

DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL

```

LL          IIIII
LL          IIIII
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LLLLLLLLLLL IIIII
LLLLLLLLLLL IIIII
SSSSSSSSS
SSSSSSSSS
SS
SS
SS
SS
SS
SSSSSS
SSSSSS
SS
SS
SS
SS
SSSSSSSSS
SSSSSSSSS

```

(2) 57
(3) 91
(4) 195
(5) 369
(6) 456

DECLARATIONS
Terminator definitions
GET_KEY_NAME - Translate terminator to key name
List of key names
VALIDATE_KEY_NAME - See if key name is valid

```

0000 1      .TITLE GET_KEY_NAME - Key translation utility procedures
0000 2      .IDENT /V04-000/
0000 3
0000 4
0000 5 :*****
0000 6 :*
0000 7 :*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 :*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 :*  ALL RIGHTS RESERVED.
0000 10 :*
0000 11 :*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 :*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 :*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 :*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 :*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 :*  TRANSFERRED.
0000 17 :*
0000 18 :*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 :*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 :*  CORPORATION.
0000 21 :*
0000 22 :*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 :*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 :*
0000 25 :*
0000 26 :*****
0000 27 :
0000 28
0000 29 :++
0000 30 : FACILITY: DCL
0000 31
0000 32 : ABSTRACT:
0000 33
0000 34 :     Routines to:
0000 35 :         Convert terminator sequences to key names
0000 36 :         Validate a key name
0000 37
0000 38 : ENVIRONMENT: Runs at any access mode, AST Reentrant
0000 39
0000 40 : AUTHOR: Steven B. Lionel, CREATION DATE: 24-Feb-1983
0000 41
0000 42 : MODIFIED BY:
0000 43
0000 44 : 1-004 - HWS0079      Harold Schultz 05-Jul-1984
0000 45 :         Change order in KEY_NAME_LIST to fix HWS0060.
0000 46
0000 47 : 1-003 - HWS0060      Harold Schultz 18-Apr-1984
0000 48 :         Change the common key name for the left VT2xx key pad from
0000 49 :         FIND - NEXT_SCREEN to E1 - E6.
0000 50
0000 51 : 1-002 - SBL1002      Steve Lionel 29-July-1983
0000 52 :         Add E1 - E6 as synonyms for editing keys.
0000 53
0000 54 : 1-001 - Original. SBL 24-Feb-1983
0000 55 :--

```

```

0000 57      .SBTTL  DECLARATIONS
0000 58      :
0000 59      : LIBRARY MACRO CALLS:
0000 60      :
0000 61      :     $SMGDEF                ; Get terminator codes
0000 62      :
0000 63      : EXTERNAL DECLARATIONS:
0000 64      :
0000 65      :     .DSABL  GBL                ; Force all external symbols to be declared
0000 66      :
0000 67      : MACROS:
0000 68      :
0000 69      :     NONE
0000 70      :
0000 71      : EQUATED SYMBOLS:
0000 72      :
0000 73      :
0000001A 0000 74      K_CTRLZ = 26
0000001B 0000 75      K_ESC  = 27
0000008F 0000 76      K_SS3  = 143
0000009B 0000 77      K_CSI  = 155
0000 78      :
0000 79      :
0000 80      : OWN STORAGE:
0000 81      :
0000 82      :     NONE
0000 83      :
0000 84      : PSECT DECLARATIONS:
0000 85      :
0000 86      :
00000000 0000 87      .PSECT _SMG$CODE PIC, USR, CON, REL, LCL, SHR, -
0000 88      EXE, RD, NOWRT, LONG
0000 89

