

```
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGGGG  RRRRRRRRRRRR  TTTTTTTTTTTTTT  LLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGGGG  RRRRRRRRRRRR  TTTTTTTTTTTTTT  LLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGGGG  RRRRRRRRRRRR  TTTTTTTTTTTTTT  LLL
SSS             MMMMMM  MMMMMM  GGG             RRR      RRR      TTT      LLL
SSS             MMMMMM  MMMMMM  GGG             RRR      RRR      TTT      LLL
SSS             MMMMMM  MMMMMM  GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSSSSSSSSSS    MMM      MMM      GGG             RRRRRRRRRRRR  TTT      LLL
SSSSSSSSSSS    MMM      MMM      GGG             RRRRRRRRRRRR  TTT      LLL
SSSSSSSSSSS    MMM      MMM      GGG             RRRRRRRRRRRR  TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSS             MMM      MMM      GGG             RRR      RRR      TTT      LLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGG      RRR      RRR      TTT      LLLLLLLLLLLLLLLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGG      RRR      RRR      TTT      LLLLLLLLLLLLLLLL
SSSSSSSSSSSSS  MMM      MMM      GGGGGGGGGG      RRR      RRR      TTT      LLLLLLLLLLLLLLLL
```

```

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

```

.....
.....
.....
.....

```

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

```



```
0001 0 %TITLE 'SMG$$DISPLAY_INPUT - Input support routines'
0002 0 MODULE SMG$$DISPLAY_INPUT (
0003 0 IDENT = '1-026' ! File: SMGDISINP.B32 Edit: STAN1026
0004 0 ) =
0005 1 BEGIN
0006 1
0007 1 *****
0008 1 *
0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0011 1 * ALL RIGHTS RESERVED.
0012 1 *
0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0018 1 * TRANSFERRED.
0019 1 *
0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0022 1 * CORPORATION.
0023 1 *
0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0026 1 *
0027 1 *
0028 1 *****
0029 1
0030 1
0031 1 ++
0032 1 FACILITY:      Screen Management
0033 1
0034 1 ABSTRACT:
0035 1 The procedures in this module act as interfaces between the
0036 1 virtual displays and pasteboards and associated data structures, and
0037 1 the keyboard input side of SMG. These routines are called to pass
0038 1 information about changes to the physical screen that have been
0039 1 brought about by input-related activities.
0040 1
0041 1 ENVIRONMENT:  User mode, Shared library routines.
0042 1
0043 1 AUTHOR: R. Reichert, CREATION DATE: 9-Mar-1983
0044 1
0045 1 MODIFIED BY:
0046 1
0047 1 1-001 - Original. Skeleton for future code. RKR 9-Mar-1983
0048 1 1-002 - Correct names of data structures and macros. PLL 15-Mar-1983
0049 1 1-003 - Add $SMG$VALIDATE_ARGCOUNT invocations. RKR 30-Mar-1983.
0050 1 1-004 - Flesh out some of the routines. RKR 11-APR-1983.
0051 1 1-005 - Add SMG$$GET_PASTEBOARD_ID. RKR 14-APR-1983.
0052 1 1-006 - Tap into output side. RKR 26-APR-1983.
0053 1 1-007 - Debug SMG$$SET_PHYSICAL_CURSOR. RKR 26-APR-1983
0054 1 1-008 - Start debugging SMG$$REPORT_CHANGE_xxxxx. RKR 29-APR-1983
0055 1 1-009 - Take advantage of some new routines.
0056 1 RKR 15-MAY-1983.
0057 1 1-010 - Delete external references to DD_ structures and counts -- no
```

```

58      0058 1 | longer needed (or available).
59      0059 1 | RKR 20-MAY-1983.
60      0060 1 | 1-011 - Start using fields in PBCB which tell SMG$$MIN_UPD what part
61      0061 1 | of WCB buffers changed.
62      0062 1 | RKR 28-MAY-1983.
63      0063 1 | 1-012 - Inform output that the physical cursor position is unknown.
64      0064 1 | Fix bug wherein NUM_CHARS was treated as the address of
65      0065 1 | a longword instead of a word.
66      0066 1 | STAN 28-May-1983.
67      0067 1 | 1-013 - Further debugging of SMG$$SET_PHYSICAL_CURSOR.
68      0068 1 | RKR 27-MAY-1983.
69      0069 1 | 1-014 - Rename PBCB_B_COLS to PBCB_W_WIDTH
70      0070 1 | STAN 1-May-1983
71      0071 1 | 1-015 - Make changed_row and changed_col optional arguments to
72      0072 1 | SMG$$REPORT_CHANGE_REPLACE. PLL 16-Jun-1983
73      0073 1 | 1-016 - Rearrange order of arguments to SMG$$SET_PHYSICAL_CURSOR, and
74      0074 1 | add two new arguments. PLL 21-Jun-1983
75      0075 1 | 1-017 - In SMG$$SET_PHYSICAL_CURSOR:
76      0076 1 |     Validate right number of arguments.
77      0077 1 |     Conditionalize some code so that some computations are
78      0078 1 |     done only if optional parameter(s) are present.
79      0079 1 |     In SMG$$MOVE_TEXT_TO_SCREEN_BUFFER:
80      0080 1 |     Optimize some code paths to achieve same optimizations
81      0081 1 |     as in SMG$$MOVE_TEXT_TO_WINDOW_BUF (in SMGDISOUT.B32).
82      0082 1 |     Throughout, switch to data structure named declarations.
83      0083 1 | RKR 24-JUN-1983.
84      0084 1 | 1-018 - Cleanup calling sequence to SMG$$SET_PHYSICAL_CURSOR, based on
85      0085 1 | how the input routines call it. PLL 27-Jun-1983
86      0086 1 | 1-019 - Pass physical device type to SMG$$SET_CURSOR_ABS_R4 if the
87      0087 1 | device is a foreign terminal. PLL 2-Aug-1983
88      0088 1 | 1-020 - Add logic to SMG$$SET_PHYSICAL_CURSOR to set scrolling region
89      0089 1 | to full screen if we are about to do input from a VT100.
90      0090 1 | RKR 3-AUG-1983.
91      0091 1 | 1-021 - Speed up SMG$$SET_PHYSICAL_CURSOR by looking at occluded
92      0092 1 | bit in PP to bypass occlusion tests. RKR 12-SEP-1983.
93      0093 1 | 1-022 - Fix the way SMG$$SET_PHYSICAL_CURSOR calculates remaining columns
94      0094 1 | available for input. No properly recognizing where the virtual
95      0095 1 | display is pasted.
96      0096 1 | RKR 14-DEC-1983
97      0097 1 | 1-023 - Minor changes for TERMTABLE support. STAN 22-JAN-1984
98      0098 1 | 1-024 - More. STAN 14-Feb-1984.
99      0099 1 | 1-025 - STAN 7-Mar-1984. Moved report_change routines to file SMGPRVINP.B32
100     0100 1 | 1-026 - STAN 19-Mar-1984. Fix bug in remaining columns calculation for
101     0101 1 | unoccluded virtual display not pasted to column 1 of pasteboard.
102     0102 1 | --

