C 12

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

```
_$2

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

```
**FILE**ID**SMGDATSTR                                                  B 2
```

```
  SSSSSSSS  MM      MM    GGGGGGGG  DDDDDDDD      AAAAAA    TTTTTTTTTT    SSSSSSSS  TTTTTTTTTT  RRRRRRRR
  SSSSSSSS  MM      MM    GGGGGGGG  DDDDDDDD      AAAAAA    TTTTTTTTTT    SSSSSSSS  TTTTTTTTTT  RRRRRRRR
SS          MMMM  MMMM  GG          DD      DD  AA      AA      TT      SS              TT      RR      RR
SS          MMMM  MMMM  GG          DD      DD  AA      AA      TT      SS              TT      RR      RR
SS          MM  MM  MM  GG          DD      DD  AA      AA      TT      SS              TT      RR      RR
SS          MM  MM  MM  GG          DD      DD  AA      AA      TT      SS              TT      RR      RR
  SSSSSS    MM      MM  GG          DD      DD  AA      AA      TT        SSSSSS        TT      RRRRRRRR
  SSSSSS    MM      MM  GG          DD      DD  AA      AA      TT        SSSSSS        TT      RRRRRRRR
        SS  MM      MM  GG  GGGGGG  DD      DD  AAAAAAAAAA      TT              SS      TT      RR  RR
        SS  MM      MM  GG  GGGGGG  DD      DD  AAAAAAAAAA      TT              SS      TT      RR  RR
        SS  MM      MM  GG      GG  DD      DD  AA      AA      TT              SS      TT      RR    RR         ....
        SS  MM      MM  GG      GG  DD      DD  AA      AA      TT              SS      TT      RR    RR         ....
SSSSSSSS    MM      MM    GGGGGG    DDDDDDDD    AA      AA      TT      SSSSSSSS        TT      RR      RR       ....
SSSSSSSS    MM      MM    GGGGGG    DDDDDDDD    AA      AA      TT      SSSSSSSS        TT      RR      RR       ....

RRRRRRRR    EEEEEEEEEE    QQQQQQ
RRRRRRRR    EEEEEEEEEE    QQQQQQ
RR      RR  EE          QQ      QQ
RR      RR  EE          QQ      QQ
RR      RR  EE          QQ      QQ
RR      RR  EE          QQ      QQ
RRRRRRRR    EEEEEEEE    QQ      QQ
RRRRRRRR    EEEEEEEE    QQ      QQ
RR  RR      EE          QQ  QQ  QQ
RR  RR      EE          QQ  QQ  QQ
RR    RR    EE          QQ    QQ
RR    RR    EE          QQ    QQ
RR      RR  EEEEEEEEEE    QQQQ  QQ
RR      RR  EEEEEEEEEE    QQQQ  QQ
```

```
! Data Structure Definitions for RTL SMG$ facility
! File:  SMGDATSTR.REQ  Edit: STAN1054
!
!****************************************************************************
!*                                                                          *
!*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
!*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
!*  ALL RIGHTS RESERVED.                                                    *
!*                                                                          *
!*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
!*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
!*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
!*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
!*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
!*  TRANSFERRED.                                                            *
!*                                                                          *
!*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
!*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
!*  CORPORATION.                                                            *
!*                                                                          *
!*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
!*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
!*                                                                          *
!*                                                                          *
!****************************************************************************
!
!
!++
! FACILITY:     Screen Management
!
! ABSTRACT:
!
!       This file contains data structure definitions for screen management
!       routines.  Display Control Block, Pasteboard Control Block, Window
!       Control Block, and Pasting Packet are defined here.
!
!
! MODIFIED BY:
!
! 1-001 - Original.  PLL 15-Mar-1983
! 1-002 - Expand PBCB to hold PBCB_L_MODE_SETTINGS.  RKR 17-Mar-1983.
! 1-003 - Add literal for initial setting of PBCB_L_MODE_SETTINGS.
!         RKR 18-Mar-1983.
! 1-004 - Add stuff for borders, alternate character sets, etc.
!         RKR 24-Mar-1983.
! 1-005 - Corrections to last edit.  RKR 24-Mar-1983.
! 1-006 - Expand WCB and PP structures.  RKR 28-Mar-1983.
! 1-007 - Fix typo in last edit.  RKR 28-Mar-1983.
! 1-008 - Pull BACKGROUND_COLOR out of PBCB since we don't know what
!         to do with it.  Add stuff pertaining to borders.
!         RKR 4-April-1983.
! 1-009 - More fixes.  RKR 4-April-1983.
! 1-010 - Clean up rest of border masks.  RKR 5-April-1983
! 1-011 - More additions for labeled borders.  RKR 7-April-1983.
! 1-012 - Add more fields to PBCB and PP.  RKR 14-APR-1983.
! 1-013 - Rearrange the bits for the line-drawing character set and the
!         bit used to designate a border element. RKR 15-APR-1983.
! 1-014 - Define 2 more bits in the DCB, one to mean that all lines are
!         full (and thus scrolling should occur thereafter), and one to
!         mean that column 80 was just written (useful on the next write
!         operation).  PLL 28-Apr-1983
! 1-015 - Define a bell bit in the DCB and WCB.  PLL 29-Apr-1983
! 1-016 - New fields in PBCB. STAN 28-Apr-1983 (2nd try).
!         Changed PID to PBID so as not to confuse a pasteboard id
!         with a process id.
! 1-017 - Added event flag numbers to PBCB. STAN 30-Apr-1983
! 1-018 - STAN 1-May-1983
!         Added SMG$K_LONGEST_SEQUENCE.
!         Added output buffer fields to PBCB.
!         Added rows and columns to PBCB.
! 1-019 - STAN 3-May-1983
!         Allow up to 16 pasteboards.
!         Put out-of-band AST routine into PBCB. Literally!
! 1-020 - STAN 4-May-1983
!         Fix typo in comment.
! 1-021 - Add PP_B_CONTROL_BITS field in PP.  RKR 5-MAY-1983
! 1-022 - Add PP_V_CONTIG bit in PP_B_CONTROL_BITS.  RKR 9-MAY-1983
! 1-023 - Add to DCB, PP, and WCB a longword that is the product of
!         the number of rows * the number of columns.  (Not strictly
!         true for PP case -- but ignore it.
!         RKR 13-MAY-1983.
! 1-024 - Add BATCHING bit to PBCB
!         STAN 16-May-1983
! 1-025 - Remove bell bit.  PLL 20-May-1983
! 1-026 - Extend DCB in anticpation of backup DCB logic.
!         Delete DD_K_MAX_VD -- should no longer be needed.
!         RKR 20-MAY-1983.
! 1-027 - Add words in the DCB to store the top and bottom of a scrolling
!         region.  PLL 25-May-1983
! 1-028 - Add new fields to PP and PBCB.  RKR 26-MAY-1983.
! 1-029 - New fields to PBCB for mailboxes et. al. STAN 1-Jun-1983.
! 1-030 - More of same. STAN  2-Jun-1983.
! 1-031 - More of same. STAN 13-Jun-1983.
! 1-032 - Add DCB_V_LABEL_CENTER control bit to DCB_L_CONTROL_BITS.
!         RKR 14-JUN-1983.
! 1-033 - Make PP_W_ROW and PP_W_COL signed fields.
!         RKR 14-JUN-1983
! 1-034 - Add bit DCB_V_PP_MISMATCH.  RKR 17-JUN-1983.
! 1-035 - Fields for output files in PBCB. STAN 18-Jun-1983.
!         Made LAST_CHANGED fields in PBCB signed.
!         Created macros for declaring structures.
!         Made CURSOR position in PBCB signed.
! 1-036 - Add structures to DCB and WCB dealing with line characteristics
!         like Double-Wide, Double-High, etc.
!         RKR 7-JUL-1983.
! 1-037 - Add bit to DCB to mark as used for autobending.  PLL 7-Jul-1983
! 1-038 - Add more longwords to DCB for autobending (used to parse
!         escape sequences).  PLL 8-Jul-1983
! 1-039 - Add 2 words to PBCB to record where the physical scrolling
!         region is on the terminal.
```

SMG

LIT

MAC

