FEATURE ARTICLE

by Matt Oppenheim

Multichannel Touch Sensors

Implement Scalable Capacitive Touch Sensing

Tired of switching? Try touching. It's now easy to implement a versatile multichannel touch sensor in various projects. Here you learn how to design a 20-channel device of your own.

One of my many misdemeanors as a child was to scuff across the carpet on a dry day and then touch my sleeping cat on the nose. The resulting static discharge resulted in a few choice words of feline outrage. Several decades later, I have learned to exploit the body's ability to act as a capacitor for more useful purposes.

In this article, I'll explain how easy it is to implement a versatile multichannel touch sensor with Atmel's Qprox QT1103 10-channel capacitive touch sensor IC, which uses QTouch technology. The project progressed through the traditional path. First, I hacked up the manufacturer's demo board to prove the capability of the technology. I then produced custom PCBs. Next, I designed a 20-channel device using two QT1103 chips. I followed that up with a 100-channel device using five of the 20-channel sensor boards. Although I'll focus on the 20-channel device in this article, I'll also mention the main technical issues I dealt with while designing the 100-channel instrument.

HOW IT WORKS

With the new all-in-one "black box" ICs, it is easy to ignore the underlying physics of how the technology works. But an understanding of the principles makes the difference between building a robust device and one with intermittent problems.

By definition, a capacitor is something that stores charge. You are all familiar with the classic two-plate symbol for a capacitor. The charge is stored on the two plates and the space between them may be empty air or filled with material (called dielectric). The bigger the plates, the more charge that the capacitor can hold. A finger also has the ability to hold charge, as my cat found out. So, when you touch one of the plates of a capacitor, you are effectively adding to the amount of charge that it can hold—increasing the total capacitance of the system. Figure 1 depicts how a finger can act as a capacitor interacting with the reference capacitor on the QTouch circuit's input pins.

Before the advent of all-in-one ICs like the QT1103, this capacitance was measured by sending a pulse along the touch sensor line and comparing the charge left after a certain amount of time with a reference level, using a comparator to digitize the output. Sometimes the internal comparator in a microprocessor would be used.

The increased capacitance due to your finger is detected

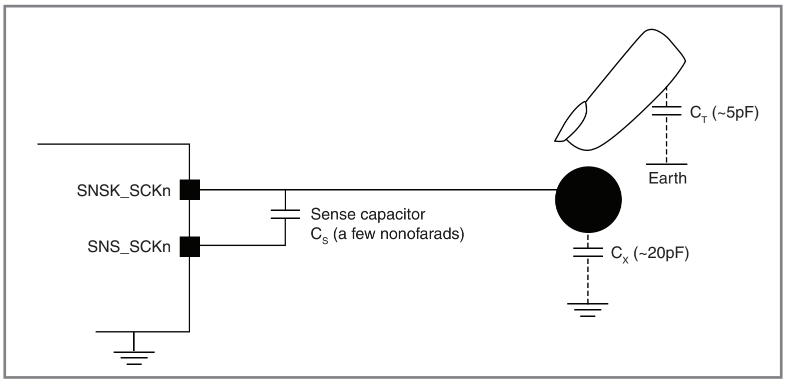

Figure 1—The QTouch measuring circuit. The finger acts as a small capacitor that adds to the capacitance already present between the two sensor pins. (Source: STMicroelectronics QST104 datasheet, which uses the same QProx charge transfer capacitive technology as the QT1103).