```

```
0000 91      .SBTTL Terminator definitions
0000 92
0000 93      :+
0000 94      : Define macros for table generation
0000 95      :-
0000 96
0000 97      .MACRO TERM1 SEQ, NAME
0000 98      .ASCII SEQ
0000 99      .BYTE SMG$K_TRM_'NAME'-128
0000 100     .ENDM
0000 101
0000 102     .MACRO TERM2 SEQ, NAME
0000 103     .ASCII SEQ
0000 104     .ASCII
0000 105     .BYTE SMG$K_TRM_'NAME'-128
0000 106     .ENDM
0000 107
0000 108     .MACRO TERM3 SEQ, NAME
0000 109     .ASCII SEQ
0000 110     .ASCII
0000 111     .BYTE 0
0000 112     .BYTE SMG$K_TRM_'NAME'-128
0000 113     .ENDM
0000 114
0000 115     :+
0000 116     : Terminators of the form <ESC>x
0000 117     :-
0000 118
0000 119     ESC_1BYTE:
0000 120     TERM1  "A", UP
0000 121     TERM1  "B", DOWN
0000 122     TERM1  "C", RIGHT
0000 123     TERM1  "D", LEFT
0000 124     TERM1  "P", PF1
0000 125     TERM1  "Q", PF2
0000 126     TERM1  "R", PF3
0000 127     TERM1  "S", PF4
00 0010 128     .BYTE 0
0011 129
0011 130     :+
0011 131     : Terminators of the form <CSI>x or <SS3>x
0011 132     :-
0011 133
0011 134     CSI_1BYTE:
0011 135     TERM1  "A", UP
0013 136     TERM1  "B", DOWN
0015 137     TERM1  "C", RIGHT
0017 138     TERM1  "D", LEFT
0019 139     TERM1  "H", ENTER
001B 140     TERM1  "P", PF1
001D 141     TERM1  "Q", PF2
001F 142     TERM1  "R", PF3
0021 143     TERM1  "S", PF4
0023 144     TERM1  "L", COMMA
0025 145     TERM1  "M", MINUS
0027 146     TERM1  "N", PERIOD
0029 147     TERM1  "P", KPO
```

```
002B 148 TERM1 "q" KP1
002D 149 TERM1 "r" KP2
002F 150 TERM1 "s" KP3
0031 151 TERM1 "t" KP4
0033 152 TERM1 "u" KP5
0035 153 TERM1 "v" KP6
0037 154 TERM1 "w" KP7
0039 155 TERM1 "x" KP8
003B 156 TERM1 "y" KP9
00 003D 157 .BYTE 0
003E 158
003E 159 :+ Terminators of the form <CSI>n~
003E 160 :-
003E 161
003E 162 CSI_2BYTE:
003E 164 TERM2 "1" E1
0041 165 TERM2 "2" E2
0044 166 TERM2 "3" E3
0047 167 TERM2 "4" E4
004A 168 TERM2 "5" E5
004D 169 TERM2 "6" E6
0000 0050 170 .WORD 0
0052 171
0052 172 :+ Terminators of the form <CSI>nn~
0052 173 :-
0052 174
0052 175 CSI_3BYTE:
0052 177 TERM3 "17" F6
0057 178 TERM3 "18" F7
005C 179 TERM3 "19" F8
0061 180 TERM3 "20" F9
0066 181 TERM3 "21" F10
006B 182 TERM3 "23" F11
0070 183 TERM3 "24" F12
0075 184 TERM3 "25" F13
007A 185 TERM3 "26" F14
007F 186 TERM3 "28" HELP
0084 187 TERM3 "29" DO
0089 188 TERM3 "31" F17
008E 189 TERM3 "32" F18
0093 190 TERM3 "33" F19
0098 191 TERM3 "34" F20
00000000 009D 192 NULL: .LONG 0
00A1 193
```

