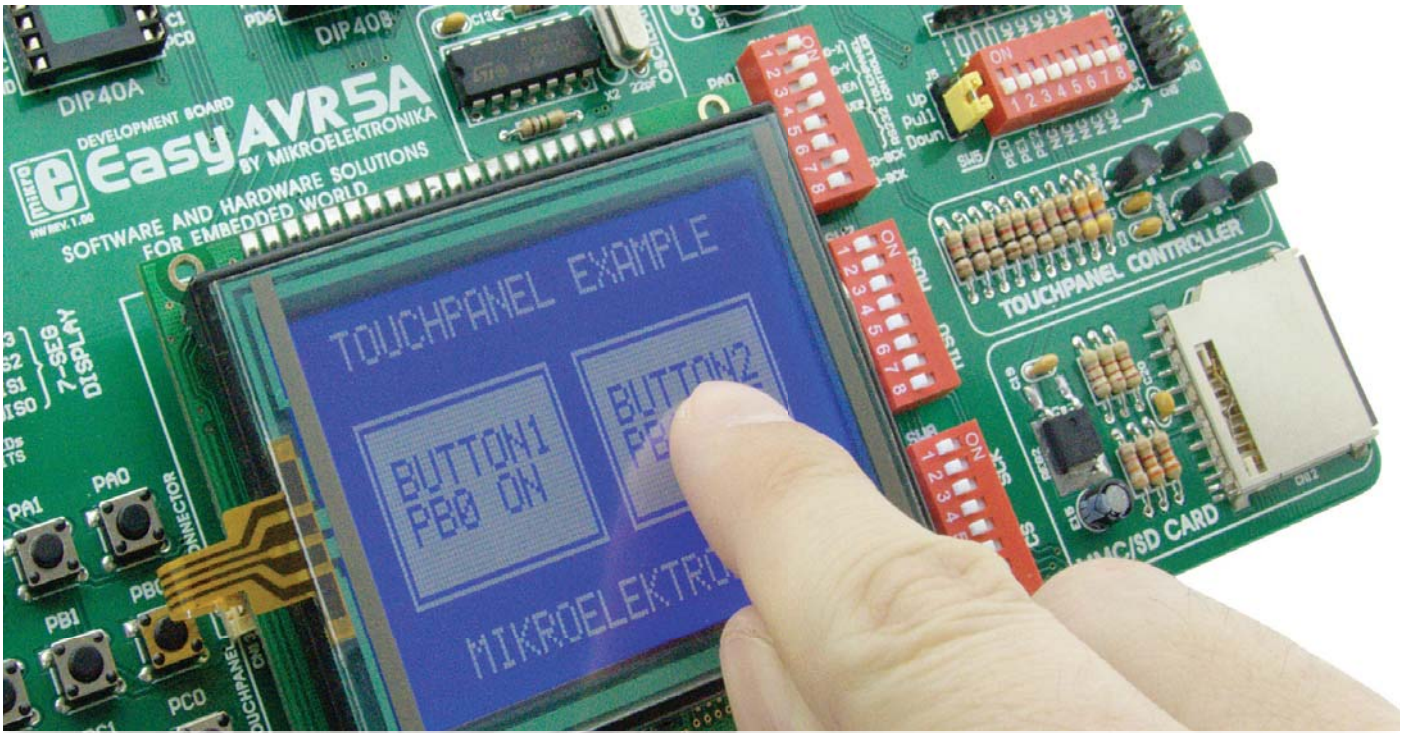


OK. Now you need a... TOUCHSCREEN



By Dusan Mihajlovic

Mikroelektronika Hardware Department

Do you want your new device to have a simple and intuitive interface? If the answer is YES, then a graphic LCD display with touch panel is the best choice because together they create a Touchscreen (Glcd + Touch Panel = Touchscreen). In that way, with a small number of electronic components you will be able to create an attractive and easy to use device.

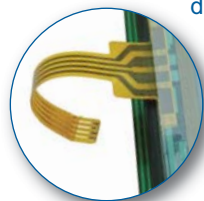
What is a touch panel? A touch panel is a thin, self-adhesive transparent panel placed over the screen of a graphic LCD. It is very sensitive to pressure so that even a soft touch causes some changes on output signal. There are a few types of touch panel. The simplest one is the resistive touch panel which will be discussed here.

Principle of operation

A resistive touch panel consists of two transparent rigid foils, forming a "sandwich" structure, that have resistive layers on their inner sides. The resistance of these layers usually does not exceed 1Kohm. The opposite sides of the foils have contacts available for use through a flat cable. The process of determining coordinates of the point in which the touch panel is pressed can be broken up into two steps. The first one is the determination of the X coordinate and the second one is the determination of the Y coordinate of the point. In order to determine the X coordinate, it is necessary to connect the left contact on the X surface to ground and the right contact to the power supply. This enables a voltage divider to be obtained by pressing the touch panel. The val-

ue of the divider is read on the bottom contact of the Y surface. Voltage can be in the range of 0V to the power supply and depends on the X coordinate. If the point is closer to the left contact of the X surface, the voltage will be closer to 0V. In order to determine the Y coordinate, it is necessary to connect the bottom contact on the Y surface to ground, and the upper contact to power supply.

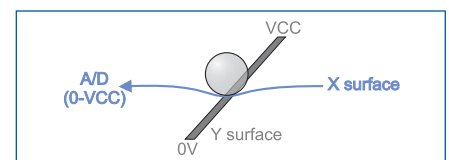
Flat cable detail



In this case, the voltage is read on the left contact of the X surface.

Connecting to microcontroller

In order to connect a touch panel to the microcontroller it is necessary to create a circuit for touch panel control. By means of this circuit, the microcontroller connects appropriate contacts of the touch panel to ground and the power supply (as described above) in order to determine the X



Determination of Y coordinate

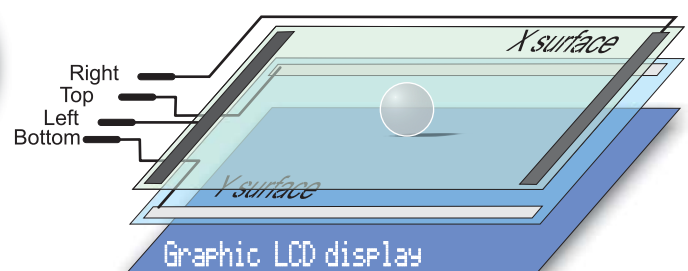
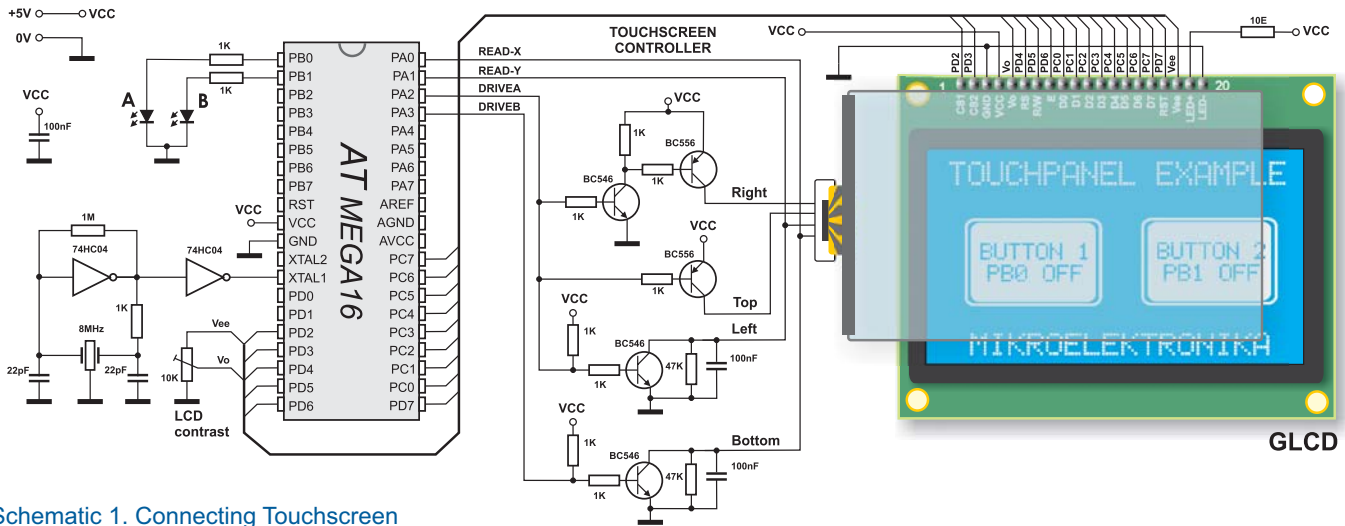
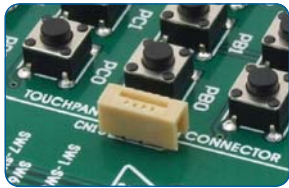


Figure 1. Touch panel internal structure

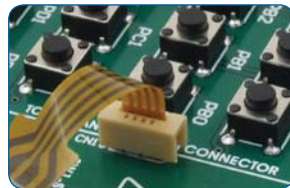


Schematic 1. Connecting Touchscreen

and Y coordinates (Refer to Schematic 1). The bottom contact of the Y surface and left contact of the X surface are connected to the microcontroller's A/D converter. The X and Y coordinates are determined by measuring voltage on these contacts, respectively. The software consists of writing a menu on graphic LCD, turning the circuit for touch panel control on/off (driving touch panel) and reading the values of A/D converter which actually represent the X and Y coordinates of the point. Once the coordinates are determined, it is possible to decide what we want the microcontroller to do. For the purpose of illustration, let us examine Example 1. It explains how to turn on/off two digital microcontroller pins, connected to LED diodes A and B, using a display and a touch panel.



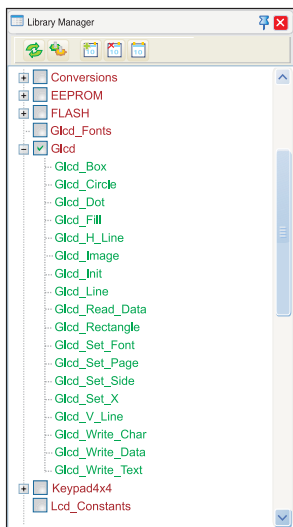
Flat cable on-board connector before...



...and after connecting touch panel.

Considering that the touch panel surface is slightly larger than the surface of the graphic LCD, in case you want greater accuracy when determining the coordinates, it is necessary to perform the software calibration of the touch panel.

Functions used in the program



- ADC_Read() Read analog value
 - Delay_ms() Delay
 - Glcd_box() Draw filled box*
 - Glcd_circle() Draw circle
 - Glcd_Dot() Draw dot
 - Glcd_Fill() Delete/fill display*
 - Glcd_H_Line() Draw horizontal line
 - Glcd_Image() Import image
 - Glcd_Init() LCD display initialization*
 - Glcd_Line() Draw line
 - Glcd_Read_Data() Read data from LCD
 - Glcd_Rectangle() Draw rectangle*
 - Glcd_Set_Font() Select font*
 - Glcd_Set_Page() Select page
 - Glcd_Set_Side() Select side of display
 - Glcd_Set_X() Determine X coordinate
 - Glcd_V_Line() Draw vertical line
 - Glcd_Write_Char() Write character
 - Glcd_Write_Data() Write data
 - Glcd_Write_Text() Write text*
- * Glcd library functions used in the program

mikroPASCAL PRO for AVR® library editor with ready to use libraries such as: Ethernet, CAN, SD/MMC etc.

NOTE: Code for this example written for AVR® microcontrollers in C, Basic and Pascal as well as the programs written for PIC® and dsPIC® microcontrollers can be found on our web site www.mikroe.com/en/article/

Example 1: Program to demonstrate touchscreen operation

```

program TouchPanel;
var GLCD_DataPort : byte at PORTC;    GLCD_DataPort_Direction : byte at DDRC; // Glcd module connections
var GLCD_CS1 : sbit at PORTD.B2;    GLCD_CS1_Direction : sbit at DDRD.B2;
var GLCD_CS2 : sbit at PORTD.B3;    GLCD_CS2_Direction : sbit at DDRD.B3;
var GLCD_RS : sbit at PORTD.B4;    GLCD_RS_Direction : sbit at DDRD.B4;
var GLCD_RW : sbit at PORTD.B5;    GLCD_RW_Direction : sbit at DDRD.B5;
var GLCD_EN : sbit at PORTD.B6;    GLCD_EN_Direction : sbit at DDRD.B6;
var GLCD_RST : sbit at PORTD.B7;    GLCD_RST_Direction : sbit at DDRD.B7; // End Glcd module connections

var DRIVE_A : sbit at PORTA.B2;    DRIVE_A_Direction : sbit at DDRA.B2;
var DRIVE_B : sbit at PORTA.B3;    DRIVE_B_Direction : sbit at DDRA.B3; // End Touch Panel module connections

var x_coord,y_coord,x_coord128,y_coord64 : longint; // scaled x-y position

function GetX() : word;
begin
    //reading X
    DRIVE_A := 1; // DRIVEA = 1 (LEFT drive on, RIGHT drive on, TOP drive off)
    DRIVE_B := 0; // DRIVEB = 0 (BOTTOM drive off)
    Delay_ms(5);
    result := ADC_Read(0); // READ-X (BOTTOM)
end;

function GetY() : word;
begin
    //reading Y
    DRIVE_A := 0; // DRIVEA = 0 (LEFT drive off, RIGHT drive off, TOP drive on)
    DRIVE_B := 1; // DRIVEB = 1 (BOTTOM drive on)
    Delay_ms(5);
    result := ADC_Read(1); // READ-X (LEFT)
end;

begin
    DRIVE_A_Direction := 1; // Set DRIVE_A pin as output
    DRIVE_B_Direction := 1; // Set DRIVE_B pin as output
    PORTB.B0 := 0; // Set PB0 pin as output (Default value 0)
    DDRB.B0 := 1;
    PORTB.B1 := 0; // Set PB1 pin as output (Default value 0)
    DDRB.B1 := 1;

    Glcd_Init(); // Initialize GLCD
    Glcd_Set_Font(@font5x7, 5, 7, 32); // Choose font, see __Lib_GlCDFonts.c in Uses folder
    Glcd_Fill(0); // Clear GLCD

    Glcd_Write_Text("TOUCHPANEL EXAMPLE;10,0,1);
    Glcd_Write_Text("MIKROELEKTRONIKA;17,7,1); // Display Buttons on GLCD:

    Glcd_Rectangle(8,16,60,48,1);
    Glcd_Rectangle(68,16,120,48,1);
    Glcd_Box(10,18,38,46,1);
    Glcd_Box(70,18,118,46,1);
    Glcd_Write_Text("BUTTON1;14,3,0);
    Glcd_Write_Text("PB0 OFF;14,4,0);
    Glcd_Write_Text("BUTTON2;74,3,0);
    Glcd_Write_Text("PB1 OFF;74,4,0);

    while (TRUE) do
    begin
        // read X-Y and convert it to 128x64 space
        x_coord := GetX();
        y_coord := GetY();
        x_coord128 := (x_coord * 128) / 1024;
        y_coord64 := 64 - (y_coord * 64) / 1024;

        //if BUTTON1 is selected
        if ((x_coord128 >= 10) and (x_coord128 <= 58) and (y_coord64 >= 18) and (y_coord64 <= 46)) then
        begin
            if(PORTB.B0 = 0) then
            begin
                PORTB.B0 := 1;
                Glcd_Write_Text("PB0 ON;14,4,0);
            end
            else
            begin
                PORTB.B0 := 0;
                Glcd_Write_Text("PB0 OFF;14,4,0);
            end;
        end;

        //if BUTTON2 is selected
        if ((x_coord128 >= 70) and (x_coord128 <= 118) and (y_coord64 >= 18) and (y_coord64 <= 46)) then
        begin
            if(PORTB.B1 = 0) then
            begin
                PORTB.B1 := 1;
                Glcd_Write_Text("PB1 ON;74,4,0);
            end
            else
            begin
                PORTB.B1 := 0;
                Glcd_Write_Text("PB1 OFF;74,4,0);
            end;
        end;
        Delay_ms(100);
    end;
end;
    
```