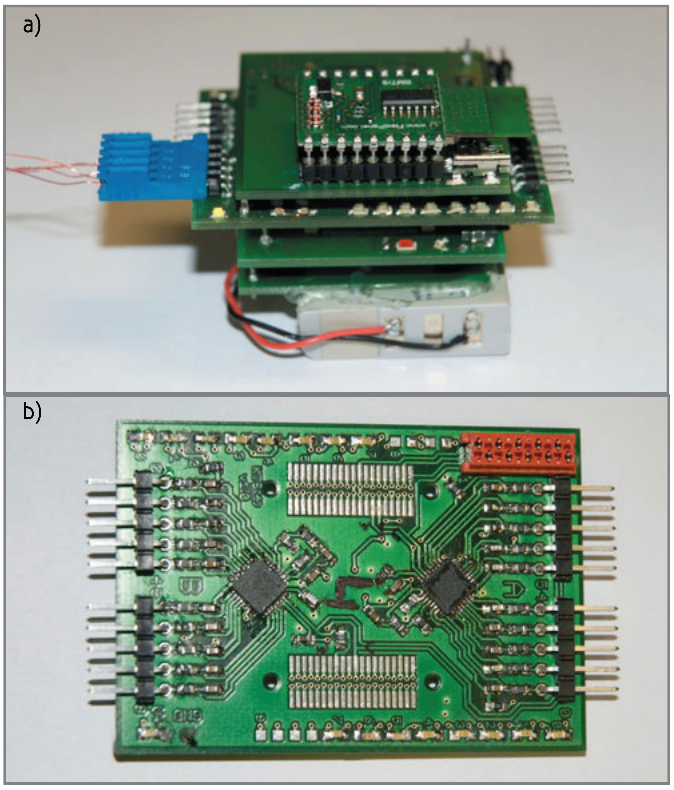

Photo 1a—The Bluetooth DIP module is on the top. The blue connector on the left shows how sensor lines attach to the 0.1″ pins visible on the edges of the sensor board. Underneath the sensor board is the dsPIC33FJ128GP706 controller board. The bottom board houses the battery-management hardware and a lithium battery. b—A close-up of the sensor board. The QT1103 chips are the two black squares.

by a corresponding increase in the charge left after the test interval. This capacitance is over the preset limit, so the comparator indicates that the sensor has been touched. It clearly takes up a lot of PCB real estate and a number of components. The advantage of chips such as the QT1103 is that they offer multiple channels with a minimum of interface lines and they take care of the sensor-triggering processing. The QTouch technology uses a sophisticated arrangement of pulses to detect the change in capacitance so as to reduce cross-sensor interference and power consumption.

DESIGN EVOLUTION

I was tasked to produce devices that would enable the Human Computer Interaction (HCI) group at Lancaster University's Infolab21 to augment surfaces with a touch-sensing capability. The first project involved modifying a QT1103 demo board to enable it to be wired to 10 keys on a standard PC keyboard. The demo board has a single QT1103 with LEDs to display the status of each channel. You can connect your own sensor pads or wires to the board by removing the PCB keypad that comes with it. This exposes 0.1″ headers. It enabled me to connect laminate wires directly to the sensor lines. The result was a classic kludge of a demo board, breadboard, and an FTDI UART-to-USB interface module to get the data to a PC. You can access the QT1103 communication lines on the demo board, but you need to solder in your own 0.1″ pitch connector. I used this to hook up my Microchip Technology dsPIC development board and get to grips with the communication protocol.

Once I was confident in the technology, I designed a PCB for a 20-channel device that used two of the QT1103 chips. This connected to a Microchip Technology dsPIC33FJ128GP706 controller board, a battery board, and a communications board with a Bluetooth module. Photo 1a is the entire device. Photo 1b is a close-up of the sensor board.

I recycled a dsPIC break-out board and battery controller board from an earlier project. Using a 16-bit digital signal processor (DSP) was overkill for this application, but it enabled me to complete the project faster than if I had designed a new board. I knew I could also reuse the C libraries I had written for that DSP. Plus, I knew that if the end application turned out to be a mass-produced commercial device (rather than a custom research instrument), the component cost would be more critical and mandate a cheaper design.

After the successful 20-channel design, I went on to enhance 100 keys of a standard PC keyboard with touch sensitivity using five of the 20 channel boards controlled by a single dsPIC. The touchpad on the back of each key was made by pressing adhesive-backed foil onto a laminated copper wire connecting to the touch sensor. The five sensor channel boards and their interconnections to the controlling dsPIC board appear in Photo 2.

