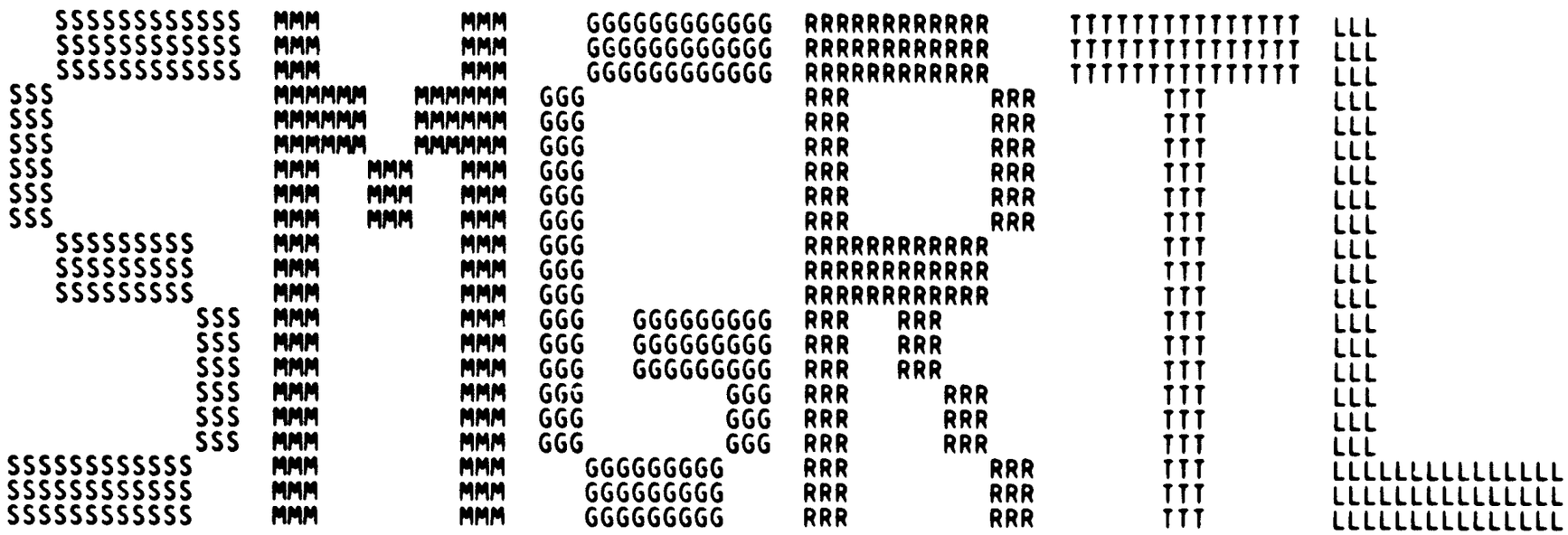

_$2

Val

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

FILEID**SMGINPUT

```
  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
```

```
;     1          0001  0 MODULE SMG$INPUT ( %TITLE 'Screen Management Facility Input Procedures'
;     2          0002  0                   IDENT = '2-017'          ! File: SMGINPUT.B32 Edit: STAN2017
;     3          0003  0                   ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !*****************************************************************************
;     7          0007  1 !*                                                                           *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                     *
;    11          0011  1 !*                                                                           *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    17          0017  1 !*  TRANSFERRED.                                                             *
;    18          0018  1 !*                                                                           *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    21          0021  1 !*  CORPORATION.                                                             *
;    22          0022  1 !*                                                                           *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*****************************************************************************
;    28          0028  1 !
;    29          0029  1
;    30          0030  1 !++
;    31          0031  1 ! FACILITY:     Screen Management Facility
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:
;    34          0034  1 !
;    35          0035  1 !       This module contains the portions of the VAX-11 Run-Time Library
;    36          0036  1 !       Screen Management Facility which deal with input.
;    37          0037  1 !
;    38          0038  1 ! ENVIRONMENT:  User mode - AST reentrant
;    39          0039  1 !
;    40          0040  1 ! AUTHOR: Steven B. Lionel, CREATION DATE: 10-Feb-1983
;    41          0041  1 !
;    42          0042  1 ! MODIFIED BY:
;    43          0043  1 !
;    44          0044  1 ! 1-001 - Original.  SBL 10-Feb-1983
;    45          0045  1 ! 1-002 - Add variant code to allow READ_STRING to work on a V3.n system.
;    46          0046  1 !         Fix READ_COMPOSED_LINE so that the equivalence string of the final
;    47          0047  1 !         key is remembered by the terminal-driver for line-recall.  Use
;    48          0048  1 !         a <CR> as a prompt if none is specified.
;    49          0049  1 !         SBL 18-May-1983
;    50          0050  1 ! 1-003 - Remember /LOCK'ed states across multiple reads.  Do "right" thing
;    51          0051  1 !         with ^Z, undefined keys in the middle of lines.  SBL 23-Jun-1983
;    52          0052  1 ! 1-004 - Prevent last echo in READ_COMPOSED_LINE unless there is
;    53          0053  1 !         something to echo.  SBL 28-Jul-1983
;    54          0054  1 ! p-003 - Modify SMG$CREATE_VIRTUAL_KEYBOARD and SMG$READ_STRING to
;    55          0055  1 !         interface with screen management output.  PLL 16-Jun-1983
;    56          0056  1 ! p-004 - More modifications for integration.  PLL 21-Jun-1983
;    57          0057  1 ! p-005 - SMG$$SET_PHYSICAL_CURSOR now takes a longword buffer length,
```

```
;    58          0058  1 !          instead of word.  PLL 28-Jun-1983
;    59          0059  1 ! p-006 - Fix a couple bugs in SMG$READ_STRING.  PLL 29-Jun-1983
;    60          0060  1 !           1.  Use default rendition for virtual display for prompt/input.
;    61          0061  1 !           2.  If at the boundary of the virtual display, then there is no
;    62          0062  1 !               room for input - issue an error.
;    63          0063  1 ! 2-001 - Combined Steves edits 1-003 and 1-004 with Pam's edits
;    64          0064  1 !          1-003 through 1-006 and added some stuff of my own.
;    65          0065  1 !          STAN 2-Aug-1983.
;    66          0066  1 ! 2-002 - FINISHED INPUT/OUTPUT INTEGRATION. STAN 3-Aug-1983.
;    67          0067  1 ! 2-003 - Bug fixes to SMG$READ_STRING for terminators and modifier
;    68          0068  1 !          masks.  PLL 9-Aug-1983
;    69          0069  1 ! 2-004 - Another attempt to get 3a modifiers recognized.  Also, pass the
;    70          0070  1 !          terminator to the smg$ output routines - the terminator may affect
;    71          0071  1 !          the cursor position.  PLL 16-Aug-1983
;    72          0072  1 ! 2-005 - In SMG$READ_STRING, don't report the change if NOECHO was requested.
;    73          0073  1 !          PLL 18-Aug-1983
;    74          0074  1 ! 2-006 - Fix last edit.  PLL 22-Aug-1983
;    75          0075  1 ! 2-007 - In SMG$READ_STRING, don't report the terminator if TRMNOECHO was
;    76          0076  1 !          requested.  PLL 9-Sep-1983
;    77          0077  1 ! 2-008 - Another fix to SMG$READ_STRING:  if a partial escape sequence
;    78          0078  1 !          terminated the read, do single character QIOs till we find the end
;    79          0079  1 !          of the terminator.  (This prevents keypad keys from echoing on the
;    80          0080  1 !          screen.)  PLL 13-Sep-1983
;    81          0081  1 ! 2-009 - Cleanup SMG$READ_COMPOSED_LINE.  PLL 5-Oct-1983
;    82          0082  1 ! 2-010 - GIVE ERROR if try to input to a batched display. STAN 14-Mar-1984.
;    83          0083  1 !          Fix length of buffers for renditions.
;    84          0084  1 !          Make renditions work.
;    85          0085  1 !          Change echo of CTRL/Z to track terminal driver change.
;    86          0086  1 ! 2-011 - Return terminator code of SMG$K_TRM_TIMEOUT on timeout. STAN
;    87          0087  1 ! 2-012 - Turn off renditions only if DCB was passed. STAN 28-MAR-1984.
;    88          0088  1 ! 2-013 - Add the codes for F1-F20 function keys, and editing keys to the code
;    89          0089  1 !          that checks for partial escapes.  PLL 2-Apr-1984
;    90          0090  1 ! 2-014 - Validate number of arguments. STAN 3-Jun-1984.
;    91          0091  1 !          Add F1-F20 code for READ COMPOSE LINE.
;    92          0092  1 ! 2-015 - Take CTRL/Z echo from system logical TTY$EXIT_STRING. STAN 7-Jul-1984.
;    93          0093  1 ! 2-016 - Fix to partial escape handling - after we find the end of the
;    94          0094  1 !          terminator, fix the return status to be success.  PLL 9-Jul-1984
;    95          0095  1 ! 2-017 - Remove call to TRNLNM on V3 systems. STAN 24-Jul-1984.
;    96          0096  1 !          Remove informational errors.
;    97          0097  1 !--
```

```
;   99     0098  1 %SBTTL 'Declarations'
;  100     0099  1 !
;  101     0100  1 ! PROLOGUE FILE:
;  102     0101  1 !
;  103     0102  1
;  104     0103  1 REQUIRE 'RTLIN:SMGPROLOG';                      . Screen management definitions
;  105     0181  1
;  106     0182  1 !
;  107     0183  1 ! TABLE OF CONTENTS:
;  108     0184  1 !
;  109     0185  1
;  110     0186  1 FORWARD ROUTINE
;  111     0187  1     SMG$CREATE_VIRTUAL_KEYBOARD,                ! Create a virtual keyboard
;  112     0188  1     SMG$DELETE_VIRTUAL_KEYBOARD,                ! Delete a virtual keyboard
;  113     0189  1     SMG$READ_STRING,                            ! Read a string
;  114     0190  1     SMG$READ_COMPOSED_LINE,                     ! Read a composed line
;  115     0191  1     SMG$SET_KEYPAD_MODE,                        ! Set mode of numeric keypad
;  116     0192  1     SMG$CANCEL_INPUT,                           ! Cancel outstanding input
;  117     0193  1     SMG$$SET_PROMPT_STRING,                     ! Create a prompt string
;  118     0194  1     SMG$$DELETE_PROMPT_STRING: NOVALUE,         ! Delete a prompt string
;  119     0195  1     SMG$$INIT_QUEUES,                           ! Initialize queues
;  120     0196  1     SMG$$CLEANUP: SMG$$CLEANUP$LNK,             ! Cleanup from error or close
;  121     0197  1     SMG$$INPUT_EXIT_HANDLER: NOVALUE;           ! Exit handler for input
;  122     0198  1
;  123     0199  1 !
;  124     0200  1 ! MACROS:
;  125     0201  1 !
;  126     0202  1 !       NONE
;  127     0203  1 !
;  128     0204  1 ! EQUATED SYMBOLS:
;  129     0205  1 !
;  130     0206  1
;  131     0207  1 LITERAL
;  132     0208  1     K_QPS_SIZE = 90;     ! Size of prompt strings on QPS_QUEUE.
;  133     0209  1                          ! 80 character bytes + possibly a 10 byte
;  134     0210  1                          ! set cursor sequence.
;  135     0211  1
;  136     0212  1 !
;  137     0213  1 ! FIELDS:
;  138     0214  1 !
;  139     0215  1 !       NONE
;  140     0216  1 !
;  141     0217  1 ! OWN STORAGE:
;  142     0218  1 !
;  143     0219  1
;  144     0220  1 !+
;  145     0221  1 ! Queues of control blocks and prompt strings.
;  146     0222  1 !-
;  147     0223  1
;  148     0224  1 OWN
;  149     0225  1     KQB_QUEUE: VOLATILE VECTOR [2, LONG],       ! Keyboard Queue Block
;  150     0226  1     QPS_QUEUE: VOLATILE VECTOR [2, LONG],       ! Queued Prompt Strings
;  151     0227  1     V_QUEUE_INIT: VOLATILE INITIAL (0); ! 1 if queues have been initialized
;  152     0228  1
;  153     0229  1 !+
;  154     0230  1 ! Declare strings which set the keypad into and out of applications mode.
;  155     0231  1 !-
```

```
;  156          0232  1
;  157          0233  1 OWN
;  158          0234  1     T_KPDAPP_DECCRT:    ! Sets DECCRT (and VT5x) into applications mode
;  159          0235  1         VECTOR [2, BYTE] PSECT (_SMG$CODE) INITIAL (BYTE (27,'=')),
;  160          0236  1     T_KPDNUM_DECCRT:    ! Sets DECCRT and (VT5x) into numeric mode
;  161          0237  1         VECTOR [2, BYTE] PSECT (_SMG$CODE) INITIAL (BYTE (27,'>'));
;  162          0238  1
;  163          0239  1 !
;  164          0240  1 !
;  165          0241  1 ! EXTERNAL DECLARATIONS:
;  166          0242  1 !
;  167          0243  1
;  168          0244  1 EXTERNAL ROUTINE
;  169          0245  1     LIB$GET_EF,                     ! Allocate local event flag number
;  170          0246  1     LIB$FREE_EF,                    ! Free local event flag number
;  171          0247  1     LIB$GET_VM,                     ! Allocate virtual memory
;  172          0248  1     LIB$FREE_VM,                    ! Deallocate virtual memory
;  173          0249  1     LIB$ANALYZE_SDESC_R2: LIB$ANALYZE_SDESC_R2$LINKAGE, ! Analyze string desc
;  174          0250  1     LIB$SCOPY_R_DX,                 ! Copy string
;  175          0251  1     LIB$SCOPY_R_DX6: LIB$SCOPY_R_DX6$LINKAGE,   ! Copy string
;  176          0252  1     SMG$$TERM_TO_KEYCODE,           ! Translate terminator to keycode
;  177          0253  1     SMG$$LOOKUP_KEY: SMG$$LOOKUP_KEY$LNK,       ! Lookup item in tree
;  178          0254  1     SMG$$GET_PASTEBOARD_ID,         ! SMG$ I/O interface routine
;  179          0255  1     SMG$$SET_PHYSICAL_CURSOR,       !  ''
;  180          0256  1     SMG$$REPORT_CHANGE_REPLACE,     !  ''
;  181          0257  1     SMG$$OUTPUT,                    ! Output directly to terminal
;  182          0258  1     SMG$$SET_ATTRIBUTES_ON,         ! Set attributes on
;  183          0259  1     SMG$$SET_ATTRIBUTES_OFF,        ! Set attributes off
;  184          0260  1     SYS$QIOW;                       ! $QIOW system service
;  185          0261  1
;  186          0262  1 EXTERNAL LITERAL
;  187          0263  1     SMG$_STRTERESC,         ! String terminated by escape
;  188          0264  1     SMG$_EOF,               ! End-of-file
;  189          0265  1     SMG$_FILTOOLON,         ! File-specification is too long (over 255 characters)
;  190          0266  1     SMG$_ILLBATFNC,         ! Operation not permitted while display is batched
;  191          0267  1     SMG$_INVMAXLEN,         ! Invalid maximum length (greater than 512 characters)
;  192          0268  1     SMG$_INVSTANAM,         ! Invalid state name
;  193          0269  1     SMG$_PROTOOLON,         ! Prompt string is too long
;  194          0270  1     SMG$_INVCOL,            ! Invalid column
;  195          0271  1     SMG$_INVDIS_ID;         ! Invalid display id
;  196          0272  1
;  197          0273  1 EXTERNAL
;  198          0274  1     LIB$AB_UPCASE;          ! Upcase translate table
```

```
;  200     0275  1 %SBTTL 'SMG$CREATE_VIRTUAL_KEYBOARD'
;  201     0276  1 GLOBAL ROUTINE SMG$CREATE_VIRTUAL_KEYBOARD (
;  202     0277  1     KEYBOARD_ID: REF VECTOR [, LONG],
;  203     0278  1     FILESPEC: REF BLOCK [, BYTE],
;  204     0279  1     DEFAULT_FILESPEC: REF BLOCK [, BYTE],
;  205     0280  1     RESULTANT_FILESPEC: REF BLOCK [, BYTE]
;  206     0281  1     ) =
;  207     0282  1
;  208     0283  1 !++
;  209     0284  1 ! FUNCTIONAL DESCRIPTION:
;  210     0285  1 !
;  211     0286  1 !       This procedure creates the association between a filespec (terminal
;  212     0287  1 !       name or RMS file) and a virtual-keyboard-identifier.  The keyboard-id
;  213     0288  1 !       is passed to other screen management procedures to uniquely identify
;  214     0289  1 !       the input stream being acted upon.
;  215     0290  1 !
;  216     0291  1 !       If "filespec" does not refer to a terminal, the file is opened using
;  217     0292  1 !       RMS and all further access to that file is performed through RMS.  If
;  218     0293  1 !       "filespec" is a terminal, a channel is assigned to the terminal, its
;  219     0294  1 !       keypad is set to "application mode" (if supported), and line editing
;  220     0295  1 !       is enabled.  These attributes are restored to their previous values
;  221     0296  1 !       when the virtual-keyboard is deleted, either by a call to
;  222     0297  1 !       SMG$DELETE_VIRTUAL_KEYBOARD or automatically when the image exits.
;  223     0298  1 !
;  224     0299  1 ! CALLING SEQUENCE:
;  225     0300  1 !
;  226     0301  1 !       RET_STATUS.wlc.v = SMG$CREATE_VIRTUAL_KEYBOARD (
;  227     0302  1 !                                  KEYBOARD-ID.wl.r
;  228     0303  1 !                                  [, [FILESPEC.rt.dx]
;  229     0304  1 !                                  [, [DEFAULT_FILESPEC.rt.dx]
;  230     0305  1 !                                  [, [RESULTANT_FILESPEC.wt.dx] ]]])
;  231     0306  1 !
;  232     0307  1 ! FORMAL PARAMETERS:
;  233     0308  1 !
;  234     0309  1 !       KEYBOARD-ID     A longword into which will be written the
;  235     0310  1 !                       virtual-keyboard-id of the created virtual-keyboard.
;  236     0311  1 !
;  237     0312  1 !       FILESPEC        A string containing the file specification of the
;  238     0313  1 !                       file or terminal to be used for this virtual-keyboard.
;  239     0314  1 !                       If omitted, 'SYS$INPUT:' is used.
;  240     0315  1 !
;  241     0316  1 !       DEFAULT_FILESPEC A string containing the default file specification.
;  242     0317  1 !                       If omitted, the null string is used.
;  243     0318  1 !
;  244     0319  1 !       RESULTANT_FILESPEC A string into which is written the expanded or
;  245     0320  1 !                       resultant file specification of the file used.
;  246     0321  1 !
;  247     0322  1 ! IMPLICIT INPUTS:
;  248     0323  1 !
;  249     0324  1 !       NONE
;  250     0325  1 !
;  251     0326  1 ! IMPLICIT OUTPUTS:
;  252     0327  1 !
;  253     0328  1 !       NONE
;  254     0329  1 !
;  255     0330  1 ! COMPLETION STATUS:
;  256     0331  1 !
```

```
;  257          0332  1 !       SS$_NORMAL      Normal successful completion
;  258          0333  1 !       SMG$_FILTOOLON  File-specification is too long (over 255 characters)
;  259          0334  1 !       SMG$_WRONUMARG  Wrong number of arguments
;  260          0335  1 !       LIB$_INSEF      Insufficient event flags
;  261          0336  1 !       LIB$_INSVIRMEM  Insufficient virtual memory
;  262          0337  1 !       LIB$_INVSTRDES  Invalid string descriptor
;  263          0338  1 !       RMS$_xyz        Any error possible from $OPEN or $CONNECT
;  264          0339  1 !       SS$_xyz         Any error possible from $GETDVIW, $ASSIGN or $DCLEXH
;  265          0340  1 !
;  266          0341  1 ! SIDE EFFECTS:
;  267          0342  1 !
;  268          0343  1 !       Opens file using RMS or assigns a channel to it.
;  269          0344  1 !       Declares an exit handler
;  270          0345  1 !       Sets terminal to keypad application mode.
;  271          0346  1 !
;  272          0347  1 !--
;  273          0348  1
;  274          0349  2     BEGIN
;  275          0350  2
;  276          0351  2     LOCAL
;  277          0352  2         KCB: REF KCB_R_KCB_STRUCT,                  ! Keyboard Control Block
;  278          0353  2         FAB: $FAB_DECL,                             ! RMS File Attributes Block
;  279          0354  2         NAM: $NAM_DECL,                             ! RMS NAM block
;  280          0355  2         XAB_FHC: $XABFHC_DECL,                      ! File Header Char. XAB
;  281          0356  2         RESULTANT_STRING: VECTOR [NAM$C_MAXRSS, BYTE]; ! Resultant name string
;  282          0357  2
;  283          0358  2     BUILTIN
;  284          0359  2         CALLG,
;  285          0360  2         NULLPARAMETER;
;  286          0361  2
;  287          0362  2     !+
;  288          0363  2     ! Literals for argument positions.
;  289          0364  2     !-
;  290          0365  2
;  291          0366  2     LITERAL
;  292          0367  2         K_KEYBOARD_ID = 1,
;  293          0368  2         K_FILESPEC = 2,
;  294          0369  2         K_DEFAULT_FILESPEC = 3,
;  295          0370  2         K_RESULTANT_FILESPEC = 4;
;  296          0371  2
;  297          0372  2     $SMG$VALIDATE_ARGCOUNT(1,4);
;  298          0373  2
;  299          0374  2     !+
;  300          0375  2     ! Verify assumption that a RAB is not longer than the positive-offset
;  301          0376  2     ! portion of a Keyboard Control Block.
;  302          0377  2     !-
;  303          0378  2
;  304          0379  2     $ASSUME (RAB$C_BLN, LEQU, KCB_S_KCB_STRUCT - KCB_K_ORIGIN_OFFSET);
;  305          0380  2
;  306          0381  2     !+
;  307          0382  2     ! Initialize FAB, NAM  and XAB.
;  308          0383  2     !-
;  309          0384  2
;  310          0385  2     $FAB_INIT (FAB=FAB, NAM=NAM, XAB=XAB_FHC);
;  311        P 0386  2     $NAM_INIT (NAM=NAM, ESA=RESULTANT_STRING, ESS=NAM$C_MAXRSS,
;  312          0387  2         RSA=RESULTANT_STRING, RSS=NAM$C_MAXRSS);
;  313          0388  2     $XABFHC_INIT (XAB=XAB_FHC);
```

```
;  314     0389  2
;  315     0390  2      !+
;  316     0391  2      ! Get the file specification.
;  317     0392  2      !-
;  318     0393  2
;  319     0394  2      IF NOT NULLPARAMETER (K_FILESPEC)   ! FILESPEC present?
;  320     0395  2      THEN
;  321     0396  3          BEGIN
;  322     0397  3          LOCAL
;  323     0398  3              STRING_PTR,             ! Pointer to string
;  324     0399  3              STRING_LEN: WORD,       ! Length of string
;  325     0400  3              ANL_STATUS;             ! Status from LIB$ANALYZE_SDESC_R2
;  326     0401  3
;  327     0402  3          ANL_STATUS = LIB$ANALYZE_SDESC_R2 (FILESPEC [0,0,0,0];
;  328     0403  3              STRING_LEN, STRING_PTR);
;  329     0404  3          IF NOT .ANL_STATUS
;  330     0405  3          THEN
;  331     0406  3              RETURN .ANL_STATUS;
;  332     0407  3
;  333     0408  3          IF .STRING_LEN GTRU 255
;  334     0409  3          THEN
;  335     0410  3              RETURN SMG$_FILTOOLON;
;  336     0411  3          FAB [FAB$L_FNA] = .STRING_PTR;
;  337     0412  3          FAB [FAB$B_FNS] = .STRING_LEN;
;  338     0413  3          END
;  339     0414  2      ELSE
;  340     0415  3          BEGIN
;  341     0416  3          !+
;  342     0417  3          ! Use 'SYS$INPUT:'
;  343     0418  3          !-
;  344     0419  3          FAB [FAB$B_FNS] = %CHARCOUNT ('SYS$INPUT:');
;  345     0420  3          FAB [FAB$L_FNA] = UPLIT BYTE ('SYS$INPUT:');
;  346     0421  2          END;
;  347     0422  2
;  348     0423  2      !+
;  349     0424  2      ! Get default filespec, if any.