; End of list

```
00A1 195 .SBTTL GET_KEY_NAME - Translate terminator to key name
00A1 196 :++
00A1 197 : FUNCTIONAL DESCRIPTION:
00A1 198 :
00A1 199 : GET_KEY_NAME translates a terminator sequence to a key name.
00A1 200 :
00A1 201 : CALLING SEQUENCE:
00A1 202 :
00A1 203 : ret_status = GET_KEY_NAME (terminator.rt.r, term_length.rl.v,
00A1 204 : key_name_addr.wa.r)
00A1 205 :
00A1 206 : FORMAL PARAMETERS:
00A1 207 :
00000004 00A1 208 : terminator = 4 ; The terminator string, passed by reference.
00A1 209 :
00000008 00A1 210 : term_length = 8 ; The length of the terminator string, passed
00A1 211 : by immediate value.
00A1 212 :
0000000C 00A1 213 : key_name_addr = 12 ; A longword into which is placed the address
00A1 214 : of a counted ASCII string containing the
00A1 215 : key name. If no key is found, a pointer
00A1 216 : to an empty string is stored.
00A1 217 :
00A1 218 : IMPLICIT INPUTS:
00A1 219 :
00A1 220 : NONE
00A1 221 :
00A1 222 : IMPLICIT OUTPUTS:
00A1 223 :
00A1 224 : NONE
00A1 225 :
00A1 226 : COMPLETION STATUS:
00A1 227 :
00A1 228 : 1 - Translation successful
00A1 229 : 0 - Translation unsuccessful
00A1 230 :
00A1 231 : SIDE EFFECTS:
00A1 232 :
00A1 233 : NONE
00A1 234 :
00A1 235 : --
000C 00A1 236 :
00A1 237 : .ENTRY GET_KEY_NAME, ^M<R2,R3>
00A3 238 :
52 04 AC 7D 00A3 239 : MOVQ terminator(AP), R2 ; Get terminator address and length
00A7 240 : ; into R2 and R3
50 53 02 C3 00A7 241 : SUBL3 #2, R3, R0 ; Is length within bounds?
00AB 242 : BLSS NOTRANS ; If not, no translation
03 50 D1 00AD 243 : CMPL R0, #3 ; Must be between 2 and 5 chars
00B0 244 : BGTRU NOTRANS ; No translation
00B2 245 :
51 82 9A 00B2 246 : MOVZBL (R2)+, R1 ; Get first character
1B 51 91 00B5 247 : CMPB R1, #K_ESC ; Is the first character <ESC>?
00B8 248 : BNEQ NOT_ESC ; Skip if not
53 D7 00BA 249 : DECL R3 ; Look at next character
51 82 9A 00BC 250 : MOVZBL (R2)+, R1
5B 8F 51 91 00BF 251 : CMPB R1, #A/[ ; <ESC>[ is the same as <CSI>
```

```

4F 8F 3C 13 00C3 252      BEQL  USE_CSI      ; Skip if it is
51 91 00C5 253      CMPB  R1, #^A/O/      ; <ESC>0 is the same as <SS3>
27 13 00C9 254      BEQL  USE_SS3      ; Skip if it is
3F 51 91 00CB 255      CMPB  R1, #^A/?/      ; <ESC>? is <SS3> on VT52s
22 13 00CE 256      BEQL  USE_SS3      ; Skip if it is
00D0 257
00D0 258
00D0 259 :+ We get here if the character following <ESC> does not make it <CSI>
00D0 260 : or <SS3>. It must then be a single character to look up in ESC_1BYTE.
00D0 261 :-
00D0 262
50 53 D7 00D0 263      DECL  R3      ; Look at next character
16 12 00D2 264      BNEQ  NOTRANS      ; If not now zero, no translation
CF 9E 00D4 265      MOVAB W^ESC_1BYTE, R0 ; Get table address
52 51 D0 00D9 266      MOVL  R1, R2      ; Load R2 with character
32 11 00DC 267      BRB  SEARCH_1BYTE ; Skip to table search code
00DE 268
00DE 269 :+
00DE 270 : We get here if the first character was not <ESC>. Check for <CSI> or <SS3>.
00DE 271 :-
00DE 272
00DE 273 NOT_ESC:
9B 8F 51 91 00DE 274      CMPB  R1, #K_CSI      ; Is it <CSI>?
1D 13 00E2 275      BEQL  USE_CSI      ; Skip if it is
8F 8F 51 91 00E4 276      CMPB  R1, #K_SS3      ; Is it <SS3>?
08 13 00E8 277      BEQL  USE_SS3      ; Skip if it is
00EA 278
00EA 279 :+
00EA 280 : We get here if the terminator has no translation
00EA 281 :-
00EA 282
00EA 283 NOTRANS:
0C BC B0 AF 9E 00EA 284      MOVAB B^NULL, @key_name_addr(AP) ; Return address of null string
50 D4 00EF 285      CLRL  R0      ; Return failure
04 00F1 286      RET
00F2 287
00F2 288 :+
00F2 289 : The terminator starts with <SS3>.
00F2 290 :-
00F2 291
00F2 292 USE_SS3:
02 53 D1 00F2 293      CMPL  R3, #2      ; There must be only 1 character
00F5 294      BNEQ  NOTRANS      ; following <SS3>.
