5$:      ADDL3   #28, PP, R0                                 ;3003
                  59                         60  3C  000D2          MOVZWL  (R0), R9
                                             58  D4  000D5          CLRL    R                                           ;3010
                                             2A  11  000D7          BRB     7$
                                         28  BE  DD  000D9  6$:      PUSHL   @40(SP)                                     ;3005
                                         34  BE  DD  000DC          PUSHL   @52(SP)
                7E  04  AC                  22  C1  000DF          ADDL3   #34, PP, -(SP)
              9E4A      9E46                9E  28  000E4          MOVC3   @(SP)+, @(SF)+[FROM_INDEX], @(SP)+-
                                                                            [TO_INDEX]
                                         24  BE  DD  000EA          PUSHL   @36(SP)                                     ;3006
                                         30  BE  DD  000ED          PUSHL   @48(SP)
                7E  04  AC                  22  C1  000F0          ADDL3   #34, PP, -(SP)
              9E4A      9E46                9E  28  000F5          MOVC3   @(SP)+, @(SP)+[FROM_INDEX], @(SP)+-
                                                                            [TO_INDEX]
                  56          04  AE  C0  000FB                   ADDL2   DCB_COLS, FROM_INDEX                        ;3009
                  5A          08  AE  C0  000FF                   ADDL2   WCB_COLS, TO_INDEX                          ;3010
                D2  58              59  F3  00103  7$:      AOBLEQ  R9, R, 6$                                           ;3003
                  33          48  AE  E9  00107  8$:      BLBC    72(SP), 12$                                         ;3016
                50  04  AC                  1E  C1  0010B          ADDL3   #30, PP, R0                                 ;3019
                  56                         60  3C  00110          MOVZWL  (R0), FROM_INDEX
                50  04  AC                  20  C1  00113          ADDL3   #32, PP, R0                                 ;3020
                  5A                         60  3C  00118          MOVZWL  (R0), TO_INDEX
                                             58  D4  0011B          CLRL    R                                           ;3022
                                             1B  11  0011D          BRB     11$
                  0F        48  AE  E9  0011F  9$:      BLBC    72(SP), 10$                                         ;3024
                            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$:      ADDL2   DCB_COLS, FROM_INDEX                        ;3027
                  5A          08  AE  C0  00136          ADDL2   WCB_COLS, TO_INDEX                          ;3028
                E1  58              59  F3  0013A  11$:      AOBLEQ  R9, R, 9$                                           ;3022
                        020E        31  0013E  12$:      BRW     27$                                                 ;2967
                51  04  AC                  2F  C1  00141  13$:      ADDL3   #47, PP, R1                                 ;3052
                  50                         61  3C  00146          MOVZWL  (R1), R0
                                             50  D7  00149          DECL    R0
                  3C  AE      06  AB  3C  0014B                   MOVZWL  6(WCB), 60(SP)                              ;3053
                  50          3C  AE  C4  00150                   MULL2   60(SP), R0
                52  04  AC                  33  C1  00154          ADDL3   #51, PP, R2                                 ;3054
                  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                    ;3056
                                             59  D6  0016D          INCL    DLN
                50  04  AC                  2F  C1  0016F          ADDL3   #47, PP, R0                                 ;3057
                  58                         60  3C  00174          MOVZWL  (R0), WLN
                  50          FF  A8  9E  00177                   MOVAB   -1(R8), R0                                  ;3058
            1C  AE  50          08  AE  C5  0017B                   MULL3   WCB_COLS, R0, WLP
                50  04  AC                  1C  C1  00181          ADDL3   #28, PP, R0                                 ;3060
```

