C 12

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

FILEID**SMGINPUT

F 1

```
  SSSSSSSS  MM      MM    GGGGGGGG    IIIIII    NN      NN  PPPPPPPP    UU      UU  TTTTTTTTTT
  SSSSSSSS  MM      MM    GGGGGGGG    IIIIII    NN      NN  PPPPPPPP    UU      UU  TTTTTTTTTT
SS          MMMM  MMMM  GG              II      NN      NN  PP      PP  UU      UU      TT
SS          MMMM  MMMM  GG              II      NN      NN  PP      PP  UU      UU      TT
SS          MM  MM  MM  GG              II      NNNN    NN  PP      PP  UU      UU      TT
SS          MM  MM  MM  GG              II      NNNN    NN  PP      PP  UU      UU      TT
  SSSSSS    MM      MM  GG              II      NN  NN  NN  PPPPPPPP    UU      UU      TT
  SSSSSS    MM      MM  GG              II      NN  NN  NN  PPPPPPPP    UU      UU      TT
        SS  MM      MM  GG  GGGGGG      II      NN    NNNN  PP          UU      UU      TT
        SS  MM      MM  GG  GGGGGG      II      NN    NNNN  PP          UU      UU      TT
        SS  MM      MM  GG      GG      II      NN      NN  PP          UU      UU      TT          ....
        SS  MM      MM  GG      GG      II      NN      NN  PP          UU      UU      TT          ....
SSSSSSSS    MM      MM    GGGGGG      IIIIII    NN      NN  PP          UUUUUUUUUU      TT          ....
SSSSSSSS    MM      MM    GGGGGG      IIIIII    NN      NN  PP          UUUUUUUUUU      TT          ....

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

```
0 MODULE SMG$INPUT ( %TITLE 'Screen Management Facility Input Procedures'
0                    IDENT = '2-017'          ! File: SMGINPUT.B32 Edit: STAN2017
0                    ) =
1 BEGIN
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! FACILITY:     Screen Management Facility
1 !
1 ! ABSTRACT:
1 !
1 !       This module contains the portions of the VAX-11 Run-Time Library
1 !       Screen Management Facility which deal with input.
1 !
1 ! ENVIRONMENT:  User mode - AST reentrant
1 !
1 ! AUTHOR: Steven B. Lionel, CREATION DATE: 10-Feb-1983
1 !
1 ! MODIFIED BY:
1 !
1 ! 1-001 - Original.  SBL 10-Feb-1983
1 ! 1-002 - Add variant code to allow READ_STRING to work on a V3.n system.
1 !         Fix READ_COMPOSED_LINE so that the equivalence string of the final
1 !         key is remembered by the terminal-driver for line-recall.  Use
1 !         a <CR> as a prompt if none is specified.
1 !         SBL 18-May-1983
1 ! 1-003 - Remember /LOCK'ed states across multiple reads.  Do "right" thing
1 !         with ^Z, undefined keys in the middle of lines.  SBL 23-Jun-1983
1 ! 1-004 - Prevent last echo in READ_COMPOSED_LINE unless there is
1 !         something to echo.  SBL 28-Jul-1983
1 ! p-003 - Modify SMG$CREATE_VIRTUAL_KEYBOARD and SMG$READ_STRING to
1 !         interface with screen management output.  PLL 16-Jun-1983
1 ! p-004 - More modifications for integration.  PLL 21-Jun-1983
1 ! p-005 - SMG$$SET_PHYSICAL_CURSOR now takes a longword buffer length,
```

```
1 !             instead of word.  PLL 28-Jun-1983
1 ! p-006 - Fix a couple bugs in SMG$READ_STRING.  PLL 29-Jun-1983
1 !               1.  Use default rendition for virtual display for prompt/input.
1 !               2.  If at the boundary of the virtual display, then there is no
1 !                   room for input - issue an error.
1 ! 2-001 - Combined Steves edits 1-003 and 1-004 with Pam's edits
1 !         1-003 through 1-006 and added some stuff of my own.
1 !         STAN 2-Aug-1983.
1 ! 2-002 - FINISHED INPUT/OUTPUT INTEGRATION. STAN 3-Aug-1983.
1 ! 2-003 - Bug fixes to SMG$READ_STRING for terminators and modifier
1 !         masks.  PLL 9-Aug-1983
1 ! 2-004 - Another attempt to get 3a modifiers recognized.  Also, pass the
1 !         terminator to the smg$ output routines - the terminator may affect
1 !         the cursor position.  PLL 16-Aug-1983
1 ! 2-005 - In SMG$READ_STRING, don't report the change if NOECHO was requested.
1 !         PLL 18-Aug-1983
1 ! 2-006 - Fix last edit.  PLL 22-Aug-1983
1 ! 2-007 - In SMG$READ_STRING, don't report the terminator if TRMNOECHO was
1 !         requested.  PLL 9-Sep-1983
1 ! 2-008 - Another fix to SMG$READ_STRING:  if a partial escape sequence
1 !         terminated the read, do single character QIOs till we find the end
1 !         of the terminator.  (This prevents keypad keys from echoing on the
1 !         screen.)  PLL 13-Sep-1983
1 ! 2-009 - Cleanup SMG$READ_COMPOSED_LINE.  PLL 5-Oct-1983
1 ! 2-010 - GIVE ERROR if try to input to a batched display. STAN 14-Mar-1984.
1 !         Fix length of buffers for renditions.
1 !         Make renditions work.
1 !         Change echo of CTRL/Z to track terminal driver change.
1 ! 2-011 - Return terminator code of SMG$K_TRM_TIMEOUT on timeout. STAN
1 ! 2-012 - Turn off renditions only if DCB was passed. STAN 28-MAR-1984.
1 ! 2-013 - Add the codes for F1-F20 function keys, and editing keys to the code
1 !         that checks for partial escapes.  PLL 2-Apr-1984
1 ! 2-014 - Validate number of arguments. STAN 3-Jun-1984.
1 !         Add F1-F20 code for READ COMPOSE LINE.
1 ! 2-015 - Take CTRL/Z echo from system logical TTY$EXIT_STRING. STAN 7-Jul-1984.
1 ! 2-016 - Fix to partial escape handling - after we find the end of the
1 !         terminator, fix the return status to be success.  PLL 9-Jul-1984
1 ! 2-017 - Remove call to TRNLNM on V3 systems. STAN 24-Jul-1984.
1 !         Remove informational errors.
1 !--
```

```
%SBTTL 'Declarations'
!
! PROLOGUE FILE:
!

REQUIRE 'RTLIN:SMGPROLOG';                      ! Screen management definitions

!
! TABLE OF CONTENTS:
!

FORWARD ROUTINE
    SMG$CREATE_VIRTUAL_KEYBOARD,                ! Create a virtual keyboard
    SMG$DELETE_VIRTUAL_KEYBOARD,                ! Delete a virtual keyboard
    SMG$READ_STRING,                            ! Read a string
    SMG$READ_COMPOSED_LINE,                     ! Read a composed line
    SMG$SET_KEYPAD_MODE,                        ! Set mode of numeric keypad
    SMG$CANCEL_INPUT,                           ! Cancel outstanding input
    SMG$$SET_PROMPT_STRING,                     ! Create a prompt string
    SMG$$DELETE_PROMPT_STRING: NOVALUE,         ! Delete a prompt string
    SMG$$INIT_QUEUES,                           ! Initialize queues
    SMG$$CLEANUP: SMG$$CLEANUP$LNK,             ! Cleanup from error or close
    SMG$$INPUT_EXIT_HANDLER: NOVALUE;           ! Exit handler for input

!
! MACROS:
!
!       NONE
!
! EQUATED SYMBOLS:
!

LITERAL
    K_QPS_SIZE = 90;        ! Size of prompt strings on QPS_QUEUE.
                            ! 80 character bytes + possibly a 10 byte
                            ! set cursor sequence.

!
! FIELDS:
!
!       NONE
!
! OWN STORAGE:
!

!+
! Queues of control blocks and prompt strings.
!-

OWN
    KQB_QUEUE: VOLATILE VECTOR [2, LONG],       ! Keyboard Queue Block
    QPS_QUEUE: VOLATILE VECTOR [2, LONG],       ! Queued Prompt Strings
    V_QUEUE_INIT: VOLATILE INITIAL (0); ! 1 if queues have been initialized

!+
! Declare strings which set the keypad into and out of applications mode.
!-
```

```
1
1 OWN
1     T_KPDAPP_DECCRT:     ! Sets DECCRT (and VT5x) into applications mode
1         VECTOR [2, BYTE] PSECT (_SMG$CODE) INITIAL (BYTE (27,'=')),
1     T_KPDNUM_DECCRT:     ! Sets DECCRT and (VT5x) into numeric mode
1         VECTOR [2, BYTE] PSECT (_SMG$CODE) INITIAL (BYTE (27,'>'));
1
1 !
1 !
1 ! EXTERNAL DECLARATIONS:
1 !
1
1 EXTERNAL ROUTINE
1     LIB$GET_EF,                     ! Allocate local event flag number
1     LIB$FREE_EF,                    ! Free local event flag number
1     LIB$GET_VM,                     ! Allocate virtual memory
1     LIB$FREE_VM,                    ! Deallocate virtual memory
1     LIB$ANALYZE_SDESC_R2: LIB$ANALYZE_SDESC_R2$LINKAGE, ! Analyze string desc
1     LIB$SCOPY_R_DX,                 ! Copy string
1     LIB$SCOPY_R_DX6: LIB$SCOPY_R_DX6$LINKAGE,   ! Copy string
1     SMG$$TERM_TO_KEYCODE,           ! Translate terminator to keycode
1     SMG$$LOOKUP_KEY: SMG$$LOOKUP_KEY$LNK,       ! Lookup item in tree
1     SMG$$GET_PASTEBOARD_ID,         ! SMG$ I/O interface routine
1     SMG$$SET_PHYSICAL_CURSOR,       !  "
1     SMG$$REPORT_CHANGE_REPLACE,     !  "
1     SMG$$OUTPUT,                    ! Output directly to terminal
1     SMG$$SET_ATTRIBUTES_ON,         ! Set attributes on
1     SMG$$SET_ATTRIBUTES_OFF,        ! Set attributes off
1     SYS$QIOW;                       ! $QIOW system service
1
1 EXTERNAL LITERAL
1     SMG$_STRTERESC,         ! String terminated by escape
1     SMG$_EOF,               ! End-of-file
1     SMG$_FILTOOLON,         ! File-specification is too long (over 255 characters)
1     SMG$_ILLBATFNC,         ! Operation not permitted while display is batched
1     SMG$_INVMAXLEN,         ! Invalid maximum length (greater than 512 characters)
1     SMG$_INVSTANAM,         ! Invalid state name
1     SMG$_PROTOOLON,         ! Prompt string is too long
1     SMG$_INVCOL,            ! Invalid column
1     SMG$_INVDIS_ID;         ! Invalid display id
1
1 EXTERNAL
1     LIB$AB_UPCASE;          ! Upcase translate table
```

```
1 %SBTTL 'SMG$CREATE_VIRTUAL_KEYBOARD'
1 GLOBAL ROUTINE SMG$CREATE_VIRTUAL_KEYBOARD (
1       KEYBOARD_ID: REF VECTOR [, LONG],
1       FILESPEC: REF BLOCK [, BYTE],
1       DEFAULT_FILESPEC: REF BLOCK [, BYTE],
1       RESULTANT_FILESPEC: REF BLOCK [, BYTE]
1       ) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This procedure creates the association between a filespec (terminal
1 !       name or RMS file) and a virtual-keyboard-identifier.  The keyboard-id
1 !       is passed to other screen management procedures to uniquely identify
1 !       the input stream being acted upon.
1 !
1 !       If "filespec" does not refer to a terminal, the file is opened using
1 !       RMS and all further access to that file is performed through RMS.  If
1 !       "filespec" is a terminal, a channel is assigned to the terminal, its
1 !       keypad is set to "application mode" (if supported), and line editing
1 !       is enabled.  These attributes are restored to their previous values
1 !       when the virtual-keyboard is deleted, either by a call to
1 !       SMG$DELETE_VIRTUAL_KEYBOARD or automatically when the image exits.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       RET_STATUS.wlc.v = SMG$CREATE_VIRTUAL_KEYBOARD (
1 !                               KEYBOARD-ID.wl.r
1 !                               [, [FILESPEC.rt.dx]
1 !                               [, [DEFAULT_FILESPEC.rt.dx]
1 !                               [, [RESULTANT_FILESPEC.wt.dx] ]]])
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       KEYBOARD-ID     A longword into which will be written the
1 !                       virtual-keyboard-id of the created virtual-keyboard.
1 !
1 !       FILESPEC        A string containing the file specification of the
1 !                       file or terminal to be used for this virtual-keyboard.
1 !                       If omitted, 'SYS$INPUT:' is used.
1 !
1 !       DEFAULT_FILESPEC A string containing the default file specification.
1 !                       If omitted, the null string is used.
1 !
1 !       RESULTANT_FILESPEC A string into which is written the expanded or
1 !                       resultant file specification of the file used.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! COMPLETION STATUS:
1 !
```

```
  1 !          SS$_NORMAL        Normal successful completion
  1 !          SMG$_FILTOOLON    File-specification is too long (over 255 characters)
  1 !          SMG$_WRONUMARG    Wrong number of arguments
  1 !          LIB$_INSEF        Insufficient event flags
  1 !          LIB$_INSVIRMEM    Insufficient virtual memory
  1 !          LIB$_INVSTRDES    Invalid string descriptor
  1 !          RMS$_xyz          Any error possible from $OPEN or $CONNECT
  1 !          SS$_xyz           Any error possible from $GETDVIW, $ASSIGN or $DCLEXH
  1 !
  1 ! SIDE EFFECTS:
  1 !
  1 !          Opens file using RMS or assigns a channel to it.
  1 !          Declares an exit handler
  1 !          Sets terminal to keypad application mode.
  1 !
  1 !--
  1
  2      BEGIN
  2
  2      LOCAL
  2          KCB: REF KCB_R_KCB_STRUCT,                   ! Keyboard Control Block
  2          FAB: $FAB_DECL,                              ! RMS File Attributes Block
  2          NAM: $NAM_DECL,                              ! RMS NAM block
  2          XAB_FHC: $XABFHC_DECL,                       ! File Header Char. XAB
  2          RESULTANT_STRING: VECTOR [NAM$C_MAXRSS, BYTE]; ! Resultant name string
  2
  2      BUILTIN
  2          CALLG,
  2          NULLPARAMETER;
  2
  2      !+
  2      ! Literals for argument positions.
  2      !-
  2
  2      LITERAL
  2          K_KEYBOARD_ID = 1,
  2          K_FILESPEC = 2,
  2          K_DEFAULT_FILESPEC = 3,
  2          K_RESULTANT_FILESPEC = 4;
  2
  2      $SMG$VALIDATE_ARGCOUNT(1,4);
  2
  2      !+
  2      ! Verify assumption that a RAB is not longer than the positive-offset
  2      ! portion of a Keyboard Control Block.
  2      !-
  2
  2      $ASSUME (RAB$C_BLN, LEQU, KCB_S_KCB_STRUCT - KCB_K_ORIGIN_OFFSET);
  2
  2      !+
  2      ! Initialize FAB, NAM  and XAB.
  2      !-
  2
  2      $FAB_INIT (FAB=FAB, NAM=NAM, XAB=XAB_FHC);
P 2      $NAM_INIT (NAM=NAM, ESA=RESULTANT_STRING, ESS=NAM$C_MAXRSS,
  2          RSA=RESULTANT_STRING, RSS=NAM$C_MAXRSS);
  2      $XABFHC_INIT (XAB=XAB_FHC);
```

```
    !+
    ! Get the file specification.
    !-

    IF NOT NULLPARAMETER (K_FILESPEC)   ! FILESPEC present?
    THEN
        BEGIN
        LOCAL
            STRING_PTR,             ! Pointer to string
            STRING_LEN: WORD,       ! Length of string
            ANL_STATUS;             ! Status from LIB$ANALYZE_SDESC_R2

        ANL_STATUS = LIB$ANALYZE_SDESC_R2 (FILESPEC [0,0,0,0];
            STRING_LEN, STRING_PTR);
        IF NOT .ANL_STATUS
        THEN
            RETURN .ANL_STATUS;

        IF .STRING_LEN GTRU 255
        THEN
            RETURN SMG$_FILTOOLON;
        FAB [FAB$L_FNA] = .STRING_PTR;
        FAB [FAB$B_FNS] = .STRING_LEN;
        END
    ELSE
        BEGIN
        !+
        ! Use 'SYS$INPUT:'
        !-
        FAB [FAB$B_FNS] = %CHARCOUNT ('SYS$INPUT:');
        FAB [FAB$L_FNA] = UPLIT BYTE ('SYS$INPUT:');
        END;

    !+
    ! Get default filespec, if any.
    !-

    IF NOT NULLPARAMETER (K_DEFAULT_FILESPEC)
    THEN
        BEGIN
        LOCAL
            STRING_PTR,             ! Pointer to string
            STRING_LEN: WORD,       ! Length of string
            ANL_STATUS;             ! Status from LIB$ANALYZE_SDESC_R2

        ANL_STATUS = LIB$ANALYZE_SDESC_R2 (DEFAULT_FILESPEC [0,0,0,0];
            STRING_LEN, STRING_PTR);
        IF NOT .ANL_STATUS
        THEN
            RETURN .ANL_STATUS;

        IF .STRING_LEN GTRU 255
        THEN
            RETURN SMG$_FILTOOLON;
        FAB [FAB$L_DNA] = .STRING_PTR;
        FAB [FAB$B_DNS] = .STRING_LEN;
```

```
	END;

    !+
    ! Parse the file specification to see whether or not it is
    ! a terminal.
    !-

	BEGIN
	LOCAL
	    PARSE_STATUS;
	PARSE_STATUS = $PARSE (FAB = FAB);
	IF NOT .PARSE_STATUS
	THEN
	    RETURN .PARSE_STATUS;
	END;

    !+
    ! If KQB_QUEUE and QPS_QUEUE have not yet been initialized, initialize them.
    ! SMG$$INIT_QUEUES also declares an exit handler.
    !-

    IF NOT .V_QUEUE_INIT
    THEN
	BEGIN
	LOCAL
	    INIT_STATUS;
	INIT_STATUS = SMG$$INIT_QUEUES ();
	IF NOT .INIT_STATUS
	THEN
	    RETURN .INIT_STATUS;
	END;

    !+
    ! Allocate a Keyboard Control Block
    !-

	BEGIN
	LOCAL
	    VM_POINTER,		! Pointer to allocated memory
	    VM_STATUS;		! Status from LIB$GET_VM
	VM_STATUS = LIB$GET_VM (%REF(KCB_S_KCB_STRUCT), VM_POINTER);
	IF NOT .VM_STATUS
	THEN
	    RETURN .VM_STATUS;
	CH$FILL (0, KCB_S_KCB_STRUCT, .VM_POINTER);
	KCB = .VM_POINTER + KCB_K_ORIGIN_OFFSET;
	END;

    !+
    ! Allocate a Keyboard Queue Block and put this KCB on the queue.
    ! Note that KQBs are never removed from the queue; they are inactivated
    ! by having the KQB_A_KCB field set to zero.
    !-

	BEGIN
	LOCAL
	    KQB: REF KQB_R_KQB_STRUCT,	! Keyboard Queue Block
```

```
              KQB_PTR,      ! Pointer to allocated block
              VM_STATUS;
          BUILTIN
              INSQUE;

          VM_STATUS = LIB$GET_VM (%REF(KQB_S_KQB_STRUCT), KQB_PTR);
          IF NOT .VM_STATUS
          THEN
              RETURN SMG$$CLEANUP (KCB [KCB_R_KCB], .VM_STATUS);
          KQB = .KQB_PTR;
          KQB [KQB_A_KCB] = .KCB;
          KCB [KCB_A_KQB] = .KQB;
          INSQUE (KQB [KQB_Q_QUEUE_LINK], KQB_QUEUE);
          END;

      !+
      ! See if this file is a terminal.
      !-

          BEGIN
          BIND
              FAB_DEV = FAB [FAB$L_DEV]: BLOCK [4, BYTE];
          IF NOT .FAB_DEV [DEV$V_TRM]
          THEN
              KCB [KCB_V_RMS] = 1;          ! Not a terminal
          END;

      !+
      ! Now split dpending on whether the file is a terminal or not.
      !-

      IF NOT .KCB [KCB_V_RMS]
      THEN
          BEGIN
          !+
          ! The file is a terminal.  Assign a channel and use $QIOs.
          !-

          LOCAL
              DEVNAME_DESCR: BLOCK [8, BYTE], ! Descriptor for device name
              ASSIGN_STATUS, QIO_STATUS1, QIO_STATUS2, QIO_STATUS3;

          BIND
              KCB_DEVDEPEND1 = KCB [KCB_L_DEVDEPEND1]: BLOCK [4, BYTE],
              KCB_DEVDEPEND2 = KCB [KCB_L_DEVDEPEND2]: BLOCK [4, BYTE];

          !+
          ! Fill in device name descriptor and assign a channel.
          !-

          DEVNAME_DESCR [DSC$W_LENGTH] = .NAM [NAM$B_ESL];
          DEVNAME_DESCR [DSC$B_CLASS] = DSC$K_CLASS_S;
          DEVNAME_DESCR [DSC$B_DTYPE] = DSC$K_DTYPE_T;
          DEVNAME_DESCR [DSC$A_POINTER] = RESULTANT_STRING;
          ASSIGN_STATUS = $ASSIGN (
              DEVNAM = DEVNAME_DESCR,       ! Device name descriptor
              CHAN = KCB [KCB_W_CHANNEL]);
```

```
	IF NOT .ASSIGN_STATUS
	THEN
	    RETURN SMG$$CLEANUP (KCB [KCB_R_KCB], .ASSIGN_STATUS);

	!+
	! Allocate an event flag number.
	!-

	    BEGIN
	    LOCAL
		EF_STATUS;
	    EF_STATUS = LIB$GET_EF (KCB [KCB_L_EFN]);
	    IF NOT .EF_STATUS
	    THEN
		RETURN SMG$$CLEANUP (KCB [KCB_R_KCB], .EF_STATUS);
	    END;

	!+
	! Fill in the QIO argument lists.
	!-

	KCB [KCB_L_QIO1_ARGCNT] = KCB_K_QIO1_ARGCNT;
	KCB [KCB_L_QIO1_EFN] = .KCB [KCB_L_EFN];
	KCB [KCB_L_QIO1_CHAN] = .KCB [KCB_W_CHANNEL];
	KCB [KCB_L_QIO1_FUNC] = IO$_READVBLK + IO$M_ESCAPE;
	KCB [KCB_A_QIO1_IOSB] = KCB [KCB_R_IOSB];
	KCB [KCB_L_QIO2_ARGCNT] = KCB_K_QIO2_ARGCNT;
	KCB [KCB_L_QIO2_EFN] = .KCB [KCB_L_EFN];
	KCB [KCB_L_QIO2_CHAN] = .KCB [KCB_W_CHANNEL];
	KCB [KCB_A_QIO2_IOSB] = KCB [KCB_R_IOSB];


	!+
	! Get the terminal's current characteristics.
	!-

	KCB [KCB_L_QIO2_FUNC] = IO$_SENSEMODE;
	KCB [KCB_L_QIO2_P1] = KCB [KCB_R_TERMCHAR];
	KCB [KCB_L_QIO2_P2] = KCB_S_TERMCHAR;

	QIO_STATUS1 = CALLG (KCB [KCB_R_QIO2], SYS$QIOW);
	IF .QIO_STATUS1
	THEN
	    QIO_STATUS1 = .KCB [KCB_W_IOSB_STATUS];
	IF NOT .QIO_STATUS1
	THEN
	    RETURN SMG$$CLEANUP (KCB [KCB_R_KCB], .QIO_STATUS1);

	!+
	! Save the device-dependent characteristics.
	!-

	KCB [KCB_L_OLD_DEVDEPEND1] = .KCB [KCB_L_DEVDEPEND1];
	KCB [KCB_L_OLD_DEVDEPEND2] = .KCB [KCB_L_DEVDEPEND2];

	!+
	! Indicate if the enter/exit keypad mode sequence is compatible
```

```
3            ! with that for DECCRTs.
3            !-
3
3            IF .KCB_DEVDEPEND2 [TT2$V_DECCRT] OR
3                (.KCB [KCB_B_DEVTYPE] EQL DT$_VT5X) OR
3                (.KCB [KCB_B_DEVTYPE] EQL DT$_VT55) OR
4                ((.KCB [KCB_B_DEVTYPE] GEQU DT$_LA36) AND    ! Covers LA36,120,34,38,
3                 (.KCB [KCB_B_DEVTYPE] LEQU DT$_LA100)) OR   ! 12,24,100
3                (.KCB [KCB_B_DEVTYPE] EQL DT$_VK100) OR
4                (.KCB [KCB_B_DEVTYPE] EQL DT$_VT173)
3            THEN
3                KCB [KCB_V_KPDSEQ_DECCRT] = 1;
3
3            END
2        ELSE
3            BEGIN
3            !+
3            ! Device is not a terminal.  Use RMS.
3            !-
3
3            LOCAL
3                OPEN_STATUS, CONNECT_STATUS, VM_STATUS;
3
3            !+
3            ! Open file.
3            !-
3
3                OPEN_STATUS;
3            OPEN_STATUS = $OPEN (FAB=FAB);
3            IF NOT .OPEN_STATUS
3            THEN
3                RETURN SMG$$CLEANUP (KCB [KCB_R_KCB], .OPEN_STATUS);
3
3            KCB [RAB$B_BID] = RAB$C_BID;
3            KCB [RAB$B_BLN] = RAB$C_BLN;
3            KCB [RAB$L_FAB] = FAB;
3
3            !+
3            ! Connect to the file.
3            !-
3
3            CONNECT_STATUS = $CONNECT (RAB=KCB [KCB_R_RAB]);
3            IF NOT .CONNECT_STATUS
3            THEN
3                RETURN SMG$$CLEANUP (KCB [KCB_R_KCB], .CONNECT_STATUS);
3
3            !+
3            ! Store the IFI for a future close.
3            !-
3
3            KCB [KCB_W_IFI] = .FAB [FAB$W_IFI];
3
3            !+
3            ! Allocate a buffer that is large enough for the file.
3            !-
3
3            KCB [RAB$W_USZ] = .FAB [FAB$W_MRS];
```

```
3              IF .KCB [RAB$W_USZ] EQL 0
3              THEN
4                  BEGIN
4                  KCB [RAB$W_USZ] = .FAB [FAB$W_BLS];
4                  IF .XAB_FHC [XAB$W_LRL] GTRU .KCB [RAB$W_USZ]
4                  THEN
4                      KCB [RAB$W_USZ] = .XAB_FHC [XAB$W_LRL];
3                  END;
3              VM_STATUS = LIB$GET_VM (%REF(.KCB [RAB$W_USZ]), KCB [RAB$L_UBF]);
3              IF NOT .VM_STATUS
3              THEN
3                  RETURN SMG$$CLEANUP (KCB [KCB_R_KCB], .VM_STATUS);
3
3              !+
3              ! Set additional values needed in the RAB.
3              !-
3
3              KCB [RAB$B_RAC] = RAB$C_SEQ;
3              KCB [RAB$V_LOC] = 1;
3              KCB [RAB$V_RAH] = 1;
3
2              END;
2
2          !+
2          ! Store RESULTANT_NAME in KCB and caller's filespec, if desired.
2          !-
2
3          BEGIN
3          LOCAL
3              COPY_STATUS,
3              STRPTR,         ! Pointer to string
3              STRLEN: WORD;   ! Length of string
3
3          !+
3          ! Try resultant name string first.
3          !-
3          STRPTR = RESULTANT_STRING;
3          STRLEN = .NAM [NAM$B_RSL];
3          IF .STRLEN EQL 0
3          THEN
4              BEGIN
4              !+
4              ! Try expanded name string.
4              !-
4              STRLEN = .NAM [NAM$B_ESL];
4              IF .STRLEN EQL 0
4              THEN
5                  BEGIN
5                  !+
5                  ! Use filename.
5                  !-
5                  STRPTR = .FAB [FAB$L_FNA];
5                  STRLEN = .FAB [FAB$B_FNS];
4                  END;
3              END;
3
3          IF NOT NULLPARAMETER (K_RESULTANT_FILESPEC)
```

```
	THEN
	    BEGIN
	    COPY_STATUS = LIB$SCOPY_R_DX6 (.STRLEN, .STRPTR,
	        RESULTANT_FILESPEC [0,0,0,0]);
	    IF NOT .COPY_STATUS
	    THEN
	        RETURN SMG$$CLEANUP (KCB [KCB_R_KCB], .COPY_STATUS);
	    END;

	!+
	! Store the resultant name string in the KCB.  It may be used
	! later with output.
	!-
	BEGIN
	LOCAL
	    DEVNAM_DSC : BLOCK [8, BYTE],            ! dsc for name
	    DVI_ITMLST : VECTOR [1*3 + 1] INITIAL    ! item list for $GETDVI
	          (DVI$_DEVNAM     ^ 16 +64, 0, 0,!  result name string
	           0),                                !  terminater

	    DVI_IOSB      : VECTOR [4, WORD],        ! I/O Status block for $GETDVI
	    STATUS,                                  ! status retd by called routines
	    DEV_DEVNAM : VECTOR [64, BYTE],          ! Buffer for result name
	                                             ! string
	    DEV_NAMLEN : VOLATILE WORD;              ! Length of returned
	                                             ! resultant name string

	BIND
	    DVI_DEVNAM  = DVI_ITMLST + 4,            ! make it easy to ref items
	    DVI_NAMLEN  = DVI_ITMLST + 8;            !