;  350     0425  2      !-
;  351     0426  2
;  352     0427  2      IF NOT NULLPARAMETER (K_DEFAULT_FILESPEC)
;  353     0428  2      THEN
;  354     0429  3          BEGIN
;  355     0430  3          LOCAL
;  356     0431  3              STRING_PTR,             ! Pointer to string
;  357     0432  3              STRING_LEN: WORD,       ! Length of string
;  358     0433  3              ANL_STATUS;             ! Status from LIB$ANALYZE_SDESC_R2
;  359     0434  3
;  360     0435  3          ANL_STATUS = LIB$ANALYZE_SDESC_R2 (DEFAULT_FILESPEC [0,0,0,0];
;  361     0436  3              STRING_LEN, STRING_PTR);
;  362     0437  3          IF NOT .ANL_STATUS
;  363     0438  3          THEN
;  364     0439  3              RETURN .ANL_STATUS;
;  365     0440  3
;  366     0441  3          IF .STRING_LEN GTRU 255
;  367     0442  3          THEN
;  368     0443  3              RETURN SMG$_FILTOOLON;
;  369     0444  3          FAB [FAB$L_DNA] = .STRING_PTR;
;  370     0445  3          FAB [FAB$B_DNS] = .STRING_LEN;
```

```
;  371     0446  2          END;
;  372     0447  2
;  373     0448  2      !+
;  374     0449  2      ! Parse the file specification to see whether or not it is
;  375     0450  2      ! a terminal.
;  376     0451  2      !-
;  377     0452  2
;  378     0453  3          BEGIN
;  379     0454  3          LOCAL
;  380     0455  3              PARSE_STATUS;
;  381     0456  3          PARSE_STATUS = $PARSE (FAB = FAB);
;  382     0457  3          IF NOT .PARSE_STATUS
;  383     0458  3          THEN
;  384     0459  3              RETURN .PARSE_STATUS;
;  385     0460  2          END;
;  386     0461  2
;  387     0462  2      !+
;  388     0463  2      ! If KQB_QUEUE and QPS_QUEUE have not yet been initialized, initialize them.
;  389     0464  2      ! SMG$$INIT_QUEUES also declares an exit handler.
;  390     0465  2      !-
;  391     0466  2
;  392     0467  2      IF NOT .V_QUEUE_INIT
;  393     0468  2      THEN
;  394     0469  3          BEGIN
;  395     0470  3          LOCAL
;  396     0471  3              INIT_STATUS;
;  397     0472  3          INIT_STATUS = SMG$$INIT_QUEUES ();
;  398     0473  3          IF NOT .INIT_STATUS
;  399     0474  3          THEN
;  400     0475  3              RETURN .INIT_STATUS;
;  401     0476  2          END;
;  402     0477  2
;  403     0478  2      !+
;  404     0479  2      ! Allocate a Keyboard Control Block
;  405     0480  2      !-
;  406     0481  2
;  407     0482  3          BEGIN
;  408     0483  3          LOCAL
;  409     0484  3              VM_POINTER,          ! Pointer to allocated memory
;  410     0485  3              VM_STATUS;           ! Status from LIB$GET_VM
;  411     0486  3          VM_STATUS = LIB$GET_VM (%REF(KCB_S_KCB_STRUCT), VM_POINTER);
;  412     0487  3          IF NOT .VM_STATUS
;  413     0488  3          THEN
;  414     0489  3              RETURN .VM_STATUS;
;  415     0490  3          CH$FILL (0, KCB_S_KCB_STRUCT, .VM_POINTER);
;  416     0491  3          KCB = .VM_POINTER + KCB_K_ORIGIN_OFFSET;
;  417     0492  2          END;
;  418     0493  2
;  419     0494  2      !+
;  420     0495  2      ! Allocate a Keyboard Queue Block and put this KCB on the queue.
;  421     0496  2      ! Note that KQBs are never removed from the queue; they are inactivated
;  422     0497  2      ! by having the KQB_A_KCB field set to zero.
;  423     0498  2      !-
;  424     0499  2
;  425     0500  3          BEGIN
;  426     0501  3          LOCAL
;  427     0502  3              KQB: REF KQB_R_KQB_STRUCT,  ! Keyboard Queue Block
```

```
;  428     0503  3              KQB_PTR,    ! Pointer to allocated block
;  429     0504  3              VM_STATUS;
;  430     0505  3          BUILTIN
;  431     0506  3              INSQUE;
;  432     0507  3
;  433     0508  3          VM_STATUS = LIB$GET_VM (%REF(KQB_S_KQB_STRUCT), KQB_PTR);
;  434     0509  3          IF NOT .VM_STATUS
;  435     0510  3          THEN
;  436     0511  3              RETURN SMG$$CLEANUP (KCB [KCB_R_KCB], .VM_STATUS);
;  437     0512  3          KQB = .KQB_PTR;
;  438     0513  3          KQB [KQB_A_KCB] = .KCB;
;  439     0514  3          KCB [KCB_A_KQB] = .KQB;
;  440     0515  3          INSQUE (KQB [KQB_Q_QUEUE_LINK], KQB_QUEUE);
;  441     0516  2          END;
;  442     0517  2
;  443     0518  2      !+
;  444     0519  2      ! See if this file is a terminal.
;  445     0520  2      !-
;  446     0521  2
;  447     0522  3          BEGIN
;  448     0523  3          BIND
;  449     0524  3              FAB_DEV = FAB [FAB$L_DEV]: BLOCK [4, BYTE];
;  450     0525  3          IF NOT .FAB_DEV [DEV$V_TRM]
;  451     0526  3          THEN
;  452     0527  3              KCB [KCB_V_RMS] = 1;        ! Not a terminal
;  453     0528  2          END;
;  454     0529  2
;  455     0530  2      !+
;  456     0531  2      ! Now split dpending on whether the file is a terminal or not.
;  457     0532  2      !-
;  458     0533  2
;  459     0534  2      IF NOT .KCB [KCB_V_RMS]
;  460     0535  2      THEN
;  461     0536  3          BEGIN
;  462     0537  3          !+
;  463     0538  3          ! The file is a terminal.  Assign a channel and use $QIOs.
;  464     0539  3          !-
;  465     0540  3
;  466     0541  3          LOCAL
;  467     0542  3              DEVNAME_DESCR: BLOCK [8, BYTE], ! Descriptor for device name
;  468     0543  3              ASSIGN_STATUS, QIO_STATUS1, QIO_STATUS2, QIO_STATUS3;
;  469     0544  3
;  470     0545  3          BIND
;  471     0546  3              KCB_DEVDEPEND1 = KCB [KCB_L_DEVDEPEND1]: BLOCK [4, BYTE],
;  472     0547  3              KCB_DEVDEPEND2 = KCB [KCB_L_DEVDEPEND2]: BLOCK [4, BYTE];
;  473     0548  3
;  474     0549  3          !+
;  475     0550  3          ! Fill in device name descriptor and assign a channel.
;  476     0551  3          !-
;  477     0552  3
;  478     0553  3          DEVNAME_DESCR [DSC$W_LENGTH] = .NAM [NAM$B_ESL];
;  479     0554  3          DEVNAME_DESCR [DSC$B_CLASS] = DSC$K_CLASS_S;
;  480     0555  3          DEVNAME_DESCR [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  481     0556  3          DEVNAME_DESCR [DSC$A_POINTER] = RESULTANT_STRING;
;  482   P 0557  3          ASSIGN_STATUS = $ASSIGN (
;  483   P 0558  3              DEVNAM = DEVNAME_DESCR,     ! Device name descriptor
;  484     0559  3              CHAN = KCB [KCB_Q_CHANNEL]);
```

```
;  485          0560  3          IF NOT .ASSIGN_STATUS
;  486          0561  3          THEN
;  487          0562  3              RETURN SMG$$CLEANUP (KCB [KCB_R_KCB], .ASSIGN_STATUS);
;  488          0563  3
;  489          0564  3          !+
;  490          0565  3          ! Allocate an event flag number.
;  491          0566  3          !-
;  492          0567  3
;  493          0568  4              BEGIN
;  494          0569  4              LOCAL
;  495          0570  4                  EF_STATUS;
;  496          0571  4              EF_STATUS = LIB$GET_EF (KCB [KCB_L_EFN]);
;  497          0572  4              IF NOT .EF_STATUS
;  498          0573  4              THEN
;  499          0574  4                  RETURN SMG$$CLEANUP (KCB [KCB_R_KCB], .EF_STATUS);
;  500          0575  3              END;
;  501          0576  3
;  502          0577  3          !+
;  503          0578  3          ! Fill in the QIO argument lists.
;  504          0579  3          !-
;  505          0580  3
;  506          0581  3          KCB [KCB_L_QIO1_ARGCNT] = KCB_K_QIO1_ARGCNT;
;  507          0582  3          KCB [KCB_L_QIO1_EFN] = .KCB [KCB_L_EFN];
;  508          0583  3          KCB [KCB_L_QIO1_CHAN] = .KCB [KCB_Q_CHANNEL];
;  509          0584  3          KCB [KCB_L_QIO1_FUNC] = IO$_READVBLK + IO$M_ESCAPE;
;  510          0585  3          KCB [KCB_A_QIO1_IOSB] = KCB [KCB_R_IOSB];
;  511          0586  3          KCB [KCB_L_QIO2_ARGCNT] = KCB_K_QIO2_ARGCNT;
;  512          0587  3          KCB [KCB_L_QIO2_EFN] = .KCB [KCB_L_EFN];
;  513          0588  3          KCB [KCB_L_QIO2_CHAN] = .KCB [KCB_Q_CHANNEL];
;  514          0589  3          KCB [KCB_A_QIO2_IOSB] = KCB [KCB_R_IOSB];
;  515          0590  3
;  516          0591  3
;  517          0592  3          !+
;  518          0593  3          ! Get the terminal's current characteristics.
;  519          0594  3          !-
;  520          0595  3
;  521          0596  3          KCB [KCB_L_QIO2_FUNC] = IO$_SENSEMODE;
;  522          0597  3          KCB [KCB_L_QIO2_P1] = KCB [KCB_R_TERMCHAR];
;  523          0598  3          KCB [KCB_L_QIO2_P2] = KCB_S_TERMCHAR;
;  524          0599  3
;  525          0600  3          QIO_STATUS1 = CALLG (KCB [KCB_R_QIO2], SYS$QIOW);
;  526          0601  3          IF .QIO_STATUS1
;  527          0602  3          THEN
;  528          0603  3              QIO_STATUS1 = .KCB [KCB_W_IOSB_STATUS];
;  529          0604  3          IF NOT .QIO_STATUS1
;  530          0605  3          THEN
;  531          0606  3              RETURN SMG$$CLEANUP (KCB [KCB_R_KCB], .QIO_STATUS1);
;  532          0607  3
;  533          0608  3          !+
;  534          0609  3          ! Save the device-dependent characteristics.
;  535          0610  3          !-
;  536          0611  3
;  537          0612  3          KCB [KCB_L_OLD_DEVDEPEND1] = .KCB [KCB_L_DEVDEPEND1];
;  538          0613  3          KCB [KCB_L_OLD_DEVDEPEND2] = .KCB [KCB_L_DEVDEPEND2];
;  539          0614  3
;  540          0615  3          !+
;  541          0616  3          ! Indicate if the enter/exit keypad mode sequence is compatible
```

```
;  542     0617  3          ! with that for DECCRTs.
;  543     0618  3          !-
;  544     0619  3
;  545     0620  3          IF .KCB_DEVDEPEND2 [TT2$V_DECCRT] OR
;  546     0621  3              (.KCB [KCB_B_DEVTYPE] EQL DT$_VT5X) OR
;  547     0622  3              (.KCB [KCB_B_DEVTYPE] EQL DT$_VT55) OR
;  548     0623  4              ((.KCB [KCB_B_DEVTYPE] GEQU DT$_LA36) AND    ! Covers LA36,120,34,38,
;  549     0624  3               (.KCB [KCB_B_DEVTYPE] LEQU DT$_LA100)) OR  ! 12,24,100
;  550     0625  3              (.KCB [KCB_B_DEVTYPE] EQL DT$_VR100) OR
;  551     0626  4              (.KCB [KCB_B_DEVTYPE] EQL DT$_VT173)
;  552     0627  3          THEN
;  553     0628  3              KCB [KCB_V_KPDSEQ_DECCRT] = 1;
;  554     0629  3
;  555     0630  3          END
;  556     0631  2      ELSE
;  557     0632  3          BEGIN
;  558     0633  3          !+
;  559     0634  3          ! Device is not a terminal.  Use RMS.
;  560     0635  3          !-
;  561     0636  3
;  562     0637  3          LOCAL
;  563     0638  3              OPEN_STATUS, CONNECT_STATUS, VM_STATUS;
;  564     0639  3
;  565     0640  3          !+
;  566     0641  3          ! Open file.
;  567     0642  3          !-
;  568     0643  3
;  569     0644  3              OPEN_STATUS;
;  570     0645  3          OPEN_STATUS = $OPEN (FAB=FAB);
;  571     0646  3          IF NOT .OPEN_STATUS
;  572     0647  3          THEN
;  573     0648  3              RETURN SMG$$CLEANUP (KCB [KCB_R_KCB], .OPEN_STATUS);
;  574     0649  3
;  575     0650  3          KCB [RAB$B_BID] = RAB$C_BID;
;  576     0651  3          KCB [RAB$B_BLN] = RAB$C_BLN;
;  577     0652  3          KCB [RAB$L_FAB] = FAB;
;  578     u653  3
;  579     0654  3          !+
;  580     0655  3          ! Connect to the file.
;  581     0656  3          !-
;  582     0657  3
;  583     0658  3          CONNECT_STATUS = $CONNECT (RAB=KCB [KCB_R_RAB]);
;  584     0659  3          IF NOT .CONNECT_STATUS
;  585     0660  3          THEN
;  586     0661  3              RETURN SMG$$CLEANUP (KCB [KCB_R_KCB], .CONNECT_STATUS);
;  587     0662  3
;  588     0663  3          !+
;  589     0664  3          ! Store the IFI for a future close.
;  590     0665  3          !-
;  591     0666  3
;  592     0667  3          KCB [KCB_W_IFI] = .FAB [FAB$W_IFI];
;  593     0668  3
;  594     0669  3          !+
;  595     0670  3          ! Allocate a buffer that is large enough for the file.
;  596     0671  3          !-
;  597     0672  3
;  598     0673  3          KCB [RAB$W_USZ] = .FAB [FAB$W_MRS];
```

```
;  599     0674  3          IF .KCB [RAB$W_USZ] EQL 0
;  600     0675  3          THEN
;  601     0676  4              BEGIN
;  602     0677  4              KCB [RAB$W_USZ] = .FAB [FAB$W_BLS];
;  603     0678  4              IF .XAB_FHC [XAB$W_LRL] GTRU .KCB [RAB$W_USZ]
;  604     0679  4              THEN
;  605     0680  4                  KCB [RAB$W_USZ] = .XAB_FHC [XAB$W_LRL];
;  606     0681  3              END;
;  607     0682  3          VM_STATUS = LIB$GET_VM (%REF(.KCB [RAB$W_USZ]), KCB [RAB$L_UBF]);
;  608     0683  3          IF NOT .VM_STATUS
;  609     0684  3          THEN
;  610     0685  3              RETURN SMG$$CLEANUP (KCB [KCB_R_KCB], .VM_STATUS);
;  611     0686  3
;  612     0687  3          !+
;  613     0688  3          ! Set additional values needed in the RAB.
;  614     0689  3          !-
;  615     0690  3
;  616     0691  3          KCB [RAB$B_RAC] = RAB$C_SEQ;
;  617     0692  3          KCB [RAB$V_LOC] = 1;
;  618     0693  3          KCB [RAB$V_RAH] = 1;
;  619     0694  3
;  620     0695  2          END;
;  621     0696  2
;  622     0697  2      !+
;  623     0698  2      ! Store RESULTANT_NAME in KCB and caller's filespec, if desired.
;  624     0699  2      !-
;  625     0700  2
;  626     0701  3      BEGIN
;  627     0702  3      LOCAL
;  628     0703  3          COPY_STATUS,
;  629     0704  3          STRPTR,          ! Pointer to string
;  630     0705  3          STRLEN: WORD;    ! Length of string
;  631     0706  3
;  632     0707  3      !+
;  633     0708  3      ! Try resultant name string first.
;  634     0709  3      !-
;  635     0710  3      STRPTR = RESULTANT_STRING;
;  636     0711  3      STRLEN = .NAM [NAM$B_RSL];
;  637     0712  3      IF .STRLEN EQL 0
;  638     0713  3      THEN
;  639     0714  4          BEGIN
;  640     0715  4          !+
;  641     0716  4          ! Try expanded name string.
;  642     0717  4          !-
;  643     0718  4          STRLEN = .NAM [NAM$B_ESL];
;  644     0719  4          IF .STRLEN EQL 0
;  645     0720  4          THEN
;  646     0721  5              BEGIN
;  647     0722  5              !+
;  648     0723  5              ! Use filename.
;  649     0724  5              !-
;  650     0725  5              STRPTR = .FAB [FAB$L_FNA];
;  651     0726  5              STRLEN = .FAB [FAB$B_FNS];
;  652     0727  4              END;
;  653     0728  3          END;
;  654     0729  3
;  655     0730  3      IF NOT NULLPARAMETER (K_RESULTANT_FILESPEC)
```

```
;  656          0731  3      THEN
;  657          0732  4          BEGIN
;  658          0733  4          COPY_STATUS = LIB$SCOPY_R_DX6 (.STRLEN, .STRPTR,
;  659          0734  4              RESULTANT_FILESPEC [0,0,0,0]);
;  660          0735  4          IF NOT .COPY_STATUS
;  661          0736  4          THEN
;  662          0737  4              RETURN SMG$$CLEANUP (KCB [KCB_R_KCB], .COPY_STATUS);
;  663          0738  3          END;
;  664          0739  3
;  665          0740  3      !+
;  666          0741  3      ! Store the resultant name string in the KCB.  It may be used
;  667          0742  3      ! later with output.
;  668          0743  3      !-
;  669          0744  4      BEGIN
;  670          0745  4      LOCAL
;  671          0746  4          DEVNAM_DSC : BLOCK [8, BYTE],              ! dsc for name
;  672          0747  4          DVI_ITMLST : VECTOR [1+3 + 1] INITIAL      ! item list for $GETDVI
;  673          0748  4                  (DVI$_DEVNAM     ^ 16 +64, 0, 0,  !  result name string
;  674          0749  4                   0),                             !  terminater
;  675          0750  4
;  676          0751  4          DVI_IOSB        : VECTOR [4, WORD],        ! I/O Status block for $GETDVI
;  677          0752  4          STATUS,                                    ! status retd by called routines
;  678          0753  4          DEV_DEVNAM : VECTOR [64, BYTE],            ! Buffer for result name
;  679          0754  4                                                     ! string
;  680          0755  4          DEV_NAMLEN : VOLATILE WORD;                ! Length of returned
;  681          0756  4                                                     ! resultant name string
;  682          0757  4
;  683          0758  4      BIND
;  684          0759  4          DVI_DEVNAM  = DVI_ITMLST + 4,              ! make it easy to ref items
;  685          0760  4          DVI_NAMLEN  = DVI_ITMLST + 8;              !
;  686          0761  4
;  687          0762  4      DVI_DEVNAM  = DEV_DEVNAM;
;  688          0763  4      DVI_NAMLEN  = DEV_NAMLEN;
;  689          0764  4
;  690          0765  4      DEVNAM_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  691          0766  4      DEVNAM_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
;  692          0767  4      DEVNAM_DSC [DSC$W_LENGTH] = .STRLEN;
;  693          0768  4      DEVNAM_DSC [DSC$A_POINTER] = .STRPTR; ! dsc needed for $GETDVI
;  694          0769  4
;  695        P 0770  4      STATUS = $GETDVI(   EFN     = .KCB [KCB_L_EFN],
;  696        P 0771  4                          IOSB    =  DVI_IOSB,
;  697        P 0772  4                          DEVNAM  =  DEVNAM_DSC,
;  698          0773  4                          ITMLST  =  DVI_ITMLST);
;  699          0774  4      IF NOT .STATUS THEN RETURN (.STATUS);
;  700          0775  4
;  701          0776  4      $WAITFR (EFN = .KCB [KCB_L_EFN]);              ! make $GETDVI synchronous
;  702          0777  4
;  703          0778  4      !+
;  704          0779  4      ! When the operation completes, the final status
;  705          0780  4      ! is left in the first word of the I/O status block.
;  706          0781  4      !-
;  707          0782  4
;  708          0783  4      IF NOT .DVI_IOSB [0] THEN RETURN (.DVI_IOSB [0]);
;  709          0784  4
;  710          0785  4
;  711          0786  4      KCB [KCB_W_DEVNAM_LENGTH]= .DEV_NAMLEN;       ! Length of device name
;  712          0787  4      CH$MOVE (.DEV_NAMLEN, DEV_DEVNAM, KCB [KCB_T_DEVNAM_STRING]);
```

```
;  713          0788  4
;  714          0789  3      END;
;  715          0790  3
;  716          0791  2      END;
;  717          0792  2
;  718          0793  2      !+
;  719          0794  2      ! Store the check value in the KCB so we can recognize a valid KCB
;  720          0795  2      ! the next time we see one.
;  721          0796  2      !-
;  722          0797  2
;  723          0798  2      KCB [KCB_L_CHECK] = .KCB;
;  724          0799  2
;  725          0800  2      !+
;  726          0801  2      ! Init to no pasteboard.
;  727          0802  2      !-
;  728          0803  2
;  729          0804  2      KCB [KCB_L_PASTEBOARD_ID] = -1;
;  730          0805  2
;  731          0806  2      !+
;  732          0807  2      ! Set the terminal's keypad to application mode (if not a file).
;  733          0808  2      !-
;  734          0809  2
;  735          0810  2      IF NOT .KCB [KCB_V_RMS]
;  736          0811  2      THEN
;  737          0812  2          SMG$SET_KEYPAD_MODE (%REF(KCB [KCB_R_KCB]), %REF(1));
;  738          0813  2
;  739          0814  2      !+
;  740          0815  2      ! Store the KCB address in the user's argument.
;  741          0816  2      !-
;  742          0817  2
;  743          0818  2      KEYBOARD_ID [0] = KCB [KCB_R_KCB];
;  744          0819  2
;  745          0820  2      RETURN SS$_NORMAL;
;  746          0821  2
;  747          0822  1      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, T3$                      ; 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$:     BLBC    QIO_STATUS1, T7$                    ; 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$:     PUSHAB  36(R0)                                          ; 0682
                    08    AE            60  3C 0024C          MOVZWL  (R0), 8(SP)                                     ;
                              08        AE  9F 00250          PUSHAB  8(SP)                                           ;
                        6A              02  FB 00253          CALLS   #2, LIB$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  DC 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$:   MOVC3   #16, P.AAB, DVI_ITMLST                          ; 0749
                    64  AE          18  AE  9E 002A6          MOVAB   DEV_DEVNAM, DVI_DEVNAM                          ; 0762
                    68  AE          16  AE  9E 002AB          MOVAB   DEV_NAMLEN, DVI_NAMLEN                          ; 0763
                    72  AE        010E  8F  B0 002B0          MOVW    #270, DEVNAM_DSC+2                              ; 0765
                    70  AE              57  B0 002B6          MOVW    STRLEN, DEVNAM_DSC                              ; 0767
                    74  AE              59  D0 002BA          MOVL    STRPTR, DEVNAM_DSC+4                            ; 0768
                                        7E  7C 002BE          CLRQ    -(SP)                                           ; 0773
                                        7E  D4 002C0          CLRL    -(SP)                                           ;
                                  64    AE  9F 002C2          PUSHAB  DVI_IOSB                                        ;
                                  70    AE  9F 002C5          PUSHAB  DVI_ITMLST                                      ;
                                0084    CE  9F 002C8          PUSHAB  DEVNAM_DSC                                      ;
                                        7E  D4 002CC          CLRL    -(SP)                                           ;
                                  D0    A8  DD 002CE          PUSHL   -48(KCB)                                        ;
                  00000000G             00  FB 002D1          CALLS   #8, SYS$GETDVI                                  ;
                                        45  50  E9 002D8      BLBC    STATUS, 26$                                     ; 0774
                                  D0    A8  DD 002DB          PUSHL   -48(KCB)                                        ; 0776
                  00000000G             00  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
```

```
;  749     0823  1 %SBTTL 'SMG$READ_STRING'
;  750     0824  1 GLOBAL ROUTINE SMG$READ_STRING (
;  751     0825  1     KEYBOARD_ID: REF VECTOR [, LONG],
;  752     0826  1     OUT_STRING: REF BLOCK [, BYTE],
;  753     0827  1     PROMPT_STRING: REF BLOCK [, BYTE],
;  754     0828  1     MAX_LENGTH: REF VECTOR [, LONG],
;  755     0829  1     MODIFIERS: REF VECTOR [, LONG],
;  756     0830  1     TIMEOUT: REF VECTOR [, LONG],
;  757     0831  1     TERMINATOR_SET: REF BLOCK [,BYTE],
;  758     0832  1     OUT_LENGTH: REF VECTOR [, WORD],
;  759     0833  1     TERMINATOR_CODE: REF VECTOR [, WORD],
;  760     0834  1     DISPLAY_ID: REF VECTOR [, LONG]
;  761     0835  1     ) =
;  762     0836  1
;  763     0837  1 !++
;  764     0838  1 ! FUNCTIONAL DESCRIPTION:
;  765     0839  1 !