```
                                48   AE                60   3C 00186          MOVZWL  (R0), 72(SP)
                             50 04   AC                22   C1 0018A          ADDL3   #34, PP, R0                                    ; 3066
                                34   AE                60   9E 0018F          MOVAB   (R0), 52(SP)
                                                   44   AE   D4 00193          CLRL    R
                                                     01AE   31 00196          BRW     26$
                                                       50   D4 00199 14$:      CLRL    R0                                            ; 3063
                                     51            10   AE   D0 0019B          MOVL    IN_LINE_CHAR, R1
                                                   6941   95 0019F          TSTB    (D[N)[RT]
                                                     0E   12 001A2          BNEQ    16$
                                                       50   D6 001A4          INCL    R0
                                     51            0C   AE   D0 001A6          MOVL    OUT_LINE_CHAR, R1                             ; 3064
                                                   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$:      ADDL3   #250, PBCB, R1                               ; 3075
                                40   AE                61   9E 001BB          MOVAB   (R1), 64(SP)
                                     6D                50   E9 001BF          BLBC    R0, 19$
                                     50            0C   AE   D0 001C2          MOVL    OUT_LINE_CHAR, R0                             ; 3072
                                                   6840   95 001C6          TSTB    (WLN)[R0]
                                                     64   13 001C9          BEQL    19$
                                     E0            40   BE   E9 001CB          BLBC    @64(SP), 15$                                  ; 3075
                          38   AE      28   BE       1C   AE   C1 001CF          ADDL3   WLP, @40(SP), 56(SP)                        ; 3077
                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$:      MOVL    #1, RE_BORDER
                                                   28   BE   DD 0020B          PUSHL   @40(SP)                                       ; 3078
                                                   34   BE   DD 0020E          PUSHL   @52(SP)
                             9E4A            9E46    3C   BE   28 00211          MOVC3   @60(SP), @(SP)+[FROM_INDEX], @(SP)+-
                                                                                [TO_INDEX]
                                                   24   BE   DD 00218          PUSHL   @36(SP)
                                                   30   BE   DD 0021B          PUSHL   @48(SP)
                             9E4A            9E46    3C   BE   28 0021E          MOVC3   @60(SP), @(SP)+[FROM_INDEX], @(SP)+-
                                                                                [TO_INDEX]
                                     03            40   AE   E8 00225          BLBS    64(SP), 18$
                                                     0105   31 00229          BRW     25$
                                                     00F7   31 0022C 18$:      BRW     24$
                                     50            10   AE   D0 0022F 19$:      MOVL    IN_LINE_CHAR, R0                              ; 3088
                                                   6940   95 00233          TSTB    (D[N)[R0]
                                                     03   12 00236          BNEQ    20$
                                                     0088   31 00238          BRW     22$
                                     50            0C   AE   D0 0023B 20$:      MOVL    OUT_LINE_CHAR, R0                             ; 3089
                                                   6840   95 0023F          TSTB    (WLN)[R0]
                                                     7F   12 00242          BNEQ    22$
                                     7B            40   BE   E9 00244          BLBC    @64(SP), 22$                                  ; 3092
                          40   AE      28   BE       1C   AE   C1 00248          ADDL3   WLP, @40(SP), 64(SP)                        ; 3094
                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$:  MOVL    #1, RE_BORDER                                              ;
                        38   AE      34   BE     3C 00284          MOVZWL  @52(SP), 56(SP)                                            ; 3095
                                     38   AE     D7 00289          DECL    56(SP)                                                     ;
                        38   AE                  02  C6 0028C      DIVL2   #2, 56(SP)                                                 ;
                7E      28   BE      20   AE     C1 00290          ADDL3   SPEC_TO_INDEX, @40(SP), -(SP)                              ;
                                     34   BE     DD 00296          PUSHL   @52(SP)                                                    ;
                9E        9E46      40   AE     28 00299          MOVC3   64(SP), @(SP)+[FROM_INDEX], @(SP)+                         ;
                7E      24   BE      20   AE     C1 0029F          ADDL3   SPEC_TO_INDEX, @36(SP), -(SP)                              ;
                                     30   BE     DD 002A5          PUSHL   @48(SP)                                                    ;
                9E        9E46      40   AE     28 002A8          MOVC3   64(SP), @(SP)+[FROM_INDEX], @(SP)+                         ;
                            7F      40   AE     E9 002AE          BLBC    64(SP), 25$                                                ;
                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$                                                        ; 3092
                            40      40   BE     E9 002C3 22$:      BLBC    @64(SP), 23$                                               ; 3105
                        40   AE      34   BE     3C 002C7          MOVZWL  @52(SP), 64(SP)                                            ; 3107
                                     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$                                                        ; 3105
                                     28   BE     DD 00307 23$:      PUSHL   @40(SP)                                                    ; 3111
                                     34   BE     DD 0030A          PUSHL   @52(SP)                                                    ;
            9E4A          9E46      3C   BE     28 0030D          MOVC3   @60(SP), @(SP)+[FROM_INDEX], @(SP)+-                       ;
                                                                          [TO_INDEX]                                                 ;
                                     24   BE     DD 00314          PUSHL   @36(SP)                                                    ;
                                     30   BE     DD 00317          PUSHL   @48(SP)                                                    ;
            9E4A          9E46      3C   BE     28 0031A          MOVC3   @60(SP), @(SP)+[FROM_INDEX], @(SP)+-                       ;
                                                                          [TO_INDEX]                                                 ;
                                     00   BE     D5 00321          TSTL    @0(SP)                                                     ;
                                         0B  13 00324              BEQL    25$                                                        ;
                                     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                                               ; 3120
                    5A      08   AE     C0 00335          ADDL2   WCB_COLS, TO_INDEX                                                 ; 3121
                20   AE      08   AE     C0 00339          ADDL2   WCB_COLS, SPEC_TO_INDEX                                            ; 3122
```

```
                                                59  D6 0033E          INCL    DLN                                     ; 3123
                                                58  D6 00340          INCL    WLN                                     ; 3124
                                1C    AE    08  AE  C0 00342          ADDL2   WCB_COLS, WLP                           ; 3125
      FE4A        44    AE                01  48  AE  F1 00347 26$:    ACBL    72(SP), #1, R, 14$                      ; 3060
                  58          14    AE 000000FA  8F  C1 0034F 27$:    ADDL3   #250, PBCB, R8                          ; 3136
                              6D                68  E9 00358          BLBC    (R8), 31$
                  50          04    AC          1C  C1 0035B          ADDL3   #28, PP, R0                             ; 3139
                              56                60  3C 00360          MOVZWL  (R0), R6
                                                30  13 00363          BEQL    29$
                  51          04    AC          18  C1 00365          ADDL3   #24, PP, R1                             ; 3142
                              50                61  32 0036A          CVTWL   (R1), R0
                                                10  15 0036D          BLEQ    28$
                              59          0C    AE  D0 0036F          MOVL    OUT_LINE_CHAR, R9                       ; 3147
                  5A          10    AE          01  C1 00373          ADDL3   #1, IN_LINE_CHAR, R10
                6049                6A          56  28 00378          MOVC3   R6, (R10), (R0)[R9]
                                                16  11 0037D          BRB     29$
                              50          02    A7  3C 0037F 28$:     MOVZWL  2(DCB), R0                              ; 3152
                              50                56  C2 00383          SUBL2   R6, R0
                  59          0C    AE          01  C1 00386          ADDL3   #1, OUT_LINE_CHAR, R9                   ; 3153
                              5A          10    AE  D0 0038B          MOVL    IN_LINE_CHAR, R10
                  69          01  A04A          56  28 0038F          MOVC3   R6, 1(R0)[R10], (R9)
                                          15    2F  A7  E9 00395 29$:     BLBC    47(DCB), 30$                        ; 3162
                                          01    18  AE  D1 00399          CMPL    RE_BORDER, #1                       ; 3163
                                                0F  12 0039D          BNEQ    30$
                                                5B  DD 0039F          PUSHL   WCB                                     ; 3168
                                          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$:     ADDL3   #28, PP, R0                             ; 3178
                                                60  B5 003B3          TSTW    (R0)
                                                11  13 003B5          BEQL    31$
                                          0C    BE  94 003B7          CLRB    @OUT_LINE_CHAR                          ; 3185
                      0C    BE          02    AB  00  3B 003BA          SKPC    #0, 2(WCB), @OUT_LINE_CHAR              ; 3186
                                                50  D5 003C0          TSTL    BYTES_REMAINING                         ; 3189
                                                04  13 003C2          BEQL    31$
                              0C        BE          61  90 003C4          MOVB    (FOUND_BYTE), @OUT_LINE_CHAR            ; 3191
                                        50          01  D0 003C8 31$:     MOVL    #1, R0                              ; 3196
                                                    04  003CB 32$:     RET                                             ; 3197