	DVI_DEVNAM  = DEV_DEVNAM;
	DVI_NAMLEN  = DEV_NAMLEN;

	DEVNAM_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
	DEVNAM_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
	DEVNAM_DSC [DSC$W_LENGTH] = .STRLEN;
	DEVNAM_DSC [DSC$A_POINTER] = .STRPTR; ! dsc needed for $GETDVI

	STATUS = $GETDVI(   EFN      = .KCB [KCB_L_EFN],
	                    IOSB     =  DVI_IOSB,
	                    DEVNAM   =  DEVNAM_DSC,
	                    ITMLST   =  DVI_ITMLST);
	IF NOT .STATUS THEN RETURN (.STATUS);

	$WAITFR (EFN = .KCB [KCB_L_EFN]);           ! make $GETDVI synchronous

	!+
	! When the operation completes, the final status
	! is left in the first word of the I/O status block.
	!-

	IF NOT .DVI_IOSB [0] THEN RETURN (.DVI_IOSB [0]);

	KCB [KCB_W_DEVNAM_LENGTH]= .DEV_NAMLEN;     ! Length of device name
	CH$MOVE ( .DEV_NAMLEN, DEV_DEVNAM, KCB [KCB_T_DEVNAM_STRING]);
```

```
                 END;

                 END;

                 !+
                 ! Store the check value in the KCB so we can recognize a valid KCB
                 ! the next time we see one.
                 !-

                 KCB [KCB_L_CHECK] = .KCB;

                 !+
                 ! Init to no pasteboard.
                 !-

                 KCB [KCB_L_PASTEBOARD_ID] = -1;

                 !+
                 ! Set the terminal's keypad to application mode (if not a file).
                 !-

                 IF NOT .KCB [KCB_V_RMS]
                 THEN
                     SMG$SET_KEYPAD_MODE (%REF(KCB [KCB_R_KCB]), %REF(1));

                 !+
                 ! Store the KCB address in the user's argument.
                 !-

                 KEYBOARD_ID [0] = KCB [KCB_R_KCB];

                 RETURN SS$_NORMAL;

                 END;                                  ! End of routine SMG$CREATE_VIRTUAL_KEYBOARD

                                        .TITLE  SMG$INPUT Screen Management Facility Input Proc
                                                          edures
                                        .IDENT  \2-017\

                                        .PSECT  _SMG$DATA,NOEXE,  PIC,2

                   00000 KQB_QUEUE:
                                        .BLKB   8
                   00008 QPS_QUEUE:
                                        .BLKB   8
         00000000  00010 V_QUEUE_INIT:
                                        .LONG   0                                  ;

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

               1B  00000 T_KPDAPP_DECCRT:
                                        .BYTE   27                                 ;
               3D  00001                .ASCII  \=\                                ;
                   00002                .BLKB   2
               1B  00004 T_KPDNUM_DECCRT:
```

```
                                                       .BYTE    27                                           :
                                    3E 00005           .ASCII   \>\                                          :
  3A  54  55  50  4E  49  24  53  59  53 00006 P.AAA:  .ASCII   \SYS$INPUT:\                                 :
  00000000  00000000  00000000  00200040 00010 P.AAB:  .LONG    2097216, 0, 0, 0                             :

                                                       .EXTRN   LIB$GET_EF, LIB$FREE_EF
                                                       .EXTRN   LIB$GET_VM, LIB$FREE_VM
                                                       .EXTRN   LIB$ANALYZE_SDESC_R2
                                                       .EXTRN   LIB$SCOPY_R_DX, LIB$SCOPY_R_DX6
                                                       .EXTRN   SMG$$TERM_TO_KEYCODE
                                                       .EXTRN   SMG$$LOOKUP_KEY
                                                       .EXTRN   SMG$$GET_PASTEBOARD_ID
                                                       .EXTRN   SMG$$SET_PHYSICAL_CURSOR
                                                       .EXTRN   SMG$$REPORT_CHANGE_REPLACE
                                                       .EXTRN   SMG$$OUTPUT, SMG$$SET_ATTRIBUTES_ON
                                                       .EXTRN   SMG$$SET_ATTRIBUTES_OFF
                                                       .EXTRN   SYS$QIOW, SMG$_STRTERESC
                                                       .EXTRN   SMG$_EOF, SMG$_FILTOOLON
                                                       .EXTRN   SMG$_ILLBATFNC, SMG$_INVMAXLEN
                                                       .EXTRN   SMG$_INVSTANAM, SMG$_PROTOOLON
                                                       .EXTRN   SMG$_INVCOL, SMG$_INVDIS_ID
                                                       .EXTRN   LIB$AB_UPCASE, SMG$_WRONUMARG
                                                       .EXTRN   SYS$PARSE, SYS$ASSIGN
                                                       .EXTRN   SYS$OPEN, SYS$CONNECT
                                                       .EXTRN   SYS$GETDVI, SYS$WAITFR

                                  0FFC 00000           .ENTRY   SMG$CREATE_VIRTUAL_KEYBOARD, Save R2,R3,R4,-; 0276
                                                                R5,R6,R7,R8,R9,R10,R11                      :
                  5B 00000000G  00  9E 00002           MOVAB    LIB$ANALYZE_SDESC_R2, R11                   :
                  5A 00000000G  00  9E 00009           MOVAB    LIB$GET_VM, R10                             :
                  5E      FDAC  CE  9E 00010           MOVAB    -596(SP), SP                                :
          50      6C            01  83 00015           SUBB3    #1, (AP), DIFF                              : 0372
                  03            50  91 00019           CMPB     DIFF, #3                                    :
                  08                1B 0001C           BLEQU    1$                                          :
                  50 00000000G  8F  D0 0001E           MOVL     #SMG$_WRONUMARG, R0                         :
                                    04 00025           RET                                                  :
0050  8F  00      6E            00  2C 00026 1$:       MOVC5    #0, (SP), #0, #80, $RMS_PTR                 : 0385
                          B0    AD     0002D                                                                :
          B0  AD          5003  8F  B0 0002F           MOVW     #20483, $RMS_PTR                            :
          C6  AD            02  90 00035               MOVB     #2, $RMS_PTR+22                             :
          CF  AD            02  90 00039               MOVB     #2, $RMS_PTR+31                             :
          D4  AD          FF24  CD  9E 0003D           MOVAB    XAB_FHC, $RMS_PTR+36                        :
          D8  AD          FF50  CD  9E 00043           MOVAB    NAM, $RMS_PTR+40                            :
0060  8F  00      6E            00  2C 00049           MOVC5    #0, (SP), #0, #96, $RMS_PTR                 : 0387
                          FF50  CD     00050                                                                :
        FF50  CD          6002  8F  B0 00053           MOVW     #24578, $RMS_PTR                            :
        FF52  CD            01  8E 0005A               MNEGB    #1, $RMS_PTR+2                              :
        FF54  CD            78  AE  9E 0005F           MOVAB    RESULTANT_STRING, $RMS_PTR+4                :
        FF5A  CD            01  8E 00065               MNEGB    #1, $RMS_PTR+10                             :
        FF5C  CD            78  AE  9E 0006A           MOVAB    RESULTANT_STRING, $RMS_PTR+12               :
  2C      00      6E            00  2C 00070           MOVC5    #0, (SP), #0, #44, $RMS_PTR                 : 0388
                          FF24  CD     00075                                                                :
        FF24  CD          2C1D  8F  B0 00078           MOVW     #11293, $RMS_PTR                            :
              02            6C  91 0007F               CMPB     (AP), #2                                    : 0394
                            1F  1F 00082               BLSSU    2$                                          :
                      08    AC  D5 00084               TSTL     8(AP)                                       :
                            1A  13 00087               BEQL     2$                                          :
```

```
                             50         08    AC  D0 00089          MOVL     FILESPEC, R0                           ; 0402
                                              6B  16 0008D          JSB      LIB$ANALYZE_SDESC_R2                   ;
                             6F               50  E9 0008F          BLBC     ANL_STATUS, 8$                         ; 0404
                   00FF      8F               51  B1 00092          CMPW     STRING_LEN, #255                       ; 0408
                                              2E  1A 00097          BGTRU    4$                                     ;
                     DC      AD               52  D0 00099          MOVL     STRING_PTR, FAB+44                     ; 0411
                     E4      AD               51  90 0009D          MOVB     STRING_LEN, FAB+52                     ; 0412
                                              0A  11 000A1          BRB      3$                                     ; 0394
                     E4      AD               0A  90 000A3  2$:     MOVB     #10, FAB+52                            ; 0419
                     DC      AD       FF3B    CF  9E 000A7          MOVAB    P.AAA, FAB+44                          ; 0420
                             03               6C  91 000AD  3$:     CMPB     (AP), #3                               ; 0427
                                              25  1F 000B0          BLSSU    6$                                     ;
                                        0C    AC  D5 000B2          TSTL     12(AP)                                 ;
                                              20  13 000B5          BEQL     6$                                     ;
                             50         0C    AC  D0 000B7          MOVL     DEFAULT_FILESPEC, R0                   ; 0435
                                              6B  16 000BB          JSB      LIB$ANALYZE_SDESC_R2                   ;
                             41               50  E9 000BD          BLBC     ANL_STATUS, 8$                         ; 0437
                   00FF      8F               51  B1 000C0          CMPW     STRING_LEN, #255                       ; 0441
                                              08  1B 000C5          BLEQU    5$                                     ;
                             50  00000000G    8F  D0 000C7  4$:     MOVL     #SMG$_FILTOOLON, R0                    ; 0443
                                              04  000CE             RET                                             ;
                     E0      AD               52  D0 000CF  5$:     MOVL     STRING_PTR, FAB+48                     ; 0444
                     E5      AD               51  90 000D3          MOVB     STRING_LEN, FAB+53                     ; 0445
                                        B0    AD  9F 000D7  6$:     PUSHAB   FAB                                    ; 0456
              00000000G      00               01  FB 000DA          CALLS    #1, SYS$PARSE                          ;
                             1D               50  E9 000E1          BLBC     PARSE_STATUS, 8$                       ; 0457
                             08  00000000'    EF  E8 000E4          BLBS     V_QUEUE_INIT, 7$                       ; 0467
                  0000V      CF               00  FB 000EB          CALLS    #0, SMG$$INIT_QUEUES                   ; 0472
                             0E               50  E9 000F0          BLBC     INIT_STATUS, 8$                        ; 0473
                                        08    AE  9F 000F3  7$:     PUSHAB   VM_POINTER                             ; 0486
                     08      AE         E2    8F  9A 000F6          MOVZBL   #226, 8(SP)                            ;
                                        08    AE  9F 000FB          PUSHAB   8(SP)                                  ;
                             6A               02  FB 000FE          CALLS    #2, LIB$GET_VM                         ;
                             01               50  E8 00101  8$:     BLBS     VM_STATUS, 9$                          ; 0487
                                              04  00104             RET                                             ;
    00E2     8F       00                 6E   00  2C 00105  9$:     MOVC5    #0, (SP), #0, #226, @VM_POINTER        ; 0490
                                        08    BE     0010C                                                          ;
             58      08      AE               30  C1 0010E          ADDL3    #48, VM_POINTER, KCB                   ; 0491
                                        0C    AE  9F 00113          PUSHAB   KQB_PTR                                ; 0508
                     08      AE               0C  D0 00116          MOVL     #12, 8(SP)                             ;
                                        08    AE  9F 0011A          PUSHAB   8(SP)                                  ;
                             6A               02  FB 0011D          CALLS    #2, LIB$GET_VM                         ;
                             54               50  E9 00120          BLBC     VM_STATUS, 12$                         ; 0509
                             50         0C    AE  D0 00123          MOVL     KQB_PTR, KQB                           ; 0512
                     08      A0               58  D0 00127          MOVL     KCB, 8(KQB)                            ; 0513
                     D8      A8               50  D0 0012B          MOVL     KQB, -40(KCB)                          ; 0514
              00000000'      EF               60  0E 0012F          INSQUE   (KQB), KQB_QUEUE                       ; 0515
             04      F0      AD               02  E0 00136          BBS      #2, FAB_DEV, 10$                       ; 0525
                     FC      A8               01  88 0013B          BISB2    #1, -4(KCB)                            ; 0527
                             03         FC    A8  E9 0013F  10$:    BLBC     -4(KCB), 11$                           ; 0534
                                        00B5      31 00143          BRW      16$                                    ;
                             52         F4    A8  9E 00146  11$:    MOVAB    -12(KCB), R2                           ; 0547
                     70      AE       FF5B    CD  9B 0014A          MOVZBW   NAM+11, DEVNAME_DESCR                  ; 0553
                     72      AE       010E    8F  B0 00150          MOVW     #270, DEVNAME_DESCR+2                  ; 0555
                     74      AE         78    AE  9E 00156          MOVAB    RESULTANT_STRING, DEVNAME_DESCR+4      ; 0556
                                              7E  7C 0015B          CLRQ     -(SP)                                  ; 0559
                                        D4    A8  9F 0015D          PUSHAB   -44(KCB)                               ;
```

```
                          7C   AE  9F 00160          PUSHAB   DEVNAME_DESCR                  ;
      00000000G  00            04  FB 00163          CALLS    #4, SYS$ASSIGN                 ;
                 56            50  E9 0016A          BLBC     ASSIGN_STATUS, 13$             ; 0560
                          D0   A8  9F 0016D          PUSHAB   -48(KCB)                       ; 0571
      00000000G  00            01  FB 00170          CALLS    #1, LIB$GET_EF                 ;
                 49            50  E9 00177   12$:   BLBC     EF_STATUS, 13$                 ; 0572
                 68            0C  D0 0017A          MOVL     #12, (KCB)                     ; 0581
           04    A8       D0   A8  D0 0017D          MOVL     -48(KCB), 4(KCB)               ; 0582
           08    A8       D4   A8  32 00182          CVTWL    -44(KCB), 8(KCB)               ; 0583
           0C    A8     4031   8F  3C 00187          MOVZWL   #16433, 12(KCB)                ; 0584
                 50       68   A8  9E 0018D          MOVAB    104(R8), R0                    ; 0585
           10    A8            50  D0 00191          MOVL     R0, 16(KCB)                    ;
           34    A8            0C  D0 00195          MOVL     #12, 52(KCB)                   ; 0586
           38    A8       D0   A8  D0 00199          MOVL     -48(KCB), 56(KCB)              ; 0587
           3C    A8       D4   A8  32 0019E          CVTWL    -44(KCB), 60(KCB)              ; 0588
           44    A8            50  D0 001A3          MOVL     R0, 68(KCB)                    ; 0589
           40    A8            27  D0 001A7          MOVL     #39, 64(KCB)                   ; 0596
           50    A8       EC   A8  9E 001AB          MOVAB    -20(R8), 80(KCB)               ; 0597
           54    A8            0C  D0 001B0          MOVL     #12, 84(KCB)                   ; 0598
      00000000G  00       34   A8  FA 001B4          CALLG    52(KCB), SYS$QIOW              ; 0600
                 52            50  E9 001BC          BLBC     QIO_STATUS1, 17$               ; 0601
                 50       68   A8  32 001BF          CVTWL    104(KCB), QIO_STATUS1          ; 0603
                 4B            50  E9 001C3   13$:   BLBC     QIO_STATUS1, 17$               ; 0604
           E0    A8       F0   A8  7D 001C6          MOVQ     -16(KCB), -32(KCB)             ; 0612
    26           62            1D  E0 001CB          BBS      #29, (R2), 15$                 ; 0620
           40    8F       ED   A8  91 001CF          CMPB     -19(KCB), #64                  ; 0621
                               1F  13 001D4          BEQL     15$                            ;
           41    8F       ED   A8  91 001D6          CMPB     -19(KCB), #65                  ; 0622
                               18  13 001DB          BEQL     15$                            ;
                 20       ED   A8  91 001DD          CMPB     -19(KCB), #32                  ; 0623
                               06  1F 001E1          BLSSU    14$                            ;
                 25       ED   A8  91 001E3          CMPB     -19(KCB), #37                  ; 0624
                               0C  1B 001E7          BLEQU    15$                            ;
                 02       ED   A8  91 001E9   14$:   CMPB     -19(KCB), #2                   ; 0625
                               06  13 001ED          BEQL     15$                            ;
                 03       ED   A8  91 001EF          CMPB     -19(KCB), #3                   ; 0626
                               6D  12 001F3          BNEQ     20$                            ;
           FC    A8            08  88 001F5   15$:   BISB2    #8, -4(KCB)                    ; 0628
                               67  11 001F9          BRB      20$                            ; 0534
                          B0   AD  9F 001FB   16$:   PUSHAB   FAB                            ; 0645
      00000000G  00            01  FB 001FE          CALLS    #1, SYS$OPEN                   ;
           10    AE            50  D0 00205          MOVL     R0, OPEN_STATUS                ;
                 07       10   AE  E8 00209          BLBS     OPEN_STATUS, 18$               ; 0646
                 50       10   AE  D0 0020D          MOVL     OPEN_STATUS, R0                ; 0648
                             0085  31 00211   17$:   BRW      22$                            ;
                 68     4401   8F  B0 00214   18$:   MOVW     #17409, (KCB)                  ; 0650
           3C    A8       B0   AD  9E 00219          MOVAB    FAB, 60(KCB)                   ; 0652
                               58  DD 0021E          PUSHL    KCB                            ; 0658
      00000000G  00            01  FB 00220          CALLS    #1, SYS$CONNECT                ;
                 6F            50  E9 00227          BLBC     CONNECT_STATUS, 22$            ; 0659
           D4    A8       B2   AD  B0 0022A          MOVW     FAB+2, -44(KCB)                ; 0667
                 50       20   A8  9E 0022F          MOVAB    32(KCB), R0                    ; 0673
                 60       E6   AD  B0 00233          MOVW     FAB+54, (R0)                   ;
                               10  12 00237          BNEQ     19$                            ; 0674
                 60       EC   AD  B0 00239          MOVW     FAB+60, (R0)                   ; 0677
                 60     FF2E   CD  B1 0023D          CMPW     XAB_FHC+10, (R0)               ; 0678
                               05  1B 00242          BLEQU    19$                            ;
```

```
                 60      FF2E   CD  B0 00244        MOVW    XAB FHC+10, (R0)                          ; 0680
                           24   A8  9F 00249 19$:   PUSHAB  36(KCB)                                   ; 0682
             08  AE             60  3C 0024C        MOVZWL  (R0), 8(SP)                               ;
                           08   AE  9F 00250        PUSHAB  8(SP)                                     ;
                 6A             02  FB 00253        CALLS   #2, LIB$GET_VM                            ;
                 40             50  E9 00256        BLBC    VM_STATUS, 22$                            ; 0683
                           1E   A8  94 00259        CLRB    30(KCB)                                   ; 0691
             05  A8      0102   8F  A8 0025C        BISW2   #258, 5(KCB)                              ; 0692
                 59        78   AE  9E 00262 20$:   MOVAB   RESULTANT_STRING, STRPTR                  ; 0710
                 57      FF53   CD  9B 00266        MOVZBW  NAM+3, STRLEN                             ; 0711
                                0F  12 0026B        BNEQ    21$                                       ; 0712
                 57      FF5B   CD  9B 0026D        MOVZBW  NAM+11, STRLEN                            ; 0718
                                08  12 00272        BNEQ    21$                                       ; 0719
                 59        DC   AD  D0 00274        MOVL    FAB+44, STRPTR                            ; 0725
                 57        E4   AD  9B 00278        MOVZBW  FAB+52, STRLEN                            ; 0726
                 04             6C  91 0027C 21$:   CMPB    (AP), #4                                  ; 0730
                                1E  1F 0027F        BLSSU   23$                                       ;
                           10   AC  D5 00281        TSTL    16(AP)                                    ;
                                19  13 00284        BEQL    23$                                       ;
                 52        10   AC  D0 00286        MOVL    RESULTANT_FILESPEC, R2                    ; 0734
                 51             59  D0 0028A        MOVL    STRPTR, R1                                ;
                 50             57  3C 0028D        MOVZWL  STRLEN, R0                                ;
                     00000000G  00  16 00290        JSB     LIB$SCOPY_R_DX6                           ;
                 06             50  E8 00296        BLBS    COPY_STATUS, 23$                          ; 0735
           0000V CF             00  FB 00299 22$:   CALLS   #0, SMG$$CLEANUP                          ; 0737
                                    04 0029E        RET                                               ;
   60    AE      FD4C   CF      10  28 0029F 23$:   MOVC3   #16, P.AAB, DVI_ITMLST                    ; 0749
                 64  AE    18   AE  9E 002A6        MOVAB   DEV_DEVNAM, DVI_DEVNAM                    ; 0762
                 68  AE    16   AE  9E 002AB        MOVAB   DEV_NAMLEN, DVI_NAMLEN                    ; 0763
                 72  AE  010E   8F  B0 002B0        MOVW    #270, DEVNAM_DSC+2                        ; 0765
                 70  AE             57  B0 002B6    MOVW    STRLEN, DEVNAM_DSC                        ; 0767
                 74  AE             59  D0 002BA    MOVL    STRPTR, DEVNAM_DSC+4                      ; 0768
                                7E  7C 002BE        CLRQ    -(SP)                                     ; 0773
                                7E  D4 002C0        CLRL    -(SP)                                     ;
                           64   AE  9F 002C2        PUSHAB  DVI_IOSB                                  ;
                           70   AE  9F 002C5        PUSHAB  DVI_ITMLST                                ;
                         0084   CE  9F 002C8        PUSHAB  DEVNAM_DSC                                ;
                                7E  D4 002CC        CLRL    -(SP)                                     ;
                           D0   A8  DD 002CE        PUSHL   -48(KCB)                                  ;
       00000000G 00             08  FB 002D1        CALLS   #8, SYS$GETDVI                            ;
                 45             50  E9 002D8        BLBC    STATUS, 26$                               ; 0774
                           D0   A8  DD 002DB        PUSHL   -48(KCB)                                  ; 0776
       00000000G 00             01  FB 002DE        CALLS   #1, SYS$WAITFR                            ;
                 05        58   AE  E8 002E5        BLBS    DVI_IOSB, 24$                             ; 0783
                 50        58   AE  3C 002E9        MOVZWL  DVI_IOSB, R0                              ;
                                    04 002ED        RET                                               ;
             70  A8        16   AE  B0 002EE 24$:   MOVW    DEV_NAMLEN, 112(KCB)                      ; 0786
   72    A8  18  AE        16   AE  28 002F3        MOVC3   DEV_NAMLEN, DEV_DEVNAM, 114(KCB)          ; 0787
             DC  A8             58  D0 002FA        MOVL    KCB, -36(KCB)                             ; 0798
             F8  A8             01  CE 002FE        MNEGL   #1, -8(KCB)                               ; 0804
                 13        FC   A8  E8 00302        BLBS    -4(KCB), 25$                              ; 0810
             04  AE             01  D0 00306        MOVL    #1, 4(SP)                                 ; 0812
                           04   AE  9F 0030A        PUSHAB  4(SP)                                     ;
             04  AE             58  D0 0030D        MOVL    KCB, 4(SP)                                ;
                           04   AE  9F 00311        PUSHAB  4(SP)                                     ;
           0000V CF             02  FB 00314        CALLS   #2, SMG$SET_KEYPAD_MODE                   ;
             04  BC             58  D0 00319 25$:   MOVL    KCB, @KEYBOARD_ID                         ; 0818
```

```
                    50          01  D0 0031D         MOVL    #1, R0                                  ; 0820
                                    04 00320 26$:    RET                                             ; 0822

; Routine Size:  801 bytes,    Routine Base:  _SMG$CODE + 0020
```

```
%SBTTL 'SMG$READ_STRING'
GLOBAL ROUTINE SMG$READ_STRING (
    KEYBOARD_ID: REF VECTOR [, LONG],
    OUT_STRING: REF BLOCK [, BYTE],
    PROMPT_STRING: REF BLOCK [, BYTE],
    MAX_LENGTH: REF VECTOR [, LONG],
    MODIFIERS: REF VECTOR [, LONG],
    TIMEOUT: REF VECTOR [, LONG],
    TERMINATOR_SET: REF BLOCK [,BYTE],
    OUT_LENGTH: REF VECTOR [, WORD],
    TERMINATOR_CODE: REF VECTOR [, WORD],
    DISPLAY_ID: REF VECTOR [, LONG]
    ) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine reads a sequence of characters from the virtual-keyboard
!       specified and returns to the caller the characters read.  The caller
!       may also specify that a code indicating the terminator be returned.
!
! CALLING SEQUENCE:
!
!       RET_STATUS.wlc.v = SMG$READ_STRING (
!                               KEYBOARD_ID.rl.r,
!                               OUT_STRING.wt.dx
!                               [, [PROMPT_STRING.rt.dx]
!                               [, [MAX_LENGTH.rl.r]
!                               [, [MODIFIERS.rl.r]
!                               [, [TIMEOUT.rl.r]
!                               [, [TERMINATOR_SET.rt.ds]
!                               [, [OUT_LENGTH.wwu.r]
!                               [, [TERMINATOR_CODE.wwu.r]
!                               [, [DISPLAY_ID.rl.r] ]]]]]]]])
!
!
! FORMAL PARAMETERS:
!
!       KEYBOARD_ID     A longword containing the identification of the
!                       virtual-keyboard from which to read.  A virtual-
!                       keyboard is created by calling
!                       SMG$CREATE_VIRTUAL_KEYBOARD.
!
!       PROMPT_STRING   An optional string to be used as the prompt for
!                       the read operation.
!
!       OUT_STRING      A string into which is written the characters read.
!
!       MAX_LENGTH      A longword containing the maximum number of characters
!                       to read.  The maximum value of MAX_LENGTH is 512.  If
!                       omitted, 512 is used.
!
!       MODIFIERS       A longword bitmask that specifies optional behavior.
!                       The bits defined are the same as for the $QIO
!                       item-list entry TRM$_MODIFIERS.
!
!                       See the terminal driver section of the I/O User's
```