;  766     0840  1 !       This routine reads a sequence of characters from the virtual-keyboard
;  767     0841  1 !       sperified and returns to the caller the characters read.  The caller
;  768     0842  1 !       may also specify that a code indicating the terminator be returned.
;  769     0843  1 !
;  770     0844  1 ! CALLING SEQUENCE:
;  771     0845  1 !
;  772     0846  1 !       RET_STATUS.wlc.v = SMG$READ_STRING (
;  773     0847  1 !                                 KEYBOARD_ID.rl.r,
;  774     0848  1 !                                 OUT_STRING.wt.dx
;  775     0849  1 !                                 [, [PROMPT_STRING.rt.dx]
;  776     0850  1 !                                 [, [MAX_LENGTH.rl.r]
;  777     0851  1 !                                 [, [MODIFIERS.rl.r]
;  778     0852  1 !                                 [, [TIMEOUT.rl.r]
;  779     0853  1 !                                 [, [TERMINATOR_SET.rt.ds]
;  780     0854  1 !                                 [, [OUT_LENGTH.wwu.r]
;  781     0855  1 !                                 [, [TERMINATOR_CODE.wwu.r]
;  782     0856  1 !                                 [, [DISPLAY_ID.rl.r] ]]]]]]]])
;  783     0857  1 !
;  784     0858  1 !
;  785     0859  1 ! FORMAL PARAMETERS:
;  786     0860  1 !
;  787     0861  1 !       KEYBOARD_ID     A longword containing the identification of the
;  788     0862  1 !                       virtual-keyboard from which to read.  A virtual-
;  789     0863  1 !                       keyboard is created by calling
;  790     0864  1 !                       SMG$CREATE_VIRTUAL_KEYBOARD.
;  791     0865  1 !
;  792     0866  1 !       PROMPT_STRING   An optional string to be used as the prompt for
;  793     0867  1 !                       the read operation.
;  794     0868  1 !
;  795     0869  1 !       OUT_STRING      A string into which is written the characters read.
;  796     0870  1 !
;  797     0871  1 !       MAX_LENGTH      A longword containing the maximum number of characters
;  798     0872  1 !                       to read.  The maximum value of MAX_LENGTH is 512.  If
;  799     0873  1 !                       omitted, 512 is used.
;  800     0874  1 !
;  801     0875  1 !       MODIFIERS       A longword bitmask that specifies optional behavior.
;  802     0876  1 !                       The bits defined are the same as for the $QIO
;  803     0877  1 !                       item-list entry TRM$_MODIFIERS.
;  804     0878  1 !
;  805     0879  1 !                       See the terminal driver section of the I/O User's
```

```
;  806     0880  1 !                          Guide for more information on MODIFIERS.  The
;  807     0881  1 !                          TRM$ symbols are defined by the $TRMDEF macro/module
;  808     0882  1 !                          in Digital-supplies system symbol libraries.
;  809     0883  1 !
;  810     0884  1 !       TIMEOUT            Optional timeout count.  If specified, all characters
;  811     0885  1 !                          typed in before the timeout are returned in the buffer.
;  812     0886  1 !                          If omitted, characters are returned in the buffer until
;  813     0887  1 !                          a terminator is seen.
;  814     0888  1 !
;  815     0889  1 !       TERMINATOR_SET     Descriptor of terminator mask, 0 - 8 bytes
;  816     0890  1 !                          (class, dtype ignored).  If omitted, the VAX-11
;  817     0891  1 !                          RMS standard terminator set is used.  See the terminal
;  818     0892  1 !                          driver section of the VAX/VMS I/O User's Guide for
;  819     0893  1 !                          more information on specifying terminators.
;  820     0894  1 !
;  821     0895  1 !       OUT_LENGTH         An unsigned word into which is written the number
;  822     0896  1 !                          of characters read or the maximum length of
;  823     0897  1 !                          OUT_STRING, whichever is less.
;  824     0898  1 !
;  825     0899  1 !       TERMINATOR_CODE    An unsigned word into which is written a code
;  826     0900  1 !                          indicating what character or key terminated the
;  827     0901  1 !                          read.  For more information on the values for
;  828     0902  1 !                          TERMINATOR_CODE, see the description of terminator
;  829     0903  1 !                          codes in the Screen Management functional specification.
;  830     0904  1 !
;  831     0905  1 !       DISPLAY_ID         A longword containing the identification of the
;  832     0906  1 !                          virtual display in which the read is to be performed.
;  833     0907  1 !                          If specified, the read begins at the current cursor
;  834     0908  1 !                          position in that virtual-display.  If omitted, the
;  835     0909  1 !                          read begins in the current cursor position.
;  836     0910  1 !
;  837     0911  1 ! IMPLICIT INPUTS:
;  838     0912  1 !
;  839     0913  1 !       NONE
;  840     0914  1 !
;  841     0915  1 ! IMPLICIT OUTPUTS:
;  842     0916  1 !
;  843     0917  1 !       NONE
;  844     0918  1 !
;  845     0919  1 ! COMPLETION STATUS:
;  846     0920  1 !
;  847     0921  1 !       SS$_NORMAL         Normal successful completion
;  848     0922  1 !       SS$_CANCEL         I/O operation cancelled (by SMG$CANCEL_INPUT)
;  849     0923  1 !       SMG$_EOF           End-of-file
;  850     0924  1 !       SMG$_INVDIS_ID     Invalid display-id
;  851     0925  1 !       SMG$_INVKBD_ID     Invalid keyboard-id
;  852     0926  1 !       SMG$_WRONUMARG     Wrong number of arguments
;  853     0927  1 !       LIB$_xxx           Any error from LIB$SCOPY_R_DX
;  854     0928  1 !       RMS$_xxx           Any error from $GET (except RMS$_EOF)
;  855     0929  1 !       SS$_xxx            Any error from $QIOW
;  856     0930  1 !
;  857     0931  1 ! SIDE EFFECTS:
;  858     0932  1 !
;  859     0933  1 !       NONE
;  860     0934  1 !
;  861     0935  1 !--
;  862     0936  1
```

```
;  863     0937  2      BEGIN
;  864     0938  2
;  865     0939  2      LOCAL
;  866     0940  2          KCB: REF KCB_R_KCB_STRUCT,
;  867     0941  2          BUFFER: VECTOR[512, BYTE],
;  868     0942  2          BUFFER_PTR,
;  869     0943  2          BUFFER_LEN: WORD,
;  870     0944  2          KEY_CODE: WORD,
;  871     0945  2          ITEM_LIST: $ITMLST_DECL (ITEMS=4),      ! QIO item list
;  872     0946  2          QIO_STATUS,
;  873     0947  2          COPY_STATUS,
;  874     0948  2          RET_STATUS,
;  875     0949  2          DISPLAY_ID_FLAG : INITIAL (0);
;  876     0950  2
;  877     0951  2      BUILTIN
;  878     0952  2          ACTUALCOUNT,
;  879     0953  2          CALLG,
;  880     0954  2          NULLPARAMETER,
;  881     0955  2          TESTBITSC;
;  882     0956  2
;  883     0957  2      !+
;  884     0958  2      ! Define literals for the argument positions.
;  885     0959  2      !-
;  886     0960  2
;  887     0961  2      LITERAL
;  888     0962  2          K_KEYBOARD_ID = 1,
;  889     0963  2          K_OUT_STRING = 2,
;  890     0964  2          K_PROMPT_STRING = 3,
;  891     0965  2          K_MAX_LENGTH = 4,
;  892     0966  2          K_MODIFIERS = 5,
;  893     0967  2          K_TIMEOUT = 6,
;  894     0968  2          K_TERMINATOR_SET = 7,
;  895     0969  2          K_OUT_LENGTH = 8,
;  896     0970  2          K_TERMINATOR_CODE = 9,
;  897     0971  2          K_DISPLAY_ID = 10;
;  898     0972  2
;  899     0973  2      !+
;  900     0974  2      ! Define literals for the item list positions.
;  901     0975  2      !-
;  902     0976  2
;  903     0977  2      LITERAL
;  904     0978  2          K_MODIFIER_MASK = 0,
;  905     0979  2          K_PROMPT = 1,
;  906     0980  2          K_NEXT = 2,      ! The next available position
;  907     0981  2          K_ALLOWED_MODIFIERS = IO$M_CVTLOW +   ! 3A modifiers
;  908     0982  2                               IO$M_DSABLMBX +          ! 5f40
;  909     0983  2                               IO$M_ESCAPE +
;  910     0984  2                               IO$M_NOECHO +
;  911     0985  2                               IO$M_NOFILTR +
;  912     0986  2                               IO$M_PURGE +
;  913     0987  2                               IO$M_TRMNOECHO;
;  914     0988  2
;  915     0989  2      $SMG$VALIDATE_ARGCOUNT(2,10);
;  916     0990  2
;  917     0991  2      !+
;  918     0992  2      ! Fetch and validate KCB.
;  919     0993  2      !-
```

```
;  920     0994  2
;  921     0995  2      $SMG$VALIDATE_KCB (KEYBOARD_ID, KCB);
;  922     0996  2
;  923     0997  2      !+
;  924     0998  2      ! Assume success.
;  925     0999  2      !-
;  926     1000  2
;  927     1001  2      RET_STATUS = SS$_NORMAL;
;  928     1002  2
;  929     1003  2      !+
;  930     1004  2      ! Split depending on whether or not we are using RMS.
;  931     1005  2      !-
;  932     1006  2
;  933     1007  2      IF .KCB [KCB_V_RMS]
;  934     1008  2      THEN
;  935     1009  2
;  936     1010  2          !+
;  937     1011  2          ! Read a line using $GET.
;  938     1012  2          !-
;  939     1013  2
;  940     1014  3          BEGIN
;  941     1015  3          LOCAL
;  942     1016  3              GET_STATUS;
;  943     1017  3
;  944     1018  3          GET_STATUS = $GET (RAB = KCB [KCB_R_RAB]);
;  945     1019  3          IF NOT .GET_STATUS
;  946     1020  3          THEN
;  947     1021  4              BEGIN
;  948     1022  4              IF .GET_STATUS EQLU RMS$_EOF
;  949     1023  4              THEN
;  950     1024  4                  RET_STATUS = SMG$_EOF
;  951     1025  4              ELSE
;  952     1026  4                  RET_STATUS = .GET_STATUS;
;  953     1027  3              END;
;  954     1028  3
;  955     1029  3          !+
;  956     1030  3          ! If user asked for TERMINATOR_CODE, return <CR>.
;  957     1031  3          !-
;  958     1032  3
;  959     1033  3          IF NOT NULLPARAMETER (K_TERMINATOR_CODE)
;  960     1034  3          THEN
;  961     1035  3              TERMINATOR_CODE [0] = SMG$K_TRM_CR;
;  962     1036  3
;  963     1037  3          !+
;  964     1038  3          ! Set BUFFER_PTR and BUFFER_LEN.
;  965     1039  3          !-
;  966     1040  3
;  967     1041  3          BUFFER_PTR = .KCB [RAB$L_RBF];
;  968     1042  3          BUFFER_LEN = .KCB [RAB$W_RSZ];
;  969     1043  3          END
;  970     1044  3
;  971     1045  2      ELSE
;  972     1046  2
;  973     1047  2          !+
;  974     1048  2          ! Use $QIO.
;  975     1049  2          !-
;  976     1050  2
```

```
;  977          1051  3          BEGIN
;  978          1052  3          LOCAL
;  979          1053  3              CURSOR_BUF : VECTOR [15, BYTE],      ! buffer for set cursor sequence
;  980          1054  3              CURSOR_BUF_LEN : INITIAL (0),        ! used number of bytes in cursor_buf
;  981          1055  3              FREE_COLS;                           ! calc. needed if display_id used
;  982          1056  3
;  983          1057  3          !+
;  984          1058  3          ! If the last read ended in ^Z, return SMG$_EOF.
;  985          1059  3          !-
;  986          1060  3
;  987          1061  3          IF TESTBITSC (KCB [KCB_V_CTRLZ])
;  988          1062  3          THEN
;  989          1063  3              RETURN SMG$_EOF;     ! End of file
;  990          1064  3
;  991          1065  3          !+
;  992          1066  3          ! Build the item list for the QIO.
;  993          1067  3          !-
;  994          1068  3
;  995        P 1069  3          $ITMLST_INIT (ITMLST=ITEM_LIST,
;  996        P 1070  3              (ITMCOD=TRM$_MODIFIERS, BUFADR=TRM$M_TM_ESCAPE, BUFSIZ=0),
;  997          1071  3              (ITMCOD=TRM$_PROMPT, BUFADR=0, BUFSIZ=0));
;  998          1072  3
;  999          1073  3          !+
; 1000          1074  3          ! Set up the $QIOW argument list.
; 1001          1075  3          !-
; 1002          1076  3
; 1003          1077  3          KCB [KCB_L_QIO1_FUNC] = IO$_READVBLK+IO$M_EXTEND;
; 1004          1078  3          KCB [KCB_L_QIO1_P1] = BUFFER;
; 1005          1079  3          KCB [KCB_L_QIO1_P2] = %ALLOCATION(BUFFER);
; 1006          1080  3          KCB [KCB_L_QIO1_P3] = 0;
; 1007          1081  3          KCB [KCB_L_QIO1_P4] = 0;
; 1008          1082  3          KCB [KCB_L_QIO1_P5] = ITEM_LIST;
; 1009          1083  3          KCB [KCB_L_QIO1_P6] = %ALLOCATION(ITEM_LIST) - 28; ! Don't count timeout
; 1010          1084  3                                                             ! or term. set yet,
; 1011          1085  3                                                             ! and $ITMLST_DECL
; 1012          1086  3                                                             ! adds extra 4.
; 1013          1087  3
; 1014          1088  3          !+
; 1015          1089  3          ! Check for optional input parameters.  This is done as a nested if
; 1016          1090  3          ! rather than as a series of NULLPARAMETER tests for speed.
; 1017          1091  3          !-
; 1018          1092  3
; 1019          1093  3          IF ACTUALCOUNT () GEQU K_MAX_LENGTH
; 1020          1094  3          THEN
; 1021          1095  4              BEGIN
; 1022          1096  4              LOCAL
; 1023          1097  4                  CUR_ITEM;        ! Pointer into ITEM_LIST
; 1024          1098  4              CUR_ITEM = ITEM_LIST [K_NEXT,0,0,0,0];
; 1025          1099  4
; 1026          1100  4              IF MAX_LENGTH [0] NEQA 0
; 1027          1101  4              THEN
; 1028          1102  5                  BEGIN
; 1029          1103  5                  KCB [KCB_L_QIO1_P2] = .MAX_LENGTH [0];
; 1030          1104  5                  IF .KCB [KCB_L_QIO1_P2] GTRU %ALLOCATION(BUFFER)
; 1031          1105  5                  THEN
; 1032          1106  5                      RETURN SMG$_INVMAXLEN;
; 1033          1107  4                  END;
```

```
;1034   1108  4               IF ACTUALCOUNT () GEQU K_MODIFIERS
;1035   1109  4               THEN
;1036   1110  5                   BEGIN
;1037   1111  5                   IF MODIFIERS [0] NEQA 0
;1038   1112  5                   THEN
;1039   1113  5                       ITEM_LIST [K_MODIFIER_MASK,ITM$L_BUFADR] =
;1040   1114  5                           .ITEM_LIST [K_MODIFIER_MASK,ITM$L_BUFADR] OR
;1041   1115  5                           .MODIFIERS [0];
;1042   1116  5                   IF ACTUALCOUNT () GEQU K_TIMEOUT
;1043   1117  5                   THEN
;1044   1118  6                       BEGIN
;1045   1119  6                       IF TIMEOUT [0] NEQA 0
;1046   1120  6                       THEN
;1047   1121  7                           BEGIN
;1048   1122  7                           $LIB$WWORD_A (CUR_ITEM) = 0;
;1049   1123  7                           $LIB$WWORD_A (CUR_ITEM) = TRM$_TIMEOUT;
;1050   1124  7                           $LIB$WLONG_A (CUR_ITEM) = .TIMEOUT [0];
;1051   1125  7                           $LIB$WLONG_A (CUR_ITEM) = 0;
;1052   1126  6                           END;
;1053   1127  6                       IF ACTUALCOUNT () GEQU K_TERMINATOR_SET
;1054   1128  6                       THEN
;1055   1129  7                           BEGIN
;1056   1130  7                           IF TERMINATOR_SET [0,0,0,0] NEQA 0
;1057   1131  7                           THEN
;1058   1132  8                               BEGIN
;1059   1133  8                               $LIB$WWORD_A (CUR_ITEM) = .TERMINATOR_SET [DSC$W_LENGTH];
;1060   1134  8                               $LIB$WWORD_A (CUR_ITEM) = TRM$_TERM;
;1061   1135  8                               $LIB$WLONG_A (CUR_ITEM) = .TERMINATOR_SET [DSC$A_POINTER];
;1062   1136  8                               $LIB$WLONG_A (CUR_ITEM) = 0;
;1063   1137  7                               END;
;1064   1138  7                           IF NOT NULLPARAMETER (K_DISPLAY_ID)
;1065   1139  7                           THEN
;1066   1140  8                               BEGIN                       ! display_id present
;1067   1141  8                               !+
;1068   1142  8                               ! Interfacing with output's virtual displays.  Make sure user can
;1069   1143  8                               ! not type beyond the display's boundaries by limiting the input
;1070   1144  8                               ! size to the number of free columns (allowing room for prompt, if
;1071   1145  8                               ! any).
;1072   1146  8                               ! Notice that we need some info from SMG$$SET_PROMPT_STRING (prompt
;1073   1147  8                               ! string length), and we need to provide it with some info (set cursor
;1074   1148  8                               ! sequence, if any).  So get the info here and save it.
;1075   1149  8                               !-
;1076   1150  8                               LOCAL
;1077   1151  8                                   OUT_STATUS;
;1078   1152  8                               DISPLAY_ID_FLAG = 1;        ! for quick checks later
;1079   1153  8                               IF .KCB [KCB_L_PASTEBOARD_ID] LSS 0 ! pb not established yet
;1080   1154  8                               THEN
;1081   1155  9                                   BEGIN
;1082   1156  9                                   OUT_STATUS = SMG$$GET_PASTEBOARD_ID (%REF (.KCB [KCB_W_DEVNAM_LENGTH]),
;1083   1157  9                                                                        KCB [KCB_T_DEVNAM_STRING],
;1084   1158  9                                                                        KCB [KCB_L_PASTEBOARD_ID]);
;1085   1159  9                                   IF NOT .OUT_STATUS THEN RETURN (.OUT_STATUS);
;1086   1160  8                                   END;
;1087   1161  8                               OUT_STATUS = SMG$$SET_PHYSICAL_CURSOR (.DISPLAY_ID,
;1088   1162  8                                                                      KCB [KCB_L_PASTEBOARD_ID],
;1089   1163  8                                                                      CURSOR_BUF,
;1090   1164  8                                                                      CURSOR_BUF_LEN,
```

```
;  1091     1165  8                                                          FREE_COLS);
;  1092     1166  8                          IF NOT .OUT_STATUS THEN RETURN .OUT_STATUS;
;  1093     1167  7                          END;                            ! display_id present
;  1094     1168  6                      END;
;  1095     1169  5                  END;
;  1096     1170  4              END;
;  1097     1171  4
;  1098     1172  4              !+
;  1099     1173  4              ! Change item-list length to reflect modifications.
;  1100     1174  4              !-
;  1101     1175  4
;  1102     1176  4              KCB [KCB_L_QIO1_P6] = .CUR_ITEM - .KCB [KCB_L_QIO1_P5];
;  1103     1177  3              END;
;  1104     1178  3
;  1105     1179  3          !+
;  1106     1180  3          ! Set up the prompt string.
;  1107     1181  3          !-
;  1108     1182  3
;  1109     1183  4              BEGIN
;  1110     1184  4              LOCAL
;  1111     1185  4                  STATUS,
;  1112     1186  4                  NULL_STRING : BLOCK [8,BYTE];
;  1113     1187  4              NULL_STRING [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  1114     1188  4              NULL_STRING [DSC$B_CLASS] = DSC$K_CLASS_S;
;  1115     1189  4              NULL_STRING [DSC$W_LENGTH] = 0;
;  1116     1190  4              NULL_STRING [DSC$A_POINTER] = UPLIT (' ');
;  1117     1191  4
;  1118     1192  4              IF .DISPLAY_ID_FLAG OR
;  1119     1193  4                 NOT NULLPARAMETER (K_PROMPT_STRING)
;  1120     1194  4              THEN
;  1121     1195  5                  BEGIN                               ! setup prompt
;  1122     1196  5                  STATUS = SMG$$SET_PROMPT_STRING (
;  1123     1197  6                                  (IF NULLPARAMETER (K_PROMPT_STRING)
;  1124     1198  6                                  THEN
;  1125     1199  6                                      NULL_STRING
;  1126     1200  6                                  ELSE
;  1127     1201  5                                      PROMPT_STRING [0,0,0,0]),
;  1128     1202  5                                  ITEM_LIST [K_PROMPT, 0,0,0,0],
;  1129     1203  5                                  .CURSOR_BUF_LEN,
;  1130     1204  5                                  CURSOR_BUF);
;  1131     1205  5                  IF NOT .STATUS
;  1132     1206  5                  THEN
;  1133     1207  5                      RETURN .STATUS;
;  1134     1208  4                  END;                                ! setup prompt
;  1135     1209  4              IF .DISPLAY_ID_FLAG
;  1136     1210  4              THEN
;  1137     1211  5                  BEGIN
;  1138     1212  5                  IF PROMPT_STRING [0,0,0,0] NEQA 0
;  1139     1213  5                  THEN
;  1140     1214  5                      FREE_COLS = .FREE_COLS - .PROMPT_STRING [DSC$W_LENGTH];
;  1141     1215  5                  IF .FREE_COLS LEQ 0
;  1142     1216  5                  THEN
;  1143     1217  5                      !+
;  1144     1218  5                      ! No room to accept input.  Exit.
;  1145     1219  5                      !-
;  1146     1220  5                      RETURN (SMG$_INVCOL);           ! improve this message later
;  1147     1221  5
```

```
;  1148     1222  5                  IF .KCB [KCB_L_QIO1_P2] GTR .FREE_COLS
;  1149     1223  5                  THEN
;  1150     1224  5                      KCB [KCB_L_QIO1_P2] = .FREE_COLS;
;  1151     1225  4                  END;
;  1152     1226  3              END;
;  1153     1227
;  1154     1228              !**
;  1155     1229              ! If this routine is supposed to execute on a V3.n system, map back
;  1156     1230              ! the item-list entries into the old-style of $QIO.