```



```
: 104      0103 1 %SBTTL 'Declarations'
: 105      0104 1
: 106      0105 1 SWITCHES:
: 107      0106 1
: 108      0107 1
: 109      0108 1
: 110      0109 1 LINKAGES:
: 111      0110 1
: 112      0111 1 NONE
: 113      0112 1
: 114      0113 1 TABLE OF CONTENTS:
: 115      0114 1
: 116      0115 1
: 117      0116 1 FORWARD ROUTINE
: 118      0117 1
: 119      0118 1 ! Private entry points:
: 120      0119 1
: 121      0120 1 SMG$$GET_PASTEBOARD_ID, ! Find pasteboard id to match
: 122      0121 1 ! device name.
: 123      0122 1 SMG$$MOVE_TEXT_TO_SCREEN_BUF, ! Map a single virtual display
: 124      0123 1 ! to the WCB screen buffer.
: 125      0124 1
: 126      0125 1 SMG$$SET_PHYSICAL_CURSOR; ! Set physical cursor on screen
: 127      0126 1
: 128      0127 1
: 129      0128 1 INCLUDE FILES
: 130      0129 1
: 131      0130 1
: 132      0131 1 REQUIRE 'RTLIN:SMGPROLOG'; ! defines psects, macros, tcb,
: 133      0209 1 ! wcb, & terminal symbols
: 134      0210 1
: 135      0211 1
: 136      0212 1 EXTERNAL REFERENCES
: 137      0213 1
: 138      0214 1
: 139      0215 1 EXTERNAL
: 140      0216 1 PBD_L_COUNT, ! No. of pasteboards we currently have
: 141      0217 1
: 142      0218 1 PBD_A_PBCB : VECTOR [PBD_K_MAX_PB, LONG],
: 143      0219 1 ! Table of addresses of PBCB's
: 144      0220 1
: 145      0221 1 PBD_V_PB_AVAIL : BITVECTOR [PBD_K_MAX_PB];
: 146      0222 1 ! Bit vector of pasteboard id numbers in use.
: 147      0223 1
: 148      0224 1 EXTERNAL ROUTINE
: 149      0225 1 LIB$GET_VM, ! Allocate heap storage
: 150      0226 1
: 151      0227 1 SMG$INSERT_CHARS, ! Insert char in virtual display buffer
: 152      0228 1 ! and cause output.
: 153      0229 1
: 154      0230 1 SMG$$FILL_WINDOW_BUFFER, ! Map all virtual display buffers to
: 155      0231 1 ! the window buffer for a given PBCB
: 156      0232 1
: 157      0233 1 SMG$$FLUSH_BUFFER, ! Flush any pending output to terminal
: 158      0234 1
: 159      0235 1 SMG$$FORCE_SCROLL_REG, ! Force scroll region to specified
: 160      0236 1 ! lines.
```



```

: 161      0237 1
: 162      0238 1      SMG$$LOCATE_PP,      ! Locate pasting packet that joins a virtual
: 163      0239 1      ! display to a pasteboard.
: 164      0240 1
: 165      0241 1      SMG$$MOVE_TEXT_TO_WINDOW_BUF,      ! Map single virtual display to
: 166      0242 1      ! window buffer.
: 167      0243 1
: 168      0244 1      SMG$$OCCLUDE,      ! Determine overlap between two rectangular
: 169      0245 1      ! areas.
: 170      0246 1      SMG$$MIN_UPD,      ! Minimum output routine
: 171      0247 1
: 172      0248 1      SMG$$PUT_TEXT_TO_BUFFER,      ! Text to virtual display buffer
: 173      0249 1
: 174      0250 1      SMG$$REPORT_CHANGE_INSERT,      ! Report change to physical
: 175      0251 1      ! screen in insert mode.
: 176      0252 1
: 177      0253 1      SMG$$REPORT_CHANGE_REPLACE;      ! Report change to physical
: 178      0254 1      ! screen in replaec mode.
: 179      0255 1
: 180      0256 1      EXTERNAL LITERAL
: 181      0257 1
: 182      0258 1      SMG$_FATERRLIB,      ! Fatal error in library procedure
: 183      0259 1      SMG$_INVARG,      ! Invalid argument
: 184      0260 1      SMG$_INVCOL,      ! Invalid column number
: 185      0261 1      SMG$_INVDIS_ID,      ! Invalid virtual display id
: 186      0262 1      SMG$_INVPAS_ID,      ! Invalid pasteboard id
: 187      0263 1      SMG$_INVROW;      ! Invalid row number
: 188      0264 1
: 189      0265 1      !<BLF/PAGE>