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

; 3139          3198  1 !<BLF/PAGE>
```

```
;  3141     3199  1 %SBTTL 'SMGOCCLUDE - Check for occlusion'
;  3142     3200  1 GLOBAL ROUTINE SMG$$OCCLUDE (   LOWER : REF VECTOR [,WORD,SIGNED],
;  3143     3201  1                                 UPPER : REF VECTOR [,WORD,SIGNED],
;  3144     3202  1                                 REWRITE : REF VECTOR [,WORD,SIGNED]
;  3145     3203  1                               ) =
;  3146     3204  1 !++
;  3147     3205  1 ! FUNCTIONAL DESCRIPTION:
;  3148     3206  1 !
;  3149     3207  1 !       This routine inspects the relative positioning of two areas --
;  3150     3208  1 !       a LOWER area and an UPPER area.  It determines whether the
;  3151     3209  1 !       UPPER area occludes any portion of the LOWER area.  If occlusion
;  3152     3210  1 !       is found, a rectangular area which is the occluded part is
;  3153     3211  1 !       isolated and its coordinates are returned as the area which
;  3154     3212  1 !       must be rewritten.
;  3155     3213  1 !
;  3156     3214  1 ! CALLING SEQUENCE:
;  3157     3215  1 !
;  3158     3216  1 !       CODE.wl.v = SMG$$OCCLUDE (
;  3159     3217  1 !                                  LOWER.rw.r,
;  3160     3218  1 !                                  UPPER.rw.r,
;  3161     3219  1 !                                  REWRITE.waw.r
;  3162     3220  1 !                                 )
;  3163     3221  1 !
;  3164     3222  1 ! FORMAL PARAMETERS:
;  3165     3223  1 !
;  3166     3224  1 !       LOWER.raw.r     Descripion of a rectangular area represent
;  3167     3225  1 !                       a "lower" area.!
;  3168     3226  1 !       UPPER.raw.r     Description of a rectangular area representing
;  3169     3227  1 !                       an "upper" area.
;  3170     3228  1 !
;  3171     3229  1 !       REWRITE.waw.r   Description of a rectangular area which
;  3172     3230  1 !                       represents the area of the lower which is
;  3173     3231  1 !                       occluded by the upper ( if any).
;  3174     3232  1 !
;  3175     3233  1 !
;  3176     3234  1 ! IMPLICIT INPUTS:
;  3177     3235  1 !
;  3178     3236  1 !       NONE
;  3179     3237  1 !
;  3180     3238  1 ! IMPLICIT OUTPUTS:
;  3181     3239  1 !
;  3182     3240  1 !       NONE
;  3183     3241  1 !
;  3184     3242  1 ! ROUTINE VALUE
;  3185     3243  1 !
;  3186     3244  1 !
;  3187     3245  1 ! SIDE EFFECTS:
;  3188     3246  1 !
;  3189     3247  1 !       NONE
;  3190     3248  1 !--
;  3191     3249  1
;  3192     3250  2     BEGIN
;  3193     3251  2     LITERAL
;  3194     3252  2         ROW_START = 0,
;  3195     3253  2         NUM_ROW   = 1,
;  3196     3254  2         COL_START = 2,
;  3197     3255  2         NUM_COL   = 3;
```

```
;  3198          3256  2
;  3199          3257  2      LOCAL
;  3200          3258  2          ANS,
;  3201          3259  2          STATUS,
;  3202          3260  2          LOWER_BOTTOM_ROW: SIGNED,
;  3203          3261  2          LOWER_RIGHT_COL: SIGNED,
;  3204          3262  2          UPPER_BOTTOM_ROW: SIGNED,
;  3205          3263  2          UPPER_RIGHT_COL: SIGNED;
;  3206          3264  2
;  3207          3265  2  MACRO
;  3208          3266  2          LOWER_ROW_START          = LOWER [ROW_START]%,
;  3209          3267  2          LOWER_NO_ROW             = LOWER [NUM_ROW]%,
;  3210          3268  2          LOWER_COL_START          = LOWER [COL_START]%,
;  3211          3269  2          LOWER_NO_COL             = LOWER [NUM_COL]%,
;  3212          3270  2
;  3213          3271  2          UPPER_ROW_START          = UPPER [ROW_START]%,
;  3214          3272  2          UPPER_NO_ROW             = UPPER [NUM_ROW]%,
;  3215          3273  2          UPPER_COL_START          = UPPER [COL_START]%,
;  3216          3274  2          UPPER_NO_COL             = UPPER [NUM_COL]%,
;  3217          3275  2
;  3218          3276  2          REWRITE_ROW_START        = REWRITE [ROW_START]%,
;  3219          3277  2          REWRITE_NO_ROW           = REWRITE [NUM_ROW]%,
;  3220          3278  2          REWRITE_COL_START        = REWRITE [COL_START]%,
;  3221          3279  2          REWRITE_NO_COL           = REWRITE [NUM_COL]%;
;  3222          3280  2
;  3223          3281  2  !+
;  3224          3282  2  ! Calculate the other bounds of the specified rectangles.
;  3225          3283  2  !-
;  3226          3284  2      LOWER_BOTTOM_ROW = .LOWER_ROW_START + .LOWER_NO_ROW - 1;
;  3227          3285  2      LOWER_RIGHT_COL =  .LOWER_COL_START + .LOWER_NO_COL - 1;
;  3228          3286  2      UPPER_BOTTOM_ROW = .UPPER_ROW_START + .UPPER_NO_ROW - 1;
;  3229          3287  2      UPPER_RIGHT_COL  = .UPPER_COL_START + .UPPER_NO_COL - 1;
;  3230          3288  2
;  3231          3289  2  !+
;  3232          3290  2  ! Assume no occlusion occurs until it proves otherwise.
;  3233          3291  2  !-
;  3234          3292  2      STATUS = 0;
;  3235          3293  2
;  3236          3294  2  !+
;  3237          3295  2  ! Check for relative position of upper left-hand corner of UPPER
;  3238          3296  2  ! rectangle with respect to LOWER rectangle.
;  3239          3297  2  !-
;  3240          3298  2      ANS = SMG$$POINT_IN_RECT_R3 (UPPER_COL_START,       ! X1
;  3241          3299  2                                   UPPER_ROW_START,       ! Y1
;  3242          3300  2                                   .LOWER);
;  3243          3301  2
;  3244          3302  2      CASE .ANS FROM 0 TO 10 OF
;  3245          3303  2      SET
;  3246          3304  3          [0]:BEGIN
;  3247          3305  3              REWRITE_ROW_START = .UPPER_ROW_START;
;  3248          3306  3              REWRITE_COL_START = .UPPER_COL_START;
;  3249          3307  3              !+
;  3250          3308  3              ! Check relative position of lower right-hand corner of
;  3251          3309  3              ! UPPER rectangle with respect to LOWER rectangle.
;  3252          3310  3              !-
;  3253          3311  3              ANS = SMG$$POINT_IN_RECT_R3 (UPPER_RIGHT_COL,       ! X1
;  3254          3312  3                                          UPPER_BOTTOM_ROW,       ! Y1
```

