

```

*****
' *                                     KEYB                                     *
'-----
' * Task      : Makes a function available which reads a character from *
' *            the keyboard. The status of the INSERT, CAPS LOCK and *
' *            NUM LOCK keys is displayed on the screen. *
'-----
' * Author      : Michael Tischer *
' * Developed on : 06/09/91 *
' * Last update  : 01/20/92 *
*****
'
DECLARE SUB PrintInverse (Text AS STRING)
DECLARE SUB WriteChar (KChar AS INTEGER, CColor AS INTEGER)
DECLARE FUNCTION GetPage% ()
DECLARE SUB Inikey ()
DECLARE FUNCTION NegFlag% (Flag%, FlagReg%, Col%, Row%, Text AS STRING)
DECLARE FUNCTION GetKey% ()

'$INCLUDE: 'QB.BI'          'Include file contains register declarations

CONST TRUE = -1              'Define the truth
CONST FALSE = NOT TRUE

CONST FZ = 1                 'Display row in the flag
CONST FS = 65                'Display column in the flag

'-- BIOS keyboard status bits -----
CONST SCRL = 16              'SCROLL LOCK bit
CONST NUML = 32              'NUM LOCK bit
CONST CAPL = 64              'CAPS LOCK bit
CONST INS = 128              'INSERT bit

'-- Codes of some keys, returned (e.g., by GETKEY) -----
CONST BEL = 7                'Bell character code
CONST BS = 8                 'Backspace key code
CONST TB = 9                 'Tab key code
CONST LF = 10                'Linefeed key code
CONST CR = 13                'Carriage return code
CONST ESC = 27               'Escape key code
CONST F1 = 315               'Function keys
CONST F2 = 316
CONST F3 = 317
CONST F4 = 318
CONST F5 = 319
CONST F6 = 320
CONST F7 = 321
CONST F8 = 322
CONST F9 = 323
CONST F10 = 324
CONST CUP = 328              'Cursor up
CONST CLEFT = 331            'Cursor left
CONST CRIGHT = 333           'Cursor right
CONST CDOWN = 329            'Cursor down

'-- Global variables -----
DIM SHARED Insert AS INTEGER 'INSERT key status
DIM SHARED Caps AS INTEGER  'CAPS LOCK key status
DIM SHARED Num AS INTEGER    'NUM LOCK key status

'-- Main program -----
DIM CKey AS INTEGER          'Get ASCII code for a key

Inikey                      'Initialize keyboard flag
CLS                          'Clear screen
PRINT "KEYB (C) 1987, 91 by MICHAEL TISCHER"
PRINT
PRINT "You can type some characters and change the status of the";
PRINT "INSERT, CAPS and NUM modes. The upper-right corner of the"
PRINT "screen documents the changes to these keys.";
PRINT "Press <Enter> or <F1> to end the program. ";
PRINT : PRINT "Type text here: ";
DO
    'Input loop

```

```

CKey = GetKey                                     'Read key
IF CKey < 256 THEN                                'No extended key code?
    PRINT (CHR$(CKey));                          'Display character
END IF
LOOP UNTIL (CKey = CR) OR (CKey = F1)             'Repeat until the user
                                                ' presses <F1> or <CR>
END

'*****
'* GetKey : Reads a character and displays flag status. *
'* Input : None *
'* Output : Code of captured key: *
'* < 256 : Normal key code. *
'* >= 256 : Extended key code. *
'*****
'
FUNCTION GetKey%

DIM Reg AS RegType                                'Processor registers for interrupt call

DO
    Reg.ax = &H200                                'Read function number for keyboard status
    CALL INTERRUPT(&H16, Reg, Reg)                'Call BIOS keyboard interrupt

    '-- Pass new flag status -----
    Insert = NegFlag(Insert, ((Reg.ax MOD 256) AND INS), FS + 9, FZ, "INS")
    Caps = NegFlag(Caps, ((Reg.ax MOD 256) AND CAPL), FS + 3, FZ, "CAPS ")
    Num = NegFlag(Num, ((Reg.ax MOD 256) AND NUML), FS, FZ, "NUM")

    Reg.ax = &H100                                'Character function number ready?
    CALL INTERRUPT(&H16, Reg, Reg)                'Call BIOS keyboard interrupt
    IF (Reg.flags AND 64) = 0 THEN                 'Keys are ready
        Reg.ax = &H0                                'Read function number for key
        CALL INTERRUPT(&H16, Reg, Reg)            'Call BIOS keyboard interrupt
        IF (Reg.ax MOD 256) = 0 THEN               'Is it an extended key?
            GetKey = (Reg.ax \ 256) OR &H100        'Yes --> An extended key
        ELSE                                       'No --> Not an extended key
            GetKey = (Reg.ax MOD 256)
        END IF
    END IF
    EXIT DO                                        'End loop
END IF
LOOP                                              'Repeat until key captured
END FUNCTION

'*****
'* GetPage : Gets the current page. *
'* Input : None *
'* Output : The current screen page. *
'*****
'
FUNCTION GetPage%

DIM Register AS RegType                            'Processor registers for interrupt call

Register.ax = &H1500                                'AH = Function number
CALL INTERRUPT(&H10, Register, Register)           'Call BIOS interrupt
GetPage = Register.bx \ 256                        'Register.BH returns screen page

END FUNCTION

'*****
'* IniKey : Initializes the keyboard flags. *
'* Input : None *
'* Output : None *
'* Info : The keyboard flags change when pressed or released, *
'* based on information passed by the GetKey function. *
'*****
'
SUB Inikey

SHARED Insert AS INTEGER                            'INSERT key status
SHARED Caps AS INTEGER                             'CAPS LOCK key status
SHARED Num AS INTEGER                              'NUM LOCK key status

DIM Register AS RegType                            'Processor registers for interrupt call