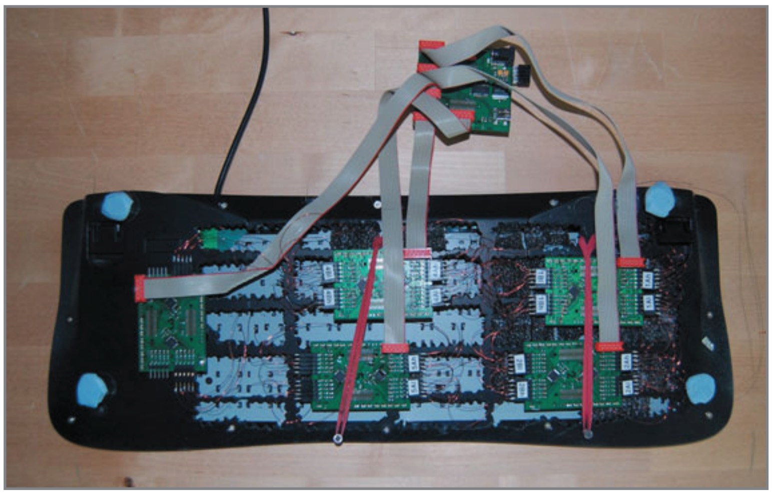

Photo 2—The rear of the 100-channel touch sensor enabled keyboard. Five of the 20-channel sensor boards are connected to the base of the keyboard. The dsPIC controller board is connected with ribbon cables to each of the sensor boards.

QT1103 DESIGN

The QT1103 is advantageous because it requires only a single sensor line for each of the touch sensors. Some other chips require a reference ground line for each channel, which is fine for a static PCB touchpad layout, but would give me a wiring headache because the sensor lines are thin-laminated wire stuck onto the back of a keypad.

Each sensor line has a reference capacitor and resistor. You can alter a line's sensitivity by changing the capacitance. By decreasing the reference capacitor, the channel's sensitivity is increased (so that it will trigger through a sheet of plastic, for example). If you are using large pieces of copper for sensor pads, the value of the reference capacitor may need to be increased to reduce the triggering threshold.

Each chip interfaces with a microcontroller using two or three I/O lines. The *CHANGE line goes low whenever the QT1103 detects a change on any of the sensor lines. The host microcontroller then requests the new status information by sending the character "P" to the QT1103. The chip then sends two 8-bit bytes. These encode the status of the 10 sensor lines (see Figure 2). Then the *CHANGE line returns high. The chip enables you to send and receive data using two separate lines—or to do both on the same line. I used the single line for both transmission and reception. Figure 2 shows the data protocol. The transmission protocol is standard UART. Don't be fooled by the term "one-wire" in the datasheet. It means standard UART, but using only one wire to both transmit and receive.

The transmission lines need to be held high with 100-kΩ resistors. Obviously, I had to change the status of the data line from transmit to receive for each cycle of sensor status updates. If for some reason your microcontroller can't send and receive on the same line, you have to use separate wires for sending and receiving.

The QT1103 datasheet shows a number of options that can be set with resistors (e.g., how quickly the chip enters its low-power mode). The chip can be set into a "simplified" mode using a single resistor—R55 in Figure 3. Sufficient for most applications, this mode is used by the demo board. The only option to consider in the simplified mode is whether to enable the adjacent key suppression feature. This prevents multiple adjacent channels from triggering from a single touch, which can happen with closely spaced keys or with a wet surface. This option is set by another 1-MΩ resistor connected to line SNS0 (R in Figure 3). If this is pulled high, the option is on. If it is pulled low, it is set off. The datasheet includes the full details and the other options.

For my purposes, one of the

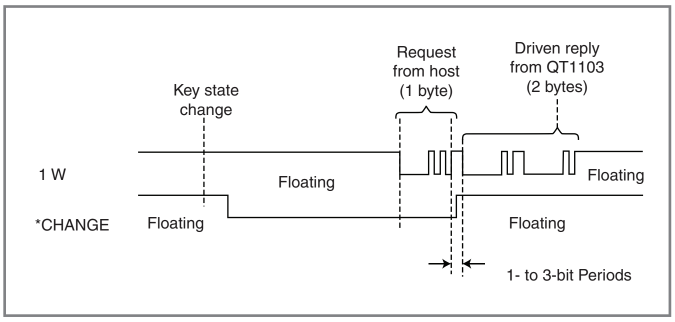

Figure 2—The QT1103 data protocol. The controller sends the character "P" once the *CHANGE line goes low. Then the sensor chip sends the status of the 10 channels in 2 bytes. (Source: Qprox QT1103 datasheet)