F3 12 00F5 295      MOVAB W^CSI_1BYTE, R0 ; No translation if not 1 character
CF 9E 00F7 296      MOVZBL (R2), R2 ; Use same table as <CSI>.
52 62 9A 00FC 297      BRB  SEARCH_1BYTE ; Load R2 with character
OF 11 00FF 298      ; Use next byte only
0101 299
0101 300 :+
0101 301 : The terminator starts with <CSI>.
0101 302 :-
0101 303
0101 304 USE_CSI:
03 53 D1 0101 305      CMPL  R3, #3      ; Are there 1, 2 or >=3 characters
0104 306      BGTR  SEARCH_3BYTE ; after <CSI>?
2E 14 0104 307      BEQL  SEARCH_2BYTE ; >=3 more bytes
16 13 0106 308      ; 2 more bytes
```

```
50  FF05 CF 9E 0108 309      MOVAB  W^CSI_1BYTE, R0      ; 1 more byte
52  52  62  9A 010D 310      MOVZBL (R2), R2      ; Load R2 with character
      0110 311
      0110 312
      0110 313      ;+
      0110 314      ; Use only the next byte to determine the key. R0 has been loaded with the
      0110 315      ; table address and R2 is the byte to look at.
      0110 316      ; -
      0110 317
      0110 318      SEARCH_1BYTE:
51  80  9A 0110 318      MOVZBL (R0)+, R1      ; Get byte to compare against
52  D5  13 0113 319      BEQL  NOTRANS      ; End of table?
      51  91 0115 320      CMPB  R1, R2      ; Compare character
      3A  13 0118 321      BEQL  FOUND      ; End if found
      50  D6 011A 322      INCL  R0      ; Skip over key code
      F2  11 011C 323      BRB  SEARCH_1BYTE      ; Repeat until found or table end
      011E 324
      011E 325      ;+
      011E 326      ; Use the next two bytes to determine the key. (R2) is the word to look at.
      011E 327      ; -
      011E 328
      011E 329      SEARCH_2BYTE:
50  FF1C CF 9E 011E 330      MOVAB  W^CSI_2BYTE, R0      ; Get table address
52  52  62  3C 0123 331      MOVZWL (R2), R2      ; Get two bytes from terminator
51  80  3C 0126 332 10$: MOVZWL (R0)+, R1      ; Get two bytes to compare against
      BF  13 0129 333      BEQL  NOTRANS      ; End of table?
52  51  B1 012B 334      CMPW  R1, R2      ; Compare characters
      24  13 012E 335      BEQL  FOUND      ; End if found
      50  D6 0130 336      INCL  R0      ; Skip over key code
      F2  11 0132 337      BRB  10$      ; Repeat until found or table end
      0134 338
      0134 339      ;+
      0134 340      ; Use the next two bytes to determine the key. (R2) starts the bytes
      0134 341      ; to look at. Note that table CSI_3BYTE uses longword entries with the
      0134 342      ; high byte zero.
      0134 343      ; -
      0134 344
      0134 345      SEARCH_3BYTE:
      04  53 D1 0134 346      CMPL  R3, #4      ; More than 3 bytes after <CSI>?
50  50  FF15 CF 9E 0137 347      BGTR  NOTRANS      ; No translation if so
52  62  18 00 EF 0139 348      MOVAB  W^CSI_3BYTE, R0      ; Get table address
      51  80 D0 0143 349      EXTZV #0, #24, (R2), R2      ; Get next three bytes
      09  13 0146 350 10$: MOVL  (R0)+, R1      ; Get four bytes to compare against
      52  51 D1 0148 351      BEQL  15$      ; End of table?
      07  13 014B 352      CMPL  R1, R2      ; Compare characters
      50  D6 014D 353      BEQL  FOUND      ; End if found
      F2  11 014F 354      INCL  R0      ; Skip over key code
      FF96 31 0151 355      BRB  10$      ; Repeat until found or table end
      0154 356
      0154 357 15$: BRW  NOTRANS
      0154 358
      0154 359      ;+
      0154 360      ; Translation found. Find the name of this key and return success.
      0154 361      ; -
      0154 362
      0154 363      FOUND:
64'AF 0116'8F 60 3A 0154 364      LOCC  (R0), #KEY_NAME_LIST_LEN, B^KEY_NAME_LIST ; Find key name
      OC BC 01 A1 9E 015B 365      MOVAB 1(R1), @key_name_addr(AP) ; Store key name address
```