```

```

Register.ax = &H200          'Read function number for keyboard status
CALL INTERRUPT(&H16, Register, Register) 'Call BIOS keyboard interrupt

IF (Register.ax AND INS) THEN          'Set INSERT flag
    Insert = FALSE
ELSE
    Insert = TRUE
END IF

IF (Register.ax AND CAPL) THEN          'Set CAPS LOCK flag
    Caps = FALSE
ELSE
    Caps = TRUE
END IF

IF (Register.ax AND NUML) THEN          'Set NUM LOCK flag
    Num = FALSE
ELSE
    Num = TRUE
END IF

END SUB

'*****
'* NegFlag : Negates flag and displays corresponding text.      *
'* Input   : See below                                          *
'* Output  : New flag status (True = on, False = off ).        *
'*****
'
FUNCTION NegFlag% (Flag%, FlagReg%, Col%, Row%, Text AS STRING)

DIM CurRow AS INTEGER          'Stores the current cursor position
DIM CurCol AS INTEGER          'Stores the current cursor position

'-- Test for change in status -----

IF Flag% AND (FlagReg% = 0) OR (NOT Flag%) AND (FlagReg% <> 0) THEN
    CurRow = CSRLIN             'Yes: Set current row
    CurCol = POS(0)             'Set current column
    LOCATE Row%, Col%           'Cursor in position for flag names
    IF FlagReg% = 0 THEN         'If flag is out
        NegFlag = FALSE         'Result of function: Flag off
        PRINT SPACE$(LEN(Text)) 'Clear flag display
    ELSE                         'Flag is on
        NegFlag% = TRUE         'Result of function: Flag on
        PrintInverse (Text)     'Display flag name
    END IF
    LOCATE CurRow, CurCol       'Restore old cursor position
ELSE                             'Unchanged status
    NegFlag% = Flag%           'Return remaining flag status
END IF

END FUNCTION

'*****
'* PrintInverse: Writes inverse video string to the current screen *
'*              page, in the current position.                     *
'* Input       : See below                                          *
'* Output      : None                                              *
'*****
SUB PrintInverse (Text AS STRING)

CONST INVERS = &H70             'Attribute byte for inverse display
DIM Counter AS INTEGER          'Loop counter

FOR Counter = 1 TO LEN(Text)    'Display all text characters
    CALL WriteChar(ASC(MID$(Text, Counter, 1)), INVERS)
    LOCATE CSRLIN, POS(0) + 1    'Indent cursor one position
NEXT

END SUB

'*****
'* WriteChar : Writes a character with specified attributes to the *
'*            current screen page, in the current position.        *
'* Input     : See below                                          *

```

```

' *   Output       : None                                     *
' ****
'
SUB WriteChar (KChar AS INTEGER, CColor AS INTEGER)

DIM Register AS RegType          'Processor registers for interrupt call

Register.ax = &H9 * 256 + KChar    'AH = Funct. No: Write character
                                   'AL = ASCII code of character
Register.bx = GetPage * 256 + CColor 'BH = Curr. screen pg; BL = CColor
Register.cx = 1                    'Display characters once
CALL INTERRUPT(&H10, Register, Register) 'Call BIOS interrupt

END SUB

```