

```

/*****
/*
/*----- K E Y C -----*/
/* Task : Makes a function available for reading a character */
/* from the keyboard. The upper-right corner of the */
/* screen lists the status of INSERT, CAPS LOCK and */
/* NUM LOCK. */
/*-----*/
/* Author : Michael Tischer */
/* Developed on : 08/13/87 */
/* Last update : 04/07/95 */
/*-----*/
/* Memory model : SMALL */
*****/

/*== Add include files =====*/

#include <dos.h>
#include <bios.h>
#include <stdio.h>

/*== Type definitions =====*/

typedef unsigned char BYTE; /* Create a byte */

/*== Macros =====*/

#ifdef __TURBOC__ /* Definitions for TURBO C */

#define GetKbKey() ( bioskey( 0 ) )
#define GetKbReady() ( bioskey( 1 ) != 0 )
#define GetKbStatus() ( bioskey( 2 ) )

#else /* Definitions for Microsoft C Compiler */

#define GetKbKey() ( _bios_keybrd( _KEYBRD_READ ) )
#define GetKbReady() ( _bios_keybrd( _KEYBRD_READY ) != 0 )
#define GetKbStatus() ( _bios_keybrd( _KEYBRD_SHIFTSTATUS ) )

#endif

/*== Constants =====*/

/*-- Bit layout in BIOS keyboard status -----*/

#define SCRL 16 /* SCROLL LOCK bit */
#define NUML 32 /* NUM LOCK bit */
#define CAPL 64 /* CAPS LOCK bit */
#define INS 128 /* INSERT bit */

#define TRUE ( 0 == 0 ) /* Constants make reading the */
#define FALSE ( 0 == 1 ) /* program code easier */

#define FR 0 /* Row in which the flags should be displayed */
#define FC 65 /* Column in which the flags should be displayed */
#define FlagColour 0x70 /* Black characters on white background */

/*-- Codes of some keys as returned by GETKEY() -----*/
#define BEL 7 /* Bell character code */
#define BS 8 /* Backspace key code */
#define TAB 9 /* Tab key code */
#define LF 10 /* Linefeed key code */
#define CR 13 /* Carriage return code */
#define ESC 27 /* Escape key code */
#define F1 315 /* Function keys */
#define F2 316
#define F3 317
#define F4 318
#define F5 319
#define F6 320
#define F7 321
#define F8 322
#define F9 323
#define F10 324
#define CUP 328 /* Cursor keys */

```

```

#define CLEFT      331
#define CRIGHT     333
#define CDOWN      328

/== Global variables =====*/

BYTE Insert,          /* INSERT flag status */
    Num,              /* NUM flag status */
    Caps;             /* CAPS flag status */

/*****
/* GETPAGE : Gets the current screen page.
/* Input   : None
/* Output  : See below
*****/

BYTE GetPage( void )
{
    union REGS Register; /* Register variables for interrupt call */

    Register.h.ah = 15; /* Function number */
    int86(0x10, &Register, &Register); /* Call interrupt 10H */
    return(Register.h.bh); /* Current screen page number */
}

/*****
/* SETPOS: Sets the cursor position in the current screen page.
/* Input   : ScCol = New cursor column
/*          : ScRow = New cursor row
/* Output  : None
/* Info    : The screen cursor may change when you call this
/*          : function if the current screen page is the specified
/*          : screen page.
*****/

void SetPos(BYTE ScCol, BYTE ScRow)
{
    union REGS Register; /* Register variables for interrupt call */

    Register.h.ah = 2; /* Function number */
    Register.h.bh = GetPage(); /* Screen page */
    Register.h.dh = ScRow; /* Screen row */
    Register.h.dl = ScCol; /* Screen column */
    int86(0x10, &Register, &Register); /* Call interrupt 10H */
}

/*****
/* GETPOS: Gets the cursor position in the current screen page.
/* Input   : None
/* Output  : - ScCol = Pointer to variable containing current column
/*          : - ScRow = Pointer to variable containing current row
*****/

void GetPos(BYTE * ScCol, BYTE * ScRow)
{
    union REGS Register; /* Register variables for interrupt call */

    Register.h.ah = 3; /* Function number */
    Register.h.bh = GetPage(); /* Screen page */
    int86(0x10, &Register, &Register); /* Call interrupt 10H */
    *ScCol = Register.h.dl; /* Read function result */
    *ScRow = Register.h.dh; /* from the registers */
}

/*****
/* WRITECHAR: Writes a character with attribute to the current
/*            cursor position in the current screen page.
/* Input     : - CharCode = ASCII code of character to be displayed
/*            : - CAttr   = Character attribute
/* Output    : None
*****/

void WriteChar(char CharCode, BYTE CAttr)
{
    union REGS Register; /* Register variables for interrupt call */

```

```

Register.h.ah = 9;                                /* Function number */
Register.h.al = CharCode;                          /* Character to be displayed */
Register.h.bh = GetPage();                         /* Screen page */
Register.h.bl = CAttr;                            /* Color of char. to be displayed */
Register.x.cx = 1;                                /* Display character once */
int86(0x10, &Register, &Register);               /* Call interrupt 10H */
}

