The Art of Capacitive Touch Sensing

By (Mark Lee, Senior Application Engineer, Cypress Semiconductor Corp.)

Executive Summary

Touch sensors have been around for years, but recent advances in mixed signal programmable devices are making capacitance-based touch sensors a practical and value-added alternative to mechanical switches in a wide range of consumer products. This article walks through a design example of a touch-sensitive button that can be actuated through a thick glass overlay.

Typical capacitive sensor designs specify an overlay of 3mm or less. Sensing a finger through an overlay becomes increasingly more difficult as the overlay thickness increases. In other words, as the overlay thickness increases, the process of tuning the system moves from science to art. To demonstrate how to make a capacitive sensor that pushes the limits of today's technology, the thickness of the glass overlay in this example is set at 10mm. Glass is easy to work with, readily available, and transparent so you can see the underlying sensor pads. Glass overlays also have direct application in white goods.

Finger Capacitance

At the heart of any capacitive sensing system is a set of conductors that interact with electric fields. The tissue of the human body is filled with conductive electrolytes covered by a layer of skin, a lossy dielectric. It is the conductive property of fingers that makes capacitive touch sensing possible.

A simple parallel plate capacitor has two conductors separated by a dielectric layer. Most of the energy in this system is concentrated directly between the plates. Some of the energy spills over into the area outside the plates, and the electric field lines associated with this effect are called fringing fields. Part of the challenge of making a practical capacitive sensor is to design a set of printed circuit traces which direct fringing fields into an active sensing area accessible to a user. A parallel plate capacitor is not a good choice for such a sensor pattern.

Placing a finger near fringing electric fields adds conductive surface area to the capacitive system. The additional charge storage capacity added by the finger is known as finger capacitance, C_F. The capacitance of the sensor without a finger present is denoted as C_P in this article, which stands for parasitic capacitance.

A common misconception about capacitive sensors is that the finger needs to be grounded for the system to work. A finger can be sensed because it can hold a charge, and this occurs if the finger is floating or grounded.

PCB Layout of the Sensor

Figure 1 shows the top view of a printed circuit board (PCB) which implements one of the capacitive sensor buttons in this design example. The button diameter is 10mm, the average size of an adult fingertip. The PCB assembled for this demonstration circuit contains four buttons with centers spaced 20mm apart. The ground plane is also on the top layer, as shown in the figure. The sensor pad is isolated from the ground plane by a uniform gap. The size of the gap is an important design parameter. If the gap is set too small, too much field energy will go directly to ground. If set too large, control is lost over how the energy is directed through the overlay. The selected gap size of 0.5mm works well for directing the fringing fields through 10mm of glass overlay.

Figure 1. Top View of PCB

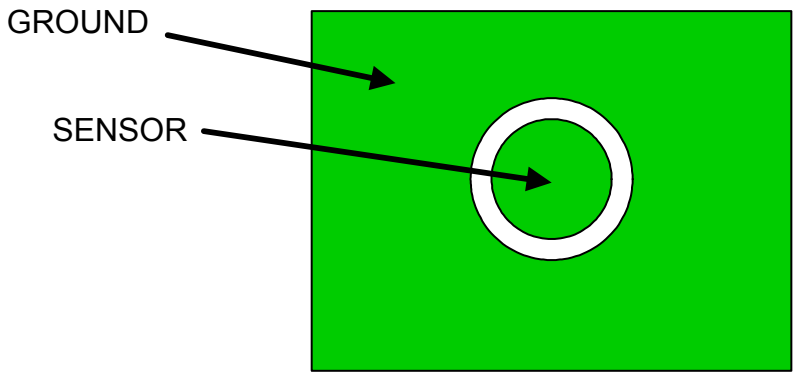
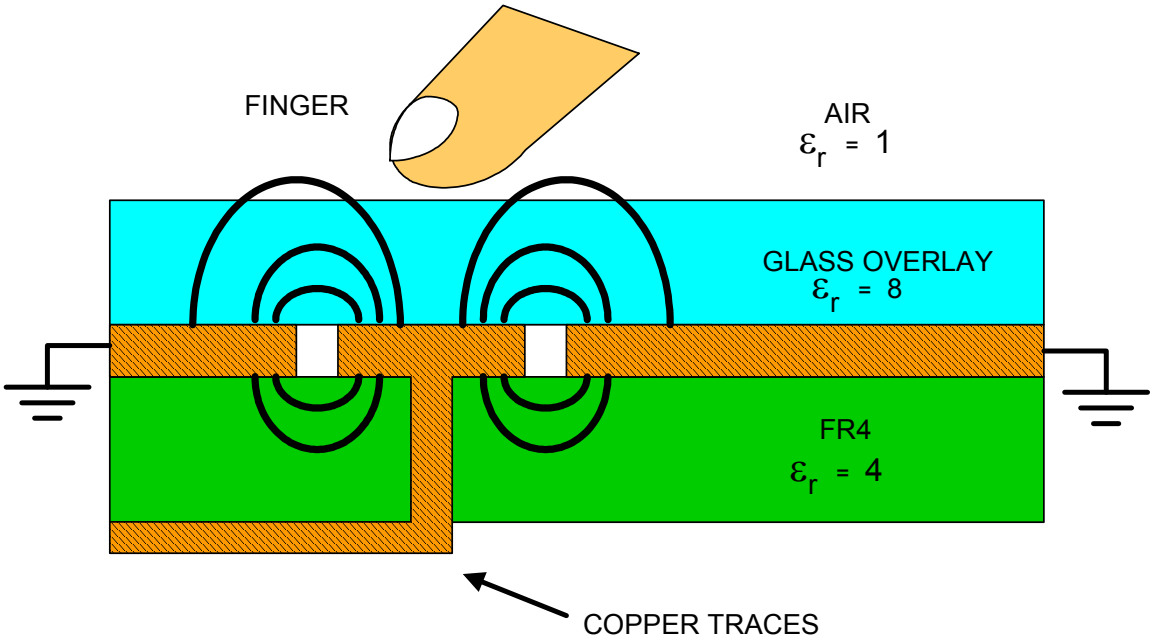


