Working with a Touch Screen

Touch point computed by a microcontroller

Design by W. Couzijn

Everyone nowadays is familiar with screens with which you can draw or type by touching a finger or a pen to the screen. How do such screens work, and how is the point of contact read out?

The circuit presented in this article makes it very easy to use a touch screen in your own projects. All that you need is a single IC, which is a standard microcontroller costing only a few pounds. The circuit is self-calibrating and very energy efficient.

There are several different types of touch screens. Some are based on capacitance or resistance, while others use optical effects or guided waves. The circuit described here is limited to resistive touch screens, since these are inexpensive and can easily be read out. The screen consists of two plastic sheets placed one on top of the other. These have a conductive coating with a relatively high specific resistance. Tiny discs between the two sheets keep them separated. The top sheet has highly conductive strips along the two edges in the vertical direction of the screen, while the bottom sheet has similar strips along the two edges in the horizontal direction. Each of the two sheets thus represents a sort of potentiometer, one aligned to the height of the screen and the other to its width. The 'wipers' of these two potentiometers make contact with each other when the screen is pressed. The settings of the two potentiometers are then directly related to the location where the screen is pressed. The four strips that form the outer ends of the two potentiometers are accessible to the user via a small cable. This allows the position where the screen is touched to be measured.

The circuit: a single-chip solution

The circuit for reading out the touch screen is shown in **Figure 2**. It contains essentially only one IC, which

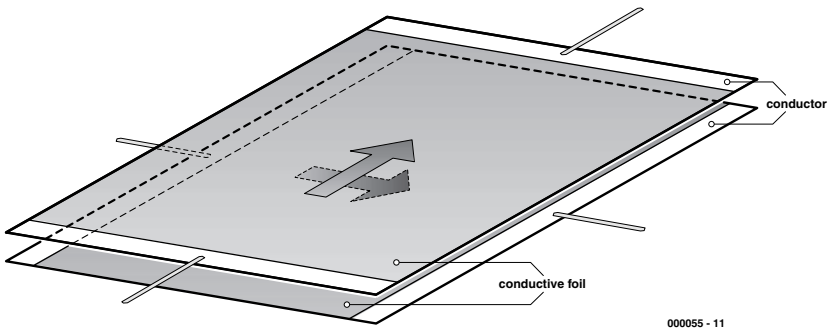


Figure 1. Construction of a resistive touch screen.

is a Microchip PIC12C671 microcontroller. This attractively priced 8-pin controller IC has an on-board 8-bit A/D converter and an internal RC oscillator that runs at approximately 4 MHz.

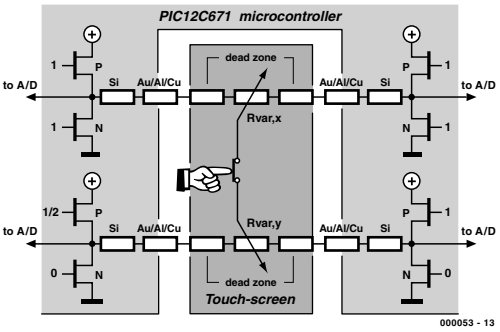
The remaining components are two transistors, three resistors, a diode, a voltage regulator and two capacitors. These are not necessary for reading the actual screen, but they provide power to the microcontroller and support communication with the PC. Since a serial PC port works with voltages that are significantly higher than 5 V, voltage conversion is necessary. The port is driven by software such that a voltage of around 18 V is produced between pins 4 and 7 of the 9-pin RS232 connection. This is rectified and reduced to a stable level of 5 V for the microcontroller. Pins 2 and 3 are the data channels RxD and TxD of the COM port.

Position detection

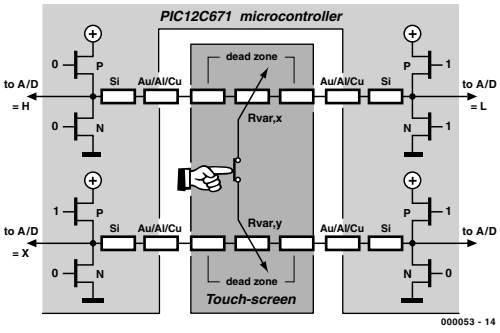
If you want to know where the screen is being touched, the first thing you have to do is to determine whether the screen is actually being touched (see **Figure 3a**). The microcontroller does this by configuring pins GP0 and GP2 as outputs and pins GP1 and GP4 as inputs, for measuring the X axis. Pins GP0 and GP2 have low (earth) levels, while pin GP1 has an internal pull-up resistor. If the screen is being pressed, there will be a conductive path from GP1 to earth, and the level on GP1 will thus be low. If the screen is not being pressed, there will be no conductive path from GP1 to earth, and it will be held at a high level by the internal pull-up resistor.

Determining the X position

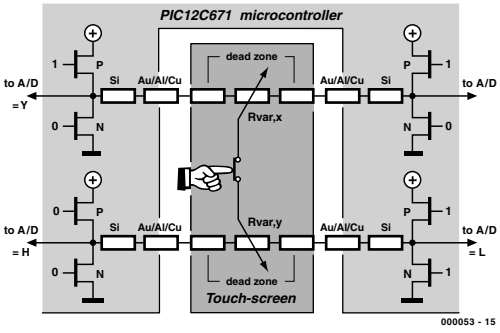
The position in the X direction can now be determined as shown in **Figure 3b.** The microcontroller sets GP0 high, which causes a current to flow through the X-axis potentiometer. The internal pull-up resistor is disconnected, since it would otherwise disturb the measurement. Since GP1 and GP4 are configured as inputs, they should have sufficiently high impedance that the position of the 'wiper' of the potentiometer can be read via these pins, as the result of a simple voltage divider circuit. The microcontroller now makes A/D measurements of the voltages on the two outputs (GP0 and GP2) and the input GP1. The voltages on GP0 and GP2 represent offsets that must be used to correct the measured value of the voltage on GP1. This is necessary because the A/D con-



a) Is the screen being pressed?



b) Determine the X position.



c) Determine the Y position.

Figure 3. The three successive stages of determining the X-Y position:

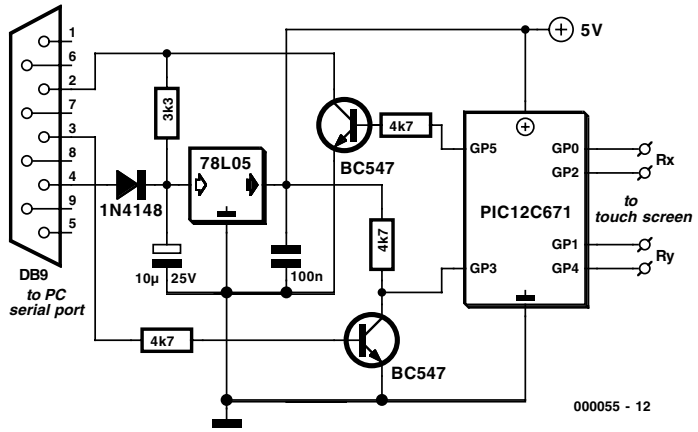


Figure 2. Schematic diagram of the touch screen circuit.

verter is never perfect, and in addition a fairly strong current flows through the digital outputs GP0 and GP2, which consequently do not supply exactly 0 V and 5 V. After these offset voltages have been taken into account, the calculation of the X position is completely independent of the value of the supply voltage, the voltage drops at the outputs, the offset of the A/D converter and the range of the A/D converter. The formula for this is:

$$X_{pos} = S\frac{128(X-L)-T(H-L)}{(H-L)(128-T-B)}$$

S = size (0 - 127)
X, L, H = A/D converter values
T, B = offsets (100%/128)

This formula has also been extended to include three variables that can be specified by the user. These are the two offset values for the left and right sides of the screen, plus a scale factor. These are not calibration factors! With these variables, the user can set the location of the (0,0) pixel and specify how many pixels must be placed in each of the two directions. This allows a sort of virtual screen to be superimposed on the physical touch screen, as illustrated in **Figure 4**. For example, if there are only two touch areas in the underlying LCD screen, the user can make two pixels that represent these two areas, and crop the edges of the screen by adjusting the offset values.

Determining the Y position

You can easily guess how the Y position is determined. As shown in **Figure 3c**, pins GP0 and GP2 are now configured as inputs, while pins GP1 and GP4 are configured as low-level and high-level outputs, respectively.
After the voltages on pins GP1, GP4 and GP0 have been measured, the same formula as before can be used to calculate the Y position in pixels:

$$Y_{pos} = S\frac{128(Y-L)-T(H-L)}{(H-L)(128-T-B)}$$

S = size (0 - .127)
Y, L, H = A/D converter values
T, B = offsets (100%/128)

In order to save energy, the voltage is removed from the touch screen after the position has been measured.

Measurement resolution

The built-in A/D converter of the PIC12C671 is 8 bits wide. Unfortunately, the resolution with which the touch position can ultimately be calculated is not this fine, due to non-linearity of the A/D converter and a loss of resolution caused by the offset corrections in the two formulas. However, a resolution of 6 bits is definitely achievable. It is possible to obtain even higher resolution than this by using a few tricks that work very well. For example, in place of applying a continuous 5-V level to the potentiometer while the analogue signal is being acquired, which takes several clock cycles, the level can be set to 0 V for 50% of the time and 5 V for the remainder of the time. If this process is repeated for several measurement cycles, the resolution can be increased to 8 bits.

There is also a small amount of hysteresis built into the microcontroller software, in order to prevent jitter in the measurement value when the position lies on the boundary between two pixels.

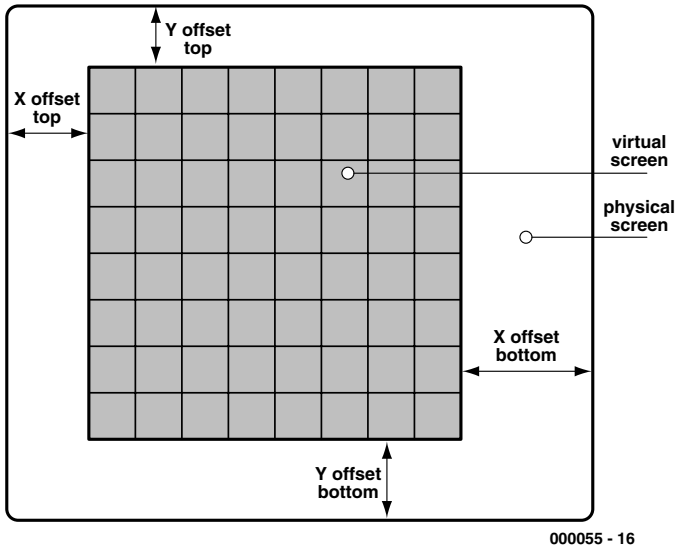


Figure 4. A virtual screen is superimposed on the physical screen.

Serial communication with the PC

A number of commands have been defined for requesting the touch position and setting the offset and scale factors. These are sent to the microcontroller using the normal serial data protocol: 8 data bits with 1 start bit and 1 stop bit, no parity, and 9600 baud. As already noted, the IC has an internal RC oscillator than runs at approximately 4 MHz. The exact frequency of the oscillator depends on a number of factors, such as temperature, supply voltage, production variables and ageing. The resulting deviations in the oscillator frequency can be large enough to cause bit errors and framing errors in the serial communication with the PC, which is naturally undesirable. To

Contents of floppy disk # 000055-II

Contents.txt	contents list
Copyright.txt	copyright notice
Egavga.bgi	auxiliary file for Test.exe
Test.exe	test program
Touch.asm	source code
Touch.hex	Intel hex file for programming the microcontroller
Tst.tpe	auxiliary file for Test.exe
Tst.tpr	auxiliary file for Test.exe

avoid such problems, the microcontroller carefully analyses the pulses that it receives from the PC, and then adjusts its internal oscillator based on timing measurements on the start bit. In this way, the communications never end up in limbo.

Commands

The touch position is requested by sending the hexadecimal code 'EF' to the microcontroller. The response is a 7-bit X position with the 8th bit low, followed by a 7-bit Y position with the 8th bit high. The 8th bit is thus used to distinguish the two data bytes (bit 7 of the final byte is always high).

The codes '0F', '2F' and '8F' are used to set the upper offset, lower offset and scale factor for the X direction. The scale factor specifies the number of pixels in the virtual screen, while the two offset values specify the sizes of the upper and lower borders that are not allowed to be used. Each command byte must be followed by a byte containing a 7-bit value for the factor to be set, with the 8th bit high. The set value is returned (echoed) to the PC as confirmation of its correction reception. Commands '4F', '6F' en 'AF' function in the same manner for setting the offsets and scale factor in the Y direction.

Finally, the command 'CF' can be used to read out all settings, along with the version number of the microcontroller software and the result of the software self-test. The software version number is sent first as a BCD value, followed by the two X offset values, the two Y offset values, the X scale factor and the Y scale factor. The response concludes with a byte for the result of the self-test, which normally has a value of zero. If the horizontal-axis sheet is not conductive, which indicates a broken contact, bit 0 is set high. Bit 1 has the same significance with regard to the vertical-axis sheet. Bit 2 is set if the horizontal-axis sheet is shorted, and bit 3 indicates the same situation for the vertical-axis sheet.

Construction and software

The software for this circuit consists of an assembly-language program for the microcontroller (provided as source code and a compiled hex file) and a DOS program for the PC. The latter program allows the functions of the circuit to be tested. The software is available from Readers Services on a floppy disk (order number 000055-11) and from the Elektor Internet site. There is no ready-made printed circuit board available for this project, but the circuit is so simple that it can easily be built on a piece of prototyping board. If you're handy, you should even be able to fit everything into a DB-9 connector.

(000055-1)