```
!          RKR 11-JUL-1983.
! 1-040 - Fix typo.  RKR 11-JUL-1983.
! 1-041 - Save original terminal width and height. STAN 22-Aug-1983
! 1-042 - STAN 31-Aug-1983 Line characteristics types.
! 1-043 - Add a truncation icon attribute bit to the DCB. PLL 1-Sep-1983
! 1-044 - Get rid of 2 unused fields in the DCB by renaming one to
!         a simulated device type and leaving the other one as a placeholder.
!         PLL 2-Sep-1983
! 1-045 - Added some terminal characteristics constants.
!         STAN 5-Sep-1983.
! 1-046 - Use up DCB_B_FILL by turning it into DCB_B_LABEL_REND.
!         RKR 15-SEP-1983.
! 1-047 - Add a user line drawing bit the rendition attribute. PLL 21-Sep-1983
! 1-048 - Background color byte. STAN 27-Sep-1983.
! 1-049 - STAN 14-Oct-1983. Added wide and high bits; AST-reentrancy bits.
! 1-050 - STAN 14-Oct-1983. CTRL/O bit.
! 1-051 - STAN 17-Oct-1983. Add Cancel control/O bit.
! 1-052 - STAN 15-Jan-1984. Add TERMTABLE.
! 1-053 - STAN 21-Feb-1984. Add LF, TAB, and BS optimization bits in PBD.
! 1-054 - STAN 6-Mar-1983. Add NOTABS bit.
!--
```

```
SMG

!+
! P

! t
! e
! s
! t
! p
! o
! q
! p
!-
MAC
```

```
!+
! Critical sizes and counts for virtual displays and pasteboards
!-

LITERAL

    PBD_K_MAX_PB = 16,      ! Maximum number of pasteboards we can track.
                            ! It controls the range of pasteboard id's we
                            ! will allocate and the size of the pasteboard
                            ! directory (PBD) structure in OWN storage.
                            ! Currently constrained not to exceeed 32 by
                            ! usage of FFC instruction in $GET_NEXT_PID.

    SMG$K_LONGEST_SEQUENCE = 255;

                            ! Longest control or escape sequence that
                            ! can be returned by TERMTABLE routines.
                            ! This value can be used to preallocate a
                            ! buffer to hold the text or can be used to tell
                            ! if the next sequence desired could overflow
                            ! your buffer.
```

```
!+
! Virtual Display Control Block (DCB)
! -----------------------------------
!       This data structure defines the layout of a Virtual Display
! Control Block.  The area is allocated in heap storage.  One such
! block is allocated for each new virtual display created by callers.
! It contains dimensions of the virtual display and pointers to other
! buffers associated with this display.  It also contains pointers to
! the pasteboards onto which it is pasted.  This area is deallocated
! when the virtual display is deleted -- not when it is unpasted.
!-
MACRO

    DCB_Q_COORD         =  0, 0, 00, 0%,   ! Really 0, 0, 64, 0
                ! Quadword containing next four words.  These 4 fields
                ! define the coordinate system for this virtual display
                ! and their address is transmitted to pass the 4 fields
                ! as a single parameter

        DCB_W_ROW_START =  0, 0, 16, 0%,   ! Row number of 1st row. (=1)
        DCB_W_NO_ROWS   =  2, 0, 16, 0%,   ! Number of rows
        DCB_W_COL_START =  4, 0, 16, 0%,   ! Col number of 1st col (=1)
        DCB_W_NO_COLS   =  6, 0, 16, 0%,   ! Number of columns

    DCB_Q_LABEL_DESC    =  8, 0, 00, 0%,    ! Really 8, 0, 64, 0
                                            ! Dynamic string descriptor
                                            ! for border label text

    DCB_A_TEXT_BUF      = 16, 0, 32, 0%,    ! Addr. of buffer containing
                                            ! text for this virtual
                                            ! display.

    DCB_A_ATTR_BUF      = 20, 0, 32, 0%,    ! Addr. of buffer containing
                                            ! video attributes for each
                                            ! character position in
                                            ! TEXT_BUF.

    DCB_A_CHAR_SET_BUF  = 24, 0, 32, 0%,    ! Addr. of buffer containing
                                            ! character set codes for
                                            ! each character in TEXT_BUF.
                                            ! This buffer allocated only
                                            ! when needed.

    DCB_L_BATCH_LEVEL   = 28, 0, 32, 0%,    ! Number of levels of
                                            ! batching in effect for this
                                            ! display.
                                            ! Incremented by call to
                                            ! SMG$START_DISPLAY_UPDATE
                                            ! and decrement toward zero
                                            ! by each call to
                                            ! SMG$END_DISPLAY_UPDATE.
                                            ! Output flows from this
                                            ! virtual display to the
                                            ! screen only when this
                                            ! variable is zero.
    DCB_A_PP_NEXT       = 32, 0, 32, 0%,
    DCB_A_PP_PREV       = 36, 0, 32, 0%,
                                            ! Above two longwords are the
                                            ! queue header for the chain
                                            ! of Pasting Packets tied to
                                            ! this virtual display.

