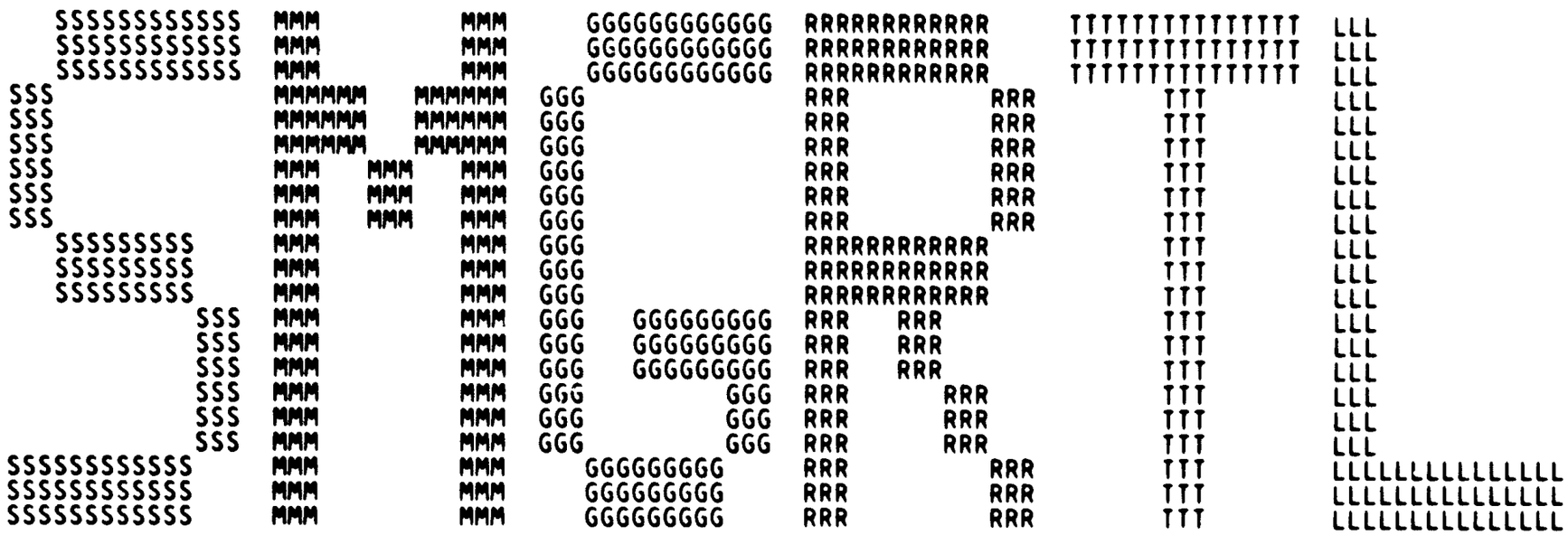

_$2

Val

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

FILEID**SMGKEYPAD

```
  SSSSSSSS  MM      MM    GGGGGGGG  KK      KK  EEEEEEEEEE  YY      YY  PPPPPPPP      AAAAAA    DDDDDDDD
  SSSSSSSS  MM      MM    GGGGGGGG  KK      KK  EEEEEEEEEE  YY      YY  PPPPPPPP      AAAAAA    DDDDDDDD
SS          MMMM  MMMM  GG          KK      KK  EE          YY      YY  PP      PP  AA      AA  DD      DD
SS          MMMM  MMMM  GG          KK      KK  EE          YY      YY  PP      PP  AA      AA  DD      DD
SS          MM  MM  MM  GG          KK    KK    EE            YY  YY    PP      PP  AA      AA  DD      DD
SS          MM  MM  MM  GG          KK    KK    EE            YY  YY    PP      PP  AA      AA  DD      DD
  SSSSSS    MM      MM  GG          KKKKKK      EEEEEEEE        YY      PPPPPPPP    AA      AA  DD      DD
  SSSSSS    MM      MM  GG          KKKKKK      EEEEEEEE        YY      PPPPPPPP    AA      AA  DD      DD
        SS  MM      MM  GG  GGGGGG  KK    KK    EE              YY      PP          AAAAAAAAAA  DD      DD
        SS  MM      MM  GG  GGGGGG  KK    KK    EE              YY      PP          AAAAAAAAAA  DD      DD
        SS  MM      MM  GG      GG  KK      KK  EE              YY      PP          AA      AA  DD      DD      ....
        SS  MM      MM  GG      GG  KK      KK  EE              YY      PP          AA      AA  DD      DD      ....
SSSSSSSS    MM      MM    GGGGGG    KK      KK  EEEEEEEEEE      YY      PP          AA      AA  DDDDDDDD      ....
SSSSSSSS    MM      MM    GGGGGG    KK      KK  EEEEEEEEEE      YY      PP          AA      AA  DDDDDDDD      ....

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$KEYPAD ( %TITLE 'Screen Management Facility Keypad Procedures'
;     2          0002  0                     IDENT = '1-006'          ! File: SMGKEYPAD.B32 Edit: SBL1006
;     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 procedures to manage keypad key definitions.
;    36          0036  1 !
;    37          0037  1 ! ENVIRONMENT:  User mode - AST reentrant
;    38          0038  1 !
;    39          0039  1 ! AUTHOR: Steven B. Lionel, CREATION DATE: 10-Feb-1983
;    40          0040  1 !
;    41          0041  1 ! MODIFIED BY:
;    42          0042  1 !
;    43          0043  1 ! 1-001 - Original.  SBL 10-Feb-1983
;    44          0044  1 ! 1-002 - Use 'DEFAULT' as the default state name.  SBL 18-May-1983
;    45          0045  1 ! 1-003 - Keep the default state name in the KTH, so it can be sticky
;    46          0046  1 !         across operations.  SBL 23-Jun-1983
;    47          0047  1 ! 1-004 - Add SMG$LIST_KEY_DEFS and SMG$SET_DEFAULT_STATE.  When creating
;    48          0048  1 !         a "total key", swap the bytes of the key code so that control
;    49          0049  1 !         keys sort in the proper order.  SBL 10-Aug-1983
;    50          0050  1 ! 1-005 - Set KTH_A_DEF_KEYCODE in SMG$SET_DEFAULT_STATE.  SBL 14-Sep-1983
;    51          0051  1 ! 1-006 - Insert required NUL into TKEY in SMG$SET_DEFAULT_STATE.  SBL 15-Sep-1983
;    52          0052  1 !--
;    53          0053  1
```

```
;    55          0054  1 %SBTTL 'Declarations'
;    56          0055  1 !
;    57          0056  1 ! PROLOGUE FILE:
;    58          0057  1 !
;    59          0058  1
;    60          0059  1 REQUIRE 'RTLIN:SMGPROLOG';                          . Screen management definitions
;    61          0137  1
;    62          0138  1 !
;    63          0139  1 ! LINKAGE
;    64          0140  1 !
;    65          0141  1
;    66          0142  1 LINKAGE
;    67          0143  1     CONSTRUCT_TKEY$LNK =
;    68          0144  1         CALL (REGISTER=6, REGISTER=7; REGISTER=7);
;    69          0145  1
;    70          0146  1 !
;    71          0147  1 ! TABLE OF CONTENTS:
;    72          0148  1 !
;    73          0149  1
;    74          0150  1 FORWARD ROUTINE
;    75          0151  1     SMG$CREATE_KEY_TABLE,                          ! Create a key table
;    76          0152  1     SMG$ADD_KEY_DEF,                               ! Add a key definition
;    77          0153  1     SMG$GET_KEY_DEF,                               ! Get a key definition
;    78          0154  1     SMG$DELETE_KEY_DEF,                            ! Delete a key definition
;    79          0155  1     SMG$SET_DEFAULT_STATE,                         ! Set or get a default state
;    80          0156  1     SMG$$LOOKUP_KEY: SMG$$LOOKUP_KEY$LNK,          ! Lookup a key
;    81          0157  1     CONSTRUCT_TKEY: CONSTRUCT_TKEY$LNK,            ! Construct the total key
;    82          0158  1     COMPARE_ROUTINE,                               ! Compare two entries
;    83          0159  1     ALLOC_ROUTINE,                                 ! Allocate an entry
;    84          0160  1     SMG$DEFINE_KEY,                                ! Process DEFINE/KEY
;    85          0161  1     DEFINE_KEY_HANDLER,                            ! Handler for SMG$DEFINE_KEY
;    86          0162  1     SMG$LOAD_KEY_DEFS,                             ! Load a file of key defs
;    87          0163  1     SMG$LIST_KEY_DEFS,                             ! List key definitions
;    88          0164  1     TRAVERSE_ROUTINE;                              ! Traverse key table
;    89          0165  1
;    90          0166  1 !
;    91          0167  1 ! MACROS:
;    92          0168  1 !
;    93          0169  1 !       NONE
;    94          0170  1 !
;    95          0171  1 ! EQUATED SYMBOLS:
;    96          0172  1 !
;    97          0173  1
;    98          0174  1 !+
;    99          0175  1 ! Define a mask of those control keys used by advanced line editing.
;   100          0176  1 !-
;   101          0177  1
;   102          0178  1 LITERAL
;   103          0179  1     K_CTRLA = 1,
;   104          0180  1     K_CTRLB = 2,
;   105          0181  1     K_CTRLD = 4,
;   106          0182  1     K_CTRLE = 5,
;   107          0183  1     K_CTRLF = 6,
;   108          0184  1     K_BS = 8,
;   109          0185  1     K_HT = 9,
;   110          0186  1     K_LF = 10,
;   111          0187  1     M_EDIT_KEYS = (1^K_CTRLA) + (1^K_CTRLB) + (1^K_CTRLD) + (1^K_CTRLE) +
```

```
;  112     0188  1          (1^K_CTRLF) + (1^K_BS) + (1^K_LF);
;  113     0189  1
;  114     0190  1 !
;  115     0191  1 ! FIELDS:
;  116     0192  1 !
;  117     0193  1 !       NONE
;  118     0194  1 !
;  119     0195  1 ! OWN STORAGE:
;  120     0196  1 !
;  121     0197  1 !       NONE
;  122     0198  1 !
;  123     0199  1 ! EXTERNAL DECLARATIONS:
;  124     0200  1 !
;  125     0201  1
;  126     0202  1 EXTERNAL ROUTINE
;  127     0203  1     CLI$DCL_PARSE,                  ! Parse command line
;  128     0204  1     CLI$GET_VALUE,                  ! Get value of item from command line
;  129     0205  1     CLI$PRESENT,                    ! Indicate whether item is present
;  130     0206  1     LIB$GET_VM,                     ! Allocate virtual memory
;  131     0207  1     LIB$ANALYZE_SDESC_R2: LIB$ANALYZE_SDESC_R2$LINKAGE, ! Analyze string desc
;  132     0208  1     LIB$SCOPY_R_DX6: LIB$SCOPY_R_DX6$LINKAGE,   ! Copy string
;  133     0209  1     LIB$SCOPY_DXDX6: LIB$SCOPY_DXDX6$LINKAGE,   ! Copy string
;  134     0210  1     LIB$SFREE1_DD,                  ! Free string
;  135     0211  1     LIB$SFREE1_DD6: LIB$SFREE1_DD6$LINKAGE,     ! Free string
;  136     0212  1     LIB$SFREEN_DD,                  ! Free strings
;  137     0213  1     LIB$SIGNAL,                     ! Signal continuable error
;  138     0214  1     LIB$INSERT_TREE,                ! Insert item in balanced tree
;  139     0215  1     LIB$LOOKUP_TREE,                ! Find item in tree
;  140     0216  1     LIB$TRAVERSE_TREE,              ! Traverse balanced tree
;  141     0217  1     SMG$CREATE_VIRTUAL_KEYBOARD,    ! Create a virtual keyboard
;  142     0218  1     SMG$DELETE_VIRTUAL_KEYBOARD,    ! Delete a virtual keyboard
;  143     0219  1     SMG$READ_STRING,                ! Read a string
;  144     0220  1     SMG$$KEYCODE_TO_NAME,           ! Convert key code to name
;  145     0221  1     SMG$$NAME_TO_KEYCODE;           ! Convert key name to key code
;  146     0222  1
;  147     0223  1 EXTERNAL LITERAL
;  148     0224  1     CLI$_FACILITY: UNSIGNED(6),     ! CLI facility code
;  149     0225  1     CLI$_NOCOMD,                    ! No command on line
;  150     0226  1     SMG$_FILTOOLON,                 ! Filespec too long
;  151     0227  1     SMG$_INVDEFATT,                 ! Invalid key-definition attributes
;  152     0228  1     SMG$_INVKEYNAM,                 ! Invalid key name
;  153     0229  1     SMG$_INVSTANAM,                 ! Invalid state name
;  154     0230  1     SMG$_KEYDEFPRO,                 ! Key definition is protected
;  155     0231  1     SMG$_KEYNOTDEF,                 ! Key is not defined
;  156     0232  1     SMG$_NOMOREKEYS,                ! No more keys
;  157     0233  1     SMG$_PREDEFREP;                 ! Previous key-definition replaced
;  158     0234  1
;  159     0235  1 EXTERNAL
;  160     0236  1     LIB$AB_UPCASE,                  ! Upcase translate table
;  161     0237  1     SMG$$AB_DEFKEY_CLD;             ! Command line definition for DEFINE/KEY
```

```
;  163     0238  1 %SBTTL 'SMG$CREATE_KEY_TABLE'
;  164     0239  1 GLOBAL ROUTINE SMG$CREATE_KEY_TABLE (
;  165     0240  1     KEY_TABLE_ID: REF VECTOR [, LONG]
;  166     0241  1     ) =
;  167     0242  1
;  168     0243  1 !++
;  169     0244  1 ! FUNCTIONAL DESCRIPTION:
;  170     0245  1 !
;  171     0246  1 !       This procedure creates a key-definition table, which can then have
;  172     0247  1 !       key definitions added to it by calls to SMG$ADD_KEY_DEF,
;  173     0248  1 !       SMG$DEFINE_KEY and SMG$LOAD_KEY_DEFS.
;  174     0249  1 !
;  175     0250  1 ! CALLING SEQUENCE:
;  176     0251  1 !
;  177     0252  1 !       RET_STATUS.wlc.v = SMG$CREATE_KEY_TABLE (
;  178     0253  1 !                                 KEY_TABLE_ID.wl.r)
;  179     0254  1 !
;  180     0255  1 ! FORMAL PARAMETERS:
;  181     0256  1 !
;  182     0257  1 !       KEY_TABLE_ID    A longword into which is written the identification
;  183     0258  1 !                       of the newly-created key-definition table.  This
;  184     0259  1 !                       identification can then be passed to other Screen
;  185     0260  1 !                       Management procedures to uniquely identify this set
;  186     0261  1 !                       of key definitions.
;  187     0262  1 !
;  188     0263  1 ! IMPLICIT INPUTS:
;  189     0264  1 !
;  190     0265  1 !       NONE
;  191     0266  1 !
;  192     0267  1 ! IMPLICIT OUTPUTS:
;  193     0268  1 !
;  194     0269  1 !       NONE
;  195     0270  1 !
;  196     0271  1 ! COMPLETION STATUS:
;  197     0272  1 !
;  198     0273  1 !       SS$_NORMAL      Normal successful completion
;  199     0274  1 !       LIB$_INSVIRMEM  Insufficient virtual memory
;  200     0275  1 !
;  201     0276  1 ! SIDE EFFECTS:
;  202     0277  1 !
;  203     0278  1 !       NONE
;  204     0279  1 !
;  205     0280  1 !--
;  206     0281  1
;  207     0282  2     BEGIN
;  208     0283  2
;  209     0284  2     LOCAL
;  210     0285  2         KTH: REF KTH_R_KTH_STRUCT,              ! Key table header
;  211     0286  2         VM_STATUS;
;  212     0287  2
;  213     0288  2     !+
;  214     0289  2     ! Allocate and initialize the key table header.  For a default terminator
;  215     0290  2     ! mask, use all control characters except those used by advanced line
;  216     0291  2     ! editing and <HT>.
;  217     0292  2     !-
;  218     0293  2
;  219     0294  2     VM_STATUS = LIB$GET_VM (%REF(KTH_S_KTH_STRUCT), KEY_TABLE_ID [0]);
```

```
;  220          0295  2      IF NOT .VM_STATUS
;  221          0296  2      THEN
;  222          0297  2          RETURN .VM_STATUS;
;  223          0298  2      KTH = .KEY_TAB[E_ID [0];
;  224          0299  2      KTH [KTH_A_TREEHEAD] = 0;
;  225          0300  2      KTH [KTH_L_FLAGS] = 0;
;  226          0301  2      KTH [KTH_L_MODIFIERS] = 0;
;  227          0302  2      KTH [KTH_L_TERM_MASK] = %X'FFFFFFFF' - M_EDIT_KEYS - (1^K_HT);
;  228          0303  2
;  229          0304  2      !+
;  230          0305  2      ! Establish 'DEFAULT' as the default state.  The string contains
;  231          0306  2      ! the default state name, a byte of zero, and two bytes for the
;  232          0307  2      ! keycode.
;  233          0308  2      !-
;  234          0309  2
;  235          0310  2      KTH [KTH_W_DEF_STATE_LEN] = %CHARCOUNT('DEFAULT') + 3;
;  236          0311  2      KTH [KTH_B_DEF_STATE_CLASS] = DSC$K_CLASS_S;
;  237          0312  2      KTH [KTH_B_DEF_STATE_DTYPE] = DSC$K_DTYPE_T;
;  238          0313  2      KTH [KTH_A_DEF_STATE_POINTER] = KTH [KTH_T_DEF_STATE_STRING];
;  239          0314  2      KTH [KTH_A_DEF_KEYCODE] = CH$MOVE (
;  240          0315  2          %CHARCOUNT('DEFAULT') + 1,
;  241          0316  2          UPLIT BYTE ('DEFAULT',0),
;  242          0317  2          KTH [KTH_T_DEF_STATE_STRING]);
;  243          0318  2
;  244          0319  2      KTH [KTH_L_CHECK] = .KTH;
;  245          0320  2
;  246          0321  2      RETURN SS$_NORMAL;
;  247          0322  2
;  248          0323  1      END;                                    ! End of routine SMG$CREATE_KEY_TABLE

                                                                    .TITLE  SMG$KEYPAD Screen Management Facility Keypad Pr
                                                                                ocedures
;                                                                   .IDENT  \1-006\

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

                                54  4C  55  41  46  45  44  00000 P.AAA:  .ASCII  \DEFAULT\                                 ;
                                                 00  00007              .BYTE   0                                          ;

                                                                    .EXTRN  CLI$DCL_PARSE, CLI$GET_VALUE
                                                                    .EXTRN  CLI$PRESENT, LIB$GET_VM
                                                                    .EXTRN  LIB$ANALYZE_SDESC_R2
                                                                    .EXTRN  LIB$SCOPY_R_DX6
                                                                    .EXTRN  LIB$SCOPY_DXDX6
                                                                    .EXTRN  LIB$SFREE1_DD, LIB$SFREE1_DD6
                                                                    .EXTRN  LIB$SFREEN_DD, LIB$SIGNAL
                                                                    .EXTRN  LIB$INSERT_TREE
                                                                    .EXTRN  LIB$LOOKUP_TREE
                                                                    .EXTRN  LIB$TRAVERSE_TREE
                                                                    .EXTRN  SMG$CREATE_VIRTUAL_KEYBOARD
                                                                    .EXTRN  SMG$DELETE_VIRTUAL_KEYBOARD
                                                                    .EXTRN  SMG$READ_STRING
                                                                    .EXTRN  SMG$$KEYCODE_TO_NAME
                                                                    .EXTRN  SMG$$NAME_TO_KEYCODE
                                                                    .EXTRN  CLI$_FACILITY, CLI$_NOCOMD
                                                                    .EXTRN  SMG$_FILTOOLON, SMG$_INVDEFATT
```

```
                                                                .EXTRN  SMG$_INVKEYNAM, SMG$_INVSTANAM
                                                                .EXTRN  SMG$_KEYDEFPRO, SMG$_KEYNOTDEF
                                                                .EXTRN  SMG$_NOMOREKEYS
                                                                .EXTRN  SMG$_PREDEFREP, LIB$AB_UPCASE
                                                                .EXTRN  SMG$$AB_DEFKEY_CLD

                                                 007C 00000     .ENTRY  SMG$CREATE_KEY_TABLE, Save R2,R3,R4,R5,R6    ; 0239
                                5E                 04   C2 00002  SUBL2   #4, SP                                   ;
                                          04       AC   DD 00005  PUSHL   KEY_TABLE_ID                             ; 0294
                                   04     AE       42   8F 00008  MOVZBL  #66, 4(SP)                               ;
                                          04       AE   9F 0000D  PUSHAB  4(SP)                                    ;
                          00000000G       00         02   FB 00010  CALLS   #2, LIB$GET_VM                         ;
                                          2D         50   E9 00017  BLBC    VM_STATUS, T$                          ; 0295
                                          56   04    BC   D0 0001A  MOVL    @KEY_TABLE_ID, KTH                     ; 0298
                                               04    66   7C 0001E  CLRQ    (KTH)                                  ; 0299
                                               08    A6   D4 00020  CLRL    8(KTH)                                 ; 0301
                                   0C     A6  F889    8F   32 00023  CVTWL   #-1911, 12(KTH)                       ; 0302
                                   14     A6  010E000A 8F   D0 00029  MOVL    #17694730, 20(KTH)                     ; 0310
                                   18     A6  20     A6   9E 00031  MOVAB   32(KTH), 24(KTH)                       ; 0313
                        20   A6    BE     AF         08   28 00036  MOVC3   #8, P.AAA, 32(KTH)                     ; 0317
                                   1C     A6         53   D0 0003C  MOVL    R3, 28(KTH)                            ;
                                   10     A6         56   D0 00040  MOVL    KTH, 16(KTH)                           ; 0319
                                          50         01   D0 00044  MOVL    #1, R0                                 ; 0321
                                                     04   00047 1$:  RET                                           ; 0323

; Routine Size:  72 bytes,    Routine Base:  _SMG$CODE + 0008
```

```
;  250          0324  1 %SBTTL 'SMG$ADD_KEY_DEF'
;  251          0325  1 GLOBAL ROUTINE SMG$ADD_KEY_DEF (
;  252          0326  1     KEY_TABLE_ID: REF VECTOR [, LONG],
;  253          0327  1     KEY_NAME: REF BLOCK [, BYTE],
;  254          0328  1     IF_STATE: REF BLOCK [, BYTE],
;  255          0329  1     ATTRIBUTES: REF VECTOR [, LONG],
;  256          0330  1     EQUIV_STRING: REF BLOCK [, BYTE],
;  257          0331  1     STATE_STRING: REF BLOCK [, BYTE]
;  258          0332  1     ) =
;  259          0333  1
;  260          0334  1 !++
;  261          0335  1 ! FUNCTIONAL DESCRIPTION:
;  262          0336  1 !
;  263          0337  1 !       This procedure adds a key definition to a table of key definitions.
;  264          0338  1 !       The table must have been created by a call to SMG$CREATE_KEY_TABLE.
;  265          0339  1 !
;  266          0340  1 ! CALLING SEQUENCE:
;  267          0341  1 !
;  268          0342  1 !       RET_STATUS.wlc.v = SMG$ADD_KEY_DEF (
;  269          0343  1 !                                  KEY_TABLE_ID.rl.r,
;  270          0344  1 !                                  KEY_NAME.rt.dx
;  271          0345  1 !                                  [, [IF_STATE.rt.dx]
;  272          0346  1 !                                  [, [ATTRIBUTES.rl.r]
;  273          0347  1 !                                  [, [EQUIV_STRING.rt.dx]
;  274          0348  1 !                                  [, [STATE_STRING.rt.dx] ]]]])
;  275          0349  1 !