```



```

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



```

: 248 0323 2 ! control block.
: 249 0324 2
: 250 0325 2 $SMG$VALIDATE_ARGCOUNT (3, 3); ! Test for right no. of args
: 251 0326 2
: 252 0327 2 !+
: 253 0328 2 !- If we don't have any pasteboards yet, it can't match. Return error.
: 254 0329 2
: 255 0330 2 IF .PBD_L_COUNT LEQ 0
: 256 0331 2 THEN
: 257 0332 2 RETURN ( SMG$_INVARG);
: 258 0333 2
: 259 0334 2 !+
: 260 0335 2 !- Loop through all the pasteboards we've got trying to match name.
: 261 0336 2
: 262 0337 2 INCR I FROM 0 TO .PBD_L_COUNT -1
: 263 0338 2 DO
: 264 0339 3 BEGIN ! Loop thru pasteboards
: 265 0340 3 IF (PBCB = .PBD_A_PBCB [.I]) NEQ 0
: 266 0341 3 THEN
: 267 0342 4 BEGIN ! A valid pasteboard address
: 268 0343 4 IF .DEVNAM_LEN [0] EQL .PBCB [PBCB_W_DEVNAM_LEN]
: 269 0344 4 THEN
: 270 0345 5 BEGIN ! Lengths match
: 271 0346 5 IF CH$EQL ( .DEVNAM_LEN [0], ! length
: 272 0347 5 .DEVNAM_ADDR, ! address
: 273 0348 5 .PBCB [PBCB_W_DEVNAM_LEN], ! length
: 274 0349 5 PBCB [PBCB_T_DEVNAM]) ! address
: 275 0350 5 THEN
: 276 0351 6 BEGIN ! Match found
: 277 0352 6 .PASTEBOARD_ID = .I;
: 278 0353 6 RETURN ( SS$ NORMAL);
: 279 0354 5 END; ! Match found
: 280 0355 4 END; ! Lengths match
: 281 0356 3 END; ! A valid pasteboard address
: 282 0357 2 END; ! Loop thru pasteboards
: 283 0358 2
: 284 0359 2 !+
: 285 0360 2 !- If we fall out of loop, none of our pasteboards are associated with
: 286 0361 2 the given string. Return error code.
: 287 0362 2
: 288 0363 2 RETURN (SMG$_INVARG);
: 289 0364 1 END; ! End of routine SMG$$GET_PASTEBOARD_ID

```

.TITLE SMG\$\$DISPLAY_INPUT SMG\$\$DISPLAY INPUT - Input s
 support routines

.IDENT \1-026\

.EXTRN PBD_L_COUNT, PBD_A_PBCB
 .EXTRN PBD_V-PB_AVAIL, CIB\$GET_VM
 .EXTRN SMG\$INSERT_CHARS
 .EXTRN SMG\$\$FILL_WINDOW_BUFFER
 .EXTRN SMG\$\$FLUSH_BUFFER
 .EXTRN SMG\$\$FORCE_SCROLL_REG
 .EXTRN SMG\$\$LOCATE_PP, SMG\$\$MOVE_TEXT_TO_WINDOW_BUF
 .EXTRN SMG\$\$OCCLUDE, SMG\$\$MIN_UPD
 .EXTRN SMG\$\$PUT_TEXT_TO_BUFFER


```

                                007C 00000
03                               6C 91 00002
                                08 13 00005
50 00000000G                   8F D0 00007
                                04 0000E
56 00000000G                   00 D0 0000F 1$:
                                2E 15 00016
55                               01 CE 00018
                                25 11 0001B
54 00000000G0045 D0 0001D 2$:
                                1B 13 00025
12 A4 04 BC B1 00027
                                14 12 0002C
08 BC 04 BC 2D 0002E
                                18 A4 00036
                                08 12 00038
0C BC 55 D0 0003A
50 01 D0 0003E
                                04 00041
D7 55 56 F2 00042 3$:
50 00000000G 8F D0 00046 4$:
                                04 0004D

```

```

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

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

.ENTRY SMG$$GET_PASTEBOARD_ID, Save R2,R3,R4,R5,R6 : 0267
CMPB (AP), #3 : 0325
BEQL 1$ :
MOVL #SMG$_WRONUMARG, R0 :
RET :
MOVL PBD_L_COUNT, R6 : 0330
BLEQ 4$ :
MNEGL #1, I : 0337
BRB 3$ :
MOVL PBD_A_PBCB[I], PBCB : 0340
BEQL 3$ :
CMPW @DEVNAM_LEN, 18(PBCB) : 0343
BNEQ 3$ :
CMPC5 @DEVNAM_LEN, @DEVNAM_ADDR, #0, 18(PBCB), - : 0349
24(PBCB) :
BNEQ 3$ :
MOVL I, @PASTEBOARD_ID : 0352
MOVL #1, R0 : 0353
RET :
AOBLSS R6, I, 2$ : 0337
MOVL #SMG$_INVARG, R0 : 0363
RET : 0364

```

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

; 290 0365 1 !<BLF/PAGE>


```

: 292      0366 1 %SBTTL 'SMG$$MOVE_TEXT_TO_SCREEN_BUF - Move text from display buf. to screen buf.'
: 293      0367 1 GLOBAL ROUTINE SMG$$MOVE_TEXT_TO_SCREEN_BUF (
: 294      0368 1
: 295      0369 1 PP : REF BLOCK [,BYTE]
: 296      0370 1 ) =
: 297      0371 1 ++
: 298      0372 1 FUNCTIONAL DESCRIPTION:
: 299      0373 1
: 300      0374 1 This routine moves text from the buffer located at
: 301      0375 1 .DCB [DCB_A_TEXT_BUF] into the screen text buffer.
: 302      0376 1 Array of bytes at .DCB [DCB_A_ATTR_BUF] describe the
: 303      0377 1 rendition this text must assume and is moved into the
: 304      0378 1 associated screen attribute buffer.
: 305      0379 1 Similarly, if the alternate character set buffer at
: 306      0380 1 .DCB [DCB_A_CHAR_SET_BUF] exists, it must be mapped into the
: 307      0381 1 screen alternate character set buffer.
: 308      0382 1
: 309      0383 1 CALLING SEQUENCE:
: 310      0384 1
: 311      0385 1 ret_status.wlc.v = SMG$$MOVE_TEXT_TO_SCREEN_BUF ( PP.rab.r)
: 312      0386 1
: 313      0387 1 FORMAL PARAMETERS:
: 314      0388 1
: 315      0389 1 PP.rab.r Address of pasting packet.
: 316      0390 1
: 317      0391 1 IMPLICIT INPUTS:
: 318      0392 1
: 319      0393 1 NONE
: 320      0394 1
: 321      0395 1 IMPLICIT OUTPUTS:
: 322      0396 1
: 323      0397 1 NONE
: 324      0398 1
: 325      0399 1 COMPLETION STATUS:
: 326      0400 1
: 327      0401 1 SS$_NORMAL Normal successful completion
: 328      0402 1
: 329      0403 1 SIDE EFFECTS:
: 330      0404 1
: 331      0405 1 NONE
: 332      0406 1
: 333      0407 2 BEGIN
: 334      0408 2 LOCAL
: 335      0409 2 DCB : REF $DCB_DECL, ! Addr of virtual display
: 336      0410 2 ! control block.
: 337      0411 2 PBCB : REF $PBCB_DECL, ! Addr of pasteboard control
: 338      0412 2 ! block
: 339      0413 2 WCB : REF $WCB_DECL, ! Addr of window control block
: 340      0414 2 FROM_INDEX, ! Index into source buffer
: 341      0415 2 TO_INDEX; ! Index into destination buffer
: 342      0416 2
: 343      0417 2 DCB = .PP [PP_A_DCB_ADDR];
: 344      0418 2 PBCB = .PP [PP_A_PBCB_ADDR];
: 345      0419 2 WCB = .PBCB [PBCB_A_WCB];
: 346      0420 2
: 347      0421 2 FROM_INDEX = .PP [PP_W_FROM_INDEX];
: 348      0422 2 TO_INDEX = .PP [PP_W_TO_INDEX];

```