    DCB_W_CURSOR_ROW    = 40, 0, 16, 0%,    ! Cursor row position in this
                                            ! virtual display

    DCB_W_CURSOR_COL    = 42, 0, 16, 0%,    ! Cursor col position in this
                                            ! virtual display

    DCB_W_LABEL_UNITS   = 44, 0, 16, 0%,    ! Starting position in the
                                            ! line or column indicated by
                                            ! DCB_B_LABEL_POS

    DCB_B_DEF_VIDEO_ATTR= 46, 0,  8, 0%,    ! Default video attributes of
                                            ! this virtual display
        DCB_V_RENBOL    = 46, 0,  1, 0%,    ! Bold
        DCB_V_RENREV    = 46, 1,  1, 0%,    ! Reverse video
        DCB_V_RENBLK    = 46, 2,  1, 0%,    ! Blink
        DCB_V_RENUND    = 46, 3,  1, 0%,    ! Underline

    DCB_B_DEF_DISPLAY_ATTR=47,0,  8, 0%,    ! Default display attributes
                                            ! of this virtual display
        DCB_V_BORDERED  = 47, 0,  1, 0%,    ! Bordered

        DCB_V_TRUNC_ICON = 47,1,  1, 0%,    ! Flag to use truncation icon

        DCB_V_DISPLAY_CONTROLS =            ! Flag to display carriage control
                          47, 2,  1, 0%,    ! characters such a <CR> instead
                                            ! of execute them

    DCB_B_DEF_CHAR_SET  = 48, 0,  8, 0%,    ! Default character set for
                                            ! all text in this virtual
                                            ! display.

    DCB_B_LABEL_POS     = 49, 0,  8, 0%,    ! Code for positioning of
                                            ! border label:
                                            !  0 = Top border line
                                            !  1 = Bottom border line
                                            !  2 = Left border line
                                            !  3 = Right border line

    DCB_B_LABEL_CHAR_SET= 50, 0,  8, 0%,    ! Code for character set of
                                            ! border label.

    DCB_B_LABEL_REND    = 51, 0,  8, 0%,    ! Rendition for border label

    DCB_L_CONTROL_BITS  = 52, 0, 32, 0%,    ! Control bits

        DCB_V_FULL      = 52, 0,  1, 0%,    ! All display lines used
                                            ! (next op may scroll)

        DCB_V_COL_80    = 52, 1,  1, 0%,    ! Column 80 just written
        DCB_V_LABEL_CENTER=52,2,  1, 0%,      ! If set indicates that
                                              ! border label should be
                                              ! centered -- even if virt.
                                              ! display is redimensioned.

        DCB_V_PP_MISMATCH=52, 3,  1, 0%,      ! If this bit is set it
                                              ! indicates that this virtual
                                              ! display control block
                                              ! changed in such a way that
                                              ! all associated pasting
                                              ! packets need to have their
                                              ! constants recalculated.
                                              ! However, this change occurred
                                              ! while the display was
                                              ! "batched" and could not be
                                              ! done at that time.
                                              ! SMG$END_DISPLAY_UPDATE
                                              ! senses this bit whenever it
                                              ! makes the transition to
                                              ! batch_level=0 and performs
                                              ! the psting packet recalc.
                                              ! at that time, then resets
                                              ! this bit.

        DCB_V_AUTOBENDED =52, 4,  1, 0%,      ! This DCB created by
                                              ! autobended routines

        DCB_V_ALLOW_ESC = 52, 5,  1, 0%,      ! Parse escape sequences when set

        DCB_V_LOCKED    = 52, 6,  1, 0%,      ! DCB is locked for our use

    DCB_L_DID           = 56, 0, 32, 0%,      ! Virtual display id
                                              ! (Currently the address of
                                              !  the DCB itself.)

    DCB_L_BUFSIZE       = 60, 0, 32, 0%,      ! = .DCB [DCB_W_NO_ROWS] *
                                              !    .DCB [DCB_W_NO_COLS]


    DCB_A_BACKUP_DCB    = 64, 0, 32, 0%,      ! If non-zero, address of the
                                              ! backup DCB when this DCB is
                                              ! batched.  Backup DCB holds
                                              ! the state of the DCB at the
                                              ! time batching started.

    DCB_B_STRUCT_TYPE   = 68, 0,  8, 0%,      ! Code to mark this structure
                                              ! as being a DCB

    DCB_W_DCB_LENGTH    = 69, 0, 16, 0%,      ! Stored length of a DCB

    DCB_B_FILL_2        = 71, 0,  8, 0%,      !

    DCB_W_TOP_OF_SCRREG = 72, 0, 16, 0%,      ! Top line in scrolling region

    DCB_W_BOTTOM_OF_SCRREG
```

```
                        = 74, 0, 16, 0%,    ! Bottom line in scrolling region

    DCB_A_LINE_CHAR     = 76, 0, 32, 0%,    ! Address of the line
                                            ! characteristics vector.
                                            ! This vector, one byte for
                                            ! each line, records whether
                                            ! the line is Single, Double-
                                            ! High, Double-Wide, etc.
                                            ! This vector is allocated to
                                            ! be DCB_W_NO_ROWS + 1 bytes
                                            ! long so it can be indexed
                                            ! directly by row number
                                            ! (1 through DCB_W_NO_ROWS).
    DCB_SIM_CONTROL     = 80, 0, 32, 0%,    ! Control bits for SMG$$SIM_TERM
    DCB_ARG_1           = 84, 0, 32, 0%,    ! Control sec arg 1
    DCB_ARG_2           = 88, 0, 32, 0%,    ! Control seq arg 2
    DCB_SIM_DEV_TYPE    = 92, 0, 32, 0%,    ! Device type to simulate
    DCB_UNUSED          = 96, 0, 32, 0%,    ! Unused
    DCB_SAVED_HPOS      = 100, 0, 32, 0%,   ! Saved cursor column
    DCB_SAVED_VERT      = 104, 0, 32, 0%,   ! Saved cursor row
    DCB_SAVED_VIDEO_ATTR = 108, 0, 32, 0%;  ! Saved video attributes

LITERAL

    DCB_K_STRUCT_TYPE = %X'11',  ! Code stored in DCB [DCB_B_STRUCT_TYPE]
                                 ! to mark is as being a DCB.