;  276          0350  1 ! FORMAL PARAMETERS:
;  277          0351  1 !
;  278          0352  1 !       KEY_TABLE_ID    A longword containing the identification of the
;  279          0353  1 !                       key-definition table to which this definition is
;  280          0354  1 !                       to be added.
;  281          0355  1 !
;  282          0356  1 !       KEY_NAME        A string containing the name of the key to be defined.
;  283          0357  1 !                       The key name is stripped of trailing blanks and is
;  284          0358  1 !                       converted to upper-case before use.  The key name must
;  285          0359  1 !                       be one of the key names listed in table _\_\ TBS _\_\.
;  286          0360  1 !
;  287          0361  1 !       IF_STATE        A string containing a state name which further qualifies
;  288          0362  1 !                       KEY_NAME.  If specified, this definition of KEY_NAME is
;  289          0363  1 !                       used only if the current state is IF_STATE.  See the
;  290          0364  1 !                       description of key states in the Screen Management
;  291          0365  1 !                       Functional Specification.  If omitted, 'DEFAULT'
;  292          0366  1 !                       is used.
;  293          0367  1 !
;  294          0368  1 !       ATTRIBUTES      A longword bit mask specifying additional attributes of
;  295          0369  1 !                       this key definition.  If omitted, a mask of zero is
;  296          0370  1 !                       used.  The attributes defined are:
;  297          0371  1 !
;  298          0372  1 !                         SMG$V_KEY_NOECHO - Bit 0
;  299          0373  1 !                           If set, specifies that EQUIV_STRING is not to
;  300          0374  1 !                           be echoed when this key is pressed.  If clear,
;  301          0375  1 !                           EQUIV_STRING is echoed.  If SMG$V_KEY_TERMINATE
;  302          0376  1 !                           is not set, SMG$V_KEY_NOECHO is ignored.
;  303          0377  1 !
;  304          0378  1 !                         SMG$V_KEY_TERMINATE - Bit 1
;  305          0379  1 !                           If set, specifies that when this key is pressed
;  306          0380  1 !                           (qualified by IF_STATE), that the input line is
```

```
;  307     0381  1 !                                   complete and that more characters should not be
;  308     0382  1 !                                   accepted.  If clear, more more characters may be
;  309     0383  1 !                                   accepted.
;  310     0384  1 !
;  311     0385  1 !                                 SMG$V_KEY_LOCKSTATE - Bit 2
;  312     0386  1 !                                   If set, and if STATE_STRING is specified, the
;  313     0387  1 !                                   state name specified by STATE_STRING remains the
;  314     0388  1 !                                   current state until explicitly changed by a
;  315     0389  1 !                                   subsequent keystroke whose definition includes
;  316     0390  1 !                                   a STATE_STRING.  If clear, the state name specified
;  317     0391  1 !                                   by STATE_STRING remains in effect only for the next
;  318     0392  1 !                                   defineable key.
;  319     0393  1 !
;  320     0394  1 !                                 SMG$V_KEY_PROTECTED - Bit 3
;  321     0395  1 !                                   If set, indicates that this key definition cannot
;  322     0396  1 !                                   be modified or deleted.  If clear, indicates that
;  323     0397  1 !                                   this key definition can be modified or deleted.
;  324     0398  1 !
;  325     0399  1 !                                 SMG$V_KEY_SETSTATE - Bit 4
;  326     0400  1 !                                   This bit is ignored by SMG$ADD_KEY_DEF.  It is
;  327     0401  1 !                                   returned by SMG$GET_KEY_DEF and SMG$LIST_KEY_DEFS
;  328     0402  1 !                                   to indicate that the key sets a state.
;  329     0403  1 !
;  330     0404  1 !                               The remaining bits are undefined and must be zero.
;  331     0405  1 !
;  332     0406  1 !       EQUIV_STRING    A string which is to be substituted for the keystroke
;  333     0407  1 !                       in the returned line.  Unless SMG$V_NOECHO is specified,
;  334     0408  1 !                       the equivalence string is echoed.  If omitted, no
;  335     0409  1 !                       equivalence string is defined for this key.
;  336     0410  1 !
;  337     0411  1 !       STATE_STRING    A string containing a new state name which is to become
;  338     0412  1 !                       the current state when this key is pressed.  If omitted,
;  339     0413  1 !                       no new state is defined.  If the current state is
;  340     0414  1 !                       temporary (SMG$V_KEY_LOCKSTATE was not specified for
;  341     0415  1 !                       the previous key), the current state reverts to
;  342     0416  1 !                       the default state.
;  343     0417  1 !
;  344     0418  1 ! IMPLICIT INPUTS:
;  345     0419  1 !
;  346     0420  1 !       NONE
;  347     0421  1 !
;  348     0422  1 ! IMPLICIT OUTPUTS:
;  349     0423  1 !
;  350     0424  1 !       NONE
;  351     0425  1 !
;  352     0426  1 ! COMPLETION STATUS:
;  353     0427  1 !
;  354     0428  1 !       SS$_NORMAL      Normal successful completion
;  355     0429  1 !       SMG$_INVDEFATT  Invalid key-definition attributes
;  356     0430  1 !       SMG$_INVKTB_ID  Invalid key-table id
;  357     0431  1 !       SMG$_KEYDEFPRO  Key-definition is protected against change or deletion
;  358     0432  1 !       SMG$_PREDEFREP  Previous key-definition replaced
;  359     0433  1 !       LIB$_INSVIRMEM  Insufficient virtual memory
;  360     0434  1 !       LIB$_xxx        Any error status from LIB$SCOPY_DXDX
;  361     0435  1 !
;  362     0436  1 ! SIDE EFFECTS:
;  363     0437  1 !
```

```
;  364     0438  1 !       NONE
;  365     0439  1 !
;  366     0440  1 !--
;  367     0441  1
;  368     0442  2     BEGIN
;  369     0443  2
;  370     0444  2     LOCAL
;  371     0445  2         KTH: REF KTH_R_KTH_STRUCT,              ! Key table header
;  372     0446  2         KDE: REF KDE_R_KDE_STRUCT,              ! Key definition entry
;  373     0447  2         NEW_KDE,                                ! Pointer to new KDE
;  374     0448  2         TKEY: VECTOR [34, BYTE],                ! Total key (state-key)
;  375     0449  2         TKEY_DESC: BLOCK [8, BYTE],             ! Descriptor for TKEY
;  376     0450  2         TKEY_DESC_PTR: REF BLOCK [8, BYTE],     ! Pointer to a TKEY desc
;  377     0451  2         TKEY_STATUS,                            ! Status from CONSTRUCT_TKEY
;  378     0452  2         INSERT_STATUS,                          ! Status from insert
;  379     0453  2         COPY_STATUS1, COPY_STATUS2,             ! Status from calls to
;  380     0454  2         COPY_STATUS3, COPY_STATUS4,             ! LIB$SCOPY_ routines
;  381     0455  2         RET_STATUS;                             ! Our return status
;  382     0456  2
;  383     0457  2     BUILTIN
;  384     0458  2         AP,
;  385     0459  2         NULLPARAMETER;
;  386     0460  2
;  387     0461  2     !+
;  388     0462  2     ! Define positions of arguments
;  389     0463  2     !-                                                                          ;
;  390     0464  2
;  391     0465  2     LITERAL
;  392     0466  2         K_KEY_TABLE_ID = 1,
;  393     0467  2         K_KEY_NAME = 2,
;  394     0468  2         K_IF_STATE = 3,
;  395     0469  2         K_ATTRIBUTES = 4,
;  396     0470  2         K_EQUIV_STRING = 5,
;  397     0471  2         K_STATE_STRING = 6;
;  398     0472  2
;  399     0473  2     !+
;  400     0474  2     ! Validate KEY_TABLE_ID and get the Key Table Header.
;  401     0475  2     !-
;  402     0476  2
;  403     0477  2     $SMG$VALIDATE_KTH (KEY_TABLE_ID, KTH);
;  404     0478  2
;  405     0479  2     !+
;  406     0480  2     ! Construct TKEY from IF_STATE and KEY_NAME.
;  407     0481  2     !-
;  408     0482  2
;  409     0483  2     TKEY_DESC [DSC$A_POINTER] = TKEY;
;  410     0484  2     TKEY_DESC_PTR = TKEY_DESC;
;  411     0485  2     TKEY_STATUS = CONSTRUCT_TKEY (.AP, .TKEY_DESC_PTR; TKEY_DESC_PTR);
;  412     0486  2     IF NOT .TKEY_STATUS
;  413     0487  2     THEN
;  414     0488  2         RETURN .TKEY_STATUS;
;  415     0489  2
;  416     0490  2     INSERT_STATUS = LIB$INSERT_TREE (
;  417     0491  2         KTH [KTH_A_TREEHEAD],   ! Address of treehead,
;  418     0492  2         TKEY_DESC_PTR [0,0,0,0],! Descriptor of TKEY,
;  419     0493  2         %REF(0),                ! Return address of existing entry
;  420     0494  2         COMPARE_ROUTINE,        ! Address of compare routine
```

```
;  421     0495  2          ALLOC_ROUTINE,          ! Address of allocation routine
;  422     0496  2          NEW_KDE,                ! Address of new KDE
;  423     0497  2          0);                     ! Context - not used
;  424     0498  2
;  425     0499  2      IF NOT .INSERT_STATUS
;  426     0500  2      THEN
;  427     0501  3          RETURN .INSERT_STATUS;
;  428     0502  2
;  429     0503  2      !+
;  430     0504  2      ! Check for previously defined key
;  431     0505  2      !-
;  432     0506  2
;  433     0507  2      KDE = .NEW_KDE;
;  434     0508  2      RET_STATUS = SS$_NORMAL;    ! Assume not previously defined
;   35     0509  2      IF .KDE [KDE_V_DEFINED]     ! Already defined?
;  436     0510  2      THEN
;  437     0511  3          BEGIN
;  438     0512  3          RET_STATUS = SMG$_PREDEFREP;    ! Success - Previous key definition replaced
;  439     0513  3          IF .KDE [KDE_V_PROTECTED]
;  440     0514  3          THEN
;  441     0515  3              RETURN SMG$_KEYDEFPRO;      ! Key definition protected
;  442     0516  2          END;
;  443     0517  2
;  444     0518  2      !+
;  445     0519  2      ! Turn off DEFINED and user attribute bits in KDE.  Other bits might
;  446     0520  2      ! still be set.
;  447     0521  2      ! Store items in new KDE.
;  448     0522  2      !-
;  449     0523  2
;  450     0524  2      KDE [KDE_V_DEFINED] = 0;
;  451     0525  2      KDE [KDE_V_USER_ATTR] = 0;
;  452     0526  2      IF NOT NULLPARAMETER (K_ATTRIBUTES) ! ATTRIBUTES present?
;  453     0527  2      THEN
;  454     0528  3          BEGIN
;  455     0529  3          KDE [KDE_V_USER_ATTR] = .ATTRIBUTES [0];
;  456     0530  3          KDE [KDE_V_SETSTATE] = 0;       ! Assume no state set.
;  457     0531  3          IF (.KDE [KDE_L_ATTR] AND KDE_M_RESERVED) NEQ 0
;  458     0532  3          THEN
;  459     0533  3              RETURN SMG$_INVDEFATT;      ! Invalid key-definition attributes
;  460     0534  3          IF NOT .KDE [KDE_V_TERMINATE]
;  461     0535  3          THEN
;  462     0536  3              KDE [KDE_V_NOECHO] = 0;     ! NOTERMINATE implies ECHO
;  463     0537  2          END;
;  464     0538  2
;  465     0539  2      COPY_STATUS1 = LIB$SCOPY_R_DX6 (.TKEY_DESC_PTR [DSC$W_LENGTH],
;  466     0540  2          .TKEY_DESC_PTR [DSC$A_POINTER], KDE [KDE_R_TKEY_DESC]);
;  467     0541  2      IF NOT .COPY_STATUS1
;  468     0542  2      THEN
;  469     0543  2          RETURN .COPY_STATUS1;
;  470     0544  2
;  471     0545  2      IF NOT NULLPARAMETER (K_EQUIV_STRING)
;  472     0546  2      THEN
;  473     0547  3          BEGIN
;  474     0548  3          COPY_STATUS2 = LIB$SCOPY_DXDX6 (EQUIV_STRING [0,0,0,0],
;  475     0549  3              KDE [KDE_R_EQUIV_DESC]);
;  476     0550  3          IF NOT .COPY_STATUS2
;  477     0551  3          THEN
```

```
;  478     0552  3              RETURN .COPY_STATUS2;
;  479     0553  3          END
;  480     0554  2      ELSE IF .KDE [KDE_A_EQUIV_POINTER] NEQA 0
;  481     0555  2      THEN
;  482     0556  2          LIB$SFREE1_DD6 (KDE [KDE_R_EQUIV_DESC]);
;  483     0557  2
;  484     0558  2      IF NOT NULLPARAMETER (K_STATE_STRING)       ! STATE present?
;  485     0559  2      THEN
;  486     0560  3          BEGIN
;  487     0561  3          COPY_STATUS3 = LIB$SCOPY_DXDX6 (STATE_STRING [0,0,0,0],
;  488     0562  3              KDE [KDE_R_STATE_DESC]);
;  489     0563  3          IF NOT .COPY_STATUS3
;  490     0564  3          THEN
;  491     0565  3              RETURN .COPY_STATUS3;
;  492     0566  3          KDE [KDE_V_SETSTATE] = 1;
;  493     0567  3          END
;  494     0568  2      ELSE IF .KDE [KDE_A_STATE_POINTER] NEQA 0
;  495     0569  2      THEN
;  496     0570  2          LIB$SFREE1_DD6 (KDE [KDE_R_STATE_DESC]);
;  497     0571  2
;  498     0572  2      !+
;  499     0573  2      ! If this is a control key, add that key's
;  500     0574  2      ! bit to the terminator mask.
;  501     0575  2      !-
;  502     0576  2
;  503     0577  2      KTH [KTH_L_TERM_MASK] = .KTH [KTH_L_TERM_MASK] OR
;  504     0578  2          .KDE [KDE_L_TERM_MASK];
;  505     0579  2
;  506     0580  2      !+
;  507     0581  2      ! Indicate that definition is valid and return success.
;  508     0582  2      !-
;  509     0583  2
;  510     0584  2      KDE [KDE_V_DEFINED] = 1;
;  511     0585  2      RETURN .RET_STATUS;
;  512     0586  2
;  513     0587  1      END;                                    ! End of routine SMG$ADD_KEY_DEF


                                                 .EXTRN  SMG$_INVKTB_ID

                                     0FFC 00000  .ENTRY  SMG$ADD_KEY_DEF, Save R2,R3,R4,R5,R6,R7,R8,-; 0325
                                                         R9,R10,R11
                                5E       34  C2 00002   SUBL2  #52, SP
                                5A   04  BC  D0 00005   MOVL   @KEY_TABLE_ID, KTH                      0477
                                         06  13 00009   BEQL   1$
                                5A   10  AA  D1 0000B   CMPL   16(KTH), KTH
                                         08  13 0000F   BEQL   2$
                                50 00000000G  8F  D0 00011 1$:  MOVL   #SMG$_INVKTB_ID, R0
                                         04  00018   RET
                             0C AE   10  AE  9E 00019 2$:  MOVAB  TKEY, TKEY_DESC+4                    0483
                                57   08  AE  9E 0001E   MOVAB  TKEY_DESC, TKEY_DESC_PTR               0484
                                56       5C  D0 00022   MOVL   AP, R6                                 0485
                          0000V CF            00  FB 00025   CALLS  #0, CONSTRUCT_TKEY
                                1E            50  E9 0002A   BLBC   TKEY_STATUS, 3$                   0486
                                              7E  D4 0002D   CLRL   -(SP)                              0492
                                         08   AE  9F 0002F   PUSHAB NEW_KDE
```

```
                                 0000V  CF  9F 00032          PUSHAB  ALLOC_ROUTINE
                                 0000V  CF  9F 00036          PUSHAB  COMPARE_ROUTINE
                                   10   AE  D4 0003A          CLRL    16(SP)                          ; 0493
                                   10   AE  9F 0003D          PUSHAB  16(SP)
                                        57  DD 00040          PUSHL   TKEY_DESC_PTR                   ; 0492
                                        5A  DD 00042          PUSHL   KTH
                00000000G       00      07  FB 00044          CALLS   #7, LIB$INSERT_TREE
                                60      50  E9 0004B 3$:      BLBC    INSERT_STATUS, 7$               ; 0499
                                58   04 AE  D0 0004E          MOVL    NEW_KDE, KDE                    ; 0507
                                5B      01  D0 00052          MOVL    #1, RET_STATUS                  ; 0508
                                59   30 A8  9E 00055          MOVAB   48(KDE), R9                     ; 0509
                                        69  D5 00059          TSTL    (R9)
                                        13  18 0005B          BGEQ    4$
                                5B 00000000G 8F  D0 0005D     MOVL    #SMG$_PREDEFREP, RET_STATUS     ; 0512
                08              69      03  E1 00064          BBC     #3, (R9), 4$                    ; 0513
                                50 00000000G 8F  D0 00068     MOVL    #SMG$_KEYDEFPRO, R0             ; 0515
                                        04  00 0006F          RET
                                69 800000FF 8F  CA 00070 4$:   BICL2   #-2147483393, (R9)             ; 0524
                                04      6C  91 00077          CMPB    (AP), #4                        ; 0526
                                        21  1F 0007A          BLSSU   6$
                                   10   AC  D5 0007C          TSTL    16(AP)
                                        1C  13 0007F          BEQL    6$
                                69   10 BC  90 00081          MOVB    @ATTRIBUTES, (R9)               ; 0529
                                69      10  8A 00085          BICB2   #16, (R9)                       ; 0530
                          E0    8F      69  93 00088          BITB    (R9), #224                      ; 0531
                                        08  13 0008C          BEQL    5$
                                50 00000000G 8F  D0 0008E     MOVL    #SMG$_INVDEFATT, R0             ; 0533
                                        04  00 00095          RET
                03              69      01  E0 00096 5$:      BBS     #1, (R9), 6$                    ; 0534
                                69      01  8A 0009A          BICB2   #1, (R9)                        ; 0536
                                52   0C A8  9E 0009D 6$:      MOVAB   12(KDE), R2                     ; 0540
                                51   04 A7  D0 000A1          MOVL    4(TKEY_DESC_PTR), R1
                                50      67  3C 000A5          MOVZWL  (TKEY_DESC_PTR), R0
                00000000G       00      16  00 000A8          JSB     LIB$SCOPY_R_DX6
                67                      50  E9 000AE 7$:      BLBC    COPY_STATUS1, 12$               ; 0541
                05                      6C  91 000B1          CMPB    (AP), #5                        ; 0545
                                        17  1F 000B4          BLSSU   8$
                                   14   AC  D5 000B6          TSTL    20(AP)
                                        12  13 000B9          BEQL    8$
                                51   14 A8  9E 000BB          MOVAB   20(KDE), R1                     ; 0549
                                50   14 AC  D0 000BF          MOVL    EQUIV_STRING, R0
                00000000G       00      16  00 000C3          JSB     LIB$SCOPY_DXDX6
                10                      50  E8 000C9          BLBS    COPY_STATUS2, 9$                ; 0550
                                        04  00 000CC          RET                                     ; 0552
                                   18   A8  D5 000CD 8$:      TSTL    24(KDE)                         ; 0554
                                        0A  13 000D0          BEQL    9$
                                50   14 A8  9E 000D2          MOVAB   20(KDE), R0                     ; 0556
                00000000G       00      16  00 000D6          JSB     LIB$SFREE1_DD6
                06                      6C  91 000DC 9$:      CMPB    (AP), #6                        ; 0558
                                        1B  1F 000DF          BLSSU   10$
                                   18   AC  D5 000E1          TSTL    24(AP)
                                        16  13 000E4          BEQL    10$
                                51   1C A8  9E 000E6          MOVAB   28(KDE), R1                     ; 0562
                                50   18 AC  D0 000EA          MOVL    STATE_STRING, R0
                00000000G       00      16  00 000EE          JSB     LIB$SCOPY_DXDX6
                21                      50  E9 000F4          BLBC    COPY_STATUS3, 12$               ; 0563
                69                      10  88 000F7          BISB2   #16, (R9)                       ; 0566
```

```
                                    0F  11 000FA          BRB     11$                             ; 0558
                                20  A8  D5 000FC 10$:     TSTL    32(KDE)                         ; 0568
                                    0A  13 000FF          BEQL    11$                             ;
                            50      1C  A8  9E 00101      MOVAB   28(KDE), R0                     ; 0570
                                00000000G   00  16 00105  JSB     LIB$SFREE1_DD6                  ;
                        0C  AA      34  A8  C8 0010B 11$:     BISL2   52(KDE), 12(KTH)            ; 0578
                        03  A9      80  8F  88 00110       BISB2   #128, 3(R9)                     ; 0584
                            50          5B  D0 00115       MOVL    RET_STATUS, R0                  ; 0585
                                        04  00118 12$:     RET                                     ; 0587

; Routine Size:  281 bytes,    Routine Base:  _SMG$CODE + 0050
```

```
;  515          0588  1 %SBTTL 'SMG$GET_KEY_DEF'
;  516          0589  1 GLOBAL ROUTINE SMG$GET_KEY_DEF (
;  517          0590  1     KEY_TABLE_ID: REF VECTOR [, LONG],
;  518          0591  1     KEY_NAME: REF BLOCK [, BYTE],
;  519          0592  1     IF_STATE: REF BLOCK [, BYTE],
;  520          0593  1     ATTRIBUTES: REF VECTOR [, LONG],
;  521          0594  1     EQUIV_STRING: REF BLOCK [, BYTE],
;  522          0595  1     STATE_STRING: REF BLOCK [, BYTE]
;  523          0596  1     ) =
;  524          0597  1
;  525          0598  1 !++
;  526          0599  1 ! FUNCTIONAL DESCRIPTION:
;  527          0600  1 !
;  528          0601  1 !       This procedure returns the key definition associated with a
;  529          0602  1 !       specified key and state name.
;  530          0603  1 !
;  531          0604  1 ! CALLING SEQUENCE:
;  532          0605  1 !
;  533          0606  1 !       RET_STATUS.wlc.v = SMG$GET_KEY_DEF (
;  534          0607  1 !                                   KEY_TABLE_ID.rl.r,
;  535          0608  1 !                                   KEY_NAME.rt.dx
;  536          0609  1 !                                   [, [IF_STATE.rt.dx]
;  537          0610  1 !                                   [, [ATTRIBUTES.wl.r]
;  538          0611  1 !                                   [, [EQUIV_STRING.wt.dx]
;  539          0612  1 !                                   [, [STATE_STRING.wt.dx] ]]]])
;  540          0613  1 !
;  541          0614  1 ! FORMAL PARAMETERS:
;  542          0615  1 !
;  543          0616  1 !       KEY_TABLE_ID    A longword containing the identification of the
;  544          0617  1 !                       key-definition table in which this key is
;  545          0618  1 !                       to be looked up.
;  546          0619  1 !
;  547          0620  1 !       KEY_NAME        A string containing the name of the key whose
;  548          0621  1 !                       definition is to be returned.
;  549          0622  1 !                       The key name is stripped of trailing blanks and is
;  550          0623  1 !                       converted to upper-case before use.  The key name must
;  551          0624  1 !                       be one of the key names listed in table _\_\ TBS _\_\.
;  552          0625  1 !
;  553          0626  1 !       IF_STATE        A string containing a state name which further qualifies
;  554          0627  1 !                       KEY_NAME. If omitted, the null state is used.
;  555          0628  1 !
;  556          0629  1 !       ATTRIBUTES      A longword into which is stored the attributes
;  557          0630  1 !                       bitmask for this key definition.  See the description
;  558          0631  1 !                       of SMG$ADD_KEY_DEF for more information.
;  559          0632  1 !
;  560          0633  1 !       EQUIV_STRING    A string into which is stored the equivalence-string
;  561          0634  1 !                       for this key definition.
;  562          0635  1 !
;  563          0636  1 !       STATE_STRING    A string into which is stored the new state-name,
;  564          0637  1 !                       if any, which is set by this key definition.  If this
;  565          0638  1 !                       key definition sets a state, the ATTRIBUTES flag
;  566          0639  1 !                       SMG$V_KEY_SETSTATE is set.
;  567          0640  1 !
;  568          0641  1 ! IMPLICIT INPUTS:
;  569          0642  1 !
;  570          0643  1 !       NONE
;  571          0644  1 !
```

```
;  572          0645  1 ! IMPLICIT OUTPUTS:
;  573          0646  1 !
;  574          0647  1 !       NONE
;  575          0648  1 !
;  576          0649  1 ! COMPLETION STATUS:
;  577          0650  1 !
;  578          0651  1 !       SS$_NORMAL      Normal successful completion
;  579          0652  1 !       SMG$_INVKTB_ID  Invalid key-table id
;  580          0653  1 !       SMG$_KEYNOTDEF  Key is not defined
;  581          0654  1 !       LIB$_xxx        Any error status from LIB$SCOPY_DXDX
;  582          0655  1 !
;  583          0656  1 ! SIDE EFFECTS:
;  584          0657  1 !
;  585          0658  1 !       NONE
;  586          0659  1 !
;  587          0660  1 !--
;  588          0661  1
;  589          0662  2     BEGIN
;  590          0663  2
;  591          0664  2     LOCAL
;  592          0665  2         FOUND_KDE,                          ! KDE found
;  593          0666  2         KDE: REF KDE_R_KDE_STRUCT,          ! Key definition entry
;  594          0667  2         KTH: REF KTH_R_KTH_STRUCT,          ! Key table header
;  595          0668  2         TKEY: VECTOR [34, BYTE],            ! Total key (state-key)
;  596          0669  2         TKEY_DESC: BLOCK [8, BYTE],         ! Descriptor for TKEY
;  597          0670  2         TKEY_DESC_PTR: REF BLOCK [8, BYTE], ! Pointer to TKEY_DESC
;  598          0671  2         TKEY_STATUS,                        ! Status from CONSTRUCT_TKEY
;  599          0672  2         LOOKUP_STATUS,                      ! Status from LIB$LOOKUP_TREE
;  600          0673  2         COPY_STATUS1, COPY_STATUS2,         ! Status from calls to
;  601          0674  2         COPY_STATUS3;                       ! LIB$SCOPY_ routines
;  602          0675  2
;  603          0676  2     BUILTIN
;  604          0677  2         AP,
;  605          0678  2         NULLPARAMETER;
;  606          0679  2
;  607          0680  2     !+
;  608          0681  2     ! Define positions of arguments
;  609          0682  2     !-
;  610          0683  2
;  611          0684  2     LITERAL
;  612          0685  2         K_KEY_TABLE_ID = 1,
;  613          0686  2         K_KEY_NAME = 2,
;  614          0687  2         K_IF_STATE = 3,
;  615          0688  2         K_ATTRIBUTES = 4,
;  616          0689  2         K_EQUIV_STRING = 5,
;  617          0690  2         K_STATE_STRING = 6;
;  618          0691  2
;  619          0692  2     !+
;  620          0693  2     ! Validate KEY_TABLE_ID and get the Key Table Header.
;  621          0694  2     !-
;  622          0695  2
;  623          0696  2     $SMG$VALIDATE_KTH (KEY_TABLE_ID, KTH);
;  624          0697  2
;  625          0698  2     !+
;  626          0699  2     ! Construct TKEY from IF_STATE and KEY_NAME.
;  627          0700  2     !-
;  628          0701  2
```

```
;  629          0702  2      TKEY_DESC [DSC$A_POINTER] = TKEY;
;  630          0703  2      TKEY_DESC_PTR = TKEY_DESC;
;  631          0704  2      TKEY_STATUS = CONSTRUCT_TKEY (.AP, .TKEY_DESC_PTR; TKEY_DESC_PTR);
;  632          0705  2      IF NOT .TKEY_STATUS
;  633          0706  2      THEN
;  634          0707  2          RETURN .TKEY_STATUS;
;  635          0708  2
;  636          0709  2      !+
;  637          0710  2      ! Look up key name.
;  638          0711  2      !-
;  639          0712  2
;  640          0713  2      LOOKUP_STATUS = LIB$LOOKUP_TREE (
;  641          0714  2          KTA [KTH_A_TREEHEAD],           ! Address of treehead
;  642          0715  2          .TKEY_DESC_PTR,                 ! Total state-key string
;  643          0716  2          COMPARE_ROUTINE,                ! Key comparison routine
;  644          0717  2          FOUND_KDE);                     ! Pointer to found KDE
;  645          0718  2      IF NOT .LOOKUP_STATUS
;  646          0719  2      THEN
;  647          0720  2          RETURN SMG$_KEYNOTDEF;          ! Key not defined
;  648          0721  2
;  649          0722  2      !+
;  650          0723  2      ! If KDE is not defined, return "not found".
;  651          0724  2      !-
;  652          0725  2
;  653          0726  2      KDE = .FOUND_KDE;
;  654          0727  2      IF NOT .KDE [KDE_V_DEFINED]
;  655          0728  2      THEN
;  656          0729  2          RETURN SMG$_KEYNOTDEF;  ! Key not defined
;  657          0730  2
;  658          0731  2      !+
;  659          0732  2      ! Store values.
;  660          0733  2      !-
;  661          0734  2
;  662          0735  2      IF NOT NULLPARAMETER (K_ATTRIBUTES)
;  663          0736  2      THEN
;  664          0737  2          ATTRIBUTES [0] = .KDE [KDE_L_ATTR] AND KDE_M_USER_ATTR;
;  665          0738  2
;  666          0739  2      IF NOT NULLPARAMETER (K_EQUIV_STRING)
;  667          0740  2      THEN
;  668          0741  3          BEGIN
;  669          0742  3          COPY_STATUS1 = LIB$SCOPY_DXDX6 (KDE [KDE_R_EQUIV_DESC],
;  670          0743  3              EQUIV_STRING [0,0,0,0]);
;  671          0744  3          IF NOT .COPY_STATUS1
;  672          0745  3          THEN
;  673          0746  3              RETURN .COPY_STATUS1;
;  674          0747  2          END;
;  675          0748  2
;  676          0749  2      IF NOT NULLPARAMETER (K_STATE_STRING)
;  677          0750  2      THEN
;  678          0751  3          BEGIN
;  679          0752  3          COPY_STATUS2 = LIB$SCOPY_DXDX6 (KDE [KDE_R_STATE_DESC],
;  680          0753  3              STATE_STRING [0,0,0,0]);
;  681          0754  3          IF NOT .COPY_STATUS2
;  682          0755  3          THEN
;  683          0756  3              RETURN .COPY_STATUS2;
;  684          0757  2          END;
;  685          0758  2
```

```
;  686          0759  2     RETURN SS$_NORMAL;
;  687          0760  2
;  688          0761  1     END;                                    ! End of routine SMG$GET_KEY_DEF


                                    01FC 00000          .ENTRY  SMG$GET_KEY_DEF, Save R2,R3,R4,R5,R6,R7,R8 ; 0589
                58 00000000G    00  9E 00002          MOVAB   LIB$SCOPY_DXDX6, R8
                5E              30  C2 00009          SUBL2   #48, SP
                52          04  BC  D0 0000C          MOVL    @KEY_TABLE_ID, KTH                          ; 0696
                                06  13 00010          BEQL    1$
                52          10  A2  D1 00012          CMPL    16(KTH), KTH
                                08  13 00016          BEQL    2$
                50 00000000G    8F  D0 00018 1$:      MOVL    #SMG$_INVKTB_ID, R0
                                    04 0001F          RET
             08 AE          0C  AE  9E 00020 2$:      MOVAB   TKEY, TKEY_DESC+4                           ; 0702
                57          04  AE  9E 00025          MOVAB   TKEY_DESC, TKEY_DESC_PTR                    ; 0703
                56              5C  D0 00029          MOVL    AP, R6                                      ; 0704
          0000V CF              00  FB 0002C          CALLS   #0, CONSTRUCT_TKEY
                64              50  E9 00031          BLBC    TKEY_STATUS, 8$                             ; 0705
                                5E  DD 00034          PUSHL   SP                                          ; 0714
                            0000V CF  9F 00036        PUSHAB  COMPARE_ROUTINE
                            0084  8F  BB 0003A        PUSHR   #^M<R2,R7>
    00000000G   00              04  FB 0003E          CALLS   #4, LIB$LOOKUP_TREE
                08              50  E9 00045          BLBC    LOOKUP_STATUS, 3$                           ; 0718
                57              6E  D0 00048          MOVL    FOUND_KDE, KDE                              ; 0726
                            33  A7  95 0004B          TSTB    51(KDE)                                     ; 0727
                                08  19 0004E          BLSS    4$
                50 00000000G    8F  D0 00050 3$:      MOVL    #SMG$_KEYNOTDEF, R0                         ; 0729
                                    04 00057          RET
                04              6C  91 00058 4$:      CMPB    (AP), #4                                    ; 0735
                                0A  1F 0005B          BLSSU   5$
                            10  AC  D5 0005D          TSTL    16(AP)
                                05  13 00060          BEQL    5$
             10 BC          30  A7  9A 00062          MOVZBL  48(KDE), @ATTRIBUTES                        ; 0737
                05              6C  91 00067 5$:      CMPB    (AP), #5                                    ; 0739
                                12  1F 0006A          BLSSU   6$
                            14  AC  D5 0006C          TSTL    20(AP)
                                0D  13 0006F          BEQL    6$
                50          14  A7  9E 00071          MOVAB   20(KDE), R0                                 ; 0742
                51          14  AC  D0 00075          MOVL    EQUIV_STRING, R1                            ; 0743
                                68  16 00079          JSB     LIB$SCOPY_DXDX6
                1A              50  E9 0007B          BLBC    COPY_STATUS1, 8$                            ; 0744
                06              6C  91 0007E 6$:      CMPB    (AP), #6                                    ; 0749
                                12  1F 00081          BLSSU   7$
                            18  AC  D5 00083          TSTL    24(AP)
                                0D  13 00086          BEQL    7$
                50          1C  A7  9E 00088          MOVAB   28(KDE), R0                                 ; 0752
                51          18  AC  D0 0008C          MOVL    STATE_STRING, R1                            ; 0753
                                68  16 00090          JSB     LIB$SCOPY_DXDX6
                03              50  E9 00092          BLBC    COPY_STATUS2, 8$                            ; 0754
                50              01  D0 00095 7$:      MOVL    #1, R0                                      ; 0759
                                    04 00098 8$:      RET                                                 ; 0761

; Routine Size:  153 bytes,    Routine Base:  _SMG$CODE + 0169
```

```
;  690          0762  1 %SBTTL 'SMG$DELETE_KEY_DEF'
;  691          0763  1 GLOBAL ROUTINE SMG$DELETE_KEY_DEF (
;  692          0764  1     KEY_TABLE_ID: REF VECTOR [, LONG],
;  693          0765  1     KEY_NAME: REF BLOCK [, BYTE],
;  694          0766  1     IF_STATE: REF BLOCK [, BYTE]
;  695          0767  1     ) =
;  696          0768  1
;  697          0769  1 !++
;  698          0770  1 ! FUNCTIONAL DESCRIPTION:
;  699          0771  1 !
;  700          0772  1 !       This procedure deletes a key definition from the specified table
;  701          0773  1 !       of key definitions.
;  702          0774  1 !
;  703          0775  1 ! CALLING SEQUENCE:
;  704          0776  1 !
;  705          0777  1 !       RET_STATUS.wlc.v = SMG$DELETE_KEY_DEF (
;  706          0778  1 !                                  KEY_TABLE_ID.rl.r,
;  707          0779  1 !                                  KEY_NAME.rt.dx
;  708          0780  1 !                                  [, [IF_STATE.rt.dx] ])
;  709          0781  1 !
;  710          0782  1 ! FORMAL PARAMETERS:
;  711          0783  1 !
;  712          0784  1 !       KEY_TABLE_ID    A longword containing the identification of the
;  713          0785  1 !                       key-definition table from which this key is
;  714          0786  1 !                       to be deleted.
;  715          0787  1 !
;  716          0788  1 !       KEY_NAME        A string containing the name of the key which is
;  717          0789  1 !                       to be deleted.
;  718          0790  1 !                       The key name is stripped of trailing blanks and is
;  719          0791  1 !                       converted to upper-case before use.  The key name must
;  720          0792  1 !                       be one of the key names listed in table _\_\ TBS _\_\.
;  721          0793  1 !
;  722          0794  1 !       IF_STATE        A string containing a state name which further qualifies
;  723          0795  1 !                       KEY_NAME. If omitted, the null state is used.
;  724          0796  1 !
;  725          0797  1 ! IMPLICIT INPUTS:
;  726          0798  1 !
;  727          0799  1 !       NONE
;  728          0800  1 !
;  729          0801  1 ! IMPLICIT OUTPUTS:
;  730          0802  1 !
;  731          0803  1 !       NONE
;  732          0804  1 !
;  733          0805  1 ! COMPLETION STATUS:
;  734          0806  1 !
;  735          0807  1 !       SS$_NORMAL      Normal successful completion
;  736          0808  1 !       SMG$_INVKTB_ID  Invalid key-table id
;  737          0809  1 !       SMG$_KEYNOTDEF  Key is not defined
;  738          0810  1 !       SMG$_KEYDEFPRO  Key definition is protected
;  739          0811  1 !
;  740          0812  1 ! SIDE EFFECTS:
;  741          0813  1 !
;  742          0814  1 !       NONE
;  743          0815  1 !
;  744          0816  1 !--
;  745          0817  1
;  746          0818  2     BEGIN
```

```
;  747     0819  2
;  748     0820  2      LOCAL
;  749     0821  2          FOUND_KDE,                          ! KDE found
;  750     0822  2          KDE: REF KDE_R_KDE_STRUCT,          ! Key definition entry
;  751     0823  2          KTH: REF KTH_R_KTH_STRUCT,          ! Ket table header
;  752     0824  2          TKEY: VECTOR [34, BYTE],            ! Total key (state-key)
;  753     0825  2          TKEY_DESC: BLOCK [8, BYTE],         ! Descriptor for TKEY
;  754     0826  2          TKEY_DESC_PTR: REF BLOCK [8, BYTE], ! Pointer to a TKEY desc
;  755     0827  2          TKEY_STATUS,                        ! Status from CONSTRUCT_TKEY
;  756     0828  2          LOOKUP_STATUS;                      ! Status from LIB$LOOKUP_TREE
;  757     0829  2
;  758     0830  2      BUILTIN
;  759     0831  2          AP;
;  760     0832  2
;  761     0833  2      !+
;  762     0834  2      ! Define positions of arguments
;  763     0835  2      !-
;  764     0836  2
;  765     0837  2      LITERAL
;  766     0838  2          K_KEY_TABLE_ID = 1,
;  767     0839  2          K_KEY_NAME = 2,
;  768     0840  2          K_IF_STATE = 3;
;  769     0841  2
;  770     0842  2      !+
;  771     0843  2      ! Validate KEY_TABLE_ID and get the Key Table Header.
;  772     0844  2      !-
;  773     0845  2
;  774     0846  2      $SMG$VALIDATE_KTH (KEY_TABLE_ID, KTH);
;  775     0847  2
;  776     0848  2      !+
;  777     0849  2      ! Construct TKEY from IF_STATE and KEY_NAME.
;  778     0850  2      !-
;  779     0851  2
;  780     0852  2      TKEY_DESC [DSC$A_POINTER] = TKEY;
;  781     0853  2      TKEY_DESC_PTR = TKEY_DESC;
;  782     0854  2      TKEY_STATUS = CONSTRUCT_TKEY (.AP, .TKEY_DESC_PTR; TKEY_DESC_PTR);
;  783     0855  2      IF NOT .TKEY_STATUS
;  784     0856  2      THEN
;  785     0857  2          RETURN .TKEY_STATUS;
;  786     0858  2
;  787     0859  2      !+
;  788     0860  2      ! Look up key name.
;  789     0861  2      !-
;  790     0862  2
;  791     0863  2      LOOKUP_STATUS = LIB$LOOKUP_TREE (
;  792     0864  2          KTH [KTH_A_TREEHEAD],           ! Address of treehead
;  793     0865  2          .TKEY_DESC_PTR,                 ! Total state-key string
;  794     0866  2          COMPARE_ROUTINE,                ! Key comparison routine
;  795     0867  2          FOUND_KDE);                     ! Pointer to found KDE
;  796     0868  2      IF NOT .LOOKUP_STATUS
;  797     0869  2      THEN
;  798     0870  2          RETURN SMG$_KEYNOTDEF;  ! Key not defined
;  799     0871  2
;  800     0872  2      KDE = .FOUND_KDE;
;  801     0873  2      IF NOT .KDE [KDE_V_DEFINED]
;  802     0874  2      THEN
;  803     0875  2          RETURN SMG$_KEYNOTDEF;  ! Key not defined
```

```
;  804          0876  2
;  805          0877  2      !+
;  806          0878  2      ! Delete key definition by freeing the strings and clearing KDE_V_DEFINED.
;  807          0879  2      ! We can't actually remove this KDE from the tree, but it can be reused if
;  808          0880  2      ! the same key is redefined.
;  809          0881  2      !-
;  810          0882  2
;  811          0883  2      KDE [KDE_V_DEFINED] = 0;     ! Invalidate KDE
;  812          0884  2      IF .KDE [KDE_A_EQUIV_POINTER] NEQA 0
;  813          0885  2      THEN
;  814          0886  2          LIB$SFREE1_DD6 (KDE [KDE_R_EQUIV_DESC]);
;  815          0887  2      IF .KDE [KDE_A_STATE_POINTER] NEQA 0
;  816          0888  2      THEN
;  817          0889  2          LIB$SFREE1_DD6 (KDE [KDE_R_STATE_DESC]);
;  818          0890  2      IF .KDE [KDE_A_PREFIX_POINTER] NEQA 0
;  819          0891  2      THEN
;  820          0892  2          LIB$SFREE1_DD6 (KDE [KDE_R_PREFIX_DESC]);
;  821          0893  2
;  822          0894  2      !+
;  823          0895  2      ! If this was a control key, reflect the deletion
;  824          0896  2      ! in the KTH.
;  825          0897  2      !-
;  826          0898  2
;  827          0899  2      KTH [KTH_L_TERM_MASK] = .KTH [KTH_L_TERM_MASK] AND
;  828          0900  2          (NOT .RDE [RDE_L_TERM_MASK]);
;  829          0901  2
;  830          0902  2      !+
;  831          0903  2      ! Note that we can't free TKEY since it needs to be compared
;  832          0904  2      ! against in future searches.
;  833          0905  2      !-
;  834          0906  2
;  835          0907  2      RETURN SS$_NORMAL;
;  836          0908  2
;  837          0909  1      END;                                    ! End of routine SMG$GET_KEY_DEF


                                    03FC 00000          .ENTRY  SMG$DELETE_KEY_DEF, Save R2,R3,R4,R5,R6,R7,-  ; 0763
                                                                R8,R9
                59 00000000G    00  9E 00002          MOVAB   LIB$SFREE1_DD6, R9
                5E              30  C2 00009          SUBL2   #48, SP
                58        04    BC  D0 0000C          MOVL    @KEY_TABLE_ID, KTH                                ; 0846
                                06  13 00010          BEQL    1$
                58        10    A8  D1 00012          CMPL    16(KTH), KTH
                                08  13 00016          BEQL    2$
                50 00000000G    8F  D0 00018 1$:      MOVL    #SMG$_INVKTB_ID, R0
                                04  00 0001F          RET
             08 AE        0C    AE  9E 00020 2$:      MOVAB   TKEY, TKEY_DESC+4                                 ; 0852
                57        04    AE  9E 00025          MOVAB   TKEY_DESC, TKEY_DESC_PTR                          ; 0853
                56              5C  D0 00029          MOVL    AP, R6                                            ; 0854
          0000V CF              00  FB 0002C          CALLS   #0, CONSTRUCT_TKEY
                52              50  E9 00031          BLBC    TKEY_STATUS, 8$                                   ; 0855
                                5E  DD 00034          PUSHL   SP                                                ; 0864
                      0000V     CF  9F 00036          PUSHAB  COMPARE_ROUTINE
                                57  DD 0003A          PUSHL   TKEY_DESC_PTR                                     ; 0865
```

```
                                                58  DD 0003C          PUSHL   KTH                                ; 0864
                                    00000000G   00                    04  FB 0003E          CALLS   #4, LIB$LOOKUP_TREE
                                                08                    50  E9 00045          BLBC    LOOKUP_STATUS, 3$          ; 0868
                                                57                    6E  D0 00048          MOVL    FOUND_RDE, KDE             ; 0872
                                                          33          A7  95 0004B          TSTB    51(KDE)                    ; 0873
                                                                      08  19 0004E          BLSS    4$
                                                50 00000000G          8F  D0 00050  3$:     MOVL    #SMG$_KEYNOTDEF, R0        ; 0875
                                                                          04 00057          RET
                                          33    A7          80      8F  8A 00058  4$:     BICB2   #128, 51(KDE)              ; 0883
                                                          18          A7  D5 0005D          TSTL    24(KDE)                    ; 0884
                                                                      06  13 00060          BEQL    5$
                                                50        14          A7  9E 00062          MOVAB   20(KDE), R0                ; 0886
                                                                      69  16 00066          JSB     LIB$SFREE1_DD6
                                                          20          A7  D5 00068  5$:     TSTL    32(KDE)                    ; 0887
                                                                      06  13 0006B          BEQL    6$
                                                50        1C          A7  9E 0006D          MOVAB   28(KDE), R0                ; 0889
                                                                      69  16 00071          JSB     LIB$SFREE1_DD6
                                                          28          A7  D5 00073  6$:     TSTL    40(KDE)                    ; 0890
                                                                      06  13 00076          BEQL    7$
                                                50        24          A7  9E 00078          MOVAB   36(KDE), R0                ; 0892
                                                                      69  16 0007C          JSB     LIB$SFREE1_DD6
                                          0C    A8          34      A7  CA 0007E  7$:     BICL2   52(KDE), 12(KTH)           ; 0900
                                                50                    01  D0 00083          MOVL    #1, R0                     ; 0907
                                                                          04 00086  8$:     RET                                ; 0909

; Routine Size:  135 bytes,    Routine Base:  _SMG$CODE + 0202
```

```
;  839          0910  1 %SBTTL 'SMG$SET_DEFAULT_STATE'
;  840          0911  1 GLOBAL ROUTINE SMG$SET_DEFAULT_STATE (
;  841          0912  1     KEY_TABLE_ID: REF VECTOR [, LONG],
;  842          0913  1     NEW_STATE: REF BLOCK [, BYTE],
;  843          0914  1     OLD_STATE: REF BLOCK [, BYTE]
;  844          0915  1     ) =
;  845          0916  1
;  846          0917  1 !++
;  847          0918  1 ! FUNCTIONAL DESCRIPTION:
;  848          0919  1 !
;  849          0920  1 !       This procedure sets a new default state name and/or returns the
;  850          0921  1 !       default state name for a key definition table.
;  851          0922  1 !
;  852          0923  1 ! CALLING SEQUENCE:
;  853          0924  1 !
;  854          0925  1 !       RET_STATUS.wlc.v = SMG$SET_DEFAULT_STATE (
;  855          0926  1 !                                  KEY_TABLE_ID.rl.r,
;  856          0927  1 !                                  [NEW_STATE.rt.dx]
;  857          0928  1 !                                  [, [OLD_STATE.wt.dx]])
;  858          0929  1 !
;  859          0930  1 ! FORMAL PARAMETERS:
;  860          0931  1 !
;  861          0932  1 !       KEY_TABLE_ID    A longword containing the identification of the
;  862          0933  1 !                       key-definition table for which the default state
;  863          0934  1 !                       name is to be set or returned.
;  864          0935  1 !
;  865          0936  1 !       NEW_STATE       A string containing the new default state name to be set
;  866          0937  1 !                       for this key-definition table.  If omitted, the
;  867          0938  1 !                       existing default state name is unchanged.
;  868          0939  1 !                       The state name is converted to upper case and is
;  869          0940  1 !                       stripped of trailing blanks before use.
;  870          0941  1 !
;  871          0942  1 !       OLD_STATE       A string into which is written the existing default
;  872          0943  1 !                       state name of the key-definition table.
;  873          0944  1 !
;  874          0945  1 ! IMPLICIT INPUTS:
;  875          0946  1 !
;  876          0947  1 !       NONE
;  877          0948  1 !
;  878          0949  1 ! IMPLICIT OUTPUTS:
;  879          0950  1 !
;  880          0951  1 !       NONE
;  881          0952  1 !
;  882          0953  1 ! COMPLETION STATUS:
;  883          0954  1 !
;  884          0955  1 !       SS$_NORMAL      Normal successful completion
;  885          0956  1 !       SMG$_INVKTB_ID  Invalid key-table id
;  886          0957  1 !       SMG$_INVSTANAM  Invalid state name
;  887          0958  1 !       LIB$_INVSTRDES  Invalid string descriptor
;  888          0959  1 !
;  889          0960  1 ! SIDE EFFECTS:
;  890          0961  1 !
;  891          0962  1 !       NONE
;  892          0963  1 !
;  893          0964  1 !--
;  894          0965  1
;  895          0966  2     BEGIN
```

```
;  896          0967  2
;  897          0968  2      LOCAL
;  898          0969  2          KTH: REF KTH_R_KTH_STRUCT;              ! Key table header
;  899          0970  2
;  900          0971  2      BUILTIN
;  901          0972  2          NULLPARAMETER;
;  902          0973  2
;  903          0974  2      !+
;  904          0975  2      ! Define positions of arguments
;  905          0976  2      !-
;  906          0977  2
;  907          0978  2      LITERAL
;  908          0979  2          K_KEY_TABLE_ID = 1,
;  909          0980  2          K_NEW_STATE= 2,
;  910          0981  2          K_OLD_STATE = 3;
;  911          0982  2
;  912          0983  2      !+
;  913          0984  2      ! Validate KEY_TABLE_ID and get the Key Table Header.
;  914          0985  2      !-
;  915          0986  2
;  916          0987  2      $SMG$VALIDATE_KTH (KEY_TABLE_ID, KTH);
;  917          0988  2
;  918          0989  2      !+
;  919          0990  2      ! Return the old default state name, if requested.
;  920          0991  2      !-
;  921          0992  2
;  922          0993  2      IF NOT NULLPARAMETER (K_OLD_STATE)
;  923          0994  2      THEN
;  924          0995  3          BEGIN
;  925          0996  3          LOCAL
;  926          0997  3              COPY_STATUS;
;  927          0998  3          COPY_STATUS = LIB$SCOPY_R_DX6 ((.KTH [KTH_W_DEF_STATE_LEN] - 3),
;  928          0999  3              KTH [KTH_T_DEF_STATE_STRING], OLD_STATE [0,0,0,0]);
;  929          1000  3          IF NOT .COPY_STATUS
;  930          1001  3          THEN
;  931          1002  3              RETURN .COPY_STATUS;
;  932          1003  2          END;
;  933          1004  2
;  934          1005  2      !+
;  935          1006  2      ! Set the new default state, if requested.
;  936          1007  2      !-
;  937          1008  2
;  938          1009  2      IF NOT NULLPARAMETER (K_NEW_STATE)
;  939          1010  2      THEN
;  940          1011  3          BEGIN
;  941          1012  3          LOCAL
;  942          1013  3              NEW_STATE_PTR,              ! Pointer to string
;  943          1014  3              NEW_STATE_LEN: WORD,        ! Length of string
;  944          1015  3              CHAR_PTR,                   ! Pointer to current character
;  945          1016  3              KEY_PTR,                    ! Pointer to current pos in TKEY
;  946          1017  3              ANL_STATUS;                 ! Status from LIB$ANALYZE_SDESC
;  947          1018  3
;  948          1019  3          !+
;  949          1020  3          ! Analyze NEW_STATE string.
;  950          1021  3          !-
;  951          1022  3
;  952          1023  3          ANL_STATUS = LIB$ANALYZE_SDESC_R2 (NEW_STATE [0,0,0,0];
```

```
;  953          1024  3              NEW_STATE_LEN, NEW_STATE_PTR);
;  954          1025  3          IF NOT .ANL_STATUS
;  955          1026  3          THEN
;  956          1027  3              RETURN .ANL_STATUS;
;  957          1028  3
;  958          1029  3          IF .NEW_STATE_LEN GTRU 31
;  959          1030  3          THEN
;  960          1031  3              RETURN SMG$_INVSTANAM;      ! Invalid state name
;  961          1032  3
;  962          1033  3          !+
;  963          1034  3          ! Find position of last non-blank character.
;  964          1035  3          !-
;  965          1036  3
;  966          1037  3          CHAR_PTR = .NEW_STATE_PTR + .NEW_STATE_LEN;
;  967          1038  3          WHILE ((CHAR_PTR=.CHAR_PTR-1;CH$RCHAR(.CHAR_PTR)) EQLU %C' ') DO
;  968          1039  4              BEGIN
;  969          1040  4              NEW_STATE_LEN = .NEW_STATE_LEN - 1;
;  970          1041  4              IF .NEW_STATE_LEN EQL 0 THEN EXITLOOP;
;  971          1042  3              END;
;  972          1043  3
;  973          1044  3          KEY_PTR = CH$TRANSLATE (LIB$AB_UPCASE, .NEW_STATE_LEN, .NEW_STATE_PTR,
;  974          1045  3              0, .NEW_STATE_LEN, KTH [KTH_T_DEF_STATE_STRING]);
;  975          1046  3          CH$WCHAR_A (0, KEY_PTR);
;  976          1047  3          KTH [KTH_A_DEF_KEYCODE] = .KEY_PTR;
;  977          1048  3          KTH [KTH_W_DEF_STATE_LEN] = .NEW_STATE_LEN + 3;
;  978          1049  2          END;
;  979          1050  2
;  980          1051  2      RETURN SS$_NORMAL;
;  981          1052  2
;  982          1053  1      END;                                ! End of routine SMG$SET_DEFAULT_STATE


                                00FC 00000          .ENTRY  SMG$SET_DEFAULT_STATE, Save R2,R3,R4,R5,R6,-; 0911
                                                            R7                                            ;
57          04      BC   D0 00002          MOVL    @KEY_TABLE_ID, KTH                                    ; 0987
                    06   13 00006          BEQL    1$                                                    ;
57          10      A7   D1 00008          CMPL    16(KTH), KTH                                          ;
                    08   13 0000C          BEQL    2$                                                    ;
50 00000000G        8F   D0 0000E 1$:      MOVL    #SMG$_INVKTB_ID, R0                                   ;
                         04 00015          RET                                                           ;
03                  6C   91 00016 2$:      CMPB    (AP), #3                                              ; 0993
                    1D   1F 00019          BLSSU   3$                                                    ;
            0C      AC   D5 0001B          TSTL    12(AP)                                                ;
                    18   13 0001E          BEQL    3$                                                    ;
51          20      A7   9E 00020          MOVAB   32(KTH), R1                                           ; 0999
50          14      A7   3C 00024          MOVZWL  20(KTH), R0                                           ; 0998
50                  03   C2 00028          SUBL2   #3, R0                                                ;
52          0C      AC   D0 0002B          MOVL    OLD_STATE, R2                                         ; 0999
   00000000G        00   16 0002F          JSB     LIB$SCOPY_R_DX6                                       ;
50                  50   E9 00035          BLBC    COPY_STATUS, 8$                                       ; 1000
02                  6C   91 00038 3$:      CMPB    (AP), #2                                              ; 1009
                    48   1F 0003B          BLSSU   7$                                                    ;
            08      AC   D5 0003D          TSTL    8(AP)                                                 ;
                    43   13 00040          BEQL    7$                                                    ;
```

```
                                    50        08   AC  D0 00042          MOVL    NEW_STATE, R0                                   ; 1023
                                       00000000G  00  16 00046          JSB     LIB$ANALYZE_SDESC_R2                            ;
                                    56             51  D0 0004C          MOVL    R1, NEW_STATE_LEN                               ;
                                    36             50  E9 0004F          BLBC    ANL_STATUS, 9$                                  ; 1025
                                    1F             56  B1 00052          CMPW    NEW_STATE_LE   #31                              ; 1029
                                                   08  1B 00055          BLEQU   4$                                              ;
                                    50 00000000G   8F  D0 00057          MOVL    #SMG$_INVSTANAM, R0                             ; 1031
                                                       04 0005E          RET                                                     ;
                                    50             56  3C 0005F  4$:     MOVZWL  NEW_STATE_LEN, CHAR_PTR                         ; 1037
                                    50             52  C0 00062          ADDL2   NEW_STATE_PTR, CHAR_PTR                         ;
                                    20             70  91 00065  5$:     CMPB    -(CHAR_PTR), #32                                ; 1038
                                                   04  12 00068          BNEQ    6$                                              ;
                                                   56  B7 0006A          DECW    NEW_STATE_LEN                                   ; 1040
                                                   F7  12 0006C          BNEQ    5$                                              ; 1041
00000000G  00          00       62             56  2E 0006E  6$:     MOVTC   NEW_STATE_LEN, (NEW_STATE_PTR), #0, -           ; 1045
                          20    A7             56      00077                  LIB$AB_UPCASE, NEW_STATE_LEN, 32(KTH)           ;
                                                   85  94 0007A          CLRB    (KEY_PTR)+                                      ; 1046
                          1C    A7             55  D0 0007C          MOVL    KEY_PTR, 28(KTH)                                ; 1047
                    14    A7                   56  03  A1 00080          ADDW3   #3, NEW_STATE_LEN, 2 (KTH)                      ; 1048
                                    50             01  D0 00085  7$:     MOVL    #1, R0                                          ; 1051
                                                       04 00088  8$:     RET                                                     ; 1053

; Routine Size:  137 bytes,    Routine Base:  _SMG$CODE + 0289
```

```
;  984          1054  1 %SBTTL 'SMG$$LOOKUP_KEY'
;  985          1055  1 GLOBAL ROUTINE SMG$$LOOKUP_KEY (
;  986          1056  1     KTH: REF KTH_R_KTH_STRUCT,
;  987          1057  1     TKEY_DESC: REF BLOCK [, BYTE],
;  988          1058  1     KDE: REF KDE_R_KDE_STRUCT
;  989          1059  1     ): SMG$$LOOKUP_KEY$LNK =
;  990          1060  1
;  991          1061  1 !++
;  992          1062  1 ! FUNCTIONAL DESCRIPTION:
;  993          1063  1 !
;  994          1064  1 !       This procedure is called by SMG$READ_COMPOSED_LINE to return
;  995          1065  1 !       the Key Definition Entry for a given "total key".
;  996          1066  1 !
;  997          1067  1 ! CALLING SEQUENCE:
;  998          1068  1 !
;  999          1069  1 !       ret_status.wlc.v = SMG$$LOOKUP_KEY (
; 1000          1070  1 !                                   KTH.rr.r, TKEY_DESC.rr.r, KDE.wr.r)
; 1001          1071  1 !
; 1002          1072  1 ! FORMAL PARAMETERS:
; 1003          1073  1 !
; 1004          1074  1 !
; 1005          1075  1 ! IMPLICIT INPUTS:
; 1006          1076  1 !
; 1007          1077  1 !       NONE
; 1008          1078  1 !
; 1009          1079  1 ! IMPLICIT OUTPUTS:
; 1010          1080  1 !
; 1011          1081  1 !       NONE
; 1012          1082  1 !
; 1013          1083  1 ! COMPLETION STATUS:
; 1014          1084  1 !
; 1015          1085  1 !       SS$_NORMAL      Normal successful completion
; 1016          1086  1 !
; 1017          1087  1 ! SIDE EFFECTS:
; 1018          1088  1 !
; 1019          1089  1 !       NONE
; 1020          1090  1 !
; 1021          1091  1 !--
; 1022          1092  1
; 1023          1093  2     BEGIN
; 1024          1094  2
; 1025          1095  2     LOCAL
; 1026          1096  2         KEYCODE: REF VECTOR [2, BYTE],          ! Key code
; 1027          1097  2         TEMP: BYTE,                             ! Temporary
; 1028          1098  2         FOUND_KDE,                              ! KDE found
; 1029          1099  2         LOOKUP_STATUS;                          ! Status from LIB$LOOKUP_TREE
; 1030          1100  2
; 1031          1101  2     !+
; 1032          1102  2     ! Swap the two bytes of the key code so that a string comparison works.
; 1033          1103  2     !-
; 1034          1104  2
; 1035          1105  2     KEYCODE = .TKEY_DESC [DSC$A_POINTER] + (.TKEY_DESC [DSC$W_LENGTH] - 2);
; 1036          1106  2     TEMP = .KEYCODE [0];
; 1037          1107  2     KEYCODE [0] = .KEYCODE [1];
; 1038          1108  2     KEYCODE [1] = .TEMP;
; 1039          1109  2
; 1040          1110  2     !+
```

```
;  1041          1111  2     ! Look up key name.
;  1042          1112  2     !-
;  1043          1113  2
;  1044          1114  2     LOOKUP_STATUS = LIB$LOOKUP_TREE (
;  1045          1115  2         KTH [KTH_A_TREEHEAD],                   ! Address of treehead
;  1046          1116  2         TKEY_DESC [0,0,0,0],                    ! Total state-key string
;  1047          1117  2         COMPARE_ROUTINE,                        ! Key comparison routine
;  1048          1118  2         FOUND_KDE);                             ! Pointer to found KDE
;  1049          1119  2     IF NOT .LOOKUP_STATUS
;  1050          1120  2     THEN
;  1051          1121  2         RETURN SMG$_KEYNOTDEF;                  ! Key not defined
;  1052          1122  2
;  1053          1123  2     !+
;  1054          1124  2     ! If KDE is not defined, return "not found".
;  1055          1125  2     !-
;  1056          1126  2
;  1057          1127  2     KDE = .FOUND_KDE;
;  1058          1128  2     IF NOT .KDE [KDE_V_DEFINED]
;  1059          1129  2     THEN
;  1060          1130  2         RETURN SMG$_KEYNOTDEF;  ! Key not defined
;  1061          1131  2
;  1062          1132  2     RETURN SS$_NORMAL;
;  1063          1133  2
;  1064          1134  1     END;                                        ! End of routine SMG$$LOOKUP_KEY


                              5E              08  C2 00000 SMG$$LOOKUP_KEY::
                                                              SUBL2   #8, SP                          ;  1055
                              56              51  D0 00003         MOVL    R1, R6                      ;
                              51              66  3C 00006         MOVZWL  (TKEY_DESC), R1             ;  1105
                              51        04    A6  C0 00009         ADDL2   4(TKEY_DESC), R1            ;
                                              51  D7 0000D         DECL    KEYCODE                     ;
                              6E              71  90 0000F         MOVB    -(KEYCODE), TEMP            ;  1106
                              61        01    A1  90 00012         MOVB    1(KEYCODE), (KEYCODE)       ;  1107
                        01    A1              6E  90 00016         MOVB    TEMP, 1(KEYCODE)            ;  1108
                                        04    AE  9F 0001A         PUSHAB  FOUND_KDE                   ;  1116
                                  0000V       CF  9F 0001D         PUSHAB  COMPARE_ROUTINE             ;
                                        0041  8F  BB 00021         PUSHR   #^M<R0,R6>                  ;
                  00000000G     00            04  FB 00025         CALLS   #4, LIB$LOOKUP_TREE         ;
                                09            50  E9 0002C         BLBC    LOOKUP_STATUS, 1$           ;  1119
                              56        04    AE  D0 0002F         MOVL    FOUND_KDE, KDE              ;  1127
                                        33    A6  95 00033         TSTB    51(KDE)                     ;  1128
                                              09  19 00036         BLSS    2$                          ;
                              50 00000000G    8F  D0 00038 1$:     MOVL    #SMG$_KEYNOTDEF, R0         ;  1130
                                              03  11 0003F         BRB     3$                          ;
                              50              01  D0 00041 2$:     MOVL    #1, R0                      ;  1132
                              5E              08  C0 00044 3$:     ADDL2   #8, SP                      ;  1134
                                                  05 00047         RSB                                 ;

; Routine Size:  72 bytes,    Routine Base:  _SMG$CODE + 0312
```

```
;1066    1135  1 %SBTTL 'CONSTRUCT_TKEY'
;1067    1136  1 ROUTINE CONSTRUCT_TKEY (
;1068    1137  1     CALLERS_AP: REF VECTOR [, LONG],
;1069    1138  1     TKEY_DESC: REF BLOCK [, BYTE];
;1070    1139  1     OUT_TKEY_DESC: REF BLOCK [, BYTE]
;1071    1140  1     ) : CONSTRUCT_TKEY$LNK =
;1072    1141  1
;1073    1142  1 !++
;1074    1143  1 ! FUNCTIONAL DESCRIPTION:
;1075    1144  1 !
;1076    1145  1 !       This procedure builds a "total key" from the IF_STATE and the
;1077    1146  1 !       KEY_NAME in the caller's argument list.
;1078    1147  1 !
;1079    1148  1 ! CALLING SEQUENCE:
;1080    1149  1 !
;1081    1150  1 !       CONSTRUCT_TKEY (CALLERS_AP.rl.v, TKEY_DESC.rr.r;
;1082    1151  1 !               OUT_TKEY_DESC.wa.v)
;1083    1152  1 !
;1084    1153  1 ! FORMAL PARAMETERS:
;1085    1154  1 !
;1086    1155  1 !       CALLERS_AP      - The caller's argument pointer.
;1087    1156  1 !
;1088    1157  1 !       TKEY_DESC       - A descriptor for the TKEY.
;1089    1158  1 !
;1090    1159  1 !       OUT_TKEY_DESC   - A longword into which is written the address
;1091    1160  1 !                         of the TKEY descriptor actually used.
;1092    1161  1 !
;1093    1162  1 ! IMPLICIT INPUTS:
;1094    1163  1 !
;1095    1164  1 !       NONE
;1096    1165  1 !
;1097    1166  1 ! IMPLICIT OUTPUTS:
;1098    1167  1 !
;1099    1168  1 !       NONE
;1100    1169  1 !
;1101    1170  1 ! COMPLETION STATUS:
;1102    1171  1 !
;1103    1172  1 !       SS$_NORMAL      Normal successful completion
;1104    1173  1 !
;1105    1174  1 ! SIDE EFFECTS:
;1106    1175  1 !
;1107    1176  1 !       NONE
;1108    1177  1 !
;1109    1178  1 !--
;1110    1179  1
;1111    1180  2     BEGIN
;1112    1181  2
;1113    1182  2     LOCAL
;1114    1183  2         KEY_CODE,                           ! Key code number
;1115    1184  2         CODE_STATUS,                        ! Status from conversion
;1116    1185  2         NAME_TMP: VECTOR [32, BYTE],        ! Upcased/stripped key name
;1117    1186  2         KTH: REF KTH_R_KTH_STRUCT,          ! Key table header
;1118    1187  2         TKEY_PTR: REF VECTOR [, BYTE];      ! Pointer to current pos in TKEY
;1119    1188  2
;1120    1189  2     !+
;1121    1190  2     ! Define positions of arguments
;1122    1191  2     !-
```

```
;  1123          1192  2
;  1124          1193  2      LITERAL
;  1125          1194  2          K_KEY_TABLE_ID = 1,
;  1126          1195  2          K_KEY_NAME = 2,
;  1127          1196  2          K_IF_STATE = 3;
;  1128          1197  2
;  1129          1198  2      !+
;  1130          1199  2      ! Upcase key name and strip blanks.  If key name has invalid length,
;  1131          1200  2      ! return an error.
;  1132          1201  2      !-
;  1133          1202  2
;  1134          1203  3          BEGIN
;  1135          1204  3          LOCAL
;  1136          1205  3              KEY_NAME_PTR,         ! Pointer to KEY_NAME string
;  1137          1206  3              KEY_NAME_LEN: WORD,   ! Length of key name
;  1138          1207  3              CHAR_PTR,             ! Pointer to current character in KEY_NAME
;  1139          1208  3              ANL_STATUS;           ! Status from LIB$ANALYZE_SDESC_R2
;  1140          1209  3
;  1141          1210  3          ANL_STATUS = LIB$ANALYZE_SDESC_R2 (.CALLERS_AP [K_KEY_NAME];
;  1142          1211  3              KEY_NAME_LEN, KEY_NAME_PTR);
;  1143          1212  3          IF NOT .ANL_STATUS
;  1144          1213  3          THEN
;  1145          1214  3              RETURN .ANL_STATUS;
;  1146          1215  3          IF (.KEY_NAME_LEN - 1) GTRU 30  ! Between 1 and 31?
;  1147          1216  3          THEN
;  1148          1217  3              RETURN SMG$_INVKEYNAM;              ! Invalid key name
;  1149          1218  3
;  1150          1219  3          !+
;  1151          1220  3          ! Find the position of the last non-blank character.
;  1152          1221  3          !-
;  1153          1222  3
;  1154          1223  3          CHAR_PTR = .KEY_NAME_PTR + .KEY_NAME_LEN;
;  1155          1224  3          WHILE ((CHAR_PTR=.CHAR_PTR-1;CH$RCHAR(.CHAR_PTR)) EQLU %C' ') DO
;  1156          1225  4              BEGIN
;  1157          1226  4              KEY_NAME_LEN = .KEY_NAME_LEN - 1;
;  1158          1227  4              IF .KEY_NAME_LEN EQL 0 THEN EXITLOOP;
;  1159          1228  3              END;
;  1160          1229  3
;  1161          1230  3          !+
;  1162          1231  3          ! Move an upcased counted string to NAME_TMP;
;  1163          1232  3          !-
;  1164          1233  3
;  1165          1234  3          NAME_TMP [0] = .KEY_NAME_LEN;
;  1166          1235  3          IF .NAME_TMP [0] EQL 0
;  1167          1236  3          THEN
;  1168          1237  3              RETURN SMG$_INVKEYNAM;      ! Invalid key name
;  1169          1238  3          CH$TRANSLATE (LIB$AB_UPCASE, .KEY_NAME_LEN, .KEY_NAME_PTR,
;  1170          1239  3              0, .KEY_NAME_LEN, NAME_TMP [1]);
;  1171          1240  2          END;
;  1172          1241  2
;  1173          1242  2      !+
;  1174          1243  2      ! Get the code number corresponding to this key name, if any.
;  1175          1244  2      !-
;  1176          1245  2
;  1177          1246  2      CODE_STATUS = SMG$$NAME_TO_KEYCODE (NAME_TMP, KEY_CODE);
;  1178          1247  2      IF NOT .CODE_STATUS
;  1179          1248  2      THEN
```

```
; 1180          1249  2          RETURN .CODE_STATUS;
; 1181          1250  2
; 1182          1251  2      !+
; 1183          1252  2      ! Construct TKEY by appending IF_STATE, a <NUL>, and KEY_CODE.
; 1184          1253  2      ! Swap the bytes of KEY_CODE so that they compare as strings.
; 1185          1254  2      ! Upcase and blank trim IF_STATE while we're at it.
; 1186          1255  2      !-
; 1187          1256  2
; 1188          1257  2      KTH = ..CALLERS_AP [K_KEY_TABLE_ID];
; 1189          1258  2      OUT_TKEY_DESC = KTH [KTH_R_DEF_STATE_DESCR]; ! Assume default state
; 1190          1259  2      TKEY_PTR = .KTH [KTH_A_DEF_KEYCODE];
; 1191          1260  2      IF (.CALLERS_AP[0])<0,8> GEQU K_IF_STATE
; 1192          1261  2      THEN
; 1193          1262  2          IF .CALLERS_AP [K_IF_STATE] NEQA 0
; 1194          1263  2          THEN
; 1195          1264  3              BEGIN
; 1196          1265  3              LOCAL
; 1197          1266  3                  IF_STATE_PTR,           ! Pointer to IF_STATE string
; 1198          1267  3                  IF_STATE_LEN: WORD,     ! Length of IF_STATE string
; 1199          1268  3                  CHAR_PTR,               ! Pointer to current character
; 1200          1269  3                  ANL_STATUS;             ! Status from LIB$ANALYZE_SDESC_R2
; 1201          1270  3
; 1202          1271  3              ANL_STATUS = LIB$ANALYZE_SDESC_R2 (.CALLERS_AP [K_IF_STATE];
; 1203          1272  3                  IF_STATE_LEN, IF_STATE_PTR);
; 1204          1273  3              IF NOT .ANL_STATUS
; 1205          1274  3              THEN
; 1206          1275  3                  RETURN .ANL_STATUS;
; 1207          1276  3              IF (.IF_STATE_LEN - 1) GTRU 30      ! Between 1 and 31?
; 1208          1277  3              THEN
; 1209          1278  3                  RETURN SMG$_INVSTANAM;          ! Invalid state name
; 1210          1279  3
; 1211          1280  3              !+
; 1212          1281  3              ! Find the position of the last non-blank character.
; 1213          1282  3              !-
; 1214          1283  3
; 1215          1284  3              CHAR_PTR = .IF_STATE_PTR+ .IF_STATE_LEN;
; 1216          1285  3              WHILE ((CHAR_PTR=.CHAR_PTR-1;CH$RCHAR(.CHAR_PTR)) EQLU %C' ') DO
; 1217          1286  4                  BEGIN
; 1218          1287  4                  IF_STATE_LEN = .IF_STATE_LEN - 1;
; 1219          1288  4                  IF .IF_STATE_LEN EQL 0 THEN EXITLOOP;
; 1220          1289  3                  END;
; 1221          1290  3
; 1222          1291  3              TKEY_DESC [DSC$W_LENGTH] = .IF_STATE_LEN + 3;
; 1223          1292  3              TKEY_PTR = CH$TRANSLATE (LIB$AB_UPCASE, .IF_STATE_LEN, .IF_STATE_PTR,
; 1224          1293  3                  0, .IF_STATE_LEN, .TKEY_DESC [DSC$A_POINTER]);
; 1225          1294  3              CH$WCHAR_A (0, TKEY_PTR);
; 1226          1295  3
; 1227          1296  3              OUT_TKEY_DESC = .TKEY_DESC;
; 1228          1297  3              OUT_TKEY_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
; 1229          1298  3              OUT_TKEY_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
; 1230          1299  2              END;
; 1231          1300  2
; 1232          1301  2      !+
; 1233          1302  2      ! Move the key code, swapping the bytes.
; 1234          1303  2      !-
; 1235          1304  2
; 1236          1305  2      TKEY_PTR [0]  = .KEY_CODE<8,8>;
```

```
;  1237          1306  2     TKEY_PTR [1] = .KEY_CODE<0,8>;
;  1238          1307  2
;  1239          1308  2     RETURN SS$_NORMAL;
;  1240          1309  2
;  1241          1310  1     END;                                    . End of routine CONSTRUCT_TKEY


                                        0F3C 00000 CONSTRUCT_TKEY:
                                                   .WORD   Save R2,R3,R4,R5,R8,R9,R10,R11                  ; 1136
                              5B 00000000G   00 9E 00002   MOVAB   LIB$ANALYZE_SDESC_R2, R11               ;
                              5A 00000000G   00 9E 00009   MOVAB   LIB$AB_UPCASE, R10                      ;
                              5E             24 C2 00010   SUBL2   #36, SP                               ;
                              59             57 D0 00013   MOVL    R7, R9                                 ;
                              50       08    A6 D0 00016   MOVL    8(CALLERS_AP), R0                      ; 1210
                                             6B 16 0001A   JSB     LIB$ANALYZE_SDESC_R2                   ;
                              6F             50 E9 0001C   BLBC    ANL_STATUS, 5$                         ; 1212
                              50             51 3C 0001F   MOVZWL  KEY_NAME_LEN, R0                       ; 1215
                                             50 D7 00022   DECL    R0                                     ;
                              1E             50 D1 00024   CMPL    R0, #30                                ;
                                             15 1A 00027   BGTRU   3$                                     ;
                              50             51 3C 00029   MOVZWL  KEY_NAME_LEN, CHAR_PTR                 ; 1223
                              50             52 C0 0002C   ADDL2   KEY_NAME_PTR, CHAR_PTR                 ;
                              20             70 91 0002F 1$:  CMPB   -(CHAR_PTR), #32                     ; 1224
                                             04 12 00032   BNEQ    2$                                     ;
                                             51 B7 00034   DECW    KEY_NAME_LEN                           ; 1226
                                             F7 12 00036   BNEQ    1$                                     ; 1227
                                 04    AE      51 90 00038 2$:  MOVB    KEY_NAME_LEN, NAME_TMP               ; 1234
                                             09 12 0003C   BNEQ    4$                                     ; 1235
                              50 00000000G   8F D0 0003E 3$:  MOVL    #SMG$_INVKEYNAM, R0                  ; 1237
                                             7A 11 00045   BRB     10$                                    ;
        6A          00                 62   51 2E 00047 4$:  MOVTC   KEY_NAME_LEN, (KEY_NAME_PTR), #0, - ; 1239
                                 05    AE      51    0004C         LIB$AB_UPCASE, KEY_NAME_LEN, NAME_TMP+1  ;
                                             5E DD 0004F   PUSHL   SP                                     ; 1246
                                       08    AE 9F 00051   PUSHAB  NAME_TMP                               ;
                          00000000G    00      02 FB 00054   CALLS   #2, SMG$$NAME_TO_KEYCODE             ;
                                       63      50 E9 0005B   BLBC    CODE_STATUS, 10$                     ; 1247
                                       57  04  B6 D0 0005E   MOVL    @4(CALLERS_AP), KTH                  ; 1257
                                       58  14  A7 9E 00062   MOVAB   20(R7), OUT_TKEY_DESC               ; 1258
                                       55  1C  A7 D0 00066   MOVL    28(KTH), TKEY_PTR                    ; 1259
                                       03      66 D1 0006A   CMPL    (CALLERS_AP), #3                     ; 1260
                                               47 1F 0006D   BLSSU   9$                                   ;
                                           0C  A6 D5 0006F   TSTL    12(CALLERS_AP)                       ; 1262
                                               42 13 00072   BEQL    9$                                   ;
                                       50  0C  A6 D0 00074   MOVL    12(CALLERS_AP), R0                   ; 1271
                                               6B 16 00078   JSB     LIB$ANALYZE_SDESC_R2                 ;
                                       44      50 E9 0007A   BLBC    ANL_STATUS, 10$                      ; 1273
                                       50      51 3C 0007D   MOVZWL  IF_STATE_LEN, R0                     ; 1276
                                               50 D7 00080   DECL    R0                                   ;
                                       1E      50 D1 00082   CMPL    R0, #30                              ;
                                               09 1B 00085   BLEQU   6$                                   ;
                                       50 00000000G   8F D0 00087   MOVL    #SMG$_INVSTANAM, R0            ; 1278
                                               31 11 0008E 5$:  BRB     10$                               ;
                                       50      51 3C 00090 6$:  MOVZWL  IF_STATE_LEN, CHAR_PTR             ; 1284
                                       50      52 C0 00093   ADDL2   IF_STATE_PTR, CHAR_PTR               ;
```

```
                          20          70  91 00096 7$:     CMPB    -(CHAR_PTR), #32                                ; 1285
                                      04  12 00099         BNEQ    8$                                              ;
                                      51  B7 0009B         DECW    IF_STATE_LEN                                    ; 1287
                                      F7  12 0009D         BNEQ    7$                                              ; 1288
                      69              51  03 A1 0009F 8$:  ADDW3   #3, IF_STATE_LEN, (TKEY_DESC)                  ; 1291
                6A    00              62  51 2E 000A3     MOVTC   IF_STATE_LEN, (IF_STATE_PTR), #0, -             ; 1293
                                  04  B9  51     000A8                 LIB$AB_UPCASE, IF_STATE_LEN, @4(TKEY_DESC)  ;
                                      58  85 94 000AB     CLRB    (TKEY_PTR)+                                     ; 1294
                                      59  D0 000AD         MOVL    TKEY_DESC, OUT_TKEY_DESC                        ; 1296
                                  02  A8  010E  8F  B0 000B0     MOVW    #270, 2(OUT_TKEY_DESC)                  ; 1297
                                      65    01  AE  90 000B6 9$:  MOVB    KEY_CODE+1, (TKEY_PTR)                  ; 1305
                                  01  A5  6E  90 000BA         MOVB    KEY_CODE, 1(TKEY_PTR)                   ; 1306
                                      50  01  D0 000BE         MOVL    #1, R0                                  ; 1308
                                      57  58  D0 000C1 10$:     MOVL    R8, R7                                  ; 1310
                                          04 000C4             RET                                             ;

; Routine Size:  197 bytes,    Routine Base:  _SMG$CODE + 035A
```

```
;  1243          1311  1 %SBTTL 'COMPARE_ROUTINE'
;  1244          1312  1 ROUTINE COMPARE_ROUTINE (
;  1245          1313  1     KEY_DESC: REF BLOCK [, BYTE],               ! Key being compared
;  1246          1314  1     KDE: REF KDE_R_KDE_STRUCT,                  ! Key definition being compared
;  1247          1315  1     CONTEXT                                     ! Not used
;  1248          1316  1     ) =
;  1249          1317  1
;  1250          1318  1 !++
;  1251          1319  1 ! FUNCTIONAL DESCRIPTION:
;  1252          1320  1 !
;  1253          1321  1 !       Routine called by LIB$INSERT_TREE and LIB$LOOKUP_TREE to compare
;  1254          1322  1 !       keys.
;  1255          1323  1 !
;  1256          1324  1 ! CALLING SEQUENCE:
;  1257          1325  1 !
;  1258          1326  1 !       Not directly called.
;  1259          1327  1 !
;  1260          1328  1 ! FORMAL PARAMETERS:
;  1261          1329  1 !
;  1262          1330  1 !
;  1263          1331  1 ! IMPLICIT INPUTS:
;  1264          1332  1 !
;  1265          1333  1 !       NONE
;  1266          1334  1 !
;  1267          1335  1 ! IMPLICIT OUTPUTS:
;  1268          1336  1 !
;  1269          1337  1 !       NONE
;  1270          1338  1 !
;  1271          1339  1 ! COMPLETION STATUS:
;  1272          1340  1 !
;  1273          1341  1 !       SS$_NORMAL      Normal successful completion
;  1274          1342  1 !
;  1275          1343  1 ! SIDE EFFECTS:
;  1276          1344  1 !
;  1277          1345  1 !       NONE
;  1278          1346  1 !
;  1279          1347  1 !--
;  1280          1348  1
;  1281          1349  2     BEGIN
;  1282          1350  2
;  1283          1351  2     RETURN CH$COMPARE (
;  1284          1352  2         .KEY_DESC [DSC$W_LENGTH],
;  1285          1353  2         .KEY_DESC [DSC$A_POINTER],
;  1286          1354  2         .KDE [KDE_W_TKEY_LENGTH],
;  1287          1355  2         .KDE [KDE_A_TKEY_POINTER]);
;  1288          1356  2
;  1289          1357  1     END;                                        ! End of routine COMPARE_ROUTINE


                                001C 00000 COMPARE_ROUTINE:
                                           .WORD   Save R2,R3,R4                    ; 1312
                     51    04   AC  D0 00002          MOVL    KEY_DESC, R1         ; 1352
                     50    08   AC  D0 00006          MOVL    KDE, R0              ; 1354
                     54         01  D0 0000A          MOVL    #1, R4               ; 1351
```

```
      0C   A0            00       04   B1            61  2D 0000D          CMPC5   (R1), @4(R1), #0, 12(R0), @16(R0)      ;
                                                10  B0      00014                                                          ;
                                                    03  1A 00016          BGTRU   1$                                     ;
                                           54       01  D9 00018          SBWC    #1, R4                                 ;
                                           50       54  D0 0001B 1$:      MOVL    R4, R0                                 ;
                                                    04  0001E            RET                                            ; 1357

; Routine Size:  31 bytes,    Routine Base:  _SMG$CODE + 041F
```

```
;  1291     1358  1 %SBTTL 'ALLOC_ROUTINE'
;  1292     1359  1 ROUTINE ALLOC_ROUTINE (
;  1293     1360  1     TKEY_DESC: REF BLOCK [, BYTE],          ! Key descriptor
;  1294     1361  1     KDE_POINTER: REF VECTOR [, LONG],       ! Pointer to new KDE
;  1295     1362  1     CONTEXT                                 ! Not used
;  1296     1363  1     ) =
;  1297     1364  1
;  1298     1365  1 !++
;  1299     1366  1 ! FUNCTIONAL DESCRIPTION:
;  1300     1367  1 !
;  1301     1368  1 !       Routine called by LIB$INSERT_TREE to allocate a node.
;  1302     1369  1 !
;  1303     1370  1 ! CALLING SEQUENCE:
;  1304     1371  1 !
;  1305     1372  1 !       Not directly called.
;  1306     1373  1 !
;  1307     1374  1 ! FORMAL PARAMETERS:
;  1308     1375  1 !
;  1309     1376  1 !
;  1310     1377  1 ! IMPLICIT INPUTS:
;  1311     1378  1 !
;  1312     1379  1 !       NONE
;  1313     1380  1 !
;  1314     1381  1 ! IMPLICIT OUTPUTS:
;  1315     1382  1 !
;  1316     1383  1 !       NONE
;  1317     1384  1 !
;  1318     1385  1 ! COMPLETION STATUS:                                                                    ;
;  1319     1386  1 !
;  1320     1387  1 !       SS$_NORMAL      Normal successful completion
;  1321     1388  1 !
;  1322     1389  1 ! SIDE EFFECTS:
;  1323     1390  1 !
;  1324     1391  1 !       NONE
;  1325     1392  1 !
;  1326     1393  1 !--
;  1327     1394  1
;  1328     1395  2     BEGIN
;  1329     1396  2
;  1330     1397  2     LOCAL
;  1331     1398  2         VM_STATUS,                          ! Status from LIB$GET_VM
;  1332     1399  2         KDE: REF KDE_R_KDE_STRUCT,          ! Key Definition Entry
;  1333     1400  2         KEY_CODE: WORD;
;  1334     1401  2
;  1335     1402  2     !+
;  1336     1403  2     ! Allocate the memory.
;  1337     1404  2     !-
;  1338     1405  2
;  1339     1406  2     VM_STATUS = LIB$GET_VM (%REF(KDE_S_KDE_STRUCT), KDE_POINTER [0]);
;  1340     1407  2     IF NOT .VM_STATUS
;  1341     1408  2     THEN
;  1342     1409  2         RETURN .VM_STATUS;
;  1343     1410  2
;  1344     1411  2     !+
;  1345     1412  2     ! Initialize the KDE.
;  1346     1413  2     !-
;  1347     1414  2
```

```
;  1348     1415  2      KDE = .KDE_POINTER [0];
;  1349     1416  2      CH$FILL (0, KDE_S_KDE_STRUCT, KDE [0,0,0,0]);
;  1350     1417  2      KDE [KDE_B_TKEY_CLASS] = DSC$K_CLASS_D;
;  1351     1418  2      KDE [KDE_B_TKEY_DTYPE] = DSC$K_DTYPE_T;
;  1352     1419  2      KDE [KDE_B_EQUIV_CLASS] = DSC$K_CLASS_D;
;  1353     1420  2      KDE [KDE_B_EQUIV_DTYPE] = DSC$K_DTYPE_T;
;  1354     1421  2      KDE [KDE_B_STATE_CLASS] = DSC$K_CLASS_D;
;  1355     1422  2      KDE [KDE_B_STATE_DTYPE] = DSC$K_DTYPE_T;
;  1356     1423  2      KDE [KDE_B_PREFIX_CLASS] = DSC$K_CLASS_D;
;  1357     1424  2      KDE [KDE_B_PREFIX_DTYPE] = DSC$K_DTYPE_T;
;  1358     1425  2
;  1359     1426  2      !+
;  1360     1427  2      ! If the key being defined is CTRLx, set the terminator mask to include
;  1361     1428  2      ! this control key.
;  1362     1429  2      !-
;  1363     1430  2
;  1364     1431  2
;  1365     1432  4      KEY_CODE = .((.TKEY_DESC [DSC$A_POINTER] +
;  1366     1433  2          .TKEY_DESC [DSC$W_LENGTH]) - 2)<0,16>;
;  1367     1434  2      IF .KEY_CODE NEQ 0 AND
;  1368     1435  2         .KEY_CODE LEQU SMG$K_TRM_CTRLZ
;  1369     1436  2      THEN
;  1370     1437  2          KDE [KDE_L_TERM_MASK] = 1^.KEY_CODE;
;  1371     1438  2
;  1372     1439  2      RETURN SS$_NORMAL;
;  1373     1440  2
;  1374     1441  1      END;                                    ! End of routine ALLOC_ROUTINE


                                                007C 00000 ALLOC_ROUTINE:
                                                           .WORD   Save R2,R3,R4,R5,R6                 ;  1359
                                           5E           04  C2 00002          SUBL2   #4, SP
                                                    08     AC  DD 00005          PUSHL   KDE_POINTER                 ;  1406
                                               04   AE     38  D0 00008          MOVL    #56, 4(SP)
                                                    04     AE  9F 0000C          PUSHAB  4(SP)
                                   00000000G   00          02  FB 0000F          CALLS   #2, LIB$GET_VM
                                               41          50  E9 00016          BLBC    VM_STATUS, 2$               ;  1407
                                               56     08   BC  D0 00019          MOVL    @KDE_POINTER, KDE           ;  1415
                38          00               6E          00  2C 0001D          MOVC5   #0, (SP), #0, #56, (KDE)    ;  1416
                                                           66     00022
                                           0E  A6   020E   8F  B0 00023          MOVW    #526, 14(KDE)               ;  1418
                                           16  A6   020E   8F  B0 00029          MOVW    #526, 22(KDE)               ;  1420
                                           1E  A6   020E   8F  B0 0002F          MOVW    #526, 30(KDE)               ;  1422
                                           26  A6   020E   8F  B0 00035          MOVW    #526, 38(KDE)               ;  1424
                                               50     04   AC  D0 0003B          MOVL    TKEY_DESC, R0               ;  1432
                                               52          60  3C 0003F          MOVZWL  (R0), R2                    ;  1433
                                     50        52     04   A0  C1 00042          ADDL3   4(R0), R2, R0
                                               51     FE   A0  B0 00047          MOVW    -2(R0), KEY_CODE            ;  1432
                                                           0A  13 0004B          BEQL    1$                          ;  1434
                                               1A          51  B1 0004D          CMPW    KEY_CODE, #26               ;  1435
                                                           05  1A 00050          BGTRU   1$
                                34  A6         01          51  78 00052          ASHL    KEY_CODE, #1, 52(KDE)       ;  1437
                                               50          01  D0 00057 1$:      MOVL    #1, R0                      ;  1439
                                                           04  0005A 2$:      RET                                 ;  1441
```

; Routine Size: 91 bytes,   Routine Base: _SMG$CODE + 043E

```
;1376     1442  1 %SBTTL 'SMG$DEFINE_KEY'
;1377     1443  1 GLOBAL ROUTINE SMG$DEFINE_KEY (
;1378     1444  1     KEY_TABLE_ID: REF VECTOR [, LONG],
;1379     1445  1     COMMAND_LINE: REF BLOCK [, BYTE]
;1380     1446  1     ) =
;1381     1447  1
;1382     1448  1 !++
;1383     1449  1 ! FUNCTIONAL DESCRIPTION:
;1384     1450  1 !
;1385     1451  1 !       This procedure parses and performs a DEFINE/KEY command.  It can
;1386     1452  1 !       be used by utilities which accept DEFINE/KEY commands but which
;1387     1453  1 !       do not wish to parse the commands themselves.
;1388     1454  1 !
;1389     1455  1 !       SMG$DEFINE_KEY calls CLI$DCL_PARSE to parse the command line and
;1390     1456  1 !       then makes the appropriate call to SMG$ADD_KEY_DEF.  Use of this
;1391     1457  1 !       procedure requires that the image be run under the DCL command
;1392     1458  1 !       language interpreter.
;1393     1459  1 !
;1394     1460  1 ! CALLING SEQUENCE:
;1395     1461  1 !
;1396     1462  1 !       RET_STATUS.wlc.v = SMG$DEFINE_KEY (
;1397     1463  1 !                                  KEY_TABLE_ID.rl.r,
;1398     1464  1 !                                  COMMAND_LINE.rt.dx)
;1399     1465  1 !
;1400     1466  1 ! FORMAL PARAMETERS:
;1401     1467  1 !
;1402     1468  1 !       KEY_TABLE_ID    A longword containing the identification of the
;1403     1469  1 !                       key-definition table for which the DEFINE/KEY
;1404     1470  1 !                       command is to be performed.
;1405     1471  1 !
;1406     1472  1 !       COMMAND_LINE    A string containing the entire DEFINE/KEY command
;1407     1473  1 !                       to be performed.  Obtain this by calling
;1408     1474  1 !                       CLI$GET_VALUE with the '$LINE' keyword.
;1409     1475  1 !
;1410     1476  1 ! IMPLICIT INPUTS:
;1411     1477  1 !
;1412     1478  1 !       NONE
;1413     1479  1 !
;1414     1480  1 ! IMPLICIT OUTPUTS:
;1415     1481  1 !
;1416     1482  1 !       NONE
;1417     1483  1 !
;1418     1484  1 ! COMPLETION STATUS:
;1419     1485  1 !
;1420     1486  1 !       SS$_NORMAL      Normal successful completion
;1421     1487  1 !       LIB$_xxx        Any error status from LIB$SCOPY_DXDX
;1422     1488  1 !       CLI$_xxx        Any error status returned from CLI$ routines
;1423     1489  1 !       SMG$_xxx        Any error status returned by SMG$ADD_KEY_DEF
;1424     1490  1 !
;1425     1491  1 ! SIDE EFFECTS:
;1426     1492  1 !
;1427     1493  1 !       NONE
;1428     1494  1 !
;1429     1495  1 !--
;1430     1496  1
;1431     1497  2     BEGIN
;1432     1498  2
```

```
;  1433          1499  2     LOCAL
;  1434          1500  2         DESC_LIST: BLOCKVECTOR [5, 8, BYTE],
;  1435          1501  2                                        ! List of descriptors for
;  1436          1502  2                                        ! items returned from CLI
;  1437          1503  2         ATTRIBUTES,                    ! Attributes bit mask
;  1438          1504  2         SET_STATE_PTR,                 ! Pointer to SET_STATE desc, if any
;  1439          1505  2         IF_STATE_PTR,                  ! Pointer to IF_STATE desc, if any
;  1440          1506  2         DESC_LIST_PTR: VOLATILE,       ! Pointer to DESC_LIST
;  1441          1507  2         IF_STATE_PRESENT;              ! 1 if IF_STATE was specified
;  1442          1508  2
;  1443          1509  2     BUILTIN
;  1444          1510  2         NULLPARAMETER;
;  1445          1511  2
;  1446          1512  2     !+
;  1447          1513  2     ! Define codes that index the descriptors.
;  1448          1514  2     !-
;  1449          1515  2
;  1450          1516  2     LITERAL
;  1451          1517  2         K_KEY_NAME = 0,            ! Key name (P1)
;  1452          1518  2         K_EQUIV_STRING = 1,        ! Equivalence string (P2)
;  1453          1519  2         K_IF_STATE = 2,            ! /IF_STATE
;  1454          1520  2         K_SET_STATE = 3,           ! /SET_STATE
;  1455          1521  2         K_ITEM_NAME = 4;           ! Name of command line item
;  1456          1522  2
;  1457          1523  2     !+
;  1458          1524  2     ! Enable the error handler.
;  1459          1525  2     !-
;  1460          1526  2
;  1461          1527  2     ENABLE DEFINE_KEY_HANDLER (DESC_LIST_PTR);
;  1462          1528  2
;  1463          1529  2     !+
;  1464          1530  2     ! Initialize the descriptors.
;  1465          1531  2     !-
;  1466          1532  2
;  1467          1533  3         BEGIN
;  1468          1534  3         LOCAL
;  1469          1535  3             PTR1: REF VECTOR [, LONG],
;  1470          1536  3             PTR2: REF VECTOR [, LONG];
;  1471          1537  3
;  1472          1538  3         DESC_LIST [K_KEY_NAME, DSC$W_LENGTH] = 0;
;  1473          1539  3         DESC_LIST [K_KEY_NAME, DSC$B_DTYPE] = DSC$K_DTYPE_T;
;  1474          1540  3         DESC_LIST [K_KEY_NAME, DSC$B_CLASS] = DSC$K_CLASS_D;
;  1475          1541  3         DESC_LIST [K_KEY_NAME, DSC$A_POINTER] = 0;
;  1476          1542  3
;  1477          1543  3         PTR1 = DESC_LIST [K_KEY_NAME, 0,0,0,0];
;  1478          1544  3         PTR2 = DESC_LIST [K_EQUIV_STRING, 0,0,0,0];
;  1479          1545  3         $LIB$COPY_QUAD_A (PTR1, PTR2);   ! Initialize EQUIV_STRING desc
;  1480          1546  3         $LIB$COPY_QUAD_A (PTR1, PTR2);   ! Initialize IF_STATE desc
;  1481          1547  3         $LIB$COPY_QUAD_A (PTR1, PTR2);   ! Initialize SET_STATE desc
;  1482          1548  3         $LIB$COPY_QUAD_A (PTR1, PTR2);   ! Initialize ITEM_NAME desc
;  1483          1549  3         DESC_LIST [K_ITEM_NAME, DSC$B_CLASS] = DSC$K_CLASS_S;
;  1484          1550  2         END;
;  1485          1551  2
;  1486          1552  2     !+
;  1487          1553  2     ! Set the enable argument pointer.
;  1488          1554  2     !-
;  1489          1555  2
```

```
;1490     1556  2      DESC_LIST_PTR = DESC_LIST;
;1491     1557  2
;1492     1558  2      !+
;1493     1559  2      ! Parse the command line
;1494     1560  2      !-
;1495     1561  2
;1496     1562  3          BEGIN
;1497     1563  3          LOCAL
;1498     1564  3              CLI_STATUS;
;1499     1565  3          CLI_STATUS = CLI$DCL_PARSE (COMMAND_LINE [0,0,0,0],
;1500     1566  3              SMG$$AB_DEFKEY_CLD);
;1501     1567  3          IF NOT .CLI_STATUS
;1502     1568  3          THEN
;1503     1569  3              RETURN LIB$SIGNAL (.CLI_STATUS);
;1504     1570  2          END;
;1505     1571  2
;1506     1572  2      !+
;1507     1573  2      ! Get the key name.
;1508     1574  2      !-
;1509     1575  2
;1510     1576  3          BEGIN
;1511     1577  3          LOCAL
;1512     1578  3              STATUS;
;1513     1579  3          DESC_LIST [K_ITEM_NAME, DSC$W_LENGTH] = %CHARCOUNT ('P1');
;1514     1580  3          DESC_LIST [K_ITEM_NAME, DSC$A_POINTER] = UPLIT BYTE ('P1');
;1515     1581  3
;1516     1582  3          STATUS = CLI$GET_VALUE (DESC_LIST [K_ITEM_NAME, 0,0,0,0],
;1517     1583  3              DESC_LIST [K_KEY_NAME, 0,0,0,0]);
;1518     1584  3          IF NOT .STATUS
;1519     1585  3          THEN
;1520     1586  3              RETURN LIB$SIGNAL (.STATUS);
;1521     1587  2          END;
;1522     1588  2
;1523     1589  2      !+
;1524     1590  2      ! Get the equivalence string.
;1525     1591  2      !-
;1526     1592  2
;1527     1593  3          BEGIN
;1528     1594  3          LOCAL
;1529     1595  3              STATUS;
;1530     1596  3          DESC_LIST [K_ITEM_NAME, DSC$W_LENGTH] = %CHARCOUNT ('P2');
;1531     1597  3          DESC_LIST [K_ITEM_NAME, DSC$A_POINTER] = UPLIT BYTE ('P2');
;1532     1598  3
;1533     1599  3          STATUS = CLI$GET_VALUE (DESC_LIST [K_ITEM_NAME, 0,0,0,0],
;1534     1600  3              DESC_LIST [K_EQUIV_STRING, 0,0,0,0]);
;1535     1601  3          IF NOT .STATUS
;1536     1602  3          THEN
;1537     1603  3              RETURN LIB$SIGNAL (.STATUS);
;1538     1604  2          END;
;1539     1605  2
;1540     1606  2      !+
;1541     1607  2      ! Get the new state name, if any.
;1542     1608  2      !-
;1543     1609  2
;1544     1610  2      DESC_LIST [K_ITEM_NAME, DSC$W_LENGTH] = %CHARCOUNT ('SET_STATE');
;1545     1611  2      DESC_LIST [K_ITEM_NAME, DSC$A_POINTER] = UPLIT BYTE ('SET_STATE');
;1546     1612  2      SET_STATE_PTR = 0;
```

```
;  1547          1613  2      IF CLI$PRESENT (DESC_LIST [K_ITEM_NAME, 0,0,0,0])
;  1548          1614  2      THEN
;  1549          1615  3          BEGIN
;  1550          1616  3          LOCAL
;  1551          1617  3              STATUS;
;  1552          1618  3          STATUS = CLI$GET_VALUE (DESC_LIST [K_ITEM_NAME, 0,0,0,0],
;  1553          1619  3              DESC_LIST [K_SET_STATE, 0,0,0,0]);
;  1554          1620  3          IF NOT .STATUS
;  1555          1621  3          THEN
;  1556          1622  3              RETURN LIB$SIGNAL (.STATUS);
;  1557          1623  3          SET_STATE_PTR = DESC_LIST [K_SET_STATE, 0,0,0,0];
;  1558          1624  2          END;
;  1559          1625  2
;  1560          1626  2      !+
;  1561          1627  2      ! Get the attributes.
;  1562          1628  2      !-
;  1563          1629  2
;  1564          1630  2      ATTRIBUTES = 0;
;  1565          1631  2      DESC_LIST [K_ITEM_NAME, DSC$W_LENGTH] = %CHARCOUNT ('ECHO');
;  1566          1632  2      DESC_LIST [K_ITEM_NAME, DSC$A_POINTER] = UPLIT BYTE ('ECHO');
;  1567          1633  2      IF NOT CLI$PRESENT (DESC_LIST [K_ITEM_NAME, 0,0,0,0])
;  1568          1634  2      THEN
;  1569          1635  2          ATTRIBUTES = .ATTRIBUTES OR SMG$M_KEY_NOECHO;
;  1570          1636  2
;  1571          1637  2      DESC_LIST [K_ITEM_NAME, DSC$W_LENGTH] = %CHARCOUNT ('TERMINATE');
;  1572          1638  2      DESC_LIST [K_ITEM_NAME, DSC$A_POINTER] = UPLIT BYTE ('TERMINATE');
;  1573          1639  2      IF CLI$PRESENT (DESC_LIST [K_ITEM_NAME, 0,0,0,0])
;  1574          1640  2      THEN
;  1575          1641  2          ATTRIBUTES = .ATTRIBUTES OR SMG$M_KEY_TERMINATE;
;  1576          1642  2
;  1577          1643  2      DESC_LIST [K_ITEM_NAME, DSC$W_LENGTH] = %CHARCOUNT ('LOCK_STATE');
;  1578          1644  2      DESC_LIST [K_ITEM_NAME, DSC$A_POINTER] = UPLIT BYTE ('LOCK_STATE');
;  1579          1645  2      IF CLI$PRESENT (DESC_LIST [K_ITEM_NAME, 0,0,0,0])
;  1580          1646  2      THEN
;  1581          1647  2          ATTRIBUTES = .ATTRIBUTES OR SMG$M_KEY_LOCK;
;  1582          1648  2
;  1583          1649  2      !+
;  1584          1650  2      ! See if IF_STATE was specified, and set IF_STATE_PRESENT accordingly.
;  1585          1651  2      !-
;  1586          1652  2
;  1587          1653  2      DESC_LIST [K_ITEM_NAME, DSC$W_LENGTH] = %CHARCOUNT ('IF_STATE');
;  1588          1654  2      DESC_LIST [K_ITEM_NAME, DSC$A_POINTER] = UPLIT BYTE ('IF_STATE');
;  1589          1655  2
;  1590          1656  2      IF_STATE_PTR = 0;
;  1591          1657  2      IF_STATE_PRESENT = CLI$PRESENT (DESC_LIST [K_ITEM_NAME, 0,0,0,0]);
;  1592          1658  2      IF .IF_STATE_PRESENT
;  1593          1659  2      THEN
;  1594          1660  2          IF_STATE_PTR = DESC_LIST [K_IF_STATE, 0,0,0,0];
;  1595          1661  2
;  1596          1662  2      !+
;  1597          1663  2      ! Loop adding key definitions until all IF_STATE values are used, or
;  1598          1664  2      ! if no IF_STATE, just once.
;  1599          1665  2      !-
;  1600          1666  2
;  1601          1667  2      WHILE 1 DO
;  1602          1668  3          BEGIN
;  1603          1669  3
```

```
;  1604     1670  3          !+
;  1605     1671  3          ! If IF_STATE was on the command line, get the next value.
;  1606     1672  3          !-
;  1607     1673  3
;  1608     1674  3          IF .IF_STATE_PRESENT
;  1609     1675  3          THEN
;  1610     1676  4              BEGIN
;  1611     1677  4              LOCAL
;  1612     1678  4                  STATUS;
;  1613     1679  4              STATUS = CLI$GET_VALUE (DESC_LIST [K_ITEM_NAME, 0,0,0,0],
;  1614     1680  4                  DESC_LIST [K_IF_STATE, 0,0,0,0]);
;  1615     1681  4              IF NOT .STATUS      ! Exit if no more values
;  1616     1682  4              THEN
;  1617     1683  4                  EXITLOOP;
;  1618     1684  3              END;
;  1619     1685  3
;  1620     1686  3          !+
;  1621     1687  3          ! Add the key definition.
;  1622     1688  3          !-
;  1623     1689  3
;  1624     1690  4              BEGIN
;  1625     1691  4              LOCAL
;  1626     1692  4                  ADD_STATUS;
;  1627     1693  4              ADD_STATUS = SMG$ADD_KEY_DEF (
;  1628     1694  4                  KEY_TABLE_ID [0],
;  1629     1695  4                  DESC_LIST [K_KEY_NAME, 0,0,0,0],
;  1630     1696  4                  .IF_STATE_PTR,
;  1631     1697  4                  ATTRIBUTES,
;  1632     1698  4                  DESC_LIST [K_EQUIV_STRING, 0,0,0,0],
;  1633     1699  4                  .SET_STATE_PTR);
;  1634     1700  4              IF NOT .ADD_STATUS
;  1635     1701  4              THEN
;  1636     1702  4                  RETURN LIB$SIGNAL (.ADD_STATUS);
;  1637     1703  3              END;
;  1638     1704  3
;  1639     1705  3          !+
;  1640     1706  3          ! If IF_STATE wasn't on the command line, exit.
;  1641     1707  3          !-
;  1642     1708  3
;  1643     1709  3          IF NOT .IF_STATE_PRESENT
;  1644     1710  3          THEN
;  1645     1711  3              EXITLOOP;
;  1646     1712  2          END;
;  1647     1713  2
;  1648     1714  2      !+
;  1649     1715  2      ! Free our dynamic strings.
;  1650     1716  2      !-
;  1651     1717  2
;  1652     1718  2      RETURN LIB$SFREEN_DD (%REF(4), DESC_LIST);
;  1653     1719  2
;  1654     1720  2
;  1655     1721  1      END;                                ! End of routine SMG$DEFINE_KEY


                                        31  50  00499 P.AAB:  .ASCII  \P1\
                                        32  50  0049B P.AAC:  .ASCII  \P2\
```

```
                45  54  41  54  53  5F  54  45  53  0049D P.AAD:  .ASCII  \SET_STATE\                    ;
                                    4F  48  43  45  004A6 P.AAE:  .ASCII  \ECHO\                        ;
                45  54  41  4E  49  4D  52  45  54  004AA P.AAF:  .ASCII  \TERMINATE\                    ;
            45  54  41  54  53  5F  4B  43  4F  4C  004B3 P.AAG:  .ASCII  \LOCK_STATE\                   ;
                45  54  41  54  53  5F  46  49      004BD P.AAH:  .ASCII  \IF_STATE\                    ;


                                        01FC 00000          .ENTRY  SMG$DEFINE_KEY, Save R2,R3,R4,R5,R6,R7,R8 ; 1443
                    58 00000000G    00   9E 00002          MOVAB   CLI$GET_VALUE, R8                         ;
                    57         C8   AF   9E 00009          MOVAB   P.AAB, R7                                 ;
                    56 00000000G    00   9E 0000D          MOVAB   CLI$PRESENT, R6                           ;
                    5E              34   C2 00014          SUBL2   #52, SP                                   ;
                            08      AE   D4 00017          CLRL    DESC_LIST_PTR                             ; 1497
                    6D      0135    CF   DE 0001A          MOVAL   13$, (FP)                                 ;
                 0C AE  020E0000    8F   D0 0001F          MOVL    #34471936, DESC_LIST                        1538
                            10      AE   D4 00027          CLRL    DESC_LIST+4                               ; 1541
                    51      0C      AE   9E 0002A          MOVAB   DESC_LIST, PTR1                           ; 1543
                    50      14      AE   9E 0002E          MOVAB   DESC_LIST+8, PTR2                         ; 1544
                    80              81   7D 00032          MOVQ    (PTR1)+, (PTR2)+                          ; 1545
                    80              81   7D 00035          MOVQ    (PTR1)+, (PTR2)+                          ; 1546
                    80              81   7D 00038          MOVQ    (PTR1)+, (PTR2)+                          ; 1547
                    80              81   7D 0003B          MOVQ    (PTR1)+, (PTR2)+                          ; 1548
                 2F AE              01   90 0003E          MOVB    #1, DESC_LIST+35                          ; 1549
                 08 AE      0C      AE   9E 00042          MOVAB   DESC_LIST, DESC_LIST_PTR                  ; 1556
                        00000000G   00   9F 00047          PUSHAB  SMG$$AB_DEFKEY_CLD                        ; 1565
                            08      AC   DD 0004D          PUSHL   COMMAND_LINE                              ;
                00000000G   00              02   FB 00050  CALLS   #2, CLI$DCL_PARSE                         ;
                            46              50   E9 00057  BLBC    CLI_STATUS, 1$                            ; 1567
                 2C AE                      02   B0 0005A  MOVW    #2, DESC_LIST+32                          ; 1579
                 30 AE                      67   9E 0005E  MOVAB   P.AAB, DESC_LIST+36                      ; 1580
                            0C      AE      9F 00062       PUSHAB  DESC_LIST                                 ; 1583
                            30      AE      9F 00065       PUSHAB  DESC_LIST+32                              ; 1582
                    68                      02   FB 00068  CALLS   #2, CLI$GET_VALUE                        ; 1583
                    32                      50   E9 0006B  BLBC    STATUS, 1$                                ; 1584
                 2C AE                      02   B0 0006E  MOVW    #2, DESC_LIST+32                          ; 1596
                 30 AE          02          A7   9E 00072  MOVAB   P.AAC, DESC_LIST+36                      ; 1597
                            14      AE      9F 00077       PUSHAB  DESC_LIST+8                               ; 1600
                            30      AE      9F 0007A       PUSHAB  DESC_LIST+32                              ; 1599
                    68                      02   FB 0007D  CALLS   #2, CLI$GET_VALUE                        ; 1600
                    1D                      50   E9 00080  BLBC    STATUS, 1$                                ; 1601
                 2C AE                      09   B0 00083  MOVW    #9, DESC_LIST+32                          ; 1610
                 30 AE          04          A7   9E 00087  MOVAB   P.AAD, DESC_LIST+36                      ; 1611
                                            55   D4 0008C  CLRL    SET_STATE_PTR                             ; 1612
                            2C      AE      9F 0008E       PUSHAB  DESC_LIST+32                              ; 1613
                    66                      01   FB 00091  CALLS   #1, CLI$PRESENT                          ;
                    13                      50   E9 00094  BLBC    R0, 3$                                    ;
                            24      AE      9F 00097       PUSHAB  DESC_LIST+24                              ; 1619
                            30      AE      9F 0009A       PUSHAB  DESC_LIST+32                              ; 1618
                    68                      02   FB 0009D  CALLS   #2, CLI$GET_VALUE                        ; 1619
                    03                      50   E8 000A0  1$:    BLBS    STATUS, 2$                        ; 1620
                                        008E    31   000A3          BRW     10$                               ;
                    55          24      AE      9E 000A6  2$:    MOVAB   DESC_LIST+24, SET_STATE_PTR       ; 1623
                                04      AE      D4 000AA  3$:    CLRL    ATTRIBUTES                        ; 1630
                 2C AE                      04   B0 000AD          MOVW    #4, DESC_LIST+32                  ; 1631
                 30 AE          0D          A7   9E 000B1          MOVAB   P.AAE, DESC_LIST+36               ; 1632
                                2C      AE      9F 000B6          PUSHAB  DESC_LIST+32                      ; 1633
```

```
                66              01 FB 000B9          CALLS   #1, CLI$PRESENT                         ;
                04              50 E8 000BC          BLBS    R0, 4$                                  ;
          04    AE              01 88 000BF          BISB2   #1, ATTRIBUTES                          ; 1635
          2C    AE              09 B0 000C3 4$:      MOVW    #9, DESC_LIST+32                        ; 1637
          30    AE        11    A7 9E 000C7          MOVAB   P.AAF, DESC_LIST+36                     ; 1638
                          2C    AE 9F 000CC          PUSHAB  DESC_LIST+32                            ; 1639
                66              01 FB 000CF          CALLS   #1, CLI$PRESENT                         ;
                04              50 E9 000D2          BLBC    R0, 5$                                  ;
          04    AE              02 88 000D5          BISB2   #2, ATTRIBUTES                          ; 1641
          2C    AE              0A B0 000D9 5$:      MOVW    #10, DESC_LIST+32                       ; 1643
          30    AE        1A    A7 9E 000DD          MOVAB   P.AAG, DESC_LIST+36                     ; 1644
                          2C    AE 9F 000E2          PUSHAB  DESC_LIST+32                            ; 1645
                66              01 FB 000E5          CALLS   #1, CLI$PRESENT                         ;
                04              50 E9 000E8          BLBC    R0, 6$                                  ;
          04    AE              04 88 000EB          BISB2   #4, ATTRIBUTES                          ; 1647
          2C    AE              08 B0 000EF 6$:      MOVW    #8, DESC_LIST+32                        ; 1653
          30    AE        24    A7 9E 000F3          MOVAB   P.AAH, DESC_LIST+36                     ; 1654
                                52 D4 000F8          CLRL    IF_STATE_PTR                            ; 1656
                          2C    AE 9F 000FA          PUSHAB  DESC_LIST+32                            ; 1657
                66              01 FB 000FD          CALLS   #1, CLI$PRESENT                         ;
                53              50 D0 00100          MOVL    R0, IF_STATE_PRESENT                    ;
                04              53 E9 00103          BLBC    IF_STATE_PRESENT, 7$                    ; 1658
                52        1C    AE 9E 00106          MOVAB   DESC_LIST+16, IF_STATE_PTR              ; 1660
                54              53 D2 0010A 7$:      MCOML   IF_STATE_PRESENT, R4                    ; 1709
                0C              53 E9 0010D 8$:      BLBC    IF_STATE_PRESENT, 9$                    ;
                          1C    AE 9F 00110          PUSHAB  DESC_LIST+16                            ; 1680
                          30    AE 9F 00113          PUSHAB  DESC_LIST+32                            ; 1679
                68              02 FB 00116          CALLS   #2, CLI$GET_VALUE                       ; 1680
                25              50 E9 00119          BLBC    STATUS, 12$                             ; 1681
                                55 DD 0011C 9$:      PUSHL   SET_STATE_PTR                           ; 1699
                          18    AE 9F 0011E          PUSHAB  DESC_LIST+8                             ; 1698
                          0C    AE 9F 00121          PUSHAB  ATTRIBUTES                              ; 1695
                                52 DD 00124          PUSHL   IF_STATE_PTR                            ; 1698
                          1C    AE 9F 00126          PUSHAB  DESC_LIST                               ; 1695
                          04    AC DD 00129          PUSHL   KEY_TABLE_ID                            ; 1698
              FA5A        CF              06 FB 0012C          CALLS   #6, SMG$ADD_KEY_DEF             ;
                          0A              50 E8 00131          BLBS    ADD_STATUS, 11$                 ; 1700
                                          50 DD 00134 10$:     PUSHL   ADD_STATUS                      ; 1702
        00000000G         00              01 FB 00136          CALLS   #1, LIB$SIGNAL                  ;
                                          04 00139             RET                                     ;
                          CC              54 E9 0013A 11$:     BLBC    R4, 8$                          ; 1709
                          0C    AE        9F 0013E 12$:        PUSHAB  DESC_LIST                       ; 1718
                    04    AE              04 D0 00144          MOVL    #4, 4(SP)                       ;
                          04    AE        9F 00148             PUSHAB  4(SP)                           ;
        00000000G         00              02 FB 0014B          CALLS   #2, LIB$SFREEN_DD               ;
                                          04 00152             RET                                     ; 1721
                                    0000  00153 13$:           .WORD   Save nothing                    ; 1497
                50        08    AC        D0 00155             MOVL    8(AP), R0                       ;
                50        04    A0        D0 00159             MOVL    4(R0), R0                       ;
                          D4    A0        9F 0015D             PUSHAB  DESC_LIST_PTR                   ;
                                01        DD 00160             PUSHL   #1                              ;
                                5E        DD 00162             PUSHL   SP                              ;
                7E        04    AC        7D 00164             MOVQ    4(AP), -(SP)                    ;
              0000V       CF              03 FB 00168          CALLS   #3, DEFINE_KEY_HANDLER          ;
                                          04 0016D             RET                                     ;

; Routine Size:  366 bytes,    Routine Base:  _SMG$CODE + 04C5
```

```
;  1657          1722  1 %SBTTL 'DEFINE_KEY_HANDLER'
;  1658          1723  1 ROUTINE DEFINE_KEY_HANDLER (
;  1659          1724  1     SIGARGS: REF BLOCK [, BYTE],        ! Signal arguments list
;  1660          1725  1     MCHARGS: REF BLOCK [, BYTE],        ! Mechanism arguments list
;  1661          1726  1     ENBARGS: REF VECTOR [, LONG]        ! Enable arguments list
;  1662          1727  1     ) =
;  1663          1728  1
;  1664          1729  1 !++
;  1665          1730  1 ! FUNCTIONAL DESCRIPTION:
;  1666          1731  1 !
