CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS

FEATURE ARTICLE

Tom Dahlin

Reach Out and Touch

Designing a Resistive Touchscreen

While working on a design assignment that involved an expensive off-the-shelf touchscreen interface, Tom began to consider rolling his own. Watch as he points out some of the tricks of working with resistive touchscreens.

The use of touchscreens as computer interface devices has become widespread. I don't think there are many readers out there who have not used an ATM machine with a touchscreen or at least played with one of the Apple Newton-type personal digital assistants. The popularity of touchscreens is in a large part due to the simplicity they bring to user interfaces. They eliminate the need for a large keyboard and offer the benefit of context specific menu choices.

I was recently involved in the design of a machine controller for an industrial application. We liked the advantages of a touchscreen interface, and prototyped the system using an industrial embedded PC, and a commercially available motion-control board. We purchased an off-the-shelf industrial LCD display/touchscreen combination and connected it to the PC's VGA and COM1 interfaces. We then developed the application software in VB and got the machine up and running in short order. Life was good.

The euphoria was short-lived however. Economic reality set in, and we realized we couldn't justify spending over $2k on the off-the-shelf LCD/touchscreen for the commercial product. Besides that, the mechanical form factor was wrong. With a short development cycle ahead of us, life was looking not so good.

We quickly realized that we would either have to roll our own LCD display/touchscreen, or drop back to a character-oriented LCD/16 key keypad for the user interface. While it is not practical for a small company to consider making its own touchscreen (the glass part that is), it is possible to save both money and package size by purchasing only the raw touchscreen, using a standard LCD or CRT, and designing your own controller and package. This article will show you a few of the ways of designing the touchscreen-to-computer interface.

TYPES OF TOUCHSCREENS

Not all touchscreens are created equally. There are two primary technologies used today—resistive and capacitive sense. There are others, such as IR scanning, acoustic wave, and electromagnetic technologies. Although they all have their merits, resistive- and capacitive-sense tech-

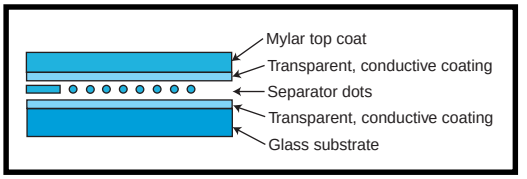

Figure 1—*Most resistive touchscreens have a construction similar to the one shown here. Two conductor layers are separated by a layer of tiny dots. The dots allow the two planes to make contact when force is applied to the top layer.*

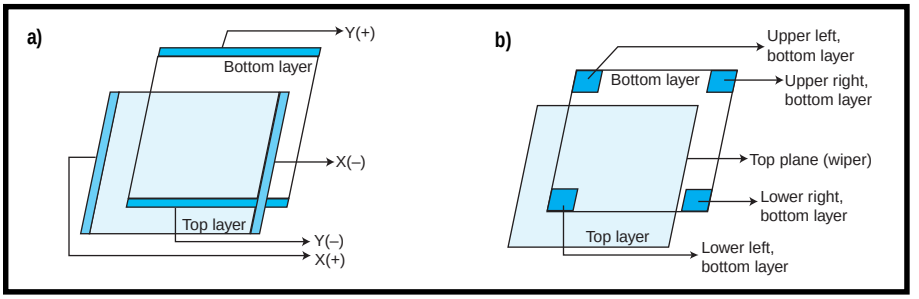

Figure 2a—*A 4-wire touchscreen has bus bars on the right and left edges of one layer, and on the top and bottom edges of the other layer.* **b—***A 5-wire version has four leads, each connected to the corner of the bottom layer, and one lead connected to the top layer that works like a potentiometer wiper.*

nologies have emerged as favorites thanks to their low cost and high resolution.

Resistive-sense technology has the added advantage of being able to detect a touch from a rubber-gloved finger, something that is a problem for capacitive type touchscreens. The other technologies mentioned are used when they offer a unique advantage, such as the sense-before-touch feature that IR scanning provides.

For a discussion of the relative pros and cons of each technology, you might want to read the application notes written by David Blass of Sharp [1], and another by VJ Kuroodi of Tritech [2].