    DCB_K_SIZE  = 112;           ! Total number of bytes in a DCB

MACRO

    $DCB_DECL   = BLOCK[DCB_K_SIZE,BYTE] %;
```

```
!+
! Pasteboard Control Block (PBCB)
! -------------------------------
!       This data structure resides in HEAP storage.  One of these areas
! is allocated whenever a new stream is established for the first time.
! It is deallocated when the pasteboard is deleted.
! It contains the fundemental information associated with a pasteboard
! and pointers to PBCB-related structures like the WCB.
!-

MACRO

    PBCB_A_PP_NEXT          =   0, 0, 32, 0%,
    PBCB_A_PP_PREV          =   4, 0, 32, 0%,
                                    ! Previous two longwords serve
                                    ! as a queue header for the
                                    ! chain of pasting packets of
                                    ! all the virtual displays that
                                    ! are pasted to this pasteboard.

    PBCB_A_WCB              =   8, 0, 32, 0%,
                                    ! Addr. of window control block
                                    ! (WCB)

    PBCB_L_MODE_SETTINGS    =  12, 0, 32, 0%,     ! Mode setting for this PBCB
        PBCB_V_BUF_ENABLED  =  12, 0,  1, 0%,     ! =1 if buffering enabled
        PBCB_V_MINUPD       =  12, 1,  1, 0%,     ! =1 if minimal update enabled
        PBCB_V_CLEAR_SCREEN=   12, 2,  1, 0%,     ! =1 if should clear screen on exit
        PBCB_V_NOTABS       =  12, 3,  1, 0%,     ! =1 if SMG should not use physical
                                                  !      tabs.

    PBCB_B_DEVTYPE          =  16, 0,  8, 0%,     ! Logical device type
                                                  ! Status are defined in
                                                  ! SMGTERM.REQ and
                                                  ! currently are:
                                                  !   UNKNOWN   = 0
                                                  !   VT05      = 1
                                                  !   VT52      = 2
                                                  !   VT100     = 3
                                                  !   VTFOREIGN = 4
                                                  !   HARDCOPY  = 5

    PBCB_B_PARITY           =  17, 0,  8, 0%,     ! parity flags

    PBCB_W_DEVNAM_LEN       =  18, 0, 16, 0%,     ! Length of the
                                                  ! resultant device name
                                                  ! string contained in
                                                  ! PBCB_T_DEVNAM.

    PBCB_L_PBID             =  20, 0, 32, 0%,     ! Pasteboard id

    PBCB_T_DEVNAM           =  24, 0,  0, 0%,     ! A 64-byte area.  This
                                                  ! buffer contains the
                                                  ! resultant device name
                                                  ! string.  Its length
                                                  ! is contained in
```

```
                                             ! PBCB_W_DEVNAM_LEN.
    PBCB_R_CHARBUF          =  88, 0,  0, 0%,    ! Start of 12-byte
                                                 ! characteristics buffer

      PBCB_L_DEVCHAR        =  88, 0, 32, 0%,    ! Device characteristics

        PBCB_B_CLASS        =  88, 0,  8, 0%,    ! Device class, e.g. DC$_TERM

        PBCB_B_PHY_DEV_TYPE=  89, 0,  8, 0%,     ! Physical device type,
                                                 ! e.g. DT$_VT100

        PBCB_W_WIDTH        =  90, 0, 16, 0%,    ! Device width

      PBCB_L_DEVDEPEND      =  92, 0, 32, 0%,    ! Primary device dependent
                                                 ! bits.  These are the bits
                                                 ! of the TT$V_xyz flavor.

        PBCB_B_ROWS         =  92, 24, 8, 0%,    ! Number of rows on terminal
                                                 ! (overlaps previous field)

      PBCB_L_DEVDEPEND2     =  96, 0, 32, 0%,    ! Secondary device
                                                 ! dependent bits.  These
                                                 ! are the bits of the
                                                 ! TT2$V_xyz flavor.

    PBCB_W_CHAN             = 100, 0, 16, 0%,    ! Channel number. 0 means
                                                 ! no channel as been assigned
                                                 ! yet.

    PBCB_B_EFN              = 102, 0,  8, 0%,    ! Primary output event flag

    PBCB_B_ASYNC_EFN        = 103, 0,  8, 0%,    ! Secondary output event flag
                                                 ! used for asynchronous operations

    PBCB_A_MBX_MSG_LIST     = 104, 0, 32, 0%,    ! List of messages that came
                                                 ! from our associated mailbox

    PBCB_A_OUTPUT_BUFFER    = 108, 0, 32, 0%,    ! Address of buffer used to
                                                 ! buffer up output sequences.

    PBCB_W_OUTPUT_BUFSIZ    = 112, 0, 16, 0%,    ! (Maximum) size of output buffer

    PBCB_W_OUTPUT_BUFLEN    = 114, 0, 16, 0%,    ! Current length of output buffer
                                                 ! i.e. number of characters in
                                                 ! the buffer. 0 means the
                                                 ! buffer is empty.

    PBCB_R_EXIT_BLOCK       = 116, 0,  0 ,0%,    ! Exit block (5 longwords)

        PBCB_L_EXIT_LINK    = 116, 0, 32, 0%,    ! system forward link to next block
        PBCB_A_EXIT_ADDR    = 120, 0, 32, 0%,    ! address of our exit handler
        PBCB_B_EXIT_ARGCNT  = 124, 0,  8, 0%,    ! argument count (=2)
        PBCB_A_EXIT_RSN     = 128, 0, 32, 0%,    ! arg 1: address to store exit reason
        PBCB_A_EXIT_PBCB    = 132, 0, 32, 0%,    ! arg 2: our PBCB address
    PBCB_L_EXIT_REASON      = 136, 0, 32, 0%,      ! exit reason (address stored
                                                   ! as first argument in exit
                                                   ! block).

    PBCB_Z_OUT_OF_BAND_RTN = 140, 0,  0, 0%,       ! ten-byte routine resides here.
        PBCB_W_ENTRY_MASK  = 140, 0, 16, 0%,       ! 0000 You may not believe it,
        PBCB_B_CALLG       = 142, 0,  8, 0%,       !   FA but 'tis so.  The first
        PBCB_B_REG_AP      = 143, 0,  8, 0%,       !   6C word is the entry mask.
        PBCB_B_ABS         = 144, 0,  8, 0%,       !   9F
        PBCB_A_BAND_HANDLER= 145, 0, 32, 0%,       !      Address of generic
        PBCB_B_RET         = 149, 0,  8, 0%,       !   04 out-of-band AST handler

    PBCB_A_BAND_ROUTINE     = 152, 0, 32, 0%,      ! Address of user's AST routine
                                                   ! for out-of-band characters.
                                                   ! 0 means out-of-band ASTs
                                                   ! are not enabled.

    PBCB_L_BAND_AST_ARG     = 156, 0, 32, 0%,      ! User's arg to his AST routine

    PBCB_M_BAND_MASK        = 160, 0, 32, 0%,      ! Character mask for out-of-
                                                   ! band ASTs currently in effect
    PBCB_L_BATCH_LEVEL      = 164, 0, 32, 0%,      ! Batching level.  If non-0,
                                                   ! then batching is in effect.

                                                   ! Next 4 fields are set
                                                   ! during mapping from
                                                   ! virtual display to
                                                   ! pasteboard buffers
                                                   ! and describe what part
                                                   ! of the WCB buffers
                                                   ! have changed since
                                                   ! last call to output.
    PBCB_W_FIRST_CHANGED_ROW = 168, 0, 16, 1%,
    PBCB_W_LAST_CHANGED_ROW  = 170, 0, 16, 1%,
    PBCB_W_FIRST_CHANGED_COL = 172, 0, 16, 1%,
    PBCB_W_LAST_CHANGED_COL  = 174, 0, 16, 1%,

    PBCB_A_OUTNAM           = 176, 0, 32, 0%,      ! Address of buffer containing
                                                   ! the output filename as
                                                   ! specified by the user
                                                   ! (or "SYS$OUTPUT" if not specified).

    PBCB_W_SPEED            = 180, 0, 16, 0%,      ! Terminal speed
      PBCB_B_TSPEED         = 180, 0,  8, 0%,      ! transmit speed
      PBCB_B_RSPEED         = 181, 0,  8, 0%,      ! receive  speed

    PBCB_W_FILL             = 182, 0, 16, 0%,      ! Terminal fill
      PBCB_B_CRFILL         = 182, 0,  8, 0%,      ! CR fill
      PBCB_B_LFFILL         = 183, 0,  8, 0%,      ! LF fill

    PBCB_A_BROADCAST_RTN    = 184, 0, 32, 0%,      ! Broadcast mailbox AST routine
    PBCB_L_BROADCAST_ARG    = 188, 0, 32, 0%,      ! Broadcast mailbox AST argument
    PBCB_A_UNSOLICIT_RTN    = 192, 0, 32, 0%,      ! Unsolicited input mailbox AST routine
    PBCB_L_UNSOLICIT_ARG    = 196, 0, 32, 0%,      ! Unsolicited input mailbox AST argument

    PBCB_Q_BROADCAST_MSG_QUEUE = 200,0,0,0%,       ! Queue for holding broadcast messages
```

```
        PBCB_L_SMGMBX_FLINK  = 200, 0, 32, 0%,      ! Forward link
        PBCB_L_SMGMBX_BLINK  = 204, 0, 32, 0%,      ! Backward link

      PBCB_W_FLAGS           = 208, 0, 16, 0%,      ! Flags
        PBCB_V_BROADCAST     = 208, 0,  1, 0%,      ! Broadcast msg trapping enabled
        PBCB_V_UNSOLICIT     = 208, 1,  1, 0%,      ! Unsolicited input notification enabled
        PBCB_V_SMGMBX_INIT   = 208, 2,  1, 0%,      ! 1 tells AST routine this is
                                                    ! an initialization call
        PBCB_V_RMS           = 208, 3,  1, 0%,      ! 1 means using RMS for output
        PBCB_V_LOCKED        = 208, 4,  1, 0%,      ! PBCB is locked
        PBCB_V_REBUILD       = 208, 5,  1, 0%,      ! A rebuild is needed
        PBCB_V_CONTROLO      = 208, 6,  1, 0%,      ! Previous output aborted by CTRL/O
        PBCB_V_CANCEL_CONTROLO=208, 7,  1, 0%,      ! Cancel CTRL/O on next QIO
        PBCB_V_BS            = 208, 8,  1, 0%,      ! 1 means terminal can do backspace
        PBCB_V_COMPLEX_BORDER= 208, 9,  1, 0%,      ! 1 means some border capability
                                                    ! is longer than a byte

      PBCB_W_ASYNC_CHAN      = 210, 0, 16, 0%,      ! Asynchronous channel to terminnal
      PBCB_W_MBX_CHAN        = 212, 0, 16, 0%,      ! Mailbox channel
      PBCB_W_SMGMBX_BUFSIZ   = 214, 0, 16, 0%,      ! Max message size for mailbox
      PBCB_A_SMGMBX_BUFFER   = 216, 0, 32, 0%,      ! Address of mailbox buffer
      PBCB_Q_SMGMBX_IOSB     = 220, 0,  0, 0%,      ! I/O status block for mbx read
      PBCB_W_OUTNAM_LEN      = 228, 0, 16, 0%,      ! Length of output name string
      PBCB_W_ORIG_WIDTH      = 230, 0, 16, 0%,      ! Original width of terminal
      PBCB_A_FAB             = 232, 0, 32, 0%,      ! Address of FAB (if file)
      PBCB_A_RAB             = 236, 0, 32, 0%,      ! Address of RAB (if file)
      PBCB_A_RBF             = 240, 0, 32, 0%,      ! Address of record buffer

      PBCB_W_TOP_SCROLL_LINE = 244, 0, 16, 0%,      ! Top scroll line
      PBCB_W_BOT_SCROLL_LINE = 246, 0, 16, 0%,      ! Bottom scroll line
                                                    ! Above 2 words record
                                                    ! where the physical
                                                    ! scrolling region is
                                                    ! currently set on the
                                                    ! terminal.
      PBCB_B_ORIG_HEIGHT     = 248, 0,  8, 0%,      ! Original number of rows on
                                                    ! terminal (reserved for
                                                    ! future use).
      PBCB_B_BACKGROUND_COLOR= 249, 0,  8, 0%,      ! Background color
      PBCB_W_INTERNAL_ATTR   = 250, 0, 16, 0%,      ! Internal attributes
          PBCB_V_WIDE        = 250, 0,  1, 0%,      ! Pasteboard allows wide lines
          PBCB_V_HIGH        = 250, 1,  1, 0%,      ! Pasteboard allows high wide lines
          PBCB_V_TABS        = 250, 2,  1, 0%,      ! Pbd allows physical tabs
      PBCB_L_TERMTABLE       = 252, 0, 32, 0%,      ! Corresponding TERMTABLE.
      PBCB_L_LONGEST_SEQUENCE= 256, 0, 32, 0%,      ! Longest capability sequence
      PBCB_A_CAP_BUFFER      = 260, 0, 32, 0%,      ! Address of capability buffer
      PBCB_L_CAP_LENGTH      = 264, 0, 32, 0%,      ! Length of last capability gotten
      PBCB_R_BORDER_VECTOR   = 268, 0,  0, 0%;      ! 16-longword border vector

LITERAL

      PBCB_K_SIZE = 332;                  ! Total size of  PBCB in bytes.

MACRO
```

```
SMG

!-

REQ
REQ
REQ
REQ
REQ
REQ
REQ
REQ

%IF
%TH
%FI

%IF
%TH



%FI

%IF
%TH
%FI

! E
```

```
    $PBCB_DECL = BLOCK[PBCB_K_SIZE,BYTE] %;

LITERAL ! masks for bits in field PBCB_L_MODE_SETTINGS

    PBCB_M_BUF_ENABLED   =  1,    ! =1 if buffering enabled
    PBCB_M_MINUPD        =  2,    ! =1 if minimal update enabled
    PBCB_M_CLEAR_SCREEN  =  4,    ! =1 if should clear screen on image exit
    PBCB_M_NOTABS        =  8;    ! =1 if SMG should not use physical tabs

LITERAL

    PBCB_K_DEF_MODE_SETTINGS =

        PBCB_M_MINUPD ;           ! Minimum update enabled
```

```
!+
! Window Control Block (WCB)
! --------------------------
!
!       This data structure resides in heap storage.  There is
! (currently) one WCB associated with each pasteboard and is pointed to
! by it.
! After an output operation, WCB_A_TEXT_BUF and WCB_A_SCR_TEXT_BUF have
! their contents swapped.  At the same time, WCB_A_ATTR_BUF and
! WCB_A_SCR_ATTR_BUF have their contents swapped.  Hence, at any point
! in time, WCB_A_SCR_TEXT_BUF and WCB_A_SCR_ATTR_BUF record what is
! currently on the screen.  WCB_A_TEXT_BUF and WCB_A_ATTR_BUF are used
! to construct the next screen full.
!-
MACRO
    WCB_Q_COORD            =  0, 0, 00, 0%,  ! Really 0, 0, 64, 0
                 ! Quadword containing next four words.  The four fields
                 ! define the coordinate system for the pasteboard and
                 ! hence the window buffer.  Their address is passed to
                 ! transmit these 4 fields as a single parameter.

        WCB_W_ROW_START =  0, 0, 16, 0%,   ! Row number of 1st row (=1)
        WCB_W_NO_ROWS   =  2, 0, 16, 0%,   ! Number of rows
        WCB_W_COL_START =  4, 0, 16, 0%,   ! Col number of 1st col (=1)
        WCB_W_NO_COLS   =  6, 0, 16, 0%,   ! Number of cols


    WCB_A_TEXT_BUF         =  8, 0, 32, 0%,   ! Address of a text buffer
                                              ! for this window.

    WCB_A_ATTR_BUF         = 12, 0, 32, 0%,   ! Address of attribute buffer
                                              ! for this window.

    WCB_A_CHAR_SET_BUF     = 16, 0, 32, 0%,   ! Address of character set
                                              ! buffer for this window.
                                              ! This buffer is allocated
                                              ! only if needed.

    WCB_A_SCR_TEXT_BUF     = 20, 0, 32, 0%,   ! Address of text buffer
                                              ! representing what is
                                              ! currently on the screen.

    WCB_A_SCR_ATTR_BUF     = 24, 0, 32, 0%,   ! Address of attribute buffer
                                              ! associated with
                                              ! WCB_A_SCR_TEXT_BUF

    WCB_A_SCR_CHAR_SET_BUF=28, 0, 32, 0%,     ! Address of character set
                                              ! buffer associated with
                                              ! WCB_A_SCH_TEXT_BUF.
                                              ! Allocated only if needed.

    WCB_W_CURR_CUR_ROW     = 32, 0, 16, 1%,
    WCB_W_CURR_CUR_COL     = 34, 0, 16, 1%,
                                              ! Cursor position in
                                              ! WCB_A_TEXT_BUF.
    WCB_W_OLD_CUR_ROW      = 36, 0, 16, 1%,
    WCB_W_OLD_CUR_COL   = 38, 0, 16, 1%,
                                          ! Cursor position in
                                          ! WCB_A_SCR_TEXT_BUF.

    WCB_L_BUFSIZE       = 40, 0, 32, 0%,  ! = .WCB [WCB_W_NO_ROWS] *
                                          !   .WCB [WCB_W_NO_COLS]

    WCB_A_LINE_CHAR     = 44, 0, 32, 0%,  ! Address of line
                                          ! characteristics vector for
                                          ! text buffer.
                                          ! This vector, one byte for
                                          ! each line, records whether
                                          ! the line is Single, Double-
                                          ! High, Double-Wide, etc.
                                          ! This vector is allocated to
                                          ! be WCB_W_NO_ROWS + 1 bytes
                                          ! long so it can be indexed
                                          ! directly by row number
                                          ! (1 through WCB_W_NO_ROWS).

    WCB_A_SCR_LINE_CHAR = 48, 0, 32, 0%;  ! Address of line
                                          ! characteristics vector for
                                          ! screen text buffer.
                                          ! This vector, one byte for
                                          ! each line, records whether
                                          ! the line is Single, Double-
                                          ! High, Double-Wide, etc.
                                          ! This vector is allocated to
                                          ! be WCB_W_NO_ROWS + 1 bytes
                                          ! long so it can be indexed
                                          ! directly by row number
                                          ! (1 through WCB_W_NO_ROWS).

LITERAL
    WCB_K_SIZE = 52;     ! No. of bytes in a WCB

MACRO

    $WCB_DECL = BLOCK[WCB_K_SIZE,BYTE] %;
```

```
!+
! Pasting Packet (PP)
!       This data structure defines the layout of a Pasting Packet.
! this area is allocated in heap storage.  One such structure exists for
! every pasting of a virtual display to a pasteboard.  It exists
! simultaneously on two queues -- the queue (DCB_A_PP_NEXT) headed in
! the DCB which contains the queue of PP's representing to which
! pasteboards this DCB is pasted, and at the same time, it is a member
! of the queue (PBCB_A_PP_NEXT) headed in the PBCB which contains the
! queue of PP's representing all virtual displays pasted to this
! pasteboard.
!-
MACRO
    PP_A_NEXT_DCB       =        0, 0, 32, 0%,
    PP_A_PREV_DCB       =        4, 0, 32, 0%,
                                ! The previous 2 longwords serve as
                                ! a queue entry for purposes of enqueing
                                ! onto queue DCB_A_PP_NEXT -- the queue
                                ! of all pasteboards to which this
                                ! virtual display is pasted.

    PP_A_NEXT_PBCB      =        8, 0, 32, 0%,
    PP_A_PREV_PBCB      =       12, 0, 32, 0%,
                                ! The previous 2 longwords serve as
                                ! a queue entry for purposes of enqueing
                                ! onto queue PBCB_A_PP_NEXT -- the queue
                                ! of all virtual displays which are
                                ! pasted to this pasteboard.

    PP_A_DCB_ADDR       =       16, 0, 32, 0%,
                                ! Address of the DCB involved in this
                                ! pasting.

    PP_A_PBCB_ADDR      =       20, 0, 32, 0%,
                                ! Address of the PBCB involved in this
                                ! pasting.

    PP_W_ROW            =       24, 0, 16, 1%,
                                ! The row number of the pasteboard onto
                                ! which row 1 of the virtual display
                                ! maps.

    PP_W_COL            =       26, 0, 16, 1%,
                                ! The column number of the pasteboard
                                ! onto which column 1 of the virtual
                                ! display maps.

    PP_W_ROWS_TO_MOVE   =       28, 0, 16, 0%,
                                ! The number of rows which have to be
                                ! moved from the display buffer to the
                                ! window buffer.  If zero, the next 3
                                ! fields are meaningless.

    PP_W_FROM_INDEX     =       30, 0, 16, 0%,
                                ! The byte index beyond the begining of
                                ! a source buffer which represents the
```

**F

```
                                ! 1st byte position to be moved when
                                ! copying from display buffers to
                                ! window buffers.

   PP_W_TO_INDEX          =    32, 0, 16, 0%,
                                ! The byte index beyond the begining of
                                ! a destination buffer which represents
                                ! the 1st byte to be deposited when
                                ! copying from a display buffers to
                                ! window buffers.

   PP_W_MOVE_LENGTH       =    34, 0, 16, 0%,
                                ! The number of columns to be moved from
                                ! display buffers to window buffers.

   PP_W_LABEL_BYTES_TO_MOVE =  36, 0, 16, 0%,
                                ! The number of bytes of the border
                                ! label which need to be moved to the
                                ! WCB text buffer when the virtual
                                ! display is mapped to the WCB text
                                ! buffer.  This field may be zero if no
                                ! characters fit the way the display is
                                ! pasted.

   PP_W_SRC_LABEL_OFF       =  38, 0, 16, 0%,
                                ! The offset (0-based) beyond the
                                ! beginning of the the label string
                                ! which represents the 1st byte of the
                                ! label to be moved.
                                ! (= length of label string -
                                !   .PP [PP_W_LABEL_BYTES_TO_MOVE]).
                                ! This field may not be valid if
                                ! .PP [PP_W_LABEL_BYTES_TO_MOVE] is 0.

   PP_W_DST_LABEL_OFF       =  40, 0, 16, 0%,
                                ! The offset (0-based) beyond the
                                ! beginning of .WCB [WCB_A_TEXT_BUF]
                                ! where the first byte of the visible
                                ! label lands.  This same offset is used
                                ! to reach the appropriate attribute
                                ! byte in WCB [WCB_A_ATTR_BUF].  This
                                ! field may not be valid if
                                ! .PP [PP_W_LABEL_BYTES_TO_MOVE] is 0.

                ! NOTE:  The 7 fields above are computed as a function
                ! of both the dimensions of the virtual display and
                ! the position on the pasteboard on which it is pasted.
                ! If either ( dimensions or pasting position) changes,
                ! these fields must be recomputed.
                ! They are initially set up as a result of pasting.
                ! Having these fields precomputed in the PP makes the
                ! output operation of moving data from display buffer
                ! to window buffer run faster since these values do not
                ! have to be continually recomputed on the fly.

   PP_B_CONTROL_BITS        =  42, 0,  8, 0%,
                                        ! Home for various PP-wide status bits.
        PP_V_OCCLUDED           =  42, 0,  1, 0%,
                                        ! Records whether the pasting of this
                                        ! virtual display to this pasteboard --
                                        ! taking into consideration all other
                                        ! pasting to this pasteboard -- is
                                        ! occluded or not.
                                        ! 1 => Occluded, 0=> Not occluded
        PP_V_CONTIG             =  42, 1,  1, 0%,
                                        ! If set, this bit means that the
                                        ! virtual display (as pasted) can be
                                        ! moved to the window buffer via a
                                        ! single CH$MOVE-- i.e. the source
                                        ! and destination fields are contiguous
                                        ! bytes.
                                        ! If not set, text must be moved on a
                                        ! row by row basis since only the bytes
                                        ! within a row are contiguous.
    PP_L_MOVE_SIZE              =  43, 0, 32, 0%,
                                        ! = .PP [PP_W_ROWS_TO_MOVE] *
                                        !    .PP [PP_W_MOVE_LENGTH]

                                        ! Next 4 fields tell where on the WCB
                                        ! buffer the part of the virtual display
                                        ! that fits within the pasteboard
                                        ! projects.  I.e., what area of the
                                        ! WCB buffers get modified when this
                                        ! virtual display is mapped to the
                                        ! pasteboard.  These fields are not
                                        ! meaningful if PP_W_ROWS_TO_MOVE is
                                        ! zero since it then doesn't even hit
                                        ! the pasteboard.
                                        ! Note: these fields do not take into
                                        ! account whether the virtual display
                                        ! is bordered.  For most instances this
                                        ! is the right thing to do.  There is
                                        ! one known quirk that needs to be fixed
                                        ! later-- If a virtual display is pasted
                                        ! to a row or column which is one unit
                                        ! outside of the pasteboard boundaries,
                                        ! then its PP_W_ROWS_TO_MOVE will be 0
                                        ! since the display itself will not
                                        ! project onto the pasteboard -- but its
                                        ! border will !!!
    PP_W_FIRST_WCB_ROW          = 47, 0, 16, 0%,
    PP_W_LAST_WCB_ROW           = 49, 0, 16, 0%,
    PP_W_FIRST_WCB_COL          = 51, 0, 16, 0%,
    PP_W_LAST_WCB_COL           = 53, 0, 16, 0%;

LITERAL
    PP_PBCB_QUEUE_OFFSET = 8,           ! Offset of the queue header for the
                                        ! pasteboard side of the chain.  This is
                                        ! the byte offset of the
                                        ! PBCB_A_NEXT_PBCB field.
    PP_K_SIZE           = 55;    ! Size in bytes of a PP
MACRO
    $PP_DECL = BLOCK[PP_K_SIZE,BYTE] %;
```

```
!+
! The following are masks of bit positions in the bytes of the
! attribute array pointed to by WCB [WCB_A_ATTR_BUF].

LITERAL
    ATTR_M_REND_BOLD    = %X'01',       ! Bold rendition
    ATTR_M_REND_REV     = %X'02',       ! Reverse video rendition
    ATTR_M_REND_BLINK   = %X'04',       ! Blink rendition
    ATTR_M_REND_UNDER   = %X'08',       ! Underline rendition

    ATTR_M_REND_GRAPHIC = %X'10',       ! Line-drawing character set
                ! If set, this bit indicates that this character must
                ! be rendered using the device's line-drawing character
                ! set.

    ATTR_M_USER_GRAPHIC = %X'40',       ! User line-drawing char set
                ! This indicates a generic line-drawing character which
                ! must be converted to the device-specific character
                ! before being output to the screen.

    ATTR_M_BORD_ELEM    = %X'80';       ! Border control bit.
                ! This bit is used to record that the associated text
                ! byte is (was) not a printable text byte, but a
                ! component of a border element.  This bit is not
                ! supplied by the caller to SMG, but is established
                ! internally while virtual displays are being mapped to
                ! the pasteboard buffer.  It is not of interest to the
                ! output routines.
                ! During the mapping phase, while the various virtual
                ! displays are mapped onto the output window buffer,
                ! this bit is used to record the fact that an element of
                ! a border occupies the corresponding cell in the WCB
                ! text buffer.  It is used to distinqiush a normal ASCII
                ! text character from an encoding of what pieces of a
                ! border element must occupy this text slot.
                ! After all virtual displays have been mapped to the
                ! WCB buffers and all intersections of border characters
                ! have been resolved,  each byte in the attribute array
                ! is inspected to see if it constains this bit.  If the
                ! bit in the attribute byte is set, the bits in the text
                ! byte are inspected to see how they should be rendered.
                ! If the device associated with the pasteboard does not
                ! support a line-drawing character set, the bits in the
                ! text byte are changed to the closest ASCII character
                ! approximation -- "+", "-", or "|".
                ! This cellular position can now be treated like any
                ! other text postion.
                ! However, if the associated device does support a line-
                ! drawing character set, the bits in the text cell are
                ! changed to the appropriate graphic code for that line-
                ! drawing set and the ATTR_M_GRAPHIC bit in the
                ! attribute array byte is turned on.
                ! The bits (in the text buffer) that encode the desired
                ! graphic are given below.

! These masks are used to set and reset these bits, e.g.
!       byte_in_attr_array = .byte_in_attr_arry OR ATTR_M_????  ! Set
!       byte_in_attr_array = .byte_in_attr_arry XOR ATTR_M_???? ! Reset
!

! The following are the corresponding constants for accessing the
! bits for interrogation purposes.
LITERAL
    ATTR_V_REND_BOLD    = %X'00',          ! Bold
    ATTR_V_REND_REV     = %X'01',          ! Reverse video
    ATTR_V_REND_BLINK   = %X'02',          ! Blink
    ATTR_V_REND_UNDER   = %X'03',          ! Underline

    ATTR_V_REND_GRAPHIC = %X'04',          ! Graphic character
    ATTR_V_USER_GRAPHIC = %X'06',          ! User graphic character
    ATTR_V_BORD_ELEM    = %X'07';          ! Border element control bit

!       These bits are used in BLISS constructs like:
!       IF .(some_place_in_the_attribute_buffer)<ATTR_V_???,1> ! If set

! These constants are used to check the line characteristics vector.
! The line characteristics vector is used to specify double wide and
! double high/double wide.

LITERAL

    LINE_K_NORMAL       = 0,               ! single width and height (must be 0)
    LINE_K_WIDE         = 1,               ! double wide
    LINE_K_UPPER_HIGH   = 2,               ! upper half of double high
    LINE_K_LOWER_HIGH   = 3;               ! lower half of double high
```

```
!+
! The following are bit definitions of bits found in the text buffer
! pointed to by WCB [WCB_A_TEXT_BUF] when a particular cell contains
! not a printable character, but an encoding of what parts of a border
! character need to be placed in this position.
!-

LITERAL
    BORD_M_RIGHT          = %X'01',
    BORD_M_UP             = %X'02',
    BORD_M_LEFT           = %X'04',
    BORD_M_DOWN           = %X'08',

    BORD_M_HORIZ          = BORD_M_RIGHT + BORD_M_LEFT,
    BORD_M_VERT           = BORD_M_UP    + BORD_M_DOWN,

    BORD_M_ULCORN         = BORD_M_DOWN  + BORD_M_RIGHT,
    BORD_M_URCORN         = BORD_M_DOWN  + BORD_M_LEFT,
    BORD_M_LLCORN         = BORD_M_UP    + BORD_M_RIGHT,
    BORD_M_LRCORN         = BORD_M_UP    + BORD_M_LEFT;

! Certain combinations of the above bit patterns are meaningful, e.g.
! BORD_M_VERT + BORD_M_RIGHT represent a "right-T" graphic
!
! These bits are used to "OR" together the right total graphic that is
! needed at a particular position on the screen to represent some
! element of a border.


! The corresponding bit positions:
!
LITERAL
    BORD_V_RIGHT          = %X'00',
    BORD_V_UP             = %X'01',
    BORD_V_LEFT           = %X'02',
    BORD_V_DOWN           = %X'03';
```

0355 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SMGDATSTR
REQ

SMGLNK
REQ

SMGKQB
SDL

SMGBLDTRM
LIS

SMGDISCHA
LIS

SMGKTH
SDL

SMGMACROS
REQ

SMGPROLOG
REQ

SMGTERM
REQ

SMGTRMSTR
R32

SMGTRMPTR
SDL

SMGSCRMAC
REQ

SMGALLESC
LIS

SMGTPACTL
REQ

SMGSCRTCB
REQ

SMGLIB
REQ

SMGTPALIB
REQ

SMGDEFKEY
LIS

SMGTABDEF
REQ

SMGTRMMAC
REQ

SMGBOOTAB
LIS