;  1667          1732  1 !       This procedure is the condition handler for SMG$DEFINE_KEY.
;  1668          1733  1 !       It intercepts CLI$ errors signalled by CLI routines which are
;  1669          1734  1 !       called from SMG$DEFINE_KEY, and also any errors signalled
;  1670          1735  1 !       by SMG$DEFINE_KEY itself.  The handler frees the dynamic strings
;  1671          1736  1 !       which were allocated by SMG$DEFINE_KEY and causes a return to
;  1672          1737  1 !       the caller of SMG$DEFINE_KEY with a return status value of the
;  1673          1738  1 !       error which was signalled.
;  1674          1739  1 !
;  1675          1740  1 ! CALLING SEQUENCE:
;  1676          1741  1 !
;  1677          1742  1 !       Called by VMS exception handling
;  1678          1743  1 !
;  1679          1744  1 ! FORMAL PARAMETERS:
;  1680          1745  1 !
;  1681          1746  1 !       SIGARGS         - The signal arguments list
;  1682          1747  1 !
;  1683          1748  1 !       MCHARGS         - The mechanism arguments list
;  1684          1749  1 !
;  1685          1750  1 !       ENBARGS         - The enable arguments list.  The enable
;  1686          1751  1 !                         argument is:
;  1687          1752  1 !
;  1688          1753  1 !                               ..ENBARGS [1] - Pointer to the first of
;  1689          1754  1 !                                               four dynamic string
;  1690          1755  1 !                                               descriptors to be freed.
;  1691          1756  1 !
;  1692          1757  1 ! IMPLICIT INPUTS:
;  1693          1758  1 !
;  1694          1759  1 !       NONE
;  1695          1760  1 !
;  1696          1761  1 ! IMPLICIT OUTPUTS:
;  1697          1762  1 !
;  1698          1763  1 !       NONE
;  1699          1764  1 !
;  1700          1765  1 ! COMPLETION STATUS:
;  1701          1766  1 !
;  1702          1767  1 !       SS$_NORMAL      Normal successful completion
;  1703          1768  1 !       SS$_RESIGNAL    Resignal condition to next handler
;  1704          1769  1 !
;  1705          1770  1 ! SIDE EFFECTS:
;  1706          1771  1 !
;  1707          1772  1 !       Causes an unwind.
;  1708          1773  1 !
;  1709          1774  1 !--
;  1710          1775  1
;  1711          1776  2     BEGIN
;  1712          1777  2
;  1713          1778  2     LOCAL
```

```
;1714    1779  2          COND_NAME: BLOCK [4, BYTE];
;1715    1780  2
;1716    1781  2      COND_NAME = .SIGARGS [CHF$L_SIG_NAME];
;1717    1782  2
;1718    1783  2      !+
;1719    1784  2      ! Return if this is SS$_UNWIND.
;1720    1785  2      !-
;1721    1786  2
;1722    1787  2      IF .COND_NAME EQLU SS$_UNWIND
;1723    1788  2      THEN
;1724    1789  2          RETURN SS$_RESIGNAL;
;1725    1790  2
;1726    1791  2      !+
;1727    1792  2      ! If this is not a zero-level error and is not from the CLI$ facility,
;1728    1793  2      ! resignal.
;1729    1794  2      !
;1730    1795  2
;1731    1796  2      IF (.COND_NAME [STS$V_FAC_NO] NEQU CLI$_FACILITY) AND
;1732    1797  3         (.MCHARGS [CHF$L_MCH_DEPTH] NEQU 0)
;1733    1798  2      THEN
;1734    1799  2          RETURN SS$_RESIGNAL;
;1735    1800  2
;1736    1801  2      !+
;1737    1802  2      ! If the error is "no command on line", just change the return status
;1738    1803  2      ! to SS$_NORMAL, otherwise use the signalled status.
;1739    1804  2      !-
;1740    1805  2
;1741    1806  2      IF .COND_NAME EQLU CLI$_NOCOMD
;1742    1807  2      THEN
;1743    1808  2          MCHARGS [CHF$L_MCH_SAVR0] = SS$_NORMAL
;1744    1809  2      ELSE
;1745    1810  2          MCHARGS [CHF$L_MCH_SAVR0] = .COND_NAME;
;1746    1811  2
;1747    1812  2      !+
;1748    1813  2      ! Free the dynamic strings.
;1749    1814  2      !-
;1750    1815  2
;1751    1816  2      IF ..ENBARGS [1] NEQA 0
;1752    1817  2      THEN
;1753    1818  2          LIB$SFREEN_DD (%REF(4), ..ENBARGS [1]);
;1754    1819  2
;1755    1820  2      !+
;1756    1821  2      ! Unwind back to the caller of SMG$DEFINE_KEY.
;1757    1822  2      !-
;1758    1823  2
;1759    1824  2      RETURN $UNWIND ();
;1760    1825  2
;1761    1826  1      END;                                ! End of routine DEFINE_KEY_HANDLER


                                                .EXTRN  SYS$UNWIND

                                0000 00000 DEFINE_KEY_HANDLER:
                                                .WORD   Save nothing                    ;1723
                    5E          04  C2 00002            SUBL2   #4, SP                  ;
                    50      04  AC  D0 00005            MOVL    SIGARGS, R0             ;1781
```

```
                                 51       04    A0  D0 00009          MOVL    4(R0), COND_NAME
                          00000920       8F          51  D1 0000D          CMPL    COND_NAME, #2336                        ; 1787
                                                      10  13 00014          BEQL    1$
                    00G         51                    0C  10 ED 00016       CMPZV   #16, #12, COND_NAME, S^CLI$_FACILITY     ; 1796
                                                      0F  13 0001B          BEQL    2$
                                 50       08    AC  D0 0001D          MOVL    MCHARGS, R0                             ; 1797
                                          08    A0  D5 00021          TSTL    8(R0)
                                                      06  13 00024          BEQL    2$
                                 50     0918    8F  3C 00026 1$:       MOVZWL  #2328, R0                               ; 1799
                                                      04  0002B          RET
                                 50       08    AC  D0 0002C 2$:       MOVL    MCHARGS, R0                             ; 1808
                          00000000G      8F          51  D1 00030          CMPL    COND_NAME, #CLI$_NOCOMD                 ; 1806
                                                      06  12 00037          BNEQ    3$
                                0C      A0          01  D0 00039          MOVL    #1, 12(R0)                              ; 1808
                                                      04  11 0003D          BRB     4$
                                0C      A0          51  D0 0003F 3$:       MOVL    COND_NAME, 12(R0)                       ; 1810
                                 50       0C    AC  D0 00043 4$:       MOVL    ENBARGS, R0                             ; 1816
                                          04    B0  D5 00047          TSTL    @4(R0)
                                                      11  13 0004A          BEQL    5$
                                          04    B0  DD 0004C          PUSHL   @4(R0)                                  ; 1818
                                04      AE          04  D0 0004F          MOVL    #4, 4(SP)
                                          04    AE  9F 00053          PUSHAB  4(SP)
                          00000000G      00          02  FB 00056          CALLS   #2, LIB$SFREEN_DD
                                                      7E  7C 0005D 5$:       CLRQ    -(SP)                                   ; 1824
                          00000000G      00          02  FB 0005F          CALLS   #2, SYS$UNWIND                          ; 1826
                                                      04  00066          RET

; Routine Size:  103 bytes,    Routine Base:  _SMG$CODE + 0633
```

```
;  1763     1827  1 %SBTTL 'SMG$LOAD_KEY_DEFS'
;  1764     1828  1 GLOBAL ROUTINE SMG$LOAD_KEY_DEFS (
;  1765     1829  1     KEY_TABLE_ID: REF VECTOR [, LONG],
;  1766     1830  1     FILESPEC: REF BLOCK [, BYTE],
;  1767     1831  1     DEFAULT_FILESPEC: REF BLOCK [, BYTE],
;  1768     1832  1     LOGNAM_FLAG: REF VECTOR [, LONG]
;  1769     1833  1     ) =
;  1770     1834  1
;  1771     1835  1 !++
;  1772     1836  1 ! FUNCTIONAL DESCRIPTION:
;  1773     1837  1 !
;  1774     1838  1 !       This procedure opens and reads a file containing DEFINE/KEY
;  1775     1839  1 !       command lines and calls SMG$DEFINE_KEY for each line in the file.
;  1776     1840  1 !       This can be used by utilities to conveniently load a default set
;  1777     1841  1 !       of key definitions.
;  1778     1842  1 !
;  1779     1843  1 !       Use of SMG$LOAD_KEY_DEFS requires that the calling program be
;  1780     1844  1 !       run under the DCL command language interpreter.
;  1781     1845  1 !
;  1782     1846  1 ! CALLING SEQUENCE:
;  1783     1847  1 !
;  1784     1848  1 !       RET_STATUS.wlc.v = SMG$LOAD_KEY_DEFS (
;  1785     1849  1 !                               KEY_TABLE_ID.rl.r,
;  1786     1850  1 !                               FILESPEC.rt.dx
;  1787     1851  1 !                               [, [DEFAULT_FILESPEC.rt.dx]
;  1788     1852  1 !                               [, [DEFAULT_LOGNAM.rt.dx] ]])
;  1789     1853  1 !
;  1790     1854  1 ! FORMAL PARAMETERS:
;  1791     1855  1 !
;  1792     1856  1 !       KEY_TABLE_ID     A longword containing the identification of the
;  1793     1857  1 !                        key-definition table into which are to be loaded
;  1794     1858  1 !                        the key definitions.
;  1795     1859  1 !
;  1796     1860  1 !       FILESPEC         A string containing the file specification for the
;  1797     1861  1 !                        file of DEFINE/KEY commands.
;  1798     1862  1 !
;  1799     1863  1 !       DEFAULT_FILESPEC A string containing the default file specification
;  1800     1864  1 !                        for the file of DEFINE/KEY commands.  If omitted,
;  1801     1865  1 !                        the empty string is used.
;  1802     1866  1 !
;  1803     1867  1 ! IMPLICIT INPUTS:
;  1804     1868  1 !
;  1805     1869  1 !       NONE
;  1806     1870  1 !
;  1807     1871  1 ! IMPLICIT OUTPUTS:
;  1808     1872  1 !
;  1809     1873  1 !       NONE
;  1810     1874  1 !
;  1811     1875  1 ! COMPLETION STATUS:
;  1812     1876  1 !
;  1813     1877  1 !       SS$_NORMAL       Normal successful completion
;  1814     1878  1 !       SMG$_FILTOOLON   File specification is too long (over 255 characters)
;  1815     1879  1 !       SMG$_xxx         Any error status returned from SMG$DEFINE_KEY
;  1816     1880  1 !       RMS$_xxx         Any error status returned from $OPEN
;  1817     1881  1 !
;  1818     1882  1 ! SIDE EFFECTS:
;  1819     1883  1 !
```

```
;1820   1884  1 !       Errors encountered while processing command lines are signalled.
;1821   1885  1 !
;1822   1886  1 !--
;1823   1887  1
;1824   1888  2     BEGIN
;1825   1889  2
;1826   1890  2     LOCAL
;1827   1891  2         KEYBOARD_ID,                                ! Keyboard id
;1828   1892  2         FILESPEC_PTR,                               ! Pointer to filespec desc
;1829   1893  2         RSLBUF: VECTOR [LNM$C_NAMLENGTH, BYTE],     ! Resultant buffer
;1830   1894  2         RSLBUF_DESC: BLOCK [8, BYTE],               ! Resultant descriptor
;1831   1895  2         LOGNAM_DESC: BLOCK [8, BYTE],               ! Logical name descriptor
;1832   1896  2         LINE_DESC: BLOCK [8, BYTE],                 ! Command line descriptor
;1833   1897  2         OPEN_STATUS;
;1834   1898  2
;1835   1899  2     BUILTIN
;1836   1900  2         ACTUALCOUNT,
;1837   1901  2         NULLPARAMETER;
;1838   1902  2
;1839   1903  2     LITERAL
;1840   1904  2         K_KEY_TABLE_ID = 1,
;1841   1905  2         K_FILESPEC = 2,
;1842   1906  2         K_DEFAULT_FILESPEC = 3,
;1843   1907  2         K_LOGNAM_FLAG = 4;
;1844   1908  2
;1845   1909  2     !+
;1846   1910  2     ! If LOGNAM_FLAG was specified, replace FNM with the null string if
;1847   1911  2     ! it doesn't translate as a logical name.
;1848   1912  2     !-
;1849   1913  2
;1850   1914  2     FILESPEC_PTR = FILESPEC [0,0,0,0];
;1851   1915  2     IF NOT NULLPARAMETER (K_LOGNAM_FLAG)
;1852   1916  2     THEN
;1853   1917  3         BEGIN
;1854   1918  3         IF .LOGNAM_FLAG [0]
;1855   1919  3         THEN
;1856   1920  4             BEGIN
;1857   1921  4             !+
;1858   1922  4             ! Set up the descriptors.
;1859   1923  4             !-
;1860   1924  4
;1861   1925  4             LOCAL
;1862   1926  4                 ANL_STATUS;
;1863   1927  4             ANL_STATUS = LIB$ANALYZE_SDESC_R2 (FILESPEC [0,0,0,0];
;1864   1928  4                 LOGNAM_DESC [DSC$W_LENGTH], LOGNAM_DESC [DSC$A_POINTER]);
;1865   1929  4             IF NOT .ANL_STATUS
;1866   1930  4             THEN
;1867   1931  4                 RETURN .ANL_STATUS;
;1868   1932  4             LOGNAM_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;1869   1933  4             LOGNAM_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
;1870   1934  4             RSLBUF_DESC [DSC$W_LENGTH] = LNM$C_NAMLENGTH;
;1871   1935  4             RSLBUF_DESC [DSC$A_POINTER] = RSLBUF;
;1872   1936  4             RSLBUF_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;1873   1937  4             RSLBUF_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
;1874   1938  4
;1875   1939  4             !+
;1876   1940  4             ! Try translating the logical name.  If it doesn't translate,
```

```
;1877     1941  4              ! use a zero-length string as the filespec.
;1878     1942  4              !-
;1879     1943  4
;1880     1944  5              IF ($TRNLOG (LOGNAM=LOGNAM_DESC, RSLBUF=RSLBUF_DESC) EQLU
;1881     1945  5                  SS$_NOTRAN)
;1882     1946  4              THEN
;1883     1947  5                  BEGIN
;1884     1948  5                  LOGNAM_DESC [DSC$W_LENGTH] = 0;
;1885     1949  5                  FILESPEC_PTR = LOGNAM_DESC;
;1886     1950  4                  END;
;1887     1951  3              END;
;1888     1952  2          END;
;1889     1953  2
;1890     1954  2
;1891     1955  2      !+
;1892     1956  2      ! Open the file.
;1893     1957  2      !-
;1894     1958  2
;1895     1959  2      OPEN_STATUS = SMG$CREATE_VIRTUAL_KEYBOARD (KEYBOARD_ID, .FILESPEC_PTR,
;1896     1960  3          (IF ACTUALCOUNT() GEQU K_DEFAULT_FILESPEC
;1897     1961  3           THEN
;1898     1962  3               .DEFAULT_FILESPEC
;1899     1963  3           ELSE
;1900     1964  2               0));
;1901     1965  2      IF NOT .OPEN_STATUS
;1902     1966  2      THEN
;1903     1967  2          RETURN .OPEN_STATUS;
;1904     1968  2
;1905     1969  2      !+
;1906     1970  2      ! Set up LINE_DESC.
;1907     1971  2      !-
;1908     1972  2
;1909     1973  2      LINE_DESC [DSC$W_LENGTH] = 0;
;1910     1974  2      LINE_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
;1911     1975  2      LINE_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
;1912     1976  2      LINE_DESC [DSC$A_POINTER] = 0;
;1913     1977  2
;1914     1978  2      !+
;1915     1979  2      ! While we're not at EOF, read records and call SMG$DEFINE_KEY to
;1916     1980  2      ! process the command line.
;1917     1981  2      !-
;1918     1982  2
;1919     1983  2      WHILE (SMG$READ_STRING (KEYBOARD_ID,LINE_DESC)) DO
;1920     1984  3          BEGIN
;1921     1985  3          LOCAL
;1922     1986  3              DEFINE_STATUS;
;1923     1987  3          DEFINE_STATUS = SMG$DEFINE_KEY (KEY_TABLE_ID [0], LINE_DESC);
;1924     1988  3          IF NOT .DEFINE_STATUS
;1925     1989  3          THEN
;1926     1990  3              SIGNAL (.DEFINE_STATUS);
;1927     1991  2          END;
;1928     1992  2
;1929     1993  2      LIB$SFREE1_DD (LINE_DESC);
;1930     1994  2      SMG$DELETE_VIRTUAL_KEYBOARD (KEYBOARD_ID);
;1931     1995  2
;1932     1996  2      RETURN SS$_NORMAL;
;1933     1997  2
```

```
; 1934        1998  1     END;                                    ! End of routine SMG$LOAD_KEY_DEFS


                                                                  .EXTRN  SYS$TRNLOG

                                                 000C 00000       .ENTRY  SMG$LOAD_KEY_DEFS, Save R2,R3                  ; 1828
                                           5E    FEE4   CE  9E 00002       MOVAB   -284(SP), SP                           ;
                                           53      08   AC  D0 00007       MOVL    FILESPEC, FILESPEC_PTR                 ; 1914
                                           04           6C  91 0000B       CMPB    (AP), #4                               ; 1915
                                                        54  1F 0000E       BLSSU   1$                                     ;
                                                  10    AC  D5 00010       TSTL    16(AP)                                 ;
                                                        4F  13 00013       BEQL    1$                                     ;
                                           4B           10  BC  E9 00015       BLBC    @LOGNAM_FLAG, 1$                       ; 1918
                                           50           08  AC  D0 00019       MOVL    FILESPEC, R0                           ; 1928
                                              00000000G     00  16 0001D       JSB     LIB$ANALYZE_SDESC_R2                   ;
                                     0C    AE              51  B0 00023       MOVW    R1, LOGNAM_DESC                        ;
                                     10    AE              52  D0 00027       MOVL    R2, LOGNAM_DESC+4                      ;
                                           4E              50  E9 0002B       BLBC    ANL_STATUS, 4$                         ; 1929
                                     0E    AE        010E  8F  B0 0002E       MOVW    #270, LOGNAM_DESC+2                    ; 1932
                                     18    AE          1C  AE  9E 00034       MOVAB   RSLBUF, RSLBUF_DESC+4                  ; 1935
                                     14    AE  010E00FF  8F  D0 00039       MOVL    #17694975, RSLBUF_DESC                 ; 1934
                                                       7E  7C 00041       CLRQ    -(SP)                                  ; 1944
                                                       7E  D4 00043       CLRL    -(SP)                                  ;
                                                  20   AE  9F 00045       PUSHAB  RSLBUF_DESC                            ;
                                                       7E  D4 00048       CLRL    -(SP)                                  ;
                                                  20   AE  9F 0004A       PUSHAB  LOGNAM_DESC                            ;
                                 00000000G        00      06  FB 0004D       CALLS   #6, SYS$TRNLOG                         ;
                                 00000629         8F              50  D1 00054       CMPL    R0, #1577                      ;
                                                                 07  12 0005B       BNEQ    1$                             ;
                                                  0C   AE  B4 0005D       CLRW    LOGNAM_DESC                            ; 1948
                                                  0C   AE  9E 00060       MOVAB   LOGNAM_DESC, FILESPEC_PTR              ; 1949
                                           53           03  6C  91 00064 1$:     CMPB    (AP), #3                       ; 1960
                                           03                   05  1F 00067       BLSSU   2$                             ;
                                                  0C   AC  DD 00069       PUSHL   DEFAULT_FILESPEC                       ; 1962
                                                       02  11 0006C       BRB     3$                                     ;
                                                       7E  D4 0006E 2$:     CLRL    -(SP)                                  ; 1960
                                                       53  DD 00070 3$:     PUSHL   FILESPEC_PTR                           ; 1959
                                                  08   AE  9F 00072       PUSHAB  KEYBOARD_ID                            ;
                                 00000000G        00      03  FB 00075       CALLS   #3, SMG$CREATE_VIRTUAL_KEYBOARD        ;
                                                  4A              50  E9 0007C 4$:     BLBC    OPEN_STATUS, 7$                ; 1965
                                     04    AE  020E0000  8F  D0 0007F       MOVL    #34471936, LINE_DESC                   ; 1973
                                                  08   AE  D4 00087       CLRL    LINE_DESC+4                            ; 1976
                                                  04   AE  9F 0008A 5$:     PUSHAB  LINE_DESC                              ; 1983
                                                  04   AE  9F 0008D       PUSHAB  KEYBOARD_ID                            ;
                                 00000000G        00      02  FB 00090       CALLS   #2, SMG$READ_STRING                    ;
                                                  19              50  E9 00097       BLBC    R0, 6$                         ;
                                                  04   AE  9F 0009A       PUSHAB  LINE_DESC                              ; 1987
                                                  04   AC  DD 0009D       PUSHL   KEY_TABLE_ID                           ;
                                     FD86         CF      02  FB 000A0       CALLS   #2, SMG$DEFINE_KEY                     ;
                                                  E2              50  E8 000A5       BLBS    DEFINE_STATUS, 5$              ; 1988
                                                                  50  DD 000A8       PUSHL   DEFINE_STATUS                  ; 1990
                                 00000000G        00              01  FB 000AA       CALLS   #1, LIB$SIGNAL                 ;
                                                                  D7  11 000B1       BRB     5$                             ; 1983
                                                  04   AE  9F 000B3 6$:     PUSHAB  LINE_DESC                              ; 1993
                                 00000000G        00              01  FB 000B6       CALLS   #1, LIB$SFREE1_DD              ;
                                                                  5E  DD 000BD       PUSHL   SP                             ; 1994
```

```
                    00000000G  00          01  FB 000BF          CALLS   #1, SMG$DELETE_VIRTUAL_KEYBOARD          ;
                               50          01  D0 000C6          MOVL    #1, R0                                  ; 1996
                                           04  000C9 7$:        RET                                             ; 1998

; Routine Size:  202 bytes,    Routine Base:  _SMG$CODE + 069A
```

```
;  1936          1999  1 %SBTTL 'SMG$LIST_KEY_DEFS'
;  1937          2000  1 GLOBAL ROUTINE SMG$LIST_KEY_DEFS (
;  1938          2001  1     KEY_TABLE_ID: REF VECTOR [, LONG],
;  1939          2002  1     CONTEXT: REF VECTOR [, LONG],
;  1940          2003  1     KEY_NAME: REF BLOCK [, BYTE],
;  1941          2004  1     IF_STATE: REF BLOCK [, BYTE],
;  1942          2005  1     ATTRIBUTES: REF VECTOR [, LONG],
;  1943          2006  1     EQUIV_STRING: REF BLOCK [, BYTE],
;  1944          2007  1     STATE_STRING: REF BLOCK [, BYTE]
;  1945          2008  1     ) =
;  1946          2009  1
;  1947          2010  1 !++
;  1948          2011  1 ! FUNCTIONAL DESCRIPTION:
;  1949          2012  1 !
;  1950          2013  1 !       This procedure, when called repeatedly, returns all of the
;  1951          2014  1 !       key definitions in a key table.
;  1952          2015  1 !
;  1953          2016  1 ! CALLING SEQUENCE:
;  1954          2017  1 !