AN INSIDE LOOK

Resistive touchscreens almost all start with a glass or hard plastic substrate, onto which a thin, transparent conductive layer (usually ITO) has been applied. Figure 1 illustrates a typical touchscreen cross-section.

A fine grid of spacer micro dots is then applied and another layer of a conductive-coated flexible plastic (usually Mylar) is laid on top. You end up with two transparent conductive planes of material separated by a few thousandths of an inch. Pressure from a stylus or finger causes the two planes to make electrical contact and forms the means of sensing the touch.

There are two commonly used types of resistive touchscreens. These are called 4-wire and 5-wire. In a 4-wire resistive touchscreen, the two planes each have two wires connected to opposite ends.

For example, the *x* plane would have wires connected to the left and right edges, and the *y* plane would have wires connected on the top and bottom edges (see Figure 2). In operation, a controller must first apply a voltage across the *x* plane thereby forming a gradient because of the resistive coating.

A touch is sensed by using the *y* plane as an input to an ADC and detecting a voltage when the two planes are forced together. The ADC reading will vary as a function of the *x* (right/left) position of the touch.

The *y* position is then calculated by removing the voltage from the *x* plane and applying it to the *y* plane, from top to bottom. The *x* plane is then used as a pickoff and its output is routed to the ADC.

In a 5-wire resistive touchscreen like the one shown in Figure 2b, the operation is similar, but the alternating *x* and *y* fields are applied across only one plane, and the other plane is used solely as a pickoff. Thus, one wire goes to each corner of the bottom plane, and a fifth wire is connected to the top plane.

To read an *x* position, the controller applies a voltage to the left two corners and ground to the right two. The fifth wire would go to the ADC. To read a *y* position, the controller grounds the bottom two corners and applies a voltage to the top two.

SMORGASBORD OF OPTIONS

There are several different ways of interfacing the glass touchscreen to your system. I'm going to show you four methods I've used with success. Two of the methods use a combination of a PIC and a touchscreen controller chip, one uses only a PIC, and the last uses no processor at all. Which method you use depends on your system requirements for scan rate, accuracy, and cost.

Interfacing to either a 4- or 5-wire touchscreen is easy thanks to a pair of chips from Burr-Brown. The ADS7843 is designed to interface to a 4-wire touchscreen, and the ADS-7845 to a 5-wire. The devices have identical hardware and control interfaces, differing only in the type of touchscreen they interface to.

Figures 3 and 4 show examples of circuits using a PIC and the devices to interface to both 4- and 5-wire touchscreens. Let's take a look at each circuit and chip individually, starting with the 4-wire device.

The ADS7843 is a single-chip interface to a 4-wire touchscreen. At its core is a 12-bit successive approximation analog-to-digital converter (ADC). It performs all the front-end analog multiplexing necessary to generate the required voltage gradients across the touchscreen planes and switch the pickoff into the ADC.

As shown in Figure 3, a PIC and an RS–232-level shifter are all that's required to build a 4-wire interface to a PC com port. The PIC-to-ADS7843 interface is simple, needing only three lines—clock,

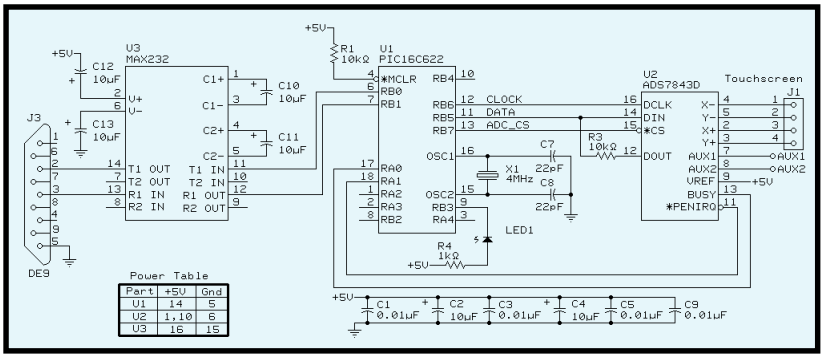

Figure 3—*3 chips are all you need for this circuit. The ADS7845 handles the analog functions and the PIC performs the sequencing, scaling, and messaging formatting. The MAX232 handles the RS-232 level shifting.*

data, and chip select.