;  1157     1231              !**
;  1158     1232
;  1159   L 1233              %IF %VARIANT
;  1160   U 1234              %THEN
;  1161   U 1235                  BEGIN
;  1162   U 1236                  KCB [KCB_L_QIO1_FUNC] = IO$_READVBLK;
;  1163   U 1237                  KCB [KCB_L_QIO1_P5] = 0;
;  1164   U 1238                  KCB [KCB_L_QIO1_P6] = 0;
;  1165   U 1239                  IF .ITEM_LIST [K_PROMPT, ITM$W_BUFSIZ] NEQ 0
;  1166   U 1240                  THEN
;  1167   U 1241                      BEGIN
;  1168   U 1242                      KCB [KCB_L_QIO1_P5] = .ITEM_LIST [K_PROMPT, ITM$L_BUFADR];
;  1169   U 1243                      KCB [KCB_L_QIO1_P6] = .ITEM_LIST [K_PROMPT, ITM$W_BUFSIZ];
;  1170   U 1244                      KCB [KCB_L_QIO1_FUNC] = IO$_READPROMPT;
;  1171   U 1245                      END;
;  1172   U 1246                  IF NOT NULLPARAMETER(K_TIMEOUT)
;  1173   U 1247                  THEN
;  1174   U 1248                      BEGIN
;  1175   U 1249                      KCB [KCB_L_QIO1_P3] = .TIMEOUT [0];
;  1176   U 1250                      KCB [KCB_L_QIO1_FUNC] = .KCB [KCB_L_QIO1_FUNC] OR IO$M_TIMED;
;  1177   U 1251                      END;
;  1178   U 1252                  IF NOT NULLPARAMETER(K_TERMINATOR_SET)
;  1179   U 1253                  THEN
;  1180   U 1254                      BEGIN
;  1181   U 1255                      LOCAL
;  1182   U 1256                          CUR_ITEM: REF $ITMLST_DECL (ITEMS=1);
;  1183   U 1257                      CUR_ITEM = ITEM_LIST [K_NEXT, 0,0,0,0];
;  1184   U 1258                      IF .CUR_ITEM [0, ITM$W_ITMCOD] NEQ TRM$_TERM
;  1185   U 1259                      THEN
;  1186   U 1260                          CUR_ITEM = CUR_ITEM [1, 0,0,0,0];
;  1187   U 1261                      KCB [KCB_L_QIO1_P4] = .CUR_ITEM;
;  1188   U 1262                      END;
;  1189   U 1263
;  1190   U 1264                  IF NOT NULLPARAMETER (K_MODIFIERS)
;  1191   U 1265                  THEN
;  1192   U 1266                      KCB [KCB_L_QIO1_FUNC] = .KCB [KCB_L_QIO1_FUNC] OR
;  1193   U 1267                                              (.MODIFIERS [0] AND K_ALLOWED_MODIFIERS);
;  1194   U 1268
;  1195   U 1269                  END;
;  1196     1270              %FI     !** End of variant code
;  1197     1271
;  1198     1272  4           BEGIN                               ! QIO w/rendition
;  1199     1273  4           LOCAL
;  1200     1274  4               PBCB : REF BLOCK [,BYTE],
;  1201     1275  4               DCB : REF BLOCK [,BYTE];
;  1202     1276  4
;  1203     1277  4           IF .DISPLAY_ID_FLAG
;  1204     1278  4           THEN
```

```
;  1205     1279  5              BEGIN                           . display_id/check rendition
;  1206     1280  5
;  1207     1281  5              !+
;  1208     1282  5              ! Make sure the prompt and input appear ir the correct video
;  1209     1283  5              ! rendition.
;  1210     1284  5              !-
;  1211     1285  5
;  1212     1286  5              $SMG$GET_DCB (.DISPLAY_ID, DCB);
;  1213     1287  5              $SMG$GET_PBCB (KCB [KCB_L_PASTEBOARD_ID], PBCB);
;  1214     1288  5
;  1215     1289  5              !+
;  1216     1290  5              ! Don't allow input if display is batched.
;  1217     1291  5              !-
;  1218     1292  5
;  1219     1293  5              IF .DCB[DCB_L_BATCH_LEVEL] NEQ 0
;  1220     1294  5              THEN
;  1221     1295  5                  RETURN  SMG$_ILLBATFNC;
;  1222     1296  5
;  1223     1297  5              !+
;  1224     1298  5              ! Handle renditions if necessary.
;  1225     1299  5              !-
;  1226     1300  5
;  1227     1301  5              SMG$$SET_ATTRIBUTES_ON (.PBCB,.DCB [DCB_B_DEF_VIDEO_ATTR]);
;  1228     1302  5
;  1229     1303  4              END;                            ! display_id/check rendition
;  1230     1304  4
;  1231     1305  4          !+
;  1232     1306  4          ! Read the string.
;  1233     1307  4          !-
;  1234     1308  4
;  1235     1309  4          QIO_STATUS = CALLG (KCB [KCB_R_QIO1], SYS$QIOW);
;  1236     1310  4          IF .QIO_STATUS
;  1237     1311  4          THEN
;  1238     1312  4              QIO_STATUS = .KCB [KCB_W_IOSB_STATUS];
;  1239     1313  4
;  1240     1314  4          !+
;  1241     1315  4          ! If we set the video rendition, turn it off now.
;  1242     1316  4          !-
;  1243     1317  4
;  1244     1318  4          IF .DISPLAY_ID_FLAG
;  1245     1319  4          THEN
;  1246     1320  4              SMG$$SET_ATTRIBUTES_OFF (.PBCB,.DCB [DCB_B_DEF_VIDEO_ATTR]);
;  1247     1321  4
;  1248     1322  4          IF NOT .QIO_STATUS
;  1249     1323  4          THEN
;  1250     1324  4              RET_STATUS = .QIO_STATUS;
;  1251     1325  3          END;                        ! QIO w/rendition
;  1252     1326  3
;  1253     1327  3          !+
;  1254     1328  3          ! Check for partial escape sequence in the buffer.  This can
;  1255     1329  3          ! happen if the QIO was limited to 2 characters and the input
;  1256     1330  3          ! string was terminated by a keypad key (which are 3 chars).
;  1257     1331  3          !-
;  1258     1332  3
;  1259     1333  3          IF .KCB [KCB_W_IOSB_STATUS] EQL SS$_PARTESCAPE
;  1260     1334  3          THEN
;  1261     1335  4              BEGIN
```

```
;  1262          1336  4              LOCAL
;  1263          1337  4                  QIO_RESULT,
;  1264          1338  4                  CHAR_BUF : BYTE,
;  1265          1339  4                  END_ESC_FLAG : INITIAL (0);
;  1266          1340  4
;  1267          1341  4              BUFFER_LEN = .KCB [KCB_W_IOSB_COUNT] + .KCB [KCB_B_IOSB_TERMLEN];
;  1268          1342  4                                   ! point past data & partial terminator
;  1269          1343  4
;  1270          1344  4              WHILE .END_ESC_FLAG EQL 0 DO
;  1271          1345  5                  BEGIN
;  1272        P 1346  5                  QIO_RESULT = $QIOW (EFN = .KCB [KCB_L_EFN],
;  1273        P 1347  5                                      CHAN = .KCB [KCB_W_CHANNEL],
;  1274        P 1348  5                                      FUNC = (IO$_READVBLK + IO$M_ESCAPE +
;  1275        P 1349  5                                              IO$M_NOECHO + IO$M_NOFILTR +
;  1276        P 1350  5                                              IO$M_TRMNOECHO),
;  1277        P 1351  5                                      P1 = CHAR_BUF,
;  1278        P 1352  5                                      P2 = 1,
;  1279          1353  5                                      P4 = .KCB [KCB_L_QIO1_P4]);
;  1280          1354  5                                         ! P4 = defined terminator set
;  1281          1355  5                  IF NOT .QIO_RESULT THEN RET_STATUS = .QIO_RESULT;
;  1282          1356  5
;  1283          1357  5                  !+
;  1284          1358  5                  ! Move terminator characters to buffer.
;  1285          1359  5                  !-
;  1286          1360  5
;  1287          1361  5                  BUFFER [.BUFFER_LEN] = .CHAR_BUF;
;  1288          1362  5                  BUFFER_LEN = .BUFFER_LEN + 1;
;  1289          1363  5                  KCB [KCB_B_IOSB_TERMLEN] = .KCB [KCB_B_IOSB_TERMLEN] + 1;
;  1290          1364  5
;  1291          1365  5                  CASE .CHAR_BUF FROM %C'A' TO %C'~' OF
;  1292          1366  5                  SET
;  1293          1367  5                      [%C'A' TO %C'D',     ! arrow keys
;  1294          1368  5                       %C'M',              ! Enter
;  1295          1369  5                       %C'P' to %C'S',     ! PFn keys
;  1296          1370  5                       %C'l' to %C'n',     ! keypad - , .
;  1297          1371  5                       %C'p' to %C'y',     ! keypad 0-9
;  1298          1372  5                       %C'~']:             ! function keys F6-F20 (no codes for
;  1299          1373  5                                           ! F1-F5), editing keys (E1-E6)
;  1300          1374  6                          BEGIN
;  1301          1375  6                          END_ESC_FLAG = 1;
;  1302          1376  6                          RET_STATUS = SS$_NORMAL;
;  1303          1377  6                                           ! get rid of ss$_partescape error
;  1304          1378  5                          END;
;  1305          1379  5
;  1306          1380  5                      [INRANGE, OUTRANGE]:
;  1307          1381  5                          ;                ! didn't find end of esc terminator
;  1308          1382  5                                           ! keep searching
;  1309          1383  5                  TES;
;  1310          1384  5
;  1311          1385  4                  END;                     ! end of WHILE loop
;  1312          1386  3              END;                         ! end of partial ESC terminator
;  1313          1387  3
;  1314          1388  3          !+
;  1315          1389  3          ! Get the terminator code.
;  1316          1390  3          !-
;  1317          1391  3
;  1318          1392  3          IF .QIO_STATUS EQL SS$_TIMEOUT
```

```
;  1319     1393  3          THEN
;  1320     1394  3              KEY_CODE = SMG$K_TRM_TIMEOUT
;  1321     1395  3          ELSE
;  1322     1396  3              KEY_CODE  = SMG$$TERM_TO_KEYCODE (BUFFER [.KCB [KCB_W_IOSB_COUNT]],
;  1323     1397  3                  .KCB [KCB_B_IOSB_TERMLEN]);
;  1324     1398  3
;  1325     1399  3          !+
;  1326     1400  3          ! If DISPLAY_ID was passed, call back output side to tell it
;  1327     1401  3          ! what changed on the screen as a result of this prompt & input.
;  1328     1402  3          !-
;  1329     1403  3
;  1330     1404  3          IF .DISPLAY_ID_FLAG
;  1331     1405  3          THEN
;  1332     1406  4              BEGIN
;  1333     1407  4              LOCAL
;  1334     1408  4                  OUT_STATUS;
;  1335     1409  4              IF PROMPT_STRING [0,0,0,0] NEQA 0
;  1336     1410  4              THEN
;  1337     1411  5                  BEGIN                       ! report prompt string change
;  1338     1412  5                  OUT_STATUS = SMG$$REPORT_CHANGE_REPLACE (.DISPLAY_ID,
;  1339     1413  5                                                          KCB [KCB_L_PASTEBOARD_ID],
;  1340     1414  5                                                          %REF (.PROMPT_STRING [DSC$W_LENGTH]),
;  1341     1415  5                                                          .PROMPT_STRING [DSC$A_POINTER]);
;  1342     1416  5                  IF NOT .OUT_STATUS AND
;  1343     1417  5                     .OUT_STATUS NEQ SMG$_STRTERESC
;  1344     1418  5                  THEN
;  1345     1419  5                      RET_STATUS = .OUT_STATUS;
;  1346     1420  4                  END;                        ! report prompt string change
;  1347     1421  4
;  1348     1422  5                  BEGIN
;  1349     1423  5                  MAP
;  1350     1424  5                      MODIFIERS : REF BLOCK [,BYTE];
;  1351     1425  5                  IF NULLPARAMETER (K_MODIFIERS) OR
;  1352     1426  6                     (NOT NULLPARAMETER (K_MODIFIERS) AND
;  1353     1427  6                      NOT (.MODIFIERS [TRM$V_TM_NOECHO]))
;  1354     1428  5                  THEN
;  1355     1429  6                      BEGIN                   ! skip if noecho requested
;  1356     1430  6                      OUT_STATUS = SMG$$REPORT_CHANGE_REPLACE (.DISPLAY_ID,
;  1357     1431  6                                                              KCB [KCB_L_PASTEBOARD_ID],
;  1358     1432  6                                                              %REF (.KCB [KCB_W_IOSB_COUNT] +
;  1359     1433  8                                                                    (IF (NOT NULLPARAMETER (K_MODIFIERS) AND
;  1360     1434  8                                                                         NOT (.MODIFIERS [TRM$V_TM_TRMNOECHO]))
;  1361     1435  7                                                                    THEN .KCB [KCB_B_IOSB_TERMLEN]
;  1362     1436  6                                                                    ELSE 0)),
;  1363     1437  6                                                              BUFFER);
;  1364     1438  6                                          ! report user's input string,
;  1365     1439  6                                          ! possibly with terminator
;  1366     1440  6                      IF NOT .OUT_STATUS AND
;  1367     1441  6                         .OUT_STATUS NEQ SMG$_STRTERESC
;  1368     1442  6                      THEN
;  1369     1443  6                          RET_STATUS = .OUT_STATUS;
;  1370     1444  5                      END;
;  1371     1445  4              END;
;  1372     1446  4
;  1373     1447  3              END;
;  1374     1448  3
;  1375     1449  3          !+
```

```
;  1376     1450  3          ! Check for a terminator of ^Z.
;  1377     1451  3          !-
;  1378     1452  3
;  1379     1453  3          IF .KEY_CODE EQL SMG$K_TRM_CTRLZ
;  1380     1454  3          THEN
;  1381     1455  4              BEGIN
;  1382     1456  4              !+
;  1383     1457  4              ! If count is zero, return SMG$_EOF now, else set
;  1384     1458  4              ! the flag so that it will be returned the next time.
;  1385     1459  4              !-
;  1386     1460  4
;  1387     1461  4              IF .KCB [KCB_W_IOSB_COUNT] EQL 0
;  1388     1462  4              THEN
;  1389     1463  4                  RET_STATUS = SMG$_EOF
;  1390     1464  4              ELSE
;  1391     1465  4                  KCB [KCB_V_CTRLZ] = 1;
;  1392     1466  3              END;
;  1393     1467  3
;  1394     1468  3          !+
;  1395     1469  3          ! If the caller asked for TERMINATOR_CODE, store it.
;  1396     1470  3          !-
;  1397     1471  3
;  1398     1472  3          IF NOT NULLPARAMETER (K_TERMINATOR_CODE)
;  1399     1473  3          THEN
;  1400     1474  3              TERMINATOR_CODE [0] = .KEY_CODE;
;  1401     1475  3
;  1402     1476  3          !+
;  1403     1477  3          ! Set BUFFER_PTR and BUFFER_LEN.
;  1404     1478  3          !-
;  1405     1479  3
;  1406     1480  3              BUFFER_PTR = BUFFER;
;  1407     1481  3              BUFFER_LEN = .KCB [KCB_W_IOSB_COUNT];
;  1408     1482  3
;  1409     1483  3              !+
;  1410     1484  3              ! Delete the prompt string, if any.
;  1411     1485  3              !-
;  1412     1486  3
;  1413     1487  3              IF NOT NULLPARAMETER (K_PROMPT_STRING) OR
;  1414     1488  3                 .DISPLAY_ID_FLAG
;  1415     1489  3              THEN
;  1416     1490  3                  SMG$$DELETE_PROMPT_STRING (ITEM_LIST [K_PROMPT,0,0,0,0]);
;  1417     1491  2              END;
;  1418     1492  2
;  1419     1493  2          !+
;  1420     1494  2          ! We now have the complete line in BUFFER, with length BUFFER_LEN.
;  1421     1495  2          ! Copy it to the user's string.
;  1422     1496  2          !-
;  1423     1497  2
;  1424     1498  2          COPY_STATUS = LIB$SCOPY_R_DX6 (.BUFFER_LEN,
;  1425     1499  2              .BUFFER_PTR, OUT_STRING [0,0,0,0]);
;  1426     1500  2          IF NOT .COPY_STATUS
;  1427     1501  2          THEN
;  1428     1502  2              RETURN .COPY_STATUS;
;  1429     1503  2
;  1430     1504  2          !+
;  1431     1505  2          ! Fill in OUT_LENGTH if requested.
;  1432     1506  2          !-
```

```
;  1433          1507  2
;  1434          1508  2      IF NOT NULLPARAMETER (K_OUT_LENGTH)
;  1435          1509  2      THEN
;  1436          1510  3          BEGIN
;  1437          1511  3          LOCAL
;  1438          1512  3              LINE_LENGTH: WORD;
;  1439          1513  3          LIB$ANALYZE_SDESC_R2 (OUT_STRING [0,0,0,0]; LINE_LENGTH);
;  1440          1514  3          IF .LINE_LENGTH LSSU .BUFFER_LEN
;  1441          1515  3          THEN
;  1442          1516  3              BUFFER_LEN = .LINE_LENGTH;
;  1443          1517  3          OUT_LENGTH [0] = .BUFFER_LEN;
;  1444          1518  2          END;
;  1445          1519  2
;  1446          1520  2      RETURN .RET_STATUS;
;  1447          1521  2
;  1448          1522  1      END;                                    ! End of routine SMG$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
                            000000000G           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                       _       #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)                                   ;
                                00000C00G     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$                         ; 1166
                          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$:            PUSHAB  CURSOR_BUF                              ; 1202
                                              0C    AE  DD 00187                 PUSHL   CURSOR_BUF_LEN                          ; 1203
                                              3C    AE  9F 0018A                 PUSHAB  ITEM_LIST+72                            ; 1202
                                              03        6C  91 0018D             CMPB    (AP), #3                                ; 1197
                                                        05  1F 00190             BLSSU   18$                                     ;
                                              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    "S"G$_ILLBATFNC, R0                        ; 1295
                                04 00224          RET                                                ;
                 7E     2E   A3 9A 00225 30$:     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$-39$,-
                                                                                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$-39$,-
                                                                                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
```

```
;1450    1523  1 %SBTTL 'SMG$READ_COMPOSED_LINE'
;1451    1524  1 GLOBAL ROUTINE SMG$READ_COMPOSED_LINE (
;1452    1525  1     KEYBOARD_ID: REF VECTOR [, LONG],
;1453    1526  1     KEY_TABLE_ID: REF VECTOR [, LONG],
;1454    1527  1     OUT_LINE: REF BLOCK [, BYTE],
;1455    1528  1     PROMPT_STRING: REF BLOCK [, BYTE],
;1456    1529  1     OUT_LENGTH: REF VECTOR [, WORD],
;1457    1530  1     DISPLAY_ID: REF VECTOR [, LONG]
;1458    1531  1     ) =
;1459    1532  1
;1460    1533  1 !++
;1461    1534  1 ! FUNCTIONAL DESCRIPTION:
;1462    1535  1 !
;1463    1536  1 !       SMG$READ_COMPOSED_LINE reads a line composed of normal keystrokes
;1464    1537  1 !       and key equivalence strings.  As keys are pressed which are
;1465    1538  1 !       defined in the key-table, the equivalence string, if any, is
;1466    1539  1 !       substituted in the input line for the key.  Attributes of
;1467    1540  1 !       the key definition control whether the equivalence string
;1468    1541  1 !       is echoed and whether the read terminates.
;1469    1542  1 !
;1470    1543  1 !       The manner of the read operation is not affected by previous calls
;1471    1544  1 !       to other Screen Management procedures, unlike SMG$READ_STRING.
;1472    1545  1 !       Characters typed are always echoed (in normal rendition).
;1473    1546  1 !       <CR> always terminates the read operation.
;1474    1547  1 !       If <CTRL/Z> is pressed and there is no definition
;1475    1548  1 !       in the key-table for CTRLZ, an appropriate string is echoed, the read terminates.
;1476    1549  1 !       If the <CTRL/Z> was the first character in the line, then the
;1477    1550  1 !       status SMG$_EOF is returned, else SMG$_EOF is returned on the next
;1478    1551  1 !       input operation.  SMG$_EOF is also returned if RMS is used for input
;1479    1552  1 !       and it returns RMS$_EOF. No other terminators are recognized except
;1480    1553  1 !       as specified by attributes in the key definitions.
;1481    1554  1 !
;1482    1555  1 ! CALLING SEQUENCE:
;1483    1556  1 !
;1484    1557  1 !       RET_STATUS.wlc.v = SMG$READ_COMPOSED_LINE (
;1485    1558  1 !                                  KEYBOARD_ID.rl.r,
;1486    1559  1 !                                  KEY_TABLE_ID.rl.r,
;1487    1560  1 !                                  OUT_LINE.wt.dx
;1488    1561  1 !                                  [, [PROMPT_STRING.rt.dx]
;1489    1562  1 !                                  [, [OUT_LENGTH.wwu.r]
;1490    1563  1 !                                  [, [DISPLAY_ID.rl.r] ]]])
;1491    1564  1 !
;1492    1565  1 ! FORMAL PARAMETERS:
;1493    1566  1 !
;1494    1567  1 !       KEYBOARD_ID     A longword containing the identification of the
;1495    1568  1 !                       virtual-keyboard from which to read.  The virtual-
;1496    1569  1 !                       keyboard is created by SMG$CREATE_VIRTUAL_KEYBOARD.
;1497    1570  1 !
;1498    1571  1 !       KEY_TABLE_ID    A longword containing the identification of the
;1499    1572  1 !                       key-definition-table to be used for translating
;1500    1573  1 !                       keystrokes.  The key-definition table-table is
;1501    1574  1 !                       created by SMG$CREATE_KEY_TABLE.
;1502    1575  1 !
;1503    1576  1 !       OUT_LINE        A string into which is written the complete composed
;1504    1577  1 !                       line.
;1505    1578  1 !
;1506    1579  1 !       PROMPT_STRING   An optional string to be used as the prompt for the
```

```
;  1507     1580  1 !                          read operation.
;  1508     1581  1 !
;  1509     1582  1 !       OUT_LENGTH         An unsigned word into which is written the number of
;  1510     1583  1 !                          characters read or the maximum length of OUT_LINE,
;  1511     1584  1 !                          whichever is less.
;  1512     1585  1 !
;  1513     1586  1 !       DISPLAY_ID         A longword containing the identification of the
;  1514     1587  1 !                          virtual-display in which the read is to be performed.
;  1515     1588  1 !                          If specified, the read begins at the beginning of the
;  1516     1589  1 !                          next line after the current cursor position in that
;  1517     1590  1 !                          virtual-display.  If omitted, the read begins at the
;  1518     1591  1 !                          beginning of the next line after the current cursor
;  1519     1592  1 !                          position.
;  1520     1593  1 !
;  1521     1594  1 ! IMPLICIT INPUTS:
;  1522     1595  1 !
;  1523     1596  1 !       NONE
;  1524     1597  1 !
;  1525     1598  1 ! IMPLICIT OUTPUTS:
;  1526     1599  1 !
;  1527     1600  1 !       NONE
;  1528     1601  1 !
;  1529     1602  1 ! COMPLETION STATUS:
;  1530     1603  1 !
;  1531     1604  1 !       SS$_NORMAL         Normal successful completion
;  1532     1605  1 !       SS$_CANCEL         I/O operation cancelled (by SMG$CANCEL_INPUT)
;  1533     1606  1 !       SMG$_EOF           End-of-file
;  1534     1607  1 !       SMG$_INVDIS_ID     Invalid display-id
;  1535     1608  1 !       SMG$_INVKBD_ID     Invalid keyboard-id
;  1536     1609  1 !       SMG$_INVKTB_ID     Invalid key-table-id
;  1537     1610  1 !       SMG$_WRONUMARG     Wrong number of arguments
;  1538     1611  1 !       LIB$_xxx           Any error from LIB$SCOPY_R_DX
;  1539     1612  1 !       RMS$_xxx           Any error from $GET (except RMS$_EOF)
;  1540     1613  1 !       SS$_xxx            Any error from $QIOW
;  1541     1614  1 !
;  1542     1615  1 ! SIDE EFFECTS:
;  1543     1616  1 !
;  1544     1617  1 !       NONE
;  1545     1618  1 !
;  1546     1619  1 !--
;  1547     1620  1
;  1548     1621  2     BEGIN
;  1549     1622  2
;  1550     1623  2     FIELD
;  1551     1624  2         FLAGS_FIELDS =
;  1552     1625  2             SET
;  1553     1626  2             V_RESET_STATE = [0,0,1,0],
;  1554     1627  2             V_SKIP_NEWLINE = [0,1,1,0],
;  1555     1628  2             V_SKIP_LAST_ECHO = [0,2,1,0],
;  1556     1629  2             V_SETPROMPT = [0,3,1,0],
;  1557     1630  2             V_ECHO_CTRLZ = [0,4,1,0]
;  1558     1631  2             TES;
;  1559     1632  2
;  1560     1633  2     LOCAL
;  1561     1634  2         KCB: REF KCB_R_KCB_STRUCT,
;  1562     1635  2         KDE: REF KDE_R_KDE_STRUCT,
;  1563     1636  2         KTH: REF KTH_R_KTH_STRUCT,
```

```
;1564    1637  2          BUFFER: VECTOR [516, BYTE], . Allow 4 chars for terminator echo
;1565    1638  2          BUFFER_PTR,
;1566    1639  2          BUFFER_LEN: WORD,
;1567    1640  2          NEXT_POS,
;1568    1641  2          TKEY: VECTOR [34, BYTE],
;1569    1642  2          TKEY_POS: REF VECTOR [, WORD],
;1570    1643  2          TKEY_DSC: BLOCK [8, BYTE],
;1571    1644  2          TKEY_DSC_PTR: REF BLOCK [8, BYTE],
;1572    1645  2          KEY_CODE: WORD,
;1573    1646  2          ITEM_LIST: $ITMLST_DECL (ITEMS=5),
;1574    1647  2          FLAGS: BLOCK [1, BYTE] FIELD (FLAGS_FIELDS),
;1575    1648  2          TERM_NAME_PTR: REF VECTOR [, BYTE],
;1576    1649  2          DISPLAY_ID_FLAG : INITIAL(0),
;1577    1650  2          QIO_STATUS,
;1578    1651  2          COPY_STATUS,
;1579    1652  2          RET_STATUS;
;1580    1653  2
;1581    1654  2      LITERAL
;1582    1655  2          K_CR = 13,          ! <CR>
;1583    1656  2          K_LF = 10,          ! <LF>
;1584    1657  2          K_CTRLZ = 26,       ! ^Z
;1585    1658  2          K_ESC = 27;         ! <ESC>
;1586    1659  2
;1587    1660  2      BIND
;1588    1661  2          T_CR = UPLIT BYTE (K_CR);
;1589    1662  2
;1590    1663  2      BUILTIN
;1591    1664  2          ACTUALCOUNT,
;1592    1665  2          CALLG,
;1593    1666  2          NULLPARAMETER,
;1594    1667  2          TESTBITSC,
;1595    1668  2          TESTBITSS;
;1596    1669  2
;1597    1670  2      !+
;1598    1671  2      ! Define literals for the argument positions.
;1599    1672  2      !-
;1600    1673  2
;1601    1674  2      LITERAL
;1602    1675  2          K_KEYBOARD_ID = 1,
;1603    1676  2          K_KEY_TABLE_ID = 2,
;1604    1677  2          K_OUT_LINE = 3,
;1605    1678  2          K_PROMPT_STRING = 4,
;1606    1679  2          K_OUT_LENGTH = 5,
;1607    1680  2          K_DISPLAY_ID = 6;
;1608    1681  2
;1609    1682  2      !+
;1610    1683  2      ! Define literals for the item list positions.
;1611    1684  2      !-
;1612    1685  2
;1613    1686  2      LITERAL
;1614    1687  2          K_MODIFIERS = 0,
;1615    1688  2          K_PROMPT = 1,
;1616    1689  2          K_INISTRNG = 2,
;1617    1690  2          K_INIOFFSET = 3,
;1618    1691  2          K_TERM = 4;
;1619    1692  2
;1620    1693  2      !+
```

```
;  1621          1694  2      ! Define the default strings which echo for control-Z.
;  1622          1695  2      !-
;  1623          1696  2
;  1624          1697  2      OWN
;  1625          1698  2          T_CTRLZ_DECCRT: VECTOR [17, BYTE] PSECT (_SMG$CODE) INITIAL (
;  1626          1699  2              BYTE (
;  1627          1700  2                  K_ESC,'7',              ! DECSC - Save Cursor
;  1628          1701  2                  K_ESC,'[7m',            ! SGR - Negative image
;  1629          1702  2                  '  Exit  ',             !  "  Exit  "
;  1630          1703  2                  K_ESC,'[m',             ! Normal rendition (in case resore fails)
;  1631          1704  2                  K_ESC,'8')),            ! DECRC - Restore Cursor
;  1632          1705  2          T_CTRLZ_OTHER: VECTOR [6, BYTE] PSECT (_SMG$CODE) INITIAL (
;  1633          1706  2              BYTE (
;  1634          1707  2                  '*EXIT*'));             ! "*EXIT*"
;  1635          1708  2
;  1636          1709  2      $SMG$VALIDATE_ARGCOUNT(3,6);
;  1637          1710  2
;  1638          1711  2      !+
;  1639          1712  2      ! Fetch and validate KCB.
;  1640          1713  2      !-
;  1641          1714  2
;  1642          1715  2      $SMG$VALIDATE_KCB (KEYBOARD_ID, KCB);
;  1643          1716  2
;  1644          1717  2      !+
;  1645          1718  2      ! Fetch and validate KTH.
;  1646          1719  2      !-
;  1647          1720  2
;  1648          1721  2      $SMG$VALIDATE_KTH (KEY_TABLE_ID, KTH);
;  1649          1722  2
;  1650          1723  2      !+
;  1651          1724  2      ! Assume success.
;  1652          1725  2      !-
;  1653          1726  2
;  1654          1727  2      RET_STATUS = SS$_NORMAL;
;  1655          1728  2
;  1656          1729  2      !+
;  1657          1730  2      ! Split depending on whether or not we are using RMS.
;  1658          1731  2      !-
;  1659          1732  2
;  1660          1733  2      IF .KCB [KCB_V_RMS]
;  1661          1734  2      THEN
;  1662          1735  2
;  1663          1736  2          !+
;  1664          1737  2          ! Read a line using $GET.
;  1665          1738  2          !-
;  1666          1739  2
;  1667          1740  3          BEGIN
;  1668          1741  3          LOCAL
;  1669          1742  3              GET_STATUS;
;  1670          1743  3
;  1671          1744  3          GET_STATUS = $GET (RAB = KCB [KCB_R_RAB]);
;  1672          1745  3          IF NOT .GET_STATUS
;  1673          1746  3          THEN
;  1674          1747  4              BEGIN
;  1675          1748  4              IF .GET_STATUS EQLU RMS$_EOF
;  1676          1749  4              THEN
;  1677          1750  4                  RET_STATUS = SMG$_EOF
```

```
;  1678     1751  4              ELSE
;  1679     1752  4                  RET_STATUS = .GET_STATUS;
;  1680     1753  3              END;
;  1681     1754  3
;  1682     1755  3          !+
;  1683     1756  3          ! Set BUFFER_PTR and BUFFER_LEN.
;  1684     1757  3          !-
;  1685     1758  3
;  1686     1759  3          BUFFER_PTR = .KCB [RAB$L_RBF];
;  1687     1760  3          BUFFER_LEN = .KCB [RAB$W_RSZ];
;  1688     1761  3          END
;  1689     1762  3
;  1690     1763  2      ELSE
;  1691     1764  2
;  1692     1765  2          !+
;  1693     1766  2          ! Use $QIO and do all the keypad substitution stuff.
;  1694     1767  2          !-
;  1695     1768  2
;  1696     1769  3          BEGIN
;  1697     1770  3
;  1698     1771  3          LOCAL
;  1699     1772  3              CURSOR_BUF : VECTOR [15, BYTE],     ! buffer for set cursor sequence
;  1700     1773  3              CURSOR_BUF_LEN : INITIAL (0),       ! used number of bytes in cursor_buf
;  1701     1774  3              FREE_COLS;                  ! calc. needed if display_id used
;  1702     1775  3
;  1703     1776  3          !+
;  1704     1777  3          ! If the last read ended in ^Z, return SMG$_EOF.
;  1705     1778  3          !-
;  1706     1779  3
;  1707     1780  3          IF TESTBITSC (KCB [KCB_V_CTRLZ])
;  1708     1781  3          THEN
;  1709     1782  3              RETURN SMG$_EOF;    ! End of file
;  1710     1783  3
;  1711     1784  3          !+
;  1712     1785  3          ! Initialize flags.
;  1713     1786  3          !-
;  1714     1787  3
;  1715     1788  3          FLAGS = 0;
;  1716     1789  3          FLAGS [V_RESET_STATE] = 1;
;  1717     1790  3
;  1718     1791  3          !+
;  1719     1792  3          ! Build the item list for the QIO.
;  1720     1793  3          !-
;  1721     1794  3
;  1722   P 1795  3          $ITMLST_INIT (ITMLST=ITEM_LIST,
;  1723   P 1796  3              (ITMCOD=TRM$_MODIFIERS,
;  1724   P 1797  3                  BUFADR=(TRM$M_TM_ESCAPE+TRM$M_TM_TRMNOECHO) OR
;  1725   P 1798  3                      .KTH [KTH_L_MODIFIERS], BUFSIZ=0),
;  1726   P 1799  3              (ITMCOD=TRM$_PROMPT, BUFADR=T_CR, BUFSIZ=1),
;  1727   P 1800  3              (ITMCOD=TRM$_INISTRNG, BUFADR=BUFFER, BUFSIZ=0),
;  1728   P 1801  3              (ITMCOD=TRM$_INIOFFSET, BUFADR=0, BUFSIZ=0),
;  1729   P 1802  3              (ITMCOD=TRM$_TERM, BUFSIZ=0,
;  1730     1803  3                  BUFADR=.KTH [KTH_L_TERM_MASK]));
;  1731     1804  3
;  1732     1805  3          IF NOT NULLPARAMETER (K_DISPLAY_ID)
;  1733     1806  3          THEN
;  1734     1807  4              BEGIN                       ! display_id present
```

```
;1735   1808  4              !+
;1736   1809  4              ! Interfacing with output's virtual displays.  Make sure user can
;1737   1810  4              ! not type beyond the display's boundaries by limiting the input
;1738   1811  4              ! size to the number of free columns (allowing room for prompt, if
;1739   1812  4              ! any).
;1740   1813  4              ! Notice that we need some info from SMG$$SET_PROMPT_STRING (prompt
;1741   1814  4              ! string length), and we need to provide it with some info (set cursor
;1742   1815  4              ! sequence, if any)   So get the info here and save it.
;1743   1816  4              !-
;1744   1817  4              LOCAL
;1745   1818  4                  OUT_STATUS;
;1746   1819  4              DISPLAY_ID_FLAG = 1;        ! for quick checks later
;1747   1820  4              IF .KCB [KCB_L_PASTEBOARD_ID] LSS 0 ! pb not established yet
;1748   1821  4              THEN
;1749   1822  5                  BEGIN
;1750   1823  5                  OUT_STATUS = SMG$$GET_PASTEBOARD_ID (%REF (.KCB [KCB_W_DEVNAM_LENGTH]),
;1751   1824  5                                                       KCB [KCB_T_DEVNAM_STRING],
;1752   1825  5                                                       KCB [KCB_L_PASTEBOARD_ID]);
;1753   1826  5                  IF NOT .OUT_STATUS THEN RETURN (.OUT_STATUS);
;1754   1827  4                  END;
;1755   1828  4              OUT_STATUS = SMG$$SET_PHYSICAL_CURSOR (.DISPLAY_ID,
;1756   1829  4                                                     KCB [KCB_L_PASTEBOARD_ID],
;1757   1830  4                                                     CURSOR_BUF,
;1758   1831  4                                                     CURSOR_BUF_LEN,
;1759   1832  4                                                     FREE_COLS);
;1760   1833  4              IF NOT .OUT_STATUS THEN RETURN .OUT_STATUS;
;1761   1834  3              END;                        ! display_id present
;1762   1835  3
;1763   1836  3          IF .DISPLAY_ID_FLAG
;1764   1837  3          THEN
;1765   1838  4              BEGIN       ! Batching check
;1766   1839  4
;1767   1840  4              LOCAL
;1768   1841  4                  DCB : REF BLOCK [,BYTE];
;1769   1842  4
;1770   1843  4              $SMG$GET_DCB (.DISPLAY_ID, DCB);
;1771   1844  4
;1772   1845  4              !+
;1773   1846  4              ! Don't allow input if display is batched.
;1774   1847  4              !-
;1775   1848  4
;1776   1849  4              IF .DCB[DCB_L_BATCH_LEVEL] NEQ 0
;1777   1850  4              THEN
;1778   1851  4                  RETURN  SMG$_ILLBATFNC;
;1779   1852  4
;1780   1853  3              END;        ! Batching check
;1781   1854  3
;1782   1855  3          !+
;1783   1856  3          ! Get prompt string, if any.
;1784   1857  3          !-
;1785   1858  3
;1786   1859  3              IF .DISPLAY_ID_FLAG OR
;1787   1860  3                 NOT NULLPARAMETER (K_PROMPT_STRING)
;1788   1861  3              THEN
;1789   1862  4                  BEGIN
;1790   1863  4                  LOCAL
;1791   1864  4                      STATUS,
```

```
;  1792     1865  4                    NULL_STRING : BLOCK [8,BYTE];
;  1793     1866  4
;  1794     1867  4                NULL_STRING [DSC$B_CLASS] = DSC$K_CLASS_S;
;  1795     1868  4                NULL_STRING [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  1796     1869  4                NULL_STRING [DSC$W_LENGTH] = 0;
;  1797     1870  4                NULL_STRING [DSC$A_POINTER] = UPLIT (' ');
;  1798     1871  4
;  1799     1872  4                STATUS = SMG$$SET_PROMPT_STRING (
;  1800     1873  5                                (IF NULLPARAMETER (K_PROMPT_STRING)
;  1801     1874  5                                 THEN
;  1802     1875  5                                     NULL_STRING
;  1803     1876  5                                 ELSE
;  1804     1877  4                                     PROMPT_STRING [0,0,0,0]),
;  1805     1878  4                                ITEM_LIST [K_PROMPT,0,0,0,0],
;  1806     1879  4                                .CURSOR_BUF_LEN,
;  1807     1880  4                                CURSOR_BUF);
;  1808     1881  4                IF NOT .STATUS
;  1809     1882  4                THEN
;  1810     1883  4                    RETURN .STATUS;
;  1811     1884  4                FLAGS [V_SETPROMPT] = 1;
;  1812     1885  3                END;
;  1813     1886  3
;  1814     1887  3            !+
;  1815     1888  3            ! Set up the $QIOW argument list.
;  1816     1889  3            !-
;  1817     1890  3
;  1818     1891  3            KCB [KCB_L_QIO1_FUNC] = IO$_READVBLK+IO$M_EXTEND;
;  1819     1892  3            KCB [KCB_L_QIO1_P1] = BUFFER;
;  1820     1893  3            KCB [KCB_L_QIO1_P2] = %ALLOCATION(BUFFER) - 4; ! Reserve 4 for term. echo
;  1821     1894  3            KCB [KCB_L_QIO1_P3] = 0;
;  1822     1895  3            KCB [KCB_L_QIO1_P4] = 0;
;  1823     1896  3            KCB [KCB_L_QIO1_P5] = ITEM_LIST;
;  1824     1897  3            KCB [KCB_L_QIO1_P6] = %ALLOCATION(ITEM_LIST) - 4; ! Extra longword added
;  1825     1898  3                                                              ! by $ITMLST_DECL.
;  1826     1899  3
;  1827     1900  3            IF .DISPLAY_ID_FLAG
;  1828     1901  3            THEN
;  1829     1902  4                    BEGIN
;  1830     1903  4                    IF PROMPT_STRING [0,0,0,0] NEQA 0
;  1831     1904  4                    THEN
;  1832     1905  4                        FREE_COLS = .FREE_COLS - .PROMPT_STRING [DSC$W_LENGTH];
;  1833     1906  4                    IF .FREE_COLS LEQ 0
;  1834     1907  4                    THEN
;  1835     1908  4                        !+
;  1836     1909  4                        ! No room to accept input.  Exit.
;  1837     1910  4                        !-
;  1838     1911  4                        RETURN (SMG$_INVCOL);       ! improve this message later
;  1839     1912  4
;  1840     1913  4                    IF .KCB [KCB_L_QIO1_P2] GTR .FREE_COLS
;  1841     1914  4                    THEN
;  1842     1915  4                        KCB [KCB_L_QIO1_P2] = .FREE_COLS;
;  1843     1916  3                    END;
;  1844     1917  3
;  1845     1918  3            !+
;  1846     1919  3            ! Start with a default state, indicate that the state lock is off and
;  1847     1920  3            ! get the pointer to the next open position in the buffer.
;  1848     1921  3            !-
```

```
;  1849          1922  3
;  1850          1923  3          TKEY_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  1851          1924  3          TKEY_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
;  1852          1925  3          TKEY_DSC [DSC$A_POINTER] = TKEY;
;  1853          1926  3          NEXT_POS = BUFFER;
;  1854          1927  3
;  1855          1928  3          !+
;  1856          1929  3          ! Loop until a terminator is seen, composing the line.
;  1857          1930  3          !-
;  1858          1931  3
;  1859          1932  3          WHILE 1 DO
;  1860          1933  4              BEGIN
;  1861          1934  4
;  1862          1935  4              !+
;  1863          1936  4              ! Prevent last portion of line from echoing unless later
;  1864          1937  4              ! shown to be necessary.
;  1865          1938  4              !-
;  1866          1939  4
;  1867          1940  4              FLAGS [V_SKIP_LAST_ECHO] = 1;
;  1868          1941  4
;  1869          1942  4              !+
;  1870          1943  4              ! Read the next part of the line.
;  1871          1944  4              !-
;  1872          1945  4
;  1873          1946  5                  BEGIN
;  1874          1947  5                  LOCAL
;  1875          1948  5                      QIO_STATUS;
;  1876          1949  5                  QIO_STATUS = CALLG (KCB [KCB_R_QIO1], SYS$QIOW);
;  1877          1950  5                  IF .QIO_STATUS
;  1878          1951  5                  THEN
;  1879          1952  5                      QIO_STATUS = .KCB [KCB_W_IOSB_STATUS];
;  1880          1953  5                  IF .KCB [KCB_W_IOSB_STATUS] EQLU SS$_PARTESCAPE
;  1881          1954  5                      THEN
;  1882          1955  6                          BEGIN
;  1883          1956  6                          !+
;  1884          1957  6                          ! Check for partial escape sequence in the buffer.  This can
;  1885          1958  6                          ! happen if the QIO was limited to 2 characters and the input
;  1886          1959  6                          ! string was terminated by a keypad key (which are 3 chars).
;  1887          1960  6                          !-
;  1888          1961  6
;  1889          1962  6                          LOCAL
;  1890          1963  6                              QIO_RESULT,
;  1891          1964  6                              CHAR_BUF : BYTE,
;  1892          1965  6                              END_ESC_FLAG : INITIAL (0);
;  1893          1966  6
;  1894          1967  6                          BUFFER_LEN = .KCB [KCB_W_IOSB_COUNT] + .KCB [KCB_B_IOSB_TERMLEN];
;  1895          1968  6                                       ! point past data & partial terminator
;  1896          1969  6
;  1897          1970  6                          WHILE .END_ESC_FLAG EQL 0 DO
;  1898          1971  7                          BEGIN
;  1899        P 1972  7                              QIO_RESULT = $QIOW (EFN = .KCB [KCB_L_EFN],
;  1900        P 1973  7                                      CHAN = .KCB [KCB_W_CHANNEL],
;  1901        P 1974  7                                      FUNC = (IO$_READVBLK + IO$M_ESCAPE +
;  1902        P 1975  7                                              IO$M_NOECHO + IO$M_NOFILTR +
;  1903        P 1976  7                                              IO$M_TRMNOECHO),
;  1904        P 1977  7                                      P1 = CHAR_BUF,
;  1905        P 1978  7                                      P2 = 1,
```

```
;1906   1979  7                                P4 = .KCB [KCB_L_QIO1_P4]);
;1907   1980  7                                     ! P4 = defined terminator set
;1908   1981  7                            IF NOT .QIO_RESULT THEN RET_STATUS = .QIO_RESULT;
;1909   1982  7
;1910   1983  7                            !+
;1911   1984  7                            ! Move terminator characters to buffer.
;1912   1985  7                            !-
;1913   1986  7
;1914   1987  7                            BUFFER [.BUFFER_LEN] = .CHAR_BUF;
;1915   1988  7                            BUFFER_LEN = .BUFFER_LEN + 1;
;1916   1989  7                            KCB [KCB_B_IOSB_TERMLEN] = .KCB [KCB_B_IOSB_TERMLEN] + 1;
;1917   1990  7
;1918   1991  7                            CASE .CHAR_BUF FROM %C'A' TO %C'~' OF
;1919   1992  7                            SET
;1920   1993  7                                [%C'A' TO %C'D',        ! arrow keys
;1921   1994  7                                %C'M',                  ! Enter
;1922   1995  7                                %C'P' to %C'S',         ! PFn keys
;1923   1996  7                                %C'l' to %C'n',         ! keypad - ,
;1924   1997  7                                %C'p' to %C'y',         ! keypad 0-9
;1925   1998  7                                %C'~']:                 ! function keys F6-F20 (no codes for
;1926   1999  7
;1927   2000  7                                    END_ESC_FLAG = 1;
;1928   2001  7
;1929   2002  7                                [INRANGE, OUTRANGE]:
;1930   2003  7                                    ;                   ! didn't find end of esc terminator
;1931   2004  7                                                        ! keep searching
;1932   2005  7                                TES;
;1933   2006  7
;1934   2007  6                            END;                        ! end of WHILE loop
;1935   2008  6
;1936   2009  6                            QIO_STATUS = .QIO_RESULT;
;1937   2010  6                                                        ! get rid of partescape error
;1938   2011  6
;1939   2012  5                            END;                        ! end of partial ESC terminator
;1940   2013  5
;1941   2014  5                        IF .QIO_STATUS NEQU SS$_NORMAL
;1942   2015  5                        THEN
;1943   2016  6                            BEGIN
;1944   2017  6                            IF NOT .QIO_STATUS
;1945   2018  6                            THEN
;1946   2019  7                                BEGIN
;1947   2020  7                                IF .QIO_STATUS EQLU SS$_BADESCAPE
;1948   2021  7                                THEN
;1949   2022  7                                    KCB [KCB_B_IOSB_TERMLEN] = 0
;1950   2023  7                                ELSE
;1951   2024  8                                    BEGIN
;1952   2025  8                                    RET_STATUS = .QIO_STATUS;
;1953   2026  8                                    EXITLOOP;
;1954   2027  7                                    END;
;1955   2028  7                                END
;1956   2029  6                            ELSE IF (.QIO_STATUS EQLU SS$_CONTROLY) OR
;1957   2030  7                                    (.QIO_STATUS EQLU SS$_CONTROLC)
;1958   2031  6                            THEN
;1959   2032  7                                BEGIN
;1960   2033  7                                ITEM_LIST [K_INIOFFSET, ITM$L_BUFADR] = 1;
;1961   2034  7                                ITEM_LIST [K_INISTRNG, ITM$W_BUFSIZ] = 0;
;1962   2035  7                                FLAGS [V_SKIP_NEWLINE] = 1;     ! Indicate that NEWLINE should not echo
```

```
;  1963     2036  7                        EXITLOOP;
;  1964     2037  7                        END
;  1965     2038  7
;  1966     2039  5                    END;
;  1967     2040  4                END;
;  1968     2041  4
;  1969     2042  4            !+
;  1970     2043  4            ! Set the initial offset to indicate the position (1-origin)
;  1971     2044  4            ! where the next read should start.  Set NEXT_POS to that
;  1972     2045  4            ! location in the buffer.  Note that NEXT_POS will then point
;  1973     2046  4            ! to the start of the terminator.  Set the initial string to
;  1974     2047  4            ! what has been typed so far.
;  1975     2048  4            !-
;  1976     2049  4
;  1977     2050  5                BEGIN
;  1978     2051  5                LOCAL
;  1979     2052  5                    NEXT_INDEX; ! Index to next available position
;  1980     2053  5                NEXT_INDEX = .KCB [KCB_W_IOSB_COUNT];
;  1981     2054  5                ITEM_LIST [K_INIOFFSET, ITM$L_BUFADR] = .NEXT_INDEX + 1;
;  1982     2055  5                ITEM_LIST [K_INISTRNG, ITM$W_BUFSIZ] = .NEXT_INDEX;
;  1983     2056  5                NEXT_POS = BUFFER [.NEXT_INDEX];
;  1984     2057  5
;  1985     2058  5                !+
;  1986     2059  5                ! If the buffer is now full, exit immediately.
;  1987     2060  5                !-
;  1988     2061  5
;  1989     2062  5                IF .NEXT_INDEX GEQU .KCB [KCB_L_QIO1_P2] ! Buffer size
;  1990     2063  5                THEN
;  1991     2064  5                    EXITLOOP;
;  1992     2065  4                END;
;  1993     2066  4
;  1994     2067  4            !+
;  1995     2068  4            ! If the terminator was <CR>, exit immediately.
;  1996     2069  4            !-
;  1997     2070  4
;  1998     2071  4            IF CH$RCHAR(.NEXT_POS) EQLU K_CR
;  1999     2072  4            THEN
;  2000     2073  5                BEGIN
;  2001     2074  5                FLAGS [V_SKIP_LAST_ECHO] = 1;
;  2002     2075  5                EXITLOOP;
;  2003     2076  4                END;
;  2004     2077  4
;  2005     2078  4            !+
;  2006     2079  4            ! Get the terminator name, if any.
;  2007     2080  4            !-
;  2008     2081  4
;  2009     2082  4            KEY_CODE = SMG$$TERM_TO_KEYCODE (.NEXT_POS, .KCB [KCB_B_IOSB_TERMLEN]);
;  2010     2083  4
;  2011     2084  4            !+
;  2012     2085  4            ! Reset state to default if needed.  Cause state to be reset
;  2013     2086  4            ! next time unless disabled later.
;  2014     2087  4            !-
;  2015     2088  4
;  2016     2089  4            IF TESTBITSS (FLAGS [V_RESET_STATE])
;  2017     2090  4            THEN
;  2018     2091  5                BEGIN
;  2019     2092  5                TKEY_DSC_PTR = KTH [KTH_R_DEF_STATE_DESCR];
```

```
;  2020     2093  5                  TKEY_POS = .KTH [KTH_A_DEF_KEYCODE];
;  2021     2094  4                  END;
;  2022     2095  4
;  2023     2096  4              !+
;  2024     2097  4              ! Construct TKEY.
;  2025     2098  4              !-
;  2026     2099  4
;  2027     2100  4              TKEY_POS [0] = .KEY_CODE;
;  2028     2101  4
;  2029     2102  4              !+
;  2030     2103  4              ! Look up key to see if it is defined as anything.
;  2031     2104  4              !-
;  2032     2105  4
;  2033     2106  5              IF (SMG$$LOOKUP_KEY (.KTH, .TKEY_DSC_PTR; KDE))
;  2034     2107  4              THEN
;  2035     2108  5                  BEGIN
;  2036     2109  5                  LOCAL
;  2037     2110  5                      NEW_LENGTH: WORD;
;  2038     2111  5                  NEW_LENGTH = .ITEM_LIST [K_INISTRNG, ITM$W_BUFSIZ] +
;  2039     2112  5                      .KDE [KDE_W_EQUIV_LENGTH];
;  2040     2113  5                  IF .NEW_LENGTH [EQU 512
;  2041     2114  5                  THEN
;  2042     2115  6                      BEGIN
;  2043     2116  6                      LOCAL
;  2044     2117  6                          CURSOR_POS;     ! Position of cursor
;  2045     2118  6
;  2046     2119  6                      !+
;  2047     2120  6                      ! Append equivalence string to initial string.
;  2048     2121  6                      ! Current cursor position may not be at end of string.
;  2049     2122  6                      !-
;  2050     2123  6
;  2051     2124  6                      ITEM_LIST [K_INISTRNG, ITM$W_BUFSIZ] = .NEW_LENGTH;
;  2052     2125  6
;  2053     2126  6                      !+
;  2054     2127  6                      ! If NOECHO, add the equivalence string size
;  2055     2128  6                      ! to the initial string offset so that we don't
;  2056     2129  6                      ! write it out.
;  2057     2130  6                      !+
;  2058     2131  6
;  2059     2132  6                      IF .KDE [KDE_V_NOECHO]
;  2060     2133  6                      THEN
;  2061     2134  6                          ITEM_LIST [K_INIOFFSET, ITM$L_BUFADR] =
;  2062     2135  6                              .ITEM_LIST [K_INIOFFSET, ITM$L_BUFADR] +
;  2063     2136  6                              .KDE [KDE_W_EQUIV_LENGTH]
;  2064     2137  6                      ELSE
;  2065     2138  6                          FLAGS [V_SKIP_LAST_ECHO] = 0;
;  2066     2139  6
;  2067     2140  6                      CURSOR_POS = .NEXT_POS - .KCB [KCB_B_ICSB_POS];
;  2068     2141  6                      IF .CURSOR_POS EQLA .NEXT_POS
;  2069     2142  6                      THEN
;  2070     2143  7                          BEGIN
;  2071     2144  7                          !+
;  2072     2145  7                          ! Cursor is at end of line.
;  2073     2146  7                          !-
;  2074     2147  7                          NEXT_POS = CH$MOVE (.KDE [KDE_W_EQUIV_LENGTH],
;  2075     2148  7                              .KDE [KDE_A_EQUIV_POINTER], .NEXT_POS);
;  2076     2149  7
```

```
;2077  2150  7                        END
;2078  2151  6                    ELSE
;2079  2152  7                        BEGIN
;2080  2153  7                        !+
;2081  2154  7                        ! Cursor is not at end of line.  Move remainder of
;2082  2155  7                        ! line over and reset offset to cursor position.
;2083  2156  7                        !-
;2084  2157  7
;2085  2158  7                        NEXT_POS = CH$MOVE ((.NEXT_POS-.CURSOR_POS), .CURSOR_POS,
;2086  2159  7                            (.CURSOR_POS+.KDE [KDE_W_EQUIV_LENGTH]));
;2087  2160  7                        CH$MOVE (.KDE [KDE_W_EQUIV_LENGTH],
;2088  2161  7                            .KDE [KDE_A_EQUIV_POINTER], .CURSOR_POS);
;2089  2162  7
;2090  2163  7                        !+
;2091  2164  7                        ! If not NOECHO, move the initial offset back to where
;2092  2165  7                        ! we need to display characters.
;2093  2166  7                        !-
;2094  2167  7
;2095  2168  7                        IF NOT .KDE [KDE_V_NOECHO]
;2096  2169  7                        THEN
;2097  2170  7                            ITEM_LIST [K_INIOFFSET, ITM$L_BUFADR] =
;2098  2171  7                                (.CURSOR_POS-BUFFER)+1;
;2099  2172  6                        END;
;2100  2173  5                    END;
;2101  2174  5
;2102  2175  5                !+
;2103  2176  5                ! If this key sets a new state, store it.
;2104  2177  5                !-
;2105  2178  5
;2106  2179  5                IF .KDE [KDE_V_SETSTATE]
;2107  2180  5                THEN
;2108  2181  6                    BEGIN
;2109  2182  6                    IF .KDE [KDE_V_LOCK]        ! Locks new state?
;2110  2183  6                    THEN
;2111  2184  6                        !+
;2112  2185  6                        ! Select the default state string as the
;2113  2186  6                        ! the basis for TKEY.
;2114  2187  6                        !-
;2115  2188  6                        TKEY_DSC_PTR = KTH [KTH_R_DEF_STATE_DESCR]
;2116  2189  6                    ELSE
;2117  2190  7                        BEGIN
;2118  2191  7                        !+
;2119  2192  7                        ! Use local TKEY descriptor.  Cause state to not
;2120  2193  7                        ! get reset to default on next iteration.
;2121  2194  7                        !-
;2122  2195  7                        TKEY_DSC_PTR = TKEY_DSC;
;2123  2196  7                        FLAGS [V_RESET_STATE] = 0;
;2124  2197  6                        END;
;2125  2198  6
;2126  2199  6                    TKEY_POS = CH$MOVE (.KDE [KDE_W_STATE_LENGTH],
;2127  2200  6                        .KDE [KDE_A_STATE_POINTER],
;2128  2201  6                        .TKEY_DSC_PTR [DSC$A_POINTER]);
;2129  2202  6                    CH$WCHAR_A (0, TKEY_POS);
;2130  2203  6                    IF .KDE [KDE_V_LOCK]
;2131  2204  6                    THEN
;2132  2205  6                        KTH [KTH_A_DEF_KEYCODE] = .TKEY_POS;
;2133  2206  6                    TKEY_DSC_PTR [DSC$W_LENGTH] = .KDE [KDE_W_STATE_LENGTH] + 3;
```

```
;  2134     2207  5                    END;
;  2135     2208  5
;  2136     2209  5                !+
;  2137     2210  5                ! If this terminates the read, exit the loop.
;  2138     2211  5                !-
;  2139     2212  5
;  2140     2213  5                IF .KDE [KDE_V_TERMINATE]
;  2141     2214  5                THEN
;  2142     2215  5                    EXITLOOP;
;  2143     2216  5
;  2144     2217  5                END
;  2145     2218  4            ELSE
;  2146     2219  5                BEGIN
;  2147     2220  5
;  2148     2221  5                !+
;  2149     2222  5                ! Key is not defined.
;  2150     2223  5                !-
;  2151     2224  5
;  2152     2225  5                LOCAL
;  2153     2226  5                    CURSOR_POS;
;  2154     2227  5
;  2155     2228  5                !+
;  2156     2229  5                ! Cursor position must be moved to the end of the line.
;  2157     2230  5                !-
;  2158     2231  5
;  2159     2232  5                CURSOR_POS = .NEXT_POS - .KCB [KCB_B_IOSB_POS];
;  2160     2233  5                ITEM_LIST [K_INIOFFSET, ITM$L_BUFADR] =
;  2161     2234  5                    (.CURSOR_POS-BUFFER)+1;
;  2162     2235  5
;  2163     2236  5                !+
;  2164     2237  5                ! If it is ^Z, then terminate anyway.
;  2165     2238  5                !-
;  2166     2239  5
;  2167     2240  5                IF CH$RCHAR (.NEXT_POS) EQLU K_CTRLZ
;  2168     2241  5                THEN
;  2169     2242  6                    BEGIN
;  2170     2243  6
;  2171     2244  6                    !+
;  2172     2245  6                    ! If ^Z was the first character in the line,
;  2173     2246  6                    ! return SMG$_EOF now.  Otherwise, record the
;  2174     2247  6                    ! fact that ^Z was typed for use on the next read.
;  2175     2248  6                    !-
;  2176     2249  6                    IF .KCB [KCB_W_IOSB_COUNT] EQL 0
;  2177     2250  6                    THEN
;  2178     2251  6                        RET_STATUS = SMG$_EOF    ! End-of-file
;  2179     2252  6                    ELSE
;  2180     2253  6                        KCB [KCB_V_CTRLZ] = 1;
;  2181     2254  6
;  2182     2255  6                    FLAGS [V_ECHO_CTRLZ] = 1;
;  2183     2256  6
;  2184     2257  6                    EXITLOOP;
;  2185     2258  6                    END
;  2186     2259  6
;  2187     2260  6                !+
;  2188     2261  6                ! Else if the terminator was ENTER or DO and wasn't defined,
;  2189     2262  6                ! treat it as <CR>.
;  2190     2263  6                !-
```

```
;  2191     2264  6
;  2192     2265  5                ELSE IF (.KEY_CODE EQL SMG$K_TRM_ENTER) OR
;  2193     2266  6                        (.KEY_CODE EQL SMG$K_TRM_DO)
;  2194     2267  5                THEN
;  2195     2268  6                    BEGIN
;  2196     2269  6                    FLAGS [V_SKIP_LAST_ECHO] = 1;
;  2197     2270  6                    EXITLOOP;
;  2198     2271  5                    END;
;  2199     2272  5
;  2200     2273  4                END;
;  2201     2274  4
;  2202     2275  3            END;
;  2203     2276  3
;  2204     2277  3        !+
;  2205     2278  3        ! Do a final read so that the last equivalence characters
;  2206     2279  3        ! (if any) will echo.  Do the read with a zero-timeout so
;  2207     2280  3        ! that we don't get any more characters.
;  2208     2281  3        !-
;  2209     2282  3
;  2210     2283  3        IF NOT .FLAGS [V_SKIP_LAST_ECHO]
;  2211     2284  3        THEN
;  2212     2285  4            BEGIN
;  2213     2286  4            KCB [KCB_L_QIO1_P2] = .ITEM_LIST [K_INISTRNG, ITM$W_BUFSIZ];
;  2214     2287  4            ITEM_LIST [K_TERM, ITM$W_ITMCOD] = TRM$_TIMEOUT;
;  2215     2288  4            ITEM_LIST [K_TERM, ITM$W_BUFSIZ] = 0;
;  2216     2289  4            ITEM_LIST [K_TERM, ITM$L_BUFADR] = 0;
;  2217     2290  4            CALLG (KCB [KCB_R_QIO1], SYS$QIOW);
;  2218     2291  3            END;
;  2219     2292  3
;  2220     2293  3        !+
;  2221     2294  3        ! If DISPLAY_ID was passed, call back output side to tell it
;  2222     2295  3        ! what changed on the screen as a result of this prompt & input.
;  2223     2296  3        !-
;  2224     2297  3
;  2225     2298  3        IF .DISPLAY_ID_FLAG
;  2226     2299  3        THEN
;  2227     2300  4            BEGIN                                                          ; F
;  2228     2301  4            LOCAL
;  2229     2302  4                OUT_STATUS;
;  2230     2303  4            IF PROMPT_STRING [0,0,0,0] NEQA 0
;  2231     2304  4            THEN
;  2232     2305  5                BEGIN                       ! report prompt string change
;  2233     2306  5                OUT_STATUS = SMG$$REPORT_CHANGE_REPLACE (.DISPLAY_ID,
;  2234     2307  5                                                        KCB [KCB_L_PASTEBOARD_ID],
;  2235     2308  5                                                        %REF (.PROMPT_STRING [DSC$W_LENGTH]),
;  2236     2309  5                                                        .PROMPT_STRING [DSC$A_POINTER]);
;  2237     2310  5                IF NOT .OUT_STATUS AND .OUT_STATUS NEQ SMG$_STRTERESC
;  2238     2311  5                THEN
;  2239     2312  5                    RETURN .OUT_STATUS;
;  2240     2313  4                END;                        ! report prompt string change
;  2241     2314  4
;  2242     2315  4            OUT_STATUS = SMG$$REPORT_CHANGE_REPLACE (.DISPLAY_ID,
;  2243     2316  4                                                    KCB [KCB_L_PASTEBOARD_ID],
;  2244     2317  4                                                    %REF (.KCB [KCB_W_IOSB_COUNT]),
;  2245     2318  4                                                    BUFFER);
;  2246     2319  4                                        ! report user's input string
;  2247     2320  4            IF NOT .OUT_STATUS THEN RETURN .OUT_STATUS;
```

```
;  2248     2321  3              END;
;  2249     2322  3
;  2250     2323  3          !+
;  2251     2324  3          ! If we need to echo for ^Z, do it.
;  2252     2325  3          !-
;  2253     2326  3
;  2254     2327  3          IF .FLAGS [V_ECHO_CTRLZ]
;  2255     2328  3          THEN
;  2256     2329  4              BEGIN
;  2257     2330  4              !+
;  2258     2331  4              ! Echo the appropriate sequence depending on whether this
;  2259     2332  4              ! is a DECCRT or not.
;  2260     2333  4              ! The 2 sequences can be found as translations of the
;  2261     2334  4              ! system logical name TTY$EXIT_STRING (set up by SYSGEN).
;  2262     2335  4              ! It has two equivalence strings. The first one is for
;  2263     2336  4              ! DECCRT's, the second one is for other terminals.
;  2264     2337  4              ! If this logical is not defined, we use internal echoes.
;  2265     2338  4              !-
;  2266     2339  4              BIND
;  2267     2340  4                  KCB_DEVDEPEND2 = KCB [KCB_L_DEVDEPEND2]: BLOCK [4, BYTE];
;  2268     2341  4
;  2269     2342  4              LOCAL
;  2270     2343  4                  STATUS,
;  2271     2344  4                  ITEMLIST        : VECTOR[4*3+1],
;  2272     2345  4                  DECCRT_LEN      : WORD,
;  2273     2346  4                  OTHER_LEN       : WORD,
;  2274     2347  4                  DECCRT_PTR,
;  2275     2348  4                  OTHER_PTR,
;  2276     2349  4                  SYSTEM_TABLE    : VECTOR[2],
;  2277     2350  4                  LOGICAL         : VECTOR[2],
;  2278     2351  4                  DECCRT_ECHO     : VECTOR[LNM$C_NAMLENGTH,BYTE],
;  2279     2352  4                  OTHER_ECHO      : VECTOR[LNM$C_NAMLENGTH,BYTE];
;  2280     2353  4
;  2281     2354  4              ITEMLIST[0]=LNM$_INDEX^16+4;
;  2282     2355  4              ITEMLIST[1]=UPLIT(0);
;  2283     2356  4              ITEMLIST[2]=0;
;  2284     2357  4
;  2285     2358  4              ITEMLIST[3]=LNM$_STRING^16+LNM$C_NAMLENGTH;
;  2286     2359  4              ITEMLIST[4]=DECCRT_ECHO;
;  2287     2360  4              ITEMLIST[5]=DECCRT_LEN;
;  2288     2361  4
;  2289     2362  4              ITEMLIST[6]=LNM$_INDEX^16+4;
;  2290     2363  4              ITEMLIST[7]=UPLIT(1);
;  2291     2364  4              ITEMLIST[8]=0;
;  2292     2365  4
;  2293     2366  4              ITEMLIST[9]=LNM$_STRING^16+LNM$C_NAMLENGTH;
;  2294     2367  4              ITEMLIST[10]=OTHER_ECHO;
;  2295     2368  4              ITEMLIST[11]=OTHER_LEN;
;  2296     2369  4
;  2297     2370  4              ITEMLIST[12]=0;
;  2298     2371  4
;  2299     2372  4              SYSTEM_TABLE[0]=%CHARCOUNT('LNM$SYSTEM_TABLE');
;  2300     2373  4              SYSTEM_TABLE[1]=UPLIT BYTE('LNM$SYSTEM_TABLE');
;  2301     2374  4              LOGICAL[0]=%CHARCOUNT('TTY$EXIT_STRING');
;  2302     2375  4              LOGICAL[1]=UPLIT BYTE('TTY$EXIT_STRING');
;  2303     2376  4
;  2304     2377  4              DECCRT_PTR=T_CTRLZ_DECCRT;
```

```
;  2305          2378  4              DECCRT_LEN=%ALLOCATION(T_CTRLZ_DECCRT);
;  2306          2379  4              OTHER_PTR=T_CTRLZ_OTHER;
;  2307          2380  4              OTHER_LEN=%ALLOCATION(T_CTRLZ_OTHER);
;  2308          2381  4
;  2309          2382  4              ! No change if logical name isn't there or
;  2310          2383  4              ! any other problem translating this name.
;  2311          2384  4
;  2312        L 2385  4          %IF NOT %VARIANT
;  2313          2386  4          %THEN
;  2314          2387  4
;  2315        P 2388  4              STATUS=$TRNLNM(TABNAM=SYSTEM_TABLE,
;  2316        P 2389  4                             LOGNAM=LOGICAL,
;  2317          2390  4                             ITMLST=ITEMLIST);
;  2318          2391  4              IF .STATUS
;  2319          2392  4              THEN
;  2320          2393  5                  BEGIN
;  2321          2394  5                  DECCRT_PTR=DECCRT_ECHO;
;  2322          2395  5                  OTHER_PTR=OTHER_ECHO;
;  2323          2396  4                  END;
;  2324          2397  4
;  2325          2398  4          %FI
;  2326          2399  4
;  2327          2400  4              KCB [KCB_L_QIO1_FUNC] = IO$_WRITEVBLK;
;  2328          2401  4              IF .KCB_DEVDEPEND2 [TT2$V_DECCRT]
;  2329          2402  4              THEN
;  2330          2403  5                  BEGIN
;  2331          2404  5                  KCB [KCB_L_QIO1_P1] = .DECCRT_PTR;
;  2332          2405  5                  KCB [KCB_L_QIO1_P2] = .DECCRT_LEN;
;  2333          2406  5                  END
;  2334          2407  4              ELSE
;  2335          2408  5                  BEGIN
;  2336          2409  5                  KCB [KCB_L_QIO1_P1] = .OTHER_PTR;
;  2337          2410  5                  KCB [KCB_L_QIO1_P2] = .OTHER_LEN;
;  2338          2411  4                  END;
;  2339          2412  4
;  2340          2413  4              CALLG (KCB [KCB_R_QIO1], SYS$QIOW);
;  2341          2414  3              END;
;  2342          2415  3
;  2343          2416  3          !+
;  2344          2417  3          ! Echo NEWLINE, if needed.
;  2345          2418  3          !-
;  2346          2419  3
;  2347          2420  3          IF NOT .FLAGS [V_SKIP_NEWLINE]
;  2348          2421  3          THEN
;  2349          2422  4              BEGIN
;  2350          2423  4              KCB [KCB_L_QIO1_FUNC] = IO$_WRITEVBLK+IO$M_NEWLINE;
;  2351          2424  4              KCB [KCB_L_QIO1_P1] = T_CR;
;  2352          2425  4              KCB [KCB_L_QIO1_P2] = 1;
;  2353          2426  4              CALLG (KCB [KCB_R_QIO1], SYS$QIOW);
;  2354          2427  3              END;
;  2355          2428  3
;  2356          2429  3          !+
;  2357          2430  3          ! Set BUFFER_PTR and BUFFER_LEN.
;  2358          2431  3          !-
;  2359          2432  3
;  2360          2433  3          BUFFER_PTR = BUFFER;
;  2361          2434  3          BUFFER_LEN = .ITEM_LIST [K_INISTRNG, ITMSW_BUFSIZ];
```

```
;  2362          2435  3
;  2363          2436  3          !+
;  2364          2437  3          ! Delete the prompt string, if any.
;  2365          2438  3          !-
;  2366          2439  3
;  2367          2440  3          IF .FLAGS [V_SETPROMPT]
;  2368          2441  3          THEN
;  2369          2442  3              SMG$$DELETE_PROMPT_STRING (ITEM_LIST [K_PROMPT,0,0,0,0]);
;  2370          2443  2          END;
;  2371          2444  2
;  2372          2445  2      !+
;  2373          2446  2      ! We now have the complete line in BUFFER, with length BUFFER_LEN.
;  2374          2447  2      ! Copy it to the user's string.
;  2375          2448  2      !-
;  2376          2449  2
;  2377          2450  2      COPY_STATUS = LIB$SCOPY_R_DX6 (.BUFFER_LEN,
;  2378          2451  2          .BUFFER_PTR, OUT_LINE [0,0,0,0]);
;  2379          2452  2      IF NOT .COPY_STATUS
;  2380          2453  2      THEN
;  2381          2454  2          RETURN .COPY_STATUS;
;  2382          2455  2
;  2383          2456  2      !+
;  2384          2457  2      ! Fill in OUT_LENGTH if requested.