```
;  3255     3313  3                                                .LOWER);
;  3256     3314  3
;  3257     3315  3                    CASE .ANS FROM 0 TO 6 OF
;  3258     3316  3                    SET
;  3259     3317  4                        [0]:BEGIN
;  3260     3318  4                            REWRITE_NO_ROW = .UPPER_NO_ROW;
;  3261     3319  4                            REWRITE_NO_COL = .UPPER_NO_COL;
;  3262     3320  3                            END;
;  3263     3321  3
;  3264     3322  4                        [2]:BEGIN
;  3265     3323  4                            REWRITE_NO_ROW = .UPPER_NO_ROW;
;  3266     3324  4                            REWRITE_NO_COL = .LOWER_RIGHT_COL -.UPPER_COL_START+1;
;  3267     3325  3                            END;
;  3268     3326  3
;  3269     3327  4                        [4]:BEGIN
;  3270     3328  4                            REWRITE_NO_ROW = .LOWER_BOTTOM_ROW -.UPPER_ROW_START+1;
;  3271     3329  4                            REWRITE_NO_COL = .UPPER_NO_COL;
;  3272     3330  3                            END;
;  3273     3331  3
;  3274     3332  4                        [6]:BEGIN
;  3275     3333  4                            REWRITE_NO_ROW = .LOWER_BOTTOM_ROW -.UPPER_ROW_START+1;
;  3276     3334  4                            REWRITE_NO_COL = .LOWER_RIGHT_COL - .UPPER_COL_START+1;
;  3277     3335  3                            END;
;  3278     3336  3                        [1,3,5]: RETURN (SMG$_FATERRLIB);
;  3279     3337  3
;  3280     3338  2                    TES;
;  3281     3339  2                    END;
;  3282     3340  2
;  3283     3341  3                [1]:BEGIN
;  3284     3342  3                    REWRITE_ROW_START = .UPPER_ROW_START;
;  3285     3343  3                    REWRITE_COL_START = .LOWER_COL_START;
;  3286     3344  3                    !+
;  3287     3345  3                    ! Check relative position of lower right-hand corner of
;  3288     3346  3                    ! UPPER rectangle with respect to lower rectangle.
;  3289     3347  3                    !-
;  3290     3348  3                    ANS = SMG$$POINT_IN_RECT_R3 (UPPER_RIGHT_COL,       ! X1
;  3291     3349  3                                                 UPPER_BOTTOM_ROW,      ! Y1
;  3292     3350  3                                                 .LOWER);
;  3293     3351  3
;  3294     3352  4                    CASE .ANS FROM 0 TO 6 OF
;  3295     3353  4                    SET
;  3296     3354  4                        [0]:BEGIN
;  3297     3355  4                            REWRITE_NO_ROW = .UPPER_NO_ROW;
;  3298     3356  4                            REWRITE_NO_COL = .UPPER_RIGHT_COL - .LOWER_COL_START +1;
;  3299     3357  4                            END;
;  3300     3358  4
;  3301     3359  4                        [1]:    RETURN (.STATUS);
;  3302     3360  4
;  3303     3361  4                        [2]:BEGIN
;  3304     3362  4                            REWRITE_NO_ROW = .UPPER_NO_ROW;
;  3305     3363  4                            REWRITE_NO_COL = .LOWER_NO_COL;
;  3306     3364  4                            END;
;  3307     3365  4
;  3308     3366  4                        [4]:BEGIN
;  3309     3367  4                            REWRITE_NO_ROW = .LOWER_BOTTOM_ROW -.UPPER_ROW_START+1;
;  3310     3368  4                            REWRITE_NO_COL = .UPPER_RIGHT_COL -.LOWER_COL_START+1;
;  3311     3369  3                            END;
```