```

```

/*****
/* WRITETEXT: Writes a character with constant color to a specific
/*              position within the current screen page.
/* Input      : - ScCol = Display column
/*              - ScRow = Display row
/*              - TEXT  = Pointer to the string to be displayed
/*              - CAttr = Attribute for character to be displayed
/* Output     : None
/* Info      : Text is a pointer to a character vector, which contains
/*              the text to be displayed, terminated with '\0'.
*****/

```

```

void WriteText(BYTE ScCol, BYTE ScRow, char *Text, BYTE CAttr)
{
    union REGS InRegister,      /* Register variables for interrupt call */
              OutRegister;

    SetPos(ScCol, ScRow);       /* Place cursor */
    InRegister.h.ah = 14;       /* Function number */
    InRegister.h.bh = GetPage(); /* Screen page */
    while (*Text)               /* Display text until '\0' */
    {
        WriteChar(' ', CAttr); /* Write character color */
        InRegister.h.al = *Text++; /* for character */
        int86(0x10, &InRegister, &OutRegister); /* Call interrupt */
    }
}

```

```

/*****
/* CLS: Clear current screen page
/* Input      : None
/* Output     : None
*****/

```

```

void Cls( void )
{
    union REGS Register; /* Register variables for interrupt call */

    Register.h.ah = 6; /* Function number: Scroll Up */
    Register.h.al = 0; /* 0 = Clear */
    Register.h.bh = 7; /* White text on black background */
    Register.x.cx = 0; /* Upper-left corner of screen */
    Register.h.dh = 24; /* Coordinates of lower- */
    Register.h.dl = 79; /* right corner of screen */
    int86(0x10, &Register, &Register); /* Call BIOS video interrupt */
}

```

```

/*****
/* NEGFLAG: Negates flag and displays corresponding text.
/* Input      : FLAG = Last flag status
/*              FLAGREG = Current flag status (0 = off)
/*              ScCol = Screen column for flag names
/*              ScRow = Screen row for flag names
/*              TEXT = Flag name
/* Output     : New flag status (TRUE = on, FALSE = off)
*****/

```

```

BYTE NegFlag(BYTE Flag, unsigned int FlagReg,
              BYTE ScCol, BYTE ScRow, char * Text)
{
    BYTE CurScRow, /* Current row */
        CurScCol; /* Current column */

    if (!(Flag == (FlagReg != 0))) /* Has flag been changed? */
    {
        GetPos(&CurScCol, &CurScRow); /* Get current cursor position */
        WriteText(ScCol, ScRow, Text, (BYTE) ((Flag) ? 0 : FlagColour));
        SetPos(CurScCol, CurScRow); /* Set old cursor position */
    }
}

```

```

    return(Flag ^1);                                /* Change flag bit 0 */
}
else return(Flag);                                /* Change back to old status */
}

/*****
/* GETKEY: Reads a character and displays flag status.
/* Input      : None
/* Output     : Code of captured key:
/*              < 256 : normal key
/*              >= 256 : extended keycode
*****/

unsigned int GetKey( void )
{
    int RcKey,                                     /* The received key */
        Status;                                   /* Keyboard status */

    do
    {
        Status = GetKbStatus();                    /* Read keyboard status */
        Insert = NegFlag(Insert, Status & INS, FC+9, FR, "INSERT");
        Caps = NegFlag(Caps, Status & CAPL, FC+3, FR, "CAPS ");
        Num = NegFlag(Num, Status & NUML, FC, FR, "NUM");
    }
    while ( !GetKbReady() );                        /* Repeat until key is ready */

    RcKey = GetKbKey();                             /* Get key */
    return ((RcKey & 255) == 0) ? (RcKey >> 8) + 256 : RcKey & 255;
}

/*****
/* INIKEY: Initializes keyboard flags.
/* Input      : None
/* Output     : None
/* Info       : The keyboard flags change to their opposite status.
/*              The GETKEY function then changes them to their current
/*              status on the next call.
*****/

void IniKey( void )
{
    int Status;                                    /* Keyboard status */

    Status = GetKbStatus();                        /* Read keyboard status */
    Insert = (Status & INS) ? FALSE : TRUE;         /* Change current */
    Caps = (Status & CAPL) ? FALSE : TRUE;          /*      status */
    Num = (Status & NUML) ? FALSE : TRUE;           /*      as needed */
}

/*****
**                               MAIN PROGRAM                               **
*****/

void main( void )
{
    unsigned int AKey;

    Cls();                                          /* Clear screen */
    SetPos(0,0);                                  /* Move cursor to upper-left corner of screen */
    printf("KEYC - (c) 1987, 92 by Michael Tischer\n\n");
    printf("You can type some character and change the status of the\n");
    printf("INSERT, CAPS and NUM modes. The upper-right corner of the\n");
    printf("screen documents the changes to these keys.\n");
    printf("Press <Enter> or <F1> to end the program. \n\n");
    printf("Type text here: ");
    IniKey();                                      /* Initialize keyboard flags */
    do
    {
        if ((AKey = GetKey()) < 256)                /* Read the key */
            printf("%c", (char) AKey);              /* Display (if normal) */
    }
    while (!(AKey == CR || AKey == F1));            /* Repeat until user presses */
                                                    /* <Enter> or <F1> */
    printf("\n");
}

```