;  1955          2018  1 !       RET_STATUS.wlc.v = SMG$LIST_KEY_DEFS (
;  1956          2019  1 !                                  KEY_TABLE_ID.rl.r,
;  1957          2020  1 !                                  CONTEXT.ml.r,
;  1958          2021  1 !                                  [, [KEY_NAME.rt.dx]
;  1959          2022  1 !                                  [, [IF_STATE.rt.dx]
;  1960          2023  1 !                                  [, [ATTRIBUTES.wl.r]
;  1961          2024  1 !                                  [, [EQUIV_STRING.wt.dx]
;  1962          2025  1 !                                  [, [STATE_STRING.wt.dx] ]]]]])
;  1963          2026  1 !
;  1964          2027  1 ! FORMAL PARAMETERS:
;  1965          2028  1 !
;  1966          2029  1 !       KEY_TABLE_ID    A longword containing the identification of the
;  1967          2030  1 !                       key-definition table which is to be listed.
;  1968          2031  1 !
;  1969          2032  1 !       CONTEXT         A longword which supplies context to the search.
;  1970          2033  1 !                       On the initial call, CONTEXT should be set to zero
;  1971          2034  1 !                       by the caller.  It will be updated by SMG$LIST_KEY_DEFS;
;  1972          2035  1 !                       the updated value should then be passed on subsequent
;  1973          2036  1 !                       calls to obtain the next key definition.
;  1974          2037  1 !
;  1975          2038  1 !       KEY_NAME        A string into which is stored the key name of the next
;  1976          2039  1 !                       definition in the key table.
;  1977          2040  1 !
;  1978          2041  1 !       IF_STATE        A string into which is stored the state name which
;  1979          2042  1 !                       qualifies the next definition in the key table
;  1980          2043  1 !
;  1981          2044  1 !       ATTRIBUTES      A longword into which is stored the attributes
;  1982          2045  1 !                       bitmask for the next key definition.  See the description
;  1983          2046  1 !                       of SMG$ADD_KEY_DEF for more information.
;  1984          2047  1 !
;  1985          2048  1 !       EQUIV_STRING    A string into which is stored the equivalence-string
;  1986          2049  1 !                       for the next key definition.
;  1987          2050  1 !
;  1988          2051  1 !       STATE_STRING    A string into which is stored the new state-name,
;  1989          2052  1 !                       if any, which is set by the next key definition.
;  1990          2053  1 !                       If this key definition sets a state, the ATTRIBUTES
;  1991          2054  1 !                       flag SMG$V_KEY_SETSTATE is set.
;  1992          2055  1 !
```

```
;1993   2056  1 ! IMPLICIT INPUTS:
;1994   2057  1 !
;1995   2058  1 !       NONE
;1996   2059  1 !
;1997   2060  1 ! IMPLICIT OUTPUTS:
;1998   2061  1 !
;1999   2062  1 !       NONE
;2000   2063  1 !
;2001   2064  1 ! COMPLETION STATUS:
;2002   2065  1 !
;2003   2066  1 !       SS$_NORMAL      Normal successful completion
;2004   2067  1 !       SMG$_INVKTB_ID  Invalid key-table id
;2005   2068  1 !       SMG$_NOMOREKEYS No more keys in table
;2006   2069  1 !       LIB$_xxx        Any error status from LIB$SCOPY_DXDX
;2007   2070  1 !
;2008   2071  1 ! SIDE EFFECTS:
;2009   2072  1 !
;2010   2073  1 !       NONE
;2011   2074  1 !
;2012   2075  1 !--
;2013   2076  1
;2014   2077  2     BEGIN
;2015   2078  2
;2016   2079  2     LOCAL
;2017   2080  2         FOUND_KDE,                          ! KDE found
;2018   2081  2         KDE: REF KDE_R_KDE_STRUCT,          ! Key definition entry
;2019   2082  2         KTH: REF KTH_R_KTH_STRUCT,          ! Key table header
;2020   2083  2         DUMMY_KDE: KDE_R_KDE_STRUCT;        ! Used to store first entry
;2021   2084  2
;2022   2085  2     BUILTIN
;2023   2086  2         AP,
;2024   2087  2         NULLPARAMETER;
;2025   2088  2
;2026   2089  2     !+
;2027   2090  2     ! Define positions of arguments
;2028   2091  2     !-
;2029   2092  2
;2030   2093  2     LITERAL
;2031   2094  2         K_KEY_TABLE_ID = 1,
;2032   2095  2         K_CONTEXT = 2,
;2033   2096  2         K_KEY_NAME = 3,
;2034   2097  2         K_IF_STATE = 4,
;2035   2098  2         K_ATTRIBUTES = 5,
;2036   2099  2         K_EQUIV_STRING = 6,
;2037   2100  2         K_STATE_STRING = 7;
;2038   2101  2
;2039   2102  2     !+
;2040   2103  2     ! Validate KEY_TABLE_ID and get the Key Table Header.
;2041   2104  2     !-
;2042   2105  2
;2043   2106  2     $SMG$VALIDATE_KTH (KEY_TABLE_ID, KTH);
;2044   2107  2
;2045   2108  2     !+
;2046   2109  2     ! If this is the first time, form a linear linked list of the tree.
;2047   2110  2     !-
;2048   2111  2
;2049   2112  2     KDE = .CONTEXT [0];
```

```
;  2050          2113  2      IF .KDE EQLA 0
;  2051          2114  2      THEN
;  2052          2115  3          BEGIN
;  2053          2116  3          LOCAL
;  2054          2117  3              LAST_KDE: REF KDE_R_KDE_STRUCT,
;  2055          2118  3              TRAVERSE_STATUS;
;  2056          2119  3
;  2057          2120  3          LAST_KDE = DUMMY_KDE;
;  2058          2121  3          DUMMY_KDE [KDE_A_NEXT] = 0;
;  2059          2122  3          TRAVERSE_STATUS = LIB$TRAVERSE_TREE (
;  2060          2123  3              KTH [KTH_A_TREEHEAD],       ! Address of treehead
;  2061          2124  3              TRAVERSE_ROUTINE,           ! Traverse routine
;  2062          2125  3              LAST_KDE);                  ! Pointer to previous KDE
;  2063          2126  3          IF NOT .TRAVERSE_STATUS
;  2064          2127  3          THEN
;  2065          2128  3              RETURN .TRAVERSE_STATUS;
;  2066          2129  3          KDE = DUMMY_KDE;        ! Get first KDE
;  2067          2130  2          END;
;  2068          2131  2
;  2069          2132  2      !+
;  2070          2133  2      ! Get address of next KDE.  If no more keys, return status.
;  2071          2134  2      ! Skip over deleted keys.
;  2072          2135  2      !-
;  2073          2136  2
;  2074          2137  2      DO
;  2075          2138  3          BEGIN
;  2076          2139  3          KDE = .KDE [KDE_A_NEXT];
;  2077          2140  3          CONTEXT [0] = .KDE;
;  2078          2141  3          IF .KDE EQLA 0
;  2079          2142  3          THEN
;  2080          2143  3              RETURN SMG$_NOMOREKEYS;
;  2081          2144  3          END
;  2082          2145  2      UNTIL .KDE [KDE_V_DEFINED];
;  2083          2146  2
;  2084          2147  2      !+
;  2085          2148  2      ! Store values.
;  2086          2149  2      !-
;  2087          2150  2
;  2088          2151  2      IF NOT NULLPARAMETER (K_ATTRIBUTES)
;  2089          2152  2      THEN
;  2090          2153  2          ATTRIBUTES [0] = .KDE [KDE_L_ATTR] AND KDE_M_USER_ATTR;
;  2091          2154  2
;  2092          2155  2      IF NOT NULLPARAMETER (K_KEY_NAME)
;  2093          2156  2      THEN
;  2094          2157  3          BEGIN
;  2095          2158  3          LOCAL
;  2096          2159  3              KEY_NAME_AREA: VECTOR [32, BYTE],
;  2097          2160  3              KEY_CODE_PTR: REF VECTOR [, BYTE],
;  2098          2161  3              KEY_CODE: WORD,
;  2099          2162  3              STATUS;
;  2100          2163  3
;  2101          2164  3          !+
;  2102          2165  3          ! Convert key code to key name.
;  2103          2166  3          !-
;  2104          2167  3
;  2105          2168  4          KEY_CODE_PTR = (.KDE [KDE_A_TKEY_POINTER] + .KDE [KDE_W_TKEY_LENGTH])
;  2106          2169  3              - 2;
```

```
;  2107     2170  3          KEY_CODE<8,8> = .KEY_CODE_PTR [0]; ! Swap bytes
;  2108     2171  3          KEY_CODE<0,8> = .KEY_CODE_PTR [1];
;  2109     2172  3          IF NOT SMG$$KEYCODE_TO_NAME (.KEY_CODE, KEY_NAME_AREA)
;  2110     2173  3          THEN
;  2111     2174  3              KEY_NAME_AREA [0] = 0;
;  2112     2175  3          STATUS = LIB$SCOPY_R_DX6 (.KEY_NAME_AREA [0], KEY_NAME_AREA [1],
;  2113     2176  3              KEY_NAME [0,0,0,0]);
;  2114     2177  3          IF NOT .STATUS
;  2115     2178  3          THEN
;  2116     2179  3              RETURN .STATUS;
;  2117     2180  2          END;
;  2118     2181  2
;  2119     2182  2      IF NOT NULLPARAMETER (K_IF_STATE)
;  2120     2183  2      THEN
;  2121     2184  3          BEGIN
;  2122     2185  3          LOCAL
;  2123     2186  3              STATUS;
;  2124     2187  3          STATUS = LIB$SCOPY_R_DX6 (.KDE [KDE_W_TKEY_LENGTH] - 3,
;  2125     2188  3              .KDE [KDE_A_TKEY_POINTER], IF_STATE [0,0,0,0]);
;  2126     2189  3          IF NOT .STATUS
;  2127     2190  3          THEN
;  2128     2191  3              RETURN .STATUS;
;  2129     2192  2          END;
;  2130     2193  2
;  2131     2194  2      IF NOT NULLPARAMETER (K_EQUIV_STRING)
;  2132     2195  2      THEN
;  2133     2196  3          BEGIN
;  2134     2197  3          LOCAL
;  2135     2198  3              STATUS;
;  2136     2199  3          STATUS = LIB$SCOPY_DXDX6 (KDE [KDE_R_EQUIV_DESC],
;  2137     2200  3              EQUIV_STRING [0,0,0,0]);
;  2138     2201  3          IF NOT .STATUS
;  2139     2202  3          THEN
;  2140     2203  3              RETURN .STATUS;
;  2141     2204  2          END;
;  2142     2205  2
;  2143     2206  2      IF NOT NULLPARAMETER (K_STATE_STRING)
;  2144     2207  2      THEN
;  2145     2208  3          BEGIN
;  2146     2209  3          LOCAL
;  2147     2210  3              STATUS;
;  2148     2211  3          STATUS = LIB$SCOPY_DXDX6 (KDE [KDE_R_STATE_DESC],
;  2149     2212  3              STATE_STRING [0,0,0,0]);
;  2150     2213  3          IF NOT .STATUS
;  2151     2214  3          THEN
;  2152     2215  3              RETURN .STATUS;
;  2153     2216  2          END;
;  2154     2217  2
;  2155     2218  2      RETURN SS$_NORMAL;
;  2156     2219  2
;  2157     2220  1      END;                                    ! End of routine SMG$LIST_KEY_DEFS


                                        03FC 00000          .ENTRY  SMG$LIST_KEY_DEFS, Save R2,R3,R4,R5,R6,R7,- ; 2000
```

```
                                                                       R8,R9                              ;
                  59 00000000G   00  9E 00002         MOVAB   LIB$SCOPY_DXDX6, R9                ;
                  58 00000000G   00  9E 00009         MOVAB   LIB$SCOPY_R_DX6, R8                ;
                  5E        A4   AE  9E 00010         MOVAB   -92(SP), SP                        ;
                  50        04   BC  D0 00014         MOVL    @KEY_TABLE_ID, KTH                 ; 2106
                            06   13     00018         BEQL    1$                                 ;
                  50        10   A0  D1 0001A         CMPL    16(KTH), KTH                       ;
                            08   13     0001E         BEQL    2$                                 ;
                  50 00000000G   8F  D0 00020 1$:     MOVL    #SMG$_INVKTB_ID, R0                ;
                                 04     00027         RET                                        ;
                  57        08   BC  D0 00028 2$:     MOVL    @CONTEXT, KDE                      ; 2112
                                 1D  12 0002C         BNEQ    3$                                 ; 2113
                  6E        24   AE  9E 0002E         MOVAB   DUMMY_KDE, LAST_KDE                ; 2120
                            50   AE  D4 00032         CLRL    DUMMY_KDE+44                       ; 2121
                                 5E  DD 00035         PUSHL   SP                                 ; 2123
                         0000V   CF  9F 00037         PUSHAB  TRAVERSE_ROUTINE                   ;
                                 50  DD 0003B         PUSHL   KTH                                ;
         00000000G   00          03  FB 0003D         CALLS   #3, LIB$TRAVERSE_TREE              ;
                     69          50  E9 00044         BLBC    TRAVERSE_STATUS, 7$                ; 2126
                     57     24   AE  9E 00047         MOVAB   DUMMY_KDE, KDE                     ; 2129
                     57     2C   A7  D0 0004B 3$:     MOVL    44(KDE), KDE                       ; 2139
              08     BC          57  D0 0004F         MOVL    KDE, @CONTEXT                      ; 2140
                                 08  12 00053         BNEQ    4$                                 ; 2141
                  50 00000000G   8F  D0 00055         MOVL    #SMG$_NOMOREKEYS, R0               ; 2143
                                 04     0005C         RET                                        ;
                            33   A7  95 0005D 4$:     TSTB    51(KDE)                            ; 2145
                                 E9  18 00060         BGEQ    3$                                 ;
                     05          6C  91 00062         CMPB    (AP), #5                           ; 2151
                                 0A  1F 00065         BLSSU   5$                                 ;
                            14   AC  D5 00067         TSTL    20(AP)                             ;
                                 05  13 0006A         BEQL    5$                                 ;
              14     BC     30   A7  9A 0006C         MOVZBL  48(KDE), @ATTRIBUTES               ; 2153
                     03          6C  91 00071 5$:     CMPB    (AP), #3                           ; 2155
                                 3D  1F 00074         BLSSU   8$                                 ;
                            0C   AC  D5 00076         TSTL    12(AP)                             ;
                                 38  13 00079         BEQL    8$                                 ;
                     50     0C   A7  3C 0007B         MOVZWL  12(KDE), R0                        ; 2168
                     50     10   A7  C0 0007F         ADDL2   16(KDE), R0                        ;
                     50          02  C2 00083         SUBL2   #2, KEY_CODE_PTR                   ; 2169
    51        08     08          60  F0 00086         INSV    (KEY_CODE_PTR), #8, #8, KEY_CODE   ; 2170
                     51     01   A0  90 0008B         MOVB    1(KEY_CODE_PTR), KEY_CODE          ; 2171
                            04   AE  9F 0008F         PUSHAB  KEY_NAME_AREA                      ; 2172
                     7E          51  3C 00092         MOVZWL  KEY_CODE, -(SP)                    ;
         00000000G   00          02  FB 00095         CALLS   #2, SMG$$KEYCODE_TO_NAME           ;
                     03          50  E8 0009C         BLBS    R0, 6$                             ;
                            04   AE  94 0009F         CLRB    KEY_NAME_AREA                      ; 2174
                     51     05   AE  9E 000A2 6$:     MOVAB   KEY_NAME_AREA+1, R1                ; 2175
                     52     0C   AC  D0 000A6         MOVL    KEY_NAME, R2                       ; 2176
                     50     04   AE  9A 000AA         MOVZBL  KEY_NAME_AREA, R0                  ;
                                 68  16 000AE         JSB     LIB$SCOPY_R_DX6                    ;
                     4F          50  E9 000B0 7$:     BLBC    STATUS, 12$                        ; 2177
                     04          6C  91 000B3 8$:     CMPB    (AP), #4                           ; 2182
                                 19  1F 000B6         BLSSU   9$                                 ;
                            10   AC  D5 000B8         TSTL    16(AP)                             ;
                                 14  13 000BB         BEQL    9$                                 ;
                     50     0C   A7  3C 000BD         MOVZWL  12(KDE), R0                        ; 2187
                     50          03  C2 000C1         SUBL2   #3, R0                             ;
```

```
                                52      10    AC  D0 000C4          MOVL    IF_STATE, R2                    ; 2188
                                51      10    A7  D0 000C8          MOVL    16(KDE), R1                     ;
                                              68  16 000CC          JSB     LIB$SCOPY_R_DX6                 ;
                                31            50  E9 000CE          BLBC    STATUS, 12$                     ; 2189
                                06            6C  91 000D1 9$:      CMPB    (AP), #6                        ; 2194
                                              12  1F 000D4          BLSSU   10$                             ;
                                      18      AC  D5 000D6          TSTL    24(AP)                          ;
                                              0D  13 000D9          BEQL    10$                             ;
                                50    14      A7  9E 000DB          MOVAB   20(KDE), R0                     ; 2199
                                51    18      AC  D0 000DF          MOVL    EQUIV_STRING, R1                ; 2200
                                              69  16 000E3          JSB     LIB$SCOPY_DXDX6                 ;
                                1A            50  E9 000E5          BLBC    STATUS, 12$                     ; 2201
                                07            6C  91 000E8 10$:     CMPB    (AP), #7                        ; 2206
                                              12  1F 000EB          BLSSU   11$                             ;
                                      1C      AC  D5 000ED          TSTL    28(AP)                          ;
                                              0D  13 000F0          BEQL    11$                             ;
                                50    1C      A7  9E 000F2          MOVAB   28(KDE), R0                     ; 2211
                                51    1C      AC  D0 000F6          MOVL    STATE_STRING, R1                ; 2212
                                              69  16 000FA          JSB     LIB$SCOPY_DXDX6                 ;
                                03            50  E9 000FC          BLBC    STATUS, 12$                     ; 2213
                                50            01  D0 000FF 11$:     MOVL    #1, R0                          ; 2218
                                                  04 00102 12$:     RET                                     ; 2220

; Routine Size:  259 bytes,    Routine Base:  _SMG$CODE + 0764
```

```
;  2159          2221  1 %SBTTL 'TRAVERSE_ROUTINE'
;  2160          2222  1 ROUTINE TRAVERSE_ROUTINE (
;  2161          2223  1     NEXT_KDE_PTR:,                          ! Pointer to next KDE
;  2162          2224  1     LAST_KDE_PTR: REF VECTOR [, LONG]       ! Contains pointer to last KDE
;  2163          2225  1     ) =
;  2164          2226  1
;  2165          2227  1 !++
;  2166          2228  1 ! FUNCTIONAL DESCRIPTION:
;  2167          2229  1 !
;  2168          2230  1 !       Routine called by LIB$TRAVERSE_TREE.  It is used to link in this
;  2169          2231  1 !       KDE to the list that is being built of KDEs in order.
;  2170          2232  1 !
;  2171          2233  1 ! CALLING SEQUENCE:
;  2172          2234  1 !
;  2173          2235  1 !       Not directly called.
;  2174          2236  1 !
;  2175          2237  1 ! FORMAL PARAMETERS·
;  2176          2238  1 !
;  2177          2239  1 !       NEXT_KDE_PTR    A longword that contains a pointer to the next
;  2178          2240  1 !                       KDE.
;  2179          2241  1 !
;  2180          2242  1 !       LAST_KDE_PTR    A longword that contains a pointer to the previous
;  2181          2243  1 !                       KDE.
;  2182          2244  1 !
;  2183          2245  1 ! IMPLICIT INPUTS:
;  2184          2246  1 !
;  2185          2247  1 !       NONE
;  2186          2248  1 !
;  2187          2249  1 ! IMPLICIT OUTPUTS:
;  2188          2250  1 !
;  2189          2251  1 !       NONE
;  2190          2252  1 !
;  2191          2253  1 ! COMPLETION STATUS:
;  2192          2254  1 !
;  2193          2255  1 !       SS$_NORMAL      Normal successful completion
;  2194          2256  1 !
;  2195          2257  1 ! SIDE EFFECTS:
;  2196          2258  1 !
;  2197          2259  1 !       NONE
;  2198          2260  1 !
;  2199          2261  1 !--
;  2200          2262  1
;  2201          2263  2     BEGIN
;  2202          2264  2
;  2203          2265  2     LOCAL
;  2204          2266  2         THIS_KDE: REF KDE_R_KDE_STRUCT, ! This Key Definition Entry
;  2205          2267  2         LAST_KDE: REF KDE_R_KDE_STRUCT; ! Previous Key Definition Entry
;  2206          2268  2
;  2207          2269  2     !+
;  2208          2270  2     ! Get pointers.
;  2209          2271  2     !-
;  2210          2272  2
;  2211          2273  2     THIS_KDE = .NEXT_KDE_PTR;
;  2212          2274  2     LAST_KDE = .LAST_KDE_PTR [0];
;  2213          2275  2
;  2214          2276  2     !+
;  2215          2277  2     ! Link in this KDE and zero the next pointer.
```

```
;  2216          2278  2     !-
;  2217          2279  2
;  2218          2280  2     LAST_KDE [KDE_A_NEXT] = THIS_KDE [0,0,0,0];
;  2219          2281  2     THIS_KDE [KDE_A_NEXT] = 0;
;  2220          2282  2
;  2221          2283  2     !+
;  2222          2284  2     ! Update the "last KDE" pointer.
;  2223          2285  2     !-
;  2224          2286  2
;  2225          2287  2     LAST_KDE_PTR [0] = THIS_KDE [0,0,0,0];
;  2226          2288  2
;  2227          2289  2     RETURN SS$_NORMAL;
;  2228          2290  2
;  2229          2291  1     END;                                    ! End of routine TRAVERSE_ROUTINE



                                        0000 00000 TRAVERSE_ROUTINE:
                                                   .WORD   Save nothing                                    ; 2222
                          51      04    AC   D0 00002    MOVL    NEXT_KDE_PTR, THIS_KDE                   ; 2273
                          50      08    BC   D0 00006    MOVL    @LAST_KDE_PTR, LAST_KDE                  ; 2274
                    2C    A0            51   D0 0000A    MOVL    THIS_KDE, 44(LAST_KDE)                   ; 2280
                                  2C    A1   D4 0000E    CLRL    44(THIS_KDE)                             ; 2281
                    08    BC            51   D0 00011    MOVL    THIS_KDE, @LAST_KDE_PTR                  ; 2287
                          50            01   D0 00015    MOVL    #1, R0                                   ; 2289
                                             04 00018    RET                                              ; 2291

; Routine Size:  25 bytes,    Routine Base:  _SMG$CODE + 0867
```

```
; 2231          2292  1 END                                   ! End of module SMG$KEYPAD
; 2232          2293  1
; 2233          2294  0 ELUDOM



                                                              .EXTRN  LIB$SIGNAL

;                                PSECT SUMMARY
;
;       Name                            Bytes                           Attributes
;
;  _SMG$CODE                             2176  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         26         0          581        00:01.0
;  _$255$DUA28:[SMGRTL.OBJ]RTLLIB.L32;1              36          5        13            8        00:00.1
;  _$255$DUA28:[SMGRTL.OBJ]SMGLIB.L32;1             469         62        13           38        00:00.5



;                                       COMMAND QUALIFIERS

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

; Size:          2124 code + 52 data bytes
; Run Time:         00:47.2
; Elapsed Time:     02:36.0
; Lines/CPU Min:     2919
; Lexemes/CPU-Min: 18300
; Memory Used:  188 pages
; Compilation Complete
```

SMGKEYUTI
LIS

SMGKEYPAD
LIS

SMGINPUT
LIS

SMGLIB
LIS