```
;  3312     3370  3                    [5]:    RETURN (.STATUS);
;  3313     3371  3
;  3314     3372  3
;  3315     3373  4                    [6]:BEGIN
;  3316     3374  4                        REWRITE_NO_ROW = .LOWER_BOTTOM_ROW -.UPPER_ROW_START+1;
;  3317     3375  4                        REWRITE_NO_COL = .LOWER_NO_COL;
;  3318     3376  4                        END;
;  3319     3377  3                    [3]: RETURN (SMG$_FATERRLIB);
;  3320     3378  3
;  3321     3379  2                TES;
;  3322     3380  2                END;
;  3323     3381  1
;  3324     3382  1            [2]:    RETURN (.STATUS);
;  3325     3383  1
;  3326     3384  1            [4]:    RETURN (.STATUS);
;  3327     3385  1
;  3328     3386  1            [5]:    RETURN (.STATUS);
;  3329     3387  1
;  3330     3388  1            [6]:    RETURN (.STATUS);
;  3331     3389  1
;  3332     3390  1            [8]:BEGIN
;  3333     3391  1                REWRITE_ROW_START = .LOWER_ROW_START;
;  3334     3392  1                REWRITE_COL_START = .UPPER_COL_START;
;  3335     3393  1                !+
;  3336     3394  1                ! Check relative position of lower right-hand corner of
;  3337     3395  1                ! UPPER rectangle with respect to LOWER rectangle.
;  3338     3396  1                !-
;  3339     3397  1                ANS = SMG$$POINT_IN_RECT_R3 (UPPER_RIGHT_COL,       ! X1
;  3340     3398  1                                            UPPER_BOTTOM_ROW,       ! Y1
;  3341     3399  1                                            .LOWER);
;  3342     3400  1
;  3343     3401  1            CASE .ANS FROM 0 TO 10 OF
;  3344     3402  1            SET
;  3345     3403  2                [0]:BEGIN
;  3346     3404  2                    REWRITE_NO_ROW = .UPPER_BOTTOM_ROW -.LOWER_ROW_START+1;
;  3347     3405  2                    REWRITE_NO_COL = .UPPER_NO_COL;
;  3348     3406  2                    END;
;  3349     3407  1
;  3350     3408  2                [2]:BEGIN
;  3351     3409  2                    REWRITE_NO_ROW = .UPPER_BOTTOM_ROW -.LOWER_ROW_START+1;
;  3352     3410  2                    REWRITE_NO_COL = .LOWER_RIGHT_COL - .UPPER_COL_START+1;
;  3353     3411  2                    END;
;  3354     3412  1
;  3355     3413  2                [4]:BEGIN
;  3356     3414  2                    REWRITE_NO_ROW = .LOWER_NO_ROW;
;  3357     3415  2                    REWRITE_NO_COL = .UPPER_NO_COL;
;  3358     3416  2                    END;
;  3359     3417  1
;  3360     3418  2                [6]:BEGIN
;  3361     3419  2                    REWRITE_NO_ROW = .LOWER_NO_ROW;
;  3362     3420  2                    REWRITE_NO_COL = .LOWER_RIGHT_COL -.UPPER_COL_START+1;
;  3363     3421  2                    END;
;  3364     3422  1                [8]:    RETURN (.STATUS);
;  3365     3423  1
;  3366     3424  1                [10]:   RETURN (.STATUS);
;  3367     3425  1
;  3368     3426  1                [1,3,5,7,9]: RETURN (SMG$_FATERRLIB);
```

```
;  3369     3427  2                TES;
;  3370     3428  2                END;
;  3371     3429  2
;  3372     3430  2
;  3373     3431  2            [9]:BEGIN
;  3374     3432  2                REWRITE_ROW_START = .LOWER_ROW_START;
;  3375     3433  2                REWRITE_COL_START = .LOWER_COL_START;
;  3376     3434  2                !+
;  3377     3435  2                ! Check relative position of lower right-hand corner of
;  3378     3436  2                ! UPPER rectangle with respect to LOWER rectangle.
;  3379     3437  2                !-
;  3380     3438  2                ANS = SMG$$POINT_IN_RECT_R3 (UPPER_RIGHT_COL,       ! X1
;  3381     3439  2                                            UPPER_BOTTOM_ROW,       ! Y1
;  3382     3440  2                                            .LOWER);
;  3383     3441  2
;  3384     3442  2                CASE .ANS FROM 0 TO 10 OF
;  3385     3443  2                SET
;  3386     3444  4                    [0]:BEGIN
;  3387     3445  4                        REWRITE_NO_ROW = .UPPER_BOTTOM_ROW -.LOWER_ROW_START+1;
;  3388     3446  4                        REWRITE_NO_COL = .UPPER_RIGHT_COL -.LOWER_COL_START+1;
;  3389     3447  4                        END;
;  3390     3448  4
;  3391     3449  4                    [1]:    RETURN (.STATUS);
;  3392     3450  4
;  3393     3451  4                    [2]:BEGIN
;  3394     3452  4                        REWRITE_NO_ROW = .UPPER_BOTTOM_ROW -.LOWER_ROW_START+1;
;  3395     3453  4                        REWRITE_NO_COL = .LOWER_NO_COL;
;  3396     3454  4                        END;
;  3397     3455  4
;  3398     3456  4                    [4]:BEGIN
;  3399     3457  4                        REWRITE_NO_ROW = .LOWER_NO_ROW;
;  3400     3458  4                        REWRITE_NO_COL = .UPPER_RIGHT_COL -.LOWER_COL_START+1;
;  3401     3459  4                        END;
;  3402     3460  4
;  3403     3461  4                    [5]:    RETURN (.STATUS);
;  3404     3462  4
;  3405     3463  4                    [6]:BEGIN
;  3406     3464  4                        REWRITE_NO_ROW = .LOWER_NO_ROW;;
;  3407     3465  4                        REWRITE_NO_COL = .LOWER_NO_COL;
;  3408     3466  4                        END;
;  3409     3467  4
;  3410     3468  4                    [8]:    RETURN (.STATUS);
;  3411     3469  4
;  3412     3470  4                    [9]:    RETURN (.STATUS);
;  3413     3471  4
;  3414     3472  4                    [10]:   RETURN (.STATUS);
;  3415     3473  4
;  3416     3474  4                    [3,7]: RETURN (SMG$_FATERRLIB);
;  3417     3475  4
;  3418     3476  4                TES;
;  3419     3477  4                END;
;  3420     3478  2
;  3421     3479  2            [10]:   RETURN (.STATUS);
;  3422     3480  2
;  3423     3481  2            [3,7]: RETURN (SMG$_FATERRLIB);
;  3424     3482  2
;  3425     3483  2        TES;
```