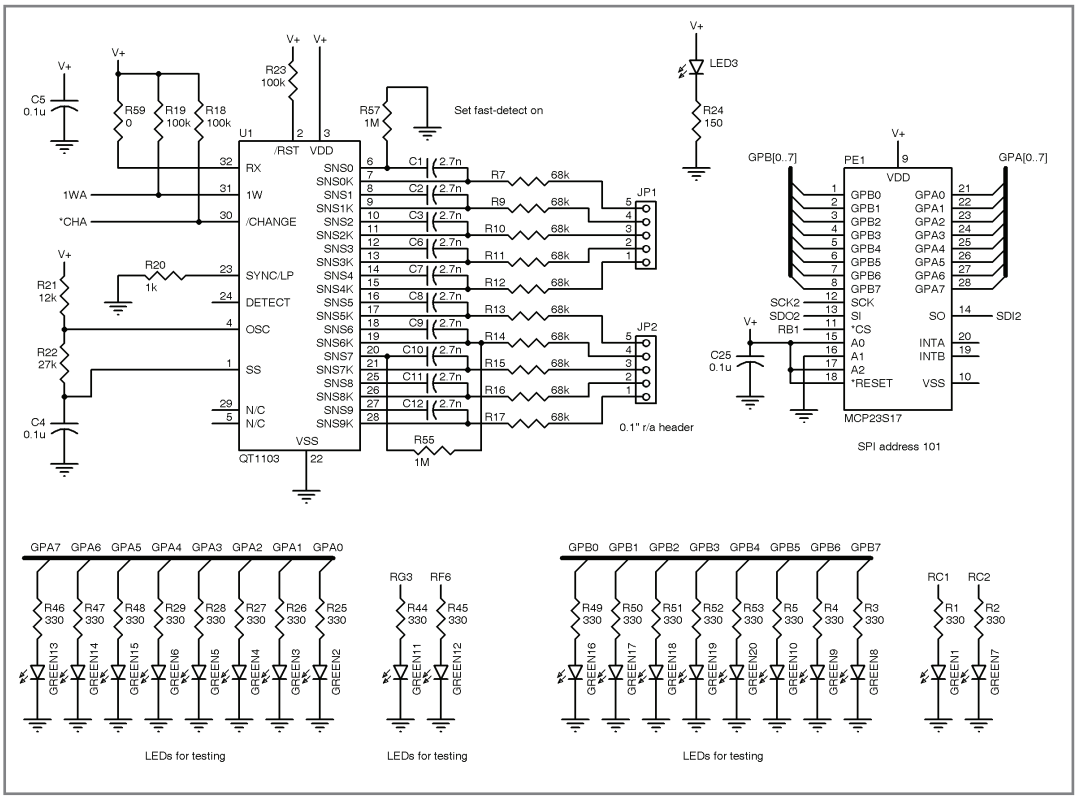

Figure 3—Each sensor channel on the QT1103 requires a resistor and reference capacitor. The MCP2317 can directly power the status LEDs.

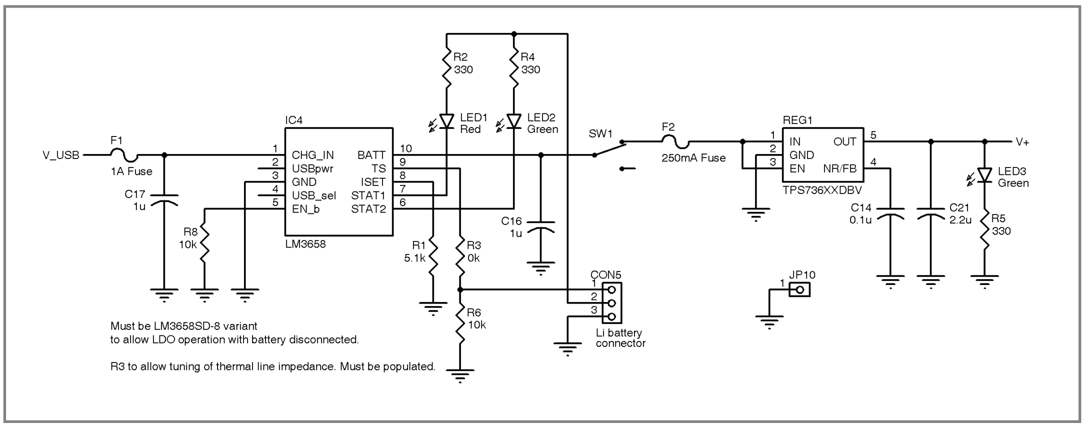

Figure 4—The LM3658B battery-management IC allows for the concurrent recharging of a Lithium cell from a powered USB hub while the board is in normal operation. The chip also works as an LDO without a battery attached.

QT1103's most useful features is that the sensor recalibrates each time it powers on. So, if you are using trailing wires for the sensor channels, which inevitably become tangled and start causing misfires, all that you have to do is cycle the power. You don't even have to untangle the wires.

HARDWARE CONSTRUCTION

I completed the schematic capture and a two-layer PCB layout with EAGLE v5.2. I packed the boards as densely as possible because I was charged per square inch for PCB manufacturing (see Figure 3).

I put ground planes under all of the sensor line tracks and kept them as short as possible because they carry an analog signal. The main construction-related challenge was soldering on the QT1103 chips because they are only available in QFN32 packages—no legs! After a little practice, I cracked this by pre-tinning all of the PCB pads (but not the pads on the IC), applying some liquid flux, carefully positioning the IC (a head torch helped me see), and then tacking down a few of the pads by heating the exposed PCB pads.

When your chip is in place, run the tip of your soldering iron down the chip's edge. The solder resist should stop bridges forming. You may need to build up the joints in some places. Then wipe down the edge again with your iron tip. If you get any bridges, keep cleaning your iron tip and wiping down the pads to remove the excess solder and leave a neat joint. Otherwise, reach for your solder wick!

I placed right-angle 0.1" headers connectors to access the sensor lines. The 20-channel device needed to be a stand-alone unit, so I placed 40-way 0.8-mm Tyco connectors on the boards to enable them to connect into a stack with the dsPIC, battery, and communications boards, which enabled a modular yet compact device. I can reuse individual boards from the stack for future projects.

On the battery board, I used a National Semiconductor LM3658B battery-management IC circuit with a rechargeable lithium battery. Figure 4 shows the battery-management circuit. The LM3658B was another of the leadless QFN packages, but I was getting the hang of soldering them. I was tempted to put a via in the PCB's base so I could hand-solder the ground pad on the chip's base. But in the end, after soldering using a fine bit, I used a Weller Pyropen with a narrow hot air nozzle to make sure it was all stuck down properly. Any problems that I had with this circuit were solved by applying a little more hot air to the QFN package with the Pyropen.

The output voltage from the management IC is fused and regulated. I always put fuses onto my circuits. I recharge the battery using a mini-USB connector on a board that clips onto the battery board. The "B" version of the LM3658 family allows you to use the battery management IC powered from the USB line without a battery attached, which is useful for testing purposes. I placed a ground solder pin on each board to ground an oscilloscope probe or multimeter in case debugging is required.

BLINKING LEDs

Light goes on. Light goes off. Electronics engineer happy.

Having an LED illuminate each time a channel was activated enabled me to verify that the sensor wires were correctly connected to their keypads and responding to touch. Plus, I can never have too many blinking LEDs! This would be an expensive luxury for a mass-produced commercial device, but these boards were designed to be as flexible as possible for research projects where the goal posts often move.

I incorporated a useful Microchip Technology MCP23S17 port expander IC to enable the maximum number of flashing lights for the minimum amount of interface. I sprung for an extra data line because I used the SPI incarnation of the MCP23S17 as opposed to the I^{2}C version. Figure 3 shows how I connected the MCP23S17.

The chip has 16 channels of I/O divided into two banks (A and B) of eight channels. Each channel can source or sink up to 25 mA—plenty enough to light up an LED. There are three SPI address lines on the chip (A0, A1, and A2), but by default they aren't used. They are enabled by enabling the relevant register bit. I was using all of the ports as outputs to drive LEDs, but you can also configure any of them to be an input and read the status via a SPI.

To use the MCP23S17 to light up LEDs, several internal registers need to be set up. The ports are configured as outputs and then the relevant bits of the output port must be set high. In the chip's default state, the register address toggles automatically between the two eight-port status banks, reducing the amount of commands needed. Refer to the code on the *Circuit Cellar* FTP site to see how I implemented

I hacked up the manufacturer's demo board to prove the capability of the technology. I then produced custom PCBs. Next, I designed a 20-channel device using two QT1103 chips. I followed that up with a 100-channel device using five of the 20-channel sensor boards.

this. I plan to use the versatile chip for future projects.

Obviously, with 20 channels and a single 16-port expander I was still a few channels short! For the 20-channel device, I used some spare I/O lines from the dsPIC to enable each sensor line to have a dedicated status LED. Once I got onto the 100-channel device, I didn't have enough spare lines to display each channel with a separate LED. So, some of the LEDs were used for two channels—which worked fine for testing that all of lines were working.

COMMUNICATIONS

The 20-channel design required Bluetooth communications. I used a Linkmatik 2.0 DIP module, which interfaces to the dsPIC using a standard UART.

The 100-channel multi-board design required a USB interface to a PC. I used an FTDI FT232RL UART-to-USB IC and a miniature USB connector. The same UART communication lines from the dsPIC and C code worked virtually unchanged for both communication devices as they are both seen as a standard UART interface by the controller.

I placed a 10-pin Micromatch connector on each of the 20 channel sensor boards so they can connect to a connector/power/USB board. The connectors are keyed and locking, which makes for a more reliable system than the traditional 0.1″ header connectors so beloved of research instrumentation. The dsPIC controller board connects to the back of this board using 40-way Tyco 0.8-mm pitch connectors. To enable all of the boards to work together, the SPI addressing on the MCP23S17 boards had to be enabled. Each of the sensor boards used different data lines for communications. No one was more surprised than me when it all worked!

CODING

I use the Mikro C_dsPIC compiler. A free version is available from Mikroelektronika, the same folks who built my dsPIC development board. It comes with a good IDE and a range of useful libraries. I had to implement a software UART interface for each of the QT1103 chips. The software UART library allowed me to define the same I/O line to both transmit and receive without throwing up errors. The IDE interfaces directly with the Mikroelektronika ICD programmer, which I used, but you can import the hex file to MPLAB and use the Microchip ICD2 without any problems. If you make your own programming cable, you can

Listing 1—When the *CHANGE line goes low while the 1-W line is still held high, a service routine is called to obtain the status for all of the 10 sensor channels for the chip. The status is displayed on the board's LEDs and transmitted via UART to the user.

```
// services touch sensor when a new sensor status is detected
  void service_QT1103(unsigned int *port, short uart_port, short board)   {
       unsigned short bytea=0x00, byteb=0x00;
       int reca, recb;
       soft_uart_init(port, uart_port, uart_port, UART_BAUD, 0); // set up uart on 1W line
       soft_uart_write(0x50);                                  // write 'P' on 1W line
       bytea = soft_uart_read(&reca);                          // read sensor status byte a
       byteb = soft_uart_read(&recb);                          // read sensor status byte b
       Display_bytes(bytea, byteb, board, port);               // display to LEDs
       code_sensor_status(bytea, byteb, port, board);          // transmit status
} // end service_QT1103

// main while loop
while (1) {
// Board 1
   if (!bit_read(PORTD,CHA1)&& bit_read(PORTB,ONEWA1)) {          // QPROX sensor A /CH goes low
       service_QT1103(&PORTB, int_onewa1, board1);
       }
   if (!bit_read(PORTD,CHB1)&& bit_read(PORTD,ONEWB1)) {          // QPROX sensor B /CH goes low
       service_QT1103(&PORTD, int_onewb1, board1);
       }

}  // end while
```

use the Microchip ICD2 directly on the same header pins as the Mikroelektronika ICD uses.

The main servicing loop is shown in Listing 1. Initially, I used a button debounce algorithm on the *CHANGE line, which triggered when that line went low. I found it more reliable and faster to look for that line to go low at the same time as the 1-W data line was still high—especially on the 100-channel design. Using this method of checking two lines allowed individual boards to be removed from the 100-channel device and for the remainder to continue to operate correctly.

The `service_QT1103` function sets up a soft UART interface on the 1-W line for the chip and writes a character "p" on the line to trigger the QT1103, which sends the sensor status in 2 bytes. These bytes are passed to two further subroutines. `Display_bytes` displays the activated channels on the board's LEDs. `code_sensor_status` adds board and sensor identification bits to the status bytes and then sends them to be transmitted to the user.

A lot happens each time a channel is triggered, but the sensor lines are only being triggered at the speed that a human can activate them, which is a glacial speed for today's microcontrollers. My main problem was that none of the hardware interfaces would work! I eventually found that the compiler had some errors in the definition file for my dsPIC, at least this was detailed on the manufacturer's forum. During the interim, I got all of the hardware working by using the software interface libraries. I knew the hardware interfaces worked because I had them all running using a different compiler during a previous project.

SWITCH TO TOUCH

The first piece of design advice I received was to minimize the number of mechanical switches in a design because they would inevitably become a mode of failure. Now I can do this cheaply and easily.

I am not going to predict the end of the mechanical switch, especially for power applications, but I do predict that we are going to see a lot more use of capacitive touch sensors. I can see many potential uses for this technology. One of the more imaginative ideas I have been asked about is the viability of implementing a touch sensor on the entrance to a bird-feeder to help monitor its use. I am not sure if my long-deceased cat would approve of this application, but he suffered for a noble cause by teaching me the fundamentals of how the body acts as a capacitor.

Matt Oppenheim (matt.oppenheim@gmail.com) holds an MSc in mechatronic systems engineering from Lancaster University. After working for 12 years in the marine and land seismic survey industry, he is now designing and prototyping instrumentation for research groups at InfoLab21, Lancaster University. Matt's technical interests include embedded design, but his first love is analog. Matt no longer owns a cat.

PROJECT FILES

To download code and additional schematics, go to ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2009/234.

RESOURCES

F. Block, H. Gellerson, M. Oppenheim, and N. Villar, "Touch-Display Keyboards and their Integration with Graphical User Interfaces," Demonstration, UIST, 2009.

Microchip Technology, "MCP23017/MCP23S17 16-bit I/O Expander with Serial Interface," DS21952B, 2007, ww1.microchip.com/downloads/en/DeviceDoc/21952b.pdf.

National Semiconductor, "LM3658B Dual Source USB/AC Li Chemistry Charger IC for Portable Applications," LM3658, 2006, www.national.com/ds/LM/LM3658.pdf.

Quantum Research Group, "QT1103 QTouch 10-Key Sensor IC," QT1103_3R0.03_0607, 2007, www.qprox.com/assets/Downloadablefile/qt1103_3r0.03(1)-15736.pdf.

SOURCES

Qprox QT1103 Capacitive touch sensor IC
Atmel Corp. | www.atmel.com

Linkmatik 2.0 DIP
FlexiPanel | www.flexipanel.com

FT232R USB UART
FTDI | www.ftdichip.com

dsPIC33FJ128GP706 Controller board and MCP23S17 IC
Microchip Technology, Inc. | www.microchip.com

MikroC for dsPIC30/33 and PIC24
MikroElektronika | www.mikroe.com/en/compilers/mikroc/dspic/

LM3658B Battery management IC
National Semiconductor | www.national.com

Toothpick 2.0 Bluetooth Transceiver
RF Solutions | www.rfsolutions.co.uk

40-way 0.8-mm Pitch connectors
Tyco Electronics | www.tycoelectronics.com