```

349      0423 2
350      0424 2
351      0425 2 +
352      0426 2 Before diverging on two copying paths, check to see if we are going
353      0427 2 to get involved with alternate character set buffers. If one exists
354      0428 2 in the DCB but does not yet exist in the WCB, we have to allocate
355      0429 2 one for the WCB and initialize it.
356      0430 2 -
357      0431 2 IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0 AND
358      0432 2 .WCB [WCB_A_SCR_CHAR_SET_BUF] EQL 0
359      0433 2 THEN
360      0434 2 BEGIN ! Alloc. and init. window alternate char. set buffer
361      0435 2 LOCAL
362      0436 2 STATUS; ! Status of LIB$GET_VM call
363      0437 2
364      0438 2 IF NOT (STATUS = LIB$GET_VM ( WCB [WCB_BUFSIZE],
365      0439 2 WCB [WCB_A_SCR_CHAR_SET_BUF]))
366      0440 2 THEN
367      0441 2 RETURN (.STATUS);
368      0442 2
369      0443 2 CH$FILL ( 0, .WCB [WCB_L_BUFSIZE], .WCB [WCB_A_SCR_CHAR_SET_BUF]);
370      0444 2 END; ! Alloc. and init. window alternate char. set buffer
371      0445 2
372      0446 2 +
373      0447 2 Check to see if we can do it with a single CH$MOVE or whether we must
374      0448 2 do it a row at a time.
375      0449 2 -
376      0450 2 IF .PP [PP_V_CONTIG]
377      0451 2 THEN
378      0452 2 BEGIN ! Can be done in single move
379      0453 2 ! Move text
380      0454 2 CH$MOVE ( .PP [PP_L_MOVE_SIZE], ! Length
381      0455 2 .DCB [DCB_A_TEXT_BUF] + .FROM_INDEX, ! source
382      0456 2 .WCB [WCB_A_SCR_TEXT_BUF] + .TO_INDEX); ! dest.
383      0457 2
384      0458 2 ! Move attributes
385      0459 2 CH$MOVE ( .PP [PP_L_MOVE_SIZE], ! Length
386      0460 2 .DCB [DCB_A_ATTR_BUF] + .FROM_INDEX, ! source
387      0461 2 .WCB [WCB_A_SCR_ATTR_BUF] + .TO_INDEX); ! dest.
388      0462 2
389      0463 2 ! Move alternate character set buffer pieces, if necessary
390      0464 2 IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
391      0465 2 THEN
392      0466 2 BEGIN ! Map alternate character set
393      0467 2 CH$MOVE ( .PP [PP_L_MOVE_SIZE], ! Length
394      0468 2 .DCB [DCB_A_CHAR_SET_BUF] + .FROM_INDEX, ! Source
395      0469 2 .WCB [WCB_A_SCR_CHAR_SET_BUF] + .TO_INDEX); ! Dest.
396      0470 2
397      0471 2 END; ! Map alternate character set
398      0472 2 ! Can be done in single move
399      0473 2 END
400      0474 2 ELSE
401      0475 2 BEGIN ! Must be done row at a time
402      0476 2 LOCAL
403      0477 2
404      0478 2
405      0479 2

```