```
;  3426     3484  2
;  3427     3485  2 !+
;  3428     3486  2 ! If we reach here, some occlusion has been detected and the output
;  3429     3487  2 ! arguments have been calculated.  All that remains is to return the
;  3430     3488  2 ! right status reporting existance of occlusion.
;  3431     3489  2 !-
;  3432     3490  2     RETURN (1);
;  3433     3491  2
;  3434     3492  1     END;                    ! End of routine SMG$$OCCLUDE


                                 0FFC 00000        .ENTRY  SMG$$OCCLUDE, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 3200
                                                           R10,R11
                              5E       08  C2 00002        SUBL2   #8, SP
                              52 04    AC  D0 00005        MOVL    LOWER, R2                             ; 3284
                                 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                            ; 3285
                              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                             ; 3286
                              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                             ; 3287
                              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                                ; 3292
                              50 04    A6  9E 00059        MOVAB   4(R6), R0                             ; 3298
                              51       56  D0 0005D        MOVL    R6, R1
                                    0000V 30 00060        BSBW    SMG$$POINT_IN_RECT_R3
                                    50  D0 00063        MOVL    R0, ANS
                                 0A 00       53  CF 00066        CASEL   ANS, #0, #10                  ; 3302
                              01B7     006F     0016     0006A 1$:  .WORD   2$-1$,-
                              01B7     01B3     01B3     00072              9$-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                             ; 3304
                                65           66   B0 00084         MOVW    (R6), (R5)                              ; 3305
                          04    A5      04   A6   B0 00087         MOVW    4(R6), 4(R5)                            ; 3306
                                51      08   AE   9E 0008C         MOVAB   UPPER_BOTTOM_ROW, R1                    ; 3311
                                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                             ; 3315
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)                             ; 3318
                                          00E1 31 000B0         BRW     24$                                     ; 3319
                          02    A5           69   B0 000B3 5$:     MOVW    (R9), 2(R5)                             ; 3323
                                             16   11 000B7         BRB     8$                                      ; 3324
                                50           66   32 000B9 6$:     CVTWL   (R6), R0                                ; 3328
                             50    6E           50   C3 000BC         SUBL3   R0, LOWER_BOTTOM_ROW, R0
                                          00B6 31 000C0         BRW     21$
                                50           66   32 000C3 7$:     CVTWL   (R6), R0                                ; 3333
                             50    6E           50   C3 000C6         SUBL3   R0, LOWER_BOTTOM_ROW, R0
                          02    A5           01   A1 000CA         ADDW3   #1, R0, 2(R5)
                                50      04   A6   32 000CF 8$:     CVTWL   4(R6), R0                               ; 3334
                             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                             ; 3341
                                65           66   B0 000DD         MOVW    (R6), (R5)                              ; 3342
                          04    A5           6A   B0 000E0         MOVW    (R10), 4(R5)                            ; 3343
                                51      08   AE   9E 000E4         MOVAB   UPPER_BOTTOM_ROW, R1                    ; 3348
                                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                             ; 3352
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$:     MOVW    (R9), 2(R5)                             ; 3355
                                             12   11 00108         BRB     14$                                     ; 3354
                          02    A5           69   B0 0010A 12$:     MOVW    (R9), 2(R5)                             ; 3362
                                             27   11 0010E         BRB     17$                                     ; 3363
                                50           66   32 00110 13$:     CVTWL   (R6), R0                                ; 3367
                             50    6E           50   C3 00113         SUBL3   R0, LOWER_BOTTOM_ROW, R0
                          02    A5           01   A1 00117         ADDW3   #1, R0, 2(R5)
                                50           6A   32 0011C 14$:     CVTWL   (R10), R0                               ; 3368
                             50    0C    AE     50   C3 0011F         SUBL3   R0, UPPER_RIGHT_COL, R0
                          06    A5           50   01   A1 00124 15$:     ADDW3   #1, R0, 6(R5)
                                             6D   11 00129         BRB     25$                                     ; 3352
                                50           66   32 0012B 16$:     CVTWL   (R6), R0                                ; 3374
                             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$:     MOVW    (R11), 6(R5)                            ; 3375
                                                   6E  11 0013B          BRB     28$                                     ; 3352
                                     55       0C   AC  D0 0013D 18$:     MOVL    REWRITE, R5                             ; 3390
                                     65            62  B0 00141          MOVW    (R2), (R5)                              ; 3391
                                04   A5       04   A6  B0 00144          MOVW    4(R6), 4(R5)                            ; 3392
                                     51       08   AE  9E 00149          MOVAB   UPPER_BOTTOM_ROW, R1                    ; 3397
                                     50       0C   AE  9E 0014D          MOVAB   UPPER_RIGHT_COL, R0
                                         0000V     30  00151             BSBW    SMG$$POINT_IN_RECT_R3
                                     53            50  D0 00154          MOVL    R0, ANS
                                0A                 00  CF 00157          CASEL   ANS, #0, #10                            ; 3401
                00C6            0025          00C6       0016     0015B 19$:     .WORD   20$-19$,-
                00C6            003F          00C6       0034     00163          39$-19$,-
                                00C2          00C6       00C2     0016B          22$-19$,-
                                                                                 39$-19$,-
                                                                                 23$-19$,-
                                                                                 39$-19$,-
                                                                                 26$-19$,-
                                                                                 39$-19$,-
                                                                                 38$-19$,-
                                                                                 39$-19$,-
                                                                                 38$-19$
                                     50            62  32 00171 20$:     CVTWL   (R2), R0                               ; 3404
                           50        08   AE       50  C3 00174          SUBL3   R0, UPPER_BOTTOM_ROW, R0
                      02   A5             50       01  A1 00179 21$:     ADDW3   #1, R0, 2(R5)
                                                   14  11 0017E          BRB     24$                                     ; 3405
                                     50            62  32 00180 22$:     CVTWL   (R2), R0                               ; 3409
                           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$                                     ; 3410
                                02   A5       04   BE  B0 0018F 23$:     MOVW    @4(SP), 2(R5)                           ; 3414
                                06   A5            68  B0 00194 24$:     MOVW    (R8), 6(R5)                             ; 3415
                                                   76  11 00198 25$:     BRB     35$                                     ; 3401
                                02   A5       04   BE  B0 0019A 26$:     MOVW    @4(SP), 2(R5)                           ; 3419
                                     50       04   A6  32 0019F 27$:     CVTWL   4(R6), R0                               ; 3420
                                     54            50  C2 001A3          SUBL2   R0, R4
                           08   A5             54       01  A1 001A6          ADDW3   #1, R4, 6(R5)
                                                   7C  11 001AB 28$:     BRB     40$                                     ; 3401
                                     54       0C   AC  D0 001AD 29$:     MOVL    REWRITE, R4                             ; 3431
                                     64            62  B0 001B1          MOVW    (R2), (R4)                              ; 3432
                                04   A4            6A  B0 001B4          MOVW    (R10), 4(R4)                            ; 3433
                                     51       08   AE  9E 001B8          MOVAB   UPPER_BOTTOM_ROW, R1                    ; 3438
                                     50       0C   AE  9E 001BC          MOVAB   UPPER_RIGHT_COL, R0
                                         0000V     30  001C0             BSBW    SMG$$POINT_IN_RECT_R3
                                     53            50  D0 001C3          MOVL    R0, ANS
                                0A                 00  CF 001C6          CASEL   ANS, #0, #10                            ; 3442
                0057            0025          0053       0016     001CA 30$:     .WORD   31$-30$,-
                0057            0048          0053       0034     001D2          38$-30$,-
                                0053          0053       0053     001DA          32$-30$,-
                                                                                 39$-30$,-
                                                                                 33$-30$,-
                                                                                 38$-30$,-
                                                                                 36$-30$,-
                                                                                 39$-30$,-
                                                                                 38$-30$,-
                                                                                 38$-30$,-
                                                                                 38$-30$
                                     50            62  32 001E0 31$:     CVTWL   (R2), R0                               ; 3445
```