GET_KEY_NAME
V04=000-

- Key translation utility procedures L 6
GET_KEY_NAME - Translate terminator to k

15-SEP-1984 23:49:09 VAX/VMS Macro V04-00
4-SEP-1984 23:40:43 [DCL.SRC]GETKEYNAM.MAR;1

Page 8
(4)

50 01 D0 0160 366 MOVL #1, R0 ; Return success
04 0163 367 RET

```
0164 369 .SBTTL List of key names
0164 370
0164 371 :+
0164 372 : KEY_NAME_LIST is a list of all possible key names whose codes are in
0164 373 : the range 256-383. This excludes control keys.
0164 374 : The format of this list is:
0164 375 :
0164 376 :         key code-128 - 1 byte
0164 377 :         ASCII key name - n bytes
0164 378 :
0164 379 : This format depends on knowing that no key names with codes higher than
0164 380 : 383 are defined, that no character between 128 and 255 can
0164 381 : appear in key names, and that the maximum length of a key name is <32.
0164 382 :
0164 383 : This list is used in two ways:
0164 384 : 1. To look up a key name (either to see if it is valid or to
0164 385 :    get the corresponding code), do a MATCHC of the ASCII key name
0164 386 :    into KEY_NAME_LIST. The byte preceding the found entry is the
0164 387 :    key code minus 128.
0164 388 :
0164 389 : 2. To convert a key code into a name, do a LOCC of the key code into
0164 390 :    KEY_NAME_LIST. The ASCII key name follows the found entry.
0164 391 : -
0164 392 :
0164 393 : +
0164 394 : Create macro to add an entry to the list.
0164 395 : -
0164 396
0164 397 .MACRO KEY_ENTRY NAME
0164 398 .BYTE <SMG$K_TRM_'NAME' - 128>
0164 399 .ASCII /NAME/
0164 400 .ENDM
0164 401
0164 402 KEY_NAME LIST:
0164 403 KEY_ENTRY PF1
0169 404 KEY_ENTRY PF2
016E 405 KEY_ENTRY PF3
0173 406 KEY_ENTRY PF4
0178 407 KEY_ENTRY KP0
017D 408 KEY_ENTRY KP1
0182 409 KEY_ENTRY KP2
0187 410 KEY_ENTRY KP3
018C 411 KEY_ENTRY KP4
0191 412 KEY_ENTRY KP5
0196 413 KEY_ENTRY KP6
019B 414 KEY_ENTRY KP7
01A0 415 KEY_ENTRY KP8
01A5 416 KEY_ENTRY KP9
01AA 417 KEY_ENTRY ENTER
01B1 418 KEY_ENTRY MINUS
01B8 419 KEY_ENTRY COMMA
01BF 420 KEY_ENTRY PERIOD
01C7 421 KEY_ENTRY UP
01CB 422 KEY_ENTRY DOWN
01D1 423 KEY_ENTRY LEFT
01D7 424 KEY_ENTRY RIGHT
01DE 425 KEY_ENTRY F6
```

GET_KEY_NAME
V04=000

- Key translation utility procedures N 6
List of key names

15-SEP-1984 23:49:09 VAX/VMS Macro V04-00
4-SEP-1984 23:40:43 [DCL.SRC]GETKEYNAM.MAR;1

Page 10
(5)