;  2385          2458  2      !-
;  2386          2459  2
;  2387          2460  2      IF NOT NULLPARAMETER (K_OUT_LENGTH)
;  2388          2461  2      THEN
;  2389          2462  3          BEGIN
;  2390          2463  3          LOCAL
;  2391          2464  3              LINE_LENGTH: WORD;
;  2392          2465  3          LIB$ANALYZE_SDESC_R2 (OUT_LINE [0,0,0,0]; LINE_LENGTH);
;  2393          2466  3          IF .LINE_LENGTH LSSU .BUFFER_LEN
;  2394          2467  3          THEN
;  2395          2468  3              BUFFER_LEN = .LINE_LENGTH;
;  2396          2469  3          OUT_LENGTH [0] = .BUFFER_LEN;
;  2397          2470  2          END;
;  2398          2471  2
;  2399          2472  2      RETURN .RET_STATUS;
;  2400          2473  2
;  2401          2474  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$$GET_PASTEBOARD_ID                      ;  1825
                            17              50  E9 000FC      BLBC    OUT_STATUS, T1$                                 ;  1826
                                20      AE  9F 000FF 10$:     PUSHAB  FREE_COLS                                       ;  1829
                                28      AE  9F 00102          PUSHAB  CURSOR_BUF_LEN
                              FD80      CD  9F 00105          PUSHAB  CURSOR_BUF
                                F8      A7  9F 00109          PUSHAB  -8(KCB)
                                18      AC  DD 0010C          PUSHL   DISPLAY_ID
                00000000G   00              05  FB 0010F      CALLS   #5, SMG$$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$:     MOVL    #^o94720, NULL_STRING                       ;  1869
                    0274    CE      FE75    CF  9E 0015C      MOVAB   P.AAE, NULL_STRING+4                        ;  1870
                                    0278    CE  9F 00163      PUSHAB  CURSOR_BUF                                  ;  1878
                                      28    AE  DD 00167      PUSHL   CURSOR_BUF_LEN                              ;  1879
                                    FD9C    CD  9F 0016A      PUSHAB  ITEM_LIST+12                                ;  1878
                            04              6C  91 0016E      CMPB    (AP), #4                                    ;  1873
                                            05  1F 00171      BLSSU   18$
                                    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                       ; 1988
                                        6E    A7  96 00256           INCB    110(KCB)                         ; 1989
                   3D         41    8F  28    AE  8F 00259           CASEB   CHAR_BUF, #65, #61               ; 1991
      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$,-                        ;
                                                                             34$-33$,-                        ;
                                                                             34$-33$,-                        ;
                                                                             30$-33$,-                        ;
                                                                             34$-33$,-                        ;
                                                                             34$-33$,-                        ;
                                                                             34$-33$,-                        ;
                                                                             34$-33$,-                        ;
                                                                             34$-33$,-                        ;
                                                                             34$-33$,-                        ;
                                                                             34$-33$,-                        ;
                                                                             34$-33$,-                        ;
```

```
                                                        34$-33$,-                          ;
                                                        34$-33$,-                          ;
                                                        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              52  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
                                  FFCA  31  00041F 54$:     BRW     27$                                             ; 2232
                        51      6F      A7  9A 00422 55$:     MOVZBL  111(KCB), CURSOR_POS
          51            04    AE        51  C3 00426          SUBL3   CURSOR_POS, NEXT_POS, CURSOR_POS
                        50    FDFC      CD  9E 0042B          MOVAB   BUFFER, R0                                      ; 2234
                        50              51  C2 00430          SUBL2   CURSOR_POS, R0
    FDB8    CD          01              50  C3 00433          SUBL3   R0, #1, ITEM_LIST+40
                        1A      04      BE  91 00439          CMPB    @NEXT_POS, #26                                  ; 2240
                                        19  12 0043D          BNEQ    58$
                                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$:    MOVL    #48, T2(KCB)                            ; 2400
                                   0B F7    A7        05  E1  00585         BBC     #5, -9(KCB), 66$                        ; 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    #1, 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
```

```
;2403          2475  1 %SBTTL 'SMG$DELETE_VIRTUAL_KEYBOARD'
;2404          2476  1 GLOBAL ROUTINE SMG$DELETE_VIRTUAL_KEYBOARD (
;2405          2477  1     KEYBOARD_ID: REF VECTOR [, LONG]
;2406          2478  1     ) =
;2407          2479  1
;2408          2480  1 !++
;2409          2481  1 ! FUNCTIONAL DESCRIPTION:
;2410          2482  1 !
;2411          2483  1 !       This routine deletes a virtual-keyboard which was created by a call
;2412          2484  1 !       to SMG$CREATE_VIRTUAL_KEYBOARD.  Any terminal attributes changed
;2413          2485  1 !       when the virtual-keyboard was created are reset to their previous
;2414          2486  1 !       values.
;2415          2487  1 !
;2416          2488  1 ! CALLING SEQUENCE:
;2417          2489  1 !
;2418          2490  1 !       RET_STATUS.wlc.v = SMG$DELETE_VIRTUAL_KEYBOARD (
;2419          2491  1 !                                   KEYBOARD_ID.r[.r)
;2420          2492  1 !
;2421          2493  1 ! FORMAL PARAMETERS:
;2422          2494  1 !
;2423          2495  1 !       KEYBOARD_ID     A longword containing the identification of the
;2424          2496  1 !                       virtual-keyboard which is to be deleted.
;2425          2497  1 !
;2426          2498  1 ! IMPLICIT INPUTS:
;2427          2499  1 !
;2428          2500  1 !       NONE
;2429          2501  1 !
;2430          2502  1 ! IMPLICIT OUTPUTS:
;2431          2503  1 !
;2432          2504  1 !       NONE
;2433          2505  1 !
;2434          2506  1 ! COMPLETION STATUS:
;2435          2507  1 !
;2436          2508  1 !       SS$_NORMAL      Normal successful completion
;2437          2509  1 !       SMG$_INVKBD_ID  Invalid keyboard-id
;2438          2510  1 !       SMG$_WRONUMARG  Wrong number of arguments
;2439          2511  1 !
;2440          2512  1 ! SIDE EFFECTS:
;2441          2513  1 !
;2442          2514  1 !       The keypad mode (numeric or application) is reset to its original state
;2443          2515  1 !       The terminal characteristics are reset to their original state
;2444          2516  1 !       The file is closed or the channel is deassigned.
;2445          2517  1 !
;2446          2518  1 !--
;2447          2519  1
;2448          2520  2     BEGIN
;2449          2521  2
;2450          2522  2     LOCAL
;2451          2523  2         KCB: REF KCB_R_KCB_STRUCT;
;2452          2524  2
;2453          2525  2     $SMG$VALIDATE_ARGCOUNT(1,1);
;2454          2526  2
;2455          2527  2     !+
;2456          2528  2     ! Fetch and validate KCB.
;2457          2529  2     !-
;2458          2530  2
;2459          2531  2     $SMG$VALIDATE_KCB (KEYBOARD_ID, KCB);
```

```
;  2460          2532  2
;  2461          2533  2      !+
;  2462          2534  2      ! Invalidate the user's KEYBOARD_ID.
;  2463          2535  2      !-
;  2464          2536  2
;  2465          2537  2      KEYBOARD_ID [0] = 0;
;  2466          2538  2
;  2467          2539  2      !+
;  2468          2540  2      ! Call SMG$$CLEANUP to do the actual close, etc.
;  2469          2541  2      ! Return with success.
;  2470          2542  2      !-
;  2471          2543  2
;  2472          2544  2      RETURN (SMG$$CLEANUP (KCB [KCB_R_KCB], SS$_NORMAL));
;  2473          2545  2
;  2474          2546  1      END;                            ! End of routine SMG$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
```

```
;2476    2547  1 %SBTTL 'SMG$SET_KEYPAD_MODE'
;2477    2548  1 GLOBAL ROUTINE SMG$SET_KEYPAD_MODE (
;2478    2549  1     KEYBOARD_ID: REF VECTOR [, LONG],
;2479    2550  1     KEYPAD_MODE: REF VECTOR [, LONG]
;2480    2551  1     ) =
;2481    2552  1
;2482    2553  1 !++
;2483    2554  1 ! FUNCTIONAL DESCRIPTION:
;2484    2555  1 !
;2485    2556  1 !       This procedure sets the state of the terminal's numeric keypad to either
;2486    2557  1 !       applications mode or numeric mode.  In applications mode, numeric
;2487    2558  1 !       keypad keys (which may also include punctuation keys) are considered
;2488    2559  1 !       function keys and are useable as terminators.  In numeric mode, these
;2489    2560  1 !       keys are equivalent to the keys with similar legends on the main
;2490    2561  1 !       key array and are not generally useable as terminators.
;2491    2562  1 !
;2492    2563  1 !       For VT5x, VT1xx (and compatible LAxx), and LK201 keyboards, the
;2493    2564  1 !       following table lists the legends on the keypad keys, their key name
;2494    2565  1 !       in applications mode, and their key value in numeric mode:
;2495    2566  1 !
;2496    2567  1 !               Legend          Applications        Numeric
;2497    2568  1 !               On Key             Mode              Mode
;2498    2569  1 !               ------          ------------        -------
;2499    2570  1 !                 0               KP0                 0
;2500    2571  1 !                 1               KP1                 1
;2501    2572  1 !                 2               KP2                 2
;2502    2573  1 !                 3               KP3                 3
;2503    2574  1 !                 4               KP4                 4
;2504    2575  1 !                 5               KP5                 5
;2505    2576  1 !                 6               KP6                 6
;2506    2577  1 !                 7               KP7                 7
;2507    2578  1 !                 8               KP8                 8
;2508    2579  1 !                 9               KP9                 9
;2509    2580  1 !                 -              MINUS                -
;2510    2581  1 !                 ,              COMMA                ,
;2511    2582  1 !                 .              PERIOD               .
;2512    2583  1 !             ENTER               ENTER              <CR>
;2513    2584  1 !
;2514    2585  1 !       Note that VT5x keyboards do not have all of these keys.
;2515    2586  1 !
;2516    2587  1 !       When a virtual-keyboard is created, the terminal is set by default
;2517    2588  1 !       to applications mode.  When the virtual-keyboard is deleted the
;2518    2589  1 !       state of the keypad is restored to its previous mode, as determined
;2519    2590  1 !       by the terminal characteristic TT2$V_APP_KEYPAD.
;2520    2591  1 !
;2521    2592  1 ! CALLING SEQUENCE:
;2522    2593  1 !
;2523    2594  1 !       RET_STATUS.wlc.v = SMG$SET_KEYPAD_MODE (
;2524    2595  1 !                                   KEYBOARD_ID.rl.r,
;2525    2596  1 !                                   NEW_MODE.rl.r)
;2526    2597  1 !
;2527    2598  1 ! FORMAL PARAMETERS:
;2528    2599  1 !
;2529    2600  1 !       KEYBOARD_ID     A longword containing the identification of the
;2530    2601  1 !                       virtual-keyboard for which the keypad mode is being
;2531    2602  1 !                       set or returned.
;2532    2603  1 !
```

```
;                                                                H 6
; 2533          2604  1 !       KEYPAD_MODE     A longword whose low bit is used to set the new
; 2534          2605  1 !                       state of the keypad mode.  If the low bit is set, the
; 2535          2606  1 !                       keypad is set into numeric mode.  If clear, the
; 2536          2607  1 !                       keypad is set into applications mode.  The default
; 2537          2608  1 !                       mode when a virtual-keyboard is created is
; 2538          2609  1 !                       applications mode.
; 2539          2610  1 !
; 2540          2611  1 ! IMPLICIT INPUTS:
; 2541          2612  1 !
; 2542          2613  1 !       NONE
; 2543          2614  1 !
; 2544          2615  1 ! IMPLICIT OUTPUTS:
; 2545          2616  1 !
; 2546          2617  1 !       NONE
; 2547          2618  1 !
; 2548          2619  1 ! COMPLETION STATUS:
; 2549          2620  1 !
; 2550          2621  1 !       SS$_NORMAL      Normal successful completion
; 2551          2622  1 !       SMG$_INVKBD_ID  Invalid keyboard-id
; 2552          2623  1 !       SMG$_WRONUMARG  Wrong number of arguments
; 2553          2624  1 !
; 2554          2625  1 ! SIDE EFFECTS:
; 2555          2626  1 !
; 2556          2627  1 !       NONE
; 2557          2628  1 !
; 2558          2629  1 !--
; 2559          2630  1
; 2560          2631  2     BEGIN
; 2561          2632  2
; 2562          2633  2     LOCAL
; 2563          2634  2         KCB: REF KCB_R_KCB_STRUCT;
; 2564          2635  2
; 2565          2636  2     BUILTIN
; 2566          2637  2         CALLG;
; 2567          2638  2
; 2568          2639  2     $SMG$VALIDATE_ARGCOUNT(2,2);
; 2569          2640  2
; 2570          2641  2     !+
; 2571          2642  2     ! Validate Keyboard Control Block
; 2572          2643  2     !-
; 2573          2644  2
; 2574          2645  2     $SMG$VALIDATE_KCB (KEYBOARD_ID, KCB);
; 2575          2646  2
; 2576          2647  2     !+
; 2577          2648  2     ! Only change the keypad mode if we're at a terminal.
; 2578          2649  2     !-
; 2579          2650  2
; 2580          2651  2     IF NOT .KCB [KCB_V_RMS]
; 2581          2652  2     THEN
; 2582          2653  3         BEGIN
; 2583          2654  3
; 2584          2655  3         BIND
; 2585          2656  3             KCB_DEVDEPEND2 = KCB [KCB_L_DEVDEPEND2]: BLOCK [4, BYTE];
; 2586          2657  3
; 2587          2658  3         !+
; 2588          2659  3         ! Select whether we want to set or clear applications mode.
; 2589          2660  3         !-
```

```
;2590    2661  3
;2591    2662  3          IF .KEYPAD_MODE [0]
;2592    2663  3          THEN
;2593    2664  4              BEGIN
;2594    2665  4              !+
;2595    2666  4              ! Set applications mode.
;2596    2667  4              !-
;2597    2668  4
;2598    2669  4              !+
;2599    2670  4              ! Mark the keypad as in applications mode.
;2600    2671  4              !-
;2601    2672  4
;2602    2673  4              KCB_DEVDEPEND2 [TT2$V_APP_KEYPAD] = 1;
;2603    2674  4
;2604    2675  4              !+
;2605    2676  4              ! Choose the proper control sequence to send.
;2606    2677  4              !-
;2607    2678  4
;2608    2679  4              IF .KCB [KCB_V_KPDSEQ_DECCRT]
;2609    2680  4              THEN
;2610    2681  5                  BEGIN
;2611    2682  5                  !+
;2612    2683  5                  ! Use sequence for DECCRT and VT5x.
;2613    2684  5                  !-
;2614    2685  5
;2615    2686  5                  KCB [KCB_L_QIO2_P1] = T_KPDAPP_DECCRT;
;2616    2687  5                  KCB [KCB_L_QIO2_P2] = %ALLOCATION(T_KPDAPP_DECCRT);
;2617    2688  5                  END
;2618    2689  4              ELSE SELECTONEU .KCB [KCB_B_DEVTYPE] OF
;2619    2690  4                  SET
;2620    2691  4                  [OTHERWISE]:
;2621    2692  5                      BEGIN
;2622    2693  5                      !+
;2623    2694  5                      ! Unknown type.
;2624    2695  5                      !-
;2625    2696  5                      KCB [KCB_L_QIO2_P1] = 0;
;2626    2697  5                      KCB [KCB_L_QIO2_P2] = 0;
;2627    2698  4                      END;
;2628    2699  4                  TES;
;2629    2700  4              END
;2630    2701  3          ELSE
;2631    2702  4              BEGIN
;2632    2703  4              !+
;2633    2704  4              ! Set numeric (non-application) mode.
;2634    2705  4              !-
;2635    2706  4
;2636    2707  4              !+
;2637    2708  4              ! Mark the keypad as in numeric mode.
;2638    2709  4              !-
;2639    2710  4
;2640    2711  4              KCB_DEVDEPEND2 [TT2$V_APP_KEYPAD] = 0;
;2641    2712  4
;2642    2713  4              !+
;2643    2714  4              ! Choose the proper control sequence to send.
;2644    2715  4              !-
;2645    2716  4
;2646    2717  4              IF .KCB [KCB_V_KPDSEQ_DECCRT]
```

```
;  2647     2718  4              THEN
;  2648     2719  5                  BEGIN
;  2649     2720  5                  !+
;  2650     2721  5                  ! Use sequence for DECCRT and VT5x.
;  2651     2722  5                  !-
;  2652     2723  5
;  2653     2724  5                  KCB [KCB_L_QIO2_P1] = T_KPDNUM_DECCRT;
;  2654     2725  5                  KCB [KCB_L_QIO2_P2] = %ALLOCATION(T_KPDNUM_DECCRT);
;  2655     2726  5                  END
;  2656     2727  4              ELSE SELECTONEU .KCB [KCB_B_DEVTYPE] OF
;  2657     2728  4                  SET
;  2658     2729  4                  [OTHERWISE]:
;  2659     2730  5                      BEGIN
;  2660     2731  5                      !+
;  2661     2732  5                      ! Unknown type.
;  2662     2733  5                      !-
;  2663     2734  5                      KCB [KCB_L_QIO2_P1] = 0;
;  2664     2735  5                      KCB [KCB_L_QIO2_P2] = 0;
;  2665     2736  4                      END;
;  2666     2737  4                  TES;
;  2667     2738  3              END;
;  2668     2739  3
;  2669     2740  3          !+
;  2670     2741  3          ! Write the sequence to the terminal.
;  2671     2742  3          !-
;  2672     2743  3
;  2673     2744  3          KCB [KCB_L_QIO2_FUNC] = IO$_WRITEVBLK+IO$M_NOFORMAT;
;  2674     2745  3          KCB [KCB_L_QIO2_P3] = 0;
;  2675     2746  3          KCB [KCB_L_QIO2_P4] = 0;
;  2676     2747  3          KCB [KCB_L_QIO2_P5] = 0;
;  2677     2748  3          KCB [KCB_L_QIO2_P6] = 0;
;  2678     2749  3          CALLG (KCB [KCB_R_QIO2], SYS$QIOW);
;  2679     2750  3
;  2680     2751  3          !+
;  2681     2752  3          ! Set the new characteristics.
;  2682     2753  3          !-
;  2683     2754  3
;  2684     2755  3          KCB [KCB_L_QIO2_FUNC] = IO$_SETMODE;
;  2685     2756  3          KCB [KCB_L_QIO2_P1] = KCB [KCB_R_TERMCHAR];
;  2686     2757  3          KCB [KCB_L_QIO2_P2] = KCB_S_TERMCHAR;
;  2687     2758  3          CALLG (KCB [KCB_R_QIO2], SYS$QIOW);
;  2688     2759  3
;  2689     2760  3          !+
;  2690     2761  3          ! Mark that the terminal characteristics have been changed.
;  2691     2762  3          !-
;  2692     2763  3
;  2693     2764  3          KCB [KCB_V_CHARS_CHANGED] = 1;
;  2694     2765  3
;  2695     2766  2          END;
;  2696     2767  2
;  2697     2768  2      RETURN SS$_NORMAL;
;  2698     2769  2
;  2699     2770  1      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 00005      CMPB    (AP), #2                                        ; 2639
                                                  08   13 00008      BEQL    1$                                              ;
                                    50 00000000G  8F   D0 0000A      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, -70(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
```

```
;  2701     2771  1 %SBTTL 'SMG$CANCEL_INPUT'
;  2702     2772  1 GLOBAL ROUTINE SMG$CANCEL_INPUT (
;  2703     2773  1     KEYBOARD_ID: REF VECTOR [, LONG]
;  2704     2774  1     ) =
;  2705     2775  1
;  2706     2776  1 !++
;  2707     2777  1 ! FUNCTIONAL DESCRIPTION:
;  2708     2778  1 !
;  2709     2779  1 !       This procedure causes immediate termination
;  2710     2780  1 !       of a terminal read-in-progress which was issued by
;  2711     2781  1 !       SMG$READ_STRING or SMG$READ_COMPOSED_LINE.  If the
;  2712     2782  1 !       virtual-keyboard is associated with an RMS file, this procedure
;  2713     2783  1 !       has no effect, since it is not possible to cancel an outstanding
;  2714     2784  1 !       RMS input operation.
;  2715     2785  1 !
;  2716     2786  1 ! CALLING SEQUENCE:
;  2717     2787  1 !
;  2718     2788  1 !       RET_STATUS.wlc.v = SMG$CANCEL_INPUT (
;  2719     2789  1 !                                   KEYBOARD_ID.rl.r)
;  2720     2790  1 !
;  2721     2791  1 ! FORMAL PARAMETERS:
;  2722     2792  1 !
;  2723     2793  1 !       KEYBOARD_ID     A longword containing the identification of the
;  2724     2794  1 !                       virtual keyboard for which the input is to be
;  2725     2795  1 !                       cancelled.
;  2726     2796  1 !
;  2727     2797  1 ! IMPLICIT INPUTS:
;  2728     2798  1 !
;  2729     2799  1 !       NONE
;  2730     2800  1 !
;  2731     2801  1 ! IMPLICIT OUTPUTS:
;  2732     2802  1 !
;  2733     2803  1 !       NONE
;  2734     2804  1 !
;  2735     2805  1 ! COMPLETION STATUS:
;  2736     2806  1 !
;  2737     2807  1 !       SS$_NORMAL      Normal successful completion
;  2738     2808  1 !       SS$_RMSNOTCAN   Success - RMS operation cannot be cancelled
;  2739     2809  1 !       SMG$_INVKBD_ID  Invalid keyboard-id
;  2740     2810  1 !       SMG$_WRONUMARG  Wrong number of arguments
;  2741     2811  1 !
;  2742     2812  1 ! SIDE EFFECTS:
;  2743     2813  1 !
;  2744     2814  1 !       NONE
;  2745     2815  1 !
;  2746     2816  1 !--
;  2747     2817  1
;  2748     2818  2     BEGIN
;  2749     2819  2
;  2750     2820  2     LOCAL
;  2751     2821  2         KCB: REF KCB_R_KCB_STRUCT,
;  2752     2822  2         CANCEL_STATUS;
;  2753     2823  2
;  2754     2824  2     $SMG$VALIDATE_ARGCOUNT(1,1);
;  2755     2825  2
;  2756     2826  2     !+
;  2757     2827  2     ! 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
```

```
;2780     2849  1 %SBTTL 'SMG$$SET_PROMPT_STRING'
;2781     2850  1 ROUTINE SMG$$SET_PROMPT_STRING (
;2782     2851  1     PROMPT_STRING: REF BLOCK [, BYTE],          ! Prompt string descriptor
;2783     2852  1     ITEM_LIST: REF $ITMLST_DECL (ITEMS=1),      ! Prompt string list item
;2784     2853  1     PRE_BUF_LEN,                                ! Pre-prompt buffer length
;2785     2854  1     PRE_BUFFER                                  ! Pre-prompt buffer
;2786     2855  1     ) =
;2787     2856  1
;2788     2857  1 !++
;2789     2858  1 ! FUNCTIONAL DESCRIPTION:
;2790     2859  1 !
;2791     2860  1 !       This procedure adds the necessary carriagecontrol to the user-specified
;2792     2861  1 !       prompt string and stores it in an item-list descriptor.
;2793     2862  1 !
;2794     2863  1 ! CALLING SEQUENCE:
;2795     2864  1 !
;2796     2865  1 !       ret_status.wlc.v = SMG$$SET_PROMPT_STRING (prompt_string.rt.dx,
;2797     2866  1 !                                                  item_list.ww.r,
;2798     2867  1 !                                                  pre_buf_len.rl.v,
;2799     2868  1 !                                                  pre_buffer.ra.r)
;2800     2869  1 !
;2801     2870  1 ! FORMAL PARAMETERS:
;2802     2871  1 !
;2803     2872  1 !       PROMPT_STRING   The descriptor of the prompt string.
;2804     2873  1 !
;2805     2874  1 !       ITEM_LIST       The QIO item-list entry for the prompt-string.
;2806     2875  1 !
;2807     2876  1 !       PRE_BUF_LEN     Length of buffer to insert before the prompt
;2808     2877  1 !                       string.  (Used when interfacing to virtual display.)
;2809     2878  1 !
;2810     2879  1 !       PRE_BUFFER      Address of string to insert before the prompt
;2811     2880  1 !                       string.  (Used when interfacing to virtual display.)
;2812     2881  1 !
;2813     2882  1 ! IMPLICIT INPUTS:
;2814     2883  1 !
;2815     2884  1 !       QPS_QUEUE       Queue of prompt strings
;2816     2885  1 !
;2817     2886  1 ! IMPLICIT OUTPUTS:
;2818     2887  1 !
;2819     2888  1 !       NONE
;2820     2889  1 !
;2821     2890  1 ! COMPLETION STATUS:
;2822     2891  1 !
;2823     2892  1 !       SS$_NORMAL      Normal successful completion
;2824     2893  1 !       Errors from LIB$GET_VM
;2825     2894  1 !
;2826     2895  1 ! SIDE EFFECTS:
;2827     2896  1 !