In the example circuit, the data input and output lines from the ADS7843 are tied together via a 10-kΩ resistor. This arrangement allows the use of a single PIC I/O line to handle both. Also shown is a PIRQ, or pen interrupt signal that can alert the PIC to the presence of a touch (or pen in a PDA application), and a BUSY signal that enables the PIC to monitor the status of the ADC conversion. The latter two signals are not used in my application, but are brought into the PIC for future use, if needed.

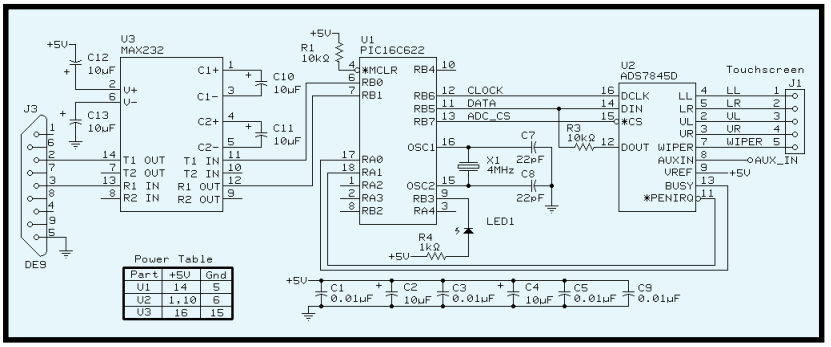

Figure 4—*By swapping an ADS7843 into the circuit shown in Figure 3, we can create a 5-wire controller. The ADS7843 and ADS7845 are 12-bit devices that provide resolution capability up to 1/4096 of the touchscreen width.*

Figure 5 shows the logic diagram and timing of the PIC interface. This timing is called 24-clock mode, referring to the single byte of control info sent to the device and the two bytes returned. Other modes are also available and overlap the shifting of data in and out to save transit time, providing an ability to get more samples per second.

To read the touchscreen, the PIC must send an 8-bit control byte to the ADS7843. This byte always has its most significant bit or start bit set. The next three bits (A2, A1 and A0) specify whether we want to read *x*, *y*, or one of the two Auxiliary ADC inputs.

The auxiliary inputs, with proper signal conditioning, can be connected to any system analog value you might want to read (e.g., a battery voltage). The next bit is called MODE and is set to zero if you want a 12-bit conversion, or a one if you want an 8-bit conversion. The following bit is called SER/DFR and is set to zero if you want to use a differential voltage reference (normally preferred) or is set to one for a single-ended type.

Lastly, the final two bits in the control byte are called PD1 and PD0. These are the power down mode select bits. If both are low, the device is powered down between conversions to save power for portable applications. If both are high, then the device is always enabled. An important fact is that the PIRQ, or pen interrupt is disabled when either of the two bits is set high. Thus, you can't use the PIRQ if you leave the device always powered.

So, if the PIC wants to read the *x* channel, it sends a $93 to the ADS-7843. This selects not only that channel, but also a 12-bit conversion, differential referencing, and non-powerdown operation. After clocking out these eight bits to the ADS7843, in return, the PIC clocks in 16 bits that contain the 12-bit result and four zero-filled trailer bits. To read the *y* channel, the PIC performs the same operation, only sending a $D3 for the control byte.

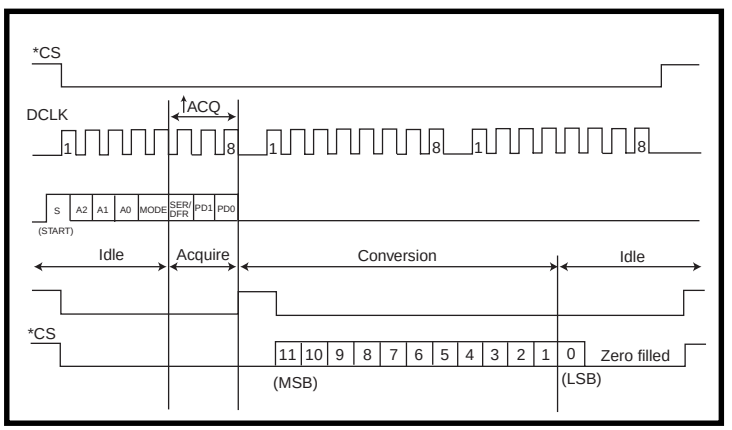

Figure 5—*eight clocks are used to shift out a control byte from the PIC to the device and another 16 clocks are used to retrieve the result. Since the data input and output lines are not simultaneously active, it's possible for them to share the same microcontroller I/O pin.*

TAKING THE 5-WIRE ROUTE

Burr-Brown has the ADS7845 device for a 5-wire touchscreen interface. Like its cousin the ADS7843, it connects to your microprocessor via a simple serial interface. It too uses a 12-bit ADC. The pinouts of the chips are nearly identical. The 5-wire device uses one of the two spare analog inputs available on the 4-wire device to accommodate the fifth wire input.

Another way to interface to a 5-wire touchscreen is to do it all with a PIC. As shown in Figure 6, this method results in a low parts count. The thing that makes this design easy is the fact that we can control the four corners of the bottom plane with the PIC's digital drivers and run the sense-plane wire directly into the PIC's on-chip ADC. Thus eliminating the need for fancy analog multiplexing using external FETs.

As you can see, we used four PIC I/O lines (RB2–5) to connect to the four corners, labeled UL (upper left), LL (lower left), UR (upper right), and LR (lower right). To generate a left-to-right voltage gradient, the PIC sets UL and LL to a low (0 V) and sets UR and LR high (5 V). It then performs an ADC conversion, reading AN0.

The presence of a voltage greater than a few counts indicates a touch. The bleed resistor R5 in Figure 6 pulls the ADC input low, so we have no problem knowing that a touch has occurred.

To generate a top-to-bottom voltage gradient, the PIC simply sets UL and UR to high and LR and LL to low. Note that LL is always low and UR is always high. Although we could hardwire them to ground and +5 V respectively, it's better to allow the PIC to do this to preserve balanced levels on all four corners.

Once the PIC has secured readings

for the *x* and *y* directions, it can adjust for offset and scale factors, and determine the *x* and *y* positions. The PIC I used was a PIC16C71 with an 8-bit ADC, which works for applications where positional accuracy is not important. Newer members of the PIC family have better accuracy and would improve this design.

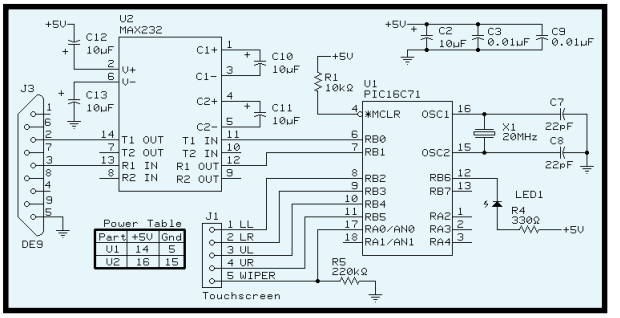

Figure 6—*Four of the PIC's digital lines are used to provide* x *and* y *voltage gradients. A single analog input is used to determine the contact point by measuring the pick-off voltage in both the* x *and the* y *planes.*

LOOK MA, NO PROCESSOR

If you want a simple interface to a 4-wire glass and you can live with a predefined output format, the TriTech TR88L811 chip makes it possible to go from the glass to a serial bitstream with only one chip. If you have an extra serial channel available and only need 10 bits of positional accuracy, then this may be the way to go.

The TR88L811 is designed for standalone applications and requires only a 1.8432-MHz crystal and an RS-232–level shifter to form a complete interface that you can attach to a spare PC COM port.

Figure 7 shows an example circuit that steals its power from the PC's COM port. The TriTech device scans the touchscreen continuously and sends a serial data packet out of its TxD pin when a touch is detected. The data packet, sent at 19,200 bps, contains five bytes—a header and two bytes each of *x* and *y* position. The position is resolved to 10 bits, which is adequate for most applications.

If a touch is maintained, the chip will send data out at a rate of approximately 200 coordinate pairs per second. The main advantage of this device is that it requires no firmware. As long as you can live with the 10-bit resolution and can work with its output format, then it's a turnkey solution.

You can buy the raw touchscreen glass from several manufacturers. I've had a good experience dealing with the Bergquist Company and I've also been successful interfacing to glass made by Elo and Microtouch.

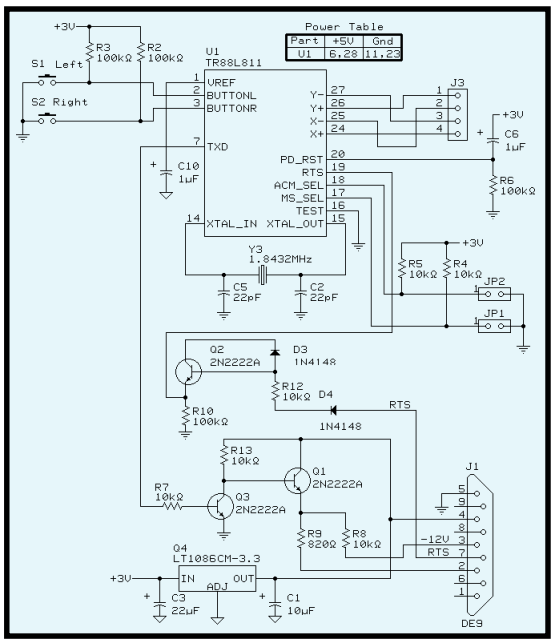

Figure 7—*The processor in this interface is actually the Tritec TR88L811, a dedicated 4-wire touchscreen interface device. It handles all of the touchscreen scanning, touch detection, and message formatting chores. The 3-VOLT device was originally developed for the PDA industry.*

DEVELOPMENT NOTES

I began this project by first locating the touchscreen-interface chips. I then needed a quick and dirty means of testing their functionality, which I did using a Basic Stamp II device with a serial LCD attached. It was a simple matter to interface the chips to the Stamp, and then use the interactive Stamp-development environment to debug the chip interfaces.

Of course, a $50 Basic Stamp is not a good solution for a production product. The production hardware was designed around a lower cost PIC, the 16C622. I intended to discard the Stamp code and write the production code in C. However, I had heard about a compiler for the Stamp BASIC, and decided to give it a try. The results were good and enabled me to use most of the stamp code with little modification. I've since used this approach (successfully) on other projects.

The compiler is the PIC BASIC Pro (PBP). It is available from microEngineering Labs for about $250.

FIRMWARE DESCRIPTION

Firmware for the 4-wire Burr-Brown design is available on the *Circuit Cellar* web site (the 5-wire firmware is nearly identical). The PIC's job is to sense a touch event, and if detected, send out a data packet containing the location of the touch. The PIC should continue to send out this information as long as a touch is detected and send a special packet at the end when the touch has gone away.

The PIC sends a five-byte data packet to communicate the *x*/*y* touch coordinates to the host processor (see Figure 8). The first byte in the five-byte header always has its most significant bit set to one. The other four bytes always have their most significant bits set to zero to allow the receiver to synchronize to the packet. The first byte is either a $C0 or $80, depending on whether the touch is active ($C0) or a touch-up event has occurred ($80).

The four bytes that follow hold the *x* and *y* coordinate values, with 14 bits allowed. Because we digitize to 12-bit resolution, bits 12 and 13 are zero filled.

The firmware was written as a state machine, as shown in Figure 9. On powerup, the code enters State 0, where it initializes the data direction registers and blinks the LED three times. A transition to State 1 follows, where the code continuously scans

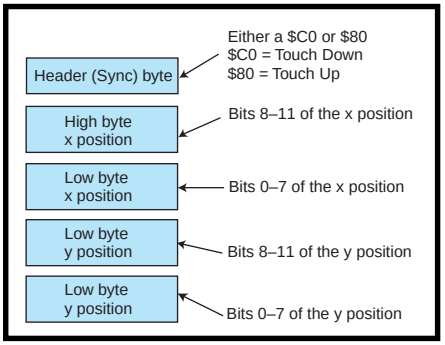

Figure 8—*This is the message format used for circuits in Figures 3, 4, and 6. A simple five-byte data packet is used to transmit the touch coordinates from the touchscreen controller to the host system.*

the touchscreen, looking for a touch.

