

An AVR Touch-Screen Interface

Abstract

General-purpose microprocessors can frequently be used to replace special-purpose chips, often at reduced cost. This project takes advantage of the AVR's multipurpose analog / digital port to create a touch-screen controller. Despite being easy to use and inexpensive, and occasionally available on the surplus market, I haven't seen any do-it-yourself articles for LCD touch screens. It's important to fill this gap, since touch screen LCDs can provide a powerful and intuitive graphical user interface for embedded systems.

The 4-wire resistive touch screen is a common type of touch screen. Here, two resistive sheets are separated a small distance. Electrodes are located along the top and bottom of one sheet and along the left and right of the other. To detect the vertical coordinate, a voltage is applied to the top electrode and the bottom electrode is grounded, and voltage is measured at either the left or right electrode. A touch shorts the two sheets together; the resistance in the vertical direction forms a voltage divider, and the voltage is read out through the resistance of the other sheet. The horizontal coordinate is read by the same method, applying voltage across the left and right electrodes.

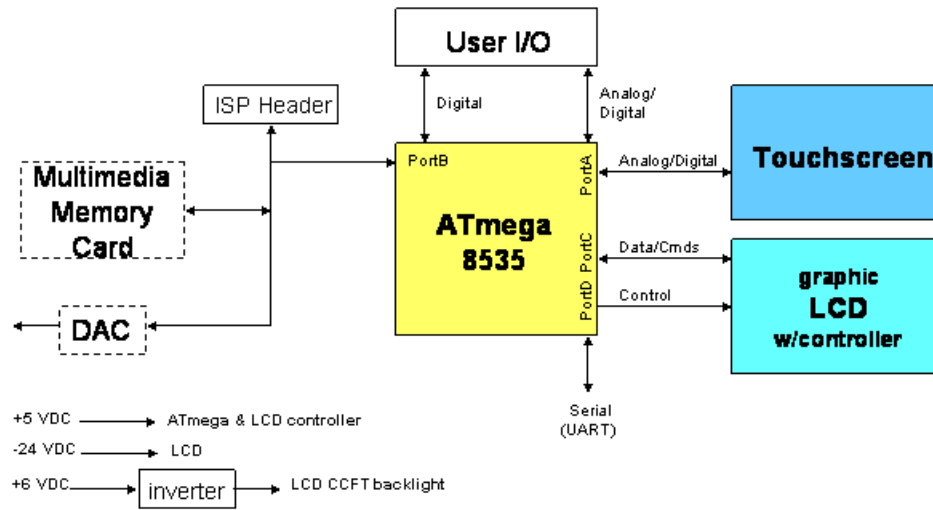
An off-the-shelf touch screen controller chip costs about \$2.50 in small quantities. Cheap, but more expensive than an ATtiny. Nearly all systems with a touch screen will have a microcontroller anyway, and AVR microprocessors have the flexibility to serve as a touch-screen controller with essentially no external components. Each of the A/D inputs can alternatively serve as a digital output, and this selection can be made on a pin-by-pin basis, giving us a software-only solution.

This project demonstrates the AVR touch-screen interface with a graphics LCD. An ATmega8535 provides touch-screen and LCD interfacing, with lines left over for in-system-programming and 'end-user' applications. The demo uses two virtual buttons. One triggers a 128-sample analog capture and graphing, and the other clears the display. The demonstration was programmed using GCC with the Procyon library. Compiled code for the "oscilloscope" demo is just over 5 KBytes.

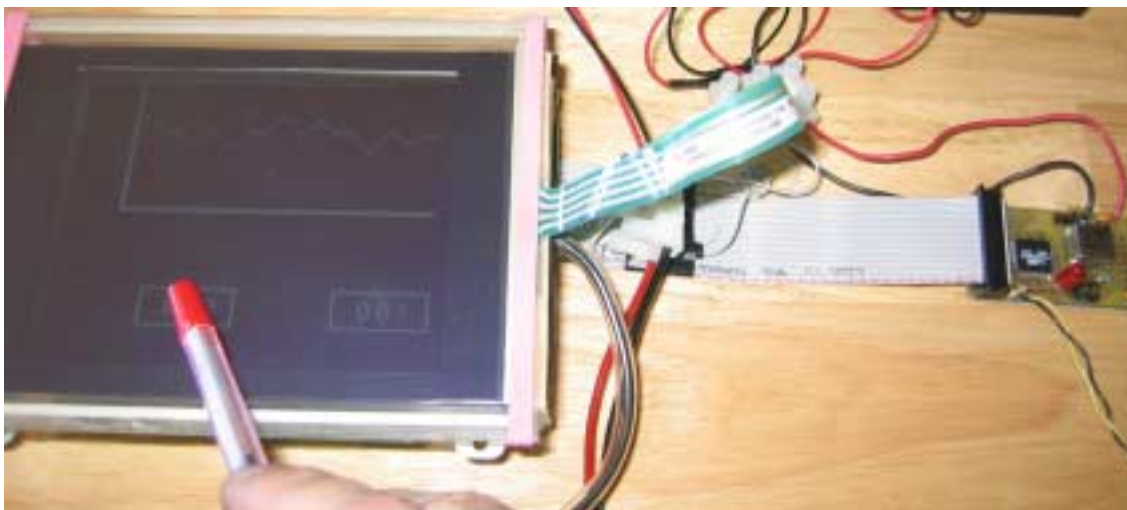
The touch-screen interface has high resolution, and can also be used for mousing and similar GUI functions. The board was laid out to support functionality such as analog and digital signal capture, and include a MMC for storing captured signals and a DAC for signal generation. But that's another story.

This project has demonstrated that the AVR analog / digital port can be used to produce a touch-screen controller with virtually no added hardware. Only 8 lines of code are required to read the touch X and Y coordinates. Even an ATtiny can serve as a touch-screen controller, and at a cost lower than touch-screen controller ICs. For only a few dollars more, an ATmega can also serve as a serial interface to the LCD itself, as well as provide additional I/O and processing, opening up an array of sophisticated GUIs.

Block Diagram



Photograph



Software Code Sample

```
u08 buttonTouch(void)
// read touchscreen x, y
// return button touch location
// high nibble is x-box, low nibble is y-box
// 0x00 is no-touch
{
    u08 b, x, y;

    // check X-axis
    DDRA = 0xA0;    // left and right are 'control'
    PORTA = 0x20;   // left = L, right = H
    msecDelay(16);
    x = a2dConvert8bit(TOP);
    // check Y-axis
    DDRA = 0x50;    // top and bottom are 'control'
    PORTA = 0x10;   // top = L, bottom = H
    msecDelay(16);
    y = a2dConvert8bit(LEFT);

    // return no-touch if outside LCD area
    if ( (y < touchTop) || (y > touchBot) )
        return 0;
    if ( (x < touchLft) || (x > touchRt) )
        return 0;

    //rprintf("Touch x,y: %d, %d\r\n",x,y);

    // quantize into bins
    x = (x-touchLft)/xBin;
    y = (y-touchTop)/yBin;
    // pack as BCD, starting the counting from 1
    b = x*10 + y + 11;
    rprintf("Button %d\r\n",b);
    return b;
}
```

Schematic