;2828     2897  1 !       None
;2829     2898  1 !
;2830     2899  1 !--
;2831     2900  1
;2832     2901  2     BEGIN
;2833     2902  2
;2834     2903  2     LOCAL
;2835     2904  2         STRING_LEN: WORD,               ! Length of PROMPT_STRING
;2836     2905  2         STRING_PTR,                     ! Pointer to PROMPT_STRING
```

```
;  2837     2906  2          ALLOC_LEN,                      ! Allocated string length
;  2838     2907  2          ALLOC_PTR,                      ! Pointer to allocated string
;  2839     2908  2          ANL_STATUS, VM_STATUS;          ! Status from called procedures
;  2840     2909  2
;  2841     2910  2      LITERAL
;  2842     2911  2          K_CR = 13;                      ! <CR>
;  2843     2912  2
;  2844     2913  2      BUILTIN
;  2845     2914  2          REMQUE;
;  2846     2915  2
;  2847     2916  2      !+
;  2848     2917  2      ! Analyze PROMPT_STRING to get the length and address.
;  2849     2918  2      !-
;  2850     2919  2
;  2851     2920  2      ANL_STATUS = LIB$ANALYZE_SDESC_R2 (PROMPT_STRING [0,0,0,0];
;  2852     2921  2          STRING_LEN, STRING_PTR);
;  2853     2922  2      IF NOT .ANL_STATUS
;  2854     2923  2      THEN
;  2855     2924  2          RETURN .ANL_STATUS;
;  2856     2925  2
;  2857     2926  2      ALLOC_LEN = .STRING_LEN + 1;
;  2858     2927  2      IF .ALLOC_LEN GTRU 65535
;  2859     2928  2      THEN
;  2860     2929  2          RETURN SMG$_PROTOOLON;  ! Prompt string is too long
;  2861     2930  2      ITEM_LIST [0,ITM$W_BUFSIZ] = .ALLOC_LEN;
;  2862     2931  2      ALLOC_PTR = 0;
;  2863     2932  2
;  2864     2933  2      !+
;  2865     2934  2      ! If ALLOC_LEN is small enough so that we can get the string from
;  2866     2935  2      ! our queue, do so.
;  2867     2936  2      !-
;  2868     2937  2
;  2869     2938  3      IF .ALLOC_LEN LEQU (K_QPS_SIZE - 10)
;  2870     2939  2      THEN
;  2871     2940  3          BEGIN
;  2872     2941  3          IF REMQUE (.QPS_QUEUE [0], ALLOC_PTR)
;  2873     2942  3          THEN
;  2874     2943  4              BEGIN
;  2875     2944  4              ALLOC_PTR = 0;      ! Failure
;  2876     2945  4              ALLOC_LEN = K_QPS_SIZE;     ! Allocate a full QPS entry.
;  2877     2946  3              END;
;  2878     2947  2          END;
;  2879     2948  2
;  2880     2949  2      !+
;  2881     2950  2      ! If pointer is still zero, we have to allocate the string with LIB$GET_VM.
;  2882     2951  2      !-
;  2883     2952  2
;  2884     2953  2      IF .ALLOC_PTR EQL 0
;  2885     2954  2      THEN
;  2886     2955  3          BEGIN
;  2887     2956  3          VM_STATUS = LIB$GET_VM (ALLOC_LEN, ITEM_LIST [0,ITM$L_BUFADR]);
;  2888     2957  3          IF NOT .VM_STATUS
;  2889     2958  3          THEN
;  2890     2959  3              RETURN .VM_STATUS;
;  2891     2960  3          ALLOC_PTR = .ITEM_LIST [0,ITM$L_BUFADR];
;  2892     2961  2          END;
;  2893     2962  2
```

```
;  2894          2963  2     !+
;  2895          2964  2     ! Store the prompt string with a leading CR for carriagecontrol.
;  2896          2965  2     !
;  2897          2966  2     ! If input is interfacing with output SMG$ routines (ie. a virtual
;  2898          2967  2     ! display present), use the caller's buffer instead of CR.  This
;  2899          2968  2     ! buffer should contain the set cursor sequence to establish the
;  2900          2969  2     ! desired cursor position.
;  2901          2970  2     !-
;  2902          2971  2
;  2903          2972  2     ITEM_LIST [0,ITM$L_BUFADR] = .ALLOC_PTR;
;  2904          2973  2     IF .PRE_BUF_LEN EQL 0
;  2905          2974  2     THEN
;  2906          2975  2         CH$WCHAR_A (K_CR, ALLOC_PTR)
;  2907          2976  2     ELSE
;  2908          2977  3         BEGIN
;  2909          2978  3         ITEM_LIST [0,ITM$W_BUFSIZ] = .STRING_LEN + .PRE_BUF_LEN;
;  2910          2979  3         ALLOC_PTR = CH$MOVE (.PRE_BUF_LEN, .PRE_BUFFER, .ALLOC_PTR);
;  2911          2980  2         END;
;  2912          2981  2     CH$MOVE (.STRING_LEN, .STRING_PTR, .ALLOC_PTR);
;  2913          2982  2     RETURN SS$_NORMAL;
;  2914          2983  2
;  2915          2984  1     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
```

```
;  2917     2985  1 %SBTTL 'SMG$$DELETE_PROMPT_STRING'
;  2918     2986  1 ROUTINE SMG$$DELETE_PROMPT_STRING (
;  2919     2987  1     ITEM_LIST: REF $ITMLST_DECL (ITEMS=1)       . Prompt string item list
;  2920     2988  1     ): NOVALUE =
;  2921     2989  1
;  2922     2990  1 !++
;  2923     2991  1 ! FUNCTIONAL DESCRIPTION:
;  2924     2992  1 !
;  2925     2993  1 !       This procedure deletes a prompt string, if any.
;  2926     2994  1 !
;  2927     2995  1 ! CALLING SEQUENCE:
;  2928     2996  1 !
;  2929     2997  1 !       SMG$$DELETE_PROMPT_STRING (item_list.rr.r)
;  2930     2998  1 !
;  2931     2999  1 ! FORMAL PARAMETERS:
;  2932     3000  1 !
;  2933     3001  1 !       ITEM_LIST       The QIO item-list entry for the prompt string.
;  2934     3002  1 !
;  2935     3003  1 ! IMPLICIT INPUTS:
;  2936     3004  1 !
;  2937     3005  1 !       NONE
;  2938     3006  1 !
;  2939     3007  1 ! IMPLICIT OUTPUTS:
;  2940     3008  1 !
;  2941     3009  1 !       QPS_QUEUE
;  2942     3010  1 !
;  2943     3011  1 ! COMPLETION STATUS:
;  2944     3012  1 !
;  2945     3013  1 !       NONE
;  2946     3014  1 !
;  2947     3015  1 ! SIDE EFFECTS:
;  2948     3016  1 !
;  2949     3017  1 !       Either inserts string on queue of prompt-strings or deallocates
;  2950     3018  1 !       the string.
;  2951     3019  1 !
;  2952     3020  1 !--
;  2953     3021  1
;  2954     3022  2     BEGIN
;  2955     3023  2
;  2956     3024  2     LOCAL
;  2957     3025  2         ALLOC_LEN;
;  2958     3026  2
;  2959     3027  2     BUILTIN
;  2960     3028  2         INSQUE;
;  2961     3029  2
;  2962     3030  2     !+
;  2963     3031  2     ! If the string length is LEQ K_QPS_SIZE, insert it on the queue,
;  2964     3032  2     ! as the allocated length is guaranteed to be at least K_QPS_SIZE.
;  2965     3033  2     !-
;  2966     3034  2
;  2967     3035  2     ALLOC_LEN = .ITEM_LIST [0,ITM$W_BUFSIZ];
;  2968     3036  2     IF .ALLOC_LEN LEQU K_QPS_SIZE
;  2969     3037  2     THEN
;  2970     3038  2         INSQUE (.ITEM_LIST [0,ITM$L_BUFADR], QPS_QUEUE)
;  2971     3039  2     ELSE
;  2972     3040  2         LIB$FREE_VM (ALLOC_LEN, ITEM_LIST [0,ITM$L_BUFADR]);
;  2973     3041  2
```

```
;  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
```

```
;  2978     3045  1 %SBTTL 'SMG$$INIT_QUEUES'
;  2979     3046  1 ROUTINE SMG$$INIT_QUEUES
;  2980     3047  1     =
;  2981     3048  1
;  2982     3049  1 !++
;  2983     3050  1 ! FUNCTIONAL DESCRIPTION:
;  2984     3051  1 !
;  2985     3052  1 !       This procedure initializes KCB_QUEUE and QPS_QUEUE, and declares
;  2986     3053  1 !       an exit handler.
;  2987     3054  1 !
;  2988     3055  1 ! CALLING SEQUENCE:
;  2989     3056  1 !
;  2990     3057  1 !       ret_status.wlc.v = SMG$$INIT_QUEUES ()
;  2991     3058  1 !
;  2992     3059  1 ! FORMAL PARAMETERS:
;  2993     3060  1 !
;  2994     3061  1 !       NONE
;  2995     3062  1 !
;  2996     3063  1 ! IMPLICIT INPUTS:
;  2997     3064  1 !
;  2998     3065  1 !       V_QUEUE_INIT
;  2999     3066  1 !       KCB_QUEUE
;  3000     3067  1 !       QPS_QUEUE
;  3001     3068  1 !
;  3002     3069  1 ! IMPLICIT OUTPUTS:
;  3003     3070  1 !
;  3004     3071  1 !       V_QUEUE_INIT set to 1
;  3005     3072  1 !       KCB_QUEUE initialized as an empty queue
;  3006     3073  1 !       QPS_QUEUE initialized as an empty queue
;  3007     3074  1 !
;  3008     3075  1 ! COMPLETION STATUS:
;  3009     3076  1 !
;  3010     3077  1 !       SS$_NORMAL      Normal successful completion
;  3011     3078  1 !       Errors from LIB$GET_VM or $DCLEXH
;  3012     3079  1 !
;  3013     3080  1 ! SIDE EFFECTS:
;  3014     3081  1 !
;  3015     3082  1 !       Declares SMG$$INPUT_EXIT_HANDLER as an exit handler
;  3016     3083  1 !
;  3017     3084  1 !--
;  3018     3085  1
;  3019     3086  2     BEGIN
;  3020     3087  2
;  3021     3088  2     LOCAL
;  3022     3089  2         AST_STATUS,
;  3023     3090  2         VM_STATUS,
;  3024     3091  2         V_ENABLE_EXITH, ! Set to 1 if exit handler should be enabled
;  3025     3092  2         EXIT_BLOCK: REF VECTOR [4, LONG],       ! Exit handler block
;  3026     3093  2         EB_PTR;         ! Pointer to allocated memory
;  3027     3094  2
;  3028     3095  2     BUILTIN
;  3029     3096  2         TESTBITCS;
;  3030     3097  2
;  3031     3098  2     !+
;  3032     3099  2     ! Assume that we don't have to enable the exit handler.
;  3033     3100  2     !-
;  3034     3101  2
```

```
;  3035     3102  2      V_ENABLE_EXITH = 0;
;  3036     3103  2
;  3037     3104  2      !+
;  3038     3105  2      ! Disable ASTs
;  3039     3106  2      !-
;  3040     3107  2
;  3041     3108  2      AST_STATUS = $SETAST (ENBFLG=0);
;  3042     3109  2
;  3043     3110  2      !+
;  3044     3111  2      ! If V_QUEUE_INIT still clear, initialize the queue.
;  3045     3112  2      !-
;  3046     3113  2
;  3047     3114  2      IF TESTBITCS (V_QUEUE_INIT)
;  3048     3115  2      THEN
;  3049     3116  3          BEGIN
;  3050     3117  3
;  3051     3118  3          !+
;  3052     3119  3          ! Initialize KQB_QUEUE to be an empty queue.
;  3053     3120  3          !-
;  3054     3121  3
;  3055     3122  3          KQB_QUEUE [0] = KQB_QUEUE;      ! Set forward link
;  3056     3123  3          KQB_QUEUE [1] = KQB_QUEUE;      ! Set backwards link
;  3057     3124  3
;  3058     3125  3          !+
;  3059     3126  3          ! Initialize QPS_QUEUE to be an empty queue.
;  3060     3127  3          !-
;  3061     3128  3
;  3062     3129  3          QPS_QUEUE [0] = QPS_QUEUE;      ! Set forward link
;  3063     3130  3          QPS_QUEUE [1] = QPS_QUEUE;      ! Set backwards link
;  3064     3131  3
;  3065     3132  3          !+
;  3066     3133  3          ! Remember that we want to declare the exit handler.
;  3067     3134  3          !-
;  3068     3135  3
;  3069     3136  3          V_ENABLE_EXITH = 1;
;  3070     3137  3
;  3071     3138  2          END;
;  3072     3139  2
;  3073     3140  2      !+
;  3074     3141  2      ! Reenable ASTs if previously enabled.
;  3075     3142  2      !-
;  3076     3143  2
;  3077     3144  2      IF .AST_STATUS EQLU SS$_WASSET
;  3078     3145  2      THEN
;  3079     3146  2          $SETAST (ENBFLG=1);
;  3080     3147  2
;  3081     3148  2      !+
;  3082     3149  2      ! If we need to enable the exit handler, do it now.  It is not
;  3083     3150  2      ! harmful if other AST-level operations go on while we are doing this.
;  3084     3151  2      !-
;  3085     3152  2
;  3086     3153  2      IF .V_ENABLE_EXITH
;  3087     3154  2      THEN
;  3088     3155  3          BEGIN
;  3089     3156  3
;  3090     3157  3          VM_STATUS = LIB$GET_VM (%REF(20), EB_PTR);
;  3091     3158  3          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:
                                                             .QWORD  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
```

```
;  3105     3171  1 %SBTTL 'SMG$$CLEANUP'
;  3106     3172  1 ROUTINE SMG$$CLEANUP (
;  3107     3173  1     KCB: REF KCB_R_KCB_STRUCT,          ! Keyboard control block
;  3108     3174  1     RET_STATUS                          ! Return status
;  3109     3175  1     ): SMG$$CLEANUP$LNK =
;  3110     3176  1
;  3111     3177  1 !++
;  3112     3178  1 ! FUNCTIONAL DESCRIPTION:
;  3113     3179  1 !
;  3114     3180  1 !       This procedure performs all the necessary actions to delete
;  3115     3181  1 !       a virtual keyboard.
;  3116     3182  1 !
;  3117     3183  1 ! CALLING SEQUENCE:
;  3118     3184  1 !
;  3119     3185  1 !       ret_status.wlc.v = SMG$$CLEANUP (KCB.mr.r, RET_STATUS.rlc.v)
;  3120     3186  1 !
;  3121     3187  1 ! FORMAL PARAMETERS:
;  3122     3188  1 !
;  3123     3189  1 !       KCB (R8)        - The address of the Keyboard Control Block being
;  3124     3190  1 !                         deleted.
;  3125     3191  1 !
;  3126     3192  1 !       RET_STATUS (R0) - A status which is to be returned by
;  3127     3193  1 !                         SMG$$CLEANUP.
;  3128     3194  1 !
;  3129     3195  1 ! IMPLICIT INPUTS:
;  3130     3196  1 !
;  3131     3197  1 !       NONE                                                          ;
;  3132     3198  1 !
;  3133     3199  1 ! IMPLICIT OUTPUTS:
;  3134     3200  1 !
;  3135     3201  1 !       NONE
;  3136     3202  1 !
;  3137     3203  1 ! COMPLETION STATUS:
;  3138     3204  1 !
;  3139     3205  1 !       The value of the RET_STATUS parameter is returned in R0.
;  3140     3206  1 !
;  3141     3207  1 ! SIDE EFFECTS:
;  3142     3208  1 !
;  3143     3209  1 !       Deletes the virtual keyboard.
;  3144     3210  1 !
;  3145     3211  1 !--
;  3146     3212  1
;  3147     3213  2     BEGIN
;  3148     3214  2
;  3149     3215  2     LOCAL
;  3150     3216  2         KQB: REF KQB_R_KQB_STRUCT;      ! Keyboard Queue Block
;  3151     3217  2
;  3152     3218  2     BUILTIN
;  3153     3219  2         CALLG;
;  3154     3220  2
;  3155     3221  2     !+
;  3156     3222  2     ! Remove this KCB from the list of KCBs.
;  3157     3223  2     !-
;  3158     3224  2
;  3159     3225  2     KQB = .KCB [KCB_A_KQB];     ! Get Keyboard Queue Block pointer
;  3160     3226  2     IF .KQB NEQA 0
;  3161     3227  2     THEN
```

```
; 3162  3228  2          KQB [KQB_A_KCB] = 0;
; 3163  3229  2
; 3164  3230  2      !+
; 3165  3231  2      ! Split for RMS vs. $QIO
; 3166  3232  2      !-
; 3167  3233  2
; 3168  3234  2      IF .KCB [KCB_V_RMS]
; 3169  3235  2      THEN
; 3170  3236  3          BEGIN
; 3171  3237  3          LOCAL
; 3172  3238  3              FAB: $FAB_DECL;
; 3173  3239  3
; 3174  3240  3          !+
; 3175  3241  3          ! Set up new FAB from saved IFI.
; 3176  3242  3          !-
; 3177  3243  3
; 3178  3244  3          $FAB_INIT (FAB=FAB);
; 3179  3245  3          FAB [FAB$W_IFI] = KCB [KCB_W_IFI];
; 3180  3246  3
; 3181  3247  3          !+
; 3182  3248  3          ! Close file if opened.
; 3183  3249  3          !-
; 3184  3250  3
; 3185  3251  3          IF .FAB [FAB$W_IFI] NEQ 0
; 3186  3252  3          THEN
; 3187  3253  3              $CLOSE (FAB=FAB);
; 3188  3254  3
; 3189  3255  3          !+
; 3190  3256  3          ! Deallocate record buffer if allocated.
; 3191  3257  3          !-
; 3192  3258  3
; 3193  3259  3          IF .KCB [RAB$L_UBF] NEQA 0
; 3194  3260  3          THEN
; 3195  3261  3              LIB$FREE_VM (%REF(.KCB [RAB$W_USZ]), KCB [RAB$L_UBF]);
; 3196  3262  3
; 3197  3263  3          END
; 3198  3264  2      ELSE
; 3199  3265  3          BEGIN
; 3200  3266  3
; 3201  3267  3          BIND
; 3202  3268  3              DEVDEPEND2 = KCB [KCB_L_DEVDEPEND2]: BLOCK [4, BYTE];
; 3203  3269  3
; 3204  3270  3          !+
; 3205  3271  3          ! If we have changed the terminal characteristics and keypad,
; 3206  3272  3          ! restore them.
; 3207  3273  3          !-
; 3208  3274  3
; 3209  3275  3          IF .KCB [KCB_V_CHARS_CHANGED]
; 3210  3276  3          THEN
; 3211  3277  4              BEGIN
; 3212  3278  4
; 3213  3279  4              !+
; 3214  3280  4              ! Restore old DEVDEPEND2 information.
; 3215  3281  4              !-
; 3216  3282  4
; 3217  3283  4              KCB [KCB_L_DEVDEPEND2] = .KCB [KCB_L_OLD_DEVDEPEND2];
; 3218  3284  4
```

```
;  3219     3285  4              !+
;  3220     3286  4              ! Reset keypad mode.
;  3221     3287  4              !-
;  3222     3288  4
;  3223     3289  4              SMG$SET_KEYPAD_MODE (%REF(KCB [KCB_R_KCB]),
;  3224     3290  4                  %REF(.DEVDEPEND2 [TT2$V_APP_KEYPAD]));
;  3225     3291  3              END;
;  3226     3292  3
;  3227     3293  3          !+
;  3228     3294  3          ! Deassign channel if assigned.
;  3229     3295  3          !-
;  3230     3296  3
;  3231     3297  3          IF .KCB [KCB_W_CHANNEL] NEQ 0
;  3232     3298  3          THEN
;  3233     3299  3              $DASSGN (CHAN = .KCB [KCB_W_CHANNEL]);
;  3234     3300  2          END;
;  3235     3301  2
;  3236     3302  2      !+
;  3237     3303  2      ! Free event flag if allocated.
;  3238     3304  2      !-
;  3239     3305  2
;  3240     3306  2      IF .KCB [KCB_L_EFN] NEQ 0
;  3241     3307  2      THEN
;  3242     3308  2          LIB$FREE_EF (KCB [KCB_L_EFN]);
;  3243     3309  2
;  3244     3310  2      !+
;  3245     3311  2      ! Deallocate KCB.
;  3246     3312  2      !-
;  3247     3313  2
;  3248     3314  2      LIB$FREE_VM (%REF(KCB_S_KCB_STRUCT), %REF (.KCB - KCB_K_ORIGIN_OFFSET));
;  3249     3315  2
;  3250     3316  2      RETURN .RET_STATUS;
;  3251     3317  2
;  3252     3318  1      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$FREE_VM, R7
                                    5E        A8  AE  9E 00009          MOVAB   -88(SP), SP
                                    56            50  D0 0000D          MOVL    R0, R6
                                    50        D8  A8  D0 00010          MOVL    -40(KCB), KQB
                                                  03  13 00014          BEQL    1$
                                              08  A0  D4 00016          CLRL    8(KQB)
                                    40        FC  A8  E9 00019 1$:      BLBC    -4(KCB), 3$
    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
                                            08 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
```

```
;3254  3319  1 %SBTTL 'SMG$$INPUT_EXIT_HANDLER'
;3255  3320  1 ROUTINE SMG$$INPUT_EXIT_HANDLER
;3256  3321  1     : NOVALUE =
;3257  3322  1
;3258  3323  1 !++
;3259  3324  1 ! FUNCTIONAL DESCRIPTION:
;3260  3325  1 !
;3261  3326  1 !       This procedure is the exit handler for the Screen Management
;3262  3327  1 !       input procedures.  It scans the list of Keyboard Queue Blocks
;3263  3328  1 !       looking for those which refer to existing KCBs.  Those KCBs
;3264  3329  1 !       are then deleted.
;3265  3330  1 !
;3266  3331  1 ! CALLING SEQUENCE:
;3267  3332  1 !
;3268  3333  1 !       Called by VMS upon image exit
;3269  3334  1 !
;3270  3335  1 ! FORMAL PARAMETERS:
;3271  3336  1 !
;3272  3337  1 !       None used
;3273  3338  1 !
;3274  3339  1 ! IMPLICIT INPUTS:
;3275  3340  1 !
;3276  3341  1 !       KQB_QUEUE
;3277  3342  1 !
;3278  3343  1 ! IMPLICIT OUTPUTS:
;3279  3344  1 !
;3280  3345  1 !       V_KQB_QUEUE_INIT set to zero
;3281  3346  1 !
;3282  3347  1 ! COMPLETION STATUS:
;3283  3348  1 !
;3284  3349  1 !       NONE
;3285  3350  1 !
;3286  3351  1 ! SIDE EFFECTS:
;3287  3352  1 !
;3288  3353  1 !       Deletes all existant virtual keyboards.
;3289  3354  1 !
;3290  3355  1 !--
;3291  3356  1
;3292  3357  2     BEGIN
;3293  3358  2
;3294  3359  2     LOCAL
;3295  3360  2         KQB: REF KQB_R_KQB_STRUCT,      ! Keyboard Queue Block
;3296  3361  2         KCB: REF KCB_R_KCB_STRUCT,      ! Keyboard Control Block
;3297  3362  2         QUEUE_HEAD_ADR: REF VECTOR [, LONG];    ! Address of queue head
;3298  3363  2
;3299  3364  2     !+
;3300  3365  2     ! Get the first KQB from the head of the queue.  Note that the
;3301  3366  2     ! queue must have already been initialized.
;3302  3367  2     !-
;3303  3368  2
;3304  3369  2     QUEUE_HEAD_ADR = KQB_QUEUE [0];
;3305  3370  2     KQB = .QUEUE_HEAD_ADR [0];
;3306  3371  2
;3307  3372  2     !+
;3308  3373  2     ! While we haven't returned to the head of the queue, if this KQB
;3309  3374  2     ! has an active KCB, delete it.
;3310  3375  2     !-
```

```
;  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
```

SMGKEYUTI
LIS

SMGKEYPAD
LIS

SMGINPUT
LIS

SMGLIB
LIS