```

406      0480 3      DCB_COLS,  ! = .DCB [DCB_W_NO_COLS]
407      0481 3      WCB_COLS;  ! = .WCB [WCB_W_NO_COLS]
408      0482 3
409      0483 3
410      0484 3      +
411      0485 3      | Extracting out these two words as local longwords makes
412      0486 3      | compiler generate better code for this critical loop.
413      0487 3      -
414      0488 3      DCB_COLS = .DCB [DCB_W_NO_COLS];
415      0489 3      WCB_COLS = .WCB [WCB_W_NO_COLS];
416      0490 3
417      0491 3      INCR R FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
418      0492 4      DO
419      0493 4      BEGIN      ! For all rows in this display
420      0494 4      | Move text
421      0495 4      CH$MOVE ( .PP [PP_W_MOVE_LENGTH],      ! length
422      0496 4      .DCB [DCB_A_TEXT_BUF] + .FROM_INDEX,  ! source
423      0497 4      .WCB [WCB_A_SCR_TEXT_BUF] + .TO_INDEX); ! dest.
424      0498 4
425      0499 4      | Move attributes
426      0500 4      CH$MOVE ( .PP [PP_W_MOVE_LENGTH],      ! length
427      0501 4      .DCB [DCB_A_ATTR_BUF] + .FROM_INDEX,  ! source
428      0502 4      .WCB [WCB_A_SCR_ATTR_BUF] + .TO_INDEX); ! dest
429      0503 4
430      0504 4
431      0505 4      FROM_INDEX = .FROM_INDEX + .DCB_COLS;
432      0506 4      TO_INDEX   = .TO_INDEX   + .WCB_COLS;
433      0507 3      END;      ! For all rows in this display
434      0508 3
435      0509 3
436      0510 3      +
437      0511 3      | Move alternate character set buffer pieces, if necessary.
438      0512 3      -
439      0513 3      IF .DCB [DCB_A_CHAR_SET_BUF] NEQ 0
440      0514 4      THEN
441      0515 4      BEGIN      ! Map alt. char. set buffer
442      0516 4      FROM_INDEX = .PP [PP_W_FROM_INDEX];
443      0517 4      TO_INDEX   = .PP [PP_W_TO_INDEX];
444      0518 4
445      0519 4      INCR R FROM 1 TO .PP [PP_W_ROWS_TO_MOVE]
446      0520 5      DO
447      0521 5      BEGIN
448      0522 5      CH$MOVE ( .PP [PP_W_MOVE_LENGTH],      ! length
449      0523 5      .DCB [DCB_A_CHAR_SET_BUF] + .FROM_INDEX, ! source
450      0524 5      .WCB [WCB_A_SCR_CHAR_SET_BUF] + .TO_INDEX); ! dest.
451      0525 5
452      0526 5      FROM_INDEX = .FROM_INDEX + .DCB_COLS;
453      0527 4      TO_INDEX   = .TO_INDEX   + .WCB_COLS;
454      0528 3      END;
455      0529 2      END;      ! Map alt. char. set buffer
456      0530 2      END;      ! Must be done row at a time
457      0531 2
458      0532 1      RETURN (SS$_NORMAL);
459      0533 1      END;      ! End of routine SMG$$MOVE_TEXT_TO_SCREEN_BUF

```


				OFFC 00000	.ENTRY	SMG\$\$MOVE_TEXT_TO_SCREEN_BUF, Save R2,R3,-	0367
						R4,R5,R6,R7,R8,R9,R10,R11	
		5E		10 C2 00002	SUBL2	#16, SP	
		57	04	AC D0 00005	MOVL	PP, R7	0417
		59	10	A7 D0 00009	MOVL	16(R7), DCB	
		50	14	A7 D0 0000D	MOVL	20(R7), PBCB	0418
		56	08	A0 D0 00011	MOVL	8(PBCB), WCB	0419
		5A	1E	A7 3C 00015	MOVZWL	30(R7), FROM_INDEX	0421
		58	20	A7 3C 00019	MOVZWL	32(R7), TO_INDEX	0422
		5B	18	A9 D0 0001D	MOVL	24(DCB), RT1	0430
				7E D4 00021	CLRL	-(SP)	
				5B D5 00023	TSTL	R11	
				20 13 00025	BEQL	2\$	
				6E D6 00027	INCL	(SP)	
			1C	A6 D5 00029	TSTL	28(WCB)	0431
				19 12 0002C	BNEQ	2\$	
			1C	A6 9F 0002E	PUSHAB	28(WCB)	0438
			28	A6 9F 00031	PUSHAB	40(WCB)	0437
		00000000G 00		02 FB 00034	CALLS	#2, LIB\$GET_VM	0438
		01		50 E8 0003B	BLP	STATUS, 1\$	
				04 0003E	RET		
28	A6	00	6E	00 2C 0003F 1\$:	MOVC5	#0, (SP), #0, 40(WCB), a28(WCB)	0442
				1C B6 00045			
		1F	2A A7	01 E1 00047 2\$:	BBC	#1, 42(R7), 3\$	0449
14	B648	10	B94A	2B A7 28 0004C	MOVC3	43(R7), a16(DCB)[FROM_INDEX], a20(WCB)-	0457
						[TO_INDEX]	
18	B648	14	B94A	2B A7 28 00055	MOVC3	43(R7), a20(DCB)[FROM_INDEX], a24(WCB)-	0463
						[TO_INDEX]	
		62		6E E9 0005E	BLBC	(SP), 8\$	0467
1C	B648	6A4B	2B	A7 28 00061	MOVC3	43(R7), (FROM_INDEX)[R11], a28(WCB)-	0472
						[TO_INDEX]	
				58 11 00069	BRB	8\$	0449
		10	AE	06 A9 3C 0006B 3\$:	MOVZWL	6(DCB), DCB_COLS	0487
		0C	AE	06 A6 3C 00070	MOVZWL	6(WCB), WCB_COLS	0488
		08	AE	1C A7 3C 00075	MOVZWL	28(R7), 8(SP)	0490
				04 AE D4 0007A	CLRL	R	0506
				1A 11 0007D	BRB	5\$	
14	B648	10	B94A	22 A7 28 0007F 4\$:	MOVC3	34(R7), a16(DCB)[FROM_INDEX], a20(WCB)-	0497
						[TO_INDEX]	
18	B648	14	B94A	22 A7 28 00088	MOVC3	34(R7), a20(DCB)[FROM_INDEX], a24(WCB)-	0503
						[TO_INDEX]	
		5A		10 AE C0 00091	ADDL2	DCB_COLS, FROM_INDEX	0505
		58		0C AE C0 00095	ADDL2	WCB_COLS, TO_INDEX	0506
	E0	04	AE	08 AE F3 00099 5\$:	AOBLEQ	8(SP), R, 4\$	0490
		21		6E E9 0009F	BLBC	(SP), 8\$	0512
		5A	1E	A7 3C 000A2	MOVZWL	30(R7), FROM_INDEX	0515
		58	20	A7 3C 000A6	MOVZWL	32(R7), TO_INDEX	0516
				59 D4 000AA	CLRL	R	0523
				10 11 000AC	BRB	7\$	
1C	B648	6A4B	22	A7 28 000AE 6\$:	MOVC3	34(R7), (FROM_INDEX)[R11], a28(WCB)-	
						[TO_INDEX]	
		5A	10	AE C0 000B6	ADDL2	DCB_COLS, FROM_INDEX	0525
		58	0C	AE C0 000BA	ADDL2	WCB_COLS, TO_INDEX	0526
	EB	59	08	AE F3 000BE 7\$:	AOBLEQ	8(SP), R, 6\$	0518
		50	01	D0 000C3 8\$:	MOVL	#1, R0	0531
				04 000C6	RET		0532

SMG\$\$DISPLAY_IN SMG\$\$DISPLAY INPUT - Input support routines
1-026 SMG\$\$MOVE_TEXT_TO_SCREEN_BUF - Move text from d

K 10

16-Sep-1984 00:27:47
14-Sep-1984 13:09:42

VAX-11 Bliss-32 V4.0-742
[SMGRTL.SRC]SMGDISINP.B32;1

Page 12
(4)

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

; 459 0533 1 !<BLF/PAGE>


```

461 0534 1 %SBTTL 'SMG$$SET PHYSICAL CURSOR - Set physical cursor'
462 0535 1 GLOBAL ROUTINE SMG$$SET_PHYSICAL_CURSOR ( DISPLAY_ID,
463 0536 1 PASTEBOARD_ID,
464 0537 1 OUT_BUFFER,
465 0538 1 OUT_BUFFER_LEN : REF VECTOR [,LONG],
466 0539 1 REMAINING_COLS,
467 0540 1 DESIRED_ROW,
468 0541 1 DESIRED_COL
469 0542 1 ) =
470 0543 1

```

++
FUNCTIONAL DESCRIPTION:

Causes the set physical cursor sequence to be deposited in the specified buffer.

If the terminal we are about to do input from is one with a settable scrolling region, we make sure that the scrolling region covers the whole screen.

If the desired row and column are not specified, the current cursor position within the virtual display are assumed. REMAINING_COLS is set to the number of columns that will be visible within the interval from the indicated cursor position and the first column which is no longer within this display. This allows the caller to control inputted strings from running outside of the virtual display. If REMAINING_COLS is zero, the desired position corresponds to a spot on the screen which is currently occluded by another virtual display.

CALLING SEQUENCE:

