

Pascal listing: KEYP.PAS

```
{*****}
{*               K E Y P               *}
{*-----*}
{* 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   : 07/08/87                                        *}
{* Last update   : 04/07/95                                        *}
{*****}
```

program KEYP;

Uses Crt,Dos; { Add Crt and Dos units }

{ \$V- } { Suppresses string length check }

type FlagText = string[6]; { Used for passing the flag name }

```
const FR      = 1; { Row in which flags are displayed }
      FC      = 65; { Column in which flags are displayed }
      FlagFore = 0; { Foreground color of flags }
      FlagBck  = 7; { Background color of flags }
```

```
{** BIOS keyboard status bits *****}
SCRL = 16; { SCROLL LOCK bit }
NUML = 32; { NUM LOCK bit }
```

```

CAPL = 64;                                { CAPS LOCK bit }
INS = 128;                                { INSERT bit }
{** Codes of some keys as presented by GETKEY *****}
BEL      = 7;                             { Bell character code }
BS       = 8;                             { Backspace character code }
TAB      = 9;                             { Tab character code }
LF       = 10;                            { Linefeed code }
CR       = 13;                            { Carriage return code }
ESC      = 27;                            { Escape character code }
F1       = 315;                           { F1 key }
F2       = 316;                           { F2 key }
F3       = 317;                           { F3 key }
F4       = 318;                           { F4 key }
F5       = 319;                           { F5 key }
F6       = 320;                           { F6 key }
F7       = 321;                           { F7 key }
F8       = 322;                           { F8 key }
F9       = 323;                           { F9 key }
F10      = 324;                           { F10 key }
CUP      = 328;                           { Cursor up }
CLEFT    = 331;                           { Cursor left }
CRIGHT   = 333;                           { Cursor right }
CDOWN    = 328;                           { Cursor down }

var Insert,                               { INSERT flag status }
    Num,                                  { NUM flag status }
    Caps      : boolean;                  { CAPS flag status }
    ForeColor,                               { Current foreground color }
    BckColor,                               { Current background color }
    key       : integer;                  { Code of key read }

{ ***** }

```

```

{* NEFLAG: Negates flag and displays text.                                     *}
{* Input : See below                                                         *}
{* Output : The new flag status (TRUE = on, FALSE = off).                   *}
{*****}

function NegFlag(Flag      : boolean;                                     { The last flag status }
                 FlagReg,   { Current flag status (0 = off) }
                 Column,    { Column for flag names }
                 Crow       : integer;                                { Row for flag names }
                 Text       : FlagText) : boolean;                    { Flag names }

var CurCrow,                                           { Current row }
    CurColumn : integer;                               { Current column }

begin
  if (Flag and (FlagReg = 0)) or                      { Test for change }
    (not(Flag) and (FlagReg <> 0)) then                { to flag status }
  begin                                              { Yes }
    CurCrow := WhereY;                               { Store current row }
    CurColumn := WhereX;                             { Store current column }
    gotoxy(Column, Crow);                           { Cursor to flag name position }
    if FlagReg = 0 then                             { Is flag reset? }
    begin                                           { Yes }
      NegFlag := false;                            { Result of the function : Flag off }
      textcolor(Black);                             { Foreground color is black }
      textbackground(Black);                        { Background color is black }
    end
  else
    begin                                           { Flag is now on }
      NegFlag:=true;                                { Result of the function : Flag on }
      textcolor(FlagFore);                          { Foreground color is FLAGFORE }
      textbackground(FlagBck);                      { Background color is FLAGBCK }
    end
  end
end

```

```

    end;
    write(Text);
    gotoxy(CurColumn, CurCrow);
    textcolor(ForeColor);
    textbackground(BckColor)
end
else
    NegFlag := Flag
end;

```

```

{*****}
{* GETKEY: Reads a character and displays flag status. *}
{* Input : None *}
{* Output : Key code - < 256 : Normal key. *}
{* >= 256 : Extended key. *}
{*****}

```

```
function Getkey : integer;
```

```

var Regs : Registers;
    keyRec : boolean;

```

```

begin
    keyRec := false;
    repeat
        Regs.ah := $2;
        intr($16, Regs);
    until keyRec;
    {** Adjust flags to new status ****}
    Insert := NegFlag(Insert, Regs.al and INS, FC+9, FR, 'INSERT');
    Caps := NegFlag(Caps, Regs.al and CAPL, FC+3, FR, 'CAPS ');
    Num := NegFlag(Num, Regs.al and NUML, FC, FR, 'NUM');

```

```

Regs.ah := $1;                { Function number: Character ready? }
intr($16, Regs);              { Call BIOS keyboard interrupt }
if (Regs.flags and FZero = 0) then
  begin
    KeyRec := true;
    Regs.ah := 0;
    intr($16, Regs);
    if (Regs.al = 0)           { Is zero flag set? }
      then Getkey := Regs.ah or $100           { Yes }
      else Getkey := Regs.al;                 { No }
    end;
  until keyRec;                { Repeat until a key is received }
end;

```

```

{*****}
{ * INIKEY: Initializes keyboard flags. * }
{ * Input : None * }
{ * Output : None * }
{ * Info : The keyboard flags are inverted from the current status. * }
{ * The next call to GETKEY displays current flag status. * }
{*****}

```

procedure Inikey;

```

var Regs : Registers;          { Register variable for interrupt call }

begin
  Regs.ah := $2;               { Read function number for keyboard status }
  intr($16, Regs);             { call BIOS keyboard interrupt }
  if (Regs.al and INS <> 0) then Insert := false      { INSERT flag }
    else Insert := true;      { set }
  if (Regs.al and CAPL <> 0) then Caps := false      { CAPS flag }

```

```

                                else Caps    := true;           { set }
if (Regs.al and NUML <> 0) then Num    := false           { NUM flag }
                                else Num      := true           { set }
end;
```

```

{ ***** }
{ * SCOLOR: Sets foreground and background colors for display. * }
{ * Input  : See below * }
{ * Output : None * }
{ * Var.   : Color is stored in the FORECOLOR and BCKCOLOR * }
{ *         : global variables. * }
{ * Info   : This procedure must be called for setting the color * }
{ *         : so that after the output of the keyboard flag status, * }
{ *         : the current text color can be restored, * }
{ *         : since in TURBO no functions exist for sensing * }
{ *         : this color. * }
{ ***** }
```

```
procedure Scolor(Foreground, Background : integer);
```

```

begin
  ForeColor := Foreground;           { Store foreground color }
  BckColor  := Background;           { Store background color }
  textcolor(Foreground);             { Store foreground color }
  textbackground(Background)         { Store background color }
end;
```

```

{ ***** }
{ *                               MAIN PROGRAM                               * }
{ ***** }
```

```
begin
```

```

Inikey;                                { Initialize keyboard flags }
Scolor(7,0);                           { Color is white on black }
clrscr;                                { Clear screen }
writeln(#13#10'KEYP (c) 1987, 92 by Michael Tischer');
writeln(#13#10'You can type some characters and change the status');
writeln('of the INSERT, CAPS and NUM modes. The upper-right corner');
writeln('of the screen documents the changes to these keys.');
```

writeln('Press <Enter> or <F1> to end the program.');	
write(#13#10'Type text here: ');	
repeat	{ Input loop }
key := Getkey;	{ Get key }
if (key < 256) then write(chr(key))	{ Output (if normal) }
until (key = 13) or (key = F1);	{ Repeat until F1 or CR }
writeln;	
end.	