```
!                               Guide for more information on MODIFIERS.  The
!                               TRM$ symbols are defined by the $TRMDEF macro/module
!                               in Digital-supplies system symbol libraries.
!
!       TIMEOUT                 Optional timeout count.  If specified, all characters
!                               typed in before the timeout are returned in the buffer.
!                               If omitted, characters are returned in the buffer until
!                               a terminator is seen.
!
!       TERMINATOR_SET          Descriptor of terminator mask, 0 - 8 bytes
!                               (class, dtype ignored).  If omitted, the VAX-11
!                               RMS standard terminator set is used.  See the terminal
!                               driver section of the VAX/VMS I/O User's Guide for
!                               more information on specifying terminators.
!
!       OUT_LENGTH              An unsigned word into which is written the number
!                               of characters read or the maximum length of
!                               OUT_STRING, whichever is less.
!
!       TERMINATOR_CODE         An unsigned word into which is written a code
!                               indicating what character or key terminated the
!                               read.  For more information on the values for
!                               TERMINATOR_CODE, see the description of terminator
!                               codes in the Screen Management functional specification.
!
!       DISPLAY_ID              A longword containing the identification of the
!                               virtual display in which the read is to be performed.
!                               If specified, the read begins at the current cursor
!                               position in that virtual-display.  If omitted, the
!                               read begins in the current cursor position.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! COMPLETION STATUS:
!
!       SS$_NORMAL              Normal successful completion
!       SS$_CANCEL              I/O operation cancelled (by SMG$CANCEL_INPUT)
!       SMG$_EOF                End-of-file
!       SMG$_INVDIS_ID          Invalid display-id
!       SMG$_INVKBD_ID          Invalid keyboard-id
!       SMG$_WRONUMARG          Wrong number of arguments
!       LIB$_xxx                Any error from LIB$SCOPY_R_DX
!       RMS$_xxx                Any error from $GET (except RMS$_EOF)
!       SS$_xxx                 Any error from $QIOW
!
! SIDE EFFECTS:
!
!       NONE
!
!--
```

```
BEGIN

LOCAL
    KCB: REF KCB_R_KCB_STRUCT,
    BUFFER: VECTOR [512, BYTE],
    BUFFER_PTR,
    BUFFER_LEN: WORD,
    KEY_CODE: WORD,
    ITEM_LIST: $ITMLST_DECL (ITEMS=4),        ! QIO item list
    QIO_STATUS,
    COPY_STATUS,
    RET_STATUS,
    DISPLAY_ID_FLAG : INITIAL (0);

BUILTIN
    ACTUALCOUNT,
    CALLG,
    NULLPARAMETER,
    TESTBITSC;

!+
! Define literals for the argument positions.
!-

LITERAL
    K_KEYBOARD_ID = 1,
    K_OUT_STRING = 2,
    K_PROMPT_STRING = 3,
    K_MAX_LENGTH = 4,
    K_MODIFIERS = 5,
    K_TIMEOUT = 6,
    K_TERMINATOR_SET = 7,
    K_OUT_LENGTH = 8,
    K_TERMINATOR_CODE = 9,
    K_DISPLAY_ID = 10;

!+
! Define literals for the item list positions.
!-

LITERAL
    K_MODIFIER_MASK = 0,
    K_PROMPT = 1,
    K_NEXT = 2,      ! The next available position
    K_ALLOWED_MODIFIERS = IO$M_CVTLOW +  ! 3A modifiers
                          IO$M_DSABLMBX +        ! 5f40
                          IO$M_ESCAPE +
                          IO$M_NOECHO +
                          IO$M_NOFILTR +
                          IO$M_PURGE +
                          IO$M_TRMNOECHO;

$SMG$VALIDATE_ARGCOUNT(2,10);

!+
! Fetch and validate KCB.
!-
```

```
	$SMG$VALIDATE_KCB (KEYBOARD_ID, KCB);

	!+
	! Assume success.
	!-

	RET_STATUS = SS$_NORMAL;

	!+
	! Split depending on whether or not we are using RMS.
	!-

	IF .KCB [KCB_V_RMS]
	THEN

	    !+
	    ! Read a line using $GET.
	    !-

	    BEGIN
	    LOCAL
		GET_STATUS;

	    GET_STATUS = $GET (RAB = KCB [KCB_R_RAB]);
	    IF NOT .GET_STATUS
	    THEN
		BEGIN
		IF .GET_STATUS EQLU RMS$_EOF
		THEN
		    RET_STATUS = SMG$_EOF
		ELSE
		    RET_STATUS = .GET_STATUS;
		END;

	    !+
	    ! If user asked for TERMINATOR_CODE, return <CR>.
	    !-

	    IF NOT NULLPARAMETER (K_TERMINATOR_CODE)
	    THEN
		TERMINATOR_CODE [0] = SMG$K_TRM_CR;

	    !+
	    ! Set BUFFER_PTR and BUFFER_LEN.
	    !-

	    BUFFER_PTR = .KCB [RAB$L_RBF];
	    BUFFER_LEN = .KCB [RAB$W_RSZ];
	    END

	ELSE

	    !+
	    ! Use $QIO.
	    !-

```

```
BEGIN
LOCAL
    CURSOR_BUF : VECTOR [15, BYTE],       ! buffer for set cursor sequence
    CURSOR_BUF_LEN : INITIAL (0),         ! used number of bytes in cursor_buf
    FREE_COLS;                            ! calc. needed if display_id used

!+
! If the last read ended in ^Z, return SMG$_EOF.
!-

IF TESTBITSC (KCB [KCB_V_CTRLZ])
THEN
    RETURN SMG$_EOF;        ! End of file

!+
! Build the item list for the QIO.
!-

$ITMLST_INIT (ITMLST=ITEM_LIST,
    (ITMCOD=TRM$_MODIFIERS, BUFADR=TRM$M_TM_ESCAPE, BUFSIZ=0),
    (ITMCOD=TRM$_PROMPT, BUFADR=0, BUFSIZ=0));

!+
! Set up the $QIOW argument list.
!-

KCB [KCB_L_QIO1_FUNC] = IO$_READVBLK+IO$M_EXTEND;
KCB [KCB_L_QIO1_P1] = BUFFER;
KCB [KCB_L_QIO1_P2] = %ALLOCATION(BUFFER);
KCB [KCB_L_QIO1_P3] = 0;
KCB [KCB_L_QIO1_P4] = 0;
KCB [KCB_L_QIO1_P5] = ITEM_LIST;
KCB [KCB_L_QIO1_P6] = %ALLOCATION(ITEM_LIST) - 28; ! Don't count timeout
                                                   ! or term. set yet,
                                                   ! and $ITMLST_DECL
                                                   ! adds extra 4.

!+
! Check for optional input parameters.  This is done as a nested if
! rather than as a series of NULLPARAMETER tests for speed.
!-

IF ACTUALCOUNT () GEQU K_MAX_LENGTH
THEN
    BEGIN
    LOCAL
        CUR_ITEM;           ! Pointer into ITEM_LIST
    CUR_ITEM = ITEM_LIST [K_NEXT,0,0,0,0];

    IF MAX_LENGTH [0] NEQA 0
    THEN
        BEGIN
        KCB [KCB_L_QIO1_P2] = .MAX_LENGTH [0];
        IF .KCB [KCB_L_QIO1_P2] GTRU %ALLOCATION(BUFFER)
        THEN
            RETURN SMG$_INVMAXLEN;
        END;
```

```
IF ACTUALCOUNT () GEQU K_MODIFIERS
THEN
    BEGIN
    IF MODIFIERS [0] NEQA 0
    THEN
        ITEM_LIST [K_MODIFIER_MASK,ITM$L_BUFADR] =
            .ITEM_LIST [K_MODIFIER_MASK,ITM$L_BUFADR] OR
            .MODIFIERS [0];
    IF ACTUALCOUNT () GEQU K_TIMEOUT
    THEN
        BEGIN
        IF TIMEOUT [0] NEQA 0
        THEN
            BEGIN
            $LIB$WWORD_A (CUR_ITEM) = 0;
            $LIB$WWORD_A (CUR_ITEM) = TRM$_TIMEOUT;
            $LIB$WLONG_A (CUR_ITEM) = .TIMEOUT [0];
            $LIB$WLONG_A (CUR_ITEM) = 0;
            END;
        IF ACTUALCOUNT () GEQU K_TERMINATOR_SET
        THEN
            BEGIN
            IF TERMINATOR_SET [0,0,0,0] NEQA 0
            THEN
                BEGIN
                $LIB$WWORD_A (CUR_ITEM) = .TERMINATOR_SET [DSC$W_LENGTH];
                $LIB$WWORD_A (CUR_ITEM) = TRM$_TERM;
                $LIB$WLONG_A (CUR_ITEM) = .TERMINATOR_SET [DSC$A_POINTER];
                $LIB$WLONG_A (CUR_ITEM) = 0;
                END;
            IF NOT NULLPARAMETER (K_DISPLAY_ID)
            THEN
                BEGIN                           ! display_id present
                !+
                ! Interfacing with output's virtual displays.  Make sure user can
                ! not type beyond the display's boundaries by limiting the input
                ! size to the number of free columns (allowing room for prompt, if
                ! any).
                ! Notice that we need some info from SMG$$SET_PROMPT_STRING (prompt
                ! string length), and we need to provide it with some info (set cursor
                ! sequence, if any).  So get the info here and save it.
                !+
                LOCAL
                    OUT_STATUS;
                DISPLAY_ID_FLAG = 1;            ! for quick checks later
                IF .KCB [KCB_L_PASTEBOARD_ID] LSS 0 ! pb not established yet
                THEN
                    BEGIN
                    OUT_STATUS = SMG$$GET_PASTEBOARD_ID (%REF (.KCB [KCB_W_DEVNAM_LENGTH]),
                                                         KCB [KCB_T_DEVNAM_STRING],
                                                         KCB [KCB_L_PASTEBOARD_ID]);
                    IF NOT .OUT_STATUS THEN RETURN (.OUT_STATUS);
                    END;
                OUT_STATUS = SMG$$SET_PHYSICAL_CURSOR (.DISPLAY_ID,
                                                       KCB [KCB_L_PASTEBOARD_ID],
                                                       CURSOR_BUF,
                                                       CURSOR_BUF_LEN,
```

```
                                                            FREE_COLS);
                                IF NOT .OUT_STATUS THEN RETURN .OUT_STATUS;
                                END;                            ! display_id present
                            END;
                        END;
                    END;

                !+
                ! Change item-list length to reflect modifications.
                !-

                KCB [KCB_L_QIO1_P6] = .CUR_ITEM - .KCB [KCB_L_QIO1_P5];
                END;

            !+
            ! Set up the prompt string.
            !-

                BEGIN
                LOCAL
                    STATUS,
                    NULL_STRING : BLOCK [8,BYTE];
                NULL_STRING [DSC$B_DTYPE] = DSC$K_DTYPE_T;
                NULL_STRING [DSC$B_CLASS] = DSC$K_CLASS_S;
                NULL_STRING [DSC$W_LENGTH] = 0;
                NULL_STRING [DSC$A_POINTER] = UPLIT (' ');

                IF .DISPLAY_ID_FLAG OR
                   NOT NULLPARAMETER (K_PROMPT_STRING)
                THEN
                    BEGIN                               ! setup prompt
                    STATUS = SMG$$SET_PROMPT_STRING (
                                    (IF NULLPARAMETER (K_PROMPT_STRING)
                                    THEN
                                        NULL_STRING
                                    ELSE
                                        PROMPT_STRING [0,0,0,0]),
                                    ITEM_LIST [K_PROMPT, 0,0,0,0],
                                    .CURSOR_BUF_LEN,
                                    CURSOR_BUF);
                    IF NOT .STATUS
                    THEN
                        RETURN .STATUS;
                    END;                                ! setup prompt
                IF .DISPLAY_ID_FLAG
                THEN
                    BEGIN
                    IF PROMPT_STRING [0,0,0,0] NEQA 0
                    THEN
                        FREE_COLS = .FREE_COLS - .PROMPT_STRING [DSC$W_LENGTH];
                    IF .FREE_COLS LEQ 0
                    THEN
                        !+
                        ! No room to accept input.  Exit.
                        !-
                        RETURN (SMG$_INVCOL);           ! improve this message later
```

```
                IF .KCB [KCB_L_QIO1_P2] GTR .FREE_COLS
                THEN
                    KCB [KCB_L_QIO1_P2] = .FREE_COLS;
                END;
            END;

        !**
        ! If this routine is supposed to execute on a V3.n system, map back
        ! the item-list entries into the old-style of $QIO.
        !**

        %IF %VARIANT
        %THEN
            BEGIN
            KCB [KCB_L_QIO1_FUNC] = IO$_READVBLK;
            KCB [KCB_L_QIO1_P5] = 0;
            KCB [KCB_L_QIO1_P6] = 0;
            IF .ITEM_LIST [K_PROMPT, ITM$W_BUFSIZ] NEQ 0
            THEN
                BEGIN
                KCB [KCB_L_QIO1_P5] = .ITEM_LIST [K_PROMPT, ITM$L_BUFADR];
                KCB [KCB_L_QIO1_P6] = .ITEM_LIST [K_PROMPT, ITM$W_BUFSIZ];
                KCB [KCB_L_QIO1_FUNC] = IO$_READPROMPT;
                END;
            IF NOT NULLPARAMETER(K_TIMEOUT)
            THEN
                BEGIN
                KCB [KCB_L_QIO1_P3] = .TIMEOUT [0];
                KCB [KCB_L_QIO1_FUNC] = .KCB [KCB_L_QIO1_FUNC] OR IO$M_TIMED;
                END;
            IF NOT NULLPARAMETER(K_TERMINATOR_SET)
            THEN
                BEGIN
                LOCAL
                    CUR_ITEM: REF $ITMLST_DECL (ITEMS=1);
                CUR_ITEM = ITEM_LIST [K_NEXT, 0,0,0,0];
                IF .CUR_ITEM [0, ITM$W_ITMCOD] NEQ TRM$_TERM
                THEN
                    CUR_ITEM = CUR_ITEM [1, 0,0,0,0];
                KCB [KCB_L_QIO1_P4] = .CUR_ITEM;
                END;

            IF NOT NULLPARAMETER (K_MODIFIERS)
            THEN
                KCB [KCB_L_QIO1_FUNC] = .KCB [KCB_L_QIO1_FUNC] OR
                                        (.MODIFIERS [0] AND K_ALLOWED_MODIFIERS);

            END;
        %FI     !** End of variant code

        BEGIN                               ! QIO w/rendition
        LOCAL
            PBCB : REF BLOCK [,BYTE],
            DCB : REF BLOCK [,BYTE];

        IF .DISPLAY_ID_FLAG
        THEN
```

```
5               BEGIN                           ! display_id/check rendition
5
5               !+
5               ! Make sure the prompt and input appear in the correct video
5               ! rendition.
5               !-
5
5               $SMG$GET_DCB (.DISPLAY_ID, DCB);
5               $SMG$GET_PBCB (KCB [KCB_L_PASTEBOARD_ID], PBCB);
5
5               !+
5               ! Don't allow input if display is batched.
5               !-
5
5               IF .DCB[DCB_L_BATCH_LEVEL] NEQ 0
5               THEN
5                   RETURN  SMG$_ILLBATFNC;
5
5               !+
5               ! Handle renditions if necessary.
5               !-
5
5               SMG$$SET_ATTRIBUTES_ON (.PBCB,.DCB [DCB_B_DEF_VIDEO_ATTR]);
5
4               END;                            ! display_id/check rendition
4
4           !+
4           ! Read the string.
4           !-
4
4           QIO_STATUS = CALLG (KCB [KCB_R_QIO1], SYS$QIOW);
4           IF .QIO_STATUS
4           THEN
4               QIO_STATUS = .KCB [KCB_W_IOSB_STATUS];
4
4           !+
4           ! If we set the video rendition, turn it off now.
4           !-
4
4           IF .DISPLAY_ID_FLAG
4           THEN
4               SMG$$SET_ATTRIBUTES_OFF (.PBCB,.DCB [DCB_B_DEF_VIDEO_ATTR]);
4
4           IF NOT .QIO_STATUS
4           THEN
4               RET_STATUS = .QIO_STATUS;
3           END;                        ! QIO w/rendition
3
3           !+
3           ! Check for partial escape sequence in the buffer.  This can
3           ! happen if the QIO was limited to 2 characters and the input
3           ! string was terminated by a keypad key (which are 3 chars).
3           !-
3
3           IF .KCB [KCB_W_IOSB_STATUS] EQL SS$_PARTESCAPE
3           THEN
4               BEGIN
```

```
            LOCAL
                QIO_RESULT,
                CHAR_BUF : BYTE,
                END_ESC_FLAG : INITIAL (0);

            BUFFER_LEN = .KCB [KCB_W_IOSB_COUNT] + .KCB [KCB_B_IOSB_TERMLEN];
                                        ! point past data & partial terminator

            WHILE .END_ESC_FLAG EQL 0 DO
                BEGIN
                QIO_RESULT = $QIOW (EFN = .KCB [KCB_L_EFN],
                                    CHAN = .KCB [KCB_W_CHANNEL],
                                    FUNC = (IO$_READVBLK + IO$M_ESCAPE +
                                            IO$M_NOECHO + IO$M_NOFILTR +
                                            IO$M_TRMNOECHO),
                                    P1 = CHAR_BUF,
                                    P2 = 1,
                                    P4 = .KCB [KCB_L_QIO1_P4]);
                                        ! P4 = defined terminator set
                IF NOT .QIO_RESULT THEN RET_STATUS = .QIO_RESULT;

                !+
                ! Move terminator characters to buffer.
                !-

                BUFFER [.BUFFER_LEN] = .CHAR_BUF;
                BUFFER_LEN = .BUFFER_LEN + 1;
                KCB [KCB_B_IOSB_TERMLEN] = .KCB [KCB_B_IOSB_TERMLEN] + 1;

                CASE .CHAR_BUF FROM %C'A' TO %C'~' OF
                SET
                    [%C'A' TO %C'D',        ! arrow keys
                     %C'M',                 ! Enter
                     %C'P' to %C'S',        ! PFn keys
                     %C'l' to %C'n',        ! keypad - , .
                     %C'p' to %C'y',        ! keypad 0-9
                     %C'~']:                ! function keys F6-F20 (no codes for
                                            ! F1-F5), editing keys (E1-E6)
                        BEGIN
                        END_ESC_FLAG = 1;
                        RET_STATUS = SS$_NORMAL;
                                            ! get rid of ss$_partescape error
                        END;

                    [INRANGE, OUTRANGE]:
                        ;                   ! didn't find end of esc terminator
                                            ! keep searching
                TES;

                END;                        ! end of WHILE loop
            END;                            ! end of partial ESC terminator

        !+
        ! Get the terminator code.
        !-

        IF .QIO_STATUS EQL SS$_TIMEOUT
```

```
	THEN
	    KEY_CODE = SMG$K_TRM_TIMEOUT
	ELSE
	    KEY_CODE  = SMG$$TERM_TO_KEYCODE (BUFFER [.KCB [KCB_W_IOSB_COUNT]],
		.KCB [KCB_B_IOSB_TERMLEN]);

	!+
	! If DISPLAY_ID was passed, call back output side to tell it
	! what changed on the screen as a result of this prompt & input.
	!-

	IF .DISPLAY_ID_FLAG
	THEN
	    BEGIN
	    LOCAL
		OUT_STATUS;
	    IF PROMPT_STRING [0,0,0,0] NEQA 0
	    THEN
		BEGIN				! report prompt string change
		OUT_STATUS = SMG$$REPORT_CHANGE_REPLACE (.DISPLAY_ID,
						KCB [KCB_L_PASTEBOARD_ID],
						%REF (.PROMPT_STRING [DSC$W_LENGTH]),
						.PROMPT_STRING [DSC$A_POINTER]);
		IF NOT .OUT_STATUS AND
		    .OUT_STATUS NEQ SMG$_STRTERESC
		THEN
		    RET_STATUS = .OUT_STATUS;
		END;				! report prompt string change

	    BEGIN
	    MAP
		MODIFIERS : REF BLOCK [,BYTE];
	    IF NULLPARAMETER (K_MODIFIERS) OR
	       (NOT NULLPARAMETER (K_MODIFIERS) AND
		NOT (.MODIFIERS [TRM$V_TM_NOECHO]))
	    THEN
		BEGIN				! skip if noecho requested
		OUT_STATUS = SMG$$REPORT_CHANGE_REPLACE (.DISPLAY_ID,
						KCB [KCB_L_PASTEBOARD_ID],
						%REF (.KCB [KCB_W_IOSB_COUNT] +
						    (IF (NOT NULLPARAMETER (K_MODIFIERS) AND
							NOT (.MODIFIERS [TRM$V_TM_TRMNOECHO]))
						    THEN .KCB [KCB_B_IOSB_TERMLEN]
						    ELSE 0)),
						BUFFER);
					! report user's input string,
					! possibly with terminator
		IF NOT .OUT_STATUS AND
		    .OUT_STATUS NEQ SMG$_STRTERESC
		THEN
		    RET_STATUS = .OUT_STATUS;
		END;
	    END;

	    END;

	!+
```

```
              ! Check for a terminator of ^Z.
              !-

              IF .KEY_CODE EQL SMG$K_TRM_CTRLZ
              THEN
                  BEGIN
                  !+
                  ! If count is zero, return SMG$_EOF now, else set
                  ! the flag so that it will be returned the next time.
                  !-

                  IF .KCB [KCB_W_IOSB_COUNT] EQL 0
                  THEN
                      RET_STATUS = SMG$_EOF
                  ELSE
                      KCB [KCB_V_CTRLZ] = 1;
                  END;

              !+
              ! If the caller asked for TERMINATOR_CODE, store it.
              !-

              IF NOT NULLPARAMETER (K_TERMINATOR_CODE)
              THEN
                  TERMINATOR_CODE [0] = .KEY_CODE;

              !+
              ! Set BUFFER_PTR and BUFFER_LEN.
              !-

              BUFFER_PTR = BUFFER;
              BUFFER_LEN = .KCB [KCB_W_IOSB_COUNT];

              !+
              ! Delete the prompt string, if any.
              !-

              IF NOT NULLPARAMETER (K_PROMPT_STRING) OR
                  .DISPLAY_ID_FLAG
              THEN
                  SMG$$DELETE_PROMPT_STRING (ITEM_LIST [K_PROMPT,0,0,0,0]);
              END;

          !+
          ! We now have the complete line in BUFFER, with length BUFFER_LEN.
          ! Copy it to the user's string.
          !-

          COPY_STATUS = LIB$SCOPY_R_DX6 (.BUFFER_LEN,
              .BUFFER_PTR, OUT_STRING [0,0,0,0]);
          IF NOT .COPY_STATUS
          THEN
              RETURN .COPY_STATUS;

          !+
          ! Fill in OUT_LENGTH if requested.
          !-
```

```
      2
      2        IF NOT NULLPARAMETER (K_OUT_LENGTH)
      2        THEN
      3            BEGIN
      3            LOCAL
      3                LINE_LENGTH: WORD;
      3            LIB$ANALYZE_SDESC_R2 (OUT_STRING [0,0,0,0]; LINE_LENGTH);
      3            IF .LINE_LENGTH LSSU .BUFFER_LEN
      3            THEN
      3                BUFFER_LEN = .LINE_LENGTH;
      3            OUT_LENGTH [0] = .BUFFER_LEN;
      2            END;
      2
      2        RETURN .RET_STATUS;
      2
      1        END;                                           ! End of routine SMG$READ_STRING


                                00341          .BLKB    3
              00  00  00  20    00344 P.AAC:   .ASCII   \ \<0><0><0>                                        ;

                                               .EXTRN   SMG$_INVKBD_ID, SYS$GET
                                               .EXTRN   SMG$_INVPAS_ID, PBD_L_COUNT
                                               .EXTRN   PBD_A_PBCB, PBD_V_PB_AVAIL

                          0FFC  00000          .ENTRY   SMG$READ_STRING, Save R2,R3,R4,R5,R6,R7,R8,-; 0824
                                                        R9,R10,R11                                  ;
        5B 00000000G    00  9E  00002          MOVAB    SYS$QIOW, R11                               ;
        5A 00000000G    8F  D0  00009          MOVL     #SMG$_EOF, R10                              ;
        5E      FDA4    CE  9E  00010          MOVAB    -604(SP), SP                                ;
                        56  D4  00015          CLRL     DISPLAY_ID_FLAG                             ; 0937
  50    6C              02  83  00017          SUBB3    #2, (AP), DIFF                              ; 0989
        08              50  91  0001B          CMPB     DIFF, #8                                    ;
                        08  1B  0001E          BLEQU    1$                                          ;
        50 00000000G    8F  D0  00020          MOVL     #SMG$_WRONUMARG, R0                         ;
                            04  00027          RET                                                  ;
        52        04    BC  D0  00028 1$:      MOVL     @KEYBOARD_ID, KCB                           ; 0995
                        06  13  0002C          BEQL     2$                                          ;
        52        DC    A2  D1  0002E          CMPL     -36(KCB), KCB                               ;
                        08  13  00032          BEQL     3$                                          ;
        50 00000000G    8F  D0  00034 2$:      MOVL     #SMG$_INVKBD_ID, R0                         ;
                            04  0003B          RET                                                  ;
        57              01  D0  0003C 3$:      MOVL     #1, RET_STATUS                              ; 1001
        36        FC    A2  E9  0003F          BLBC     -4(KCB), 7$                                 ; 1007
                        52  DD  00043          PUSHL    KCB                                         ; 1018
   00000000G  00        01  FB  00045          CALLS    #1, SYS$GET                                 ;
              11        50  E8  0004C          BLBS     GET_STATUS, 5$                              ; 1019
   0001827A   8F        50  D1  0004F          CMPL     GET_STATUS, #98938                          ; 1022
                        05  12  00056          BNEQ     4$                                          ;
              57        5A  D0  00058          MOVL     R10, RET_STATUS                             ; 1024
                        03  11  0005B          BRB      5$                                          ;
              57        50  D0  0005D 4$:      MOVL     GET_STATUS, RET_STATUS                      ; 1026
              09        6C  91  00060 5$:      CMPB     (AP), #9                                    ; 1033
                        09  1F  00063          BLSSU    6$                                          ;
                  24    AC  D5  00065          TSTL     36(AP)                                      ;
                        04  13  00068          BEQL     6$                                          ;
```

```
          24   BC           0D  B0 0006A        MOVW    #13, @TERMINATOR_CODE              ; 1035
               59      28   A2  D0 0006E 6$:    MOVL    40(KCB), BUFFER_PTR                ; 1041
               58      22   A2  B0 00072        MOVW    34(KCB), BUFFER_LEN                ; 1042
                          03A6  31 00076        BRW     56$                                ; 1007
                       08   AE  D4 00079 7$:    CLRL    CURSOR_BUF_LEN                     ; 1051
   04     FC   A2           01  E5 0007C        BBCC    #1, -4(KCB), 8$                    ; 1061
               50           5A  D0 00081        MOVL    R10, R0                            ; 1063
                                04 00084        RET
               50      28   AE  9E 00085 8$:    MOVAB   ITEM_LIST, $$ITMBLKPTR             ; 1071
                            80  D4 00089        CLRL    ($$ITMBLKPTR)+
               80    4000   8F  3C 0008B        MOVZWL  #16384, ($$ITMBLKPTR)+
                            80  D4 00090        CLRL    ($$ITMBLKPTR)+
               80 00040000  8F  D0 00092        MOVL    #262144, ($$ITMBLKPTR)+
                            80  7C 00099        CLRQ    ($$ITMBLKPTR)+
                            80  D4 0009B        CLRL    ($$ITMBLKPTR)+
          0C   A2    8031   8F  3C 0009D        MOVZWL  #32817, 12(KCB)                    ; 1077
          1C   A2      5C   AE  9E 000A3        MOVAB   BUFFER, 28(KCB)                    ; 1078
               54      20   A2  9E 000A8        MOVAB   32(KCB), R4                        ; 1079
               64    0200   8F  3C 000AC        MOVZWL  #512, (R4)
                       24   A2  7C 000B1        CLRQ    36(KCB)                            ; 1080
          2C   A2      28   AE  9E 000B4        MOVAB   ITEM_LIST, 44(KCB)                 ; 1082
          30   A2           18  D0 000B9        MOVL    #24, 48(KCB)                       ; 1083
               04           6C  91 000BD        CMPB    (AP), #4                           ; 1093
                            03  1E 000C0        BGEQU   9$
                          00A4  31 000C2        BRW     16$
               53      40   AE  9E 000C5 9$:    MOVAB   ITEM_LIST+24, CUR_ITEM             ; 1098
                       10   AC  D5 000C9        TSTL    MAX_LENGTH                         ; 1100
                            15  13 000CC        BEQL    10$
               64      10   BC  D0 000CE        MOVL    @MAX_LENGTH, (R4)                  ; 1103
      00000200 8F           64  D1 000D2        CMPL    (R4), #512                         ; 1104
                            08  1B 000D9        BLEQU   10$
               50 00000000G 8F  D0 000DB        MOVL    #SMG$_INVMAXLEN, R0                ; 1106
                                04 000E2        RET
               05           6C  91 000E3 10$:   CMPB    (AP), #5                           ; 1108
                            7B  1F 000E6        BLSSU   15$
                       14   AC  D5 000E8        TSTL    MODIFIERS                          ; 1111
                            05  13 000EB        BEQL    11$
          2C   AE      14   BC  C8 000ED        BISL2   @MODIFIERS, ITEM_LIST+4            ; 1115
               06           6C  91 000F2 11$:   CMPB    (AP), #6                           ; 1116
                            6C  1F 000F5        BLSSU   15$
                       18   AC  D5 000F7        TSTL    TIMEOUT                            ; 1119
                            0D  13 000FA        BEQL    12$
               83 00020000  8F  D0 000FC        MOVL    #131072, (CUR_ITEM)+               ; 1122
               83      18   BC  D0 00103        MOVL    @TIMEOUT, (CUR_ITEM)+              ; 1124
                            83  D4 00107        CLRL    (CUR_ITEM)+                        ; 1125
               07           6C  91 00109 12$:   CMPB    (AP), #7                           ; 1127
                            55  1F 0010C        BLSSU   15$
               50      1C   AC  D0 0010E        MOVL    TERMINATOR_SET, R0                 ; 1130
                            0C  13 00112        BEQL    13$
               83           60  B0 00114        MOVW    (R0), (CUR_ITEM)+                  ; 1133
               83           03  B0 00117        MOVW    #3, (CUR_ITEM)+                    ; 1134
               83      04   A0  D0 0011A        MOVL    4(R0), (CUR_ITEM)+                 ; 1135
                            83  D4 0011E        CLRL    (CUR_ITEM)+                        ; 1136
               0A           6C  91 00120 13$:   CMPB    (AP), #10                          ; 1138
                            3E  1F 00123        BLSSU   15$
                       28   AC  D5 00125        TSTL    40(AP)                             ;
                            39  13 00128        BEQL    15$                                ;
```

```
                          56           01  D0 0012A             MOVL    #1, DISPLAY_ID_FLAG                        ; 1152
                                 F8    A2  D5 0012D             TSTL    -8(KCB)                                    ; 1153
                                       18  18 00130             BGEQ    14$
                                 F8    A2  9F 00132             PUSHAB  -8(KCB)                                    ; 1158
                                 72    A2  9F 00135             PUSHAB  114(KCB)                                   ; 1157
                     08    AE    70    A2  32 00138             CVTWL   112(KCB), 8(SP)                            ; 1156
                                 08    AE  9F 0013D             PUSHAB  8(SP)
             00000000G     00          03  FB 00140             CALLS   #3, SMG$$GET_PASTEBOARD_ID                 ; 1158
                           5D          50  E9 00147             BLBC    OUT_STATUS, 21$                            ; 1159
                                 04    AE  9F 0014A  14$:       PUSHAB  FREE_COLS                                  ; 1162
                                 0C    AE  9F 0014D             PUSHAB  CURSOR_BUF_LEN
                                 20    AE  9F 00150             PUSHAB  CURSOR_BUF
                                 F8    A2  9F 00153             PUSHAB  -8(KCB)
                                 28    AC  DD 00156             PUSHL   DISPLAY_ID
             00000000G     00          05  FB 00159             CALLS   #5, SMG$$SET_PHYSICAL_CURSOR
                           44          50  E9 00160             BLBC    OUT_STATUS, 21$                            ; 1165
    30   A2                53    2C    A2  C3 00163  15$:       SUBL3   44(KCB), CUR_ITEM, 48(KCB)                 ; 1176
                     10    AE 010E0000 8F  D0 00169  16$:       MOVL    #17694720, NULL_STRING                     ; 1189
                     14    AE  FE87    CF  9E 00171             MOVAB   P.AAC, NULL_STRING+4                       ; 1190
                           0A          56  E8 00177             BLBS    DISPLAY_ID_FLAG, 17$                       ; 1192
                           03          6C  91 0017A             CMPB    (AP), #3                                   ; 1193
                                       2C  1F 0017D             BLSSU   22$
                                 0C    AC  D5 0017F             TSTL    12(AP)
                                       27  13 00182             BEQL    22$
                                 18    AE  9F 00184  17$:       PUSHAB  CURSOR_BUF                                 ; 1202
                                 0C    AE  DD 00187             PUSHL   CURSOR_BUF_LEN                             ; 1203
                                 3C    AE  9F 0018A             PUSHAB  ITEM_LIST+12                               ; 1202
                           03          6C  91 0018D             CMPB    (AP), #3                                   ; 1197
                                       05  1F 00190             BLSSU   18$
                                 0C    AC  D5 00192             TSTL    12(AP)
                                       08  12 00195             BNEQ    19$
                           50    1C    AE  9E 00197  18$:       MOVAB   NULL_STRING, R0
                                       50  DD 0019B             PUSHL   R0
                                       03  11 0019D             BRB     20$
                                 0C    AC  DD 0019F  19$:       PUSHL   PROMPT_STRING                              ; 1201
                 0000V     CF          04  FB 001A2  20$:       CALLS   #4, SMG$$SET_PROMPT_STRING                 ; 1202
                           01          50  E8 001A7  21$:       BLBS    STATUS, 22$                                ; 1205
                                       04 001AA                 RET
                           23          56  E9 001AB  22$:       BLBC    DISPLAY_ID_FLAG, 25$                       ; 1209
                                 0C    AC  D5 001AE             TSTL    PROMPT_STRING                              ; 1212
                                       08  13 001B1             BEQL    23$
                           50    0C    BC  3C 001B3             MOVZWL  @PROMPT_STRING, R0                         ; 1214
                     04    AE          50  C2 001B7             SUBL2   R0, FREE_COLS
                           50    04    AE  D0 001BB  23$:       MOVL    FREE_COLS, R0                              ; 1215
                                       08  14 001BF             BGTR    24$
                           50 00000000G 8F D0 001C1             MOVL    #SMG$_INVCOL, R0                           ; 1220
                                       04 001C8                 RET
                           50          64  D1 001C9  24$:       CMPL    (R4), R0                                   ; 1222
                                       03  15 001CC             BLEQ    25$
                           64          50  D0 001CE             MOVL    R0, (R4)                                   ; 1224
                           5E          56  E9 001D1  25$:       BLBC    DISPLAY_ID_FLAG, 31$                       ; 1277
                           50    28    BC  D0 001D4             MOVL    @DISPLAY_ID, R0                            ; 1286
                     28    BC    38    A0  D1 001D8             CMPL    56(R0), @DISPLAY_ID
                                       06  12 001DD             BNEQ    26$
                           11    44    A0  91 001DF             CMPB    68(R0), #17
                                       08  13 001E3             BEQL    27$
                           50 00000000G 8F D0 001E5  26$:       MOVL    #SMG$_INVDIS_ID, R0                        ;
```

```
                                              04 001EC          RET                                                        ;
                      53          28     BC   D0 001ED 27$:     MOVL    @DISPLAY_ID, DCB                                   ;
                      50          F8     A2   D0 001F1          MOVL    -8(KCB), R0                                        ; 1287
                                         11   19 001F5          BLSS    28$                                                ;
            00000000G 00                 50   D1 001F7          CMPL    R0, PBD_L_COUNT                                    ;
                                         08   14 001FE          BGTR    28$                                                ;
         08 00000000G 00                 50   E0 00200          BBS     R0, PBD_V_PB_AVAIL, 29$                            ;
                      50 00000000G       8F   D0 00208 28$:     MOVL    #SMG$_INVPAS_ID, R0                                ;
                                              04 0020F          RET                                                        ;
                      55 00000000G0040        D0 00210 29$:     MOVL    PBD_A_PBCB[R0], PBCB                               ;
                                  1C     A3   D5 00218          TSTL    28(DCB)                                            ; 1293
                                         08   13 0021B          BEQL    30$                                                ;
                      50 00000000G       8F   D0 0021D          MOVL    #SMG$_ILLBATFNC, R0                                ; 1295
                                              04 00224          RET                                                        ;
                      7E          2E     A3   9A 00225 30$:     MOVZBL  46(DCB), -(SP)                                     ; 1301
                                         55   DD 00229          PUSHL   PBCB                                               ;
            00000000G 00                 02   FB 0022B          CALLS   #2, SMG$$SET_ATTRIBUTES_ON                         ;
                      6B                 62   FA 00232 31$:     CALLG   (KCB), SYS$QIOW                                    ; 1309
                      54                 50   D0 00235          MOVL    R0, QIO_STATUS                                     ;
                      04                 54   E9 00238          BLBC    QIO_STATUS, 32$                                    ; 1310
                      54          68     A2   32 0023B          CVTWL   104(KCB), QIO_STATUS                               ; 1312
                      0D                 56   E9 0023F 32$:     BLBC    DISPLAY_ID_FLAG, 33$                               ; 1318
                      7E          2E     A3   9A 00242          MOVZBL  46(DCB), -(SP)                                     ; 1320
                                         55   DD 00246          PUSHL   PBCB                                               ;
            00000000G 00                 02   FB 00248          CALLS   #2, SMG$$SET_ATTRIBUTES_OFF                        ;
                      03                 54   E8 0024F 33$:     BLBS    QIO_STATUS, 34$                                    ; 1322
                      57                 54   D0 00252          MOVL    QIO_STATUS, RET_STATUS                             ; 1324
                 01FC 8F          68     A2   B1 00255 34$:     CMPW    104(KCB), #508                                     ; 1333
                                         0D   12 0025B          BNEQ    36$                                                ;
                                         53   D4 0025D          CLRL    END_ESC_FLAG                                       ; 1335
                      50          6E     A2   9A 0025F          MOVZBL  110(KCB), R0                                       ; 1341
         58       6A  A2                 50   A1 00263          ADDW3   R0, 106(KCB), BUFFER_LEN                           ;
                                         53   D5 00268 35$:     TSTL    END_ESC_FLAG                                       ; 1344
                                         03   13 0026A 36$:     BEQL    37$                                                ;
                                       00BF   31 0026C          BRW     42$                                                ;
                                         7E   7C 0026F 37$:     CLRQ    -(SP)                                              ; 1353
                                  28     A2   DD 00271          PUSHL   40(KCB)                                            ;
                      7E                 01   7D 00274          MOVQ    #1, -(SP)                                          ;
                                  20     AE   9F 00277          PUSHAB  CHAR_BUF                                           ;
                                         7E   7C 0027A          CLRQ    -(SP)                                              ;
                                         7E   D4 0027C          CLRL    -(SP)                                              ;
                      7E        5271     8F   3C 0027E          MOVZWL  #21105, -(SP)                                      ;
                      7E          D4     A2   32 00283          CVTWL   -44(KCB), -(SP)                                    ;
                                  D0     A2   DD 00287          PUSHL   -48(KCB)                                           ;
                      6B                 0C   FB 0028A          CALLS   #12, SYS$QIOW                                      ;
                      03                 50   E8 0028D          BLBS    QIO_RESULT, 38$                                    ; 1355
                      57                 50   D0 00290          MOVL    QIO_RESULT, RET_STATUS                             ;
                      51                 58   3C 00293 38$:     MOVZWL  BUFFER_LEN, R1                                     ; 1361
              5C AE41             0C     AE   90 00296          MOVB    CHAR_BUF, BUFFER[R1]                               ;
                                         58   B6 0029C          INCW    BUFFER_LEN                                         ; 1362
                                  6E     A2   96 0029E          INCB    110(KCB)                                           ; 1363
            3D    41  8F          0C     AE   8F 002A1          CASEB   CHAR_BUF, #65, #61                                 ; 1365
 007E         007E        007E         007E      002A7 39$:     .WORD   40$-39$,-                                          ;
 FFC1         FFC1        FFC1         FFC1      002AF                  40$-39$,-                                          ;
 FFC1         FFC1        FFC1         FFC1      002B7                  40$-39$,-                                          ;
 007E         FFC1        FFC1         007E      002BF                  40$-39$,-                                          ;
 FFC1         007E        007E         007E      002C7                  35$-39$,-                                          ;
```

```
FFC1    FFC1    FFC1    FFC1    002CF        35$-39$,-
FFC1    FFC1    FFC1    FFC1    002D7        35$-39$,-
FFC1    FFC1    FFC1    FFC1    002DF        35$-39$,-
FFC1    FFC1    FFC1    FFC1    002E7        35$-39$,-
FFC1    FFC1    FFC1    FFC1    002EF        35$-39$,-
007E    FFC1    FFC1    FFC1    002F7        35$-39$,-
007E    FFC1    007E    007E    002FF        35$-39$,-
007E    007E    007E    007E    00307        40$-39$,-
007E    007E    007E    007E    0030F        35$-39$,-
FFC1    FFC1    FFC1    007E    00317        35$-39$,-
                007E    FFC1    0031F        40$-39$,-
                                             40$-39$,-
                                             40$-39$,-
                                             40$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             40$-39$,-
                                             40$-39$,-
                                             40$-39$,-
                                             35$-39$,-
                                             40$-39$,-
                                             40$-39$,-
                                             40$-39$,-
                                             40$-39$,-
                                             40$-39$,-
                                             40$-39$,-
                                             40$-39$,-
                                             40$-39$,-
                                             40$-39$,-
                                             40$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             35$-39$,-
                                             40$-39$
```

```
                                   06  11 00323          BRB     41$                                        ;
                    53             01  D0 00325 40$:     MOVL    #1, END_ESC_FLAG                           ; 1375
                    57             01  D0 00328          MOVL    #1, RET_STATUS                             ; 1376
                                 FF3A  31 0032B 41$:     BRW     35$                                        ; 1344
          0000022C  8F             54  D1 0032E 42$:     CMPL    QIO_STATUS, #556                           ; 1392
                                   07  12 00335          BNEQ    43$                                        ;
                    53       01FD  8F  B0 00337          MOVW    #509, KEY_CODE                             ; 1394
                                   16  11 0033C          BRB     44$                                        ;
                    7E         6E  A2  9A 0033E 43$:     MOVZBL  110(KCB), -(SP)                            ; 1397
                    50         6A  A2  3C 00342          MOVZWL  106(KCB), R0                               ; 1396
                            60 AE40    9F 00346          PUSHAB  BUFFER[R0]                                 ;
          00000000G 00             02  FB 0034A          CALLS   #2, SMG$$TERM_TO_KEYCODE                   ;
                    53             50  B0 00351          MOVW    R0, KEY_CODE                               ;
                    03             56  E8 00354 44$:     BLBS    DISPLAY_ID_FLAG, 45$                       ; 1404
                                 0087  31 00357          BRW     50$                                        ;
                    50         0C  AC  D0 0035A 45$:     MOVL    PROMPT_STRING, R0                          ; 1409
                                   26  13 0035E          BEQL    46$                                        ;
                               04  A0  DD 00360          PUSHL   4(R0)                                      ; 1415
                 04 AE             60  3C 00363          MOVZWL  (R0), 4(SP)                                ; 1414
                               04  AE  9F 00367          PUSHAB  4(SP)                                      ;
                               F8  A2  9F 0036A          PUSHAB  -8(KCB)                                    ; 1413
                               28  AC  DD 0036D          PUSHL   DISPLAY_ID                                 ;
          00000000G 00             04  FB 00370          CALLS   #4, SMG$$REPORT_CHANGE_REPLACE             ;
                    0C             50  E8 00377          BLBS    OUT_STATUS, 46$                            ; 1416
          00000000G 8F             50  D1 0037A          CMPL    OUT_STATUS, #SMG$_STRTERESC                ; 1417
                                   03  13 00381          BEQL    46$                                        ;
                    57             50  D0 00383          MOVL    OUT_STATUS, RET_STATUS                     ; 1419
                    05             6C  91 00386 46$:     CMPB    (AP), #5                                   ; 1425
                                   14  1F 00389          BLSSU   47$                                        ;
                               14  AC  D5 0038B          TSTL    20(AP)                                     ;
                                   0F  13 0038E          BEQL    47$                                        ;
                    05             6C  91 00390          CMPB    (AP), #5                                   ; 1426
                                   4C  1F 00393          BLSSU   50$                                        ;
                               14  AC  D5 00395          TSTL    20(AP)                                     ;
                                   47  13 00398          BEQL    50$                                        ;
            42      14  BC         06  E0 0039A          BBS     #6, @MODIFIERS, 50$                        ; 1427
                               5C  AE  9F 0039F 47$:     PUSHAB  BUFFER                                     ; 1431
                    05             6C  91 003A2          CMPB    (AP), #5                                   ; 1433
                                   10  1F 003A5          BLSSU   48$                                        ;
                               14  AC  D5 003A7          TSTL    20(AP)                                     ;
                                   0B  13 003AA          BEQL    48$                                        ;
            06      14  BC         0C  E0 003AC          BBS     #12, @MODIFIERS, 48$                       ; 1434
                    51         6E  A2  9A 003B1          MOVZBL  110(KCB), R1                               ; 1435
                                   02  11 003B5          BRB     49$                                        ;
                                   51  D4 003B7 48$:     CLRL    R1                                         ; 1433
                    54         6A  A2  3C 003B9 49$:     MOVZWL  106(KCB), R4                               ;
         04 AE      51             54  C1 003BD          ADDL3   R4, R1, 4(SP)                              ;
                               04  AE  9F 003C2          PUSHAB  4(SP)                                      ; 1432
                               F8  A2  9F 003C5          PUSHAB  -8(KCB)                                    ; 1431
                               28  AC  DD 003C8          PUSHL   DISPLAY_ID                                 ;
          00000000G 00             04  FB 003CB          CALLS   #4, SMG$$REPORT_CHANGE_REPLACE             ;
                    0C             50  E8 003D2          BLBS    OUT_STATUS, 50$                            ; 1440
          00000000G 8F             50  D1 003D5          CMPL    OUT_STATUS, #SMG$_STRTERESC                ; 1441
                                   03  13 003DC          BEQL    50$                                        ;
                    57             50  D0 003DE          MOVL    OUT_STATUS, RET_STATUS                     ; 1443
                    1A             53  B1 003E1 50$:     CMPW    KEY_CODE, #26                              ; 1453
                                   0E  12 003E4          BNEQ    52$                                        ;
```

```
                6A   A2  B5 003E6        TSTW    106(KCB)                              ; 1461
                     05  12 003E9        BNEQ    51$                                   ;
          57         5A  D0 003EB        MOVL    R10, RET_STATUS                       ; 1463
                     04  11 003EE        BRB     52$                                   ;
       FC A2         02  88 003F0 51$:   BISB2   #2, -4(KCB)                           ; 1465
          09         6C  91 003F4 52$:   CMPB    (AP), #9                              ; 1472
                     09  1F 003F7        BLSSU   53$                                   ;
                24   AC  D5 003F9        TSTL    36(AP)                                ;
                     04  13 003FC        BEQL    53$                                   ;
       24 BC         53  B0 003FE        MOVW    KEY_CODE, @TERMINATOR_CODE            ; 1474
          59    5C   AE  9E 00402 53$:   MOVAB   BUFFER, BUFFER_PTR                    ; 1480
          58    6A   A2  B0 00406        MOVW    106(KCB), BUFFER_LEN                  ; 1481
          03         6C  91 0040A        CMPB    (AP), #3                              ; 1487
                     05  1F 0040D        BLSSU   54$                                   ;
                0C   AC  D5 0040F        TSTL    12(AP)                                ;
                     03  12 00412        BNEQ    55$                                   ;
          08         56  E9 00414 54$:   BLBC    DISPLAY_ID_FLAG, 56$                  ; 1488
                34   AE  9F 00417 55$:   PUSHAB  ITEM_LIST+12                          ; 1490
    0000V CF         01  FB 0041A        CALLS   #1, SMG$$DELETE_PROMPT_STRING         ;
          52    08   AC  D0 0041F 56$:   MOVL    OUT_STRING, R2                        ; 1499
          51         59  D0 00423        MOVL    BUFFER_PTR, R1                        ;
          50         58  3C 00426        MOVZWL  BUFFER_LEN, R0                        ;
        00000000G    00  16 00429        JSB     LIB$SCOPY_R_DX6                       ;
          26         50  E9 0042F        BLBC    COPY_STATUS, 59$                      ; 1500
          08         6C  91 00432        CMPB    (AP), #8                              ; 1508
                     1E  1F 00435        BLSSU   58$                                   ;
                20   AC  D5 00437        TSTL    32(AP)                                ;
                     19  13 0043A        BEQL    58$                                   ;
          50    08   AC  D0 0043C        MOVL    OUT_STRING, R0                        ; 1513
        00000000G    00  16 00440        JSB     LIB$ANALYZE_SDESC_R2                  ;
          50         51  D0 00446        MOVL    R1, LINE_LENGTH                       ;
          58         50  B1 00449        CMPW    LINE_LENGTH, BUFFER_LEN               ; 1514
                     03  1E 0044C        BGEQU   57$                                   ;
          58         50  B0 0044E        MOVW    LINE_LENGTH, BUFFER_LEN               ; 1516
       20 BC         58  B0 00451 57$:   MOVW    BUFFER_LEN, @OUT_LENGTH               ; 1517
          50         57  D0 00455 58$:   MOVL    RET_STATUS, R0                        ; 1520
                     04 00458 59$:       RET                                           ; 1522
```

; Routine Size: 1113 bytes, Routine Base: _SMG$CODE + 0348

```
%SBTTL 'SMG$READ_COMPOSED_LINE'
GLOBAL ROUTINE SMG$READ_COMPOSED_LINE (
    KEYBOARD_ID: REF VECTOR [, LONG],
    KEY_TABLE_ID: REF VECTOR [, LONG],
    OUT_LINE: REF BLOCK [, BYTE],
    PROMPT_STRING: REF BLOCK [, BYTE],
    OUT_LENGTH: REF VECTOR [, WORD],
    DISPLAY_ID: REF VECTOR [, LONG]
    ) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       SMG$READ_COMPOSED_LINE reads a line composed of normal keystrokes
!       and key equivalence strings.  As keys are pressed which are
!       defined in the key-table, the equivalence string, if any, is
!       substituted in the input line for the key.  Attributes of
!       the key definition control whether the equivalence string
!       is echoed and whether the read terminates.
!
!       The manner of the read operation is not affected by previous calls
!       to other Screen Management procedures, unlike SMG$READ_STRING.
!       Characters typed are always echoed (in normal rendition).
!       <CR> always terminates the read operation.
!       If <CTRL/Z> is pressed and there is no definition
!       in the key-table for CTRLZ, an appropriate string is echoed, the read terminates.
!       If the <CTRL/Z> was the first character in the line, then the
!       status SMG$_EOF is returned, else SMG$_EOF is returned on the next
!       input operation.  SMG$_EOF is also returned if RMS is used for input
!       and it returns RMS$_EOF. No other terminators are recognized except
!       as specified by attributes in the key definitions.
!
! CALLING SEQUENCE:
!
!       RET_STATUS.wlc.v = SMG$READ_COMPOSED_LINE (
!                               KEYBOARD_ID.rl.r,
!                               KEY_TABLE_ID.rl.r,
!                               OUT_LINE.wt.dx
!                               [, [PROMPT_STRING.rt.dx]
!                               [, [OUT_LENGTH.wwu.r]
!                               [, [DISPLAY_ID.rl.r] ]]])
!
! FORMAL PARAMETERS:
!
!       KEYBOARD_ID     A longword containing the identification of the
!                       virtual-keyboard from which to read.  The virtual-
!                       keyboard is created by SMG$CREATE_VIRTUAL_KEYBOARD.
!
!       KEY_TABLE_ID    A longword containing the identification of the
!                       key-definition-table to be used for translating
!                       keystrokes.  The key-definition table-table is
!                       created by SMG$CREATE_KEY_TABLE.
!
!       OUT_LINE        A string into which is written the complete composed
!                       line.
!
!       PROMPT_STRING   An optional string to be used as the prompt for the
```

```
1 !                             read operation.
1 !
1 !         OUT_LENGTH          An unsigned word into which is written the number of
1 !                             characters read or the maximum length of OUT_LINE,
1 !                             whichever is less.
1 !
1 !         DISPLAY_ID          A longword containing the identification of the
1 !                             virtual-display in which the read is to be performed.
1 !                             If specified, the read begins at the beginning of the
1 !                             next line after the current cursor position in that
1 !                             virtual-display.  If omitted, the read begins at the
1 !                             beginning of the next line after the current cursor
1 !                             position.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !         NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !         NONE
1 !
1 ! COMPLETION STATUS:
1 !
1 !         SS$_NORMAL          Normal successful completion
1 !         SS$_CANCEL          I/O operation cancelled (by SMG$CANCEL_INPUT)
1 !         SMG$_EOF            End-of-file
1 !         SMG$_INVDIS_ID      Invalid display-id
1 !         SMG$_INVKBD_ID      Invalid keyboard-id
1 !         SMG$_INVKTB_ID      Invalid key-table-id
1 !         SMG$_WRONUMARG      Wrong number of arguments
1 !         LIB$_xxx            Any error from LIB$SCOPY_R_DX
1 !         RMS$_xxx            Any error from $GET (except RMS$_EOF)
1 !         SS$_xxx             Any error from $QIOW
1 !
1 ! SIDE EFFECTS:
1 !
1 !         NONE
1 !
1 !--
1
2       BEGIN
2
2       FIELD
2           FLAGS_FIELDS =
2               SET
2               V_RESET_STATE = [0,0,1,0],
2               V_SKIP_NEWLINE = [0,1,1,0],
2               V_SKIP_LAST_ECHO = [0,2,1,0],
2               V_SETPROMPT = [0,3,1,0],
2               V_ECHO_CTRLZ = [0,4,1,0]
2               TES;
2
2       LOCAL
2           KCB: REF KCB_R_KCB_STRUCT,
2           KDE: REF KDE_R_KDE_STRUCT,
2           KTH: REF KTH_R_KTH_STRUCT,
```

```
        BUFFER: VECTOR [516, BYTE], ! Allow 4 chars for terminator echo
        BUFFER_PTR,
        BUFFER_LEN: WORD,
        NEXT_POS,
        TKEY: VECTOR [34, BYTE],
        TKEY_POS: REF VECTOR [, WORD],
        TKEY_DSC: BLOCK [8, BYTE],
        TKEY_DSC_PTR: REF BLOCK [8, BYTE],
        KEY_CODE: WORD,
        ITEM_LIST: $ITMLST_DECL (ITEMS=5),
        FLAGS: BLOCK [1, BYTE] FIELD (FLAGS_FIELDS),
        TERM_NAME_PTR: REF VECTOR [, BYTE],
        DISPLAY_ID_FLAG : INITIAL(0),
        QIO_STATUS,
        COPY_STATUS,
        RET_STATUS;

    LITERAL
        K_CR = 13,          ! <CR>
        K_LF = 10,          ! <LF>
        K_CTRLZ = 26,       ! ^Z
        K_ESC = 27;         ! <ESC>

    BIND
        T_CR = UPLIT BYTE (K_CR);

    BUILTIN
        ACTUALCOUNT,
        CALLG,
        NULLPARAMETER,
        TESTBITSC,
        TESTBITSS;

    !+
    ! Define literals for the argument positions.
    !-

    LITERAL
        K_KEYBOARD_ID = 1,
        K_KEY_TABLE_ID = 2,
        K_OUT_LINE = 3,
        K_PROMPT_STRING = 4,
        K_OUT_LENGTH = 5,
        K_DISPLAY_ID = 6;

    !+
    ! Define literals for the item list positions.
    !-

    LITERAL
        K_MODIFIERS = 0,
        K_PROMPT = 1,
        K_INISTRNG = 2,
        K_INIOFFSET = 3,
        K_TERM = 4;

    !+
```

```
        ! Define the default strings which echo for control-Z.
        !-

        OWN
            T_CTRLZ_DECCRT: VECTOR [17, BYTE] PSECT (_SMG$CODE) INITIAL (
                BYTE (
                    K_ESC,'7',                  ! DECSC - Save Cursor
                    K_ESC,'[7m',                ! SGR - Negative image
                    ' Exit ',                   ! " Exit "
                    K_ESC,'[m',                 ! Normal rendition (in case resore fails)
                    K_ESC,'8')),                ! DECRC - Restore Cursor
            T_CTRLZ_OTHER: VECTOR [6, BYTE] PSECT (_SMG$CODE) INITIAL (
                BYTE (
                    '*EXIT*'));                 ! "*EXIT*"

        $SMG$VALIDATE_ARGCOUNT(3,6);

        !+
        ! Fetch and validate KCB.
        !-

        $SMG$VALIDATE_KCB (KEYBOARD_ID, KCB);

        !+
        ! Fetch and validate KTH.
        !-

        $SMG$VALIDATE_KTH (KEY_TABLE_ID, KTH);

        !+
        ! Assume success.
        !-

        RET_STATUS = SS$_NORMAL;

        !+
        ! Split depending on whether or not we are using RMS.
        !-

        IF .KCB [KCB_V_RMS]
        THEN

            !+
            ! Read a line using $GET.
            !-

            BEGIN
            LOCAL
                GET_STATUS;

            GET_STATUS = $GET (RAB = KCB [KCB_R_RAB]);
            IF NOT .GET_STATUS
            THEN
                BEGIN
                IF .GET_STATUS EQLU RMS$_EOF
                THEN
                    RET_STATUS = SMG$_EOF
```

```
                ELSE
                    RET_STATUS = .GET_STATUS;
                END;

            !+
            ! Set BUFFER_PTR and BUFFER_LEN.
            !-

            BUFFER_PTR = .KCB [RAB$L_RBF];
            BUFFER_LEN = .KCB [RAB$W_RSZ];
            END

        ELSE

            !+
            ! Use $QIO and do all the keypad substitution stuff.
            !-

            BEGIN

            LOCAL
                CURSOR_BUF : VECTOR [15, BYTE],     ! buffer for set cursor sequence
                CURSOR_BUF_LEN : INITIAL (0),       ! used number of bytes in cursor_buf
                FREE_COLS;                          ! calc. needed if display_id used

            !+
            ! If the last read ended in ^Z, return SMG$_EOF.
            !-

            IF TESTBITSC (KCB [KCB_V_CTRLZ])
            THEN
                RETURN SMG$_EOF;    ! End of file

            !+
            ! Initialize flags.
            !-

            FLAGS = 0;
            FLAGS [V_RESET_STATE] = 1;

            !+
            ! Build the item list for the QIO.
            !-

P           $ITMLST_INIT (ITMLST=ITEM_LIST,
P               (ITMCOD=TRM$_MODIFIERS,
P                   BUFADR=(TRM$M_TM_ESCAPE+TRM$M_TM_TRMNOECHO) OR
P                       .KTH [KTH_L_MODIFIERS], BUFSIZ=0),
P               (ITMCOD=TRM$_PROMPT, BUFADR=T_CR, BUFSIZ=1),
P               (ITMCOD=TRM$_INISTRNG, BUFADR=BUFFER, BUFSIZ=0),
P               (ITMCOD=TRM$_INIOFFSET, BUFADR=0, BUFSIZ=0),
P               (ITMCOD=TRM$_TERM, BUFSIZ=0,
                    BUFADR=.KTH [KTH_L_TERM_MASK]));

            IF NOT NULLPARAMETER (K_DISPLAY_ID)
            THEN
                BEGIN                               ! display_id present
```

```
          !+
          ! Interfacing with output's virtual displays.  Make sure user can
          ! not type beyond the display's boundaries by limiting the input
          ! size to the number of free columns (allowing room for prompt, if
          ! any).
          ! Notice that we need some info from SMG$$SET_PROMPT_STRING (prompt
          ! string length), and we need to provide it with some info (set cursor
          ! sequence, if any).  So get the info here and save it.
          !+
          LOCAL
              OUT_STATUS;
          DISPLAY_ID_FLAG = 1;          ! for quick checks later
          IF .KCB [KCB_L_PASTEBOARD_ID] LSS 0 ! pb not established yet
          THEN
              BEGIN
              OUT_STATUS = SMG$$GET_PASTEBOARD_ID (%REF (.KCB [KCB_W_DEVNAM_LENGTH]),
                                                   KCB [KCB_T_DEVNAM_STRING],
                                                   KCB [KCB_L_PASTEBOARD_ID]);
              IF NOT .OUT_STATUS THEN RETURN (.OUT_STATUS);
              END;
          OUT_STATUS = SMG$$SET_PHYSICAL_CURSOR (.DISPLAY_ID,
                                                 KCB [KCB_L_PASTEBOARD_ID],
                                                 CURSOR_BUF,
                                                 CURSOR_BUF_LEN,
                                                 FREE_COLS);
          IF NOT .OUT_STATUS THEN RETURN .OUT_STATUS;
          END;                          ! display_id present

      IF .DISPLAY_ID_FLAG
      THEN
          BEGIN        ! Batching check

          LOCAL
              DCB : REF BLOCK [,BYTE];

          $SMG$GET_DCB (.DISPLAY_ID, DCB);

          !+
          ! Don't allow input if display is batched.
          !-

          IF .DCB[DCB_L_BATCH_LEVEL] NEQ 0
          THEN
              RETURN  SMG$_ILLBATFNC;

          END;         ! Batching check

      !+
      ! Get prompt string, if any.
      !-

          IF .DISPLAY_ID_FLAG OR
             NOT NULLPARAMETER (K_PROMPT_STRING)
          THEN
              BEGIN
              LOCAL
                  STATUS,
```

```
                        NULL_STRING : BLOCK [8,BYTE];

                    NULL_STRING [DSC$B_CLASS] = DSC$K_CLASS_S;
                    NULL_STRING [DSC$B_DTYPE] = DSC$K_DTYPE_T;
                    NULL_STRING [DSC$W_LENGTH] = 0;
                    NULL_STRING [DSC$A_POINTER] = UPLIT (' ');

                    STATUS = SMG$$SET_PROMPT_STRING (
                                    (IF NULLPARAMETER (K_PROMPT_STRING)
                                     THEN
                                         NULL_STRING
                                     ELSE
                                         PROMPT_STRING [0,0,0,0]),
                                    ITEM_LIST [K_PROMPT,0,0,0,0],
                                    .CURSOR_BUF_LEN,
                                    CURSOR_BUF);
                    IF NOT .STATUS
                    THEN
                        RETURN .STATUS;
                    FLAGS [V_SETPROMPT] = 1;
                    END;

            !+
            ! Set up the $QIOW argument list.
            !-

            KCB [KCB_L_QIO1_FUNC] = IO$_READVBLK+IO$M_EXTEND;
            KCB [KCB_L_QIO1_P1] = BUFFER;
            KCB [KCB_L_QIO1_P2] = %ALLOCATION(BUFFER) - 4; ! Reserve 4 for term. echo
            KCB [KCB_L_QIO1_P3] = 0;
            KCB [KCB_L_QIO1_P4] = 0;
            KCB [KCB_L_QIO1_P5] = ITEM_LIST;
            KCB [KCB_L_QIO1_P6] = %ALLOCATION(ITEM_LIST) - 4; ! Extra longword added
                                                              ! by $ITMLST_DECL.

            IF .DISPLAY_ID_FLAG
            THEN
                    BEGIN
                    IF PROMPT_STRING [0,0,0,0] NEQA 0
                    THEN
                        FREE_COLS = .FREE_COLS - .PROMPT_STRING [DSC$W_LENGTH];
                    IF .FREE_COLS LEQ 0
                    THEN
                        !+
                        ! No room to accept input.  Exit.
                        !-
                        RETURN (SMG$_INVCOL);       ! improve this message later

                    IF .KCB [KCB_L_QIO1_P2] GTR .FREE_COLS
                    THEN
                        KCB [KCB_L_QIO1_P2] = .FREE_COLS;
                    END;

            !+
            ! Start with a default state, indicate that the state lock is off and
            ! get the pointer to the next open position in the buffer.
            !-
```

```
	TKEY_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
	TKEY_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
	TKEY_DSC [DSC$A_POINTER] = TKEY;
	NEXT_POS = BUFFER;

	!+
	! Loop until a terminator is seen, composing the line.
	!-

	WHILE 1 DO
	    BEGIN

	    !+
	    ! Prevent last portion of line from echoing unless later
	    ! shown to be necessary.
	    !-

	    FLAGS [V_SKIP_LAST_ECHO] = 1;

	    !+
	    ! Read the next part of the line.
	    !-

		BEGIN
		LOCAL
		    QIO_STATUS;
		QIO_STATUS = CALLG (KCB [KCB_R_QIO1], SYS$QIOW);
		IF .QIO_STATUS
		THEN
		    QIO_STATUS = .KCB [KCB_W_IOSB_STATUS];
		IF .KCB [KCB_W_IOSB_STATUS] EQLU SS$_PARTESCAPE
		    THEN
			BEGIN
			!+
			! Check for partial escape sequence in the buffer.  This can
			! happen if the QIO was limited to 2 characters and the input
			! string was terminated by a keypad key (which are 3 chars).
			!-

			LOCAL
			    QIO_RESULT,
			    CHAR_BUF : BYTE,
			    END_ESC_FLAG : INITIAL (0);

			BUFFER_LEN = .KCB [KCB_W_IOSB_COUNT] + .KCB [KCB_B_IOSB_TERMLEN];
					! point past data & partial terminator

			WHILE .END_ESC_FLAG EQL 0 DO
			BEGIN
			    QIO_RESULT = $QIOW (EFN = .KCB [KCB_L_EFN],
					CHAN = .KCB [KCB_W_CHANNEL],
					FUNC = (IO$_READVBLK + IO$M_ESCAPE +
						IO$M_NOECHO + IO$M_NOFILTR +
						IO$M_TRMNOECHO),
					P1 = CHAR_BUF,
					P2 = 1,
```

```
                        P4 = .KCB [KCB_L_QIO1_P4]);
                            ! P4 = defined terminator set
                IF NOT .QIO_RESULT THEN RET_STATUS = .QIO_RESULT;

                !+
                ! Move terminator characters to buffer.
                !-

                BUFFER [.BUFFER_LEN] = .CHAR_BUF;
                BUFFER_LEN = .BUFFER_LEN + 1;
                KCB [KCB_B_IOSB_TERMLEN] = .KCB [KCB_B_IOSB_TERMLEN] + 1;

                CASE .CHAR_BUF FROM %C'A' TO %C'~' OF
                SET
                    [%C'A' TO %C'D',        ! arrow keys
                    %C'M',                  ! Enter
                    %C'P' to %C'S',         ! PFn keys
                    %C'l' to %C'n',         ! keypad - , .
                    %C'p' to %C'y',         ! keypad 0-9
                    %C'~']:                 ! function keys F6-F20 (no codes for

                        END_ESC_FLAG = 1;

                    [INRANGE, OUTRANGE]:
                        ;                   ! didn't find end of esc terminator
                                            ! keep searching
                TES;

            END;                            ! end of WHILE loop

            QIO_STATUS = .QIO_RESULT;
                                            ! get rid of partescape error

            END;                            ! end of partial ESC terminator

        IF .QIO_STATUS NEQU SS$_NORMAL
        THEN
            BEGIN
            IF NOT .QIO_STATUS
            THEN
                BEGIN
                IF .QIO_STATUS EQLU SS$_BADESCAPE
                THEN
                    KCB [KCB_B_IOSB_TERMLEN] = 0
                ELSE
                    BEGIN
                    RET_STATUS = .QIO_STATUS;
                    EXITLOOP;
                    END;
                END
            ELSE IF (.QIO_STATUS EQLU SS$_CONTROLY) OR
                    (.QIO_STATUS EQLU SS$_CONTROLC)
            THEN
                BEGIN
                ITEM_LIST [K_INIOFFSET, ITM$L_BUFADR] = 1;
                ITEM_LIST [K_INISTRNG, ITM$W_BUFSIZ] = 0;
                FLAGS [V_SKIP_NEWLINE] = 1;     ! Indicate that NEWLINE should not echo
```

```
                        EXITLOOP;
                        END

                    END;
                END;

            !+
            ! Set the initial offset to indicate the position (1-origin)
            ! where the next read should start.  Set NEXT_POS to that
            ! location in the buffer.  Note that NEXT_POS will then point
            ! to the start of the terminator.  Set the initial string to
            ! what has been typed so far.
            !-

                BEGIN
                LOCAL
                    NEXT_INDEX; ! Index to next available position
                NEXT_INDEX = .KCB [KCB_W_IOSB_COUNT];
                ITEM_LIST [K_INIOFFSET, ITM$L_BUFADR] = .NEXT_INDEX + 1;
                ITEM_LIST [K_INISTRNG, ITM$W_BUFSIZ] = .NEXT_INDEX;
                NEXT_POS = BUFFER [.NEXT_INDEX];

                !+
                ! If the buffer is now full, exit immediately.
                !-

                IF .NEXT_INDEX GEQU .KCB [KCB_L_QIO1_P2] ! Buffer size
                THEN
                    EXITLOOP;
                END;

            !+
            ! If the terminator was <CR>, exit immediately.
            !-

            IF CH$RCHAR(.NEXT_POS) EQLU K_CR
            THEN
                BEGIN
                FLAGS [V_SKIP_LAST_ECHO] = 1;
                EXITLOOP;
                END;

            !+
            ! Get the terminator name, if any.
            !-

            KEY_CODE = SMG$$TERM_TO_KEYCODE (.NEXT_POS, .KCB [KCB_B_IOSB_TERMLEN]);

            !+
            ! Reset state to default if needed.  Cause state to be reset
            ! next time unless disabled later.
            !-

            IF TESTBITSS (FLAGS [V_RESET_STATE])
            THEN
                BEGIN
                TKEY_DSC_PTR = KTH [KTH_R_DEF_STATE_DESCR];
```

```
                TKEY_POS = .KTH [KTH_A_DEF_KEYCODE];
                END;

            !+
            ! Construct TKEY.
            !-

            TKEY_POS [0] = .KEY_CODE;

            !+
            ! Look up key to see if it is defined as anything.
            !-

            IF (SMG$$LOOKUP_KEY (.KTH, .TKEY_DSC_PTR; KDE))
            THEN
                BEGIN
                LOCAL
                    NEW_LENGTH: WORD;
                NEW_LENGTH = .ITEM_LIST [K_INISTRNG, ITM$W_BUFSIZ] +
                    .KDE [KDE_W_EQUIV_LENGTH];
                IF .NEW_LENGTH LEQU 512
                THEN
                    BEGIN
                    LOCAL
                        CURSOR_POS;     ! Position of cursor

                    !+
                    ! Append equivalence string to initial string.
                    ! Current cursor position may not be at end of string.
                    !-

                    ITEM_LIST [K_INISTRNG, ITM$W_BUFSIZ] = .NEW_LENGTH;

                    !+
                    ! If NOECHO, add the equivalence string size
                    ! to the initial string offset so that we don't
                    ! write it out.
                    !+

                    IF .KDE [KDE_V_NOECHO]
                    THEN
                        ITEM_LIST [K_INIOFFSET, ITM$L_BUFADR] =
                            .ITEM_LIST [K_INIOFFSET, ITM$L_BUFADR] +
                            .KDE [KDE_W_EQUIV_LENGTH]
                    ELSE
                        FLAGS [V_SKIP_LAST_ECHO] = 0;

                    CURSOR_POS = .NEXT_POS - .KCB [KCB_B_IOSB_POS];
                    IF .CURSOR_POS EQLA .NEXT_POS
                    THEN
                        BEGIN
                        !+
                        ! Cursor is at end of line.
                        !-
                        NEXT_POS = CH$MOVE (.KDE [KDE_W_EQUIV_LENGTH],
                            .KDE [KDE_A_EQUIV_POINTER], .NEXT_POS);
```

```
                        END
                ELSE
                        BEGIN
                        !+
                        ! Cursor is not at end of line.  Move remainder of
                        ! line over and reset offset to cursor position.
                        !-

                        NEXT_POS = CH$MOVE ((.NEXT_POS-.CURSOR_POS), .CURSOR_POS,
                            (.CURSOR_POS+.KDE [KDE_W_EQUIV_LENGTH]));
                        CH$MOVE (.KDE [KDE_W_EQUIV_LENGTH],
                            .KDE [KDE_A_EQUIV_POINTER], .CURSOR_POS);

                        !+
                        ! If not NOECHO, move the initial offset back to where
                        ! we need to display characters.
                        !-

                        IF NOT .KDE [KDE_V_NOECHO]
                        THEN
                            ITEM_LIST [K_INIOFFSET, ITM$L_BUFADR] =
                                (.CURSOR_POS-BUFFER)+1;
                        END;
                END;

            !+
            ! If this key sets a new state, store it.
            !-

            IF .KDE [KDE_V_SETSTATE]
            THEN
                BEGIN
                IF .KDE [KDE_V_LOCK]            ! Locks new state?
                THEN
                        !+
                        ! Select the default state string as the
                        ! the basis for TKEY.
                        !-
                        TKEY_DSC_PTR = KTH [KTH_R_DEF_STATE_DESCR]
                ELSE
                        BEGIN
                        !+
                        ! Use local TKEY descriptor.  Cause state to not
                        ! get reset to default on next iteration.
                        !-
                        TKEY_DSC_PTR = TKEY_DSC;
                        FLAGS [V_RESET_STATE] = 0;
                        END;

                TKEY_POS = CH$MOVE (.KDE [KDE_W_STATE_LENGTH],
                    .KDE [KDE_A_STATE_POINTER],
                    .TKEY_DSC_PTR [DSC$A_POINTER]);
                CH$WCHAR_A (0, TKEY_POS);
                IF .KDE [KDE_V_LOCK]
                THEN
                    KTH [KTH_A_DEF_KEYCODE] = .TKEY_POS;
                TKEY_DSC_PTR [DSC$W_LENGTH] = .KDE [KDE_W_STATE_LENGTH] + 3;
```

```
                    END;

                !+
                ! If this terminates the read, exit the loop.
                !-

                IF .KDE [KDE_V_TERMINATE]
                THEN
                    EXITLOOP;

                END
            ELSE
                BEGIN

                !+
                ! Key is not defined.
                !-

                LOCAL
                    CURSOR_POS;

                !+
                ! Cursor position must be moved to the end of the line.
                !-

                CURSOR_POS = .NEXT_POS - .KCB [KCB_B_IOSB_POS];
                ITEM_LIST [K_INIOFFSET, ITM$L_BUFADR] =
                    (.CURSOR_POS-BUFFER)+1;

                !+
                ! If it is ^Z, then terminate anyway.
                !-

                IF CH$RCHAR (.NEXT_POS) EQLU K_CTRLZ
                THEN
                    BEGIN

                    !+
                    ! If ^Z was the first character in the line,
                    ! return SMG$_EOF now.  Otherwise, record the
                    ! fact that ^Z was typed for use on the next read.
                    !-
                    IF .KCB [KCB_W_IOSB_COUNT] EQL 0
                    THEN
                        RET_STATUS = SMG$_EOF   ! End-of-file
                    ELSE
                        KCB [KCB_V_CTRLZ] = 1;

                    FLAGS [V_ECHO_CTRLZ] = 1;

                    EXITLOOP;
                    END

                !+
                ! Else if the terminator was ENTER or DO and wasn't defined,
                ! treat it as <CR>.
                !-
```

```
                ELSE IF (.KEY_CODE EQL SMG$K_TRM_ENTER) OR
                        (.KEY_CODE EQL SMG$K_TRM_DO)
                THEN
                    BEGIN
                    FLAGS [V_SKIP_LAST_ECHO] = 1;
                    EXITLOOP;
                    END;

                END;

            END;

        !+
        ! Do a final read so that the last equivalence characters
        ! (if any) will echo.  Do the read with a zero-timeout so
        ! that we don't get any more characters.
        !-

        IF NOT .FLAGS [V_SKIP_LAST_ECHO]
        THEN
            BEGIN
            KCB [KCB_L_QIO1_P2] = .ITEM_LIST [K_INISTRNG, ITM$W_BUFSIZ];
            ITEM_LIST [K_TERM, ITM$W_ITMCOD] = TRM$_TIMEOUT;
            ITEM_LIST [K_TERM, ITM$W_BUFSIZ] = 0;
            ITEM_LIST [K_TERM, ITM$L_BUFADR] = 0;
            CALLG (KCB [KCB_R_QIO1], SYS$QIOW);
            END;

        !+
        ! If DISPLAY_ID was passed, call back output side to tell it
        ! what changed on the screen as a result of this prompt & input.
        !-

        IF .DISPLAY_ID_FLAG
        THEN
            BEGIN
            LOCAL
                OUT_STATUS;
            IF PROMPT_STRING [0,0,0,0] NEQA 0
            THEN
                BEGIN                       ! report prompt string change
                OUT_STATUS = SMG$$REPORT_CHANGE_REPLACE (.DISPLAY_ID,
                                                KCB [KCB_L_PASTEBOARD_ID],
                                                %REF (.PROMPT_STRING [DSC$W_LENGTH]),
                                                .PROMPT_STRING [DSC$A_POINTER]);
                IF NOT .OUT_STATUS AND .OUT_STATUS NEQ SMG$_STRTERESC
                THEN
                    RETURN .OUT_STATUS;
                END;                        ! report prompt string change

            OUT_STATUS = SMG$$REPORT_CHANGE_REPLACE (.DISPLAY_ID,
                                                KCB [KCB_L_PASTEBOARD_ID],
                                                %REF (.KCB [KCB_W_IOSB_COUNT]),
                                                BUFFER);
                                        ! report user's input string
            IF NOT .OUT_STATUS THEN RETURN .OUT_STATUS;
```

```
	END;

    !+
    ! If we need to echo for ^Z, do it.
    !-

    IF .FLAGS [V_ECHO_CTRLZ]
    THEN
	BEGIN
	!+
	! Echo the appropriate sequence depending on whether this
	! is a DECCRT or not.
	! The 2 sequences can be found as translations of the
	! system logical name TTY$EXIT_STRING (set up by SYSGEN).
	! It has two equivalence strings. The first one is for
	! DECCRT's, the second one is for other terminals.
	! If this logical is not defined, we use internal echoes.
	!-
	BIND
	    KCB_DEVDEPEND2 = KCB [KCB_L_DEVDEPEND2]: BLOCK [4, BYTE];

	LOCAL
	    STATUS,
	    ITEMLIST		: VECTOR[4*3+1],
	    DECCRT_LEN		: WORD,
	    OTHER_LEN		: WORD,
	    DECCRT_PTR,
	    OTHER_PTR,
	    SYSTEM_TABLE	: VECTOR[2],
	    LOGICAL		: VECTOR[2],
	    DECCRT_ECHO		: VECTOR[LNM$C_NAMLENGTH,BYTE],
	    OTHER_ECHO		: VECTOR[LNM$C_NAMLENGTH,BYTE];

	ITEMLIST[0]=LNM$_INDEX^16+4;
	ITEMLIST[1]=UPLIT(0);
	ITEMLIST[2]=0;

	ITEMLIST[3]=LNM$_STRING^16+LNM$C_NAMLENGTH;
	ITEMLIST[4]=DECCRT_ECHO;
	ITEMLIST[5]=DECCRT_LEN;

	ITEMLIST[6]=LNM$_INDEX^16+4;
	ITEMLIST[7]=UPLIT(1);
	ITEMLIST[8]=0;

	ITEMLIST[9]=LNM$_STRING^16+LNM$C_NAMLENGTH;
	ITEMLIST[10]=OTHER_ECHO;
	ITEMLIST[11]=OTHER_LEN;

	ITEMLIST[12]=0;

	SYSTEM_TABLE[0]=%CHARCOUNT('LNM$SYSTEM_TABLE');
	SYSTEM_TABLE[1]=UPLIT BYTE('LNM$SYSTEM_TABLE');
	LOGICAL[0]=%CHARCOUNT('TTY$EXIT_STRING');
	LOGICAL[1]=UPLIT BYTE('TTY$EXIT_STRING');

	DECCRT_PTR=T_CTRLZ_DECCRT;
```

```
                DECCRT_LEN=%ALLOCATION(T_CTRLZ_DECCRT);
                OTHER_PTR=T_CTRLZ_OTHER;
                OTHER_LEN=%ALLOCATION(T_CTRLZ_OTHER);

                ! No change if logical name isn't there or
                ! any other problem translating this name.

        %IF NOT %VARIANT
        %THEN

                STATUS=$TRNLNM(TABNAM=SYSTEM_TABLE,
                               LOGNAM=LOGICAL,
                               ITMLST=ITEMLIST);
                IF .STATUS
                THEN
                    BEGIN
                    DECCRT_PTR=DECCRT_ECHO;
                    OTHER_PTR=OTHER_ECHO;
                    END;

        %FI

                KCB [KCB_L_QIO1_FUNC] = IO$_WRITEVBLK;
                IF .KCB_DEVDEPEND2 [TT2$V_DECCRT]
                THEN
                    BEGIN
                    KCB [KCB_L_QIO1_P1] = .DECCRT_PTR;
                    KCB [KCB_L_QIO1_P2] = .DECCRT_LEN;
                    END
                ELSE
                    BEGIN
                    KCB [KCB_L_QIO1_P1] = .OTHER_PTR;
                    KCB [KCB_L_QIO1_P2] = .OTHER_LEN;
                    END;

                CALLG (KCB [KCB_R_QIO1], SYS$QIOW);
                END;

        !+
        ! Echo NEWLINE, if needed.
        !-

        IF NOT .FLAGS [V_SKIP_NEWLINE]
        THEN
            BEGIN
            KCB [KCB_L_QIO1_FUNC] = IO$_WRITEVBLK+IO$M_NEWLINE;
            KCB [KCB_L_QIO1_P1] = T_CR;
            KCB [KCB_L_QIO1_P2] = 1;
            CALLG (KCB [KCB_R_QIO1], SYS$QIOW);
            END;

        !+
        ! Set BUFFER_PTR and BUFFER_LEN.
        !-

        BUFFER_PTR = BUFFER;
        BUFFER_LEN = .ITEM_LIST [K_INISTRNG, ITM$W_BUFSIZ];
```

```
3
3            !+
3            ! Delete the prompt string, if any.
3            !-
3
3            IF .FLAGS [V_SETPROMPT]
3            THEN
3                SMG$$DELETE_PROMPT_STRING (ITEM_LIST [K_PROMPT,0,0,0,0]);
3            END;
2
2        !+
2        ! We now have the complete line in BUFFER, with length BUFFER_LEN.
2        ! Copy it to the user's string.
2        !-
2
2        COPY_STATUS = LIB$SCOPY_R_DX6 (.BUFFER_LEN,
2            .BUFFER_PTR, OUT_LINE [0,0,0,0]);
2        IF NOT .COPY_STATUS
2        THEN
2            RETURN .COPY_STATUS;
2
2        !+
2        ! Fill in OUT_LENGTH if requested.
2        !-
2
2        IF NOT NULLPARAMETER (K_OUT_LENGTH)
2        THEN
3            BEGIN
3            LOCAL
3                LINE_LENGTH: WORD;
3            LIB$ANALYZE_SDESC_R2 (OUT_LINE [0,0,0,0]; LINE_LENGTH);
3            IF .LINE_LENGTH LSSU .BUFFER_LEN
3            THEN
3                BUFFER_LEN = .LINE_LENGTH;
3            OUT_LENGTH [0] = .BUFFER_LEN;
2            END;
2
2        RETURN .RET_STATUS;
2
1        END;                                ! End of routine SMG$READ_COMPOSED_LINE


                                0D  007A1 P.AAD:   .BYTE   13                 ;
                                    007A2          .BLKB   2
                                1B  007A4 T_CTRLZ_DECCRT:
                                                   .BYTE   27                 ;
                                37  007A5          .ASCII  \7\                ;
                                1B  007A6          .BYTE   27                 ;
                        6D  37  5B  007A7          .ASCII  \[7m\              ;
            20  74  69  78  45  20  007AA          .ASCII  \ Exit \           ;
                                1B  007B0          .BYTE   27                 ;
                            6D  5B  007B1          .ASCII  \[m\               ;
                                1B  007B3          .BYTE   27                 ;
                                38  007B4          .ASCII  \8\                ;
                                    007B5          .BLKB   3
            2A  54  49  58  45  2A  007B8 T_CTRLZ_OTHER:
```

```
                                                                   .ASCII  \*EXIT*\                                              ;
                                                007BE              .BLKB   2
                                 00  00  00  20 007C0 P.AAE:       .ASCII  \ \<0><0><0>                                          ;
                                       00000000 007C4 P.AAF:       .LONG   0                                                     ;
                                       00000001 007C8 P.AAG:       .LONG   1                                                     ;
4C 42 41 54 5F 4D 45 54 53 59 53 24 4D 4E 4C    007CC P.AAH:       .ASCII  \LNM$SYSTEM_TABLE\                                    ;
                                             45 007DB                                                                            ;
47 4E 49 52 54 53 5F 54 49 58 45 24 59 54 54    007DC P.AAI:       .ASCII  \TTY$EXIT_STRING\                                     ;

                                                      T_CR=           P.AAD
                                                                   .EXTRN  SMG$_INVKTB_ID, SYS$TRNLNM

                                           0FFC 00000              .ENTRY  SMG$READ_COMPOSED_LINE, Save R2,R3,R4,R5,-    ; 1524
                                                                           R6,R7,R8,R9,R10,R11
                     5E      FB0C    CE  9E 00002              MOVAB   -1268(SP), SP                                         ;
                                     7E  D4 00007              CLRL    DISPLAY_ID_FLAG                                       ; 1621
            50       6C              03  83 00009              SUBB3   #3, (AP), DIFF                                        ; 1709
                     03              50  91 0000D              CMPB    DIFF, #3                                              ;
                                     08  1B 00010              BLEQU   1$                                                    ;
                     50 00000000G    8F  D0 00012              MOVL    #SMG$_WRONUMARG, R0                                   ;
                                     04     00019              RET                                                           ;
                     57        04    BC  D0 0001A 1$:          MOVL    @KEYBOARD_ID, KCB                                     ; 1715
                                     06  13 0001E              BEQL    2$                                                    ;
                     57        DC    A7  D1 00020              CMPL    -36(KCB), KCB                                         ;
                                     08  13 00024              BEQL    3$                                                    ;
                     50 00000000G    8F  D0 00026 2$:          MOVL    #SMG$_INVKBD_ID, R0                                   ;
                                     04     0002D              RET                                                           ;
                     5B        08    BC  D0 0002E 3$:          MOVL    @KEY_TABLE_ID, KTH                                    ; 1721
                                     06  13 00032              BEQL    4$                                                    ;
                     5B        10    AB  D1 00034              CMPL    16(KTH), KTH                                          ;
                                     08  13 00038              BEQL    5$                                                    ;
                     50 00000000G    8F  D0 0003A 4$:          MOVL    #SMG$_INVKTB_ID, R0                                   ;
                                     04     00041              RET                                                           ;
               14    AE              01  D0 00042 5$:          MOVL    #1, RET_STATUS                                        ; 1727
                     30        FC    A7  E9 00046              BLBC    -4(KCB), 8$                                           ; 1733
                                     57  DD 0004A              PUSHL   KCB                                                   ; 1744
          00000000G  00              01  FB 0004C              CALLS   #1, SYS$GET                                           ;
                     17              50  E8 00053              BLBS    GET_STATUS, 7$                                        ; 1745
          0001827A   8F              50  D1 00056              CMPL    GET_STATUS, #98938                                    ; 1748
                                     0A  12 0005D              BNEQ    6$                                                    ;
               14    AE 00000000G    8F  D0 0005F              MOVL    #SMG$_EOF, RET_STATUS                                 ; 1750
                                     04  11 00067              BRB     7$                                                    ;
               14    AE              50  D0 00069 6$:          MOVL    GET_STATUS, RET_STATUS                                ; 1752
               1C    AE        28    A7  D0 0006D 7$:          MOVL    40(KCB), BUFFER_PTR                                   ; 1759
               10    AE        22    A7  B0 00072              MOVW    34(KCB), BUFFER_LEN                                   ; 1760
                             0561    31     00077              BRW     70$                                                   ; 1733
                               24    AE  D4 0007A 8$:          CLRL    CURSOR_BUF_LEN                                        ; 1769
            08       FC      A7      01  E5 0007D              BBCC    #1, -4(KCB), 9$                                       ; 1780
                     50 00000000G    8F  D0 00082              MOVL    #SMG$_EOF, R0                                         ; 1782
                                     04     00089              RET                                                           ;
                               08    AE  94 0008A 9$:          CLRB    FLAGS                                                 ; 1788
               08    AE              01  88 0008D              BISB2   #1, FLAGS                                             ; 1789
                     50      FD90    CD  9E 00091              MOVAB   ITEM_LIST, $$ITMBLKPTR                                ; 1803
                                     80  D4 00096              CLRL    ($$ITMBLKPTR)+                                        ;
            80       08   AB 00005000    8F  C9 00098              BISL3   #20480, 8(KTH), ($$ITMBLKPTR)+                        ;
                                     80  D4 000A1              CLRL    ($$ITMBLKPTR)+                                        ;
                     80 00040001     8F  D0 000A3              MOVL    #262145, ($$ITMBLKPTR)+                               ;
```

```
              80       FF08  CF  9E 000AA        MOVAB   T_CR, ($$ITMBLKPTR)+                  ;
                             80  D4 000AF        CLRL    ($$ITMBLKPTR)+                        ;
              80   00050000  8F  D0 000B1        MOVL    #327680, ($$ITMBLKPTR)+               ;
              80       FDFC  CD  9E 000B8        MOVAB   BUFFER, ($$ITMBLKPTR)+                ;
                             80  D4 000BD        CLRL    ($$ITMBLKPTR)+                        ;
              80   00080000  8F  D0 000BF        MOVL    #524288, ($$ITMBLKPTR)+               ;
                             80  7C 000C6        CLRQ    ($$ITMBLKPTR)+                        ;
              80   00030000  8F  D0 000C8        MOVL    #196608, ($$ITMBLKPTR)+               ;
              80         0C  AB  D0 000CF        MOVL    12(KTH), ($$ITMBLKPTR)+               ;
                             80  7C 000D3        CLRQ    ($$ITMBLKPTR)+                        ;
              06             6C  91 000D5        CMPB    (AP), #6                              ; 1805
                             3F  1F 000D8        BLSSU   12$                                   ;
                         18  AC  D5 000DA        TSTL    24(AP)                                ;
                             3A  13 000DD        BEQL    12$                                   ;
              6E             01  D0 000DF        MOVL    #1, DISPLAY_ID_FLAG                   ; 1819
                         F8  A7  D5 000E2        TSTL    -8(KCB)                               ; 1820
                             18  18 000E5        BGEQ    10$                                   ;
                         F8  A7  9F 000E7        PUSHAB  -8(KCB)                               ; 1825
                         72  A7  9F 000EA        PUSHAB  114(KCB)                              ; 1824
      20      AE         70  A7  32 000ED        CVTWL   112(KCB), 32(SP)                      ; 1823
                         20  AE  9F 000F2        PUSHAB  32(SP)                                ;
00000000G     00             03  FB 000F5        CALLS   #3, SMG$$GET_PASTEBOARD_ID            ; 1825
              17             50  E9 000FC        BLBC    OUT_STATUS, 11$                       ; 1826
                         20  AE  9F 000FF  10$:  PUSHAB  FREE_COLS                             ; 1829
                         28  AE  9F 00102        PUSHAB  CURSOR_BUF_LEN                        ;
                       FD80  CD  9F 00105        PUSHAB  CURSOR_BUF                            ;
                         F8  A7  9F 00109        PUSHAB  -8(KCB)                               ;
                         18  AC  DD 0010C        PUSHL   DISPLAY_ID                            ;
00000000G     00             05  FB 0010F        CALLS   #5, SMG$$SET_PHYSICAL_CURSOR          ;
              70             50  E9 00116  11$:  BLBC    OUT_STATUS, 21$                       ; 1833
              2D             6E  E9 00119  12$:  BLBC    DISPLAY_ID_FLAG, 16$                  ; 1836
              50         18  BC  D0 0011C        MOVL    @DISPLAY_ID, R0                       ; 1843
      18      BC         38  A0  D1 00120        CMPL    56(R0), @DISPLAY_ID                   ;
                             06  12 00125        BNEQ    13$                                   ;
              11         44  A0  91 00127        CMPB    68(R0), #17                           ;
                             08  13 0012B        BEQL    14$                                   ;
              50  00000000G  8F  D0 0012D  13$:  MOVL    #SMG$_INVDIS_ID, R0                   ;
                                 04 00134        RET                                           ;
              50         18  BC  D0 00135  14$:  MOVL    @DISPLAY_ID, DCB                      ;
                         1C  A0  D5 00139        TSTL    28(DCB)                               ; 1849
                             08  13 0013C        BEQL    15$                                   ;
              50  00000000G  8F  D0 0013E        MOVL    #SMG$_ILLBATFNC, R0                   ; 1851
                                 04 00145        RET                                           ;
              0A             6E  E8 00146  15$:  BLBS    DISPLAY_ID_FLAG, 17$                  ; 1859
              04             6C  91 00149  16$:  CMPB    (AP), #4                              ; 1860
                             43  1F 0014C        BLSSU   23$                                   ;
                         10  AC  D5 0014E        TSTL    16(AP)                                ;
                             3E  13 00151        BEQL    23$                                   ;
    0270      CE   010E0000  8F  D0 00153  17$:  MOVL    #17694720, NULL_STRING                ; 1869
    0274      CE       FE75  CF  9E 0015C        MOVAB   P.AAE, NULL_STRING+4                  ; 1870
                       0278  CE  9F 00163        PUSHAB  CURSOR_BUF                            ; 1878
                         28  AE  DD 00167        PUSHL   CURSOR_BUF_LEN                        ; 1879
                       FD9C  CD  9F 0016A        PUSHAB  ITEM_LIST+12                          ; 1878
              04             6C  91 0016E        CMPB    (AP), #4                              ; 1873
                             05  1F 00171        BLSSU   18$                                   ;
                         10  AC  D5 00173        TSTL    16(AP)                                ;
                             09  12 00176        BNEQ    19$                                   ;
```

```
                     50      027C   CE  9E 00178 18$:      MOVAB   NULL_STRING, R0                         ;
                                    50  DD 0017D           PUSHL   R0                                      ;
                                    03  11 0017F           BRB     20$                                     ;
                               10   AC  DD 00181 19$:      PUSHL   PROMPT_STRING                           ; 1877
              0000V   CF            04  FB 00184 20$:      CALLS   #4, SMG$$SET_PROMPT_STRING              ; 1878
                      01            50  E8 00189 21$:      BLBS    STATUS, 22$                             ; 1881
                                        04 0018C           RET                                             ;
                  08  AE            08  88 0018D 22$:      BISB2   #8, FLAGS                               ; 1884
                  0C  A7     8031   8F  3C 00191 23$:      MOVZWL  #32817, 12(KCB)                         ; 1891
                  1C  A7     FDFC   CD  9E 00197           MOVAB   BUFFER, 28(KCB)                         ; 1892
                  20  A7     0200   8F  3C 0019D           MOVZWL  #512, 32(KCB)                           ; 1893
                               24   A7  7C 001A3           CLRQ    36(KCB)                                 ; 1894
                  2C  A7     FD90   CD  9E 001A6           MOVAB   ITEM_LIST, 44(KCB)                      ; 1896
                  30  A7            3C  D0 001AC           MOVL    #60, 48(KCB)                            ; 1897
                      25            6E  E9 001B0           BLBC    DISPLAY_ID_FLAG, 26$                    ; 1900
                               10   AC  D5 001B3           TSTL    PROMPT_STRING                           ; 1903
                                    08  13 001B6           BEQL    24$                                     ;
                      50       10   BC  3C 001B8           MOVZWL  @PROMPT_STRING, R0                      ; 1905
                  20  AE            50  C2 001BC           SUBL2   R0, FREE_COLS                           ;
                      50       20   AE  D0 001C0 24$:      MOVL    FREE_COLS, R0                           ; 1906
                                    08  14 001C4           BGTR    25$                                     ;
                      50 00000000G  8F  D0 001C6           MOVL    #SMG$_INVCOL, R0                        ; 1911
                                        04 001CD           RET                                             ;
                      50       20   A7  D1 001CE 25$:      CMPL    32(KCB), R0                             ; 1913
                                    04  15 001D2           BLEQ    26$                                     ;
                  20  A7            50  D0 001D4           MOVL    R0, 32(KCB)                             ; 1915
                FDD2  CD     010E   8F  B0 001D8 26$:      MOVW    #270, TKEY_DSC+2                        ; 1923
                FDD4  CD     FDD8   CD  9E 001DF           MOVAB   TKEY, TKEY_DSC+4                        ; 1925
                  04  AE     FDFC   CD  9E 001E6           MOVAB   BUFFER, NEXT_POS                        ; 1926
                  08  AE            04  88 001EC 27$:      BISB2   #4, FLAGS                               ; 1940
           00000000G  00            67  FA 001F0           CALLG   (KCB), SYS$QIOW                         ; 1949
                      52            50  D0 001F7           MOVL    R0, QIO_STATUS                          ;
                      04            52  E9 001FA           BLBC    QIO_STATUS, 28$                         ; 1950
                      52       68   A7  32 001FD           CVTWL   104(KCB), QIO_STATUS                    ; 1952
                01FC  8F       68   A7  B1 00201 28$:      CMPW    104(KCB), #508                          ; 1953
                                    03  13 00207           BEQL    29$                                     ;
                                  00DA  31 00209           BRW     37$                                     ;
                                    53  D4 0020C 29$:      CLRL    END_ESC_FLAG                            ; 1955
                      50       6E   A7  9A 0020E           MOVZBL  110(KCB), R0                            ; 1967
      10      AE      6A   A7       50  A1 00212           ADDW3   R0, 106(KCB), BUFFER_LEN                ;
                                    53  D5 00218 30$:      TSTL    END_ESC_FLAG                            ; 1970
                                    03  13 0021A           BEQL    31$                                     ;
                                  00C4  31 0021C           BRW     36$                                     ;
                                    7E  7C 0021F 31$:      CLRQ    -(SP)                                   ; 1979
                               28   A7  DD 00221           PUSHL   40(KCB)                                 ;
                      7E            01  7D 00224           MOVQ    #1, -(SP)                               ;
                               3C   AE  9F 00227           PUSHAB  CHAR_BUF                                ;
                                    7E  7C 0022A           CLRQ    -(SP)                                   ;
                                    7E  D4 0022C           CLRL    -(SP)                                   ;
                      7E     5271   8F  3C 0022E           MOVZWL  #21105, -(SP)                           ;
                      7E       D4   A7  32 00233           CVTWL   -44(KCB), -(SP)                         ;
                               D0   A7  DD 00237           PUSHL   -48(KCB)                                ;
           00000000G  00            0C  FB 0023A           CALLS   #12, SYS$QIOW                           ;
                      04            50  E8 00241           BLBS    QIO_RESULT, 32$                         ; 1981
                  14  AE            50  D0 00244           MOVL    QIO_RESULT, RET_STATUS                  ;
                      51       10   AE  3C 00248 32$:      MOVZWL  BUFFER_LEN, R1                          ; 1987
           FDFC CD41           28   AE  90 0024C           MOVB    CHAR_BUF, BUFFER[R1]                    ;
```

```
                              10    AE  B6 00253        INCW    BUFFER_LEN
                              6E    A7  96 00256        INCB    110(KCB)
           3D      41   8F    28    AE  8F 00259        CASEB   CHAR_BUF, #65, #61
 007E     007E     007E       007E         0025F 33$:   .WORD   34$-33$,-
 FFB9     FFB9     FFB9       FFB9         00267                34$-33$,-
 FFB9     FFB9     FFB9       FFB9         0026F                34$-33$,-
 007E     FFB9     FFB9       007E         00277                34$-33$,-
 FFB9     007E     007E       007E         0027F                30$-33$,-
 FFB9     FFB9     FFB9       FFB9         00287                30$-33$,-
 FFB9     FFB9     FFB9       FFB9         0028F                30$-33$,-
 FFB9     FFB9     FFB9       FFB9         00297                30$-33$,-
 FFB9     FFB9     FFB9       FFB9         0029F                30$-33$,-
 FFB9     FFB9     FFB9       FFB9         002A7                30$-33$,-
 007E     FFB9     FFB9       FFB9         002AF                30$-33$,-
 007E     FFB9     007E       007E         002B7                30$-33$,-
 007E     007E     007E       007E         002BF                34$-33$,-
 007E     007E     007E       007E         002C7                30$-33$,-
 FFB9     FFB9     FFB9       007E         002CF                30$-33$,-
                   007E       FFB9         002D7                34$-33$,-
                                                                34$-33$,-
                                                                34$-33$,-
                                                                34$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                30$-33$,-
                                                                34$-33$,-
                                                                34$-33$,-
                                                                34$-33$,-
                                                                30$-33$,-
                                                                34$-33$,-
                                                                34$-33$,-
                                                                34$-33$,-
                                                                34$-33$,-
                                                                34$-33$,-
                                                                34$-33$,-
                                                                34$-33$,-
```

```
                                                          34$-33$,-                        ;
                                                          34$-33$,-                        ;
                                                          34$-33$,-                        ;
                                                          30$-33$,-                        ;
                                                          30$-33$,-                        ;
                                                          30$-33$,-                        ;
                                                          30$-33$,-                        ;
                                                          34$-33$                          ;
                            03  11 002DB        BRB       35$                              ;
            53              01  D0 002DD 34$:   MOVL      #1, END_ESC_FLAG                 ; 2000
                          FF35  31 002E0 35$:   BRW       30$                              ; 1970
            52              50  D0 002E3 36$:   MOVL      QIO_RESULT, QIO_STATUS           ; 2009
            01              52  D1 002E6 37$:   CMPL      QIO_STATUS, #1                   ; 2014
                            35  13 002E9        BEQL      42$                              ;
            10              52  E8 002EB        BLBS      QIO_STATUS, 39$                  ; 2017
            3C              52  D1 002EE        CMPL      QIO_STATUS, #60                  ; 2020
                            05  12 002F1        BNEQ      38$                              ;
                      6E    A7  94 002F3        CLRB      110(KCB)                         ; 2022
                            28  11 002F6        BRB       42$                              ;
        14     AE           52  D0 002F8 38$:   MOVL      QIO_STATUS, RET_STATUS           ; 2025
                            1F  11 002FC        BRB       41$                              ; 2024
  00000611     8F           52  D1 002FE 39$:   CMPL      QIO_STATUS, #1553                ; 2029
                            09  13 00305        BEQL      40$                              ;
  00000651     8F           52  D1 00307        CMPL      QIO_STATUS, #1617                ; 2030
                            10  12 0030E        BNEQ      42$                              ;
      FDB8     CD           01  D0 00310 40$:   MOVL      #1, ITEM_LIST+40                 ; 2033
                    FDA8    CD  B4 00315        CLRW      ITEM_LIST+24                     ; 2034
        08     AE           02  88 00319        BISB2     #2, FLAGS                        ; 2035
                          014C  31 0031D 41$:   BRW       60$                              ; 2032
        50            6A    A7  3C 00320 42$:   MOVZWL    106(KCB), NEXT_INDEX             ; 2053
      FDB8     CD     01    A0  9E 00324        MOVAB     1(R0), ITEM_LIST+40              ; 2054
      FDA8     CD           50  B0 0032A        MOVW      NEXT_INDEX, ITEM_LIST+24         ; 2055
        04     AE   FDFC  CD40  9E 0032F        MOVAB     BUFFER[NEXT_INDEX], NEXT_POS     ; 2056
        20     A7           50  D1 00336        CMPL      NEXT_INDEX, 32(KCB)              ; 2062
                            E1  1E 0033A        BGEQU     41$                              ;
               0D     04    BE  91 0033C        CMPB      @NEXT_POS, #13                   ; 2071
                            03  12 00340        BNEQ      43$                              ;
                          0123  31 00342        BRW       59$                              ;
               7E     6E    A7  9A 00345 43$:   MOVZBL    110(KCB), -(SP)                  ; 2082
                      08    AE  DD 00349        PUSHL     NEXT_POS                         ;
  00000000G    00           02  FB 0034C        CALLS     #2, SMG$$TERM_TO_KEYCODE         ;
        18     AE           50  B0 00353        MOVW      R0, KEY_CODE                     ;
  09    08     AE           00  E3 00357        BBCS      #0, FLAGS, 44$                   ; 2089
               5A     14    AB  9E 0035C        MOVAB     20(R11), TKEY_DSC_PTR            ; 2092
        0C     AE     1C    AB  D0 00360        MOVL      28(KTH), TKEY_POS                ; 2093
        0C     BE     18    AE  B0 00365 44$:   MOVW      KEY_CODE, @TKEY_POS              ; 2100
               51           5A  D0 0036A        MOVL      TKEY_DSC_PTR, R1                 ; 2106
               50           5B  D0 0036D        MOVL      KTH, R0                          ;
           00000000G        00  16 00370        JSB       SMG$$LOOKUP_KEY                  ;
               03           50  E8 00376        BLBS      R0, 45$                          ;
                          00A6  31 00379        BRW       55$                              ;
               59     14    A6  3C 0037C 45$:   MOVZWL    20(KDE), R9                      ; 2112
  50    FDA8   CD           59  A1 00380        ADDW3     R9, ITEM_LIST+24, NEW_LENGTH     ;
        0200   8F           50  B1 00386        CMPW      NEW_LENGTH, #512                 ; 2113
                            54  1A 0038B        BGTRU     49$                              ;
      FDA8     CD           50  B0 0038D        MOVW      NEW_LENGTH, ITEM_LIST+24         ; 2124
               07     30    A6  E9 00392        BLBC      48(KDE), 46$                     ; 2132
```

```
                 FDB8   CD           59  C0 00396       ADDL2   R9, ITEM_LIST+40                          ; 2136
                                     04  11 0039B       BRB     47$                                       ; 2134
                   08   AE           04  8A 0039D 46$:  BICB2   #4, FLAGS                                 ; 2138
                        58       6F  A7  9A 003A1 47$:  MOVZBL  111(KCB), CURSOR_POS                      ; 2140
            58     04   AE           58  C3 003A5       SUBL3   CURSOR_POS, NEXT_POS, CURSOR_POS          ;
                   04   AE           58  D1 003AA       CMPL    CURSOR_POS, NEXT_POS                      ; 2141
                                     0C  12 003AE       BNEQ    48$                                       ;
      04    BE     18   B6           59  28 003B0       MOVC3   R9, @24(KDE), @NEXT_POS                   ; 2148
                   04   AE           53  D0 003B6       MOVL    R3, NEXT_POS                              ;
                                     25  11 003BA       BRB     49$                                       ; 2141
            53     04   AE           58  C3 003BC 48$:  SUBL3   CURSOR_POS, NEXT_POS, R3                  ; 2158
          6948          68           53  28 003C1       MOVC3   R3, (CURSOR_POS), (R9)[CURSOR_POS]        ; 2159
                   04   AE           53  D0 003C6       MOVL    R3, NEXT_POS                              ;
            68     18   B6           59  28 003CA       MOVC3   R9, @24(KDE), (CURSOR_POS)                ; 2161
                        0E       30  A6  E8 003CF       BLBS    48(KDE), 49$                              ; 2168
                        50     FDFC  CD  9E 003D3       MOVAB   BUFFER, R0                                ; 2171
                        50           58  C2 003D8       SUBL2   CURSOR_POS, R0                            ;
    FDB8    CD          01           50  C3 003DB       SUBL3   R0, #1, ITEM_LIST+40                      ;
            34     30   A6           04  E1 003E1 49$:  BBC     #4, 48(KDE), 53$                          ; 2179
            06     30   A6           02  E1 003E6       BBC     #2, 48(KDE), 50$                          ; 2182
                        5A       14  AB  9E 003EB       MOVAB   20(R11), TKEY_DSC_PTR                     ; 2188
                                     09  11 003EF       BRB     51$                                       ;
                        5A     FDD0  CD  9E 003F1 50$:  MOVAB   TKEY_DSC, TKEY_DSC_PTR                    ; 2195
                   08   AE           01  8A 003F6       BICB2   #1, FLAGS                                 ; 2196
      04    BA     20   B6       1C  A6  28 003FA 51$:  MOVC3   28(KDE), @32(KDE), @4(TKEY_DSC_PTR)       ; 2201
                   0C   AE           53  D0 00401       MOVL    R3, TKEY_POS                              ;
                                 0C  BE  94 00405       CLRB    @TKEY_POS                                 ; 2202
                                 0C  AE  D6 00408       INCL    TKEY_POS                                  ;
            05     30   A6           02  E1 0040B       BBC     #2, 48(KDE), 52$                          ; 2203
                   1C   AB       0C  AE  D0 00410       MOVL    TKEY_POS, 28(KTH)                         ; 2205
            6A     1C   A6           03  A1 00415 52$:  ADDW3   #3, 28(KDE), (TKEY_DSC_PTR)               ; 2206
            4D     30   A6           01  E0 0041A 53$:  BBS     #1, 48(KDE), 60$                          ; 2213
                                   FDCA  31 0041F 54$:  BRW     27$                                       ;
                        51       6F  A7  9A 00422 55$:  MOVZBL  111(KCB), CURSOR_POS                      ; 2232
            51     04   AE           51  C3 00426       SUBL3   CURSOR_POS, NEXT_POS, CURSOR_POS          ;
                        50     FDFC  CD  9E 0042B       MOVAB   BUFFER, R0                                ; 2234
                        50           51  C2 00430       SUBL2   CURSOR_POS, R0                            ;
    FDB8    CD          01           50  C3 00433       SUBL3   R0, #1, ITEM_LIST+40                      ;
                        1A       04  BE  91 00439       CMPB    @NEXT_POS, #26                            ; 2240
                                     19  12 0043D       BNEQ    58$                                       ;
                                 6A  A7  B5 0043F       TSTW    106(KCB)                                  ; 2249
                                     0A  12 00442       BNEQ    56$                                       ;
                   14   AE 00000000G 8F  D0 00444       MOVL    #SMG$_EOF, RET_STATUS                     ; 2251
                                     04  11 0044C       BRB     57$                                       ;
                   FC   A7           02  88 0044E 56$:  BISB2   #2, -4(KCB)                               ; 2253
                   08   AE           10  88 00452 57$:  BISB2   #16, FLAGS                                ; 2255
                                     14  11 00456       BRB     60$                                       ; 2242
                 010E   8F       18  AE  B1 00458 58$:  CMPW    KEY_CODE, #270                            ; 2265
                                     08  13 0045E       BEQL    59$                                       ;
                 0128   8F       18  AE  B1 00460       CMPW    KEY_CODE, #296                            ; 2266
                                     B7  12 00466       BNEQ    54$                                       ;
                   08   AE           04  88 00468 59$:  BISB2   #4, FLAGS                                 ; 2269
            1A     08   AE           02  E0 0046C 60$:  BBS     #2, FLAGS, 61$                            ; 2283
                   20   A7     FDA8  CD  3C 00471       MOVZWL  ITEM_LIST+24, 32(KCB)                     ; 2286
                 FDC0   CD 00020000  8F  D0 00477       MOVL    #131072, ITEM_LIST+48                     ; 2288
                               FDC4  CD  D4 00480       CLRL    ITEM_LIST+52                              ; 2289
            00000000G   00           67  FA 00484       CALLG   (KCB), SYS$QIOW                           ; 2290
```

```
                  47                6E  E9 0048B 61$:    BLBC    DISPLAY_ID_FLAG, 63$                          ; 2298
                  50          10    AC  D0 0048E         MOVL    PROMPT_STRING, R0                             ; 2303
                                    24  13 00492         BEQL    62$                                           ;
                              04    A0  DD 00494         PUSHL   4(R0)                                         ; 2309
             1C   AE                60  3C 00497         MOVZWL  (R0), 28(SP)                                  ; 2308
                              1C    AE  9F 0049B         PUSHAB  28(SP)                                        ;
                              F8    A7  9F 0049E         PUSHAB  -8(KCB)                                       ; 2307
                              18    AC  DD 004A1         PUSHL   DISPLAY_ID                                    ;
      00000000G   00                04  FB 004A4         CALLS   #4, SMG$$REPORT_CHANGE_REPLACE                ;
                  0A                50  E8 004AB         BLBS    OUT_STATUS, 62$                               ; 2310
      00000000G   8F                50  D1 004AE         CMPL    OUT_STATUS, #SMG$_STRTERESC                   ;
                                    01  13 004B5         BEQL    62$                                           ;
                                        04 004B7         RET                                                   ;
                            FDFC    CD  9F 004B8 62$:    PUSHAB  BUFFER                                        ; 2316
             1C   AE          6A    A7  3C 004BC         MOVZWL  106(KCB), 28(SP)                              ; 2317
                              1C    AE  9F 004C1         PUSHAB  28(SP)                                        ;
                              F8    A7  9F 004C4         PUSHAB  -8(KCB)                                       ; 2316
                              18    AC  DD 004C7         PUSHL   DISPLAY_ID                                    ;
      00000000G   00                04  FB 004CA         CALLS   #4, SMG$$REPORT_CHANGE_REPLACE                ;
                  01                50  E8 004D1         BLBS    OUT_STATUS, 63$                               ; 2320
                                        04 004D4         RET                                                   ;
  03        08    AE                04  E0 004D5 63$:    BBS     #4, FLAGS, 64$                                ; 2327
                                  00C8  31 004DA         BRW     68$                                           ;
           0244   CE 00010004       8F  D0 004DD 64$:    MOVL    #65540, ITEMLIST                              ; 2354
           0248   CE        FAEF    CF  9E 004E6         MOVAB   P.AAF, ITEMLIST+4                             ; 2355
                            024C    CE  D4 004ED         CLRL    ITEMLIST+8                                    ; 2356
           0250   CE 000200FF       8F  D0 004F1         MOVL    #131327, ITEMLIST+12                          ; 2358
           0254   CE        0134    CE  9E 004FA         MOVAB   DECCRT_ECHO, ITEMLIST+16                      ; 2359
           0258   CE          2C    AE  9E 00501         MOVAB   DECCRT_LEN, ITEMLIST+20                       ; 2360
           025C   CE 00010004       8F  D0 00507         MOVL    #65540, ITEMLIST+24                           ; 2362
           0260   CE        FAC9    CF  9E 00510         MOVAB   P.AAG, ITEMLIST+28                            ; 2363
                            0264    CE  D4 00517         CLRL    ITEMLIST+32                                   ; 2364
           0268   CE 000200FF       8F  D0 0051B         MOVL    #131327, ITEMLIST+36                          ; 2366
           026C   CE          34    AE  9E 00524         MOVAB   OTHER_ECHO, ITEMLIST+40                       ; 2367
           0270   CE          30    AE  9E 0052A         MOVAB   OTHER_LEN, ITEMLIST+44                        ; 2368
                            0274    CE  D4 00530         CLRL    ITEMLIST+48                                   ; 2370
           023C   CE                10  D0 00534         MOVL    #16, SYSTEM_TABLE                             ; 2372
           0240   CE        FAA4    CF  9E 00539         MOVAB   P.AAH, SYSTEM_TABLE+4                         ; 2373
           0234   CE                0F  D0 00540         MOVL    #15, LOGICAL                                  ; 2374
           0238   CE        FAA8    CF  9E 00545         MOVAB   P.AAI, LOGICAL+4                              ; 2375
                  53        FA69    CF  9E 0054C         MOVAB   T_CTRLZ_DECCRT, DECCRT_PTR                    ; 2377
             2C   AE                11  B0 00551         MOVW    #17, DECCRT_LEN                               ; 2378
                  52        FA74    CF  9E 00555         MOVAB   T_CTRLZ_OTHER, OTHER_PTR                      ; 2379
             30   AE                06  B0 0055A         MOVW    #6, OTHER_LEN                                 ; 2380
                            0244    CE  9F 0055E         PUSHAB  ITEMLIST                                      ; 2390
                                    7E  D4 00562         CLRL    -(SP)                                         ;
                            023C    CE  9F 00564         PUSHAB  LOGICAL                                       ;
                            0248    CE  9F 00568         PUSHAB  SYSTEM_TABLE                                  ;
                                    7E  D4 0056C         CLRL    -(SP)                                         ;
      00000000G   00                05  FB 0056E         CALLS   #5, SYS$TRNLNM                                ;
                  09                50  E9 00575         BLBC    STATUS, 65$                                   ; 2391
                  53        0134    CE  9E 00578         MOVAB   DECCRT_ECHO, DECCRT_PTR                       ; 2394
                  52          34    AE  9E 0057D         MOVAB   OTHER_ECHO, OTHER_PTR                         ; 2395
             0C   A7                30  D0 00581 65$:    MOVL    #48, 12(KCB)                                  ; 2400
  0B        F7    A7                05  E1 00585         BBC     #5, -9(KCB), 66$                              ; 2401
             1C   A7                53  D0 0058A         MOVL    DECCRT_PTR, 28(KCB)                           ; 2404
             20   A7          2C    AE  3C 0058E         MOVZWL  DECCRT_LEN, 32(KCB)                           ; 2405
```

```
                                09 11 00593      BRB     67$                                   ; 2401
         1C  A7             52  D0 00595 66$:    MOVL    OTHER_PTR, 28(KCB)                    ; 2409
         20  A7      30     AE  3C 00599         MOVZWL  OTHER_LEN, 32(KCB)                    ; 2410
   00000000G 00             67  FA 0059E 67$:    CALLG   (KCB), SYS$QIOW                       ; 2413
   17    08  AE             01  E0 005A5 68$:    BBS     #1, FLAGS, 69$                        ; 2420
         0C  A7    0430     8F  3C 005AA         MOVZWL  #1072, 12(KCB)                        ; 2423
         1C  A7    FA02     CF  9E 005B0         MOVAB   T_CR, 28(KCB)                         ; 2424
         20  A7             01  D0 005B6         MOVL    #T, 32(KCB)                           ; 2425
   00000000G 00             67  FA 005BA         CALLG   (KCB), SYS$QIOW                       ; 2426
         1C  AE    FDFC     CD  9E 005C1 69$:    MOVAB   BUFFER, BUFFER_PTR                    ; 2433
         10  AE    FDA8     CD  B0 005C7         MOVW    ITEM_LIST+24, BUFFER_LEN              ; 2434
   09    08  AE             03  E1 005CD         BBC     #3, FLAGS, 70$                        ; 2440
                   FD9C     CD  9F 005D2         PUSHAB  ITEM_LIST+12                          ; 2442
       0000V CF             01  FB 005D6         CALLS   #1, SMG$$DELETE_PROMPT_STRING         ;
             52      0C     AC  D0 005DB 70$:    MOVL    OUT_LINE, R2                          ; 2451
             51      1C     AE  D0 005DF         MOVL    BUFFER_PTR, R1                        ;
             50      10     AE  3C 005E3         MOVZWL  BUFFER_LEN, R0                        ;
                 00000000G  00  16 005E7         JSB     LIB$SCOPY_R_DX6                       ;
             2A             50  E9 005ED         BLBC    COPY_STATUS, 73$                      ; 2452
             05             6C  91 005F0         CMPB    (AP), #5                              ; 2460
                            21  1F 005F3         BLSSU   72$                                   ;
                     14     AC  D5 005F5         TSTL    20(AP)                                ;
                            1C  13 005F8         BEQL    72$                                   ;
             50      0C     AC  D0 005FA         MOVL    OUT_LINE, R0                          ; 2465
                 00000000G  00  16 005FE         JSB     LIB$ANALYZE_SDESC_R2                  ;
             50             51  D0 00604         MOVL    R1, LINE_LENGTH                       ;
         10  AE             50  B1 00607         CMPW    LINE_LENGTH, BUFFER_LEN               ; 2466
                            04  1E 0060B         BGEQU   71$                                   ;
         10  AE             50  B0 0060D         MOVW    LINE_LENGTH, BUFFER_LEN               ; 2468
         14  BC      10     AE  B0 00611 71$:    MOVW    BUFFER_LEN, @OUT_LENGTH               ; 2469
             50      14     AE  D0 00616 72$:    MOVL    RET_STATUS, R0                        ; 2472
                                04 0061A 73$:    RET                                           ; 2474

; Routine Size:  1563 bytes,    Routine Base:  _SMG$CODE + 07EB
```

```
%SBTTL 'SMG$DELETE_VIRTUAL_KEYBOARD'
GLOBAL ROUTINE SMG$DELETE_VIRTUAL_KEYBOARD (
    KEYBOARD_ID: REF VECTOR [, LONG]
    ) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine deletes a virtual-keyboard which was created by a call
!       to SMG$CREATE_VIRTUAL_KEYBOARD.  Any terminal attributes changed
!       when the virtual-keyboard was created are reset to their previous
!       values.
!
! CALLING SEQUENCE:
!
!       RET_STATUS.wlc.v = SMG$DELETE_VIRTUAL_KEYBOARD (
!                                   KEYBOARD_ID.r[.r)
!
! FORMAL PARAMETERS:
!
!       KEYBOARD_ID       A longword containing the identification of the
!                         virtual-keyboard which is to be deleted.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! COMPLETION STATUS:
!
!       SS$_NORMAL        Normal successful completion
!       SMG$_INVKBD_ID    Invalid keyboard-id
!       SMG$_WRONUMARG    Wrong number of arguments
!
! SIDE EFFECTS:
!
!       The keypad mode (numeric or application) is reset to its original state
!       The terminal characteristics are reset to their original state
!       The file is closed or the channel is deassigned.
!
!--

    BEGIN

    LOCAL
        KCB: REF KCB_R_KCB_STRUCT;

    $SMG$VALIDATE_ARGCOUNT(1,1);

    !+
    ! Fetch and validate KCB.
    !-

    $SMG$VALIDATE_KCB (KEYBOARD_ID, KCB);
```

```
2
2       !+
2       ! Invalidate the user's KEYBOARD_ID.
2       !-
2
2       KEYBOARD_ID [0] = 0;
2
2       !+
2       ! Call SMG$$CLEANUP to do the actual close, etc.
2       ! Return with success.
2       !-
2
2       RETURN (SMG$$CLEANUP (KCB [KCB_R_KCB], SS$_NORMAL));
2
1       END;                            ! End of routine SMG$DELETE_VIRTUAL_KEYBOARD


                              0100 00000         .ENTRY  SMG$DELETE_VIRTUAL_KEYBOARD, Save R8     ; 2476
        01              6C  91 00002         CMPB    (AP), #1                                 ; 2525
                        08  13 00005         BEQL    1$                                       ;
        50 00000000G    8F  D0 00007         MOVL    #SMG$_WRONUMARG, R0                      ;
                            04 0000E         RET                                              ;
        58          04  BC  D0 0000F 1$:     MOVL    @KEYBOARD_ID, KCB                        ; 2531
                        06  13 00013         BEQL    2$                                       ;
        58          DC  A8  D1 00015         CMPL    -36(KCB), KCB                            ;
                        08  13 00019         BEQL    3$                                       ;
        50 00000000G    8F  D0 0001B 2$:     MOVL    #SMG$_INVKBD_ID, R0                      ;
                            04 00022         RET                                              ;
                    04  BC  D4 00023 3$:     CLRL    @KEYBOARD_ID                             ; 2537
        50              01  D0 00026         MOVL    #1, R0                                   ; 2544
  0000V CF              00  FB 00029         CALLS   #0, SMG$$CLEANUP                         ;
                            04 0002E         RET                                              ; 2546

; Routine Size:  47 bytes,    Routine Base:  _SMG$CODE + 0E06
```

```
%SBTTL 'SMG$SET_KEYPAD_MODE'
GLOBAL ROUTINE SMG$SET_KEYPAD_MODE (
    KEYBOARD_ID: REF VECTOR [, LONG],
    KEYPAD_MODE: REF VECTOR [, LONG]
    ) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This procedure sets the state of the terminal's numeric keypad to either
!       applications mode or numeric mode.  In applications mode, numeric
!       keypad keys (which may also include punctuation keys) are considered
!       function keys and are useable as terminators.  In numeric mode, these
!       keys are equivalent to the keys with similar legends on the main
!       key array and are not generally useable as terminators.
!
!       For VT5x, VT1xx (and compatible LAxx), and LK201 keyboards, the
!       following table lists the legends on the keypad keys, their key name
!       in applications mode, and their key value in numeric mode:
!
!               Legend          Applications        Numeric
!               On Key              Mode              Mode
!               ------          ------------        -------
!                 0                 KP0                0
!                 1                 KP1                1
!                 2                 KP2                2
!                 3                 KP3                3
!                 4                 KP4                4
!                 5                 KP5                5
!                 6                 KP6                6
!                 7                 KP7                7
!                 8                 KP8                8
!                 9                 KP9                9
!                 -                MINUS               -
!                 ,                COMMA               ,
!                 .               PERIOD               .
!               ENTER              ENTER              <CR>
!
!       Note that VT5x keyboards do not have all of these keys.
!
!       When a virtual-keyboard is created, the terminal is set by default
!       to applications mode.  When the virtual-keyboard is deleted the
!       state of the keypad is restored to its previous mode, as determined
!       by the terminal characteristic TT2$V_APP_KEYPAD.
!
! CALLING SEQUENCE:
!
!       RET_STATUS.wlc.v = SMG$SET_KEYPAD_MODE (
!                               KEYBOARD_ID.rl.r,
!                               NEW_MODE.rl.r)
!
! FORMAL PARAMETERS:
!
!       KEYBOARD_ID     A longword containing the identification of the
!                       virtual-keyboard for which the keypad mode is being
!                       set or returned.
!
```

```
1 !      KEYPAD_MODE      A longword whose low bit is used to set the new
1 !                       state of the keypad mode.  If the low bit is set, the
1 !                       keypad is set into numeric mode.  If clear, the
1 !                       keypad is set into applications mode.  The default
1 !                       mode when a virtual-keyboard is created is
1 !                       applications mode.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !      NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !      NONE
1 !
1 ! COMPLETION STATUS:
1 !
1 !      SS$_NORMAL       Normal successful completion
1 !      SMG$_INVKBD_ID   Invalid keyboard-id
1 !      SMG$_WRONUMARG   Wrong number of arguments
1 !
1 ! SIDE EFFECTS:
1 !
1 !      NONE
1 !
1 !--
1
2     BEGIN
2
2     LOCAL
2         KCB: REF KCB_R_KCB_STRUCT;
2
2     BUILTIN
2         CALLG;
2
2     $SMG$VALIDATE_ARGCOUNT(2,2);
2
2     !+
2     ! Validate Keyboard Control Block
2     !-
2
2     $SMG$VALIDATE_KCB (KEYBOARD_ID, KCB);
2
2     !+
2     ! Only change the keypad mode if we're at a terminal.
2     !-
2
2     IF NOT .KCB [KCB_V_RMS]
2     THEN
3         BEGIN
3
3         BIND
3             KCB_DEVDEPEND2 = KCB [KCB_L_DEVDEPEND2]: BLOCK [4, BYTE];
3
3         !+
3         ! Select whether we want to set or clear applications mode.
3         !-
```

```
	IF .KEYPAD_MODE [0]
	THEN
	    BEGIN
	    !+
	    ! Set applications mode.
	    !-

	    !+
	    ! Mark the keypad as in applications mode.
	    !-

	    KCB_DEVDEPEND2 [TT2$V_APP_KEYPAD] = 1;

	    !+
	    ! Choose the proper control sequence to send.
	    !-

	    IF .KCB [KCB_V_KPDSEQ_DECCRT]
	    THEN
		BEGIN
		!+
		! Use sequence for DECCRT and VT5x.
		!-

		KCB [KCB_L_QIO2_P1] = T_KPDAPP_DECCRT;
		KCB [KCB_L_QIO2_P2] = %ALLOCATION(T_KPDAPP_DECCRT);
		END
	    ELSE SELECTONEU .KCB [KCB_B_DEVTYPE] OF
		SET
		[OTHERWISE]:
		    BEGIN
		    !+
		    ! Unknown type.
		    !-
		    KCB [KCB_L_QIO2_P1] = 0;
		    KCB [KCB_L_QIO2_P2] = 0;
		    END;
		TES;
	    END
	ELSE
	    BEGIN
	    !+
	    ! Set numeric (non-application) mode.
	    !-

	    !+
	    ! Mark the keypad as in numeric mode.
	    !-

	    KCB_DEVDEPEND2 [TT2$V_APP_KEYPAD] = 0;

	    !+
	    ! Choose the proper control sequence to send.
	    !-

	    IF .KCB [KCB_V_KPDSEQ_DECCRT]
```

```
                THEN
                    BEGIN
                    !+
                    ! Use sequence for DECCRT and VT5x.
                    !-

                    KCB [KCB_L_QIO2_P1] = T_KPDNUM_DECCRT;
                    KCB [KCB_L_QIO2_P2] = %ALLOCATION(T_KPDNUM_DECCRT);
                    END
                ELSE SELECTONEU .KCB [KCB_B_DEVTYPE] OF
                    SET
                    [OTHERWISE]:
                        BEGIN
                        !+
                        ! Unknown type.
                        !-
                        KCB [KCB_L_QIO2_P1] = 0;
                        KCB [KCB_L_QIO2_P2] = 0;
                        END;
                    TES;
                END;

            !+
            ! Write the sequence to the terminal.
            !-

            KCB [KCB_L_QIO2_FUNC] = IO$_WRITEVBLK+IO$M_NOFORMAT;
            KCB [KCB_L_QIO2_P3] = 0;
            KCB [KCB_L_QIO2_P4] = 0;
            KCB [KCB_L_QIO2_P5] = 0;
            KCB [KCB_L_QIO2_P6] = 0;
            CALLG (KCB [KCB_R_QIO2], SYS$QIOW);

            !+
            ! Set the new characteristics.
            !-

            KCB [KCB_L_QIO2_FUNC] = IO$_SETMODE;
            KCB [KCB_L_QIO2_P1] = KCB [KCB_R_TERMCHAR];
            KCB [KCB_L_QIO2_P2] = KCB_S_TERMCHAR;
            CALLG (KCB [KCB_R_QIO2], SYS$QIOW);

            !+
            ! Mark that the terminal characteristics have been changed.
            !-

            KCB [KCB_V_CHARS_CHANGED] = 1;

            END;

        RETURN SS$_NORMAL;

        END;                            ! End of routine SMG$SET_KEYPAD_MODE
```

```
                                   000C 00000       .ENTRY  SMG$SET_KEYPAD_MODE, Save R2,R3                 ; 2548
               53 00000000G  00   9E 00002          MOVAB   SYS$QIOW, R3                                    ;
               02            6C   91 00009          CMPB    (AP), #2                                        ; 2639
                             08   13 0000C          BEQL    1$                                              ;
               50 00000000G  8F   D0 0000E          MOVL    #SMG$_WRONUMARG, R0                             ;
                                  04 00015          RET                                                     ;
               52        04  BC   D0 00016 1$:      MOVL    @KEYBOARD_ID, KCB                               ; 2645
                             06   13 0001A          BEQL    2$                                              ;
               52        DC  A2   D1 0001C          CMPL    -36(KCB), KCB                                   ;
                             08   13 00020          BEQL    3$                                              ;
               50 00000000G  8F   D0 00022 2$:      MOVL    #SMG$_INVKBD_ID, R0                             ;
                                  04 00029          RET                                                     ;
               54        FC  A2   E8 0002A 3$:      BLBS    -4(KCB), 8$                                     ; 2651
               12        08  BC   E9 0002E          BLBC    @KEYPAD_MODE, 4$                                ; 2662
          F6   A2        80  8F   88 00032          BISB2   #128, -10(KCB)                                  ; 2673
     1E   FC   A2            03   E1 00037          BBC     #3, -4(KCB), 6$                                 ; 2679
          50   A2      F18B  CF   9E 0003C          MOVAB   T_KPDAPP_DECCRT, 80(KCB)                        ; 2686
                             10   11 00042          BRB     5$                                              ; 2687
          F6   A2        80  8F   8A 00044 4$:      BICB2   #128, -10(KCB)                                  ; 2711
     0C   FC   A2            03   E1 00049          BBC     #3, -4(KCB), 6$                                 ; 2717
          50   A2      F17D  CF   9E 0004E          MOVAB   T_KPDNUM_DECCRT, 80(KCB)                        ; 2724
          54   A2            02   D0 00054 5$:      MOVL    #2, 84(KCB)                                     ; 2725
                             03   11 00058          BRB     7$                                              ; 2717
                         50  A2   7C 0005A 6$:      CLRQ    80(KCB)                                         ; 2734
          40   A2      0130  8F   3C 0005D 7$:      MOVZWL  #304, 64(KCB)                                   ; 2744
                         58  A2   7C 00063          CLRQ    88(KCB)                                         ; 2745
                         60  A2   7C 00066          CLRQ    96(KCB)                                         ; 2747
               63        34  A2   FA 00069          CALLG   52(KCB), SYS$QIOW                               ; 2749
          40   A2            23   D0 0006D          MOVL    #35, 64(KCB)                                    ; 2755
          50   A2        EC  A2   9E 00071          MOVAB   -20(R2), 80(KCB)                                ; 2756
          54   A2            0C   D0 00076          MOVL    #12, 84(KCB)                                    ; 2757
               63        34  A2   FA 0007A          CALLG   52(KCB), SYS$QIOW                               ; 2758
          FC   A2            04   88 0007E          BISB2   #4, -4(KCB)                                     ; 2764
               50            01   D0 00082 8$:      MOVL    #1, R0                                          ; 2768
                                  04 00085          RET                                                     ; 2770

; Routine Size:  134 bytes,    Routine Base:  _SMG$CODE + 0E35
```

```
%SBTTL 'SMG$CANCEL_INPUT'
GLOBAL ROUTINE SMG$CANCEL_INPUT (
    KEYBOARD_ID: REF VECTOR [, LONG]
    ) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This procedure causes immediate termination
!       of a terminal read-in-progress which was issued by
!       SMG$READ_STRING or SMG$READ_COMPOSED_LINE.  If the
!       virtual-keyboard is associated with an RMS file, this procedure
!       has no effect, since it is not possible to cancel an outstanding
!       RMS input operation.
!
! CALLING SEQUENCE:
!
!       RET_STATUS.wlc.v = SMG$CANCEL_INPUT (
!                               KEYBOARD_ID.rl.r)
!
! FORMAL PARAMETERS:
!
!       KEYBOARD_ID     A longword containing the identification of the
!                       virtual keyboard for which the input is to be
!                       cancelled.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! COMPLETION STATUS:
!
!       SS$_NORMAL      Normal successful completion
!       SS$_RMSNOTCAN   Success - RMS operation cannot be cancelled
!       SMG$_INVKBD_ID  Invalid keyboard-id
!       SMG$_WRONUMARG  Wrong number of arguments
!
! SIDE EFFECTS:
!
!       NONE
!
!--

    BEGIN

    LOCAL
        KCB: REF KCB_R_KCB_STRUCT,
        CANCEL_STATUS;

    $SMG$VALIDATE_ARGCOUNT(1,1);

    !+
    ! Validate Keyboard Control Block
```

```
; 2758      2828  2       !-
; 2759      2829  2
; 2760      2830  2       $SMG$VALIDATE_KCB (KEYBOARD_ID, KCB);
; 2761      2831  2
; 2762      2832  2       !+
; 2763      2833  2       ! Only cancel input if we're at a terminal.
; 2764      2834  2       !-
; 2765      2835  2
; 2766      2836  2       IF NOT .KCB [KCB_V_RMS]
; 2767      2837  2       THEN
; 2768      2838  3           BEGIN
; 2769      2839  3
; 2770      2840  3           CANCEL_STATUS = $CANCEL (CHAN=.KCB [KCB_W_CHANNEL]);
; 2771      2841  3           IF NOT .CANCEL_STATUS
; 2772      2842  3           THEN
; 2773      2843  3               RETURN .CANCEL_STATUS;
; 2774      2844  2           END;
; 2775      2845  2
; 2776      2846  2       RETURN SS$_NORMAL;
; 2777      2847  2
; 2778      2848  1       END;                                    ! End of routine SMG$CANCEL_INPUT


                                                        .EXTRN  SYS$CANCEL

                                        0000 00000      .ENTRY  SMG$CANCEL_INPUT, Save nothing          ; 2772
                  01             6C  91 00002           CMPB    (AP), #1                                ; 2824
                                 08  13 00005           BEQL    1$                                      ;
                  50 00000000G   8F  D0 00007           MOVL    #SMG$_WRONUMARG, R0                     ;
                                     04 0000E           RET                                             ;
                  50       04    BC  D0 0000F 1$:       MOVL    @KEYBOARD_ID, KCB                       ; 2830
                                 06  13 00013           BEQL    2$                                      ;
                  50       DC    A0  D1 00015           CMPL    -36(KCB), KCB                           ;
                                 08  13 00019           BEQL    3$                                      ;
                  50 00000000G   8F  D0 0001B 2$:       MOVL    #SMG$_INVKBD_ID, R0                     ;
                                     04 00022           RET                                             ;
                  0E       FC    A0  E8 00023 3$:       BLBS    -4(KCB), 4$                             ; 2836
                  7E       D4    A0  32 00027           CVTWL   -44(KCB), -(SP)                         ; 2840
        00000000G 00             01  FB 0002B           CALLS   #1, SYS$CANCEL                          ;
                  03             50  E9 00032           BLBC    CANCEL_STATUS, 5$                       ; 2841
                  50             01  D0 00035 4$:       MOVL    #1, R0                                  ; 2846
                                     04 00038 5$:       RET                                             ; 2848

; Routine Size:  57 bytes,    Routine Base:  _SMG$CODE + 0EBB
```

```
1 %SBTTL 'SMG$$SET_PROMPT_STRING'
1 ROUTINE SMG$$SET_PROMPT_STRING (
1     PROMPT_STRING: REF BLOCK [, BYTE],           ! Prompt string descriptor
1     ITEM_LIST: REF $ITMLST_DECL (ITEMS=1),       ! Prompt string list item
1     PRE_BUF_LEN,                                 ! Pre-prompt buffer length
1     PRE_BUFFER                                   ! Pre-prompt buffer
1     ) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This procedure adds the necessary carriagecontrol to the user-specified
1 !       prompt string and stores it in an item-list descriptor.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       ret_status.wlc.v = SMG$$SET_PROMPT_STRING (prompt_string.rt.dx,
1 !                                                  item_list.ww.r,
1 !                                                  pre_buf_len.rl.v,
1 !                                                  pre_buffer.ra.r)
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       PROMPT_STRING   The descriptor of the prompt string.
1 !
1 !       ITEM_LIST       The QIO item-list entry for the prompt-string.
1 !
1 !       PRE_BUF_LEN     Length of buffer to insert before the prompt
1 !                       string.  (Used when interfacing to virtual display.)
1 !
1 !       PRE_BUFFER      Address of string to insert before the prompt
1 !                       string.  (Used when interfacing to virtual display.)
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       QPS_QUEUE       Queue of prompt strings
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! COMPLETION STATUS:
1 !
1 !       SS$_NORMAL      Normal successful completion
1 !       Errors from LIB$GET_VM
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None
1 !
1 !--
1
2   BEGIN
2
2   LOCAL
2       STRING_LEN: WORD,               ! Length of PROMPT_STRING
2       STRING_PTR,                     ! Pointer to PROMPT_STRING
```

```
	    ALLOC_LEN,				! Allocated string length
	    ALLOC_PTR,				! Pointer to allocated string
	    ANL_STATUS, VM_STATUS;		! Status from called procedures

	LITERAL
	    K_CR = 13;				! <CR>

	BUILTIN
	    REMQUE;

	!+
	! Analyze PROMPT_STRING to get the length and address.
	!-

	ANL_STATUS = LIB$ANALYZE_SDESC_R2 (PROMPT_STRING [0,0,0,0];
	    STRING_LEN, STRING_PTR);
	IF NOT .ANL_STATUS
	THEN
	    RETURN .ANL_STATUS;

	ALLOC_LEN = .STRING_LEN + 1;
	IF .ALLOC_LEN GTRU 65535
	THEN
	    RETURN SMG$_PROTOOLON;	! Prompt string is too long
	ITEM_LIST [0,ITM$W_BUFSIZ] = .ALLOC_LEN;
	ALLOC_PTR = 0;

	!+
	! If ALLOC_LEN is small enough so that we can get the string from
	! our queue, do so.
	!-

	IF .ALLOC_LEN LEQU (K_QPS_SIZE - 10)
	THEN
	    BEGIN
	    IF REMQUE (.QPS_QUEUE [0], ALLOC_PTR)
	    THEN
		BEGIN
		ALLOC_PTR = 0;		! Failure
		ALLOC_LEN = K_QPS_SIZE;	! Allocate a full QPS entry.
		END;
	    END;

	!+
	! If pointer is still zero, we have to allocate the string with LIB$GET_VM.
	!-

	IF .ALLOC_PTR EQL 0
	THEN
	    BEGIN
	    VM_STATUS = LIB$GET_VM (ALLOC_LEN, ITEM_LIST [0,ITM$L_BUFADR]);
	    IF NOT .VM_STATUS
	    THEN
		RETURN .VM_STATUS;
	    ALLOC_PTR = .ITEM_LIST [0,ITM$L_BUFADR];
	    END;
```

```
!+
! Store the prompt string with a leading CR for carriagecontrol.
!
! If input is interfacing with output SMG$ routines (ie. a virtual
! display present), use the caller's buffer instead of CR.  This
! buffer should contain the set cursor sequence to establish the
! desired cursor position.
!-

ITEM_LIST [0,ITM$L_BUFADR] = .ALLOC_PTR;
IF .PRE_BUF_LEN EQL 0
THEN
    CH$WCHAR_A (K_CR, ALLOC_PTR)
ELSE
    BEGIN
    ITEM_LIST [0,ITM$W_BUFSIZ] = .STRING_LEN + .PRE_BUF_LEN;
    ALLOC_PTR = CH$MOVE (.PRE_BUF_LEN, .PRE_BUFFER, .ALLOC_PTR);
    END;
CH$MOVE (.STRING_LEN, .STRING_PTR, .ALLOC_PTR);
RETURN SS$_NORMAL;

END;                                    ! End of routine SMG$$SET_PROMPT_STRING
```

```
                                   00FC 00000 SMG$$SET_PROMPT_STRING:
                                                      .WORD   Save R2,R3,R4,R5,R6,R7              ; 2850
            5E             04   C2 00002              SUBL2   #4, SP                              ;
            50        04   AC   D0 00005              MOVL    PROMPT_STRING, R0                   ; 2920
              00000000G    00   16 00009              JSB     LIB$ANALYZE_SDESC_R2                ;
            56             51   7D 0000F              MOVQ    R1, STRING_LEN                      ;
            71             50   E9 00012              BLBC    ANL_STATUS, 6$                      ; 2922
            6E             56   3C 00015              MOVZWL  STRING_LEN, ALLOC_LEN               ; 2926
                           6E   D6 00018              INCL    ALLOC_LEN                           ;
      0000FFFF    8F       6E   D1 0001A              CMPL    ALLOC_LEN, #65535                   ; 2927
                           08   1B 00021              BLEQU   1$                                  ;
         50 00000000G      8F   D0 00023              MOVL    #SMG$_PROTOOLON, R0                 ; 2929
                           04      0002A              RET                                         ;
           08     BC       6E   B0 0002B 1$:          MOVW    ALLOC_LEN, @ITEM_LIST               ; 2930
                           53   D4 0002F              CLRL    ALLOC_PTR                           ; 2931
      00000050    8F       6E   D1 00031              CMPL    ALLOC_LEN, #80                      ; 2938
                           0F   1A 00038              BGTRU   2$                                  ;
         53 00000000'      FF   0F 0003A              REMQUE  @QPS_QUEUE, ALLOC_PTR               ; 2941
                           06   1C 00041              BVC     2$                                  ;
                           53   D4 00043              CLRL    ALLOC_PTR                           ; 2944
            6E        5A   8F   9A 00045              MOVZBL  #90, ALLOC_LEN                      ; 2945
                           53   D5 00049 2$:          TSTL    ALLOC_PTR                           ; 2953
                           18   12 0004B              BNEQ    3$                                  ;
            52        08   AC   D0 0004D              MOVL    ITEM_LIST, R2                       ; 2956
                      04   A2   9F 00051              PUSHAB  4(R2)                               ;
                      04   AE   9F 00054              PUSHAB  ALLOC_LEN                           ;
      00000000G   00       02   FB 00057              CALLS   #2, LIB$GET_VM                      ;
            25             50   E9 0005E              BLBC    VM_STATUS, 6$                       ; 2957
            53        04   A2   D0 00061              MOVL    4(R2), ALLOC_PTR                    ; 2960
            50        08   AC   7D 00065 3$:          MOVQ    ITEM_LIST, R0                       ; 2972
```

```
                  04    A0          53 D0 00069        MOVL    ALLOC_PTR, 4(R0)                              ;
                                    51 D5 0006D        TSTL    R1                                            ; 2973
                                    05 12 0006F        BNEQ    4$                                            ;
                        83          0D 90 00071        MOVB    #13, (ALLOC_PTR)+                             ; 2975
                                    09 11 00074        BRB     5$                                            ;
            60          56          51 A1 00076 4$:    ADDW3   R1, STRING_LEN, (R0)                          ; 2978
            63    10    BC          51 28 0007A        MOVC3   R1, @PRE_BUFFER, (ALLOC_PTR)                  ; 2979
            63          67          56 28 0007F 5$:    MOVC3   STRING_LEN, (STRING_PTR), (ALLOC_PTR)         ; 2981
                        50          01 D0 00083        MOVL    #1, R0                                        ; 2982
                                       04 00086 6$:    RET                                                   ; 2984

; Routine Size:  135 bytes,    Routine Base:  _SMG$CODE + 0EF4
```

```
%SBTTL 'SMG$$DELETE_PROMPT_STRING'
ROUTINE SMG$$DELETE_PROMPT_STRING (
    ITEM_LIST: REF $ITMLST_DECL (ITEMS=1)       ! Prompt string item list
    ): NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This procedure deletes a prompt string, if any.
!
! CALLING SEQUENCE:
!
!       SMG$$DELETE_PROMPT_STRING (item_list.rr.r)
!
! FORMAL PARAMETERS:
!
!       ITEM_LIST       The QIO item-list entry for the prompt string.
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       QPS_QUEUE
!
! COMPLETION STATUS:
!
!       NONE
!
! SIDE EFFECTS:
!
!       Either inserts string on queue of prompt-strings or deallocates
!       the string.
!--

    BEGIN

    LOCAL
        ALLOC_LEN;

    BUILTIN
        INSQUE;

    !+
    ! If the string length is LEQ K_QPS_SIZE, insert it on the queue,
    ! as the allocated length is guaranteed to be at least K_QPS_SIZE.
    !-

    ALLOC_LEN = .ITEM_LIST [0,ITM$W_BUFSIZ];
    IF .ALLOC_LEN LEQU K_QPS_SIZE
    THEN
        INSQUE (.ITEM_LIST [0,ITM$L_BUFADR], QPS_QUEUE)
    ELSE
        LIB$FREE_VM (ALLOC_LEN, ITEM_LIST [0,ITM$L_BUFADR]);
```

```
; 2974          3042  2     RETURN;
; 2975          3043  2
; 2976          3044  1     END;                                    ! End of routine SMG$$DELETE_PROMPT_STRING



                                                     0000 00000 SMG$$DELETE_PROMPT_STRING:
                                                                        .WORD   Save nothing                          ; 2986
                       7E         04    BC    3C 00002                   MOVZWL  @ITEM_LIST, ALLOC_LEN                 ; 3035
                       50         04    AC    D0 00006                   MOVL    ITEM_LIST, R0                         ; 3038
           0000005A    8F               6E    D1 0000A                   CMPL    ALLOC_LEN, #90                        ; 3036
                                        09    1A 00011                   BGTRU   1$                                    ;
           00000000'   EF         04    B0    0E 00013                   INSQUE  @4(R0), QPS_QUEUE                     ; 3038
                                              04 0001B                   RET                                           ;
                                  04    A0    9F 0001C 1$:               PUSHAB  4(R0)                                 ; 3040
                                  04    AE    9F 0001F                   PUSHAB  ALLOC_LEN                             ;
           00000000G   00               02    FB 00022                   CALLS   #2, LIB$FREE_VM                       ;
                                              04 00029                   RET                                           ; 3044

; Routine Size:  42 bytes,    Routine Base:  _SMG$CODE + 0F7B
```

```
1 %SBTTL 'SMG$$INIT_QUEUES'
1 ROUTINE SMG$$INIT_QUEUES
1     =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This procedure initializes KCB_QUEUE and QPS_QUEUE, and declares
1 !       an exit handler.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       ret_status.wlc.v = SMG$$INIT_QUEUES ()
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       V_QUEUE_INIT
1 !       KCB_QUEUE
1 !       QPS_QUEUE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       V_QUEUE_INIT set to 1
1 !       KCB_QUEUE initialized as an empty queue
1 !       QPS_QUEUE initialized as an empty queue
1 !
1 ! COMPLETION STATUS:
1 !
1 !       SS$_NORMAL      Normal successful completion
1 !       Errors from LIB$GET_VM or $DCLEXH
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Declares SMG$$INPUT_EXIT_HANDLER as an exit handler
1 !
1 !--
1
2     BEGIN
2
2     LOCAL
2         AST_STATUS,
2         VM_STATUS,
2         V_ENABLE_EXITH, ! Set to 1 if exit handler should be enabled
2         EXIT_BLOCK: REF VECTOR [4, LONG],       ! Exit handler block
2         EB_PTR;         ! Pointer to allocated memory
2
2     BUILTIN
2         TESTBITCS;
2
2     !+
2     ! Assume that we don't have to enable the exit handler.
2     !-
2
```

```
V_ENABLE_EXITH = 0;

!+
! Disable ASTs
!-

AST_STATUS = $SETAST (ENBFLG=0);

!+
! If V_QUEUE_INIT still clear, initialize the queue.
!-

IF TESTBITCS (V_QUEUE_INIT)
THEN
    BEGIN

    !+
    ! Initialize KQB_QUEUE to be an empty queue.
    !-

    KQB_QUEUE [0] = KQB_QUEUE;        ! Set forward link
    KQB_QUEUE [1] = KQB_QUEUE;        ! Set backwards link

    !+
    ! Initialize QPS_QUEUE to be an empty queue.
    !-

    QPS_QUEUE [0] = QPS_QUEUE;        ! Set forward link
    QPS_QUEUE [1] = QPS_QUEUE;        ! Set backwards link

    !+
    ! Remember that we want to declare the exit handler.
    !-

    V_ENABLE_EXITH = 1;

    END;

!+
! Reenable ASTs if previously enabled.
!-

IF .AST_STATUS EQLU SS$_WASSET
THEN
    $SETAST (ENBFLG=1);

!+
! If we need to enable the exit handler, do it now.  It is not
! harmful if other AST-level operations go on while we are doing this.
!-

IF .V_ENABLE_EXITH
THEN
    BEGIN

    VM_STATUS = LIB$GET_VM (%REF(20), EB_PTR);
    IF NOT .VM_STATUS
```

```
; 3092          3159  3            THEN
; 3093          3160  3                RETURN .VM_STATUS;
; 3094          3161  3            EXIT_BLOCK = .EB_PTR;
; 3095          3162  3            EXIT_BLOCK [1] = SMG$$INPUT_EXIT_HANDLER;         ! Handler address
; 3096          3163  3            EXIT_BLOCK [2] = 1;        ! 1 argument
; 3097          3164  3            EXIT_BLOCK [3] = EXIT_BLOCK [4];          ! Address to store reason
; 3098          3165  3            $DCLEXH (DESBLK = EXIT_BLOCK [0]);
; 3099          3166  2            END;
; 3100          3167  2
; 3101          3168  2        RETURN SS$_NORMAL;
; 3102          3169  2
; 3103          3170  1        END;                                  ! End of routine SMG$$INIT_KQB_QUEUE


                                                           .EXTRN  SYS$SETAST, SYS$DCLEXH

                                       001C 00000 SMG$$INIT_QUEUES:
                                                           .WORD   Save R2,R3,R4                           ; 3046
                      54 00000000G   00  9E 00002          MOVAB   SYS$SETAST, R4                          ;
                      53 00000000'   EF  9E 00009          MOVAB   KQB_QUEUE, R3                           ;
                      5E             08  C2 00010          SUBL2   #8, SP                                  ;
                                     52  D4 00013          CLRL    V_ENABLE_EXITH                          ; 3102
                                     7E  D4 00015          CLRL    -(SP)                                   ; 3108
                      64             01  FB 00017          CALLS   #1, SYS$SETAST                          ;
            14     10 A3             00  E2 0001A          BBSS    #0, V_QUEUE_INIT, 1$                    ; 3114
                      63             63  9E 0001F          MOVAB   KQB_QUEUE, KQB_QUEUE                    ; 3122
                   04 A3             63  9E 00022          MOVAB   KQB_QUEUE, KQB_QUEUE+4                  ; 3123
                   08 A3          08 A3  9E 00026          MOVAB   QPS_QUEUE, QPS_QUEUE                    ; 3129
                   0C A3          08 A3  9E 0002B          MOVAB   QPS_QUEUE, QPS_QUEUE+4                  ; 3130
                      52             01  D0 00030          MOVL    #1, V_ENABLE_EXITH                      ; 3136
                      09             50  D1 00033 1$:      CMPL    AST_STATUS, #9                          ; 3144
                      05             12  00036             BNEQ    2$                                      ;
                                     01  DD 00038          PUSHL   #1                                      ; 3146
                      64             01  FB 0003A          CALLS   #1, SYS$SETAST                          ;
                      30             52  E9 0003D 2$:      BLBC    V_ENABLE_EXITH, 3$                      ; 3153
                                  04 AE  9F 00040          PUSHAB  EB_PTR                                  ; 3157
                   04 AE             14  D0 00043          MOVL    #20, 4(SP)                              ;
                                  04 AE  9F 00047          PUSHAB  4(SP)                                   ;
              00000000G 00           02  FB 0004A          CALLS   #2, LIB$GET_VM                          ;
                      1F             50  E9 00051          BLBC    VM_STATUS, 4$                           ; 3158
                      50          04 AE  D0 00054          MOVL    EB_PTR, EXIT_BLOCK                      ; 3161
                   04 A0       0000V CF  9E 00058          MOVAB   SMG$$INPUT_EXIT_HANDLER, 4(EXIT_BLOCK)  ; 3162
                   08 A0             01  D0 0005E          MOVL    #1, 8(EXIT_BLOCK)                       ; 3163
                   0C A0          10 A0  9E 00062          MOVAB   16(R0), 12(EXIT_BLOCK)                  ; 3164
                                     50  DD 00067          PUSHL   EXIT_BLOCK                              ; 3165
              00000000G 00           01  FB 00069          CALLS   #1, SYS$DCLEXH                          ;
                      50             01  D0 00070 3$:      MOVL    #1, R0                                  ; 3168
                                     04  00073 4$:         RET                                             ; 3170

; Routine Size:  116 bytes,    Routine Base:  _SMG$CODE + 0FA5
```

```
1 %SBTTL 'SMG$$CLEANUP'
1 ROUTINE SMG$$CLEANUP (
1     KCB: REF KCB_R_KCB_STRUCT,           ! Keyboard control block
1     RET_STATUS                           ! Return status
1     ): SMG$$CLEANUP$LNK =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This procedure performs all the necessary actions to delete
1 !       a virtual keyboard.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       ret_status.wlc.v = SMG$$CLEANUP (KCB.mr.r, RET_STATUS.rlc.v)
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       KCB (R8)        - The address of the Keyboard Control Block being
1 !                         deleted.
1 !
1 !       RET_STATUS (R0) - A status which is to be returned by
1 !                         SMG$$CLEANUP.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! COMPLETION STATUS:
1 !
1 !       The value of the RET_STATUS parameter is returned in R0.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Deletes the virtual keyboard.
1 !
1 !--
1
2     BEGIN
2
2     LOCAL
2         KQB: REF KQB_R_KQB_STRUCT;       ! Keyboard Queue Block
2
2     BUILTIN
2         CALLG;
2
2     !+
2     ! Remove this KCB from the list of KCBs.
2     !-
2
2     KQB = .KCB [KCB_A_KQB];       ! Get Keyboard Queue Block pointer
2     IF .KQB NEQA 0
2     THEN
```

```
	KQB [KQB_A_KCB] = 0;

    !+
    ! Split for RMS vs. $QIO
    !-

    IF .KCB [KCB_V_RMS]
    THEN
	BEGIN
	LOCAL
	    FAB: $FAB_DECL;

	!+
	! Set up new FAB from saved IFI.
	!-

	$FAB_INIT (FAB=FAB);
	FAB [FAB$W_IFI] = KCB [KCB_W_IFI];

	!+
	! Close file if opened.
	!-

	IF .FAB [FAB$W_IFI] NEQ 0
	THEN
	    $CLOSE (FAB=FAB);

	!+
	! Deallocate record buffer if allocated.
	!-

	IF .KCB [RAB$L_UBF] NEQA 0
	THEN
	    LIB$FREE_VM (%REF(.KCB [RAB$W_USZ]), KCB [RAB$L_UBF]);

	END
    ELSE
	BEGIN

	BIND
	    DEVDEPEND2 = KCB [KCB_L_DEVDEPEND2]: BLOCK [4, BYTE];

	!+
	! If we have changed the terminal characteristics and keypad,
	! restore them.
	!-

	IF .KCB [KCB_V_CHARS_CHANGED]
	THEN
	    BEGIN

	    !+
	    ! Restore old DEVDEPEND2 information.
	    !-

	    KCB [KCB_L_DEVDEPEND2] = .KCB [KCB_L_OLD_DEVDEPEND2];

```

```
                !+
                ! Reset keypad mode.
                !-

                SMG$SET_KEYPAD_MODE (%REF(KCB [KCB_R_KCB]),
                    %REF(.DEVDEPEND2 [TT2$V_APP_KEYPAD]));
                END;

            !+
            ! Deassign channel if assigned.
            !-

            IF .KCB [KCB_W_CHANNEL] NEQ 0
            THEN
                $DASSGN (CHAN = .KCB [KCB_W_CHANNEL]);
            END;

        !+
        ! Free event flag if allocated.
        !-

        IF .KCB [KCB_L_EFN] NEQ 0
        THEN
            LIB$FREE_EF (KCB [KCB_L_EFN]);

        !+
        ! Deallocate KCB.
        !-

        LIB$FREE_VM (%REF(KCB_S_KCB_STRUCT), %REF (.KCB - KCB_K_ORIGIN_OFFSET));

        RETURN .RET_STATUS;

        END;                                            ! End of routine SMG$$CLEANUP


                                                         .EXTRN  SYS$CLOSE, SYS$DASSGN

                                 00FC 00000 SMG$$CLEANUP:
                                                         .WORD   Save R2,R3,R4,R5,R6,R7                    ; 3172
                  57 00000000G  00  9E 00002             MOVAB   LIB$FREE_VM, R7                           ;
                  5E        A8  AE  9E 00009             MOVAB   -88(SP), SP                               ;
                  56            50  D0 0000D             MOVL    R0, R6                                    ;
                  50        D8  A8  D0 00010             MOVL    -40(KCB), KQB                             ; 3225
                                03  13 00014             BEQL    1$                                        ; 3226
                            08  A0  D4 00016             CLRL    8(KQB)                                    ; 3228
                  40        FC  A8  E9 00019 1$:         BLBC    -4(KCB), 3$                               ; 3234
  0050   8F       00   6E       00  2C 0001D             MOVC5   #0, (SP), #0, #80, $RMS_PTR               ; 3244
                            08  AE     00024
              08  AE      5003  8F  B0 00026             MOVW    #20483, $RMS_PTR                          ;
              1E  AE            02  90 0002C             MOVB    #2, $RMS_PTR+22                           ;
              27  AE            02  90 00030             MOVB    #2, $RMS_PTR+31                           ;
                  50        D4  A8  9E 00034             MOVAB   -44(KCB), R0                              ; 3245
              0A  AE            50  B0 00038             MOVW    R0, FAB+2                                 ;
                                0A  13 0003C             BEQL    2$                                        ; 3251
                            08  AE  9F 0003E             PUSHAB  FAB                                       ; 3253
```

```
                     00000000G  00              01  FB 00041      CALLS   #1, SYS$CLOSE                    ;
                                         24     A8  D5 00048 2$:  TSTL    36(KCB)                          ; 3259
                                                40  13 0004B      BEQL    5$                               ;
                                         24     A8  9F 0004D      PUSHAB  36(KCB)                          ; 3261
                           08   AE       20     A8  3C 00050      MOVZWL  32(KCB), 8(SP)                   ;
                                         08     AE  9F 00055      PUSHAB  8(SP)                            ;
                                67              02  FB 00058      CALLS   #2, LIB$FREE_VM                  ;
                                                30  11 0005B      BRB     5$                               ; 3234
                   1B      FC   A8              02  E1 0005D 3$:  BBC     #2, -4(KCB), 4$                  ; 3275
                           F4   A8       E4     A8  D0 00062      MOVL    -28(KCB), -12(KCB)               ; 3283
   04   AE     F6   A8          01              07  EF 00067      EXTZV   #7, #1, -10(KCB), 4(SP)          ; 3290
                                         04     AE  9F 0006E      PUSHAB  4(SP)                            ;
                           04   AE              58  D0 00071      MOVL    KCB, 4(SP)                       ; 3289
                                         04     AE  9F 00075      PUSHAB  4(SP)                            ;
                         FD9F   CF              02  FB 00078      CALLS   #2, SMG$SET_KEYPAD_MODE          ;
                                         D4     A8  B5 0007D 4$:  TSTW    -44(KCB)                         ; 3297
                                                0B  13 00080      BEQL    5$                               ;
                                7E       D4     A8  32 00082      CVTWL   -44(KCB), -(SP)                  ; 3299
                     00000000G  00              01  FB 00086      CALLS   #1, SYS$DASSGN                   ;
                                         D0     A8  D5 0008D 5$:  TSTL    -48(KCB)                         ; 3306
                                                0A  13 00090      BEQL    6$                               ;
                                         D0     A8  9F 00092      PUSHAB  -48(KCB)                         ; 3308
                     00000000G  00              01  FB 00095      CALLS   #1, LIB$FREE_EF                  ;
                           04   AE       D0     A8  9E 0009C 6$:  MOVAB   -48(R8), 4(SP)                   ; 3314
                                         04     AE  9F 000A1      PUSHAB  4(SP)                            ;
                           04   AE       E2     8F  9A 000A4      MOVZBL  #226, 4(SP)                      ;
                                         04     AE  9F 000A9      PUSHAB  4(SP)                            ;
                                67              02  FB 000AC      CALLS   #2, LIB$FREE_VM                  ;
                                50              56  D0 000AF      MOVL    RET_STATUS, R0                   ; 3316
                                                    04 000B2      RET                                      ; 3318

; Routine Size:  179 bytes,    Routine Base:  _SMG$CODE + 1019
```

```
%SBTTL 'SMG$$INPUT_EXIT_HANDLER'
ROUTINE SMG$$INPUT_EXIT_HANDLER
    : NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This procedure is the exit handler for the Screen Management
!       input procedures.  It scans the list of Keyboard Queue Blocks
!       looking for those which refer to existing KCBs.  Those KCBs
!       are then deleted.
!
! CALLING SEQUENCE:
!
!       Called by VMS upon image exit
!
! FORMAL PARAMETERS:
!
!       None used
!
! IMPLICIT INPUTS:
!
!       KQB_QUEUE
!
! IMPLICIT OUTPUTS:
!
!       V_KQB_QUEUE_INIT set to zero
!
! COMPLETION STATUS:
!
!       NONE
!
! SIDE EFFECTS:
!
!       Deletes all existant virtual keyboards.
!
!--

    BEGIN

    LOCAL
        KQB: REF KQB_R_KQB_STRUCT,          ! Keyboard Queue Block
        KCB: REF KCB_R_KCB_STRUCT,          ! Keyboard Control Block
        QUEUE_HEAD_ADR: REF VECTOR [, LONG];    ! Address of queue head

    !+
    ! Get the first KQB from the head of the queue.  Note that the
    ! queue must have already been initialized.
    !-

    QUEUE_HEAD_ADR = KQB_QUEUE [0];
    KQB = .QUEUE_HEAD_ADR [0];

    !+
    ! While we haven't returned to the head of the queue, if this KQB
    ! has an active KCB, delete it.
    !-
```

```
; 3311          3376  2
; 3312          3377  2        WHILE (.KQB NEQA .QUEUE_HEAD_ADR) DO
; 3313          3378  3            BEGIN
; 3314          3379  3            KCB = .KQB [KQB_A_KCB];
; 3315          3380  3            IF .KCB NEQA 0
; 3316          3381  3            THEN
; 3317          3382  3                SMG$$CLEANUP (KCB [KCB_R_KCB], 0);
; 3318          3383  3            KQB = .KQB [KQB_A_FLINK];
; 3319          3384  2            END;
; 3320          3385  2
; 3321          3386  2        RETURN;
; 3322          3387  2
; 3323          3388  1        END;                                    ! End of routine SMG$$INPUT_EXIT_HANDLER


                                     010C 00000 SMG$$INPUT_EXIT_HANDLER:
                                                       .WORD   Save R2,R3,R8                          ; 3320
            53 00000000'   EF    9E 00002              MOVAB   KQB_QUEUE, QUEUE_HEAD_ADR              ; 3369
            52             63    D0 00009              MOVL    (QUEUE_HEAD_ADR), KQB                  ; 3370
            53             52    D1 0000C 1$:          CMPL    KQB, QUEUE_HEAD_ADR                    ; 3377
                           12    13 0000F              BEQL    3$                                     ;
            58        08   A2    D0 00011              MOVL    8(KQB), KCB                            ; 3379
                           07    13 00015              BEQL    2$                                     ; 3380
                           50    D4 00017              CLRL    R0                                     ; 3382
        FF2F   CF          00    FB 00019              CALLS   #0, SMG$$CLEANUP                       ;
            52             62    D0 0001E 2$:          MOVL    (KQB), KQB                             ; 3383
                           E9    11 00021              BRB     1$                                     ; 3377
                                 04 00023 3$:          RET                                            ; 3388

; Routine Size:  36 bytes,    Routine Base:  _SMG$CODE + 10CC
```

```
; 3325          3389  1 END                                     ! End of module SMG$INPUT
; 3326          3390  1
; 3327          3391  0 ELUDOM


;                                       PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;   _SMG$DATA                       20  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;   _SMG$CODE                     4336  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)


;                               Library Statistics
;
;                                         -------- Symbols --------      Pages        Processing
;       File                              Total    Loaded   Percent      Mapped       Time
;
;   _$255$DUA28:[SYSLIB]STARLET.L32;1      9776       164         1         581        00:01.1
;   _$255$DUA28:[SMGRTL.OBJ]RTLLIB.L32;1     36         4        11           8        00:00.1
;   _$255$DUA28:[SMGRTL.OBJ]SMGLIB.L32;1    469       146        31          38        00:00.4


;                                       COMMAND QUALIFIERS

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

; Size:           4223 code + 133 data bytes
; Run Time:          01:34.3
; Elapsed Time:      05:12.7
; Lines/CPU Min:     2158
; Lexemes/CPU-Min: 21145
; Memory Used:   569 pages
; Compilation Complete
```

0358 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SMGKEYUTI
LIS

SMGKEYPAD
LIS

SMGINPUT
LIS

SMGLIB
LIS