01E2	426	KEY_ENTRY	F7
01E6	427	KEY_ENTRY	F8
01EA	428	KEY_ENTRY	F9
01EE	429	KEY_ENTRY	F10
01F3	430	KEY_ENTRY	F11
01F8	431	KEY_ENTRY	F12
01FD	432	KEY_ENTRY	F13
0202	433	KEY_ENTRY	F14
0207	434	KEY_ENTRY	HELP
020D	435	KEY_ENTRY	DO
0211	436	KEY_ENTRY	F17
0216	437	KEY_ENTRY	F18
021B	438	KEY_ENTRY	F19
0220	439	KEY_ENTRY	F20
0225	440	KEY_ENTRY	E1
0229	441	KEY_ENTRY	E2
022D	442	KEY_ENTRY	E3
0231	443	KEY_ENTRY	E4
0235	444	KEY_ENTRY	E5
0239	445	KEY_ENTRY	E6
023D	446	KEY_ENTRY	FIND
0243	447	KEY_ENTRY	INSERT_HERE
0250	448	KEY_ENTRY	REMOVE
0258	449	KEY_ENTRY	SELECT
0260	450	KEY_ENTRY	PREV_SCREEN
026D	451	KEY_ENTRY	NEXT_SCREEN
027A	452		
00000116 027A	453	KEY_NAME_LIST_LEN = .-KEY_NAME_LIST	
027A	454		

```
027A 456 .SBTTL VALIDATE_KEY_NAME - See if key name is valid
027A 457 :++
027A 458 : FUNCTIONAL DESCRIPTION:
027A 459 :
027A 460 : VALIDATE_KEY_NAME verifies a key name to see if it is valid.
027A 461 :
027A 462 : CALLING SEQUENCE:
027A 463 :
027A 464 : ret_status.wlc.v = VALIDATE_KEY_NAME (key_name.rt.ds)
027A 465 :
027A 466 :
027A 467 : FORMAL PARAMETERS:
027A 468 :
00000004 027A 469 : key_name = 4 ; The descriptor of a string
027A 470 : containing the name of the key to look up. It
027A 471 : is assumed that the string has been upcased and
027A 472 : stripped of trailing blanks.
027A 473 :
027A 474 : IMPLICIT INPUTS:
027A 475 :
027A 476 : NONE
027A 477 :
027A 478 : IMPLICIT OUTPUTS:
027A 479 :
027A 480 : NONE
027A 481 :
027A 482 : COMPLETION STATUS:
027A 483 :
027A 484 : 1 - Key found
027A 485 : 0 - Key not found
027A 486 :
027A 487 : SIDE EFFECTS:
027A 488 :
027A 489 : NONE
027A 490 :
027A 491 :--
027A 492 :
003C 027A 493 .ENTRY VALIDATE_KEY_NAME , ^M<R2,R3,R4,R5>
50 04 BC 7D 027C 494 MOVQ @key_name(AP), R0 ; Get descriptor into R0-R1
50 50 3C 0280 495 MOVZWL R0, R0 ; Make length a longword
1F 23 13 0283 496 BEQL 10$ ; Exit if zero length
5E 1E 1A 0285 497 CMPL R0, #31 ; Exceeds maximum key name length?
7E 50 C2 0288 498 BGTRU 10$ ; If so, exit with failure
01 AE 61 50 28 028A 499 SUBL2 R0, SP ; Create space for key name on stack
50 50 90 028D 500 MOVB R0, -(SP) ; Move length
5D 5E C3 0290 501 MOVC3 R0, (R1), 1(SP) ; Move key name
FEC2 CF 0116 8F 6E 50 39 0295 502 SUBL3 SP, FP, R0 ; Get string length in R0
02A2 503 MATCHC R0, (SP), #KEY_NAME_LIST_LEN, W^KEY_NAME_LIST
50 04 12 02A2 504 ; Look up key name
50 01 D0 02A4 505 BNEQ 10$ ; Skip if not found
50 04 02A7 507 RET ; Return success
04 D4 02A8 508 10$: CLRL R0 ; Return failure
04 02AA 509 RET
```

GET_KEY_NAME
V04=000

C 7
- Key translation utility procedures
VALIDATE_KEY_NAME - See if key name is v
02AB 511 .END
; End of module GET_KEY_NAME

15-SEP-1984 23:49:09 VAX/VMS Macro V04-00
4-SEP-1984 23:40:43 [DCL.SRC]GETKEYNAM.MAR;1

Page 12
(7)

GET_KEY_NAME
Symbol table

D 7
- Key translation utility procedures

15-SEP-1984 23:49:09 VAX/VMS Macro V04-00
4-SEP-1984 23:40:43 [DCL.SRC]GETKEYNAM.MAR;1

Page 13
(7)