```
                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$                             ; 3446
                                50              62  32  001EF   32$:    CVTWL   (R2), R0                        ; 3452
                50      08      AE              50  C3  001F2           SUBL3   R0, UPPER_BOTTOM_ROW, R0        ;
          02    A4              50              01  A1  001F7           ADDW3   #1, R0, 2(R4)                  ;
                                                19  11  001FC           BRB     37$                             ; 3453
                        02      A4      04      BE  B0  001FE   33$:    MOVW    @4(SP), 2(R4)                   ; 3457
                                50              6A  32  00203   34$:    CVTWL   (R10), R0                       ; 3458
                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$:    BRB     40$                             ; 3442
                        02      A4      04      BE  B0  00212   36$:    MOVW    @4(SP), 2(R4)                   ; 3464
                        06      A4              6B  B0  00217   37$:    MOVW    (R11), 6(R4)                    ; 3465
                                                0C  11  0021B           BRB     40$                             ; 3442
                                50              57  D0  0021D   38$:    MOVL    STATUS, R0                      ; 3479
                                                                04  00220 RET                                     ;
                                50 00000000G    8F  D0  00221   39$:    MOVL    #SMG$_FATERRLIB, R0             ; 3481
                                                                04  00228 RET                                     ;
                                50              01  D0  00229   40$:    MOVL    #1, R0                          ; 3490
                                                                04  0022C RET                                     ; 3492

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

; 3435          3493  1 !<BLF/PAGE>
```

```
;  3437     3494  1 %SBTTL 'SMG$$POINT_IN_RECT_R3 - Point inside rectangle'
;  3438     3495  1 GLOBAL ROUTINE SMG$$POINT_IN_RECT_R3 (  ! Point in rectangle
;  3439     3496  1                     X1 : REF VECTOR [,WORD,SIGNED], ! X coordinate of point
;  3440     3497  1                     Y1 : REF VECTOR [,WORD,SIGNED], ! Y coordinate of point
;  3441     3498  1                     RECT : REF VECTOR [,WORD,SIGNED]
;  3442     3499  1                                     ) : POINT_IN_RECT_LINK  =
;  3443     3500  1 !++
;  3444     3501  1 ! FUNCTIONAL DESCRIPTION:
;  3445     3502  1 !
;  3446     3503  1 !       This routine returns a code indicating the relative placement
;  3447     3504  1 ! of a point with respect to a horizontally-oriented window (that is,
;  3448     3505  1 ! one of the sides of the rectangle is parallel to the x-axis.
;  3449     3506  1 ! (See diagram below).
;  3450     3507  1 !
;  3451     3508  1 !       This is the inner-most function of the Cohen-Sutherland clipping
;  3452     3509  1 ! algorithm.
;  3453     3510  1 !
;  3454     3511  1 !       These codes have two significant properties:
;  3455     3512  1 !
;  3456     3513  1 !       1.  If the CODE of two points are each zero, both points are
;  3457     3514  1 !           within the window and no clipping needs to be done.
;  3458     3515  1 !
;  3459     3516  1 !       2.  If the logical AND of the CODEs of two points is non-zero,
;  3460     3517  1 !           the points are both outside of the window and are so
;  3461     3518  1 !           positioned that the line joining them does not intersect
;  3462     3519  1 !           the window. (I.e., it is a line segment which need not
;  3463     3520  1 !           be even partially displayed.
;  3464     3521  1 !
;  3465     3522  1 !
;  3466     3523  1 !       1001          |       1000          |       1010
;  3467     3524  1 !                     |                     |
;  3468     3525  1 !                     |                     |
;  3469     3526  1 !     ----------------+---------------------+---------- RECT [DCB_W_ROW_START]
;  3470     3527  1 !                     |                     |         ^
;  3471     3528  1 !                     |       Window        |         |
;  3472     3529  1 !                     |                     |         | RECT [DCB_W_NO_ROWS]
;  3473     3530  1 !       0001          |       0000          |  0010   |
;  3474     3531  1 !                     |                     |         V
;  3475     3532  1 !     ----------------+---------------------+----------
;  3476     3533  1 !                     |                     |
;  3477     3534  1 !       0101          |       0100          |  0110
;  3478     3535  1 !                     |                     |
;  3479     3536  1 !                     |<-RECT [DCB_W_NO_COLS]>|
;  3480     3537  1 !                     |                     |
;  3481     3538  1 !       RECT [DCB_W_COL_START]
;  3482     3539  1 !
;  3483     3540  1 !
;  3484     3541  1 ! CALLING SEQUENCE:
;  3485     3542  1 !
;  3486     3543  1 !       CODE.wl.v = SMG$$POINT_IN_RECT_R3 (X1.rw.r, Y1.rw.r, RECT.rab.r)
;  3487     3544  1 !
;  3488     3545  1 ! FORMAL PARAMETERS:
;  3489     3546  1 !
;  3490     3547  1 !       X1.rw.r         X coordinate of point
;  3491     3548  1 !       Y1.rw.r         Y coordinate of point
;  3492     3549  1 !       RECT.rab.r      Discription of the rectangular area.
;  3493     3550  1 !
```

```
;  3494     3551  1 !
;  3495     3552  1 ! IMPLICIT INPUTS:
;  3496     3553  1 !
;  3497     3554  1 !       NONE
;  3498     3555  1 !
;  3499     3556  1 ! IMPLICIT OUTPUTS:
;  3500     3557  1 !
;  3501     3558  1 !       NONE
;  3502     3559  1 !
;  3503     3560  1 ! ROUTINE VALUE
;  3504     3561  1 !
;  3505     3562  1 !       One of the 9 codes indicated in the diagram above.
;  3506     3563  1 !
;  3507     3564  1 ! SIDE EFFECTS:
;  3508     3565  1 !
;  3509     3566  1 !       NONE
;  3510     3567  1 !--
;  3511     3568  1
;  3512     3569  2     BEGIN
;  3513     3570  2
;  3514     3571  2     LITERAL
;  3515     3572  2         ROW_START = 0,
;  3516     3573  2         NUM_ROW   = 1,
;  3517     3574  2         COL_START = 2,
;  3518     3575  2         NUM_COL   = 3;
;  3519     3576  2
;  3520     3577  3     RETURN (
;  3521     3578  4         (
;  3522     3579  4         IF .X1[0] LSS .RECT [COL_START]
;  3523     3580  4         THEN                    %B'0001'    ! to left
;  3524     3581  4         ELSE
;  3525     3582  5             BEGIN
;  3526     3583  6             IF .X1[0] GEQ (.RECT [COL_START] + .RECT [NUM_COL])
;  3527     3584  5             THEN                %B'0010'    ! to right
;  3528     3585  5             ELSE                0           ! neither to right or left
;  3529     3586  5             END
;  3530     3587  4         )
;  3531     3588  3         +
;  3532     3589  4         (
;  3533     3590  5         IF .Y1[0] GEQ (.RECT [ROW_START] + .RECT [NUM_ROW])
;  3534     3591  4         THEN                    %B'0100'    ! below bottom
;  3535     3592  4         ELSE
;  3536     3593  5             BEGIN
;  3537     3594  5             IF .Y1[0] LSS .RECT [ROW_START]
;  3538     3595  5             THEN                %B'1000'    ! above top
;  3539     3596  5             ELSE                0           ! Neither above or below
;  3540     3597  5             END
;  3541     3598  2         ) ) ;
;  3542     3599  2
;  3543     3600  2
;  3544     3601  1     END;                    ! End of routine SMG$$POINT_IN_RECT_R3



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

```
                                          SUBL2   #4, SP                  ; 3495
          04  A2        60  B1 00003      CMPW    (X1), 4(RECT)           ; 3579
                        05  18 00007      BGEQ    1$
              53        01  D0 00009      MOVL    #1, R3
                        1C  11 0000C      BRB     4$
              53    04  A2  32 0000E 1$:  CVTWL   4(RECT), R3             ; 3583
              6E    06  A2  32 00012      CVTWL   6(RECT), (SP)
              53        6E  C0 00016      ADDL2   (SP), R3
  53      60  10        00  EC 00019      CMPV    #0, #16, (X1), R3
                        05  19 0001E      BLSS    2$
              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                  ; 3582
              50        62  32 0002A 4$:  CVTWL   (RECT), R0              ; 3590
              6E    02  A2  32 0002D      CVTWL   2(RECT), (SP)
              50        6E  C0 00031      ADDL2   (SP), R0
  50      61  10        00  EC 00034      CMPV    #0, #16, (Y1), R0
                        05  19 00039      BLSS    5$
              50        04  D0 0003B      MOVL    #4, R0
                        0C  11 0003E      BRB     7$
              62        61  B1 00040 5$:  CMPW    (Y1), (RECT)            ; 3594
                        05  18 00043      BGEQ    6$
              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                  ; 3589
              50        53  D0 0004F      MOVL    R3, R0                  ; 3577
              5E        04  C0 00052      ADDL2   #4, SP                  ; 3601
                            05 00055      RSB

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


; 3545          3602  1 !<BLF/PAGE>
```

```
; 3547          3603  1 END                       ! End of module SMG$DISPLAY_OUTPUT
; 3548          3604  1
; 3549          3605  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
;  _$255$DUA18:[SYSLIB]STARLET.L32;1               9776        16         0         581       00:01.0
;  _$255$DUA18:[SMGRTL.OBJ]RTLLIB.L32;1              36         0         0           8       00:00.1
;  _$255$DUA18:[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$:SMGDISOUT/UPDATE=(BUG$:SMGDISOUT
;       )

; Size:          4667 code + 1 data bytes
; Run Time:         01:41.9
; Elapsed Time:     02:01.0
; Lines/CPU Min:     2123
; Lexemes/CPU-Min: 19673
; Memory Used:  537 pages
; Compilation Complete
```

0446 AH-EF71A-SE
VAX/VMS V4.1 SRC LST MCRF UPD

SMGMIN
LIS

SMGDISOUT
LIS