When a touch is detected, a transition to State 2 occurs, where the controller sends out a five-byte data packet, and turns on the LED. While in State 2, the controller continues to scan the touchscreen and sends out a new data packet each as long as a touch is detected. If a touch is not detected, a transition to State 3 occurs, where a final data packet is sent with the header byte set to $80 indicating a touch-up event, the LED is turned off, and a transition back to State 1 is initiated.

A low-level subroutine called `Convert` is used to talk directly to the touchscreen controller chip. The routine is similar for both 4- and 5-wire chips. `Convert` passes a variable called channel, which contains a 0 or 1. This variable controls whether we read the *x* or *y* channel of the device. The 12-bit result comes back in a 16-bit word named ADC.

`Convert` is called by a higher level subroutine named `Read_Glass`. This routine determines if a touch has occurred and sets a flag called `touch` to indicate such. A touch is determined to have occurred if the result of a `Convert` operation is above a certain noise threshold. When `Convert` is called and no touch has occurred, a value near zero is returned.

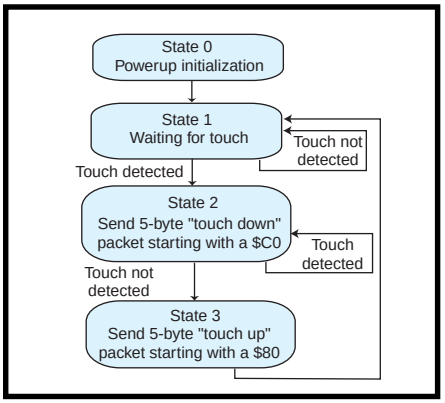

Figure 9—*In this flow diagram for Figures 3, 4, and 6, the controller continuously scans the touchscreen, detects the touch, and sends out a five-byte data packet.*

WRAPPING IT UP

Well, there you have it. You've seen four different means of interfacing a resistive touchscreen to your system. The Burr-Brown chips offer high accuracy, off-the-shelf solutions to both 4- and 5-wire glass types. My roll-your-own PIC interface is a minimal parts count design and is best suited to low-accuracy applications. The TriTech chip offers a unique solution in that it involves no firmware. Those of you whose favorite programming language is solder will appreciate that.

Tom Dahlin is a published author who runs a consulting business where he has developed the electronics for products such as a Talking Trash Compactor used in the fast food industry, an automated french fry dispenser, and (don't laugh) a pig sperm heater/controller for the animal-breeding industry. Tom has 20 years of design engineering experience at 3M and Honeywell, and was most recently director of electronics engineering at Stratasys, a manufacturer of 3D model-making machines. When it really gets crazy he likes to retreat to his cabin in Michigan's Upper Peninsula. You can reach him at tdahlin@isd.net.

REFERENCES

D. Blass, *Touch Screens for Flat Panel Applications*, Sharp application note, January 18, 1996.

V. Kuroodi, *Pen Input Capability for Portable Devices*, TriTech Microelectronics application note, 1996.

SOURCES

TR88801/802 4-wire controllers
TriTech Microelectronics Int'l Inc.
(408) 894-1900
Fax: (408) 894-1919

ADS-7843/-7845
Burr-Brown Corp.
(520) 746-1111
Fax: (520) 889-1510
www.burr-brown.com

Touchscreen glass
Bergquist Touch Products
(800) 796-6824 • (612) 835-2322
Fax: (612) 835-4156
www.bergquistcompany.com

Microtouch Systems Inc.
(800) 642-7686 • (978) 659-9000
Fax: (978) 659-9105
www.microtouch.com

PIC16C71
Microchip Technology, Inc.
(888) 628-6247 • (480) 786-7200
Fax: (480) 899-9210
www.microchip.com

Basic Stamp
Parallax, Inc.
(916) 624-8333 • Fax: (916) 624-8003
www.parallaxinc.com

Circuit Cellar, the Magazine for Computer Applications. Reprinted by permission. For subscription information, call (860) 875-2199, subscribe@circuitcellar.com or www.circuitcellar.com/subscribe.htm.