CSI_1BYTE	00000011	R	02	SMGSK_TRM_MINUS	= 0000010F		
CSI_2BYTE	0000003E	R	02	SMGSK_TRM_NEXT_SCREEN	= 0000013C		
CSI_3BYTE	00000052	R	02	SMGSK_TRM_PERIOD	= 00000111		
ESC_1BYTE	00000000	R	02	SMGSK_TRM_PF1	= 00000100		
FOUND	00000154	R	02	SMGSK_TRM_PF2	= 00000101		
GET_KEY_NAME	000000A1	RG	02	SMGSK_TRM_PF3	= 00000102		
KEY_NAME	= 00000004			SMGSK_TRM_PF4	= 00000103		
KEY_NAME_ADDR	= 0000000C			SMGSK_TRM_PREV_SCREEN	= 0000013B		
KEY_NAME_LIST	= 00000164	R	02	SMGSK_TRM_REMOVE	= 00000139		
KEY_NAME_LIST_LEN	= 00000116			SMGSK_TRM_RIGHT	= 00000115		
K_CSI	= 0000009B			SMGSK_TRM_SELECT	= 0000013A		
K_CTRLZ	= 0000001A			SMGSK_TRM_UP	= 00000112		
K_ESC	= 0000001B			TERMINATOR	= 00000004		
K_SS3	= 0000008F			TERM_LENGTH	= 00000008		
NOTRANS	000000EA	R	02	USE_CSI	00000101	R	02
NOT_ESC	000000DE	R	02	USE_SS3	000000F2	R	02
NUL	0000009D	R	02	VALIDATE_KEY_NAME	0000027A	RG	02
SEARCH_1BYTE	00000110	R	02				
SEARCH_2BYTE	0000011E	R	02				
SEARCH_3BYTE	00000134	R	02				
SMGSK_TRM_COMMA	= 00000110						
SMGSK_TRM_DO	= 00000128						
SMGSK_TRM_DOWN	= 00000113						
SMGSK_TRM_E1	= 00000137						
SMGSK_TRM_E2	= 00000138						
SMGSK_TRM_E3	= 00000139						
SMGSK_TRM_E4	= 0000013A						
SMGSK_TRM_E5	= 0000013B						
SMGSK_TRM_E6	= 0000013C						
SMGSK_TRM_ENTER	= 0000010E						
SMGSK_TRM_F10	= 00000122						
SMGSK_TRM_F11	= 00000123						
SMGSK_TRM_F12	= 00000124						
SMGSK_TRM_F13	= 00000125						
SMGSK_TRM_F14	= 00000126						
SMGSK_TRM_F17	= 00000129						
SMGSK_TRM_F18	= 0000012A						
SMGSK_TRM_F19	= 0000012B						
SMGSK_TRM_F20	= 0000012C						
SMGSK_TRM_F6	= 0000011E						
SMGSK_TRM_F7	= 0000011F						
SMGSK_TRM_F8	= 00000120						
SMGSK_TRM_F9	= 00000121						
SMGSK_TRM_FIND	= 00000137						
SMGSK_TRM_HELP	= 00000127						
SMGSK_TRM_INSERT_HERE	= 00000138						
SMGSK_TRM_KP0	= 00000104						
SMGSK_TRM_KP1	= 00000105						
SMGSK_TRM_KP2	= 00000106						
SMGSK_TRM_KP3	= 00000107						
SMGSK_TRM_KP4	= 00000108						
SMGSK_TRM_KP5	= 00000109						
SMGSK_TRM_KP6	= 0000010A						
SMGSK_TRM_KP7	= 0000010B						
SMGSK_TRM_KP8	= 0000010C						
SMGSK_TRM_KP9	= 0000010D						
SMGSK_TRM_LEFT	= 00000114						

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR
_SMG\$CODE	000002AB (683.)	02 (2.)	PIC USR

CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE
CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC	LONG

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	14	00:00:00.04	00:00:01.77
Command processing	136	00:00:00.74	00:00:07.20
Pass 1	153	00:00:03.97	00:00:15.08
Symbol table sort	0	00:00:00.25	00:00:00.76
Pass 2	84	00:00:01.25	00:00:04.77
Symbol table output	8	00:00:00.09	00:00:00.57
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	397	00:00:06.36	00:00:30.17

The working set limit was 1050 pages.
21071 bytes (42 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 228 non-local and 4 local symbols.
511 source lines were read in Pass 1, producing 20 object records in Pass 2.
12 pages of virtual memory were used to define 11 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[SYSLIB]SYSBLDMLB.MLB;1	0
-\$255\$DUA28:[DCL.OBJ]DCL.MLB;1	0
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	0
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	4

262 GETS were required to define 4 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:GETKEYNAM/OBJ=OBJ\$:GETKEYNAM MSRC\$:GETKEYNAM/UPDATE=(ENH\$:GETKEYNAM)+EXECML\$/LIB+LIB\$:DCL/LIB+SYSS\$LIBRARY:SYSBLDMLB/L

0070

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY