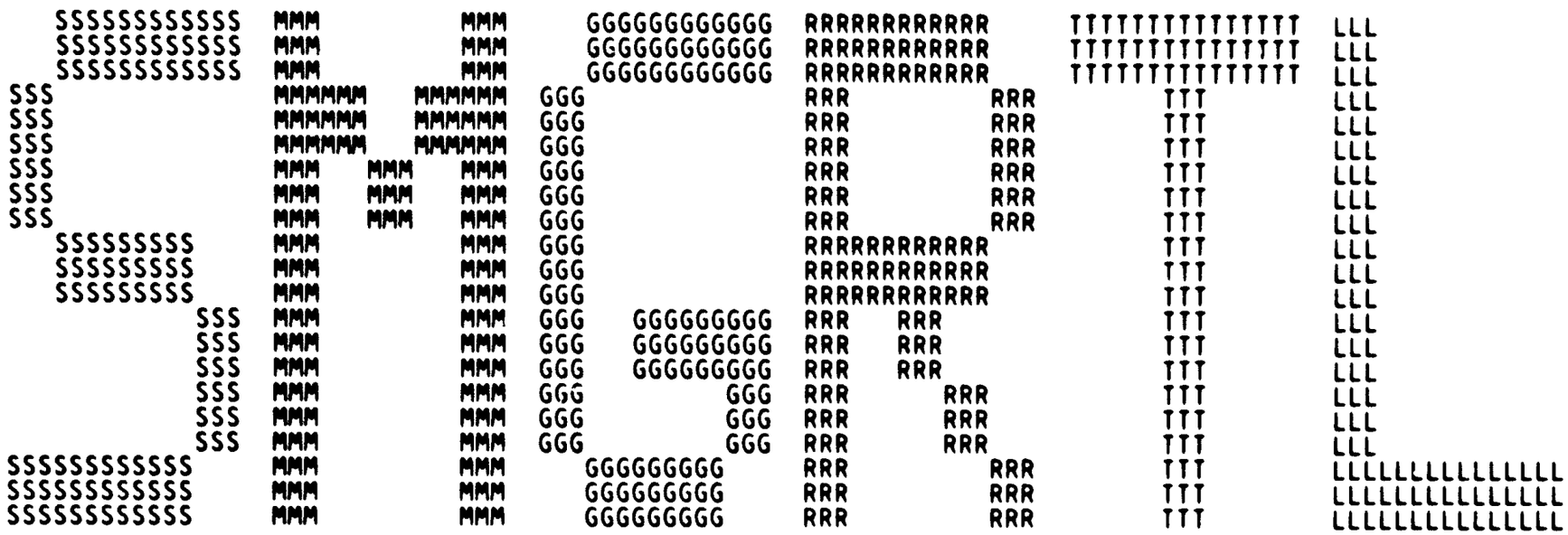

_$2

Val

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

FILEID**SMGLIB

```
  SSSSSSSS  MM      MM    GGGGGGGG  LL            IIIIII    BBBBBBBB
  SSSSSSSS  MM      MM    GGGGGGGG  LL            IIIIII    BBBBBBBB
SS          MMMM  MMMM  GG          LL              II      BB      BB
SS          MMMM  MMMM  GG          LL              II      BB      BB
SS          MM  MM  MM  GG          LL              II      BB      BB
SS          MM  MM  MM  GG          LL              II      BB      BB
  SSSSSS    MM      MM  GG          LL              II      BBBBBBBB
  SSSSSS    MM      MM  GG          LL              II      BBBBBBBB
        SS  MM      MM  GG  GGGGGG  LL              II      BB      BB
        SS  MM      MM  GG  GGGGGG  LL              II      BB      BB
        SS  MM      MM  GG      GG  LL              II      BB      BB      ....
        SS  MM      MM  GG      GG  LL              II      BB      BB      ....
SSSSSSSS    MM      MM    GGGGGG    LLLLLLLLLL    IIIIII    BBBBBBBB      ....
SSSSSSSS    MM      MM    GGGGGG    LLLLLLLLLL    IIIIII    BBBBBBBB      ....

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'SMGLIB - Screen Management BLISS Library'
;  0002  0     ! Run-Time Library Screen Management BLISS Definition Library
;  0003  0     ! File: SMGLIB.REQ, Edit: STAN1009
;  0004  0     !
;  0005  0     !*****************************************************************************
;  0006  0     !*                                                                           *
;  0007  0     !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;  0008  0     !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;  0009  0     !*  ALL RIGHTS RESERVED.                                                     *
;  0010  0     !*                                                                           *
;  0011  0     !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
;  0012  0     !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
;  0013  0     !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
;  0014  0     !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
;  0015  0     !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
;  0016  0     !*  TRANSFERRED.                                                             *
;  0017  0     !*                                                                           *
;  0018  0     !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
;  0019  0     !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
;  0020  0     !*  CORPORATION.                                                             *
;  0021  0     !*                                                                           *
;  0022  0     !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
;  0023  0     !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;  0024  0     !*                                                                           *
;  0025  0     !*                                                                           *
;  0026  0     !*****************************************************************************
;  0027  0     !
;  0028  0     ! Author: Steven B. Lionel, 29-April-1983
;  0029  0     !
;  0030  0     ! 1-001 - Original.  SBL 29-April-1983
;  0031  0     ! 1-002 - Correct logical name for RTLML.  SBL 30-Apr-1983
;  0032  0     ! 1-003 - Don't include SMGDEF since it comes from STARLET.  SBL 2-May-1983
;  0033  0     ! 1-004 - Add in SMGTERM.REQ and SMGLNK.REQ for SMG$$ESCAPE_GENERATOR.
;  0034  0     !         LEB 23-May-1983
;  0035  0     ! 1-005 - Fix reference for SMGTERM and SMGLNK to RTLIN.  LEB 24-May-1983
;  0036  0     ! 1-006 - Add output-specific require files.  PLL 14-Jun-1983
;  0037  0     ! 1-007 - Conditionally require SMGDEF.  (These definitions may already be
;  0038  0     !         in Starlet, depending on the version of Starlet used.)  Move macros
;  0039  0     !         and linkages to SMGMACROS.REQ and SMGLNK.REQ.  PLL 21-Jun-1983
;  0040  0     ! 1-008 - Require SMGTRMPTR.
;  0041  0     ! 1-009 - Until V4 build, temporarily define SMG$K_TOP, etc.
;  0042  0     !--
;  0043  0     !
;  0044  0     !+
;  0045  0     ! This file is the master source for SMGLIB.L32.
;  0046  0     ! It contains definitions for macros and symbols used internally to the
;  0047  0     ! Run-Time Library Screen Management procedures.
;  0048  0     !-
;  0049  0     !
;  0050  0     SWITCHES ADDRESSING_MODE (EXTERNAL=GENERAL, NONEXTERNAL=WORD_RELATIVE);
;  0051  0     !
;  0052  0     LIBRARY 'RTLSTARLE';                    ! SYS$LIBRARY:STARLET.L32
;  0053  0     !
;  0054  0     !
;  0055  0     !+
;  0056  0     ! Screen Management specific definitions.
;  0057  0     !-
```

```
;  0058  0
;  0059  0     REQUIRE 'RTLML:SMGKCB';
```

```
;  R0060  0      ! ****************************************************************************************************
;  R0061  0      !  Created 15-SEP-1984 22:52:20  by  VAX-11 SDL V2.0      Source: 15-SEP-1984 22:49:53 _$255$DUA28:[SMGRTL.SRC]SMGK
;  R0062  0      ! ****************************************************************************************************
;  R0063  0
;  R0064  0
;  R0065  0      !*** MODULE KCBDEF IDENT 1-002 ***
;  R0066  0      literal KCB_K_ORIGIN_OFFSET = 48;
;  R0067  0      literal KCB_K_QIO1_ARGCNT = 12;
;  R0068  0      literal KCB_K_QIO2_ARGCNT = 12;
;  R0069  0      literal KCB_S_TERMCHAR = 12;
;  R0070  0      literal KCB_S_KCB = 52;
;  R0071  0      literal KCB_S_QIO2 = 52;
;  R0072  0      literal KCB_S_IOSB = 8;
;  R0073  0      literal KCB_S_DEVNAM_STRING = 64;
;  R0074  0      FIELD KCB_STRUCT$FIELDSET =
;  R0075  0        SET
;  R0076  0        KCB_L_EFN = [-48,0,32,0] ,
;  R0077  0          KCB_W_CHANNEL = [-44,0,16,1] ,
;  R0078  0          KCB_W_IFI = [-44,0,16,1] ,
;  R0079  0        KCB_R_UNION1 = [-44,0,16,0] ,
;  R0080  0        KCB_W_AST_CHANNEL = [-42,0,16,1] ,
;  R0081  0        KCB_A_KQB = [-40,0,32,0] ,
;  R0082  0        KCB_L_CHECK = [-36,0,32,0] ,
;  R0083  0        KCB_L_OLD_DEVDEPEND1 = [-32,0,32,0] ,
;  R0084  0        KCB_L_OLD_DEVDEPEND2 = [-28,0,32,0] ,
;  R0085  0        KCB_L_DEVCHAR = [-24,0,32,0] ,
;  R0086  0          KCB_B_DEVCLASS = [-20,0,8,1] ,
;  R0087  0          KCB_B_DEVTYPE = [-19,0,8,1] ,
;  R0088  0          KCB_W_PAGE_WIDTH = [-18,0,16,1] ,
;  R0089  0          KCB_L_DEVDEPEND1 = [-16,0,32,0] ,
;  R0090  0          KCB_L_DEVDEPEND2 = [-12,0,32,0] ,
;  R0091  0        KCB_R_TERMCHAR = [-20,0,0,0] ,
;  R0092  0        KCB_L_PASTEBOARD_ID = [-8,0,32,0] ,
;  R0093  0          KCB_V_RMS = [-4,0,1,0] ,
;  R0094  0          KCB_V_CTRLZ = [-4,1,1,0] ,
;  R0095  0          KCB_V_CHARS_CHANGED = [-4,2,1,0] ,
;  R0096  0          KCB_V_KPDSEQ_DECCRT = [-4,3,1,0] ,
;  R0097  0        KCB_R_FLAGS = [-4,0,32,0] ,
;  R0098  0          KCB_R_RAB = [0,0,8,0]
;  R0099  0            KCB_L_QIO1_ARGCNT = [0,0,32,0] ,
;  R0100  0            KCB_L_QIO1_EFN = [4,0,32,0] ,
;  R0101  0            KCB_L_QIO1_CHAN = [8,0,32,0] ,
;  R0102  0            KCB_L_QIO1_FUNC = [12,0,32,0] ,
;  R0103  0            KCB_A_QIO1_IOSB = [16,0,32,0] ,
;  R0104  0            KCB_A_QIO1_ASTADR = [20,0,32,0] ,
;  R0105  0            KCB_L_QIO1_ASTPRM = [24,0,32,0] ,
;  R0106  0            KCB_L_QIO1_P1 = [28,0,32,0] ,
;  R0107  0            KCB_L_QIO1_P2 = [32,0,32,0] ,
;  R0108  0            KCB_L_QIO1_P3 = [36,0,32,0] ,
;  R0109  0            KCB_L_QIO1_P4 = [40,0,32,0] ,
;  R0110  0            KCB_L_QIO1_P5 = [44,0,32,0] ,
;  R0111  0            KCB_L_QIO1_P6 = [48,0,32,0] ,
;  R0112  0          KCB_R_QIO1 = [0,0,0,0] ,
;  R0113  0        KCB_R_KCB = [0,0,0,0]
;  R0114  0          KCB_L_QIO2_ARGCNT = [52,0,32,0] ,
;  R0115  0          KCB_L_QIO2_EFN = [56,0,32,0] ,
;  R0116  0          KCB_L_QIO2_CHAN = [60,0,32,0] ,
```

```
;  R0117  0          KCB_L_QIO2_FUNC = [64,0,32,0] ,
;  R0118  0          KCB_A_QIO2_IOSB = [68,0,32,0] ,
;  R0119  0          KCB_A_QIO2_ASTADR = [72,0,32,0] ,
;  R0120  0          KCB_L_QIO2_ASTPRM = [76,0,32,0] ,
;  R0121  0          KCB_L_QIO2_P1 = [80,0,32,0] ,
;  R0122  0          KCB_L_QIO2_P2 = [84,0,32,0] ,
;  R0123  0          KCB_L_QIO2_P3 = [88,0,32,0] ,
;  R0124  0          KCB_L_QIO2_P4 = [92,0,32,0] ,
;  R0125  0          KCB_L_QIO2_P5 = [96,0,32,0] ,
;  R0126  0          KCB_L_QIO2_P6 = [100,0,32,0] ,
;  R0127  0        KCB_R_QIO2 = [52,0,0,0]
;  R0128  0          KCB_W_IOSB_STATUS = [104,0,16,1] ,
;  R0129  0          KCB_W_IOSB_COUNT = [106,0,16,0] ,
;  R0130  0          KCB_B_IOSB_TERMINATOR = [108,0,8,0] ,
;  R0131  0          KCB_B_IOSB_TERMLEN = [110,0,8,0] ,
;  R0132  0          KCB_B_IOSB_POS = [111,0,8,0] ,
;  R0133  0        KCB_R_IOSB = [104,0,0,0]
;  R0134  0        KCB_W_DEVNAM_LENGTH = [112,0,16,1] ,
;  R0135  0        KCB_T_DEVNAM_STRING = [114,0,0,0]
;  R0136  0        TES;
;  R0137  0      literal KCB_S_KCB_STRUCT = 226;
;  R0138  0      MACRO KCB_R_KCB_STRUCT = BLOCK [KCB_S_KCB_STRUCT,byte] FIELD (KCB_STRUCT$FIELDSET) %;
```

```
;  0139  0     REQUIRE 'RTLML:SMGKDE';
```

```
;  R0140  0      ! ****************************************************************************************************
;  R0141  0      !  Created 15-SEP-1984 22:52:26  by  VAX-11 SDL V2.0      Source: 15-SEP-1984 22:50:00 _$255$DUA28:[SMGRTL.SRC]SMGK
;  R0142  0      ! ****************************************************************************************************
;  R0143  0
;  R0144  0
;  R0145  0      !*** MODULE KDEDEF IDENT 1-003 ***
;  R0146  0      literal KDE_M_NOECHO = 1;
;  R0147  0      literal KDE_M_TERMINATE = 2;
;  R0148  0      literal KDE_M_LOCK = 4;
;  R0149  0      literal KDE_M_PROTECTED = 8;
;  R0150  0      literal KDE_M_SETSTATE = 16;
;  R0151  0      literal KDE_M_RESERVED = 224;
;  R0152  0      literal KDE_M_DEFINED = -2147483648;
;  R0153  0      literal KDE_M_USER_ATTR = 255;
;  R0154  0      literal KDE_M_SMG_ATTR = -256;
;  R0155  0      literal KDE_S_HEADER = 10;
;  R0156  0      literal KDE_S_TKEY_DESC = 8;
;  R0157  0      literal KDE_S_EQUIV_DESC = 8;
;  R0158  0      literal KDE_S_STATE_DESC = 8;
;  R0159  0      literal KDE_S_PREFIX_DESC = 8;
;  R0160  0      FIELD KDE_STRUCT$FIELDSET =
;  R0161  0        SET
;  R0162  0        KDE_W_IF_STATE_LENGTH = [10,0,16,0] ,
;  R0163  0          KDE_W_TKEY_LENGTH = [12,0,16,0] ,
;  R0164  0          KDE_B_TKEY_DTYPE = [14,0,8,1] ,
;  R0165  0          KDE_B_TKEY_CLASS = [15,0,8,1] ,
;  R0166  0          KDE_A_TKEY_POINTER = [16,0,32,0] ,
;  R0167  0        KDE_R_TKEY_DESC = [12,0,0,0] ,
;  R0168  0          KDE_W_EQUIV_LENGTH = [20,0,16,0] ,
;  R0169  0          KDE_B_EQUIV_DTYPE = [22,0,8,1] ,
;  R0170  0          KDE_B_EQUIV_CLASS = [23,0,8,1] ,
;  R0171  0          KDE_A_EQUIV_POINTER = [24,0,32,0] ,
;  R0172  0        KDE_R_EQUIV_DESC = [20,0,0,0] ,
;  R0173  0          KDE_W_STATE_LENGTH = [28,0,16,0] ,
;  R0174  0          KDE_B_STATE_DTYPE = [30,0,8,1] ,
;  R0175  0          KDE_B_STATE_CLASS = [31,0,8,1] ,
;  R0176  0          KDE_A_STATE_POINTER = [32,0,32,0] ,
;  R0177  0        KDE_R_STATE_DESC = [28,0,0,0] ,
;  R0178  0          KDE_W_PREFIX_LENGTH = [36,0,16,0] ,
;  R0179  0          KDE_B_PREFIX_DTYPE = [38,0,8,1] ,
;  R0180  0          KDE_B_PREFIX_CLASS = [39,0,8,1] ,
;  R0181  0          KDE_A_PREFIX_POINTER = [40,0,32,0] ,
;  R0182  0        KDE_R_PREFIX_DESC = [36,0,0,0] ,
;  R0183  0        KDE_A_NEXT = [44,0,32,0] ,
;  R0184  0          KDE_L_ATTR = [48,0,32,0] ,
;  R0185  0              KDE_V_NOECHO = [48,0,1,0] ,
;  R0186  0              KDE_V_TERMINATE = [48,1,1,0] ,
;  R0187  0              KDE_V_LOCK = [48,2,1,0] ,
;  R0188  0              KDE_V_PROTECTED = [48,3,1,0] ,
;  R0189  0              KDE_V_SETSTATE = [48,4,1,0] ,
;  R0190  0              KDE_V_RESERVED = [48,5,3,0] ,
;  R0191  0              KDE_V_DEFINED = [48,31,1,0] ,
;  R0192  0            KDE_R_ATTR_BITS = [48,0,32,0] ,
;  R0193  0              KDE_V_USER_ATTR = [48,0,8,0] ,
;  R0194  0              KDE_V_SMG_ATTR = [48,8,24,0] ,
;  R0195  0            KDE_R_ATTR_FIELDS = [48,0,32,0] ,
;  R0196  0          KDE_R_ATTR_UNION = [48,0,32,0] ,
```

```
;  R0197  0          KDE_r_fill_1 = [48,0,32,0] ,
;  R0198  0        KDE_r_fill_0 = [48,0,32,0] ,
;  R0199  0        KDE_L_TERM_MASK = [52,0,32,0]
;  R0200  0        TES;
;  R0201  0      literal KDE_S_KDE_STRUCT = 56;
;  R0202  0      MACRO KDE_R_KDE_STRUCT = BLOCK [KDE_S_KDE_STRUCT,byte] FIELD (KDE_STRUCT$FIELDSET) %;
```

```
;   0203  0     REQUIRE 'RTLML:SMGKTH';
```

```
;  R0204  0      ! ****************************************************************************************************
;  R0205  0      !  Created 15-SEP-1984 22:52:35  by  VAX-11 SDL V2.0       Source: 15-SEP-1984 22:50:13  $255$DUA28:[SMGRTL.SRC]SMGK
;  R0206  0      ! ****************************************************************************************************
;  R0207  0
;  R0208  0
;  R0209  0      !*** MODULE KTHDEF IDENT 1-002 ***
;  R0210  0      literal KTH_S_DEF_STATE_DESCR = 8;
;  R0211  0      literal KTH_S_DEF_STATE_STRING = 34;
;  R0212  0      FIELD KTH_STRUCT$FIELDSET =
;  R0213  0        SET
;  R0214  0        KTH_A_TREEHEAD = [0,0,32,0] ,
;  R0215  0          KTH_L_FLAGS = [4,0,32,0] ,
;  R0216  0          KTH_r_fill_1 = [4,0,32,0] ,
;  R0217  0        KTH_r_fill_0 = [4,0,32,0] ,
;  R0218  0        KTH_L_MODIFIERS = [8,0,32,0] ,
;  R0219  0        KTH_L_TERM_MASK = [12,0,32,0] ,
;  R0220  0        KTH_L_CHECK = [16,0,32,0] ,
;  R0221  0          KTH_W_DEF_STATE_LEN = [20,0,16,0] ,
;  R0222  0          KTH_B_DEF_STATE_DTYPE = [22,0,8,1] ,
;  R0223  0          KTH_B_DEF_STATE_CLASS = [23,0,8,1] ,
;  R0224  0          KTH_A_DEF_STATE_POINTER = [24,0,32,0] ,
;  R0225  0        KTH_R_DEF_STATE_DESCR = [20,0,0,0] ,
;  R0226  0        KTH_A_DEF_KEYCODE = [28,0,32,0] ,
;  R0227  0        KTH_T_DEF_STATE_STRING = [32,0,0,0]
;  R0228  0        TES;
;  R0229  0      literal KTH_S_KTH_STRUCT = 66;
;  R0230  0      MACRO KTH_R_KTH_STRUCT = BLOCK [KTH_S_KTH_STRUCT,byte] FIELD (KTH_STRUCT$FIELDSET) %;
```

```
;   0231  0     REQUIRE 'RTLML:SMGKQB';
```

```
;  R0232  0     ! ****************************************************************************************************
;  R0233  0     !  Created 15-SEP-1984 22:52:30  by  VAX-11 SDL V2.0      Source: 15-SEP-1984 22:50:06  _$255$DUA28:[SMGRTL.SRC]SMGK
;  R0234  0     ! ****************************************************************************************************
;  R0235  0
;  R0236  0
;  R0237  0     !*** MODULE KQBDEF IDENT 1-001 ***
;  R0238  0     literal KQB_S_fill_0 = 8;
;  R0239  0     FIELD KQB_STRUCT$FIELDSET =
;  R0240  0       SET
;  R0241  0         KQB_Q_QUEUE_LINK = [0,0,0,0] ,
;  R0242  0           KQB_A_FLINK = [0,0,32,0] ,
;  R0243  0           KQB_A_BLINK = [4,0,32,0] ,
;  R0244  0         KQB_r_fill_1 = [0,0,0,0] ,
;  R0245  0       KQB_r_fill_0 = [0,0,0,0] ,
;  R0246  0       KQB_A_KCB = [8,0,32,0]
;  R0247  0       TES;
;  R0248  0     literal KQB_S_KQB_STRUCT = 12;
;  R0249  0     MACRO KQB_R_KQB_STRUCT = BLOCK [KQB_S_KQB_STRUCT,byte] FIELD (KQB_STRUCT$FIELDSET) %;
```

```
;   0250  0     REQUIRE 'RTLIN:SMGLNK';
```

```
;  R0251  0     ! Linkage Definitions for RTL SMG$ facility
;  R0252  0     ! File:  SMGLNK.REQ     Edit: RKR1004
;  R0253  0     !
;  R0254  0     !
;  R0255  0     !*****************************************************************************
;  R0256  0     !*                                                                           *
;  R0257  0     !*  COPYRIGHT (c) 1-78, 1980, 1982, 1984 BY                                  *
;  R0258  0     !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;  R0259  0     !*  ALL RIGHTS RESERVED.                                                     *
;  R0260  0     !*                                                                           *
;  R0261  0     !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
;  R0262  0     !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
;  R0263  0     !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
;  R0264  0     !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
;  R0265  0     !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
;  R0266  0     !*  TRANSFERRED.                                                             *
;  R0267  0     !*                                                                           *
;  R0268  0     !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
;  R0269  0     !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
;  R0270  0     !*  CORPORATION.                                                             *
;  R0271  0     !*                                                                           *
;  R0272  0     !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
;  R0273  0     !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;  R0274  0     !*                                                                           *
;  R0275  0     !*                                                                           *
;  R0276  0     !*****************************************************************************
;  R0277  0     !
;  R0278  0     !
;  R0279  0     !++
;  R0280  0     ! FACILITY:     Screen Management
;  R0281  0     !
;  R0282  0     ! ABSTRACT:
;  R0283  0     !
;  R0284  0     ! This file contains linkage definitions which are needed by
;  R0285  0     ! various routines in the screen package.
;  R0286  0     !
;  R0287  0     !
;  R0288  0     !
;  R0289  0     ! MODIFIED BY:
;  R0290  0     !
;  R0291  0     ! 1-001 - Original.  PLL 7-Mar-1983
;  R0292  0     ! 1-002 - Add linkages for input routines.  (Formerly in SMGLIB.REQ.)
;  R0293  0     !         PLL 21-Jun-1983
;  R0294  0     ! 1-003 - Add linkages for pasteboard batching routines.
;  R0295  0     !         RKR 2-DEC-1983.
;  R0296  0     ! 1-004 - Names introduced in last edit too long, shorten them. RKR 2-DEC-1983.
;  R0297  0     !--
;  R0298  0     !
;  R0299  0     !
;  R0300  0     LINKAGE
;  R0301  0             SMG$$ESC_R2_LNK = JSB (REGISTER = 0, REGISTER = 1, REGISTER = 2) :
;  R0302  0                               NOPRESERVE (0, 1, 2),
;  R0303  0             !
;  R0304  0             SMG$$ESC_R4_LNK = JSB (REGISTER = 0, REGISTER = 1, REGISTER = 2,
;  R0305  0                               REGISTER = 3, REGISTER = 4) :
;  R0306  0                               NOPRESERVE (0, 1, 2, 3, 4);
;  R0307  0
```

```
;  R0308  0
;  R0309  0      !+
;  R0310  0      ! Linkages for input
;  R0311  0      !-
;  R0312  0
;  R0313  0      LINKAGE
;  R0314  0          SMG$$CLEANUP$LNK =
;  R0315  0              CALL (REGISTER=8, REGISTER=0),
;  R0316  0          SMG$$LOOKUP_KEY$LNK =
;  R0317  0              JSB (REGISTER=0, REGISTER=1; REGISTER=6):
;  R0318  0              NOTUSED (2,3,4,5,7,8,9,10,11),
;  R0319  0          SMG$$VALIDATE_KTH$LNK =
;  R0320  0              JSB (REGISTER=0; REGISTER=8):
;  R0321  0              NOPRESERVE (2,3) NOTUSED (4,5,6,7,9,10,11);
;  R0322  0
;  R0323  0      !+
;  R0324  0      ! Linkages for pasteboard batching routines
;  R0325  0      !-
;  R0326  0      LINKAGE
;  R0327  0          SMG$$BEGIN_PBD_UPDATE$LNK =
;  R0328  0              JSB (REGISTER=0):
;  R0329  0              NOPRESERVE (0,1) NOTUSED (2,3,4,5,6,7,8,9,10,11),
;  R0330  0
;  R0331  0          SMG$$END_PBD_UPDATE$LNK =
;  R0332  0              JSB (REGISTER=0):
;  R0333  0              NOPRESERVE (0,1,2) NOTUSED (3,4,5,6,7,8,9,10,11);
;  R0334  0
;  R0335  0
```

```
;   0336  0     REQUIRE 'RTLIN:SMGTERM';
```

```
R0337  0     ! Terminal Definitions for RTL SMG$ facility
R0338  0     ! File:  SMGTERM.REQ    Edit: STAN1006
R0339  0     !
R0340  0     !
R0341  0     !****************************************************************************
R0342  0     !*                                                                          *
R0343  0     !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
R0344  0     !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
R0345  0     !*  ALL RIGHTS RESERVED.                                                    *
R0346  0     !*                                                                          *
R0347  0     !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
R0348  0     !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
R0349  0     !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
R0350  0     !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
R0351  0     !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
R0352  0     !*  TRANSFERRED.                                                            *
R0353  0     !*                                                                          *
R0354  0     !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
R0355  0     !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
R0356  0     !*  CORPORATION.                                                            *
R0357  0     !*                                                                          *
R0358  0     !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
R0359  0     !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
R0360  0     !*                                                                          *
R0361  0     !*                                                                          *
R0362  0     !****************************************************************************
R0363  0     !
R0364  0     !
R0365  0     !++
R0366  0     ! FACILITY:     Screen Management
R0367  0     !
R0368  0     ! ABSTRACT:
R0369  0     !
R0370  0     !  This file contains terminal type definitions and escape sequences
R0371  0     !  which are needed by various routines in the screen management package.
R0372  0     !
R0373  0     !
R0374  0     !
R0375  0     ! MODIFIED BY:
R0376  0     !
R0377  0     ! 1-006 - STAN 15-MAR-1984. Comment obsolete items.
R0378  0     ! 1-005 - STAN 22-Jan-1984. Add VTTERMTABLE.
R0379  0     ! 1-004 - STAN 5-Oct-1983. Changed BLOB and DIAMOND to be above control
R0380  0     !         character range.  Thus they represent a high order nibble
R0381  0     !         of 10 and 6 respectively.
R0382  0     ! 1-003 - Added VT, HT, BLOB, and DIAMOND. PLL 23-Sep-1983
R0383  0     ! 1-002 - STAN 1-May-1983
R0384  0     !         Removed BUFSIZE.
R0385  0     !         Added TAB.
R0386  0     ! 1-001 - Original.  PLL 15-Mar-1983
R0387  0     !--
```

```
;  R0388  0      !+
;  R0389  0      ! Characters
;  R0390  0      ! ----------
;  R0391  0      !-
;  R0392  0      LITERAL
;  R0393  0          BLANK       = %X'20',       ! blank (space)
;  R0394  0          BS          = %X'08',       ! Backspace
;  R0395  0          FF          = %X'0C',       ! Form feed
;  R0396  0          LF          = %X'0A',       ! Line feed
;  R0397  0          CR          = %X'0D',       ! Carriage return
;  R0398  0          ESC         = %X'1B',       ! Escape
;  R0399  0          LB          = %X'5B',       ! Left bracket (VT100)
;  R0400  0          SEMI        = %X'3B',       ! Semi-colon
;  R0401  0          NULL        = %X'00',       ! Null
;  R0402  0          BELL        = %X'07',       ! Bell character
;  R0403  0          TAB         = %X'09',       ! Horizontal tab
;  R0404  0          A           = %X'41',       ! Capital letter A
;  R0405  0          B           = %X'42',       ! Capital letter B
;  R0406  0          C           = %X'43',       ! Capital letter C
;  R0407  0          H           = %X'48',       ! Capital letter H
;  R0408  0          f           = %X'66',       ! Small letter f
;  R0409  0          TWO         = %X'32',       ! Two
;  R0410  0          VT          = %X'0B',       ! Vertical tab
;  R0411  0          HT          = %X'09',       ! Horizontal tab (same as TAB)
;  R0412  0
;  R0413  0          !+
;  R0414  0          ! The following two codes form a %X'10' and a %X'6' in the high order nibble.
;  R0415  0          !-
;  R0416  0
;  R0417  0          BLOB        = %X'100',      ! This is not the correct ascii
;  R0418  0          DIAMOND     = %X'60';       ! This is also made up
;  R0419  0
;  R0420  0      !+
;  R0421  0      ! Miscellaneous constants
;  R0422  0      ! ----------------------
;  R0423  0      !-
;  R0424  0      LITERAL
;  R0425  0          CB          = %X'1F';       ! Line/column bias in message
;  R0426  0
;  R0427  0      !+
;  R0428  0      ! Terminal type definitions
;  R0429  0      ! -------------------------
;  R0430  0      !
;  R0431  0      ! Types VT05, VT52, and VT100 will be going away soon.
;  R0432  0      !-
;  R0433  0      LITERAL
;  R0434  0          UNKNOWN     = 0,            ! Non-graphics or unknown type
;  R0435  0          VT05        = 1,            ! VT05 series terminal (OBSOLETE)
;  R0436  0          VT52        = 2,            ! VT5x series terminal (OBSOLETE)
;  R0437  0          VT100       = 3,            ! VT100 series terminal (OBSOLETE)
;  R0438  0          VTFOREIGN   = 4,            ! Foreign terminal (FT1-8)
;  R0439  0          HARDCOPY    = 5,            ! Hardcopy device
;  R0440  0          VTTERMTABLE = 6;            ! Terminal support is via TERMTABLE
;  R0441  0
;  R0442  0      !+
;  R0443  0      ! VT05 Codes (OBSOLETE)
;  R0444  0      ! ----------
```

```
;  R0445  0      !-
;  R0446  0      LITERAL
;  R0447  0          VT05_SC     = %X'0E',       ! Set cursor position
;  R0448  0          VT05_HOME   = %X'1D',       ! Set cursor to home
;  R0449  0          VT05_CUP    = %X'1A',       ! Cursor up
;  R0450  0          VT05_EOL    = %X'1E',       ! Erase to end of line
;  R0451  0          VT05_EOS    = %X'1F';       ! Erase to end of screen
;  R0452  0
;  R0453  0      !+
;  R0454  0      ! VT52 Codes (OBSOLETE)
;  R0455  0      ! ----------
;  R0456  0      !-
;  R0457  0      LITERAL
;  R0458  0          VT52_SC     = %X'59',       ! Set cursor position
;  R0459  0          VT52_HOME   = %X'48',       ! Set cursor to home
;  R0460  0          VT52_DWN    = %X'49',       ! Down scroll
;  R0461  0          VT52_EOS    = %X'4A',       ! Erase to end of screen
;  R0462  0          VT52_EOL    = %X'4B' ;      ! Erase to end of line
;  R0463  0
;  R0464  0      !+
;  R0465  0      ! VT100 Codes (OBSOLETE)
;  R0466  0      ! -----------
;  R0467  0      !-
;  R0468  0      LITERAL
;  R0469  0          VT100_SC    = %X'66',       ! Set cursor position
;  R0470  0          VT100_DWN   = %X'4D',       ! Down scroll
;  R0471  0          VT100_EOS   = %X'4A',       ! Erase to end of screen
;  R0472  0          VT100_EOL   = %X'4B',       ! Erase to end of line
;  R0473  0          VT100_SGR   = %X'6D',       ! Select graphic rendition
;  R0474  0          VT100_SM    = %X'72' ;      ! Set scrolling region
```

```
;   0475  0     REQUIRE 'RTLIN:SMGDATSTR';
```

```
;  R0476  0      ! Data Structure Definitions for RTL SMG$ facility
;  R0477  0      ! File:  SMGDATSTR.REQ  Edit: STAN1054
;  R0478  0      !
;  R0479  0      !
;  R0480  0      !*****************************************************************************
;  R0481  0      !*                                                                           *
;  R0482  0      !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;  R0483  0      !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;  R0484  0      !*  ALL RIGHTS RESERVED.                                                     *
;  R0485  0      !*                                                                           *
;  R0486  0      !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
;  R0487  0      !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
;  R0488  0      !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
;  R0489  0      !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
;  R0490  0      !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
;  R0491  0      !*  TRANSFERRED.                                                             *
;  R0492  0      !*                                                                           *
;  R0493  0      !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
;  R0494  0      !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
;  R0495  0      !*  CORPORATION.                                                             *
;  R0496  0      !*                                                                           *
;  R0497  0      !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
;  R0498  0      !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;  R0499  0      !*                                                                           *
;  R0500  0      !*                                                                           *
;  R0501  0      !*****************************************************************************
;  R0502  0      !
;  R0503  0
;  R0504  0      !++
;  R0505  0      ! FACILITY:     Screen Management
;  R0506  0      !
;  R0507  0      ! ABSTRACT:
;  R0508  0      !
;  R0509  0      !       This file contains data structure definitions for screen management
;  R0510  0      !       routines.  Display Control Block, Pasteboard Control Block, Window
;  R0511  0      !       Control Block, and Pasting Packet are defined here.
;  R0512  0      !
;  R0513  0      !
;  R0514  0      !
;  R0515  0      ! MODIFIED BY:
;  R0516  0      !
;  R0517  0      ! 1-001 - Original.  PLL 15-Mar-1983
;  R0518  0      ! 1-002 - Expand PBCB to hold PBCB_L_MODE_SETTINGS.  RKR 17-Mar-1983.
;  R0519  0      ! 1-003 - Add literal for initial setting of PBCB_L_MODE_SETTINGS.
;  R0520  0      !         RKR 18-Mar-1983.
;  R0521  0      ! 1-004 - Add stuff for borders, alternate character sets, etc.
;  R0522  0      !         RKR 24-Mar-1983.
;  R0523  0      ! 1-005 - Corrections to last edit.  RKR 24-Mar-1983.
;  R0524  0      ! 1-006 - Expand WCB and PP structures.  RKR 28-Mar-1983.
;  R0525  0      ! 1-007 - Fix typo in last edit.  RKR 28-Mar-1983.
;  R0526  0      ! 1-008 - Pull BACKGROUND_COLOR out of PBCB since we don't know what
;  R0527  0      !         to do with it.  Add stuff pertaining to borders.
;  R0528  0      !         RKR 4-April-1983.
;  R0529  0      ! 1-009 - More fixes.  RKR 4-April-1983.
;  R0530  0      ! 1-010 - Clean up rest of border masks.  RKR 5-April-1983
;  R0531  0      ! 1-011 - More additions for labeled borders.  RKR 7-April-1983.
;  R0532  0      ! 1-012 - Add more fields to PBCB and PP.  RKR 14-APR-1983.
```

```
;  R0533  0      ! 1-013 - Rearrange the bits for the line-drawing character set and the
;  R0534  0      !         bit used to designate a border element. RKR 15-APR-1983.
;  R0535  0      ! 1-014 - Define 2 more bits in the DCB, one to mean that all lines are
;  R0536  0      !         full (and thus scrolling should occur thereafter), and one to
;  R0537  0      !         mean that column 80 was just written (useful on the next write
;  R0538  0      !         operation).  PLL 28-Apr-1983
;  R0539  0      ! 1-015 - Define a bell bit in the DCB and WCB.  PLL 29-Apr-1983
;  R0540  0      ! 1-016 - New fields in PBCB. STAN 28-Apr-1983 (2nd try).
;  R0541  0      !         Changed PID to PBID so as not to confuse a pasteboard id
;  R0542  0      !         with a process id.
;  R0543  0      ! 1-017 - Added event flag numbers to PBCB. STAN 30-Apr-1983
;  R0544  0      ! 1-018 - STAN 1-May-1983
;  R0545  0      !         Added SMG$K_LONGEST_SEQUENCE.
;  R0546  0      !         Added output buffer fields to PBCB.
;  R0547  0      !         Added rows and columns to PBCB.
;  R0548  0      ! 1-019 - STAN 3-May-1983
;  R0549  0      !         Allow up to 16 pasteboards.
;  R0550  0      !         Put out-of-band AST routine into PBCB. Literally!
;  R0551  0      ! 1-020 - STAN 4-May-1983
;  R0552  0      !         Fix typo in comment.
;  R0553  0      ! 1-021 - Add PP_B_CONTROL_BITS field in PP.  RKR 5-MAY-1983
;  R0554  0      ! 1-022 - Add PP_V_CONTIG bit in PP_B_CONTROL_BITS.  RKR 9-MAY-1983
;  R0555  0      ! 1-023 - Add to DCB, PP, and WCB a longword that is the product of
;  R0556  0      !         the number of rows * the number of columns.  (Not strictly
;  R0557  0      !         true for PP case -- but ignore it.
;  R0558  0      !         RKR 13-MAY-1983.
;  R0559  0      ! 1-024 - Add BATCHING bit to PBCB
;  R0560  0      !         STAN 16-May-1983
;  R0561  0      ! 1-025 - Remove bell bit.  PLL 20-May-1983
;  R0562  0      ! 1-026 - Extend DCB in anticpation of backup DCB logic.
;  R0563  0      !         Delete DD_K_MAX_VD -- should no longer be needed.
;  R0564  0      !         RKR 20-MAY-1983.
;  R0565  0      ! 1-027 - Add words in the DCB to store the top and bottom of a scrolling
;  R0566  0      !         region.  PLL 25-May-1983
;  R0567  0      ! 1-028 - Add new fields to PP and PBCB.  RKR 26-MAY-1983.
;  R0568  0      ! 1-029 - New fields to PBCB for mailboxes et. al. STAN 1-Jun-1983.
;  R0569  0      ! 1-030 - More of same. STAN  2-Jun-1983.
;  R0570  0      ! 1-031 - More of same. STAN 13-Jun-1983.
;  R0571  0      ! 1-032 - Add DCB_V_LABEL_CENTER control bit to DCB_L_CONTROL_BITS.
;  R0572  0      !         RKR 14-JUN-1983.
;  R0573  0      ! 1-033 - Make PP_W_ROW and PP_W_COL signed fields.
;  R0574  0      !         RKR 14-JUN-1983
;  R0575  0      ! 1-034 - Add bit DCB_V_PP_MISMATCH.  RKR 17-JUN-1983.
;  R0576  0      ! 1-035 - Fields for output files in PBCB. STAN 18-Jun-1983.
;  R0577  0      !         Made LAST_CHANGED fields in PBCB signed.
;  R0578  0      !         Created macros for declaring structures.
;  R0579  0      !         Made CURSOR position in PBCB signed.
;  R0580  0      ! 1-036 - Add structures to DCB and WCB dealing with line characteristics
;  R0581  0      !         like Double-Wide, Double-High, etc.
;  R0582  0      !         RKR 7-JUL-1983.
;  R0583  0      ! 1-037 - Add bit to DCB to mark as used for autobending.  PLL 7-Jul-1983
;  R0584  0      ! 1-038 - Add more longwords to DCB for autobending (used to parse
;  R0585  0      !         escape sequences).  PLL 8-Jul-1983
;  R0586  0      ! 1-039 - Add 2 words to PBCB to record where the physical scrolling
;  R0587  0      !         region is on the terminal.
;  R0588  0      !         RKR 11-JUL-1983.
;  R0589  0      ! 1-040 - Fix typo.  RKR 11-JUL-1983.
```

```
;  R0590  0      ! 1-041 - Save original terminal width and height. STAN 22-Aug-1983
;  R0591  0      ! 1-042 - STAN 31-Aug-1983 Line characteristics types.
;  R0592  0      ! 1-043 - Add a truncation icon attribute bit to the DCB. PLL 1-Sep-1983
;  R0593  0      ! 1-044 - Get rid of 2 unused fields in the DCB by renaming one to
;  R0594  0      !         a simulated device type and leaving the other one as a placeholder.
;  R0595  0      !         PLL 2-Sep-1983
;  R0596  0      ! 1-045 - Added some terminal characteristics constants.
;  R0597  0      !         STAN 5-Sep-1983.
;  R0598  0      ! 1-046 - Use up DCB_B_FILL by turning it into DCB_B_LABEL_REND.
;  R0599  0      !         RKR 15-SEP-1983.
;  R0600  0      ! 1-047 - Add a user line drawing bit the rendition attribute. PLL 21-Sep-1983
;  R0601  0      ! 1-048 - Background color byte. STAN 27-Sep-1983.
;  R0602  0      ! 1-049 - STAN 14-Oct-1983. Added wide and high bits; AST-reentrancy bits.
;  R0603  0      ! 1-050 - STAN 14-Oct-1983. CTRL/O bit.
;  R0604  0      ! 1-051 - STAN 17-Oct-1983. Add Cancel control/O bit.
;  R0605  0      ! 1-052 - STAN 15-Jan-1984. Add TERMTABLE.
;  R0606  0      ! 1-053 - STAN 21-Feb-1984. Add LF, TAB, and BS optimization bits in PBD.
;  R0607  0      ! 1-054 - STAN 6-Mar-1983. Add NOTABS bit.
;  R0608  0      !--
```

```
;  R0609  ;       !+
;  R0610  0       ! Critical sizes and counts for virtual displays and pasteboards
;  R0611  0       !-
;  R0612  0
;  R0613  0       LITERAL
;  R0614  0
;  R0615  0           PBD_K_MAX_PB = 16,  ! Maximum number of past.boards we can track.
;  R0616  0                               ! It controls the range of pasteboard id's we
;  R0617  0                               ! will allocate and the size of the pasteboard
;  R0618  0                               ! directory (PBD) structure in OWN storage.
;  R0619  0                               ! Currently constrained not to exceeed 32 by
;  R0620  0                               ! usage of FFC instruction in $GET_NEXT_PID.
;  R0621  0
;  R0622  0           SMG$K_LONGEST_SEQUENCE = 255;
;  R0623  0
;  R0624  0                               ! Longest control or escape sequence that
;  R0625  0                               ! can be returned by TERMTABLE routines.
;  R0626  0                               ! This value can be used to preallocate a
;  R0627  0                               ! buffer to hold the text or can be used to tell
;  R0628  0                               ! if the next sequence desired could overflow
;  R0629  0                               ! your buffer.
```

```
;  R0630  0      !+
;  R0631  0      ! Virtual Display Control Block (DCB)
;  R0632  0      ! -----------------------------------
;  R0633  0      !       This data structure defines the layout of a Virtual Display
;  R0634  0      ! Control Block.  The area is allocated in heap storage.  One such
;  R0635  0      ! block is allocated for each new virtual display created by callers.
;  R0636  0      ! It contains dimensions of the virtual display and pointers to other
;  R0637  0      ! buffers associated with this display.  It also contains pointers to
;  R0638  0      ! the pasteboards onto which it is pasted.  This area is deallocated
;  R0639  0      ! when the virtual display is deleted -- not when it is unpasted.
;  R0640  0      !-
;  R0641  0      MACRO
;  R0642  0
;  R0643  0          DCB_Q_COORD         =  0, 0, 00, 0%,   ! Really 0, 0, 64, 0
;  R0644  0                             ! Quadword containing next four words.  These 4 fields
;  R0645  0                             ! define the coordinate system for this virtual display
;  R0646  0                             ! and their address is transmitted to pass the 4 fields
;  R0647  0                             ! as a single parameter
;  R0648  0
;  R0649  0              DCB_W_ROW_START =  0, 0, 16, 0%,  ! Row number of 1st row. (=1)
;  R0650  0              DCB_W_NO_ROWS   =  2, 0, 16, 0%,  ! Number of rows
;  R0651  0              DCB_W_COL_START =  4, 0, 16, 0%,  ! Col number of 1st col (=1)
;  R0652  0              DCB_W_NO_COLS   =  6, 0, 16, 0%,  ! Number of columns
;  R0653  0
;  R0654  0          DCB_Q_LABEL_DESC    =  8, 0, 00, 0%,   ! Really 8, 0, 64, 0
;  R0655  0                                                 ! Dynamic string descriptor
;  R0656  0                                                 ! for border label text
;  R0657  0
;  R0658  0          DCB_A_TEXT_BUF      = 16, 0, 32, 0%,   ! Addr. of buffer containing
;  R0659  0                                                 ! text for this virtual
;  R0660  0                                                 ! display.
;  R0661  0
;  R0662  0          DCB_A_ATTR_BUF      = 20, 0, 32, 0%,   ! Addr. of buffer containing
;  R0663  0                                                 ! video attributes for each
;  R0664  0                                                 ! character position in
;  R0665  0                                                 ! TEXT_BUF.
;  R0666  0
;  R0667  0          DCB_A_CHAR_SET_BUF  = 24, 0, 32, 0%,   ! Addr. of buffer containing
;  R0668  0                                                 ! character set codes for
;  R0669  0                                                 ! each character in TEXT_BUF.
;  R0670  0                                                 ! This buffer allocated only
;  R0671  0                                                 ! when needed.
;  R0672  0
;  R0673  0          DCB_L_BATCH_LEVEL   = 28, 0, 32, 0%,   ! Number of levels of
;  R0674  0                                                 ! batching in effect for this
;  R0675  0                                                 ! display.
;  R0676  0                                                 ! Incremented by call to
;  R0677  0                                                 ! SMG$START_DISPLAY_UPDATE
;  R0678  0                                                 ! and decrement toward zero
;  R0679  0                                                 ! by each call to
;  R0680  0                                                 ! SMG$END_DISPLAY_UPDATE.
;  R0681  0                                                 ! Output flows from this
;  R0682  0                                                 ! virtual display to the
;  R0683  0                                                 ! screen only when this
;  R0684  0                                                 ! variable is zero.
;  R0685  0
;  R0686  0          DCB_A_PP_NEXT       = 32, 0, 32, 0%,
```

```
;  R0687  0          DCB_A_PP_PREV       = 36, 0, 32, 0%,
;  R0688  0                                                  ! Above two longwords are the
;  R0689  0                                                  ! queue header for the chain
;  R0690  0                                                  ! of Pasting Packets tied to
;  R0691  0                                                  ! this virtual display.
;  R0692  0
;  R0693  0          DCB_W_CURSOR_ROW    = 40, 0, 16, 0%,    ! Cursor row position in this
;  R0694  0                                                  ! virtual display
;  R0695  0
;  R0696  0          DCB_W_CURSOR_COL    = 42, 0, 16, 0%,    ! Cursor col position in this
;  R0697  0                                                  ! virtual display
;  R0698  0
;  R0699  0          DCB_W_LABEL_UNITS   = 44, 0, 16, 0%,    ! Starting position in the
;  R0700  0                                                  ! line or column indicated by
;  R0701  0                                                  ! DCB_B_LABEL_POS
;  R0702  0
;  R0703  0          DCB_B_DEF_VIDEO_ATTR= 46, 0,  8, 0%,    ! Default video attributes of
;  R0704  0                                                  ! this virtual display
;  R0705  0              DCB_V_RENBOL    = 46, 0,  1, 0%,    ! Bold
;  R0706  0              DCB_V_RENREV    = 46, 1,  1, 0%,    ! Reverse video
;  R0707  0              DCB_V_RENBLK    = 46, 2,  1, 0%,    ! Blink
;  R0708  0              DCB_V_RENUND    = 46, 3,  1, 0%,    ! Underline
;  R0709  0
;  R0710  0          DCB_B_DEF_DISPLAY_ATTR=47,0,  8, 0%,    ! Default display attributes
;  R0711  0                                                  ! of this virtual display
;  R0712  0              DCB_V_BORDERED  = 47, 0,  1, 0%,    ! Bordered
;  R0713  0
;  R0714  0              DCB_V_TRUNC_ICON = 47,1,  1, 0%,    ! Flag to use truncation icon
;  R0715  0
; MR0716  0              DCB_V_DISPLAY_CONTROLS =            ! Flag to display carriage control
;  R0717  0                               47, 2,  1, 0%,    ! characters such a <CR> instead
;  R0718  0                                                  ! of execute them
;  R0719  0
;  R0720  0          DCB_B_DEF_CHAR_SET  = 48, 0,  8, 0%,    ! Default character set for
;  R0721  0                                                  ! all text in this virtual
;  R0722  0                                                  ! display.
;  R0723  0
;  R0724  0          DCB_B_LABEL_POS     = 49, 0,  8, 0%,    ! Code for positioning of
;  R0725  0                                                  ! border label:
;  R0726  0                                                  !  0 = Top border line
;  R0727  0                                                  !  1 = Bottom border line
;  R0728  0                                                  !  2 = Left border line
;  R0729  0                                                  !  3 = Right border line
;  R0730  0
;  R0731  0          DCB_B_LABEL_CHAR_SET= 50, 0,  8, 0%,    ! Code for character set of
;  R0732  0                                                  ! border label.
;  R0733  0
;  R0734  0          DCB_B_LABEL_REND    = 51, 0,  8, 0%,    ! Rendition for border label
;  R0735  0
;  R0736  0          DCB_L_CONTROL_BITS  = 52, 0, 32, 0%,    ! Control bits
;  R0737  0
;  R0738  0              DCB_V_FULL      = 52, 0,  1, 0%,    ! All display lines used
;  R0739  0                                                  ! (next op may scroll)
;  R0740  0
;  R0741  0              DCB_V_COL_80    = 52, 1,  1, 0%,    ! Column 80 just written
;  R0742  0
;  R0743  0              DCB_V_LABEL_CENTER=52,2,  1, 0%,    ! If set indicates that
```

```
;  R0744  0                                          ! border label should be
;  R0745  0                                          ! centered -- even if virt.
;  R0746  0                                          ! display is redimensioned.
;  R0747  0
;  R0748  0          DCB_V_PP_MISMATCH=52, 3,  1, 0%,  ! If this bit is set it
;  R0749  0                                          ! indicates that this virtual
;  R0750  0                                          ! display control block
;  R0751  0                                          ! changed in such a way that
;  R0752  0                                          ! all associated pasting
;  R0753  0                                          ! packets need to have their
;  R0754  0                                          ! constants recalculated.
;  R0755  0                                          ! However, this change occurred
;  R0756  0                                          ! while the display was
;  R0757  0                                          ! "batched" and could not be
;  R0758  0                                          ! done at that time.
;  R0759  0                                          ! SMG$END_DISPLAY_UPDATE
;  R0760  0                                          ! senses this bit whenever it
;  R0761  0                                          ! makes the transition to
;  R0762  0                                          ! batch_level=0 and performs
;  R0763  0                                          ! the psting packet recalc.
;  R0764  0                                          ! at that time, then resets
;  R0765  0                                          ! this bit.
;  R0766  0
;  R0767  0          DCB_V_AUTOBENDED =52, 4,  1, 0%,  ! This DCB created by
;  R0768  0                                          ! autobended routines
;  R0769  0
;  R0770  0          DCB_V_ALLOW_ESC = 52, 5,  1, 0%,  ! Parse escape sequences when set
;  R0771  0
;  R0772  0          DCB_V_LOCKED    = 52, 6,  1, 0%,  ! DCB is locked for our use
;  R0773  0
;  R0774  0        DCB_L_DID         = 56, 0, 32, 0%,  ! Virtual display id
;  R0775  0                                          ! (Currently the address of
;  R0776  0                                          !  the DCB itself.)
;  R0777  0
;  R0778  0        DCB_L_BUFSIZE     = 60, 0, 32, 0%,  ! = .DCB [DCB_W_NO_ROWS] *
;  R0779  0                                          !   .DCB [DCB_W_NO_COLS]
;  R0780  0
;  R0781  0
;  R0782  0        DCB_A_BACKUP_DCB  = 64, 0, 32, 0%,  ! If non-zero, address of the
;  R0783  0                                          ! backup DCB when this DCB is
;  R0784  0                                          ! batched.  Backup DCB holds
;  R0785  0                                          ! the state of the DCB at the
;  R0786  0                                          ! time batching started.
;  R0787  0
;  R0788  0        DCB_B_STRUCT_TYPE = 68, 0,  8, 0%,  ! Code to mark this structure
;  R0789  0                                          ! as being a DCB
;  R0790  0
;  R0791  0        DCB_W_DCB_LENGTH  = 69, 0, 16, 0%,  ! Stored length of a DCB
;  R0792  0
;  R0793  0        DCB_B_FILL_2      = 71, 0,  8, 0%,  !
;  R0794  0
;  R0795  0        DCB_W_TOP_OF_SCRREG = 72, 0, 16, 0%,  ! Top line in scrolling region
;  R0796  0
;  R0797  0        DCB_W_BOTTOM_OF_SCRREG
;  R0798  0                          = 74, 0, 16, 0%,  ! Bottom line in scrolling region
;  R0799  0
;  R0800  0
```

```
;  R0801  0          DCB_A_LINE_CHAR     = 76, 0, 32, 0%,    ! Address of the line
;  R0802  0                                                   ! characteristics vector.
;  R0803  0                                                   ! This vector, one byte for
;  R0804  0                                                   ! each line, records whether
;  R0805  0                                                   ! the line is Single, Double-
;  R0806  0                                                   ! High, Double-Wide, etc.
;  R0807  0                                                   ! This vector is allocated to
;  R0808  0                                                   ! be DCB_W_NO_ROWS + 1 bytes
;  R0809  0                                                   ! long so it can be indexed
;  R0810  0                                                   ! directly by row number
;  R0811  0                                                   ! (1 through DCB_W_NO_ROWS).
;  R0812  0          DCB_SIM_CONTROL     = 80, 0, 32, 0%,    ! Control bits for SMG$$SIM_TERM
;  R0813  0          DCB_ARG_1           = 84, 0, 32, 0%,    ! Control seq arg 1
;  R0814  0          DCB_ARG_2           = 88, 0, 32, 0%,    ! Control seq arg 2
;  R0815  0          DCB_SIM_DEV_TYPE    = 92, 0, 32, 0%,    ! Device type to simulate
;  R0816  0          DCB_UNUSED          = 96, 0, 32, 0%,    ! Unused
;  R0817  0          DCB_SAVED_HPOS      = 100, 0, 32, 0%,   ! Saved cursor column
;  R0818  0          DCB_SAVED_VERT      = 104, 0, 32, 0%,   ! Saved cursor row
;  R0819  0          DCB_SAVED_VIDEO_ATTR = 108, 0, 32, 0%;  ! Saved video attributes
;  R0820  0
;  R0821  0
;  R0822  0      LITERAL
;  R0823  0
;  R0824  0          DCB_K_STRUCT_TYPE = %X'11',  ! Code stored in DCB [DCB_B_STRUCT_TYPE]
;  R0825  0                                       ! to mark is as being a DCB.
;  R0826  0
;  R0827  0          DCB_K_SIZE  = 112;           ! Total number of bytes in a DCB
;  R0828  0
;  R0829  0      MACRO
;  R0830  0
;  R0831  0          $DCB_DECL   = BLOCK[DCB_K_SIZE,BYTE] %;
```

```
;  R0832  0     !+
;  R0833  0     ! Pasteboard Control Block (PBCB)
;  R0834  0     ! -------------------------------
;  R0835  0     !       This data structure resides in HEAP storage.  One of these areas
;  R0836  0     ! is allocated whenever a new stream is established for the first time.
;  R0837  0     ! It is deallocated when the pasteboard is deleted.
;  R0838  0     ! It contains the fundemental information associated with a pasteboard
;  R0839  0     ! and pointers to PBCB-related structures like the WCB.
;  R0840  0     !-
;  R0841  0
;  R0842  0     MACRO
;  R0843  0
;  R0844  0         PBCB_A_PP_NEXT          =   0, 0, 32, 0%,
;  R0845  0         PBCB_A_PP_PREV          =   4, 0, 32, 0%,
;  R0846  0                                     ! Previous two longwords serve
;  R0847  0                                     ! as a queue header for the
;  R0848  0                                     ! chain of pasting packets of
;  R0849  0                                     ! all the virtual displays that
;  R0850  0                                     ! are pasted to this pasteboard.
;  R0851  0
;  R0852  0         PBCB_A_WCB              =   8, 0, 32, 0%,
;  R0853  0                                     ! Addr. of window control block
;  R0854  0                                     ! (WCB)
;  R0855  0
;  R0856  0         PBCB_L_MODE_SETTINGS    =  12, 0, 32, 0%,    ! Mode setting for this PBCB
;  R0857  0             PBCB_V_BUF_ENABLED  =  12, 0,  1, 0%,    ! =1 if buffering enabled
;  R0858  0             PBCB_V_MINUPD       =  12, 1,  1, 0%,    ! =1 if minimal update enabled
;  R0859  0             PBCB_V_CLEAR_SCREEN=  12, 2,  1, 0%,    ! =1 if should clear screen on exit
;  R0860  0             PBCB_V_NOTABS       =  12, 3,  1, 0%,    ! =1 if SMG should not use physical
;  R0861  0                                                      !    tabs.
;  R0862  0
;  R0863  0         PBCB_B_DEVTYPE          =  16, 0,  8, 0%,    ! Logical device type
;  R0864  0                                                      ! Status are defined in
;  R0865  0                                                      ! SMGTERM.REQ and
;  R0866  0                                                      ! currently are:
;  R0867  0                                                      !   UNKNOWN   = 0
;  R0868  0                                                      !   VT05      = 1
;  R0869  0                                                      !   VT52      = 2
;  R0870  0                                                      !   VT100     = 3
;  R0871  0                                                      !   VTFOREIGN = 4
;  R0872  0                                                      !   HARDCOPY  = 5
;  R0873  0
;  R0874  0         PBCB_B_PARITY           =  17, 0,  8, 0%,    ! parity flags
;  R0875  0
;  R0876  0         PBCB_W_DEVNAM_LEN       =  18, 0, 16, 0%,    ! Length of the
;  R0877  0                                                      ! resultant device name
;  R0878  0                                                      ! string contained in
;  R0879  0                                                      ! PBCB_T_DEVNAM.
;  R0880  0
;  R0881  0         PBCB_L_PBID             =  20, 0, 32, 0%,    ! Pasteboard id
;  R0882  0
;  R0883  0         PBCB_T_DEVNAM           =  24, 0,  0, 0%,    ! A 64-byte area.  This
;  R0884  0                                                      ! buffer contains the
;  R0885  0                                                      ! resultant device name
;  R0886  0                                                      ! string.  Its length
;  R0887  0                                                      ! is contained in
;  R0888  0                                                      ! PBCB_W_DEVNAM_LEN.
```

```
;  R0889  0
;  R0890  0          PBCB_R_CHARBUF          =  88, 0,  0, 0%,    ! Start of 12-byte
;  R0891  0                                                       ! characteristics buffer
;  R0892  0
;  R0893  0            PBCB_L_DEVCHAR        =  88, 0, 32, 0%,    ! Device characteristics
;  R0894  0
;  R0895  0              PBCB_B_CLASS        =  88, 0,  8, 0%,    ! Device class, e.g. DC$_TERM
;  R0896  0
;  R0897  0              PBCB_B_PHY_DEV_TYPE=  89, 0,  8, 0%,    ! Physical device type,
;  R0898  0                                                       ! e.g. DT$_VT100
;  R0899  0
;  R0900  0              PBCB_W_WIDTH        =  90, 0, 16, 0%,    ! Device width
;  R0901  0
;  R0902  0            PBCB_L_DEVDEPEND      =  92, 0, 32, 0%,    ! Primary device dependent
;  R0903  0                                                       ! bits.  These are the bits
;  R0904  0                                                       ! of the TT$V_xyz flavor.
;  R0905  0
;  R0906  0              PBCB_B_ROWS         =  92, 24, 8, 0%,    ! Number of rows on terminal
;  R0907  0                                                       ! (overlaps previous field)
;  R0908  0
;  R0909  0            PBCB_L_DEVDEPEND2     =  96, 0, 32, 0%,    ! Secondary device
;  R0910  0                                                       ! dependent bits.  These
;  R0911  0                                                       ! are the bits of the
;  R0912  0                                                       ! TT2$V_xyz flavor.
;  R0913  0
;  R0914  0          PBCB_W_CHAN             = 100, 0, 16, 0%,    ! Channel number. 0 means
;  R0915  0                                                       ! no channel as been assigned
;  R0916  0                                                       ! yet.
;  R0917  0
;  R0918  0          PBCB_B_EFN              = 102, 0,  8, 0%,    ! Primary output event flag
;  R0919  0
;  R0920  0          PBCB_B_ASYNC_EFN        = 103, 0,  8, 0%,    ! Secondary output event flag
;  R0921  0                                                       ! used for asynchronous operations
;  R0922  0
;  R0923  0          PBCB_A_MBX_MSG_LIST     = 104, 0, 32, 0%,    ! List of messages that came
;  R0924  0                                                       ! from our associated mailbox
;  R0925  0
;  R0926  0          PBCB_A_OUTPUT_BUFFER    = 108, 0, 32, 0%,    ! Address of buffer used to
;  R0927  0                                                       ! buffer up output sequences.
;  R0928  0
;  R0929  0          PBCB_W_OUTPUT_BUFSIZ    = 112, 0, 16, 0%,    ! (Maximum) size of output buffer
;  R0930  0
;  R0931  0          PBCB_W_OUTPUT_BUFLEN    = 114, 0, 16, 0%,    ! Current length of output buffer
;  R0932  0                                                       ! i.e. number of characters in
;  R0933  0                                                       ! the buffer. 0 means the
;  R0934  0                                                       ! buffer is empty.
;  R0935  0
;  R0936  0          PBCB_R_EXIT_BLOCK       = 116, 0,  0 ,0%,    ! Exit block (5 longwords)
;  R0937  0
;  R0938  0              PBCB_L_EXIT_LINK    = 116, 0, 32, 0%,    ! system forward link to next block
;  R0939  0              PBCB_A_EXIT_ADDR    = 120, 0, 32, 0%,    ! address of our exit handler
;  R0940  0              PBCB_B_EXIT_ARGCNT  = 124, 0,  8, 0%,    ! argument count (=2)
;  R0941  0              PBCB_A_EXIT_RSN     = 128, 0, 32, 0%,    ! arg 1: address to store exit reason
;  R0942  0              PBCB_A_EXIT_PBCB    = 132, 0, 32, 0%,    ! arg 2: our PBCB address
;  R0943  0
;  R0944  0          PBCB_L_EXIT_REASON      = 136, 0, 32, 0%,    ! exit reason (address stored
;  R0945  0                                                       ! as first argument in exit
```

```
;  R0946  0                                                          . block).
;  R0947  0
;  R0948  0        PBCB_Z_OUT_OF_BAND_RTN = 140, 0,  0, 0%,          ! ten-byte routine resides here.
;  R0949  0            PBCB_W_ENTRY_MASK  = 140, 0, 16, 0%,          ! 0000 You may not believe it,
;  R0950  0            PBCB_B_CALLG       = 142, 0,  8, 0%,          !   FA but 'tis so.  The first
;  R0951  0            PBCB_B_REG_AP      = 143, 0,  8, 0%,          !   6C word is the entry mask.
;  R0952  0            PBCB_B_ABS         = 144, 0,  8, 0%,          !   9F
;  R0953  0            PBCB_A_BAND_HANDLER= 145, 0, 32, 0%,          !      Address of generic
;  R0954  0            PBCB_B_RET         = 149, 0,  8, 0%,          !   04 out-of-band AST handler
;  R0955  0
;  R0956  0        PBCB_A_BAND_ROUTINE    = 152, 0, 32, 0%,          ! Address of user's AST routine
;  R0957  0                                                          ! for out-of-band characters.
;  R0958  0                                                          ! 0 means out-of-band ASTs
;  R0959  0                                                          ! are not enabled.
;  R0960  0
;  R0961  0        PBCB_L_BAND_AST_ARG    = 156, 0, 32, 0%,          ! User's arg to his AST routine
;  R0962  0
;  R0963  0        PBCB_M_BAND_MASK       = 160, 0, 32, 0%,          ! Character mask for out-of-
;  R0964  0                                                          ! band ASTs currently in effect
;  R0965  0        PBCB_L_BATCH_LEVEL     = 164, 0, 32, 0%,          ! Batching level.  If non-0,
;  R0966  0                                                          ! then batching is in effect.
;  R0967  0
;  R0968  0                                                          ! Next 4 fields are set
;  R0969  0                                                          ! during mapping from
;  R0970  0                                                          ! virtual display to
;  R0971  0                                                          ! pasteboard buffers
;  R0972  0                                                          ! and describe what part
;  R0973  0                                                          ! of the WCB buffers
;  R0974  0                                                          ! have changed since
;  R0975  0                                                          ! last call to output.
;  R0976  0        PBCB_W_FIRST_CHANGED_ROW = 168, 0, 16, 1%,
;  R0977  0        PBCB_W_LAST_CHANGED_ROW  = 170, 0, 16, 1%,
;  R0978  0        PBCB_W_FIRST_CHANGED_COL = 172, 0, 16, 1%,
;  R0979  0        PBCB_W_LAST_CHANGED_COL  = 174, 0, 16, 1%,
;  R0980  0
;  R0981  0        PBCB_A_OUTNAM          = 176, 0, 32, 0%,          ! Address of buffer containing
;  R0982  0                                                          ! the output filename as
;  R0983  0                                                          ! specified by the user
;  R0984  0                                                          ! (or "SYS$OUTPUT" if not specified).
;  R0985  0
;  R0986  0        PBCB_W_SPEED           = 180, 0, 16, 0%,          ! Terminal speed
;  R0987  0          PBCB_B_TSPEED        = 180, 0,  8, 0%,          ! transmit speed
;  R0988  0          PBCB_B_RSPEED        = 181, 0,  8, 0%,          ! receive  speed
;  R0989  0
;  R0990  0        PBCB_W_FILL            = 182, 0, 16, 0%,          ! Terminal fill
;  R0991  0          PBCB_B_CRFILL        = 182, 0,  8, 0%,          ! CR fill
;  R0992  0          PBCB_B_LFFILL        = 183, 0,  8, 0%,          ! LF fill
;  R0993  0
;  R0994  0        PBCB_A_BROADCAST_RTN   = 184, 0, 32, 0%,          ! Broadcast mailbox AST routine
;  R0995  0        PBCB_L_BROADCAST_ARG   = 188, 0, 32, 0%,          ! Broadcast mailbox AST argument
;  R0996  0        PBCB_A_UNSOLICIT_RTN   = 192, 0, 32, 0%,          ! Unsolicited input mailbox AST routine
;  R0997  0        PBCB_L_UNSOLICIT_ARG   = 196, 0, 32, 0%,          ! Unsolicited input mailbox AST argument
;  R0998  0
;  R0999  0        PBCB_Q_BROADCAST_MSG_QUEUE = 200,0,0,0%,          ! Queue for holding broadcast messages
;  R1000  0          PBCB_L_SMGMBX_FLINK  = 200, 0, 32, 0%,          ! Forward link
;  R1001  0          PBCB_L_SMGMBX_BLINK  = 204, 0, 32, 0%,          ! Backward link
;  R1002  0
```

```
;  R1003  0          PBCB_W_FLAGS            = 208, 0, 16, 0%,   ! Flags
;  R1004  0            PBCB_V_BROADCAST      = 208, 0,  1, 0%,   ! Broadcast msg trapping enabled
;  R1005  0            PBCB_V_UNSOLICIT      = 208, 1,  1, 0%,   ! Unsolicited input notification enabled
;  R1006  0            PBCB_V_SMGMBX_INIT    = 208, 2,  1, 0%,   ! 1 tells AST routine this is
;  R1007  0                                                       ! an initialization call
;  R1008  0            PBCB_V_RMS            = 208, 3,  1, 0%,   ! 1 means using RMS for output
;  R1009  0            PBCB_V_LOCKED         = 208, 4,  1, 0%,   ! PBCB is locked
;  R1010  0            PBCB_V_REBUILD        = 208, 5,  1, 0%,   ! A rebuild is needed
;  R1011  0            PBCB_V_CONTROLO       = 208, 6,  1, 0%,   ! Previous output aborted by CTRL/O
;  R1012  0            PBCB_V_CANCEL_CONTROLO=208, 7,  1, 0%,   ! Cancel CTRL/O on next QIO
;  R1013  0            PBCB_V_BS             = 208, 8,  1, 0%,   ! 1 means terminal can do backspace
;  R1014  0            PBCB_V_COMPLEX_BORDER= 208, 9,  1, 0%,   ! 1 means some border capability
;  R1015  0                                                       ! is longer than a byte
;  R1016  0
;  R1017  0          PBCB_W_ASYNC_CHAN       = 210, 0, 16, 0%,   ! Asynchronous channel to terminnal
;  R1018  0          PBCB_W_MBX_CHAN         = 212, 0, 16, 0%,   ! Mailbox channel
;  R1019  0          PBCB_W_SMGMBX_BUFSIZ    = 214, 0, 16, 0%,   ! Max message size for mailbox
;  R1020  0          PBCB_A_SMGMBX_BUFFER    = 216, 0, 32, 0%,   ! Address of mailbox buffer
;  R1021  0          PBCB_Q_SMGMBX_IOSB      = 220, 0,  0, 0%,   ! I/O status block for mbx read
;  R1022  0          PBCB_W_OUTNAM_LEN       = 228, 0, 16, 0%,   ! Length of output name string
;  R1023  0          PBCB_W_ORIG_WIDTH       = 230, 0, 16, 0%,   ! Original width of terminal
;  R1024  0          PBCB_A_FAB              = 232, 0, 32, 0%,   ! Address of FAB (if file)
;  R1025  0          PBCB_A_RAB              = 236, 0, 32, 0%,   ! Address of RAB (if file)
;  R1026  0          PBCB_A_RBF              = 240, 0, 32, 0%,   ! Address of record buffer
;  R1027  0
;  R1028  0          PBCB_W_TOP_SCROLL_LINE = 244, 0, 16, 0%,   ! Top scroll line
;  R1029  0          PBCB_W_BOT_SCROLL_LINE = 246, 0, 16, 0%,   ! Bottom scroll line
;  R1030  0                                                       ! Above 2 words record
;  R1031  0                                                       ! where the physical
;  R1032  0                                                       ! scrolling region is
;  R1033  0                                                       ! currently set on the
;  R1034  0                                                       ! terminal.
;  R1035  0          PBCB_B_ORIG_HEIGHT      = 248, 0,  8, 0%,   ! Original number of rows on
;  R1036  0                                                       ! terminal (reserved for
;  R1037  0                                                       ! future use).
;  R1038  0          PBCB_B_BACKGROUND_COLOR= 249, 0,  8, 0%,   ! Background color
;  R1039  0          PBCB_W_INTERNAL_ATTR    = 250, 0, 16, 0%,   ! Internal attributes
;  R1040  0              PBCB_V_WIDE         = 250, 0,  1, 0%,   ! Pasteboard allows wide lines
;  R1041  0              PBCB_V_HIGH         = 250, 1,  1, 0%,   ! Pasteboard allows high wide lines
;  R1042  0              PBCB_V_TABS         = 250, 2,  1, 0%,   ! Pbd allows physical tabs
;  R1043  0          PBCB_L_TERMTABLE        = 252, 0, 32, 0%,   ! Corresponding TERMTABLE.
;  R1044  0          PBCB_L_LONGEST_SEQUENCE= 256, 0, 32, 0%,   ! Longest capability sequence
;  R1045  0          PBCB_A_CAP_BUFFER       = 260, 0, 32, 0%,   ! Address of capability buffer
;  R1046  0          PBCB_L_CAP_LENGTH       = 264, 0, 32, 0%,   ! Length of last capability gotten
;  R1047  0          PBCB_R_BORDER_VECTOR    = 268, 0,  0, 0%;   ! 16-longword border vector
;  R1048  0
;  R1049  0
;  R1050  0      LITERAL
;  R1051  0
;  R1052  0          PBCB_K_SIZE = 332;                  ! Total size of  PBCB in bytes.
;  R1053  0
;  R1054  0      MACRO
;  R1055  0
;  R1056  0          $PBCB_DECL = BLOCK[PBCB_K_SIZE,BYTE] %;
;  R1057  0
;  R1058  0      LITERAL ! masks for bits in field PBCB_L_MODE_SETTINGS
;  R1059  0
```

```
;  R1060  0          PBCB_M_BUF_ENABLED  =  1,    ! =1 if buffering enabled
;  R1061  0          PBCB_M_MINUPD       =  2,    ! =1 if minimal update enabled
;  R1062  0          PBCB_M_CLEAR_SCREEN =  4,    ! =1 if should clear screen on image exit
;  R1063  0          PBCB_M_NOTABS       =  8;    ! =1 if SMG should not use physical tabs
;  R1064  0
;  R1065  0      LITERAL
;  R1066  0
;  R1067  0          PBCB_K_DEF_MODE_SETTINGS =
;  R1068  0
;  R1069  0              PBCB_M_MINUPD ;          ! Minimum update enabled
```

```
R1070  0      !+
R1071  0      ! Window Control Block (WCB)
R1072  0      ! --------------------------
R1073  0      !       This data structure resides in heap storage.  There is
R1074  0      ! (currently) one WCB associated with each pasteboard and is pointed to
R1075  0      ! by it.
R1076  0      ! After an output operation, WCB_A_TEXT_BUF and WCB_A_SCR_TEXT_BUF have
R1077  0      ! their contents swapped.  At the same time, WCB_A_ATTR_BUF and
R1078  0      ! WCB_A_SCR_ATTR_BUF have their contents swapped.  Hence, at any point
R1079  0      ! in time, WCB_A_SCR_TEXT_BUF and WCB_A_SCR_ATTR_BUF record what is
R1080  0      ! currently on the screen.  WCB_A_TEXT_BUF and WCB_A_ATTR_BUF are used
R1081  0      ! to construct the next screen full.
R1082  0      !-
R1083  0      MACRO
R1084  0          WCB_Q_COORD         =  0, 0, 00, 0%,  ! Really 0, 0, 64, 0
R1085  0                      ! Quadword containing next four words.  The four fields
R1086  0                      ! define the coordinate system for the pasteboard and
R1087  0                      ! hence the window buffer.  Their address is passed to
R1088  0                      ! transmit these 4 fields as a single parameter.
R1089  0
R1090  0              WCB_W_ROW_START =  0, 0, 16, 0%,  ! Row number of 1st row (=1)
R1091  0              WCB_W_NO_ROWS   =  2, 0, 16, 0%,  ! Number of rows
R1092  0              WCB_W_COL_START =  4, 0, 16, 0%,  ! Col number of 1st col (=1)
R1093  0              WCB_W_NO_COLS   =  6, 0, 16, 0%,  ! Number of cols
R1094  0
R1095  0
R1096  0
R1097  0          WCB_A_TEXT_BUF      =  8, 0, 32, 0%,  ! Address of a text buffer
R1098  0                                                ! for this window.
R1099  0
R1100  0          WCB_A_ATTR_BUF      = 12, 0, 32, 0%,  ! Address of attribute buffer
R1101  0                                                ! for this window.
R1102  0
R1103  0          WCB_A_CHAR_SET_BUF  = 16, 0, 32, 0%,  ! Address of character set
R1104  0                                                ! buffer for this window.
R1105  0                                                ! This buffer is allocated
R1106  0                                                ! only if needed.
R1107  0
R1108  0          WCB_A_SCR_TEXT_BUF  = 20, 0, 32, 0%,  ! Address of text buffer
R1109  0                                                ! representing what is
R1110  0                                                ! currently on the screen.
R1111  0
R1112  0          WCB_A_SCR_ATTR_BUF  = 24, 0, 32, 0%,  ! Address of attribute buffer
R1113  0                                                ! associated with
R1114  0                                                ! WCB_A_SCR_TEXT_BUF
R1115  0
R1116  0          WCB_A_SCR_CHAR_SET_BUF=28, 0, 32, 0%, ! Address of character set
R1117  0                                                ! buffer associated with
R1118  0                                                ! WCB_A_SCH_TEXT_BUF.
R1119  0                                                ! Allocated only if needed.
R1120  0
R1121  0          WCB_W_CURR_CUR_ROW  = 32, 0, 16, 1%,
R1122  0          WCB_W_CURR_CUR_COL  = 34, 0, 16, 1%,
R1123  0                                                ! Cursor position in
R1124  0                                                ! WCB_A_TEXT_BUF.
R1125  0          WCB_W_OLD_CUR_ROW   = 36, 0, 16, 1%,
R1126  0          WCB_W_OLD_CUR_COL   = 38, 0, 16, 1%,
```

```
;  R1127  0                                          ! Cursor position in
;  R1128  0                                          ! WCB_A_SCR_TEXT_BUF.
;  R1129  0
;  R1130  0          WCB_L_BUFSIZE       = 40, 0, 32, 0%,  ! = .WCB [WCB_W_NO_ROWS] *
;  R1131  0                                          !   .WCB [WCB_W_NO_COLS]
;  R1132  0
;  R1133  0          WCB_A_LINE_CHAR     = 44, 0, 32, 0%,  ! Address of line
;  R1134  0                                          ! characteristics vector for
;  R1135  0                                          ! text buffer.
;  R1136  0                                          ! This vector, one byte for
;  R1137  0                                          ! each line, records whether
;  R1138  0                                          ! the line is Single, Double-
;  R1139  0                                          ! High, Double-Wide, etc.
;  R1140  0                                          ! This vector is allocated to
;  R1141  0                                          ! be WCB_W_NO_ROWS + 1 bytes
;  R1142  0                                          ! long so it can be indexed
;  R1143  0                                          ! directly by row number
;  R1144  0                                          ! (1 through WCB_W_NO_ROWS).
;  R1145  0
;  R1146  0          WCB_A_SCR_LINE_CHAR = 48, 0, 32, 0%;  ! Address of line
;  R1147  0                                          ! characteristics vector for
;  R1148  0                                          ! screen text buffer.
;  R1149  0                                          ! This vector, one byte for
;  R1150  0                                          ! each line, records whether
;  R1151  0                                          ! the line is Single, Double-
;  R1152  0                                          ! High, Double-Wide, etc.
;  R1153  0                                          ! This vector is allocated to
;  R1154  0                                          ! be WCB_W_NO_ROWS + 1 bytes
;  R1155  0                                          ! long so it can be indexed
;  R1156  0                                          ! directly by row number
;  R1157  0                                          ! (1 through WCB_W_NO_ROWS).
;  R1158  0
;  R1159  0      LITERAL
;  R1160  0          WCB_K_SIZE = 52;     ! No. of bytes in a WCB
;  R1161  0
;  R1162  0      MACRO
;  R1163  0
;  R1164  0          $WCB_DECL = BLOCK[WCB_K_SIZE,BYTE] %;
;  R1165  0
```

```
;  R1166  0      !+
;  R1167  0      ! Pasting Packet (PP`
;  R1168  0      !       This data str .ture defines the layout of a Pasting Packet.
;  R1169  0      ! this area is allocated in heap storage.  One such structure exists for
;  R1170  0      ! every pasting of a virtual display to a pasteboard.  It exists
;  R1171  0      ! simultaneously on two queues -- the queue (DCB_A_PP_NEXT) headed in
;  R1172  0      ! the DCB which contains the queue of PP's representing to which
;  R1173  0      ! pasteboards this DCB is pasted, and at the same time, it is a member
;  R1174  0      ! of the queue (PBCB_A_PP_NEXT) headed in the PBCB which contains the
;  R1175  0      ! queue of PP's representing all virtual displays pasted to this
;  R1176  0      ! pasteboard.
;  R1177  0      !-
;  R1178  0      MACRO
;  R1179  0          PP_A_NEXT_DCB       =       0, 0, 32, 0%,
;  R1180  0          PP_A_PREV_DCB       =       4, 0, 32, 0%,
;  R1181  0                                      ! The previous 2 longwords serve as
;  R1182  0                                      ! a queue entry for purposes of enqueing
;  R1183  0                                      ! onto queue DCB_A_PP_NEXT -- the queue
;  R1184  0                                      ! of all pasteboards to which this
;  R1185  0                                      ! virtual display is pasted.
;  R1186  0
;  R1187  0          PP_A_NEXT_PBCB      =       8, 0, 32, 0%,
;  R1188  0          PP_A_PREV_PBCB      =      12, 0, 32, 0%,
;  R1189  0                                      ! The previous 2 longwords serve as
;  R1190  0                                      ! a queue entry for purposes of enqueing
;  R1191  0                                      ! onto queue PBCB_A_PP_NEXT -- the queue
;  R1192  0                                      ! of all virtual displays which are
;  R1193  0                                      ! pasted to this pasteboard.
;  R1194  0
;  R1195  0          PP_A_DCB_ADDR       =      16, 0, 32, 0%,
;  R1196  0                                      ! Address of the DCB involved in this
;  R1197  0                                      ! pasting.
;  R1198  0
;  R1199  0          PP_A_PBCB_ADDR      =      20, 0, 32, 0%,
;  R1200  0                                      ! Address of the PBCB involved in this
;  R1201                                         ! pasting.
;  R1202  0
;  R1203  0          PP_W_ROW            =      24, 0, 16, 1%,
;  R1204  0                                      ! The row number of the pasteboard onto
;  R1205  0                                      ! which row 1 of the virtual display
;  R1206  0                                      ! maps.
;  R1207  0
;  R1208  0          PP_W_COL            =      26, 0, 16, 1%,
;  R1209  0                                      ! The column number of the pasteboard
;  R1210  0                                      ! onto whi.h column 1 of the virtual
;  R1211  0                                      ! display maps.
;  R1212  0
;  R1213  0          PP_W_ROWS_TO_MOVE   =      28, 0, 16, 0%,
;  R1214  0                                      ! The number of rows which have to be
;  R1215  0                                      ! moved from the display buffer to the
;  R1216  0                                      ! window buffer.  If zero, the next 3
;  R1217  0                                      ! fields are meaningless.
;  R1218  0
;  R1219  0          PP_W_FROM_INDEX     =      30, 0, 16, 0%,
;  R1220  0                                      ! The byte index beyond the begining of
;  R1221  0                                      ! a source buffer which represents the
;  R1222  0                                      ! 1st byte position to be moved when
```

```
R1223  0                                    ! copying from display buffers to
R1224  0                                    ! window buffers.
R1225  0
R1226  0        PP_W_TO_INDEX       =       32, 0, 16, 0%,
R1227  0                                    ! The byte index beyond the begining of
R1228  0                                    ! a destination buffer which represents
R1229  0                                    ! the 1st byte to be deposited when
R1230  0                                    ! copying from a display buffers to
R1231  0                                    ! window buffers.
R1232  0
R1233  0        PP_W_MOVE_LENGTH    =       34, 0, 16, 0%,
R1234  0                                    ! The number of columns to be moved from
R1235  0                                    ! display buffers to window buffers.
R1236  0
R1237  0        PP_W_LABEL_BYTES_TO_MOVE =  36, 0, 16, 0%,
R1238  0                                    ! The number of bytes of the border
R1239  0                                    ! label which need to be moved to the
R1240  0                                    ! WCB text buffer when the virtual
R1241  0                                    ! display is mapped to the WCB text
R1242  0                                    ! buffer.  This field may be zero if no
R1243  0                                    ! characters fit the way the display is
R1244  0                                    ! pasted.
R1245  0
R1246  0        PP_W_SRC_LABEL_OFF      =   38, 0, 16, 0%,
R1247  0                                    ! The offset (0-based) beyond the
R1248  0                                    ! beginning of the the label string
R1249  0                                    ! which represents the 1st byte of the
R1250  0                                    ! label to be moved.
R1251  0                                    ! (= length_of_label_string -
R1252  0                                    !   .PP [PP_W_LABEL_BYTES_TO_MOVE]).
R1253  0                                    ! This field may not be valid if
R1254  0                                    ! .PP [PP_W_LABEL_BYTES_TO_MOVE] is 0.
R1255  0
R1256  0        PP_W_DST_LABEL_OFF      =   40, 0, 16, 0%,
R1257  0                                    ! The offset (0-based) beyond the
R1258  0                                    ! beginning of .WCB [WCB_A_TEXT_BUF]
R1259  0                                    ! where the first byte of the visible
R1260  0                                    ! label lands.  This same offset is used
R1261  0                                    ! to reach the appropriate attribute
R1262  0                                    ! byte in WCB [WCB_A_ATTR_BUF].  This
R1263  0                                    ! field may not be valid if
R1264  0                                    ! .PP [PP_W_LABEL_BYTES_TO_MOVE] is 0.
R1265  0
R1266  0                    ! NOTE:  The 7 fields above are computed as a function
R1267  0                    ! of both the dimensions of the virtual display and
R1268  0                    ! the position on the pasteboard on which it is pasted.
R1269  0                    ! If either ( dimensions or pasting position) changes,
R1270  0                    ! these fields must be recomputed.
R1271  0                    ! They are initially set up as a result of pasting.
R1272  0                    ! Having these fields precomputed in the PP makes the
R1273  0                    ! output operation of moving data from display buffer
R1274  0                    ! to window buffer run faster since these values do not
R1275  0                    ! have to be continually recomputed on the fly.
R1276  0
R1277  0        PP_B_CONTROL_BITS       =   42, 0,  8, 0%,
R1278  0                                    ! Home for various PP-wide status bits.
R1279  0            PP_V_OCCLUDED       =   42, 0,  1, 0%,
```

```
;  R1280  0                                ! Records whether the pasting of this
;  R1281  0                                ! virtual display to this pasteboard --
;  R1282  0                                ! taking into consideration all other
;  R1283  0                                ! pasting to this pasteboard -- is
;  R1284  0                                ! occluded or not.
;  R1285  0                                ! 1 => Occluded, 0=> Not occluded
;  R1286  0
;  R1287  0            PP_V_CONTIG         =  42, 1,  1, 0%,
;  R1288  0                                ! If set, this bit means that the
;  R1289  0                                ! virtual display (as pasted) can be
;  R1290  0                                ! moved to the window buffer via a
;  R1291  0                                ! single CH$MOVE-- i.e. the source
;  R1292  0                                ! and destination fields are contiguous
;  R1293  0                                ! bytes.
;  R1294  0                                ! If not set, text must be moved on a
;  R1295  0                                ! row by row basis since only the bytes
;  R1296  0                                ! within a row are contiguous.
;  R1297  0
;  R1298  0        PP_L_MOVE_SIZE          =  43, 0, 32, 0%,
;  R1299  0                                !  = .PP [PP_W_ROWS_TO_MOVE] *
;  R1300  0                                !    .PP [PP_W_MOVE_LENGTH]
;  R1301  0
;  R1302  0                                ! Next 4 fields tell where on the WCB
;  R1303  0                                ! buffer the part of the virtual display
;  R1304  0                                ! that fits within the pasteboard
;  R1305  0                                ! projects.  I.e., what area of the
;  R1306  0                                ! WCB buffers get modified when this
;  R1307  0                                ! virtual display is mapped to the
;  R1308  0                                ! pasteboard.  These fields are not
;  R1309  0                                ! meaningful if PP_W_ROWS_TO_MOVE is
;  R1310  0                                ! zero since it then doesn't even hit
;  R1311  0                                ! the pasteboard.
;  R1312  0                                ! Note: these fields do not take into
;  R1313  0                                ! account whether the virtual display
;  R1314  0                                ! is bordered.  For most instances this
;  R1315  0                                ! is the right thing to do.  There is
;  R1316  0                                ! one known quirk that needs to be fixed
;  R1317  0                                ! later-- If a virtual display is pasted
;  R1318  0                                ! to a row or column which is one unit
;  R1319  0                                ! outside of the pasteboard boundaries,
;  R1320  0                                ! then its PP_W_ROWS_TO_MOVE will be 0
;  R1321  0                                ! since the display itself will not
;  R1322  0                                ! project onto the pasteboard -- but its
;  R1323  0                                ! border will !!!
;  R1324  0            PP_W_FIRST_WCB_ROW      = 47, 0, 16, 0%,
;  R1325  0            PP_W_LAST_WCB_ROW       = 49, 0, 16, 0%,
;  R1326  0            PP_W_FIRST_WCB_COL      = 51, 0, 16, 0%,
;  R1327  0            PP_W_LAST_WCB_COL       = 53, 0, 16, 0%;
;  R1328  0
;  R1329  0        LITERAL
;  R1330  0            PP_PBCB_QUEUE_OFFSET = 8,   ! Offset of the queue header for the
;  R1331  0                                        ! pasteboard side of the chain.  This is
;  R1332  0                                        ! the byte offset of the
;  R1333  0                                        ! PBCB_A_NEXT_PBCB field.
;  R1334  0
;  R1335  0            PP_K_SIZE           = 55;   ! Size in bytes of a PP
;  R1336  0
```

```
;  R1337  0      MACRO
;  R1338  0
;  R1339  0          $PP_DECL = BLOCK[PP_K_SIZE,BYTE] %;
;  R1340  0
```

```
;  R1341  0      !+
;  R1342  0      ! The following are masks of bit positions in the bytes of the
;  R1343  0      ! attribute array pointed to by WCB [WCB_A_ATTR_BUF].
;  R1344  0
;  R1345  0      LITERAL
;  R1346  0          ATTR_M_REND_BOLD    = %X'01',       ! Bold rendition
;  R1347  0          ATTR_M_REND_REV     = %X'02',       ! Reverse video rendition
;  R1348  0          ATTR_M_REND_BLINK   = %X'04',       ! Blink rendition
;  R1349  0          ATTR_M_REND_UNDER   = %X'08',       ! Underline rendition
;  R1350  0
;  R1351  0          ATTR_M_REND_GRAPHIC = %X'10',       ! Line-drawing character set
;  R1352  0                      ! If set, this bit indicates that this character must
;  R1353  0                      ! be rendered using the device's line-drawing character
;  R1354  0                      ! set.
;  R1355  0
;  R1356  0          ATTR_M_USER_GRAPHIC = %X'40',       ! User line-drawing char set
;  R1357  0                      ! This indicates a generic line-drawing character which
;  R1358  0                      ! must be converted to the device-specific character
;  R1359  0                      ! before being output to the screen.
;  R1360  0
;  R1361  0          ATTR_M_BORD_ELEM    = %X'80';       ! Border control bit.
;  R1362  0                      ! This bit is used to record that the associated text
;  R1363  0                      ! byte is (was) not a printable text byte, but a
;  R1364  0                      ! component of a border element.  This bit is not
;  R1365  0                      ! supplied by the caller to SMG, but is established
;  R1366  0                      ! internally while virtual displays are being mapped to
;  R1367  0                      ! the pasteboard buffer.  It is not of interest to the
;  R1368  0                      ! output routines.
;  R1369  0                      ! During the mapping phase, while the various virtual
;  R1370  0                      ! displays are mapped onto the output window buffer,
;  R1371  0                      ! this bit is used to record the fact that an element of
;  R1372  0                      ! a border occupies the corresponding cell in the WCB
;  R1373  0                      ! text buffer.  It is used to distinqiush a normal ASCII
;  R1374  0                      ! text character from an encoding of what pieces of a
;  R1375  0                      ! border element must occupy this text slot.
;  R1376  0                      ! After all virtual displays have been mapped to the
;  R1377  0                      ! WCB buffers and all intersections of border characters
;  R1378  0                      ! have been resolved,  each byte in the attribute array
;  R1379  0                      ! is inspected to see if it constains this bit.  If the
;  R1380  0                      ! bit in the attribute byte is set, the bits in the text
;  R1381  0                      ! byte are inspected to see how they should be  endered.
;  R1382  0                      ! If the device associated with the pasteboard does not
;  R1383  0                      ! support a line-drawing character set, the bits in the
;  R1384  0                      ! text byte are changed to the closest ASCII character
;  R1385  0                      ! approximation -- "+", "-", or "|".
;  R1386  0                      ! This cellular position can now be treated like any
;  R1387  0                      ! other text postion.
;  R1388  0                      ! However, if the associated device does support a line-
;  R1389  0                      ! drawing character set, the bits in the text cell are
;  R1390  0                      ! changed to the appropriate graphic code for that line-
;  R1391  0                      ! drawing set and the ATTR_M_GRAPHIC bit in the
;  R1392  0                      ! attribute array byte is turned on.
;  R1393  0                      ! The bits (in the text buffer) that encode the desired
;  R1394  0                      ! graphic are given below.
;  R1395  0
;  R1396  0      ! These masks are used to set and reset these bits, e.g.
;  R1397  0
```

```
R1398  0      !       byte_in_attr_array = .byte_in_attr_arry OR ATTR_M_????  ! Set
R1399  0      !       byte_in_attr_array = .byte_in_attr_arry XOR ATTR_M_???? ! Reset
R1400  0      !
R1401  0
R1402  0      ! The following are the corresponding constants for accessing the
R1403  0      ! bits for interrogation purposes.
R1404  0      LITERAL
R1405  0          ATTR_V_REND_BOLD    = %X'00',       ! Bold
R1406  0          ATTR_V_REND_REV     = %X'01',       ! Reverse video
R1407  0          ATTR_V_REND_BLINK   = %X'02',       ! Blink
R1408  0          ATTR_V_REND_UNDER   = %X'03',       ! Underline
R1409  0
R1410  0          ATTR_V_REND_GRAPHIC = %X'04',       ! Graphic character
R1411  0          ATTR_V_USER_GRAPHIC = %X'06',       ! User graphic character
R1412  0          ATTR_V_BORD_ELEM    = %X'07';       ! Border element control bit
R1413  0
R1414  0      !       These bits are used in BLISS constructs like:
R1415  0      !       IF .(some_place_in_the_attribute_buffer)<ATTR_V_???,1> ! If set
R1416  0
R1417  0      ! These constants are used to check the line characteristics vector.
R1418  0      ! The line characteristics vector is used to specify double wide and
R1419  0      ! double high/double wide.
R1420  0
R1421  0      LITERAL
R1422  0
R1423  0          LINE_K_NORMAL       = 0,            ! single width and height (must be 0)
R1424  0          LINE_K_WIDE         = 1,            ! double wide
R1425  0          LINE_K_UPPER_HIGH   = 2,            ! upper half of double high
R1426  0          LINE_K_LOWER_HIGH   = 3;            ! lower half of double high
```

```
;  R1427  0      !+
;  R1428  0      ! The following are bit definitions of bits found in the text buffer
;  R1429  0      ! pointed to by WCB [WCB_A_TEXT_BUF] when a particular cell contains
;  R1430  0      ! not a printable character, but an encoding of what parts of a border
;  R1431  0      ! character need to be placed in this position.
;  R1432  0      !-
;  R1433  0
;  R1434  0      LITERAL
;  R1435  0          BORD_M_RIGHT        = %X'01',
;  R1436  0          BORD_M_UP           = %X'02',
;  R1437  0          BORD_M_LEFT         = %X'04',
;  R1438  0          BORD_M_DOWN         = %X'08',
;  R1439  0
;  R1440  0          BORD_M_HORIZ        = BORD_M_RIGHT + BORD_M_LEFT,
;  R1441  0          BORD_M_VERT         = BORD_M_UP    + BORD_M_DOWN,
;  R1442  0
;  R1443  0          BORD_M_ULCORN       = BORD_M_DOWN  + BORD_M_RIGHT,
;  R1444  0          BORD_M_URCORN       = BORD_M_DOWN  + BORD_M_LEFT,
;  R1445  0          BORD_M_LLCORN       = BORD_M_UP    + BORD_M_RIGHT,
;  R1446  0          BORD_M_LRCORN       = BORD_M_UP    + BORD_M_LEFT;
;  R1447  0
;  R1448  0      ! Certain combinations of the above bit patterns are meaningful, e.g.
;  R1449  0      ! BORD_M_VERT + BORD_M_RIGHT represent a "right-T" graphic
;  R1450  0      !
;  R1451  0      ! These bits are used to "OR" together the right total graphic that is
;  R1452  0      ! needed at a particular position on the screen to represent some
;  R1453  0      ! element of a border.
;  R1454  0
;  R1455  0
;  R1456  0
;  R1457  0      ! The corresponding bit positions:
;  R1458  0      !
;  R1459  0      LITERAL
;  R1460  0          BORD_V_RIGHT        = %X'00',
;  R1461  0          BORD_V_UP           = %X'01',
;  R1462  0          BORD_V_LEFT         = %X'02',
;  R1463  0          BORD_V_DOWN         = %X'03';
;  R1464  0
```

```
;   1465  0     REQUIRE 'RTLIN:SMGMACROS';
```

```
;  R1466  0      ! Macro Definitions for RTL SMG$ facility
;  R1467  0      ! File:  SMGMACROS.REQ  Edit: STAN1012
;  R1468  0      !
;  R1469  0      !
;  R1470  0      !*****************************************************************************
;  R1471  0      !*                                                                           *
;  R1472  0      !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;  R1473  0      !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;  R1474  0      !*  ALL RIGHTS RESERVED.                                                     *
;  R1475  0      !*                                                                           *
;  R1476  0      !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
;  R1477  0      !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
;  R1478  0      !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
;  R1479  0      !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
;  R1480  0      !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
;  R1481  0      !*  TRANSFERRED.                                                             *
;  R1482  0      !*                                                                           *
;  R1483  0      !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
;  R1484  0      !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
;  R1485  0      !*  CORPORATION.                                                             *
;  R1486  0      !*                                                                           *
;  R1487  0      !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
;  R1488  0      !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;  R1489  0      !*                                                                           *
;  R1490  0      !*                                                                           *
;  R1491  0      !*****************************************************************************
;  R1492  0      !
```

```
;  R1493  0      !++
;  R1494  0      ! FACILITY:     Screen Management
;  R1495  0      !
;  R1496  0      ! ABSTRACT:
;  R1497  0      !
;  R1498  0      !       This file contains macros used by screen management routines.
;  R1499  0      !
;  R1500  0      ! MODIFIED BY:
;  R1501  0      !
;  R1502  0      ! 1-001 - Original.  PLL 15-Mar-1983
;  R1503  0      ! 1-002 - Add $SMG$VALIDATE_ARGCOUNT.  PLL 24-Mar-1983
;  R1504  0      ! 1-003 - Add $SMG$VALIDATE_ROW_COL.  PLL 7-Apr-1983
;  R1505  0      ! 1-004 - Add lots more.  PLL 14-Apr-1983
;  R1506  0      ! 1-005 - Add $SMG$FIND_PRINT_LENGTH.  PLL 21-Apr-1983
;  R1507  0      ! 1-006 - Fix $SMG_GET_PBCB so that it can work in any module.
;  R1508  0      !         STAN 3-May-1983
;  R1509  0      ! 1-007 - Fix $SMG$FIND_PRINT_LENGTH.  PLL 12-May-1983
;  R1510  0      ! 1-008 - Delete $SMG$CHECK_FOR_WRAP since it's not used.
;  R1511  0      !         PLL 18-May-1983
;  R1512  0      ! 1-009 - Change to make display id's really be DCB addresses.
;  R1513  0      !         Deleted $SMG$GET_NEXT_DID -- no longer needed.
;  R1514  0      !         Changed $SMG$GET_DCB.
;  R1515  0      !         RKR 20-May-1983.
;  R1516  0      ! 1-010 - Add input macros.  (Formerly in SMGLIB.REQ.) PLL 21-Jun-1983
;  R1517  0      ! 1-011 - Fix SET_REND_CODE. STAN 23-Jun-1983.
;  R1518  0      ! 1-012 - Add $SMG$GET_TERM_DATA. STAN 15-Jan-1984.
;  R1519  0      !--
```

```
;  R1520  0      !+
;  R1521  0      ! $SMG$VALIDATE_ARGCOUNT
;  R1522  0      ! ----------------------
;  R1523  0      !
;  R1524  0      ! Macro used to check that a SMG$ procedure was called with the correct
;  R1525  0      ! number of arguments.  If the test fails, the procedure returns with the
;  R1526  0      ! failure status SMG$_WRONUMARG.
;  R1527  0      !
;  R1528  0      ! Format:
;  R1529  0      !       $SMG$VALIDATE_ARGCOUNT (lo, hi);
;  R1530  0      !
;  R1531  0      !       lo = Lowest number of arguments which are valid (0-255)
;  R1532  0      !       hi = Highest number of arguments which are valid (0-255)
;  R1533  0      !-
;  R1534  0
;  R1535  0      MACRO
; MR1536  0          $SMG$VALIDATE_ARGCOUNT (lo, hi) =
; MR1537  0              BEGIN
; MR1538  0              BUILTIN
; MR1539  0                  ACTUALCOUNT;
; MR1540  0              EXTERNAL LITERAL
; MR1541  0                  SMG$_WRONUMARG;
; MR1542  0
; MR1543  0              %IF lo NEQ hi
; MR1544  0              %THEN
; MR1545  0                  %IF lo NEQ 0
; MR1546  0                  %THEN
; MR1547  0                      LOCAL
; MR1548  0                          DIFF: BYTE;
; MR1549  0                      DIFF = ACTUALCOUNT () - lo;
; MR1550  0                      IF .DIFF  GTRU (hi - lo)
; MR1551  0                      THEN
; MR1552  0                          RETURN SMG$_WRONUMARG;
; MR1553  0                  %ELSE
; MR1554  0                      IF ACTUALCOUNT () GTRU hi
; MR1555  0                      THEN
; MR1556  0                          RETURN SMG$_WRONUMARG;
; MR1557  0                  %FI
; MR1558  0
; MR1559  0              %ELSE
; MR1560  0                  IF ACTUALCOUNT () NEQU lo
; MR1561  0                  THEN
; MR1562  0                      RETURN SMG$_WRONUMARG;
; MR1563  0              %FI
; MR1564  0
;  R1565  0              END %;
;  R1566  0
;  R1567  0      !+
;  R1568  0      ! $SMG$VALIDATE_ROW_COL
;  R1569  0      ! ---------------------
;  R1570  0      ! This macro checks to make sure the specified row and column are
;  R1571  0      ! within the virtual display.  SMG$_INVROW or SMG$_INVCOL is returned
;  R1572  0      ! if they are not.
;  R1573  0      !-
; MR1574  0      MACRO $SMG$VALIDATE_ROW_COL (ROW, COL) =
; MR1575  0          BEGIN
; MR1576  0          EXTERNAL LITERAL SMG$_INVROW;
```

```
;MR1577  0          EXTERNAL LITERAL SMG$_INVCOL;
;MR1578  0          IF ROW LEQ 0 OR
;MR1579  0             ROW GTR .DCB [DCB_W_NO_ROWS]
;MR1580  0          THEN
;MR1581  0              RETURN (SMG$_INVROW);
;MR1582  0
;MR1583  0          IF COL LEQ 0 OR
;MR1584  0             COL GTR .DCB [DCB_W_NO_COLS]
;MR1585  0          THEN
;MR1586  0              RETURN (SMG$_INVCOL);
;MR1587  0
; R1588  0          END; %;
; R1589  0
; R1590  0
; R1591  0      !+
; R1592  0      ! $SMG$GET_DCB
; R1593  0      ! ------------
; R1594  0      ! Macro $SMG$GET_DCB validates the suppled virtual display id (DID) and
; R1595  0      ! computes the starting address of the corresponding Display Control
; R1596  0      ! Block (DCB).   If DID is invalid, SMG$_INVDIS_ID is returned to caller
; R1597  0      ! of routine that invokes this macro.
; R1598  0      !-
; R1599  0      MACRO
;MR1600  0          $SMG$GET_DCB ( DID, DCB_ADDR) =
;MR1601  0          BEGIN
;MR1602  0          BIND LOC_DID = DID : REF BLOCK [,BYTE];
;MR1603  0          EXTERNAL LITERAL SMG$_INVDIS_ID;
;MR1604  0
;MR1605  0          IF .LOC_DID [DCB_L_DID] NEQ .DID
;MR1606  0          THEN
;MR1607  0              RETURN (SMG$_INVDIS_ID);        ! Not pointing to one of our
;MR1608  0                                              ! control blocks
;MR1609  0
;MR1610  0          IF .LOC_DID [DCB_B_STRUCT_TYPE] NEQ DCB_K_STRUCT_TYPE
;MR1611  0          THEN
;MR1612  0              RETURN (SMG$_INVDIS_ID);        ! Not pointing to a DCB
;MR1613  0
;MR1614  0          DCB_ADDR = .DID;                    ! Assume ok
; R1615  0          END; %;
; R1616  0
; R1617  0      !+
; R1618  0      ! $SMG$GET_PBCB
; R1619  0      ! -------------
; R1620  0      ! Macro $SMG$GET_PBCB validates the suppled pasteboard id (PID) and
; R1621  0      ! computes the starting address of the corresponding Pasteboard Control
; R1622  0      ! Block (PBCB).   If PID is invalid, SMG$_INVPAS_ID is returned to
; R1623  0      ! caller of routine that invokes this macro.
; R1624  0      !-
; R1625  0      MACRO
;MR1626  0          $SMG$GET_PBCB ( PID, PBCB_ADDR) =
;MR1627  0          BEGIN
;MR1628  0          EXTERNAL LITERAL SMG$_INVPAS_ID;
;MR1629  0
;MR1630  0          %IF NOT %DECLARED ( PBD_L_COUNT )
;MR1631  0          %THEN
;MR1632  0              EXTERNAL PBD_L_COUNT,
;MR1633  0                       PBD_A_PBCB     : VECTOR,
```

```
; MR1634  0                       PBD_V_PB_AVAIL : BITVECTOR;
; MR1635  0           %FI
; MR1636  0
; MR1637  0           IF .PID LSS 0                 OR      ! Too small
; MR1638  0              .PID GTR .PBD_L_COUNT      OR      ! Too big
; MR1639  0              NOT .PBD_V_PB_AVAIL [.PID]         ! Not allocated
; MR1640  0           THEN
; MR1641  0               RETURN (SMG$_INVPAS_ID);
; MR1642  0
; MR1643  0           PBCB_ADDR = .PBD_A_PBCB [ .PID];
;  R1644  0           END; %;
;  R1645  0
;  R1646  0       !+
;  R1647  0       ! $SMG$GET_NEXT_PID
;  R1648  0       ! ----------------
;  R1649  0       ! Macro $SMG$GET_NEXT_PID attempts to allocate the next available pasteboard
;  R1650  0       ! id (PID).  If no more can be allocated, it returns SMG$_TOOMANPAS to
;  R1651  0       ! the caller of the routine that invoked this macro.
;  R1652  0       ! If a new one can be allocated, it is returned as PID and the
;  R1653  0       ! corresponding bit in PBD_V_PB_AVAIL is set to one to indicate that
;  R1654  0       ! this number is in use.
;  R1655  0       !-
;  R1656  0       MACRO
; MR1657  0           $SMG$GET_NEXT_PID (PID)=
; MR1658  0           BEGIN
; MR1659  0           BUILTIN
; MR1660  0               FFC;
; MR1661  0           EXTERNAL LITERAL SMG$_TOOMANPAS;
; MR1662  0
; MR1663  0           IF .PBD_L_COUNT + 1 GTR PBD_K_MAX_PB
; MR1664  0           THEN
; MR1665  0               RETURN (SMG$_TOOMANPAS);
; MR1666  0
; MR1667  0           !+
; MR1668  0           ! If we've done our bookkeeping correctly, the FFC should find a
; MR1669  0           ! zero within PBD_K_MAX_PB bits.
; MR1670  0           !-
; MR1671  0           FFC ( ZERO,                   ! Starting bit position
; MR1672  0                 PBD_K_MAX_PB_BY_REF,    ! Number of bits to search
; MR1673  0                 PBD_V_PB_AVAIL,         ! Base of search
; MR1674  0                 PID);                   ! Position of 1st zero bit found
; MR1675  0
; MR1676  0           PBD_V_PB_AVAIL [ .PID] = 1; ! Mark as allocated
;  R1677  0           END; %;
;  R1678  0
;  R1679  0
;  R1680  0       !+
;  R1681  0       ! $SMG$GET_TERM_DATA
;  R1682  0       ! ------------------
;  R1683  0       ! Gets terminal data from a terminal table into the PBCB.
;  R1684  0       ! Assumes you have a symbol named "PBCB".
;  R1685  0       !-
;  R1686  0       MACRO
;  R1687  0
; MR1688  0           $SMG$GET_TERM_DATA(CAPABILITY,ARG1,ARG2) =
; MR1689  0
; MR1690  0               BEGIN
```

```
;MR1691  0
;MR1692  0               LOCAL
;MR1693  0
;MR1694  0                       INPUT_ARGS      : VECTOR[3],
;MR1695  0                       STATUS;
;MR1696  0
;MR1697  0               EXTERNAL ROUTINE
;MR1698  0
;MR1699  0                       SMG$GET_TERM_DATA;
;MR1700  0
;MR1701  0               IF .PBCB[PBCB_L_TERMTABLE] EQL 0
;MR1702  0                 THEN  PBCB[PBCB_L_CAP_LENGTH]=0
;MR1703  0                 ELSE  BEGIN
;MR1704  0                       INPUT_ARGS[0]=%LENGTH-1;
;MR1705  0                       %IF NOT %NULL(ARG1)
;MR1706  0                               %THEN   INPUT_ARGS[1]=ARG1
;MR1707  0                       %FI;
;MR1708  0                       %IF NOT %NULL(ARG2)
;MR1709  0                               %THEN   INPUT_ARGS[2]=ARG2
;MR1710  0                       %FI;
;MR1711  0                       STATUS=SMG$GET_TERM_DATA(PBCB[PBCB_L_TERMTABLE],
;MR1712  0                                                %REF( %NAME(SMG$K_,CAPABILITY) ),
;MR1713  0                                                PBCB[PBCB_L_LONGEST_SEQUENCE],
;MR1714  0                                                PBCB[PBCB_L_CAP_LENGTH],
;MR1715  0                                                .PBCB[PBCB_A_CAP_BUFFER],
;MR1716  0                                                INPUT_ARGS);
;MR1717  0
;MR1718  0                       IF NOT .STATUS THEN RETURN .STATUS
;MR1719  0                       END
;MR1720  0
;MR1721  0               END
;MR1722  0
;  R1723  0       %;
```

```
;  R1724  0      !+
;  R1725  0      ! Macros for manipulation queue entries
;  R1726  0      !-
;  R1727  0
;  R1728  0      !+
;  R1729  0      ! $SMG$INSET_AT_HEAD
;  R1730  0      ! ------------------
;  R1731  0      ! Macro $SMG$INSERT_AT_HEAD inserts the specified queue entry in the
;  R1732  0      ! queue at a postion now occupied by the entry pointed to by the
;  R1733  0      ! forward pointer part of the queue header.
;  R1734  0      !-
;  R1735  0      MACRO
; MR1736  0          $SMG$INSERT_AT_HEAD ( Q_ENTRY, Q_HEAD) =
; MR1737  0          BEGIN
; MR1738  0              BUILTIN INSQUE;
; MR1739  0              INSQUE ( Q_ENTRY, Q_HEAD);
;  R1740  0          END; %;
;  R1741  0
;  R1742  0      !+
;  R1743  0      ! $SMG$INSET_AT_TAIL
;  R1744  0      ! ------------------
;  R1745  0      ! Macro $SMG$INSERT_AT_TAIL inserts the specified queue entry in the
;  R1746  0      ! queue at a postion now occupied by the entry pointed to by the
;  R1747  0      ! backward pointer part of the queue header.
;  R1748  0      !-
;  R1749  0      MACRO
; MR1750  0          $SMG$INSERT_AT_TAIL ( Q_ENTRY, Q_HEAD) =
; MR1751  0          BEGIN
; MR1752  0              BUILTIN INSQUE;
; MR1753  0              INSQUE ( Q_ENTRY, .(Q_HEAD +4));
;  R1754  0          END; %;
;  R1755  0
;  R1756  0      !+
;  R1757  0      ! $SMG$REMOVE_FROM_QUEUE
;  R1758  0      ! ----------------------
;  R1759  0      ! Macro $SMG$REMOVE_FROM_QUEUE removes the specified entry from the queue.
;  R1760  0      !-
;  R1761  0      MACRO
; MR1762  0          $SMG$REMOVE_FROM_QUEUE ( Q_ENTRY ) =
; MR1763  0          BEGIN
; MR1764  0              BUILTIN REMQUE;
; MR1765  0          LOCAL
; MR1766  0              FOO;    ! Item to be thrown away
; MR1767  0          REMQUE ( Q_ENTRY, FOO);
;  R1768  0          END; %;
;  R1769  0
;  R1770  0      !+
;  R1771  0      ! $SMG$Linear
;  R1772  0      ! -----------
;  R1773  0      ! Macro $SMG$Linear linearizes a two dimensional subcript formed by a 1-based
;  R1774  0      ! row number and a 1-based column number, into a single 0-based
;  R1775  0      ! subscript.
;  R1776  0      !-
;  R1777  C      MACRO
; MR1778  0              $SMG$LINEAR (ROW_NUMBER, COLUMN_NUMBER) =
;  R1779  0              (ROW_NUMBER-1)*.DCB [DCB_W_NO_COLS] + COLUMN_NUMBER -1 %;
;  R1780  0
```

```
;  R1781  0
;  R1782  0      !+
;  R1783  0      ! $SMG$Set_rend_code
;  R1784  0      !-------------------
;  R1785  0      ! This macro sets the rendition code by checking for optional rendition set
;  R1786  0      ! and rendition_complement arguments, and using the default from the DCB.
;  R1787  0      !-
; MR1788  0      MACRO $SMG$SET_REND_CODE (SET_ARG_NO, COMP_ARG_NO) =
; MR1789  0          BEGIN
; MR1790  0          BUILTIN
; MR1791  0                  NULLPARAMETER;
; MR1792  0          REND_CODE = .DCB [DCB_B_DEF_VIDEO_ATTR];
; MR1793  0          IF NOT NULLPARAMETER (SET_ARG_NO)
; MR1794  0          THEN
; MR1795  0              REND_CODE = .REND_CODE OR ..RENDITION_SET;
; MR1796  0          IF NOT NULLPARAMETER (COMP_ARG_NO)
; MR1797  0          THEN
; MR1798  0              REND_CODE = .REND_CODE XOR ..RENDITION_COMPLEMENT;
;  R1799  0          END; %;
;  R1800  0
;  R1801  0      !+
;  R1802  0      ! $SMG$Blank_fill_DCB
;  R1803  0      ! -------------------
;  R1804  0      ! This macro puts blanks into the text buffer, the default attribute byte
;  R1805  0      ! into the attribute buffer, and the default character set byte into the
;  R1806  0      ! character set buffer.
;  R1807  0      !-
;  R1808  0
; MR1809  0      MACRO $SMG$BLANK_FILL_DCB (NUM, SRC) =
; MR1810  0          BEGIN
; MR1811  0          LOCAL
; MR1812  0              TEXT_BUF : REF VECTOR [,BYTE],
; MR1813  0              ATTR_BUF : REF VECTOR [,BYTE],
; MR1814  0              CHAR_BUF : REF VECTOR [,BYTE];
; MR1815  0
; MR1816  0          TEXT_BUF = .DCB [DCB_A_TEXT_BUF];
; MR1817  0          ATTR_BUF = .DCB [DCB_A_ATTR_BUF];
; MR1818  0          CHAR_BUF = .DCB [DCB_A_CHAR_SET_BUF];
; MR1819  0
; MR1820  0          CH$FILL (%C' ',
; MR1821  0                   NUM,
; MR1822  0                   TEXT_BUF [SRC]);
; MR1823  0
; MR1824  0          CH$FILL (.DCB [DCB_B_DEF_VIDEO_ATTR],
; MR1825  0                   NUM,
; MR1826  0                   ATTR_BUF [SRC]);
; MR1827  0
; MR1828  0          IF .CHAR_BUF NEQ 0
; MR1829  0          THEN
; MR1830  0              CH$FILL (.DCB [DCB_B_DEF_CHAR_SET],
; MR1831  0                       NUM,
; MR1832  0                       CHAR_BUF [SRC]);
;  R1833  0          END; %;                          ! end of macro $SMG$BLANK_FILL_DCB
;  R1834  0
;  R1835  0      !+
;  R1836  0      ! $SMG$Find_nonblank_len
;  R1837  0      ! ----------------------
```

```
;  R1838  0      ! This macro finds the length of a string minus trailing blanks.  Since
;  R1839  0      ! the text is already in a display buffer, there should be no funny
;  R1840  0      ! characters such as tabs.   Start at the end of the string and search
;  R1841  0      ! backwards to find the first character which is not a space.
;  R1842  0      !-
; MR1843  0      MACRO $SMG$FIND_NONBLANK_LEN (STRING_ADDR, STRING_LEN, NONBLANK_LEN) =
; MR1844  0          BEGIN
; MR1845  0
; MR1846  0          NONBLANK_LEN = STRING_LEN;                  ! initialize the length
; MR1847  0
; MR1848  0          WHILE .NONBLANK_LEN NEQU 0
; MR1849  0          DO
; MR1850  0              BEGIN
; MR1851  0              IF (CH$RCHAR (CH$PLUS (STRING_ADDR, .NONBLANK_LEN - 1))
; MR1852  0                      EQL %C' ')
; MR1853  0              THEN
; MR1854  0                  NONBLANK_LEN = .NONBLANK_LEN   1
; MR1855  0              ELSE
; MR1856  0                  EXITLOOP;
; MR1857  0              END;
;  R1858  0          END; %;
;  R1859  0
;  R1860  0      !+
;  R1861  0      ! $SMG$Shuffle
;  R1862  0      ! ------------
;  R1863  0      ! The following macro is used to move text within a display.
;  R1864  0      !-
; MR1865  0      MACRO $SMG$SHUFFLE (NUM, SRC, DST) =
; MR1866  0          BEGIN
; MR1867  0              CH$MOVE (NUM,
; MR1868  0                       TEXT_BUF [SRC],
; MR1869  0                       TEXT_BUF [DST]);
; MR1870  0
; MR1871  0              CH$MOVE (NUM,
; MR1872  0                       ATTR_BUF [SRC],
; MR1873  0                       ATTR_BUF [DST]);
; MR1874  0
; MR1875  0              IF .CHAR_BUF NEQ 0      ! exists only if char set requested
; MR1876  0              THEN
; MR1877  0                  CH$MOVE (NUM,
; MR1878  0                           CHAR_BUF [SRC],
; MR1879  0                           CHAR_BUF [DST]);
;  R1880  0          END; %;                     ! end of macro $SMG$SHUFFLE
;  R1881  0
;  R1882  0
;  R1883  0      !+
;  R1884  0      ! this is used by SMG$INSERT_CHARS only
;  R1885  0      ! This macro is used to determine how many positions a string will
;  R1886  0      ! occupy when it is printed.  It takes into account funny characters
;  R1887  0      ! such as tabs and backspaces.
;  R1888  0      !-
;  R1889  0
; MR1890  0      MACRO $SMG$FIND_PRINT_LENGTH (TEXT_LEN, TEXT_ADDR, PRINT_LEN) =
; MR1891  0      BEGIN
; MR1892  0          EXTERNAL LITERAL
; MR1893  0              SMG$_FATERRLIB;
; MR1894  0
```

```
;MR1895  0          EXTERNAL
;MR1896  0              CHAR_TABLE : VECTOR [,BYTE];
;MR1897  0
;MR1898  0          BUILTIN
;MR1899  0              SCANC;
;MR1900  0
;MR1901  0          LOCAL
;MR1902  0              ALLONES : BYTE INITIAL (-1),
;MR1903  0                                ! needed by ref
;MR1904  0              BYTES_REMAINING,  ! No. of bytes in input string yet to be
;MR1905  0                                ! processed.
;MR1906  0              IN_POINTER;       ! Current pointer into input string
;MR1907  0
;MR1908  0          PRINT_LEN = 0;        ! initialize
;MR1909  0          BYTES_REMAINING = .TEXT_LEN;
;MR1910  0          IN_POINTER = TEXT_ADDR;
;MR1911  0          TEXT_LEN = 0;         ! modify this to be the actual number
;MR1912  0                                ! of chars processed
;MR1913  0          WHILE .BYTES_REMAINING NEQ 0
;MR1914  0          DO
;MR1915  0              BEGIN   ! Overall loop
;MR1916  0              LOCAL
;MR1917  0                  NEW_BYTES_REMAINING,  ! No. of bytes remaining as returned
;MR1918  0                                        ! by SCANC
;MR1919  0                  ADDR_DIFF;            ! Addr of char in input stream whose
;MR1920  0                                        ! index into scanc table yields
;MR1921  0                                        ! non-zero code.
;MR1922  0              !+
;MR1923  0              ! See if any of the remaining input characters require special
;MR1924  0              ! treatment.
;MR1925  0              !-
;MR1926  0              SCANC ( BYTES_REMAINING,        ! No. of bytes remaining
;MR1927  0                      .IN_POINTER,            ! Current pointer to source
;MR1928  0                      CHAR_TABLE,             ! Address of SCANC table
;MR1929  0                      ALLONES;                ! Mask for ANDing
;MR1930  0                      NEW_BYTES_REMAINING,    ! New remaining no. of bytes
;MR1931  0                                              ! including the byte which
;MR1932  0                                              ! caused the instruction to
;MR1933  0                                              ! halt.  Is zero only if all
;MR1934  0                                              ! bytes did not satify search.
;MR1935  0                      ADDR_DIFF);             ! Addr of char in input stream
;MR1936  0                                              ! whose index into scanc table
;MR1937  0                                              ! yields non-zero code.
;MR1938  0
;MR1939  0              IN_POINTER = .IN_POINTER + (.BYTES_REMAINING - .NEW_BYTES_REMAINING);
;MR1940  0              PRINT_LEN = .PRINT_LEN + (.BYTES_REMAINING - .NEW_BYTES_REMAINING);
;MR1941  0                                              ! maintain printing length of chars
;MR1942  0              TEXT_LEN = .TEXT_LEN + (.BYTES_REMAINING - .NEW_BYTES_REMAINING);
;MR1943  0                                              ! pass back no. of chars processed
;MR1944  0              BYTES_REMAINING = .NEW_BYTES_REMAINING;
;MR1945  0
;MR1946  0              IF .NEW_BYTES_REMAINING EQL 0
;MR1947  0              THEN
;MR1948  0                  EXITLOOP;           ! Break out of loop -- we're done
;MR1949  0
;MR1950  0              !+
;MR1951  0              ! Dispatch on the non-zero code located to see what special
```

```
; MR1952  0               ! action is needed.
; MR1953  0               !-
; MR1954  0               CASE .CHAR_TABLE [.(.ADDR_DIFF)<0,8>] FROM 1 TO 10 OF
; MR1955  0               SET
; MR1956  0                   [1,2,3,10]:
; MR1957  0                   !+
; MR1958  0                   !    Hex Character Codes       ASCII Character
; MR1959  0                   !    ------------------        ---------------
; MR1960  0                   !    00 to 06                   NUL to ACK
; MR1961  0                   !    0E to 1A                   SO to SUB
; MR1962  0                   !    1C to 1F                   FS to US
; MR1963  0                   !       07                       BEL
; MR1964  0                   !       08                       BS
; MR1965  0                   !       7F                       DEL
; MR1966  0                   !
; MR1967  0                   ! Non-printing characters
; MR1968  0                   !-
; MR1969  0                       ;
; MR1970  0                   [4]:
; MR1971  0                   !+
; MR1972  0                   !    Hex Character Codes       ASCII Character
; MR1973  0                   !    ------------------        ---------------
; MR1974  0                   !       09                       HT
; MR1975  0                   !
; MR1976  0                   ! TABs require extra space.
; MR1977  0                   !
; MR1978  0                   ! TAB stops are assumed to be set in the following columns:
; MR1979  0                   ! 9, 17, 25, 33, 41, 49, 57, 65, 73 ( width=80)
; MR1980  0                   !
; MR1981  0                   ! 9, 17, 25, 33, 41, 49, 57, 65, 73, 81, 89, 97, 105, 113,
; MR1982  0                   ! 121, 129 ( width=132)
; MR1983  0                   !-
; MR1984  0                   BEGIN
; MR1985  0                   PRINT_LEN = .PRINT_LEN +
; MR1986  0                                   ((.DCB [DCB_W_CURSOR_COL]-1)/8+1)*8+1;
; MR1987  0                   TEXT_LEN = .TEXT_LEN + 1;   ! include as a 'printable' char
; MR1988  0                   END;
; MR1989  0                   [5,6,7,8,9]:
; MR1990  0                   !+
; MR1991  0                   !    Hex Character Codes       ASCII Character
; MR1992  0                   !    ------------------        ---------------
; MR1993  0                   !       0A                       LF
; MR1994  0                   !       0B                       VT
; MR1995  0                   !       0C                       FF
; MR1996  0                   !       0D                       CR
; MR1997  0                   !       1B                      ESC
; MR1998  0                   !
; MR1999  0                   ! These characters do not make sense when inserting characters.
; MR2000  0                   ! Throw away all text after this.
; MR2001  0                   !-
; MR2002  0                   BEGIN
; MR2003  0                   EXITLOOP;
; MR2004  0                   END;
; MR2005  0
; MR2006  0                   [INRANGE, OUTRANGE]:
; MR2007  0                   !+
; MR2008  0                   ! Should never get here -- there are no other codes in
```

```
;MR2009  0                  ! CHAR_TABLE.  If we do, we've got a problem.
;MR2010  0                  !-
;MR2011  0                  BEGIN
;MR2012  0                  RETURN SMG$_FATERRLIB;
;MR2013  0                  END;
;MR2014  0              TES;
;MR2015  0
;MR2016  0
;MR2017  0              !+
;MR2018  0              ! Re-adjust pointer and count of bytes left to account for
;MR2019  0              ! the special character(s) just processed.
;MR2020  0              !-
;MR2021  0              IN_POINTER = .IN_POINTER + 1;
;MR2022  0              BYTES_REMAINING = .BYTES_REMAINING -1;
;MR2023  0              END;    ! Overall loop
;MR2024  0
;R2025   0          END;        %;              ! End of macro $SMG$FIND_PRINT_LENGTH
;R2026   0
;R2027   0      !+
;R2028   0      ! Macros used for input routines.
;R2029   0      !-
;R2030   0
;R2031   0      MACRO
;MR2032  0          $SMG$VALIDATE_KTH (KEY_TABLE_ID, KTH) =
;MR2033  0              BEGIN
;MR2034  0              EXTERNAL LITERAL
;MR2035  0                  SMG$_INVKTB_ID;
;MR2036  0              KTH = .KEY_TABLE_ID [0];
;MR2037  0              IF .KTH EQL 0
;MR2038  0              THEN
;MR2039  0                  RETURN SMG$_INVKTB_ID;      ! Invalid key-table-id
;MR2040  0              IF .KTH [KTH_L_CHECK] NEQA .KTH
;MR2041  0              THEN
;MR2042  0                  RETURN SMG$_INVKTB_ID;      ! Invalid key-table-id
;R2043   0              END %;
;R2044   0
;R2045   0      MACRO
;MR2046  0          $SMG$VALIDATE_KCB (KEYBOARD_ID, KCB) =
;MR2047  0              BEGIN
;MR2048  0              EXTERNAL LITERAL
;MR2049  0                  SMG$_INVKBD_ID;
;MR2050  0              KCB = .KEYBOARD_ID [0];
;MR2051  0              IF .KCB EQL 0
;MR2052  0              THEN
;MR2053  0                  RETURN SMG$_INVKBD_ID;      ! Invalid keyboard-id
;MR2054  0              IF .KCB [KCB_L_CHECK] NEQA .KCB
;MR2055  0              THEN
;MR2056  0                  RETURN SMG$_INVKBD_ID;      ! Invalid keyboard-id
;R2057   0              END %;
;R2058   0
```

```
;
; L    %IF NOT %DECLARED (SMG$K_TRM_CTRLA)      ! check for 1st symbol in SMGDEF
; U    %THEN REQUIRE 'RTLML:SMGDEF';
;      %FI
;
; L    %IF NOT %DECLARED (SMG$K_TOP)
;      %THEN
;              LITERAL SMG$K_TOP=0;
;              LITERAL SMG$K_BOTTOM=1;
;              LITERAL SMG$K_LEFT=2;
;              LITERAL SMG$K_RIGHT=3;
;      %FI
;
; L    %IF NOT %DECLARED (SMG$K_ANSI_CRT)       ! check for 1st symbol in SMGTRMPTR
; U    %THEN REQUIRE 'RTLML:SMGTRMPTR';
;      %FI
;
;      ! End of file SMGLIB.REQ
```

Library Statistics

File	Symbols Total	Symbols Loaded	Symbols Percent	Pages Mapped	Processing Time
_255DUA28:[SYSLIB]STARLET.L32;1	9776	2	0	581	00:01.1

COMMAND QUALIFIERS

```
;       BLISS/LIBRARY=LIB$:/LIST=LIS$:/SOURCE=REQUIRE SRC$.SMGLIB

; Run Time:         00:16.4
; Elapsed Time:     00:20.6
; Lines/CPU Min:     7609
; Lexemes/CPU-Min: 25396
; Memory Used:  126 pages
; Library Precompilation Complete
```

SMGKEYUTI
LIS

SMGKEYPAD
LIS

SMGINPUT
LIS

SMGLIB
LIS

SMGMINUPD
LIS

SMGMIN
LIS

SMGMAPTRM
LIS