```

ret_status.wlc.v = SMG$$SET_PHYSICAL_CURSOR (
DISPLAY_ID.rl.r,
PASTEBOARD_ID.rl.r,
OUT_BUFFER.wl.r,
OUT_BUFFER_LEN.ml.r,
REMAINING_COLS.wl.r,
[,DESIRED_ROW.rl.r],
[,DESIRED_COL.rl.r])

```

FORMAL PARAMETERS:

DISPLAY_ID.rl.r	Display id of virtual display.
PASTEBOARD_ID.rl.r	Pasteboard id.
OUT_BUFFER.rl.r	Address of a buffer in which to place the set cursor sequence. This buffer should be at least 15 bytes.
OUT_BUFFER_LEN.ml.r	Address of a word which contains the current length of OUT_BUFFER. It will be updated to reflect length of the set cursor sequence added.
REMAINING_COLS.wl.r	Returned number of visible columns


```

518 0591 1 |
519 0592 1 | available on the physical screen at
520 0593 1 | this point which are entirely within
521 0594 1 | the specified virtual display and are
522 0595 1 | not occluded by another virtual display.
523 0596 1 | [ , DESIRED_ROW.rl.r ] Desired row number within the virtual
524 0597 1 | display to which the physical cursor
525 0598 1 | should be postioned. If omitted, the
526 0599 1 | current cursor row in this virtual
527 0600 1 | display is assumed.
528 0601 1 |
529 0602 1 | [ , DESIRED_COL.rl.r ] Desired column number within the virtual
530 0603 1 | display to which the physical cursor
531 0604 1 | should be postioned. If omitted, the
532 0605 1 | current cursor column in this virtual
533 0606 1 | display is assumed.
534 0607 1 |
535 0608 1 | IMPLICIT INPUTS:
536 0609 1 |
537 0610 1 | NONE
538 0611 1 |
539 0612 1 | IMPLICIT OUTPUTS:
540 0613 1 |
541 0614 1 | NONE
542 0615 1 |
543 0616 1 | COMPLETION STATUS:
544 0617 1 |
545 0618 1 | SS$ NORMAL Normal successful completion
546 0619 1 | SMG$_INVDIS_ID Invalid Display Id
547 0620 1 | SMG$_INVPA$ ID Invalid Pasteboard Id
548 0621 1 | SMG$_INVROW Invalid row specified
549 0622 1 | SMG$_INVCOL Invalid column specified
550 0623 1 |
551 0624 1 | SIDE EFFECTS:
552 0625 1 |
553 0626 1 | NONE
554 0627 1 | --
555 0628 1 |
556 0629 2 | BEGIN
557 0630 2 | BUILTIN
558 0631 2 | NULLPARAMETER;
559 0632 2 |
560 0633 2 | LOCAL
561 0634 2 | STATUS, ! Status of subroutine calls
562 0635 2 | ROW, ! Row of interest -- in virtual display
563 0636 2 | COL, ! Column of interest -- in virtual disp.
564 0637 2 | DCB : REF $DCB_DECL, ! Addr of display control block
565 0638 2 | PBCB : REF $PBCB_DECL, ! Addr of pasteboard control
566 0639 2 | ! block.
567 0640 2 | WCB : REF $WCB_DECL, ! Addr of window control block
568 0641 2 | PP : REF $PP_DECL; ! Addr of pasting packet
569 0642 2 |
570 0643 2 |
571 0644 2 | $SMG$VALIDATE_ARGCOUNT (5, 7); ! Test for right no. of args
572 0645 2 |
573 0646 2 | $SMG$GET_DCB ( .DISPLAY_ID, DCB ); ! Get DCB addr.
574 0647 2 | $SMG$GET_PBCB ( .PASTEBOARD_ID, PBCB ); ! Get PBCB addr.

```



```

: 575      0648 2      WCB = .PBCB [PBCB_A_WCB];
: 576      0649 2
: 577      0650 2
: 578      0651 2      !+
: 579      0652 2      If we are reading from a VT100 (i.e., a device with a settable
: 580      0653 2      scrolling region), we force the physical scrolling region to be the
: 581      0654 2      whole screen. If we didn't do this, the user might be trying to input
: 582      0655 2      into the last line of a virtual display which might coincide with the
: 583      0656 2      bottom line of a scrolling region. As he types the <CR> to terminate
: 584      0657 2      his input, the virtual display will scroll out from under us because
: 585      0658 2      of the <CR><LF> echoed by the terminal driver.
: 586      0659 2      This way, the only way he can get into trouble is if he tries to
: 587      0660 2      input into the last physical line of the screen.
: 588      0661 2      !-
: 589      0662 2      !**** SHOULD CHANGE WITH TERMTABLE SUPPORT
: 590      0663 2
: 591      0664 2      IF .PBCB [PBCB_B_DEVTTYPE] EQL VT100
: 592      0665 2      THEN
: 593      0666 3          BEGIN ! Scroll region check
: 594      0667 3              IF .PBCB [PBCB_W_TOP_SCROLL_LINE] NEQ 1 OR
: 595      0668 3                  .PBCB [PBCB_W_BOT_SCROLL_LINE] NEQ .PBCB [PBCB_B_ROWS]
: 596      0669 3              THEN
: 597      0670 4                  BEGIN ! Set scroll to whole screen
: 598      0671 4                      LOCAL
: 599      0672 4                          STATUS;
: 600      0673 5                          IF NOT (STATUS = SMG$$FORCE_SCROLL_REG (
: 601      0674 5                              .PBCB,
: 602      0675 5                              1,
: 603      0676 5                              .PBCB [PBCB_B_ROWS]))
: 604      0677 4                      THEN
: 605      0678 4                          RETURN .STATUS;
: 606      0679 4                      !+
: 607      0680 4                          Flush out this transaction separately. We don't want it
: 608      0681 4                          to become part of the cursor setting sequence which our
: 609      0682 4                          caller may want to output a second time.
: 610      0683 4                      !-
: 611      0684 4                          IF .PBCB [PBCB_V_BUF_ENABLED]
: 612      0685 4                          THEN
: 613      0686 4                              SMG$$FLUSH_BUFFER ( .PBCB);
: 614      0687 3                          END; ! Set scroll to whole screen
: 615      0688 2                      END; ! Scroll region check
: 616      0689 2
: 617      0690 2      !+
: 618      0691 2      If this display is not pasted to this pasteboard, quit -- but flush
: 619      0692 2      the buffer on the way out anyway.
: 620      0693 2      !-
: 621      0694 3      IF NOT (STATUS = SMG$$LOCATE_PP (.DCB, .PBCB, PP)) ! Get PP addr.
: 622      0695 3      THEN
: 623      0696 3          BEGIN
: 624      0697 3              IF .PBCB [PBCB_V_BUF_ENABLED]
: 625      0698 3              THEN
: 626      0699 3                  SMG$$FLUSH_BUFFER ( .PBCB);
: 627      0700 3              RETURN (.STATUS);
: 628      0701 3          END;
: 629      0702 3
: 630      0703 3      ROW = (IF NOT NULLPARAMETER (DESIRED_ROW) THEN ..DESIRED_ROW
: 631      0704 2                      ELSE .DCB [DCB_W_CURSOR_ROW]);

```



```

632      0705 2      COL = (IF NOT NULLPARAMETER (DESIRED_COL) THEN .DESIRED_COL
633      0706 3      ELSE .DCB [DCB_W_CURSOR_COL]);
634      0707 3
635      0708 3
636      0709 3      $SMG$VALIDATE_ROW_COL ( .ROW, .COL);
637      0710 3
638      0711 3      !+
639      0712 3      !- Perform remaining column calculation.
640      0713 3
641      0714 3      IF .PP [PP_V_OCCLUDED]
642      0715 3      THEN
643      0716 3          BEGIN ! Remaining number of col. calculation. -occluded case.
644      0717 3              LOCAL
645      0718 3                  TEMP : BLOCK [8,BYTE], ! Temporary representation of
646      0719 3                      ! area of input line in pasteboard
647      0720 3                      ! coordinates.
648      0721 3                  CURR_PP : REF $PP_DECL; ! Addr of a pasting packet
649      0722 3
650      0723 3      !+ Identify a rectangular area that consists of the part of the
651      0724 3      ! line bounded by ROW and COL and the right end of the same
652      0725 3      ! line. This represents the maximum allowable input area.
653      0726 3      !-
654      0727 3      TEMP [DCB_W_ROW_START] = .ROW + .PP [PP_W_ROW] - 1;
655      0728 3      TEMP [DCB_W_NO_ROWS] = 1;
656      0729 3      TEMP [DCB_W_COL_START] = .COL + .PP [PP_W_COL] - 1;
657      0730 3      TEMP [DCB_W_NO_COLS] = .PP [PP_W_MOVE_LENGTH] -
658      0731 3          ( .COL + .PP [PP_W_COL] ) + 2;
659      0732 3
660      0733 3      !+
661      0734 3      ! Check the rectangle isolated above against the projections of
662      0735 3      ! all virtual displays on this same pasteboard which may occlude
663      0736 3      ! parts or all of this input rectangle. The most restricted
664      0737 3      ! length of the input line is what we return to our caller.
665      0738 3      ! We start with the one pasted next in the chain (if any).
666      0739 3      !-
667      0740 3      CURR_PP = .PP [PP_A_PREV_PBCB];
668      0741 3      WHILE .CURR_PP NEQ PBCB [PBCB_A_PP_NEXT] ! While more displays
669      0742 3      DO ! remain...
670      0743 4          BEGIN ! Overall loop
671      0744 4              LOCAL
672      0745 4                  PP_BASE : REF $PP_DECL; ! Base address of current
673      0746 4                      ! pasting packet
674      0747 4
675      0748 4          PP_BASE = .CURR_PP - PP_PBCB_QUEUE_OFFSET;
676      0749 4          ! Since the queue headers for this part
677      0750 4          ! of the chain are not at relative 0 in
678      0751 4          ! the pasting packet.
679      0752 4
680      0753 4          IF .PP_BASE [PP_W_MOVE_LENGTH] NEQ 0
681      0754 4          THEN
682      0755 5              BEGIN ! Display visible
683      0756 5                  LOCAL
684      0757 5                      OVERLAP : BLOCK [8,BYTE], ! Representation of
685      0758 5                          ! overlap between input
686      0759 5                          ! line and a
687      0760 5                          ! higher-pasted virtual
688      0761 5                          ! display.

```



```

689      0762 5      TEMP1 : BLOCK [8,BYTE], ! Representation of
690      0763 5      ! current virtual display
691      0764 5      ! in pasteboard
692      0765 5      ! coordinates.
693      0766 5      NEW_DCB : REF $DCB_DECL; ! Addr of display control
694      0767 5      ! lock currently involved.
695      0768 5
696      0769 5      NEW_DCB = .PP_BASE [PP_A_DCB_ADDR];
697      0770 5
698      0771 5      TEMP1 [DCB_W_ROW_START] = .PP_BASE [PP_W_ROW];
699      0772 5      TEMP1 [DCB_W_NO_ROWS] = .NEW_DCB [DCB_W_NO_ROWS];
700      0773 5      TEMP1 [DCB_W_COL_START] = .PP_BASE [PP_W_COL];
701      0774 5      TEMP1 [DCB_W_NO_COLS] = .NEW_DCB [DCB_W_NO_COLS];
702      0775 5
703      0776 5      !+
704      0777 5      ! If the virtual display we're looking at is bordered,
705      0778 5      ! it will have a bigger footprint in the pasteboard
706      0779 5      ! buffer than its dimensions alone. Adjust for the
707      0780 5      ! increased size of the foot print.
708      0781 5      !-
709      0782 5      IF .NEW_DCB [DCB_V_BORDERED]
710      0783 5      THEN
711      0784 6      BEGIN
712      0785 6      TEMP1 [DCB_W_ROW_START] = .TEMP1 [DCB_W_ROW_START] - 1;
713      0786 6      TEMP1 [DCB_W_NO_ROWS] = .TEMP1 [DCB_W_NO_ROWS] + 2;
714      0787 6      TEMP1 [DCB_W_COL_START] = .TEMP1 [DCB_W_COL_START] - 1;
715      0788 6      TEMP1 [DCB_W_NO_COLS] = .TEMP1 [DCB_W_NO_COLS] + 2;
716      0789 5      END;
717      0790 5
718      0791 5      IF SMG$$OCCLUDE ( TEMP, ! The input line
719      0792 5      TEMP1, ! The higher-pasted display
720      0793 5      OVERLAP) ! Overlapping region (if any)
721      0794 5      THEN
722      0795 6      BEGIN ! Overlap
723      0796 6      IF .TEMP [DCB_W_COL_START] GEQ
724      0797 6      .OVERLAP [DCB_W_COL_START] AND
725      0798 6      .TEMP [DCB_W_COL_START] LEQ
726      0799 6      .OVERLAP [DCB_W_COL_START] + .OVERLAP [DCB_W_NO_COLS] - 1
727      0800 6      THEN
728      0801 7      BEGIN ! Requested start pos. occluded
729      0802 7      .REMAINING_COLS = 0;
730      0803 7      IF .PBCB [PBCB_V_BUF_ENABLED]
731      0804 7      THEN
732      0805 7      SMG$$FLUSH_BUFFER ( .PBCB);
733      0806 7      RETURN (SS$_NORMAL); ! *** Should this be a
734      0807 7      ! failure status ??? ***
735      0808 7      END ! Requested start pos. occluded
736      0809 6      ELSE
737      0810 7      BEGIN ! Tail end of input pos occluded
738      0811 7      !+
739      0812 7      ! Truncate length of input line down to the part
740      0813 7      ! that is not occluded.
741      0814 7      !-
742      0815 7      TEMP [DCB_W_NO_COLS] = .TEMP1 [DCB_W_COL_START]
743      0816 7      - .TEMP [DCB_W_COL_START];
744      0817 7      IF .TEMP [DCB_W_NO_COLS] LEQ 0
745      0818 7      THEN

```



```

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

```



```

: 803      0876 2      CH$MOVE(.PBCB[PBCB_L_CAP_LENGTH],.PBCB[PBCB_A_CAP_BUFFER],.OUT_BUFFER);
: 804      0877 2      OUT_BUFFER_LEN [0] = .PBCB[PBCB_L_CAP_LENGTH];
: 805      0878 2
: 806      0879 2      RETURN (SS$_NORMAL);
: 807      0880 1      END;

```

! End of routine SMG\$\$SET_PHYSICAL_CURSOR

				07FC 00000	.EXTRN SMG\$GET_TERM_DATA	
					.ENTRY SMG\$\$SET_PHYSICAL_CURSOR, Save R2,R3,R4,R5,-;	0535
					R6,R7,R8,R9,R10	
					MOVAB SMG\$\$FLUSH_BUFFER, R10	
50					SUBL2 #32, SP	
					SUBB3 #5, (AP), DIFF	0644
					CMPB DIFF, #2	
					BLEQU 1\$	
					MOVL #SMG\$_WRONUMARG, R0	
					RET	
					MOVL @DISPLAY_ID, R0	0646
04					CMPL 56(R0), @DISPLAY_ID	
					BNEQ 2\$	
					CMPB 68(R0), #17	
					BEQL 3\$	
					MOVL #SMG\$_INVDIS_ID, R0	
					RET	
					MOVL @DISPLAY_ID, DCB	
					MOVL @PASTEBOARD_ID, R0	0647
					BLSS 4\$	
					CMPL R0, PBD_L_COUNT	
					BGTR 4\$	
					BBS R0, PBD_V_PB_AVAIL, 5\$	
08					MOVL #SMG\$_INVPAS_ID, R0	
					RET	
					MOVL PBD_A_PBCB[R0], PBCB	
					MOVL 8(PBCB), WCB	0648
					CMPB 16(PBCB), #3	0664
					BNEQ 8\$	
					CMPW 244(PBCB), #1	0667
					BNEQ 6\$	
					MOVZBL 95(PBCB), R0	0668
					CMPW R0, 246(PBCB)	
					BEQL 8\$	
					MOVZBL 95(PBCB), -(SP)	0676
					PUSHL #1	0673
					PUSHL PBCB	0674
					CALLS #3, SMG\$\$FORCE_SCROLL_REG	
					BLBS STATUS, 7\$	0673
					RET	
					BLBC 12(PBCB), 8\$	0684
					PUSHL PBCB	0686
					CALLS #1, SMG\$\$FLUSH_BUFFER	
					PUSHAB PP	0694
					PUSHL PBCB	
					PUSHL DCB	
					CALLS #3, SMG\$\$LOCATE_PP	
					MOVL R0, STATUS	

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

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

SMG\$\$DISPLAY_IN 1-026 SMG\$\$DISPLAY INPUT - Input support routines 16-Sep-1984 00:27:47 H 11
 SMG\$\$SET_PHYSICAL_CURSOR - Set physical cursor 14-Sep-1984 13:09:42 VAX-11 Bliss-32 V4.0-742
 [SMGRTL.SRC]SMGDISINP.B32;1

Page 22
 (5)

00000000G	00	00FC	C4	9F	0023D	PUSHAB	252(PBCB)
	0E		06	FB	00241	CALLS	#6, SMG\$GET_TERM_DATA
OC BC	D4		50	E9	00248	BLBC	STATUS, 33\$
10	BC		66	28	0024B	MOV C3	(R6), @260(PBCB), @OUT_BUFFER
	50		66	D0	00252	MOVL	(R6), @OUT_BUFFER_LEN
			01	D0	00256	MOVL	#1, R0
			04	00259	33\$:	RET	

:
 :
 : 0876
 : 0877
 : 0879
 : 0880

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

; 808 0881 1 !<BLF/PAGE>


```

: 810      0882 1 END
: 811      0883 1
: 812      0884 0 ELUDOM

```

! End of module SMG\$\$DISPLAY_INPUT

PSECT SUMMARY		
Name	Bytes	Attributes
_SMG\$CODE	879	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics					
File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	4	0	581	00:01.0
-\$255\$DUA28:[SMGRTL.OBJ]RTL.LIB.L32;1	36	0	0	8	00:00.1
-\$255\$DUA28:[SMGRTL.OBJ]SMGLIB.L32;1	469	61	13	38	00:00.4

```

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

```

```

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

```


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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY