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

**FILE**ID**SMGMISC                H 1

SMGMISC.

LIS

```
0 %TITLE 'SMG$MISC - Miscellaneous routines for screen mgmt'
0 MODULE SMG$MISC (
0                 IDENT = '1-012' ! EDIT: STAN1013
0                 ) =
1 BEGIN
1 !
```



```
1 !
```

```
1 !++
1 ! FACILITY:     Screen Management
1 !
1 ! ABSTRACT:
1 !
1 !       This module contains miscellaneous routines
1 !       having to do with asynchronous terminal operation.
1 !
1 ! ENVIRONMENT:  User mode, Shared library routines.
1 !
1 ! AUTHOR: Stanley Rabinowitz, CREATION DATE: 3-May-1983
1 !
1 ! MODIFIED BY:
1 !
1 ! 1-013 Stan     24-Jul-1984     Remove informational errors.
1 ! 1-012 Stan      4-Apr-1984     Add error message for non-ANSI terminal.
1 !                                Tighten up error checking in unsolicted input.
1 ! 1-011 Stan     18-Mar-1984     Allow asking terminal for physical cursor location
1 !                                Rename some routines.
1 ! 1-010 Stan     17-Mar-1984     More fixes to documentation
1 !                                Fix bug in initializing mailbox.
1 ! 1-009 Stan     21-Feb-1984     Fix documentation
1 ! 1-008 Stan     27-Nov-1983     Unsolicited input.
1 ! 1-007 Stan     27-Sep-1983     Get background color.
1 ! 1-006 Stan     17-Aug-1983     Add items to GET_PASTEBOARD_ATTRIBUTES.
1 ! 1-005 Stan      7-Jul-1983     Call user's AST routine.
1 ! 1-004 Stan     13-Jun-1983     Finish mailbox stuff.
1 ! 1-003 Stan      2-Jun-1983     Add SMG$GET_PASTEBOARD_ATTRIBUTES.
1 ! 1-002 Stan      1-Jun-1983     Set characteristics
1 ! 1-001 Stan      3-May-1983     Original version.
1 !--
```

```
0061 1 %SBTTL 'Declarations'
0062 1 !
0063 1 ! TABLE OF CONTENTS:
0064 1 !
0065 1
0066 1 FORWARD ROUTINE
0067 1
0068 1         SMG$SET_OUT_OF_BAND_ASTS,           ! Enable out-of-band ASTs
0069 1         SMG$SET_BROADCAST_TRAPPING,         ! Enable trapping of broadcast messages
0070 1         SMG$ENABLE_UNSOLICITED_INPUT,       ! Enable unsolicited input
0071 1         SMG$GET_BROADCAST_MESSAGE,          ! Gets a broadcast message
0072 1         SMG$GET_PASTEBOARD_ATTRIBUTES,      ! Get information about pasteboard
0073 1         SMG$GET_PHYSICAL_CURSOR,            ! Get physical cursor position
0074 1
0075 1         SMG$$OUT_OF_BAND_HANDLER,           ! Generic out-of-band AST handler
0076 1         SMG$$SET_TERM_CHARACTERISTICS,      ! Set characteristics
0077 1                                             ! in a modular fashion
0078 1
0079 1         MAILBOX_AST_ROUTINE,                ! Handles messages to our mailbox
0080 1         INITIALIZE_MAILBOX,                 ! Sets up mailbox and queues
0081 1         CHAR_EXIT_HANDLER;                  ! Handler to reset terminal
0082 1                                             ! characteristics
0083 1
0084 1 !
0085 1 ! SWITCHES:
0086 1 !
0087 1 !       In the include files.
0088 1 !
0089 1
0090 1 !
0091 1 ! LINKAGES:
0092 1 !
0093 1 !       In the include files.
0094 1 !
0095 1
0096 1 !
0097 1 ! INCLUDE FILES:
0098 1 !
0099 1
0100 1 REQUIRE 'SRC$:SMGPROLOG';                 ! defines psects, macros,
0178 1                                             ! structures and terminal defs
0179 1 REQUIRE 'RTLIN:STRLNK';                    ! linkages
0364 1
0365 1 !
0366 1 ! MACROS:
0367 1 !
0368 1
0369 1 !
0370 1 ! EQUATED SYMBOLS:
0371 1 !
0372 1
0373 1 LITERAL
0374 1
0375 1         SMG$S_CHAR_EXIT_BLOCK   = 9*4;      ! Size of exit block used by
0376 1                                             ! SMG$$SET_TERMINAL_CHARACTERISTICS
0377 1                                             ! (in bytes).
0378 1
```

```
1 !
1 ! FIELDS:
1 !
1
1 FIELD
1
1         SMGMBX_FIELDS =
1
1 SET
1
1 SMGMBX_L_FLINK  = [ 0,0,32,0],   ! Forward  link in queue of mailbox blocks
1 SMGMBX_L_BLINK  = [ 4,0,32,0],   ! Backward link in queue of mailbox blocks
1 SMGMBX_A_BUFFER = [ 8,0,32,0],   ! Addres of buffer containing received message
1 SMGMBX_Q_IOSB   = [12,0, 0,0],   ! Copy of 4-word I/O status block
1 SMGMBX_W_STATUS = [12,0,16,0],   ! Status of mailbox read (from I/O status block)
1 SMGMBX_W_MSGLEN = [14,0,16,0],   ! Size of message received
1 SMGMBX_L_SENDER = [16,0,32,0],   ! PID of process that sent message
1 SMGMBX_W_BUFSIZ = [20,0,16,0],   ! Allocated size of buffer
1 SMGMBX_W_UNUSED = [22,0,16,0]    ! Reserved for future use
1
1 TES;
1
1 LITERAL
1
1         SMGMBX_S_SIZE   = 24;    ! Size of structure in bytes
1
1 MACRO
1
1         $SMGMBX_DECL     = BLOCK[SMGMBX_S_SIZE,BYTE] FIELD(SMGMBX_FIELDS) %;
1
1 !
1 ! PSECTS:
1 !
1
1 ! OWN STORAGE:
1 !
1 !       NONE    (and shouldn't have any)
1
1 LITERAL
1
1         MAX_MSG = 300,   ! *****
1         BUFQUO  = 300;   ! *****
1
1 !
1 ! EXTERNAL REFERENCES:
1 !
1
1 !EXTERNAL ROUTINE
1 !
1 !       NONE
```

```
%SBTTL 'SMG$GET_PASTEBOARD_ATTRIBUTES'
GLOBAL ROUTINE SMG$GET_PASTEBOARD_ATTRIBUTES (
    PASTEBOARD_ID,P_PIT,P_PIT_SIZE) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine gets information about a pasteboard and
!       leaves it in a user-supplied area (the PIT), the pasteboard
!       information table.
!
! CALLING SEQUENCE:
!
!       RET_STATUS.wlc.v = SMG$GET_PASTEBOARD_ATTRIBUTES (
!                               PASTEBOARD_ID.rl.r,
!                               P_PIT.wab.r,
!                               P_PIT_SIZE.rl.r)
!
! FORMAL PARAMETERS:
!
!       PASTEBOARD_ID.rl.r      Address of pasteboard id
!
!       P_PIT.wab.r             Address of a pasteboard information table.
!                               This is a byte block whose size and field
!                               references are described in SMGDEF.SDL.
!                               It is the caller's responsibility to allocate
!                               the correct size block and pass it's address
!                               to this routine.
!
!       P_PIT_SIZE.rl.r         Address of a longword containing the size of
!                               the PIT (in bytes).  For now, this size must
!                               be exactly correct.  This will allow the
!                               structure to grow in future releases.
!
! IMPLICIT INPUTS:
!
!       contents of PBCB
!
! IMPLICIT OUTPUTS:
!
!       PIT gets filled in.
!
! COMPLETION STATUS:
!
!       SMG$_INVARG     PIT is the wrong size
!       SS$_NORMAL      Normal successful completion
!
! SIDE EFFECTS:
!
!       NONE
!
!--
```

```
BEGIN

LOCAL

        STATUS,
        WCB             : REF BLOCK[,BYTE],     ! Window control block
        PBCB            : REF BLOCK[,BYTE];     ! Pasteboard control block

BIND

        PIT             = .P_PIT        : BLOCK[,BYTE], ! pasteboard info block
        PIT_SIZE        = .P_PIT_SIZE;                  ! size of user's PIT

EXTERNAL LITERAL

        SMG$_INVARG;            ! PIT is the wrong size

EXTERNAL ROUTINE

        SMG$FIND_CURSOR_DISPLAY;
```

```
$SMG$VALIDATE_ARGCOUNT(3,3);

$SMG$GET_PBCB(.PASTEBOARD_ID,PBCB);

!+
! Make sure that the PIT is the correct size.
!-

IF .PIT_SIZE NEQ SMG$S_PASTEBOARD_INFO_BLOCK
  THEN RETURN SMG$_INVARG;

WCB=.PBCB[PBCB_A_WCB];

!+
! Move arguments from the PBCB to the PIT.
!-

PIT[SMG$L_DEVCHAR]       = .PBCB[PBCB_L_DEVCHAR];
PIT[SMG$L_DEVDEPEND]     = .PBCB[PBCB_L_DEVDEPEND];
PIT[SMG$L_DEVDEPEND2]    = .PBCB[PBCB_L_DEVDEPEND2];
PIT[SMG$B_DEVCLASS]      = .PBCB[PBCB_B_CLASS];
PIT[SMG$B_SMG_DEVTYPE]   = .PBCB[PBCB_B_DEVTYPE];
PIT[SMG$B_PHY_DEVTYPE]   = .PBCB[PBCB_B_PHY_DEV_TYPE];
PIT[SMG$B_ROWS]          = .PBCB[PBCB_B_ROWS];
PIT[SMG$W_WIDTH]         = .PBCB[PBCB_W_WIDTH];
PIT[SMG$B_PARITY]        = .PBCB[PBCB_B_PARITY];
PIT[SMG$W_SPEED]         = .PBCB[PBCB_W_SPEED];
PIT[SMG$W_FILL]          = .PBCB[PBCB_W_FILL];
PIT[SMG$B_COLOR]         = .PBCB[PBCB_B_BACKGROUND_COLOR];

!+
! Move arguments from the WCB into the PIT.
!-

PIT[SMG$W_CURSOR_ROW]    = .WCB[WCB_W_CURR_CUR_ROW];
PIT[SMG$W_CURSOR_COL]    = .WCB[WCB_W_CURR_CUR_COL];

!+
! Find out which virtual display the cursor is in and
! leave that in the PIT.
!-

STATUS=SMG$FIND_CURSOR_DISPLAY(.PASTEBOARD_ID,PIT[SMG$L_CURSOR_DID]);
IF NOT .STATUS THEN RETURN .STATUS;

RETURN  SS$_NORMAL

END;
```

```
;                                   .TITLE  SMG$MISC SMG$MISC - Miscellaneous routines for
                                                     screen mg
                                    .IDENT  \1-012\

                                    .EXTRN  SMG$_INVARG, SMG$FIND_CURSOR_DISPLAY
                                    .EXTRN  SMG$_WRONUMARG, SMG$_INVPAS_ID
                                    .EXTRN  PBD_C_COUNT, PBD_A_PBCB
```

```
                                                    .EXTRN  PBD_V_PB_AVAIL

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

                                  0004 00000        .ENTRY  SMG$GET_PASTEBOARD_ATTRIBUTES, Save R2        ; 0430
            50        08    AC    D0 00002          MOVL    P_PIT, R0                                     ; 0491
            03              6C    91 00006          CMPB    (AP), #3                                      ; 0501
                            08    13 00009          BEQL    1$                                            ;
            50 00000000G    8F    D0 0000B          MOVL    #SMG$_WRONUMARG, R0                           ;
                                  04 00012          RET                                                   ;
            51        04    BC    D0 00013 1$:      MOVL    @PASTEBOARD_ID, R1                            ; 0503
                            11    19 00017          BLSS    2$                                            ;
  00000000G 00              51    D1 00019          CMPL    R1, PBD_L_COUNT                               ;
                            08    14 00020          BGTR    2$                                            ;
08 00000000G 00             51    E0 00022          BBS     R1, PBD_V_PB_AVAIL, 3$                        ;
            50 00000000G    8F    D0 0002A 2$:      MOVL    #SMG$_INVPAS_ID, R0                           ;
                                  04 00031          RET                                                   ;
            51 00000000G0041      D0 00032 3$:      MOVL    PBD_A_PBCB[R1], PBCB                          ;
            20        0C    BC    D1 0003A          CMPL    @P_PIT_SIZE, #32                              ; 0509
                            08    13 0003E          BEQL    4$                                            ;
            50 00000000G    8F    D0 00040          MOVL    #SMG$_INVARG, R0                              ; 0510
                                  04 00047          RET                                                   ;
            52        08    A1    D0 00048 4$:      MOVL    8(PBCB), WCB                                  ; 0512
            60        58    A1    7D 0004C          MOVQ    88(PBCB), (R0)                                ; 0518
       08   A0        60    A1    D0 00050          MOVL    96(PBCB), 8(R0)                               ; 0520
       0C   A0        58    A1    90 00055          MOVB    88(PBCB), 12(R0)                              ; 0521
       0D   A0        10    A1    90 0005A          MOVB    16(PBCB), 13(R0)                              ; 0522
       0E   A0        59    A1    90 0005F          MOVB    89(PBCB), 14(R0)                              ; 0523
       0F   A0        5F    A1    90 00064          MOVB    95(PBCB), 15(R0)                              ; 0524
       10   A0        5A    A1    B0 00069          MOVW    90(PBCB), 16(R0)                              ; 0525
       13   A0        11    A1    90 0006E          MOVB    17(PBCB), 19(R0)                              ; 0526
       14   A0      00B4    C1    D0 00073          MOVL    180(PBCB), 20(R0)                             ; 0527
       12   A0      00F9    C1    90 00079          MOVB    249(PBCB), 18(R0)                             ; 0529
       18   A0        20    A2    D0 0007F          MOVL    32(WCB), 24(R0)                               ; 0535
                      1C    A0    9F 00084          PUSHAB  28(R0)                                        ; 0543
                      04    AC    DD 00087          PUSHL   PASTEBOARD_ID                                 ;
  00000000G 00              02    FB 0008A          CALLS   #2, SMG$FIND_CURSOR_DISPLAY                   ;
            03              50    E9 00091          BLBC    STATUS, 5$                                    ; 0544
            50              01    D0 00094          MOVL    #1, R0                                        ; 0546
                                  04 00097 5$:      RET                                                   ; 0548

; Routine Size:  152 bytes,    Routine Base:  _SMG$CODE + 0000
```

```
%SBTTL 'SMG$GET_PHYSICAL_CURSOR'
GLOBAL ROUTINE SMG$GET_PHYSICAL_CURSOR (
        PASTEBOARD_ID,P_ROW,P_COL,P_PHYS_ROW,P_PHYS_COL) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine returns information about where
!       the physical cursor is on the pasteboard.
!       In case SMG is unsure of the cursor location (for example
!       if the pasteboard is created without clearing the screen),
!       there is also an option to inquire of the terminal
!       finding out where it thinks the cursor is.
!
! CALLING SEQUENCE:
!
!       RET_STATUS.wlc.v = SMG$GET_PHYSICAL_CURSOR (
!                               PASTEBOARD_ID.rl.r,
!                               [,P_ROW.wl.r]
!                               [,P_COL.wl.r]
!                               [,P_PHYS_ROW.wl.r]
!                               [,P_PHYS_COL.wl.r])
!
! FORMAL PARAMETERS:
!
!       PASTEBOARD_ID.rl.r      Address of pasteboard id
!
!       P_ROW.wl.r              Gets SMG's idea of what row cursor is in
!                               0 means it does not know.
!
!       P_COL.wl.r              Gets SMG's idea of what col cursor is in
!                               0 means it does not know.
!
!       P_PHYS_ROW.wl.r         Gets terminal's idea of what row cursor is in
!
!       P_PHYS_COL.wl.r         Gets terminal's idea of what col cursor is in
!                               If this and the previous arguments are omitted,
!                               then no inquiry to the terminal is made
!                               and no typeahead is purged.
!
! IMPLICIT INPUTS:
!
!       contents of PBCB
!
! IMPLICIT OUTPUTS:
!
!       SMG's idea of physical cursor location may get updated
!
! COMPLETION STATUS:
!
!       SMG$_TRMNOTANS  Terminal is not ansi
!       SS$_NORMAL      Normal successful completion
!                       (includes case where position is not known)
!
! SIDE EFFECTS:
!
!       Type ahead may be cancelled.
```

```
;  351     0606   1 !
;  352     0607   1 !--
```

```
2 BEGIN
2
2 LOCAL
2
2             STATUS,
2             WCB               : REF BLOCK[,BYTE],      ! Window control block
2             PBCB              : REF BLOCK[,BYTE];      ! Pasteboard control block
2
2 BUILTIN
2
2             NULLPARAMETER;
2
2 BIND
2
2             ROW               = .P_ROW,
2             COL               = .P_COL,
2             PHYS_ROW          = .P_PHYS_ROW,
2             PHYS_COL          = .P_PHYS_COL;
2
2 EXTERNAL LITERAL
2
2             SMG$_TRMNOTANS, ! Terminal is not ANSI
2             SMG$_INVARG;
2
2 OWN
2
2             INQUIRY           : VECTOR[4,BYTE] INITIAL( BYTE(27,'[6n') );
2
2                               ! ANSI Cursor position report
```

```
;      2 $SMG$VALIDATE_ARGCOUNT(1,5);
;      2
;      2 $SMG$GET_PBCB(.PASTEBOARD_ID,PBCB);
;      2
;      2 WCB=.PBCB[PBCB_A_WCB];
;      2
;      2 !+
;      2 ! Move physical cursor row and column from the WCB to the user, if requested.
;      2 !-
;      2
;      2 IF NOT NULLPARAMETER(P_ROW)
;      2   THEN  ROW = .WCB[WCB_W_CURR_CUR_ROW];
;      2 IF NOT NULLPARAMETER(P_COL)
;      2   THEN  COL = .WCB[WCB_W_CURR_CUR_COL];
;      2
;      2 !+
;      2 ! If the user requested that we interrogate the terminal, then go do that now.
;      2 !-
;      2
;      2 IF NOT NULLPARAMETER(P_PHYS_ROW)
;      2 OR NOT NULLPARAMETER(P_PHYS_COL)
;      3   THEN  BEGIN   ! Interrogate terminal
;      3
;      3         LOCAL
;      3
;      3                 PTR,
;      3                 SEQ             : REF VECTOR[,BYTE],
;      3                 ROW_START,
;      3                 ROW_LEN,
;      3                 COL_START,
;      3                 COL_LEN,
;      3                 RESP_BUFFER     : VECTOR[SMG$K_LONGEST_SEQUENCE,BYTE],
;      3                 IOSB            : VECTOR[4,WORD],
;      3                 TERM_ROW : VOLATILE,    ! physical row as reported by terminal
;      3                 TERM_COL : VOLATILE;    ! physical col as reported by terminal
;      3
;      3         !+
;      3         ! Check that the terminal is really ANSI and that it supports
;      3         ! the cursor position report.
;      3         !-
;      3
;      4         BEGIN
;      4         BIND TT2=PBCB[PBCB_L_DEVDEPEND2] : $BBLOCK;
;      4         IF NOT .TT2[TT2$V_ANSICRT] THEN RETURN SMG$_TRMNOTANS
;      3         END;
;      3
;      3         !+
;      3         ! Issue a QIO to the terminal, sending it the standard
;      3         ! ANSI sequence asking for the row and column of the cursor,
;      3         ! and getting back a respnse in RESP_BUFFER.
;      3         ! Purge type-ahead first, and allow the read to time out.
;      3         !-
;      3
;    P 3         STATUS=$QIOW(   FUNC    = IO$_READPROMPT OR IO$M_TIMED OR
;    P 3                                   IO$M_PURGE     OR IO$M_ESCAPE,
;    P 3                         CHAN    = .PBCB[PBCB_W_CHAN],
;    P 3                         EFN     = .PBCB[PBCB_B_EFN],
```

```
                                IOSB    = IOSB,
                                P1      = RESP_BUFFER,                  ! gets reply
                                P2      = %ALLOCATION(RESP_BUFFER),
                                P3      = 4,                            ! wait 4 seconds
                                P5      = INQUIRY,                      ! inquire sequence
                                P6      = %ALLOCATION(INQUIRY) );

        IF .STATUS THEN STATUS=.IOSB[0];
        IF NOT .STATUS THEN RETURN .STATUS;

        !+
        ! If the read succeeds, then IOSB[3] has the number of
        ! characters in the escape sequence read, and
        ! IOSB[1] points to the beginning of this escape sequence.
        ! The format of the answer is "$ [ r ; c R" where r and c
        ! are the row and column in decimal-coded ASCII.
        ! Start by pointing past the escape and the "[" and then parse a number.
        !-

        ROW_START=RESP_BUFFER[.IOSB[1]+2];

        PTR=CH$FIND_CH(.IOSB[3]-1,.ROW_START,';');
        IF CH$FAIL(.PTR)
          THEN  RETURN  0;
        ROW_LEN=.PTR-.ROW_START;

        COL_START=.PTR+1;
        PTR=CH$FIND_CH(.IOSB[3]-1,.COL_START,'R');
        IF CH$FAIL(.PTR)
          THEN  RETURN  0;
        COL_LEN=.PTR-.COL_START;

        !+
        ! Convert the decimal-coded ASCII to a number.
        !-

        SEQ=.ROW_START;
        PHYS_ROW=0;
        INCR I FROM 0 TO .ROW_LEN-1 DO
                PHYS_ROW=.PHYS_ROW*10+.SEQ[.I]-%C'0';

        SEQ=.COL_START;
        PHYS_COL=0;
        INCR I FROM 0 TO .COL_LEN-1 DO
                PHYS_COL=.PHYS_COL*10+.SEQ[.I]-%C'0';

        !+
        ! Pass the row and column back to the user, if requested.
        !-

        IF NOT NULLPARAMETER(PHYS_ROW)
          THEN  PHYS_ROW = .TERM_ROW;
        IF NOT NULLPARAMETER(PHYS_COL)
          THEN  PHYS_COL = .TERM_COL;

        END;    ! Interrogate terminal
```

```
;  498          0751  2 RETURN  SS$_NORMAL
;  499          0752  2
;  500          0753  1 END;

                                                                     .PSECT  _SMG$DATA,NOEXE,  PIC,2

                                          1B  00000 INQUIRY:.BYTE    27                                          ;
                                  6E  36  5B  00001         .ASCII   \[6n\                                       ;

                                                            .EXTRN   SMG$_TRMNOTANS, SYS$QIOW

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

                                        01FC  00000         .ENTRY   SMG$GET_PHYSICAL_CURSOR, Save R2,R3,R4,R5,- ; 0550
                                                                     R6,R7,R8                                    ;
                    5E      FEF0  CE  9E  00002             MOVAB    -272(SP), SP                                ;
                    56        10  AC  D0  00007             MOVL     P_PHYS_ROW, R6                              ; 0624
                    55        14  AC  D0  0000B             MOVL     P_PHYS_COL, R5                              ; 0625
          50        6C            01  83  0000F             SUBB3    #1, (AP), DIFF                              ; 0637
                    04            50  91  00013             CMPB     DIFF, #4                                    ;
                                  08  1B  00016             BLEQU    1$                                          ;
                    50 00000000G  8F  D0  00018             MOVL     #SMG$_WRONUMARG, R0                         ;
                                      04  0001F             RET                                                  ;
                    50        04  BC  D0  00020 1$:         MOVL     @PASTEBOARD_ID, R0                          ; 0639
                                  11  19  00024             BLSS     2$                                          ;
          00000000G 00            50  D1  00026             CMPL     R0, PBD_L_COUNT                             ;
                                  08  14  0002D             BGTR     2$                                          ;
       08 00000000G 00            50  E0  0002F             BBS      R0, PBD_V_PB_AVAIL, 3$                      ;
                    50 00000000G  8F  D0  00037 2$:         MOVL     #SMG$_INVPAS_ID, R0                         ;
                                      04  0003E             RET                                                  ;
                    51 00000000G0040  D0  0003F 3$:         MOVL     PBD_A_PBCB[R0], PBCB                        ;
                    50        08  A1  D0  00047             MOVL     8(PBCB), WCB                                ; 0641
                    02            6C  91  0004B             CMPB     (AP), #2                                    ; 0647
                                  0A  1F  0004E             BLSSU    4$                                          ;
                              08  AC  D5  00050             TSTL     8(AP)                                       ;
                                  05  13  00053             BEQL     4$                                          ;
               08   BC        20  A0  32  00055             CVTWL    32(WCB), @P_ROW                             ; 0648
                    03            6C  91  0005A 4$:         CMPB     (AP), #3                                    ; 0649
                                  0A  1F  0005D             BLSSU    5$                                          ;
                              0C  AC  D5  0005F             TSTL     12(AP)                                      ;
                                  05  13  00062             BEQL     5$                                          ;
               0C   BC        22  A0  32  00064             CVTWL    34(WCB), @P_COL                             ; 0650
                    04            6C  91  00069 5$:         CMPB     (AP), #4                                    ; 0656
                                  05  1F  0006C             BLSSU    6$                                          ;
                              10  AC  D5  0006E             TSTL     16(AP)                                      ;
                                  0D  12  00071             BNEQ     9$                                          ;
                    05            6C  91  00073 6$:         CMPB     (AP), #5                                    ; 0657
                                  03  1E  00076             BGEQU    8$                                          ;
                                00D2  31  00078 7$:         BRW      20$                                         ;
                              14  AC  D5  0007B 8$:         TSTL     20(AP)                                      ;
                                  F8  13  0007E             BEQL     7$                                          ;
                    08        63  A1  E8  00080 9$:         BLBS     99(PBCB), 10$                               ; 0680
                    50 00000000G  8F  D0  00084             MOVL     #SMG$_TRMNOTANS, R0                         ;
                                      04  0008B             RET                                                  ;
                                  04  DD  0008C 10$:        PUSHL    #4                                          ; 0699
```

```
                        00000000'  EF  9F 0008E          PUSHAB  INQUIRY                              ;
                 7E                04  7D 00094          MOVQ    #4, -(SP)                            ;
                 7E          FF    8F  9A 00097          MOVZBL  #255, -(SP)                          ;
                             24    AE  9F 0009B          PUSHAB  RESP_BUFFER                          ;
                                   7E  7C 0009E          CLRQ    -(SP)                                ;
                             28    AE  9F 000A0          PUSHAB  IOSB                                 ;
                 7E        48B7    8F  3C 000A3          MOVZWL  #18615, -(SP)                        ;
                 7E          64    A1  3C 000A8          MOVZWL  100(PBCB), -(SP)                     ;
                 7E          66    A1  9A 000AC          MOVZBL  102(PBCB), -(SP)                     ;
        00000000G 00               0C  FB 000B0          CALLS   #12, SYS$QIOW                        ;
                 04                50  E9 000B7          BLBC    STATUS, 11$                          ; 0701
                 50          08    AE  3C 000BA          MOVZWL  IOSB, STATUS                         ;
                 01                50  E8 000BE 11$:     BLBS    STATUS, 12$                          ; 0702
                                   04  000C1             RET                                          ;
                 50          12    AE  9E 000C2 12$:     MOVAB   RESP_BUFFER+2, R0                    ; 0713
                 52          0A    AE  3C 000C6          MOVZWL  IOSB+2, ROW_START                    ;
                 52                50  C0 000CA          ADDL2   R0, ROW_START                        ;
                 53          0E    AE  3C 000CD          MOVZWL  IOSB+6, R3                           ; 0715
                                   53  D7 000D1          DECL    R3                                   ;
          62     53                3B  3A 000D3          LOCC    #59, R3, (ROW_START)                 ;
                                   02  12 000D7          BNEQ    13$                                  ;
                                   51  D4 000D9          CLRL    R1                                   ;
                                   51  D5 000DB 13$:     TSTL    PTR                                  ; 0716
                                   72  13 000DD          BEQL    21$                                  ;
          58     51                52  C3 000DF          SUBL3   ROW_START, PTR, ROW_LEN              ; 0718
                 54          01    A1  9E 000E3          MOVAB   1(R1), COL_START                     ; 0720
          64     53          52    8F  3A 000E7          LOCC    #82, R3, (COL_START)                 ; 0721
                                   02  12 000EC          BNEQ    14$                                  ;
                                   51  D4 000EE          CLRL    R1                                   ;
                                   51  D5 000F0 14$:     TSTL    PTR                                  ; 0722
                                   5D  13 000F2          BEQL    21$                                  ;
          57     51                54  C3 000F4          SUBL3   COL_START, PTR, COL_LEN              ; 0724
                 51                52  D0 000F8          MOVL    ROW_START, SEQ                       ; 0730
                                   66  D4 000FB          CLRL    (R6)                                 ; 0731
                 53                01  CE 000FD          MNEGL   #1, I                                ; 0733
                                   0D  11 00100          BRB     16$                                  ;
          50     66                0A  C5 00102 15$:     MULL3   #10, (R6), R0                        ;
                 52        6341        9A 00106          MOVZBL  (I)[SEQ], R2                         ;
                 66     D0 A240        9E 0010A          MOVAB   -48(R2)[R0], (R6)                    ;
          EF     53                58  F2 0010F 16$:     AOBLSS  ROW_LEN, I, 15$                      ;
                 51                54  D0 00113          MOVL    COL_START, SEQ                       ; 0735
                                   65  D4 00116          CLRL    (R5)                                 ; 0736
          50                       01  CE 00118          MNEGL   #1, I                                ; 0738
                                   0D  11 0011B          BRB     18$                                  ;
          54     65                0A  C5 0011D 17$:     MULL3   #10, (R5), R4                        ;
                 53        6041        9A 00121          MOVZBL  (I)[SEQ], R3                         ;
                 65     D0 A344        9E 00125          MOVAB   -48(R3)[R4], (R5)                    ;
          EF     50                57  F2 0012A 18$:     AOBLSS  COL_LEN, I, 17$                      ;
  56      6C     08                00  ED 0012E          CMPZV   #0, #8, (AP), R6                     ; 0744
                                   09  19 00133          BLSS    19$                                  ;
                           6C46        D5 00135          TSTL    (AP)[R6]                             ;
                                   04  13 00138          BEQL    19$                                  ;
                 66          04    AE  D0 0013A          MOVL    TERM_ROW, (R6)                       ; 0745
  55      6C     08                00  ED 0013E 19$:     CMPZV   #0, #8, (AP), R5                     ; 0746
                                   08  19 00143          BLSS    20$                                  ;
                           6C45        D5 00145          TSTL    (AP)[R5]                             ;
                                   03  13 00148          BEQL    20$                                  ;
```

```
                65          6E  D0 0014A          MOVL    TERM_COL, (R5)                    ; 0747
                50          01  D0 0014D 20$:     MOVL    #1, R0                            ; 0751
                                04 00150          RET                                       ;
                            50  D4 00151 21$:     CLRL    R0                                ; 0753
                                04 00153          RET                                       ;

; Routine Size:  340 bytes,    Routine Base:  _SMG$CODE + 0098
```

```
1 %SBTTL 'SMG$SET_BROADCAST_TRAPPING'
1 GLOBAL ROUTINE SMG$SET_BROADCAST_TRAPPING(
1     PASTEBOARD_ID,AST_RTN,AST_ARG) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine enables trapping of messages broadcast
1 !       to the specified terminal.  If an AST routine is specified
1 !       the AST routine will be called whenever a broadcast
1 !       message comes in.  The AST routine will be called
1 !       with the first argument being the user's AST argument
1 !       (passed by value).  This can be the address of a structure
1 !       if the user needs to pass more than 32 bits of information.
1 !       If more than one terminal is involved, the user may wish
1 !       to pass along the pasteboard id as his AST argument.
1 !       The AST routine can get at the message by calling
1 !       SMG$GET_BROADCAST_MESSAGE.
1 !       If no AST routine is specified, operation is synchronous and
1 !       the user can get at broadcast messages by calling
1 !       SMG$GET_BROADCAST_MESSAGE whenever he wants to check for
1 !       receipt of a broadcast message.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       RET_STATUS.wlc.v = SMG$ENABLE_BROADCAST_TRAPPING (
1 !                               PASTEBOARD_ID.r[.r
1 !                               [, AST_RTN.szem.r]
1 !                               [, AST_ARG.rz.v])
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       PASTEBAORD_ID.rl.r      Address of pasteboard id
1 !
1 !       AST_RTN.szem.r          Address of AST routine to be called
1 !                               if a broadcast message comes in
1 !                               (optional)
1 !                               If omitted, and if the next argument is
1 !                               omitted, then no AST routine will
1 !                               get called.
1 !
1 !       AST_ARG.rz.v            optional value to be passed along to
1 !                               the AST routine
1 !                               If this is specified, but no AST_RTN
1 !                               argument is specified, then only the AST
1 !                               argument changes, but the routine remains
1 !                               the same.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       channel of associated mailbox
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       associated mailbox channel may be assigned if not
1 !       previous;y created.
1 !
```

```
1 ! COMPLETION STATUS:
1 !
1 !         SS$_NORMAL       Normal successful completion
1 !         SMG$_WRONUMARG   Wrong number of arguments
1 !         SMG$_NOT_A_TRM   Success, but device is not a terminal
1 !         LIB$_xyz         Errors from LIB$ASN_WTH_MBX
1 !         SS$_xyz          Any error possible from SMG$$SET_TERM_CHARACTERISTICS
1 !
1 ! SIDE EFFECTS:
1 !
1 !         A mailbox may be created.
1 !         The terminal characteristics get changed to BRDCSTMBOX
1 !         and NOBROADCAST. (The old values of these characteristics
1 !         are saved away so that they can be restored later.)
1 !
1 !--
```

```
2 BEGIN
2
2 LOCAL
2
2           STATUS,
2           PBCB            : REF BLOCK[,BYTE];     ! Pasteboard control block
2
2 BUILTIN
2
2           NULLPARAMETER;
2
2 EXTERNAL LITERAL
2
2           SMG$_NOT_A_TRM;         ! pasteboard device is not a terminal
```

```
2 $SMG$VALIDATE_ARGCOUNT(1,3);
2
2 $SMG$GET_PBCB(.PASTEBOARD_ID,PBCB);
2
2 !+
2 ! Make sure that the device is a terminal.
2 ! If not, return the qualified success, SMG$_NOT_A_TRM.
2 !-
2
2 IF .PBCB[PBCB_B_CLASS] NEQ DC$_TERM
2   THEN  RETURN  SMG$_NOT_A_TRM;
2
2 !+
2 ! If we don't already have a mailbox established,
2 ! then create a new channel to the terminal with an
2 ! associated mailbox.
2 !-
2
2 IF .PBCB[PBCB_W_ASYNC_CHAN] EQL 0
3   THEN  BEGIN
3         STATUS=INITIALIZE_MAILBOX(.PBCB);
3         IF NOT .STATUS THEN RETURN .STATUS
2         END;
2
2 !+
2 ! Note that broadcasts are enabled.
2 !-
2
2 PBCB[PBCB_V_BROADCAST]=1;
2
2 !+
2 ! Store the (possibly new) AST routine and argument in our PBCB.
2 !-
2
2 IF NULLPARAMETER(2)
3   THEN  BEGIN   ! no AST routine specified
3         IF NULLPARAMETER(3)
4           THEN  BEGIN   ! cancel ASTs
4                 PBCB[PBCB_A_BROADCAST_RTN]=0;
4                 PBCB[PBCB_L_BROADCAST_ARG]=0
4                 END     ! cancel ASTs
4           ELSE  BEGIN   ! change AST argument
4                 PBCB[PBCB_L_BROADCAST_ARG]=.AST_ARG
3                 END;    ! change AST argument
3         END     ! no AST routine specified
3   ELSE  BEGIN   ! New AST routine
3         PBCB[PBCB_A_BROADCAST_RTN]=.AST_RTN;
3         IF NULLPARAMETER(3)
3           THEN  PBCB[PBCB_L_BROADCAST_ARG]=0
3           ELSE  PBCB[PBCB_L_BROADCAST_ARG]=.AST_ARG
2         END;    ! New AST routine
2
2 RETURN  SS$_NORMAL
2
1 END;
```

```
                                                      .EXTRN  SMG$_NOT_A_TRM

                              0004 00000              .ENTRY  SMG$SET_BROADCAST_TRAPPING, Save R2              ; 0755
        50       6C         01  83 00002              SUBB3   #1, (AP), DIFF                                   ; 0841
                 02         50  91 00006              CMPB    DIFF, #2                                         ;
                            08  1B 00009              BLEQU   1$                                               ;
             50 00000000G   8F  D0 0000B              MOVL    #SMG$_WRONUMARG, R0                              ;
                            04  00012                 RET                                                      ;
             50       04    BC  D0 00013 1$:          MOVL    @PASTEBOARD_ID, R0                               ; 0843
                            11  19 00017              BLSS    2$                                               ;
      00000000G  00         50  D1 00019              CMPL    R0, PBD_L_COUNT                                  ;
                            08  14 00020              BGTR    2$                                               ;
   08 00000000G  00         50  E0 00022              BBS     R0, PBD_V_PB_AVAIL, 3$                           ;
             50 00000000G   8F  D0 0002A 2$:          MOVL    #SMG$_INVPAS_ID, R0                              ;
                            04  00031                 RET                                                      ;
             52 00000000G0040   D0 00032 3$:          MOVL    PBD_A_PBCB[R0], PBCB                             ;
        42       8F     58  A2  91 0003A              CMPB    88(PBCB), #66                                    ; 0850
                            08  13 0003F              BEQL    4$                                               ;
             50 00000000G   8F  D0 00041              MOVL    #SMG$_NOT_A_TRM, R0                              ; 0851
                            04  00048                 RET                                                      ;
                   00D2     C2  B5 00049 4$:          TSTW    210(PBCB)                                        ; 0859
                            0A  12 0004D              BNEQ    5$                                               ;
                            52  DD 0004F              PUSHL   PBCB                                             ; 0861
        0000V    CF         01  FB 00051              CALLS   #1, INITIALIZE_MAILBOX                           ;
                 3F         50  E9 00056              BLBC    STATUS, 12$                                      ; 0862
        00D0     C2         01  88 00059 5$:          BISB2   #1, 208(PBCB)                                    ; 0869
                 50  00BC   C2  9E 0005E              MOVAB   188(PBCB), R0                                    ; 0880
                 02         6C  91 00063              CMPB    (AP), #2                                         ; 0875
                            05  1F 00066              BLSSU   6$                                               ;
                     08     AC  D5 00068              TSTL    8(AP)                                            ;
                            10  12 0006B              BNEQ    8$                                               ;
                 03         6C  91 0006D 6$:          CMPB    (AP), #3                                         ; 0877
                            05  1F 00070              BLSSU   7$                                               ;
                     0C     AC  D5 00072              TSTL    12(AP)                                           ;
                            1A  12 00075              BNEQ    10$                                              ;
                   00B8     C2  D4 00077 7$:          CLRL    184(PBCB)                                        ; 0879
                            10  11 0007B              BRB     9$                                               ; 0880
        00B8     C2  08     AC  D0 0007D 8$:          MOVL    AST_RTN, 184(PBCB)                               ; 0887
                 03         6C  91 00083              CMPB    (AP), #3                                         ; 0888
                            05  1F 00086              BLSSU   9$                                               ;
                     0C     AC  D5 00088              TSTL    12(AP)                                           ;
                            04  12 0008B              BNEQ    10$                                              ;
                            60  D4 0008D 9$:          CLRL    (R0)                                             ; 0889
                            04  11 0008F              BRB     11$                                              ;
                 60  0C     AC  D0 00091 10$:         MOVL    AST_ARG, (R0)                                    ; 0890
                 50         01  D0 00095 11$:         MOVL    #1, R0                                           ; 0893
                            04  00098 12$:            RET                                                      ; 0895

; Routine Size:  153 bytes,    Routine Base:  _SMG$CODE + 01EC
```

```
%SBTTL 'INITIALIZE_MAILBOX';
ROUTINE INITIALIZE_MAILBOX(P_PBCB) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Creates a mailbox and assigns a channel to it.
!       Initializes a queue of mailbox blocks.
!
! CALLING SEQUENCE:
!
!       RET_STATUS.wlc.v = INITIALIZE_MAILBOX(P_PBCB.rl.v)
!
! FORMAL PARAMETERS:
!
!       P_PBCB.rl.v             Address of pasteboard control block.
!
! IMPLICIT INPUTS:
!
!       MAX_MSG
!       BUFQUO
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! COMPLETION STATUS:
!
!       SS$_NORMAL      Normal successful completion
!       SS$_xyz         Any error possible from $DASSGN or $CANCEL
!
! SIDE EFFECTS:
!
!       NONE
!--
```

```
2 BEGIN
2
2 BIND
2
2         PBCB    = .P_PBCB        : BLOCK[,BYTE];
2
2 LOCAL
2
2         STATUS,
2         NAME_DESC        : VECTOR[2];
2
2 EXTERNAL ROUTINE
2
2         LIB$GET_VM,
2         LIB$ASN_WTH_MBX;
```

```
!+
! LIB$ASN_WTH_MBX wants the name as a descriptor,
! so we build one temporarily.
!-

NAME_DESC[0]=.PBCB[PBCB_W_DEVNAM_LEN];
NAME_DESC[1]= PBCB[PBCB_T_DEVNAM];

STATUS=LIB$ASN_WTH_MBX(NAME_DESC,%REF(MAX_MSG),%REF(BUFQUO),
        PBCB[PBCB_W_ASYNC_CHAN],PBCB[PBCB_W_MBX_CHAN]);
IF NOT .STATUS THEN RETURN .STATUS;

!+
! Change the terminal characteristics to
! NOBROADCAST and BRDCSTMBX.
! Establish an exit handler to restore the old
! characteristics upon exit.
!-

STATUS=SMG$$SET_TERM_CHARACTERISTICS(PBCB[PBCB_L_PBID],
        TT$M_NOBRDCST,TT2$M_BRDCSTMBX);
IF NOT .STATUS THEN RETURN .STATUS;

!+
! Allocate a buffer to handle reads.
!-

PBCB[PBCB_W_SMGMBX_BUFSIZ]=MAX_MSG;
STATUS=LIB$GET_VM(%REF(.PBCB[PBCB_W_SMGMBX_BUFSIZ]),
                  PBCB[PBCB_A_SMGMBX_BUFFER]);
IF NOT .STATUS THEN RETURN .STATUS;

!+
! Initialize our queue of received mailbox buffers to empty.
!-

PBCB[PBCB_L_SMGMBX_FLINK]=PBCB[PBCB_L_SMGMBX_FLINK];
PBCB[PBCB_L_SMGMBX_BLINK]=PBCB[PBCB_L_SMGMBX_FLINK];

!+
! Invoke the AST routine with our special INIT flag on,
! so that it knows that no buffer was read
! and a read gets initiated.
!-

PBCB[PBCB_V_SMGMBX_INIT]=1;
STATUS=$DCLAST( ASTADR  = MAILBOX_AST_ROUTINE,
                ASTPRM  = PBCB);
IF NOT .STATUS THEN RETURN .STATUS;

RETURN  SS$_NORMAL

END;
```

```
                                        .EXTRN  LIB$GET_VM, LIB$ASN_WTH_MBX
```

```
                                                      .EXTRN  SYS$DCLAST

                                  0004 00000 INITIALIZE_MAILBOX:
                                                      .WORD   Save R2                              ; 0897
               5E           10     C2 00002           SUBL2   #16, SP
               52           04     AC D0 00005        MOVL    P_PBCB, R2                           ; 0935
        08     AE           12     A2 3C 00009        MOVZWL  18(R2), NAME_DESC                    ; 0951
        0C     AE           18     A2 9E 0000E        MOVAB   24(R2), NAME_DESC+4                  ; 0952
                          00D4     C2 9F 00013        PUSHAB  212(R2)                              ; 0955
                          00D2     C2 9F 00017        PUSHAB  210(R2)                              ;
        0C     AE         012C     8F 3C 0001B        MOVZWL  #300, 12(SP)                         ; 0954
                            0C     AE 9F 00021        PUSHAB  12(SP)                               ;
        0C     AE         012C     8F 3C 00024        MOVZWL  #300, 12(SP)                         ;
                            0C     AE 9F 0002A        PUSHAB  12(SP)                               ;
                            18     AE 9F 0002D        PUSHAB  NAME_DESC                            ;
 00000000G     00                  05 FB 00030        CALLS   #5, LIB$ASN_WTH_MBX                  ; 0955
               58                  50 E9 00037        BLBC    STATUS, 1$                           ; 0956
                                   10 DD 0003A        PUSHL   #16                                  ; 0965
                      00020000     8F DD 0003C        PUSHL   #131072                              ;
                            14     A2 9F 00042        PUSHAB  20(R2)                               ;
     0000V     CF                  03 FB 00045        CALLS   #3, SMG$$SET_TERM_CHARACTERISTICS    ;
               45                  50 E9 0004A        BLBC    STATUS, 1$                           ; 0967
      00D6     C2         012C     8F B0 0004D        MOVW    #300, 214(R2)                        ; 0973
                          00D8     C2 9F 00054        PUSHAB  216(R2)                              ; 0975
        08     AE         00D6     C2 3C 00058        MOVZWL  214(R2), 8(SP)                       ; 0974
                            08     AE 9F 0005E        PUSHAB  8(SP)                                ;
 00000000G     00                  02 FB 00061        CALLS   #2, LIB$GET_VM                       ; 0975
               27                  50 E9 00068        BLBC    STATUS, 1$                           ; 0976
               51         00C8     C2 9E 0006B        MOVAB   200(R2), R1                          ; 0982
               61                  51 D0 00070        MOVL    R1, (R1)                             ;
      00CC     C2                  51 D0 00073        MOVL    R1, 204(R2)                          ; 0983
      00D0     C2                  04 88 00078        BISB2   #4, 208(R2)                          ; 0991
                                   7E D4 0007D        CLRL    -(SP)                                ; 0993
                                   52 DD 0007F        PUSHL   R2                                   ;
                         0000V     CF 9F 00081        PUSHAB  MAILBOX_AST_ROUTINE                  ;
 00000000G     00                  03 FB 00085        CALLS   #3, SYS$DCLAST                       ;
               03                  50 E9 0008C        BLBC    STATUS, 1$                           ; 0994
               50                  01 D0 0008F        MOVL    #1, R0                               ; 0996
                                   04 00092 1$:       RET                                          ; 0998

; Routine Size:  147 bytes,    Routine Base:  _SMG$CODE + 0285
```

```
1 %SBTTL 'MAILOX_AST_ROUTINE';
1 ROUTINE MAILBOX_AST_ROUTINE(P_PBCB) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine processes messages that appear in our mailbox.
1 !       All it does is link the message into a message chain.
1 !       A user's AST routine may get called at this time.
1 !       A new block and buffer get created and a read for another
1 !       message is initiated.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       RET_STATUS.wlc.v = MAILBOX_AST_ROUTINE(P_PBCB.rl.v)
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       P_PBCB.rl.v             Address of pasteboard control block.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! COMPLETION STATUS:
1 !
1 !       SS$_NORMAL       Normal successful completion
1 !       SMG$_WRONUMARG   Wrong number of arguments
1 !       SS$_xyz          Any error possible from $DASSGN or $CANCEL
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !--
```

```
;  792      1037  2 BEGIN
;  793      1038  2
;  794      1039  2 LOCAL
;  795      1040  2
;  796      1041  2         SMGMBX          : REF $SMGMBX_DECL,
;  797      1042  2         STATUS;
;  798      1043  2
;  799      1044  2 BUILTIN
;  800      1045  2
;  801      1046  2         INSQUE;
;  802      1047  2
;  803      1048  2 BIND
;  804      1049  2
;  805      1050  2         PBCB    = .P_PBCB       : BLOCK[,BYTE];
;  806      1051  2
;  807      1052  2 EXTERNAL ROUTINE
;  808      1053  2
;  809      1054  2         LIB$GET_VM;
```

```
!+
! A read has just completed (except during initialization).
! The buffer and I/O status block for this read are
! in the PBCB.
! We look at the kind of message it is and deal with it accordingly.
!-

!+
! Create a new SMGMBX structure and link it in at the end.
!-

IF .PBCB[PBCB_V_SMGMBX_INIT]
  THEN  PBCB[PBCB_V_SMGMBX_INIT]=0
  ELSE  BEGIN   ! Copy message to our queue
        BIND BFR = .PBCB[PBCB_A_SMGMBX_BUFFER] : VECTOR[2,WORD];

        !+
        ! Do different things depending upon whether this is a
        ! broadcast message or an unsolicited input message.
        !-

        IF .BFR[0] EQL MSG$_TRMUNSOLIC
          THEN  BEGIN   ! Unsolicited input

                !+
                ! Call the user's AST routine, if one was specified.
                !-

                IF .PBCB[PBCB_A_UNSOLICIT_RTN] NEQ 0
                  THEN  BEGIN
                        BIND ROUTINE

                            BRTN    = .PBCB[PBCB_A_UNSOLICIT_RTN];

                        ! Ignore status from user's AST routine

                        BRTN(   .PBCB[PBCB_L_PBID],
                                .PBCB[PBCB_L_UNSOLICIT_ARG]);

                        END;

                END     ! Unsolicited input
          ELSE  BEGIN   ! broadcast message

                STATUS=LIB$GET_VM(%REF(SMGMBX_S_SIZE),SMGMBX);
                IF NOT .STATUS THEN SIGNAL(.STATUS);

                CH$FILL(0,SMGMBX_S_SIZE,.SMGMBX);
                INSQUE(.SMGMBX,.PBCB[PBCB_L_SMGMBX_BLINK]);

                !+
                ! Allocate a new buffer to receive mailbox messages.
                !-

                STATUS=LIB$GET_VM(%REF(.PBCB[PBCB_W_SMGMBX_BUFSIZ]),SMGMBX[SMGMBX_A_BUFFER]);
                IF NOT .STATUS THEN SIGNAL(.STATUS);

```

```
                                !+
                                ! Remember it's size (in case the default size in the PBCB changes.
                                !-

                                SMGMBX[SMGMBX_W_BUFSIZ]=.PBCB[PBCB_W_SMGMBX_BUFSIZ];

                                !+
                                ! Copy the returned message and it's I/O status block into
                                ! this structure.
                                !-

                                CH$MOVE(8,PBCB[PBCB_Q_SMGMBX_IOSB],SMGMBX[SMGMBX_Q_IOSB]);
                                CH$MOVE(.SMGMBX[SMGMBX_W_MSGLEN],
                                        .PBCB[PBCB_A_SMGMBX_BUFFER],
                                        .SMGMBX[SMGMBX_A_BUFFER]);

                                !+
                                ! Call the user's AST routine, if one was specified.
                                !-

                                IF .PBCB[PBCB_A_BROADCAST_RTN] NEQ 0
                                  THEN  BEGIN
                                        BIND ROUTINE

                                                BRTN    = .PBCB[PBCB_A_BROADCAST_RTN];

                                        ! Ignore status from user's AST routine

                                        BRTN(.PBCB[PBCB_L_BROADCAST_ARG]);

                                        END;

                                END     ! Broadcast message

                        END;    ! Copy message to our queue

!+
! Issue a read to the mailbox again so that our AST routine
! will wake up again the next time something comes in.
!-

STATUS=$QIO(    CHAN    = .PBCB[PBCB_W_MBX_CHAN],
                EFN     = .PBCB[PBCB_B_ASYNC_EFN],
                FUNC    =  IO$_READVBLK,
                IOSB    =  PBCB[PBCB_Q_SMGMBX_IOSB],
                P1      = .PBCB[PBCB_A_SMGMBX_BUFFER],
                P2      = .PBCB[PBCB_W_SMGMBX_BUFSIZ],
                ASTADR  =  MAILBOX_AST_ROUTINE,
                ASTPRM  =  PBCB);
IF NOT .STATUS THEN SIGNAL(.STATUS);

RETURN  SS$_NORMAL

END;
```

```
                                                          .EXTRN  SYS$QIO

                                          07FC 00000 MAILBOX_AST_ROUTINE:
                                                          .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10               ; 1000
                     5A 00000000G   00  9E 00002          MOVAB   LIB$GET_VM, R10                               ;
                     59 00000000G   00  9E 00009          MOVAB   LIB$SIGNAL, R9                                ;
                     5E             08  C2 00010          SUBL2   #8, SP                                        ;
                     56         04  AC  D0 00013          MOVL    P_PBCB, R6                                    ; 1050
          07     00D0  C6           02  E1 00017          BBC     #2, 208(R6), 1$                               ; 1066
                 00D0  C6           04  8A 0001D          BICB2   #4, 208(R6)                                   ; 1067
                                    18  11 00022          BRB     2$                                            ;
                     01       00D8  D6  B1 00024 1$:      CMPW    @216(R6), #1                                  ; 1076
                                    13  12 00029          BNEQ    3$                                            ;
                     50       00C0  C6  D0 0002B          MOVL    192(R6), R0                                   ; 1083
                                    72  13 00030          BEQL    6$                                            ;
                              00C4  C6  DD 00032          PUSHL   196(R6)                                       ; 1092
                                14  A6  DD 00036          PUSHL   20(R6)                                        ; 1091
                     60             02  FB 00039          CALLS   #2, (R0)                                      ;
                                    66  11 0003C 2$:      BRB     6$                                            ; 1076
                                04  AE  9F 0003E 3$:      PUSHAB  SMGMBX                                        ; 1099
                 04  AE             18  D0 00041          MOVL    #24, 4(SP)                                    ;
                                04  AE  9F 00045          PUSHAB  4(SP)                                         ;
                     6A             02  FB 00048          CALLS   #2, LIB$GET_VM                                ;
                     58             50  D0 0004B          MOVL    R0, STATUS                                    ;
                     05             58  E8 0004E          BLBS    STATUS, 4$                                    ; 1100
                                    58  DD 00051          PUSHL   STATUS                                        ;
                     69             01  FB 00053          CALLS   #1, LIB$SIGNAL                                ;
     18          00  6E             00  2C 00056 4$:      MOVC5   #0, (SP), #0, #24, @SMGMBX                    ; 1102
                                04  BE     0005B                                                                ;
               00CC  D6         04  BE  0E 0005D          INSQUE  @SMGMBX, @204(R6)                             ; 1103
                     57         04  AE  D0 00063          MOVL    SMGMBX, R7                                    ; 1109
                                08  A7  9F 00067          PUSHAB  8(R7)                                         ;
                 04  AE       00D6  C6  3C 0006A          MOVZWL  214(R6), 4(SP)                                ;
                                04  AE  9F 00070          PUSHAB  4(SP)                                         ;
                     6A             02  FB 00073          CALLS   #2, LIB$GET_VM                                ;
                     58             50  D0 00076          MOVL    R0, STATUS                                    ;
                     05             58  E8 00079          BLBS    STATUS, 5$                                    ; 1110
                                    58  DD 0007C          PUSHL   STATUS                                        ;
                     69             01  FB 0007E          CALLS   #1, LIB$SIGNAL                                ;
                 14  A7       00D6  C6  B0 00081 5$:      MOVW    214(R6), 20(R7)                               ; 1116
        0C   A7  00DC  C6           08  28 00087          MOVC3   #8, 220(R6), 12(R7)                           ; 1123
        08   B7  00D8  D6       0E  A7  28 0008E          MOVC3   14(R7), @216(R6), @8(R7)                      ; 1126
                     50       00B8  C6  D0 00096          MOVL    184(R6), R0                                   ; 1132
                                    07  13 0009B          BEQL    6$                                            ;
                              00BC  C6  DD 0009D          PUSHL   188(R6)                                       ; 1140
                     60             01  FB 000A1          CALLS   #1, (R0)                                      ;
                                    7E  7C 000A4 6$:      CLRQ    -(SP)                                         ; 1160
                                    7E  7C 000A6          CLRQ    -(SP)                                         ;
                     7E       00D6  C6  3C 000A8          MOVZWL  214(R6), -(SP)                                ;
                              00D8  C6  DD 000AD          PUSHL   216(R6)                                       ;
                                    56  DD 000B1          PUSHL   R6                                            ;
                              FF49  CF  9F 000B3          PUSHAB  MAILBOX_AST_ROUTINE                           ;
                              00DC  C6  9F 000B7          PUSHAB  220(R6)                                       ;
                                    31  DD 000BB          PUSHL   #49                                           ;
                     7E       00D4  C6  3C 000BD          MOVZWL  212(R6), -(SP)                                ;
                     7E         67  A6  9A 000C2          MOVZBL  103(R6), -(SP)                                ;
           00000000G 00             0C  FB 000C6          CALLS   #12, SYS$QIO                                  ;
```

```
         58          50 D0 000CD        MOVL    R0, STATUS                        ;
         05          58 E8 000D0        BLBS    STATUS, 7$                        ; 1161
                     58 DD 000D3        PUSHL   STATUS                            ;
         69          01 FB 000D5        CALLS   #1, LIB$SIGNAL                    ;
         50          01 D0 000D8 7$:    MOVL    #1, R0                            ; 1163
                        04 000DB        RET                                       ; 1165

; Routine Size:  220 bytes,    Routine Base:  _SMG$CODE + 0318
```

```
%SBTTL 'SMG$GET_BROADCAST_MESSAGE'
GLOBAL ROUTINE SMG$GET_BROADCAST_MESSAGE (
    PASTEBOARD_ID,P_MSG_DESC,P_MSG_LEN,P_SENDER_PID) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine asks if any broadcast messages have been
!       received.  If none have been received, it returns with
!       the sucess status SMG$_NO_MORMSG.  Normally, when a program
!       desires to process trapped messages, it would call this routine
!       repeatedly, handling each message it provided until
!       there were no more.
!
!       This routine may be called from either AST level or NON-AST level.
!
! CALLING SEQUENCE:
!
!       RET_STATUS.wlc.v = SMG$GET_BROADCAST_MESSAGE (
!                               PASTEBOARD_ID.rl.r
!                               [, P_MSG_DESC.wt.dx,
!                               [, [P_MSG_LEN.ww.r] ]
!                               [, [P_SENDER_PID])
!
! FORMAL PARAMETERS:
!
!       PASTEBOARD_ID           address of pastebaord associated
!                               with terminal in question
!
!       P_MSG_DESC              address of a descriptor for a
!                               buffer to receive the broadcast
!                               message.  If omitted, the broadcast
!                               message is discarded.
!
!       P_MSG_LEN               address of a word to receive the
!                               actual length of the received message.
!                               This is set to 0 if no messages were
!                               available.
!                               The value of this word is unpredictable
!                               if other errors occur.
!
!       P_SENDER_PID            address of a longword to receive the PID
!                               (process id) of the process that sent
!                               this broadcast message.  This may
!                               be 0 for certain system processes.
!                               This longword is not set if an
!                               error occurs.
!
! IMPLICIT INPUTS:
!
!       internal data storage holding previously received broadcast
!       messages.
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
```

```
1 ! COMPLETION STATUS:
1 !
1 !       SS$_NORMAL        Normal successful completion
1 !       SMG$_WRONUMARG    Wrong number of arguments
1 !       SMG$_NO_MORMSG    No more broadcast messages are available
1 !       LIB$_xyz          Any error possible from LIB$SCOPY_DX
1 !       SS$_xyz           Errors from mailbox driver (these errors
1 !                         may have occurred when we tried to read
1 !                         a message).
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
```

```
2 BEGIN
2
2 BIND
2
2           MSG_DESC         = .P_MSG_DESC   : BLOCK[,BYTE],
2           MSG_LEN          = .P_MSG_LEN    : WORD,
2           SENDER_PID       = .P_SENDER_PID : LONG;
2
2 EXTERNAL LITERAL
2
2           SMG$_NO_MORMSG;
2
2 EXTERNAL ROUTINE
2
2           LIB$SCOPY_R_DX,
2           LIB$GET_VM,
2           LIB$FREE_VM;
2
2 LOCAL
2
2           FOUND_FLAG,
2           FINAL_STATUS,
2           SMGMBX           : REF $SMGMBX_DECL,
2           PBCB             : REF BLOCK[,BYTE],
2 !         AST_STATUS,
2           STATUS;
2
2 BUILTIN
2
2           REMQUE,
2           NULLPARAMETER;
```

```
$SMG$VALIDATE_ARGCOUNT(1,4);

$SMG$GET_PBCB(.PASTEBOARD_ID,PBCB);

!+
! Note that we have not yet found a broadcast message.
!-

FOUND_FLAG=0;

!+
! Go look in our PBCB to see if any mailbox messages are present.
! If so, extract them off, one at a time, looking for a
! broadcast message.  Return it to the user if he supplied a buffer.
!-

!+
! Note that the SMGMBX message queue must always contain at least
! one block.  This is the block that the current read is attached to.
! We must never touch this block.  Note also that we need not
! disable ASTs at critical times to prevent our queue from
! getting messed up because INSQUE and REMQUE instructions
! are non-interruptable.
!-

WHILE 1 DO
        BEGIN   ! Look for message in queue

!       !+
!       ! Disable delivery of ASTs.
!       !-
!
!       AST_STATUS=$SETAST(ENBFLG=0);

!       IF .PBCB[PBCB_L_SMGMBX_FLINK] EQL .PBCB[PBCB_L_SMGMBX_BLINK]
!         THEN  EXITLOOP;

        !+
        ! Attempt to remove the first mailbox block from the queue.
        !-

        STATUS=REMQUE(.PBCB[PBCB_L_SMGMBX_FLINK],SMGMBX);

        !+
        ! If the queue was empty, then we merely exit the loop.
        ! This means that there are no broadcast messages available.
        !-

        IF .STATUS THEN EXITLOOP;

!       !+
!       ! Re-enable delivery of ASTs if they had previously been enabled.
!       !-
!
!       IF .AST_STATUS EQL SS$_WASSET
!         THEN  $SETAST(ENBFLG=1);

```

```
            !+
            ! Examine the removed block to see if it is a broadcast mailbox
            ! message.
            !-

            !+
            ! If the I/O status block shows that there was some error
            ! reading this message, then return that error now.
            ! Who knows? It may have been a broadcast message and we
            ! somehow lost it.
            !-

            IF NOT .SMGMBX[SMGMBX_W_STATUS]
              THEN  BEGIN   ! I/O error reading message
                    IF NOT NULLPARAMETER(3)
                      THEN  MSG_LEN=0;

                    FOUND_FLAG=1;

                    FINAL_STATUS=.SMGMBX[SMGMBX_W_STATUS]
                    END     ! I/O error reading message
              ELSE  BEGIN   ! Valid message

                    BIND

                            TYPE    = .SMGMBX[SMGMBX_A_BUFFER]      : WORD,
                            BRDMSG  = TYPE                          : VECTOR[,BYTE];

                    !+
                    ! Look at the type of the message.
                    ! This type is in the first word of the message.
                    ! If it is not a broadcast message, then we throw it away.
                    !-

                    IF .TYPE EQL MSG$_TRMBRDCST
                      THEN  BEGIN   ! Handle broadcast message

                            !+
                            ! I can't find any symbolic offsets for this message.
                            ! It is described in the I/O user's guide
                            ! under the section about terminals.
                            ! The broadcast message starts at offset 22
                            ! in this block.
                            ! Its length is stored in a word at offset 20.
                            !-

                            BIND

                                    BRDLEN  = BRDMSG[20]    : WORD,
                                    MESSAGE = BRDMSG[22]    : VECTOR[,BYTE];

                            !+
                            ! Note that we have found a broadcast message.
                            !-

                            FOUND_FLAG=1;
                            FINAL_STATUS=.SMGMBX[SMGMBX_W_STATUS];
```

```
                                !+
                                ! Return to the user the length of
                                ! the message (if he asked for it).
                                !-

                                IF NOT NULLPARAMETER(3)
                                  THEN  MSG_LEN=.BRDLEN;

                                !+
                                ! Return to the user, the PID of the process
                                ! that broadcast this message,
                                ! if he asked for it.
                                !-

                                IF NOT NULLPARAMETER(4)
                                  THEN  SENDER_PID=.SMGMBX[SMGMBX_L_SENDER];

                                !+
                                ! Return the message to the user
                                ! (using the semantics of his descriptor)
                                ! if he asked for it.
                                !-

                                IF NOT NULLPARAMETER(2)
                                  THEN  BEGIN   ! Return message

                                        !+
                                        ! Copy it to the user's buffer.
                                        !-

                                        STATUS=LIB$SCOPY_R_DX(
                                                BRDLEN,
                                                MESSAGE,
                                                MSG_DESC);
                                        IF NOT .STATUS THEN RETURN .STATUS

                                        END;    ! Return message

                                END;    ! Handle broadcast message

                        END;    ! Valid message

                !+
                ! Return the buffer to virtual memory.
                !-

                STATUS=LIB$FREE_VM(%REF(.SMGMBX[SMGMBX_W_BUFSIZ]),
                                   SMGMBX[SMGMBX_A_BUFFER]);
                IF NOT .STATUS THEN RETURN .STATUS;

                !+
                ! Return the mailbox block to virtual memory.
                !-

                STATUS=LIB$FREE_VM(%REF(SMGMBX_S_SIZE),SMGMBX);
                IF NOT .STATUS THEN RETURN .STATUS;
```

```
3
3           !+
3           ! If we had found a message, then we can return now.
3           !-
3
3           IF .FOUND_FLAG
3             THEN  RETURN  .FINAL_STATUS
3
2           END;     ! Look for message in queue
2
2   !+
2   ! No messages are left in the queue.
2   ! Return a status to the user telling him that no
2   ! broadcast messages are available.
2   !-
2
2   IF NOT NULLPARAMETER(3)
2     THEN  MSG_LEN=0;
2
2   RETURN  SMG$_NO_MORMSG
2
1   END;


                                                  .EXTRN   SMG$_NO_MORMSG, LIB$SCOPY_R_DX
                                                  .EXTRN   LIB$FREE_VM

                             07FC 00000           .ENTRY   SMG$GET_BROADCAST_MESSAGE, Save R2,R3,R4,-  ; 1167
                                                           R5,R6,R7,R8,R9,R10                         ;
         5A 00000000G   00   9E 00002             MOVAB    LIB$FREE_VM, R10                           ;
         5E             08   C2 00009             SUBL2    #8, SP                                     ;
         59        08   AC   D0 0000C             MOVL     P_MSG_DESC, R9                             ; 1242
         57        0C   AC   7D 00010             MOVQ     P_MSG_LEN, R7                              ; 1243
    50   6C             01   83 00014             SUBB3    #1, (AP), DIFF                             ; 1269
         03             50   91 00018             CMPB     DIFF, #3                                   ;
                        08   1B 0001B             BLEQU    1$                                         ;
         50 00000000G   8F   D0 0001D             MOVL     #SMG$_WRONUMARG, R0                        ;
                             04 00024             RET                                                 ;
         50        04   BC   D0 00025 1$:         MOVL     @PASTEBOARD_ID, R0                         ; 1271
                        11   19 00029             BLSS     2$                                         ;
    00000000G   00      50   D1 0002B             CMPL     R0, PBD_L_COUNT                            ;
                        08   14 00032             BGTR     2$                                         ;
 08 00000000G   00      50   E0 00034             BBS      R0, PBD_V_PB_AVAIL, 3$                     ;
         50 00000000G   8F   D0 0003C 2$:         MOVL     #SMG$_INVPAS_ID, R0                        ;
                             04 00043             RET                                                 ;
         53 00000000G0040    D0 00044 3$:         MOVL     PBD_A_PBCB[R0], PBCB                       ;
                        54   D4 0004C             CLRL     FOUND_FLAG                                 ; 1277
     04  AE        00C8      0F 0004E 4$:         REMQUE   @200(PBCB), SMGMBX                         ; 1310
                        51   DC 00054             MOVPSL   R1                                         ;
 51   51   02           01   EF 00056             EXTZV    #1, #2, R1, R1                             ;
         50             51   D0 0005B             MOVL     R1, STATUS                                 ;
         03             50   E9 0005E             BLBC     STATUS, 5$                                 ; 1317
                      0094   31 00061             BRW      12$                                        ;
         52        04   AE   D0 00064 5$:         MOVL     SMGMBX, R2                                 ; 1338
         15        0C   A2   E8 00068             BLBS     12(R2), 7$                                 ;
         03             6C   91 0006C             CMPB     (AP), #3                                   ; 1340
```

```
                                  07  1F 0006F       BLSSU   6$                                  ;
                         0C       AC  D5 00071       TSTL    12(AP)                              ;
                                  02  13 00074       BEQL    6$                                  ;
                                  67  B4 00076       CLRW    (R7)                                ; 1341
                    54            01  D0 00078 6$:   MOVL    #1, FOUND_FLAG                      ; 1343
                    56   0C       A2  3C 0007B       MOVZWL  12(R2), FINAL_STATUS                ; 1345
                                  4C  11 0007F       BRB     10$                                 ;
          0053      8F   08       B2  B1 00081 7$:   CMPW    @8(R2), #83                         ; 1360
                                  44  12 00087       BNEQ    10$                                 ;
     51       08    A2            14  C1 00089       ADDL3   #20, 8(R2), R1                      ; 1374
     55       08    A2            16  C1 0008E       ADDL3   #22, 8(R2), R5                      ; 1375
                    54            01  D0 00093       MOVL    #1, FOUND_FLAG                      ; 1381
                    56   0C       A2  3C 00096       MOVZWL  12(R2), FINAL_STATUS                ; 1382
                    03            6C  91 0009A       CMPB    (AP), #3                            ; 1389
                                  08  1F 0009D       BLSSU   8$                                  ;
                         0C       AC  D5 0009F       TSTL    12(AP)                              ;
                                  03  13 000A2       BEQL    8$                                  ;
                    67            61  B0 000A4       MOVW    (R1), (R7)                          ; 1390
                    04            6C  91 000A7 8$:   CMPB    (AP), #4                            ; 1398
                                  09  1F 000AA       BLSSU   9$                                  ;
                         10       AC  D5 000AC       TSTL    16(AP)                              ;
                                  04  13 000AF       BEQL    9$                                  ;
                    68   10       A2  D0 000B1       MOVL    16(R2), (R8)                        ; 1399
                    02            6C  91 000B5 9$:   CMPB    (AP), #2                            ; 1407
                                  13  1F 000B8       BLSSU   10$                                 ;
                         08       AC  D5 000BA       TSTL    8(AP)                               ;
                                  0E  13 000BD       BEQL    10$                                 ;
                       0222       8F  BB 000BF       PUSHR   #^M<R1,R5,R9>                       ; 1414
          00000000G 00            03  FB 000C3       CALLS   #3, LIB$SCOPY_R_DX                  ;
                    3E            50  E9 000CA       BLBC    STATUS, 14$                         ; 1418
                         08       A2  9F 000CD 10$:  PUSHAB  8(R2)                               ; 1431
              04    AE   14       A2  3C 000D0       MOVZWL  20(R2), 4(SP)                       ; 1430
                         04       AE  9F 000D5       PUSHAB  4(SP)                               ;
                    6A            02  FB 000D8       CALLS   #2, LIB$FREE_VM                     ; 1431
                    2D            50  E9 000DB       BLBC    STATUS, 14$                         ; 1432
                         04       AE  9F 000DE       PUSHAB  SMGMBX                              ; 1438
              04    AE            18  D0 000E1       MOVL    #24, 4(SP)                          ;
                         04       AE  9F 000E5       PUSHAB  4(SP)                               ;
                    6A            02  FB 000E8       CALLS   #2, LIB$FREE_VM                     ;
                    1D            50  E9 000EB       BLBC    STATUS, 14$                         ; 1439
                    03            54  E8 000EE       BLBS    FOUND_FLAG, 11$                     ; 1445
                                FF5A  31 000F1       BRW     4$                                  ;
                    50            56  D0 000F4 11$:  MOVL    FINAL_STATUS, R0                    ; 1446
                                      04 000F7       RET                                         ;
                    03            6C  91 000F8 12$:  CMPB    (AP), #3                            ; 1456
                                  07  1F 000FB       BLSSU   13$                                 ;
                         0C       AC  D5 000FD       TSTL    12(AP)                              ;
                                  02  13 00100       BEQL    13$                                 ;
                                  67  B4 00102       CLRW    (R7)                                ; 1457
                    50 00000000G  8F  D0 00104 13$:  MOVL    #SMG$_NO_MORMSG, R0                 ; 1459
                                      04 0010B 14$:  RET                                         ; 1461

; Routine Size:  268 bytes,    Routine Base:  _SMG$CODE + 03F4
```

```
%SBTTL 'SMG$$OUT_OF_BAND_HANDLER - Generic out-of-band handler'
GLOBAL ROUTINE SMG$$OUT_OF_BAND_HANDLER (
        ARG, R0, R1, PC, PSL ) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine processes an out-of-band AST.
!       It runs at AST level.  It logically gets called when the
!       user types an out-of-band character.
!       Actually, another routine actually gets called first
!       and then calls this one with the same argument list.
!
! CALLING SEQUENCE:
!
!       RET_STATUS.wlc.v = SMG$$OUT_OF_BAND_HANDLER (
!                               ARG.rz.v,
!                               R0.rl.v
!                               R1.rl.v,
!                               PC.ra.v,
!                               PSL.rl.v)
!
! FORMAL PARAMETERS:
!
!       ARG             character that was typed
!       R0              register 0 at time of AST
!       R1              register 1 at time of AST
!       PC              PC when AST occurred
!       PSL             contents of PSL at time of AST
!
! IMPLICIT INPUTS:
!
!       Return PC on the stack.
!       We use this to determine who called us.
!       The address of the calling procedure is used to
!       enable us to locate the PBCB associated with the
!       terminal that the out-of-band character was typed on.
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! COMPLETION STATUS:
!
!       SS$_NORMAL      Normal successful completion
!
! SIDE EFFECTS:
!
!--
```

```
BEGIN

LOCAL

        NEXT_PC,                                ! return PC address
                                                ! should point into PBCB
        PBCB : REF BLOCK[,BYTE],                ! Gets the address of our PBCB
        PBID,                                   ! Gets our pasteboard id.
        BAND_INFO_BLOCK : BLOCK[SMG$S_BAND_INFORMATION_TABLE,BYTE],
                                                ! Band information table
        STATUS;                                 ! Local status

EXTERNAL

        PBD_A_PBCB      : VOLATILE VECTOR;      ! pasteboard directory

BUILTIN

        FP;                             ! frame pointer

EXTERNAL LITERAL

        SMG$_FATERRLIB;

BIND    FRAME   = .FP   : BLOCK[,BYTE];         ! Stack frame

!+
! Macro $GET_FIELD_OFFSET takes a field reference and replaces
! it in the text stream by the offset represented by that reference.
! (the first of the 4 items in the block structure reference).
! This macro works whether the actual argument is a true
! BLISS field name or if it is an SDL macro name.
! The way it tells the two apart is this: A field name will
! be a single lexeme, while a macro as an actual argument will
! be expanded before invocation and will appear as 4 arguments.
!-

MACRO

    $GET_FIELD_OFFSET(FIELD_REFERENCE) =

        %IF %LENGTH EQL 1

          %THEN %FIELDEXPAND(FIELD_REFERENCE,1)

          %ELSE FIELD_REFERENCE

        %FI

    %;

!+
! Find out where this routine is going to return to.
! That should be the RET at offset PBCB_B_RET in the AST
! routine that we built in our PBCB.
!-
```

```
NEXT_PC=.FRAME[SF$L_SAVE_PC];

!+
! By subtracting the offset of "PBCB_B_RET" we get the
! address of the start of the PBCB for the terminal that
! the out-of-band character was typed on.  We have to go through
! these machinations because there is no other was to tell
! which terminal caused the AST to trigger.
!-

PBCB=.NEXT_PC - $GET_FIELD_OFFSET(PBCB_B_RET);

!+
! Perform some validity checks to make sure that this is
! really the address of a PBCB.  If not, we just return
! without ever calling the user's AST routine.
!-

IF .PBCB[PBCB_B_RET] NEQ %X'04' THEN    RETURN  SMG$_FATERRLIB;
IF .PBCB[PBCB_B_ABS] NEQ %X'9F' THEN    RETURN  SMG$_FATERRLIB;

PBID=.PBCB[PBCB_L_PBID];
IF .PBID GTRU PBD_K_MAX_PB       THEN    RETURN  SMG$_FATERRLIB;
IF .PBD_A_PBCB[.PBID] NEQ .PBCB THEN    RETURN  SMG$_FATERRLIB;

!+
! Call the user's AST routine as if the system were calling it
! directly.  The difference is, though, that the first
! argument will be the address of an SMG BAND INFORMATION BLOCK,
! instead of just the character that was typed.
! If somehow no user routine was specified, we just return also.
!-

IF .PBCB[PBCB_A_BAND_ROUTINE] EQL 0
   THEN  RETURN  SMG$_FATERRLIB
   ELSE  BEGIN
         BIND ROUTINE

                 USER_RTN = .PBCB[PBCB_A_BAND_ROUTINE];

         !+
         ! Set up the band information block.
         !-

         BAND_INFO_BLOCK[SMG$L_PASTEBOARD_ID] = .PBID;
         BAND_INFO_BLOCK[SMG$L_ARG]           = .PBCB[PBCB_L_BAND_AST_ARG];
         BAND_INFO_BLOCK[SMG$L_CHARACTER]     = '    ';
         BAND_INFO_BLOCK[SMG$B_CHARACTER]     = .ARG;

         STATUS=USER_RTN( BAND_INFO_BLOCK,.R0,.R1,.PC,.PSL );
         IF NOT .STATUS THEN RETURN .STATUS
         END;

RETURN  SS$_NORMAL

END;
```

```
                                                       .EXTRN  SMG$_FATERRLIB

                           0004 00000                  .ENTRY  SMG$$OUT_OF_BAND_HANDLER, Save R2           ; 1463
           5E              0C  C2 00002                SUBL2   #12, SP                                     ;
           50        1C    AE  D0 00005                MOVL    28(FP), NEXT_PC                             ; 1568
           50      FF6B    C0  9E 00009                MOVAB   -149(R0), PBCB                              ; 1578
           04      0095    C0  91 0000E                CMPB    149(PBCB), #4                               ; 1586
                           22  12 00013                BNEQ    1$                                          ;
     9F    8F      0090    C0  91 00015                CMPB    144(PBCB), #159                             ; 1587
                           1A  12 0001B                BNEQ    1$                                          ;
           51        14    A0  D0 0001D                MOVL    20(PBCB), PBID                              ; 1589
           10              51  D1 00021                CMPL    PBID, #16                                   ; 1590
                           11  1A 00024                BGTRU   1$                                          ;
           50 00000000G0041    D1 00026                CMPL    PBD_A_PBCB[PBID], PBCB                      ; 1591
                           07  12 0002E                BNEQ    1$                                          ;
           52      0098    C0  D0 00030                MOVL    152(PBCB), R2                               ; 1601
                           08  12 00035                BNEQ    2$                                          ;
           50 00000000G    8F  D0 00037 1$:            MOVL    #SMG$_FATERRLIB, R0                         ; 1602
                               04 0003E                RET                                                 ;
           6E              51  D0 0003F 2$:            MOVL    PBID, BAND_INFO_BLOCK                       ; 1612
     04    AE      009C    C0  D0 00042                MOVL    156(PBCB), BAND_INFO_BLOCK+4                ; 1613
     08    AE  20202020    8F  D0 00048                MOVL    #538976288, BAND_INFO_BLOCK+8               ; 1614
     08    AE        04    AC  90 00050                MOVB    ARG, BAND_INFO_BLOCK+8                      ; 1615
           7E        10    AC  7D 00055                MOVQ    PC, -(SP)                                   ; 1617
           7E        08    AC  7D 00059                MOVQ    R0, -(SP)                                   ;
                     10    AE  9F 0005D                PUSHAB  BAND_INFO_BLOCK                             ;
           62              05  FB 00060                CALLS   #5, (R2)                                    ;
           03              50  E9 00063                BLBC    STATUS, 3$                                  ; 1618
           50              01  D0 00066                MOVL    #1, R0                                      ; 1621
                               04 00069 3$:            RET                                                 ; 1623

; Routine Size:  106 bytes,    Routine Base:  _SMG$CODE + 0500
```

```
1 %SBTTL 'SMG$SET_OUT_OF_BAND_ASTS'
1 GLOBAL ROUTINE SMG$SET_OUT_OF_BAND_ASTS (
1               PASTEBOARD_ID,P_CHAR_MASK,AST_RTN,AST_ARG ) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine enables new out-of-band characters to
1 !       be accepted from the specified terminal.  If one of these
1 !       characters is typed, the specified AST routine gets called.
1 !       See the RTL manual for conventions on how AST routines get called.
1 !       The first argument to the AST routine is the address of a
1 !       band information table.  This table contains things like
1 !       the pasteboard id and the character that was typed.
1 !       The format and structure of this table is described in SMGDEF.SDL.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       RET_STATUS.wlc.v = SMG$ENABLE_OUT_OF_BAND_ASTS (
1 !                               PASTEBOARD_ID.rl.r,
1 !                               P_CHAR_MASK.rl.r,
1 !                               AST_RTN.szem.r
1 !                               [, AST_ARG.rz.v])
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       PASTEBOARD_ID   Pasteboard ID for terminal to be affected.
1 !
1 !       P_CHAR_MASK     Longword mask of which control characters
1 !                       to be the new out-of-band control characters.
1 !
1 !       AST_RTN         Address of AST routine to be called
1 !
1 !       AST_ARG         Optional value to be passed along to the AST routine.
1 !                       It will appear as the second longword in the
1 !                       band information table.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       Structure holding information about out-of-band AST routines
1 !       gets updated.
1 !
1 ! COMPLETION STATUS:
1 !
1 !       SS$_NORMAL      Normal successful completion
1 !       SMG$_WRONUMARG  Wrong number of arguments
1 !       SS$_xyz         Any error possible from $QIOW
1 !
1 ! SIDE EFFECTS:
1 !
1 !--
```

```
BEGIN

BIND

        CHAR_MASK       = .P_CHAR_MASK;

LOCAL

        PBCB            : REF BLOCK[,BYTE],     ! Pasteboard control block
        MASK_BLOCK      : VECTOR[2],            ! Character mask block
                                                ! See figure 9-4 in I/O manual.
        STATUS;

BUILTIN

        ACTUALCOUNT;

$SMG$VALIDATE_ARGCOUNT(3,4);

!+
!       Get the PBCB associated with this id and store it in PBCB.
!-

$SMG$GET_PBCB(.PASTEBOARD_ID,PBCB);

!+
!       Store the address of the user's AST routine in the PBCB.
!-

PBCB[PBCB_A_BAND_ROUTINE] = .AST_RTN;

!+
!       Store his argument, if one was specified, or 0 if not.
!-

IF ACTUALCOUNT() EQL 4
  THEN  PBCB[PBCB_L_BAND_AST_ARG]=.AST_ARG
  ELSE  PBCB[PBCB_L_BAND_AST_ARG]= 0;

!+
!       Store away the mask for goodness sake in the PBCB
!       and in our terminator mask block.
!       The first longword of the terminator mask block must be 0
!       indicating that it is a short-form mask.
!-

PBCB[PBCB_M_BAND_MASK]=.CHAR_MASK;
MASK_BLOCK[0]=0;
MASK_BLOCK[1]=.CHAR_MASK;

!+
! Issue the appropriate QIO.
! The AST routine to service the out-of-band character has
! previously been built in the PBCB at offset PBCB_Z_OUT_OF_BAND_RTN.
! This handler actually resides within the PBCB.
! It is of the form:
!
```

```
; 1499        1736  2 !        0000    entry mask
; 1500        1737  2 !          FA    CALLG
; 1501        1738  2 !          6C    (AP)
; 1502        1739  2 !          9F    absolute addressing
; 1503        1740  2 !     address    longword address of SMG$$OUT_OF_BAND_HANDLER
; 1504        1741  2 !          04    RET
; 1505        1742  2 !-
; 1506        1743  2
; 1507      P 1744  2 STATUS=$QIOW(    CHAN    =  .PBCB[PBCB_W_CHAN],
; 1508      P 1745  2                  EFN     =  .PBCB[PBCB_B_ASYNC_EFN],
; 1509      P 1746  2                  FUNC    = IO$_SETMODE OR IO$M_OUTBAND,
; 1510      P 1747  2                  P1      = PBCB[PBCB_Z_OUT_OF_BAND_RTN],
; 1511        1748  2                  P2      = MASK_BLOCK);
; 1512        1749  2
; 1513        1750  2 RETURN .STATUS
; 1514        1751  2
; 1515        1752  1 END;



                                         0000 00000          .ENTRY   SMG$SET_OUT_OF_BAND_ASTS, Save nothing      ; 1625
                    5E                08   C2 00002          SUBL2    #8, SP                                      ;
          50        6C                03   83 00005          SUBB3    #3, (AP), DIFF                              ; 1696
                    01                50   91 00009          CMPB     DIFF, #1                                    ;
                                      08   1B 0000C          BLEQU    1$                                          ;
                    50 00000000G      8F   D0 0000E          MOVL     #SMG$_WRONUMARG, R0                         ;
                                      04      00015          RET                                                  ;
                    50          04    BC   D0 00016 1$:      MOVL     @PASTEBOARD_ID, R0                          ; 1702
                                      11   19 0001A          BLSS     2$                                          ;
             00000000G  00            50   D1 0001C          CMPL     R0, PBD_L_COUNT                             ;
                                      08   14 00023          BGTR     2$                                          ;
          08 00000000G  00            50   E0 00025          BBS      R0, PBD_V_PB_AVAIL, 3$                      ;
                    50 00000000G      8F   D0 0002D 2$:      MOVL     #SMG$_INVPAS_ID, R0                         ;
                                      04      00034          RET                                                  ;
                    50 00000000G0040       D0 00035 3$:      MOVL     PBD_A_PBCB[R0], PBCB                        ;
               0098 C0          0C    AC   D0 0003D          MOVL     AST_RTN, 152(PBCB)                          ; 1708
                    04                6C   91 00043          CMPB     (AP), #4                                    ; 1714
                                      08   12 00046          BNEQ     4$                                          ;
               009C C0          10    AC   D0 00048          MOVL     AST_ARG, 156(PBCB)                          ; 1715
                                      04   11 0004E          BRB      5$                                          ;
                              009C    C0   D4 00050 4$:      CLRL     156(PBCB)                                   ; 1716
               00A0 C0          08    BC   D0 00054 5$:      MOVL     @P_CHAR_MASK, 160(PBCB)                     ; 1725
                                      6E   D4 0005A          CLRL     MASK_BLOCK                                  ; 1726
                 04 AE          08    BC   D0 0005C          MOVL     @P_CHAR_MASK, MASK_BLOCK+4                  ; 1727
                                      7E   7C 00061          CLRQ     -(SP)                                       ; 1748
                                      7E   7C 00063          CLRQ     -(SP)                                       ;
                                10    AE   9F 00065          PUSHAB   MASK_BLOCK                                  ;
                              008C    C0   9F 00068          PUSHAB   140(PBCB)                                   ;
                                      7E   7C 0006C          CLRQ     -(SP)                                       ;
                                      7E   D4 0006E          CLRL     -(SP)                                       ;
                    7E        0423    8F   3C 00070          MOVZWL   #1059, -(SP)                                ;
                    7E          64    A0   3C 00075          MOVZWL   100(PBCB), -(SP)                            ;
                    7E          67    A0   9A 00079          MOVZBL   103(PBCB), -(SP)                            ;
             00000000G  00            0C   FB 0007D          CALLS    #12, SYS$QIOW                               ;
                                      04      00084          RET                                                  ; 1752
```

; Routine Size: 133 bytes, Routine Base: _SMG$CODE + 056A

```
1 %SBTTL 'SMG$ENABLE_UNSOLICITED_INPUT'
1 GLOBAL ROUTINE SMG$ENABLE_UNSOLICITED_INPUT(
1           PASTEBOARD_ID,AST_RTN,AST_ARG) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !         This routine enables calling of an AST routine upon
1 !         receipt of unsolicited input on the specified terminal.
1 !         The AST routine gets called with 2 arguments:
1 !         PASTEBOARD_ID.rl.r and AST_ARG.rz.v.  Note that no input
1 !         characters are read. It is up to the user to read the
1 !         unsolicited input using normal SMG input routines
1 !         (or cancel the type-ahead).
1 !
1 ! CALLING SEQUENCE:
1 !
1 !         RET_STATUS.wlc.v = SMG$ENABLE_UNSOLICITED_INPUT(
1 !                                   PASTEBOARD_ID.rl.r,
1 !                                   AST_RTN.szem.r
1 !                                   [, AST_ARG.rz.v])
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !         PASTEBOARD_ID   Address of pasteboard id for terminal to
1 !                         be affected.
1 !
1 !         AST_RTN         Address of AST routine to be called
1 !
1 !         AST_ARG         Optional value to be passed to AST routine.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !         NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !         Internal tables get updated.
1 !
1 ! COMPLETION STATUS:
1 !
1 !         SS$_NORMAL      Normal successful completion
1 !         SMG$_WRONUMARG  Wrong number of arguments
1 !         SS$_xyz         Any error possible from $QIOW
1 !
1 ! SIDE EFFECTS:
1 !
1 !--
```

```
2 BEGIN
2
2 BUILTIN
2
2       NULLPARAMETER;
2
2 LOCAL
2
2       PBCB            : REF $PBCB_DECL,
2       STATUS;
2
2 EXTERNAL LITERAL
2
2       SMG$_NOT_A_TRM;
2
2 $SMG$VALIDATE_ARGCOUNT(2,3);
2
2 $SMG$GET_PBCB(.PASTEBOARD_ID,PBCB);
2
2 !+
2 ! Make sure that the device is a terminal.
2 ! If not, return the qualified success, SMG$_NOT_A_TRM.
2 !-
2
2 IF .PBCB[PBCB_B_CLASS] NEQ DC$_TERM
2   THEN  RETURN  SMG$_NOT_A_TRM;
2
2 !+
2 ! If we don't already have a mailbox established,
2 ! then create a new channel to the terminal with an
2 ! associated mailbox.
2 !-
2
2 IF .PBCB[PBCB_W_ASYNC_CHAN] EQL 0
3   THEN  BEGIN
3         STATUS=INITIALIZE_MAILBOX(.PBCB);
3         IF NOT .STATUS THEN RETURN .STATUS
2         END;
2
2 !+
2 ! Note that unsolicted input is enabled.
2 !-
2
2 PBCB[PBCB_V_UNSOLICIT]=1;
2
2 !+
2 ! Store the (possibly new) AST routine and argument in our PBCB.
2 !-
2
2 PBCB[PBCB_A_UNSOLICIT_RTN]=.AST_RTN;
2 IF NULLPARAMETER(AST_ARG)
2   THEN  PBCB[PBCB_L_UNSOLICIT_ARG]=0
2   ELSE  PBCB[PBCB_L_UNSOLICIT_ARG]=.AST_ARG;
2
2 RETURN  SS$_NORMAL
2
1 END;
```

```
                                  0004 00000        .ENTRY  SMG$ENABLE_UNSOLICITED_INPUT, Save R2     ; 1754
   50            6C          02    83 00002         SUBB3   #2, (AP), DIFF                            ; 1816
                 01          50    91 00006         CMPB    DIFF, #1                                  ;
                             08    1B 00009         BLEQU   1$                                        ;
                 50 00000000G      8F    D0 0000B   MOVL    #SMG$_WRONUMARG, R0                       ;
                             04    00012            RET                                               ;
                 50          04    BC    D0 00013 1$:  MOVL  @PASTEBOARD_ID, R0                       ; 1818
                             11    19 00017         BLSS    2$                                        ;
       00000000G 00          50    D1 00019         CMPL    R0, PBD_L_COUNT                           ;
                             08    14 00020         BGTR    2$                                        ;
   08 00000000G  00          50    E0 00022         BBS     R0, PBD_V_PB_AVAIL, 3$                    ;
                 50 00000000G      8F    D0 0002A 2$:  MOVL  #SMG$_INVPAS_ID, R0                      ;
                             04    00031            RET                                               ;
                 52 00000000G0040  D0 00032 3$:     MOVL    PBD_A_PBCB[R0], PBCB                      ;
          42     8F          58    A2    91 0003A   CMPB    88(PBCB), #66                             ; 1825
                             08    13 0003F         BEQL    4$                                        ;
                 50 00000000G      8F    D0 00041   MOVL    #SMG$_NOT_A_TRM, R0                       ; 1826
                             04    00048            RET                                               ;
                       00D2  C2    B5 00049 4$:     TSTW    210(PBCB)                                 ; 1834
                             0A    12 0004D         BNEQ    5$                                        ;
                             52    DD 0004F         PUSHL   PBCB                                      ; 1836
        FC40     CF          01    FB 00051         CALLS   #1, INITIALIZE_MAILBOX                    ;
                 24          50    E9 00056         BLBC    STATUS, 9$                                ; 1837
        00D0     C2          02    88 00059 5$:     BISB2   #2, 208(PBCB)                             ; 1844
        00C0     C2          08    AC    D0 0005E   MOVL    AST_RTN, 192(PBCB)                        ; 1850
                 03          6C    91 00064         CMPB    (AP), #3                                  ; 1851
                             05    1F 00067         BLSSU   6$                                        ;
                       0C    AC    D5 00069         TSTL    12(AP)                                    ;
                             06    12 0006C         BNEQ    7$                                        ;
                       00C4  C2    D4 0006E 6$:     CLRL    196(PBCB)                                 ; 1852
                             06    11 00072         BRB     8$                                        ;
        00C4     C2    0C    AC    D0 00074 7$:     MOVL    AST_ARG, 196(PBCB)                        ; 1853
                 50          01    D0 0007A 8$:     MOVL    #1, R0                                    ; 1855
                             04    0007D 9$:        RET                                               ; 1857

; Routine Size:  126 bytes,    Routine Base:  _SMG$CODE + 05EF
```

```
1 %SBTTL 'SMG$DISABLE_UNSOLICITED_INPUT'
1 GLOBAL ROUTINE SMG$DISABLE_UNSOLICITED_INPUT(
1         PASTEBOARD_ID) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine disables unsolicited input ASTs for the
1 !       specified terminal.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       RET_STATUS.wlc.v = SMG$DISABLE_UNSOLICITED_INPUT(
1 !                               PASTEBOARD_ID.rl.r)
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       PASTEBOARD_ID   Pasteboard id for terminal to
1 !                       be affected.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       Internal tables.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       Internal tables get updated.
1 !
1 ! COMPLETION STATUS:
1 !
1 !       SS$_NORMAL      Normal successful completion
1 !       SMG$_WRONUMARG  Wrong number of arguments
1 !       SS$_xyz         Any error possible from $QIOW
1 !
1 ! SIDE EFFECTS:
1 !
1 !--
```

```
2 BEGIN
2
2 LOCAL
2
2         PBCB            : REF $PBCB_DECL,
2         STATUS;
2
2 EXTERNAL LITERAL
2
2         SMG$_NOT_A_TRM;
2
2 $SMG$VALIDATE_ARGCOUNT(1,1);
2
2 $SMG$GET_PBCB(.PASTEBOARD_ID,PBCB);
2
2 !+
2 ! Make sure that the device is a terminal.
2 ! If not, return the qualified success, SMG$_NOT_A_TRM.
2 !-
2
2 IF .PBCB[PBCB_B_CLASS] NEQ DC$_TERM
2   THEN  RETURN  SMG$_NOT_A_TRM;
2
2 !+
2 ! Note that unsolicted input is disabled.
2 !-
2
2 PBCB[PBCB_V_UNSOLICIT]=0;
2 PBCB[PBCB_A_UNSOLICIT_RTN]=0;
2 PBCB[PBCB_L_UNSOLICIT_ARG]=0;
2
2 RETURN  SS$_NORMAL
2
1 END;
```

```
                                 0000 00000       .ENTRY  SMG$DISABLE_UNSOLICITED_INPUT, Save nothing ; 1859
           01               6C  91 00002          CMPB    (AP), #1                                    ; 1906
                            08  13 00005          BEQL    1$                                          ;
           50 00000000G     8F  D0 00007          MOVL    #SMG$_WRONUMARG, R0                         ;
                                04 0000E          RET                                                 ;
           50         04    BC  D0 0000F 1$:      MOVL    @PASTEBOARD_ID, R0                          ; 1908
                            11  19 00013          BLSS    2$                                          ;
     00000000G     00       50  D1 00015          CMPL    R0, PBD_L_COUNT                             ;
                            08  14 0001C          BGTR    2$                                          ;
  08 00000000G     00       50  E0 0001E          BBS     R0, PBD_V_PB_AVAIL, 3$                      ;
           50 00000000G     8F  D0 00026 2$:      MOVL    #SMG$_INVPAS_ID, R0                         ;
                                04 0002D          RET                                                 ;
           50 00000000G0040     D0 0002E 3$:      MOVL    PBD_A_PBCB[R0], PBCB                        ;
      42   8F         58    A0  91 00036          CMPB    88(PBCB), #66                               ; 1915
                            08  13 0003B          BEQL    4$                                          ;
           50 00000000G     8F  D0 0003D          MOVL    #SMG$_NOT_A_TRM, R0                         ; 1916
                                04 00044          RET                                                 ;
         00D0  C0           02  8A 00045 4$:      BICB2   #2, 208(PBCB)                               ; 1922
```

```
                                    00C0    C0  7C 0004A          CLRQ    192(PBCB)                                      ; 1923
                            50              01  D0 0004E          MOVL    #1, R0                                         ; 1926
                                                04 00051          RET                                                    ; 1928

; Routine Size:  82 bytes,    Routine Base:  _SMG$CODE + 066D
```

```
%SBTTL 'SMG$$SET_TERM_CHARACTERISTICS'
GLOBAL ROUTINE SMG$$SET_TERM_CHARACTERISTICS (
        PBID, ON_CHARACTERISTICS1, ON_CHARACTERISTICS2,
        OFF_CHARACTERISTICS1, OFF_CHARACTERISTICS2,
        P_HANDLE,
        P_OLD_CHARACTERISTICS1, P_OLD_CHARACTERISTICS2 ) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine changes the terminal characteristics
!       for a given pasteboard.
!       It also establishes an exit handler that will restore
!       the characteristics back to what they were on image exit.
!
! CALLING SEQUENCE:
!
!       RET_STATUS.wlc.v = SMG$$SET_TERM_CHARACTERISTICS (
!                               PBID.rl.r
!                               [,ON_CHARACTERISTICS1.rl.v]
!                               [,ON_CHARACTERISTICS2.rl.v]
!                               [,OFF_CHARACTERISTICS1.rl.v]
!                               [,OFF_CHARACTERISTICS2.rl.v]
!                               [,P_HANDLE.wl.r]
!                               [,P_OLD_CHARACTERISTICS1.wl.r]
!                               [,P_OLD_CHARACTERISTICS2.wl.r])
!
! FORMAL PARAMETERS:
!
!       PBID.rl.r                       pasteboard id
!       ON_CHARACTERITSICS1.rl.v        bits to turn on in 1st characteristics
!       ON_CHARACTERITSICS2.rl.v        bits to turn on in 2nd characteristics
!       OFF_CHARACTERITSICS1.rl.v       bits to turn off in 1st characteristics
!       OFF_CHARACTERITSICS2.rl.v       bits to turn off in 2nd characteristics
!       P_HANDLE.wl.r                   longword to receive "handle" for
!                                       this request.  This handle value may
!                                       be used to refer back to this
!                                       request in a subsequent call to
!                                       SMG$$REVERT_CHARACTERISTICS.
!       P_OLD_CHARACTERISTICS1.wl.r     gets old characteristics (part 1)
!       P_OLD_CHARACTERISTICS2.wl.r     gets old characteristics (part 2)
!
! IMPLICIT INPUTS:
!
!       Terminal characteristics
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! COMPLETION STATUS:
!
!       SS$_NORMAL      Normal successful completion
!       SMG$_NOT_A_TRM  Success - but device is not a terminal
!       LIB$_xyz        Errors from LIB$GET_VM
!       SS$_xyz         Errors from $DCLEXH or $QIOW
!
```

```
1 ! SIDE EFFECTS:
1 !
1 !--
```

```
2 BEGIN
2
2 BIND
2
2         HANDLE          = .P_HANDLE,
2         OLD_CHAR1       = .P_OLD_CHARACTERISTICS1,
2         OLD_CHAR2       = .P_OLD_CHARACTERISTICS2;
2
2 EXTERNAL LITERAL
2
2         SMG$_NOT_A_TRM;          ! Device is not a terminal (success)
2
2 LOCAL
2
2         PBCB    : REF BLOCK[,BYTE],      ! Pasteboard control block
2         TTIOSB  : VECTOR[4,WORD],        ! I/O status block
2         EXIT_BLOCK                       ! Address of exit block
2                 : REF VECTOR[SMG$S_CHAR_EXIT_BLOCK/4],
2         OLD_CHARBUF     : BLOCK[12,BYTE],        ! Old characteristics buffer
2         CHAR1,                           ! Old characteristics (part 1)
2         CHAR2,                           ! Old characteristics (part 2)
2         STATUS;
2
2 BUILTIN
2
2         NULLPARAMETER;
2
2 EXTERNAL ROUTINE
2
2         LIB$GET_VM;
```

```
!+
! Get the pasteboard control block from the pasteboard id.
!-

$SMG$GET_PBCB(.PBID,PBCB);

!+
! Check that the output device is really a terminal.
! If it isn't then, return qualified success.
!-

IF .PBCB[PBCB_B_CLASS] NEQ DC$_TERM
  THEN  BEGIN
        IF NOT NULLPARAMETER(7)
          THEN  OLD_CHAR1=0;

        IF NOT NULLPARAMETER(8)
          THEN  OLD_CHAR2=0;

        RETURN  SMG$_NOT_A_TRM

        END;

!+
! Save the old characteristics buffer.
!-

CH$MOVE(12,PBCB[PBCB_R_CHARBUF],OLD_CHARBUF);

!+
! Get the old characteristics and return them to the user
! if he asked for them.
!-

CHAR1=.PBCB[PBCB_L_DEVDEPEND];
CHAR2=.PBCB[PBCB_L_DEVDEPEND2];

IF NOT NULLPARAMETER(7)
  THEN  OLD_CHAR1=.CHAR1;

IF NOT NULLPARAMETER(8)
  THEN  OLD_CHAR2=.CHAR2;

!+
! OR in the new characteristic bits requested on.
!-

IF NOT NULLPARAMETER(2)
  THEN  PBCB[PBCB_L_DEVDEPEND]=
                .PBCB[PBCB_L_DEVDEPEND] OR .ON_CHARACTERISTICS1;

IF NOT NULLPARAMETER(3)
  THEN  PBCB[PBCB_L_DEVDEPEND2]=
                .PBCB[PBCB_L_DEVDEPEND2] OR .ON_CHARACTERISTICS2;

!+
! AND out the characteristics bits requested off.
```

```
!-

IF NOT NULLPARAMETER(4)
  THEN  PBCB[PBCB_L_DEVDEPEND]=
                .PBCB[PBCB_L_DEVDEPEND] AND (NOT .OFF_CHARACTERISTICS1);

IF NOT NULLPARAMETER(5)
  THEN  PBCB[PBCB_L_DEVDEPEND2]=
                .PBCB[PBCB_L_DEVDEPEND2] AND (NOT .OFF_CHARACTERISTICS2);

!+
! Set the terminal to have the desired new characteristics.
!-

STATUS=$QIOW(  CHAN    = .PBCB[PBCB_W_CHAN],
               FUNC    = IO$_SETMODE,
               IOSB    = TTIOSB,
               P1      = PBCB[PBCB_R_CHARBUF],
               P2      = 12);
IF NOT .STATUS THEN RETURN .STATUS;
IF NOT .TTIOSB[0] THEN RETURN .TTIOSB[0];

!+
! Create an exit block to be used to set the characteristics back.
!-

STATUS=LIB$GET_VM(%REF(SMG$S_CHAR_EXIT_BLOCK),EXIT_BLOCK);
IF NOT .STATUS THEN RETURN .STATUS;

!+
! Give the user back the address of this exit block as the "handle".
!-

IF NOT NULLPARAMETER(6)
  THEN  HANDLE=.EXIT_BLOCK;

!+
! Set up the exit block.
!-

EXIT_BLOCK[0]=0;                    ! reserved for use by VMS as forward link
EXIT_BLOCK[1]=CHAR_EXIT_HANDLER;    ! specify our exit handler
EXIT_BLOCK[2]=6;                    ! six arguments
EXIT_BLOCK[3]=EXIT_BLOCK[4];        ! Store reason in next entry in exit block
EXIT_BLOCK[4]=0;                    ! Gets reason for exit
EXIT_BLOCK[5]=.PBCB[PBCB_W_CHAN];   ! Terminal channel
!+
! Longwords 6,7,8 form a 12-byte characteristics buffer.
!-
CH$MOVE(12,OLD_CHARBUF,EXIT_BLOCK[6]);

!+
! Declare an exit handler using this exit block.
!-

STATUS=$DCLEXH( DESBLK=.EXIT_BLOCK);
IF NOT .STATUS THEN RETURN .STATUS;
```

```
; 1903      2133  2
; 1904      2134  2 RETURN SS$_NORMAL
; 1905      2135  2
; 1906      2136  1 END;


                                                        .EXTRN  SYS$DCLEXH

                                    01FC 00000          .ENTRY  SMG$$SET_TERM_CHARACTERISTICS, Save R2,R3,-  ; 1930
                                                                R4,R5,R6,R7,R8                               ;
                     5E          1C  C2 00002           SUBL2   #28, SP                                      ;
                     50      04  BC  D0 00005           MOVL    @PBID, R0                                    ; 2023
                                 11  19 00009           BLSS    1$                                           ;
          00000000G  00          50  D1 0000B           CMPL    R0, PBD_L_COUNT                              ;
                                 08  14 00012           BGTR    1$                                           ;
       08 00000000G  00          50  E0 00014           BBS     R0, PBD_V_PB_AVAIL, 2$                       ;
                 50 00000000G    8F  D0 0001C 1$:       MOVL    #SMG$_INVPAS_ID, R0                          ;
                                     04 00023           RET                                                  ;
                 57 00000000G0040    D0 00024 2$:       MOVL    PBD_A_PBCB[R0], PBCB                         ;
                 42  8F      58  A7  91 0002C           CMPB    88(PBCB), #66                                ; 2030
                                 22  13 00031           BEQL    5$                                           ;
                     07          6C  91 00033           CMPB    (AP), #7                                     ; 2032
                                 08  1F 00036           BLSSU   3$                                           ;
                             1C  AC  D5 00038           TSTL    28(AP)                                       ;
                                 03  13 0003B           BEQL    3$                                           ;
                             1C  BC  D4 0003D           CLRL    @P_OLD_CHARACTERISTICS1                      ; 2033
                     08          6C  91 00040 3$:       CMPB    (AP), #8                                     ; 2035
                                 08  1F 00043           BLSSU   4$                                           ;
                             20  AC  D5 00045           TSTL    32(AP)                                       ;
                                 03  13 00048           BEQL    4$                                           ;
                             20  BC  D4 0004A           CLRL    @P_OLD_CHARACTERISTICS2                      ; 2036
                 50 00000000G    8F  D0 0004D 4$:       MOVL    #SMG$_NOT_A_TRM, R0                          ; 2038
                                     04 00054           RET                                                  ;
        08  AE       58  A7          0C  28 00055 5$:   MOVC3   #12, 88(PBCB), OLD_CHARBUF                   ; 2046
                     50      5C  A7  7D 0005B           MOVQ    92(PBCB), CHAR1                              ; 2053
                     07          6C  91 0005F           CMPB    (AP), #7                                     ; 2056
                                 09  1F 00062           BLSSU   6$                                           ;
                             1C  AC  D5 00064           TSTL    28(AP)                                       ;
                                 04  13 00067           BEQL    6$                                           ;
                 1C  BC          50  D0 00069           MOVL    CHAR1, @P_OLD_CHARACTERISTICS1               ; 2057
                     08          6C  91 0006D 6$:       CMPB    (AP), #8                                     ; 2059
                                 09  1F 00070           BLSSU   7$                                           ;
                             20  AC  D5 00072           TSTL    32(AP)                                       ;
                                 04  13 00075           BEQL    7$                                           ;
                 20  BC          51  D0 00077           MOVL    CHAR2, @P_OLD_CHARACTERISTICS2               ; 2060
                     02          6C  91 0007B 7$:       CMPB    (AP), #2                                     ; 2066
                                 0A  1F 0007E           BLSSU   8$                                           ;
                             08  AC  D5 00080           TSTL    8(AP)                                        ;
                                 05  13 00083           BEQL    8$                                           ;
                 5C  A7      08  AC  C8 00085           BISL2   ON_CHARACTERISTICS1, 92(PBCB)                ; 2068
                     03          6C  91 0008A 8$:       CMPB    (AP), #3                                     ; 2070
                                 0A  1F 0008D           BLSSU   9$                                           ;
                             0C  AC  D5 0008F           TSTL    12(AP)                                       ;
                                 05  13 00092           BEQL    9$                                           ;
                 60  A7      0C  AC  C8 00094           BISL2   ON_CHARACTERISTICS2, 96(PBCB)                ; 2072
                     04          6C  91 00099 9$:       CMPB    (AP), #4                                     ; 2078
```

```
                      0A  1F 0009C       BLSSU   10$                                    ;
              10      AC  D5 0009E       TSTL    16(AP)                                 ;
                      05  13 000A1       BEQL    10$                                    ;
      5C  A7      10  AC  CA 000A3       BICL2   OFF_CHARACTERISTICS1, 92(PBCB)         ; 2080
          05          6C  91 000A8 10$:  CMPB    (AP), #5                               ; 2082
                      0A  1F 000AB       BLSSU   11$                                    ;
              14      AC  D5 000AD       TSTL    20(AP)                                 ;
                      05  13 000B0       BEQL    11$                                    ;
      60  A7      14  AC  CA 000B2       BICL2   OFF_CHARACTERISTICS2, 96(PBCB)         ; 2084
                      7E  7C 000B7 11$:  CLRQ    -(SP)                                  ; 2094
                      7E  7C 000B9       CLRQ    -(SP)                                  ;
                      0C  DD 000BB       PUSHL   #12                                    ;
              58      A7  9F 000BD       PUSHAB  88(PBCB)                               ;
                      7E  7C 000C0       CLRQ    -(SP)                                  ;
              34      AE  9F 000C2       PUSHAB  TTIOSB                                 ;
                      23  DD 000C5       PUSHL   #35                                    ;
          7E      64  A7  3C 000C7       MOVZWL  100(PBCB), -(SP)                       ;
                      7E  D4 000CB       CLRL    -(SP)                                  ;
  00000000G  00           0C  FB 000CD   CALLS   #12, SYS$QIOW                          ;
          58          50  D0 000D4       MOVL    R0, STATUS                             ;
          61          58  E9 000D7       BLBC    STATUS, 14$                            ; 2095
          05      14  AE  E8 000DA       BLBS    TTIOSB, 12$                            ; 2096
          50      14  AE  3C 000DE       MOVZWL  TTIOSB, R0                             ;
                      04 000E2           RET                                            ;
              04      AE  9F 000E3 12$:  PUSHAB  EXIT_BLOCK                             ; 2102
      04  AE          24  D0 000E6       MOVL    #36, 4(SP)                             ;
              04      AE  9F 000EA       PUSHAB  4(SP)                                  ;
  00000000G  00           02  FB 000ED   CALLS   #2, LIB$GET_VM                         ;
          58          50  D0 000F4       MOVL    R0, STATUS                             ;
          41          58  E9 000F7       BLBC    STATUS, 14$                            ; 2103
          06          6C  91 000FA       CMPB    (AP), #6                               ; 2109
                      0A  1F 000FD       BLSSU   13$                                    ;
              18      AC  D5 000FF       TSTL    24(AP)                                 ;
                      05  13 00102       BEQL    13$                                    ;
      18  BC      04  AE  D0 00104       MOVL    EXIT_BLOCK, @P_HANDLE                  ; 2110
          56      04  AE  D0 00109 13$:  MOVL    EXIT_BLOCK, R6                         ; 2116
                      66  D4 0010D       CLRL    (R6)                                   ;
      04  A6   0000V  CF  9E 0010F       MOVAB   CHAR_EXIT_HANDLER, 4(R6)               ; 2117
      08  A6          06  D0 00115       MOVL    #6, 8(R6)                              ; 2118
      0C  A6      10  A6  9E 00119       MOVAB   16(R6), 12(R6)                         ; 2119
              10      A6  D4 0011E       CLRL    16(R6)                                 ; 2120
      14  A6      64  A7  3C 00121       MOVZWL  100(PBCB), 20(R6)                      ; 2121
  18  A6  08  AE      0C  28 00126       MOVC3   #12, OLD_CHARBUF, 24(R6)               ; 2125
                      56  DD 0012C       PUSHL   R6                                     ; 2131
  00000000G  00           01  FB 0012E   CALLS   #1, SYS$DCLEXH                         ;
          58          50  D0 00135       MOVL    R0, STATUS                             ;
          04          58  E8 00138       BLBS    STATUS, 15$                            ; 2132
          50          58  D0 0013B 14$:  MOVL    STATUS, R0                             ;
                      04 0013E           RET                                            ;
          50          01  D0 0013F 15$:  MOVL    #1, R0                                 ; 2134
                      04 00142           RET                                            ; 2136

; Routine Size:  323 bytes,    Routine Base:  _SMG$CODE + 06BF
```

```
%SBTTL 'CHAR_EXIT_HANDLER - exit handler to reset terminal characteristics'
ROUTINE CHAR_EXIT_HANDLER(RSN_ADR,RSN,CHAN,CHARBUF : VECTOR[3,LONG]) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine is established as an exit handler.
!       It restores the terminal to it's previous characteristics.
!
! CALLING SEQUENCE:
!
!       RET_STATUS.wlc.v = CHAR_EXIT_HANDLER( P_EXIT_BLOCK.rab.r)
!
! FORMAL PARAMETERS:
!
!       P_EXIT_BLOCK.rab.r      Exit block as set up by
!                               SMG$$SET_TERMINAL_CHARACTERISTICS
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! COMPLETION STATUS:
!
!       SS$_NORMAL      Normal successful completion
!
!               Errors are ignored.
!
! SIDE EFFECTS:
!
!       Previous terminal characteristics are restored
!
!--
```

```
; 1946          2174  2 BEGIN
; 1947          2175  2
; 1948          2176  2 BIND
; 1949          2177  2
; 1950          2178  2         CLASS           = CHARBUF       : BYTE;
; 1951          2179  2
; 1952          2180  2 LOCAL
; 1953          2181  2
; 1954          2182  2         STATUS;
; 1955          2183  2
; 1956          2184  2 !+
; 1957          2185  2 ! Verify that the device class to be restored is DC$_TERM.
; 1958          2186  2 ! If not, just go away.
; 1959          2187  2 ! Give no error, since this is an exit handler.
; 1960          2188  2 ! If it is a terminal, then restore the old characteristics.
; 1961          2189  2 !-
; 1962          2190  2
; 1963          2191  2 IF .CLASS EQL DC$_TERM
; 1964        P 2192  2   THEN  $QIOW(  CHAN      = .CHAN,
; 1965        P 2193  2                 FUNC      = IO$_SETMODE,
; 1966        P 2194  2                 P1        = CHARBUF,
; 1967          2195  2                 P2        = 12);
; 1968          2196  2
; 1969          2197  2 RETURN  SS$_NORMAL
; 1970          2198  2
; 1971          2199  1 END;



                                        0000 00000 CHAR_EXIT_HANDLER:
                                                          .WORD   Save nothing                  ; 2138
            42   8F        10    AC  91 00002             CMPB    CLASS, #66                    ; 2191
                                 1A  12 00007             BNEQ    1$
                                 7E  7C 00009             CLRQ    -(SP)                         ; 2195
                                 7E  7C 0000B             CLRQ    -(SP)                         ;
                                 0C  DD 0000D             PUSHL   #12                           ;
                           10    AC  9F 0000F             PUSHAB  CHARBUF                       ;
                                 7E  7C 00012             CLRQ    -(SP)                         ;
                 7E              23  7D 00014             MOVQ    #35, -(SP)                    ;
                           0C    AC  DD 00017             PUSHL   CHAN                          ;
                                 7E  D4 0001A             CLRL    -(SP)                         ;
       00000000G  00             0C  FB 0001C             CALLS   #12, SYS$QIOW                 ;
                  50             01  D0 00023 1$:         MOVL    #1, R0                        ; 2197
                                     04 00026             RET                                   ; 2199

; Routine Size:  39 bytes,    Routine Base:  _SMG$CODE + 0802
```

```
; 1973      2200  1 END
; 1974      2201  0 ELUDOM


                                                        .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                     Bytes                    Attributes
;
;   _SMG$CODE                      2089  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;   _SMG$DATA                         4  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)


;                          Library Statistics
;
;                                            -------- Symbols --------        Pages       Processing
;       File                                 Total    Loaded   Percent       Mapped      Time
;
;   _$255$DUA28:[SYSLIB]STARLET.L32;1         9776        44         0          581        00:00.9
;   _$255$DUA28:[SMGRTL.OBJ]RTLLIB.L32;1        36         0         0            8        00:00.1
;   _$255$DUA28:[SMGRTL.OBJ]SMGLIB.L32;1       469        48        10           38        00:00.3


;                               COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:SMGMISC/OBJ=OBJ$:SMGMISC MSRC$:SMGMISC/UPDATE=(ENH$:SMGMISC)

; Size:          2089 code + 4 data bytes
; Run Time:         00:43.3
; Elapsed Time:     02:02.0
; Lines/CPU Min:     3051
; Lexemes/CPU-Min: 21217
; Memory Used:  182 pages
; Compilation Complete
```

0360 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SMGNUMTAB
LIS

SMGMSGPTR
LIS

SMGSCROLL
LIS

SMGMISC
LIS

SMGMSGTXT
LIS

SMGPUTENC
LIS

SMGPUTTEX
LIS

SMGSIMTRM
LIS

SMGNUMPAR
LIS

SMGPRVINP
LIS