Figure 2 shows a cross-sectional view of the same sensor pattern. A via in the PCB connects the sensor pad to the trace on the bottom side of the board, as shown in the figure. The dielectric constant, ε_r, influences how tightly the electric field energy can pack into the material as the field tries to find the shortest path to ground. Standard window glass has an ε_r of around 8, while the FR4 material of the PCB has an ε_r of around 4. Pyrex® glass, which is commonly used in white goods, has an ε_r of around 5. In this design example, standard window glass is used. Note that the glass sheet is mounted on the PCB using the nonconductive adhesive film 468-MP from 3M.

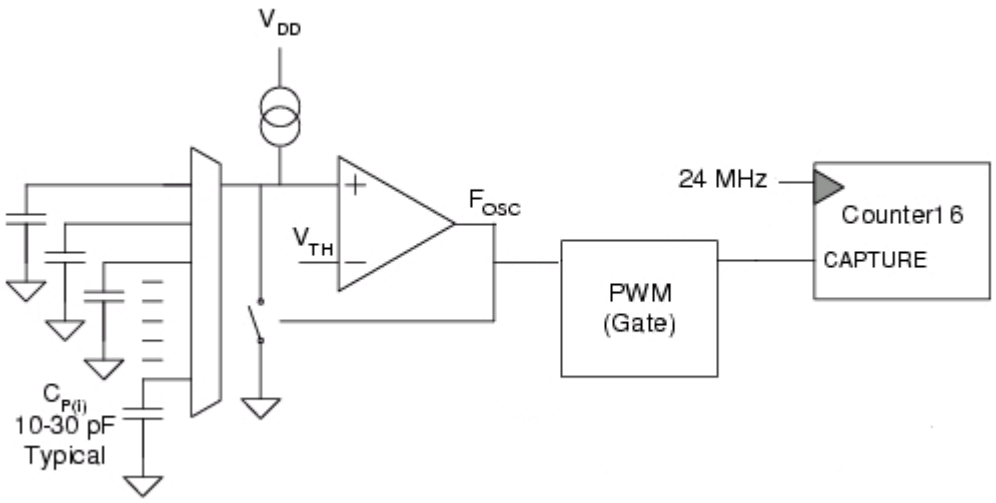
Figure 2. Cross-sectional view of PCB and overlay



CapSense 101

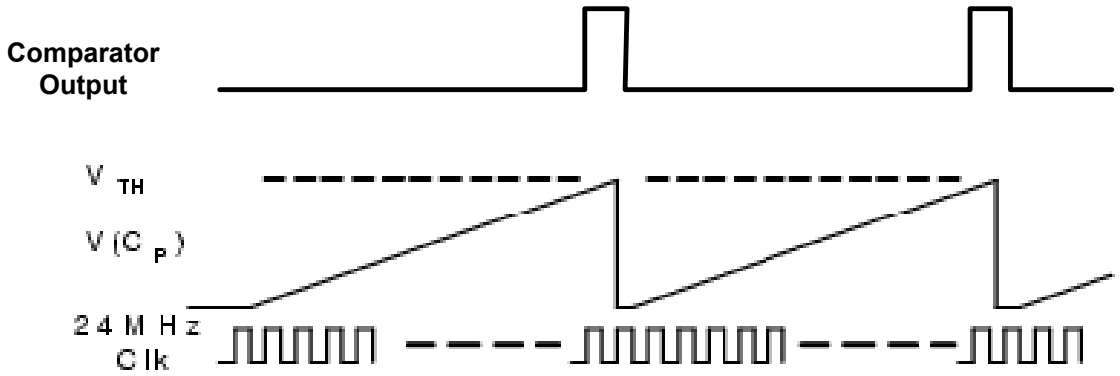
The fundamental components of a capacitive sensing system are a programmable current source, a precision analog comparator, and an analog mux bus that can sequence through an array of capacitive sensors. A relaxation oscillator functions as the capacitance sensor in the system presented in this article. A simplified circuit diagram of this oscillator is shown in figure 3.

Figure 3. The Relaxation Oscillator circuit



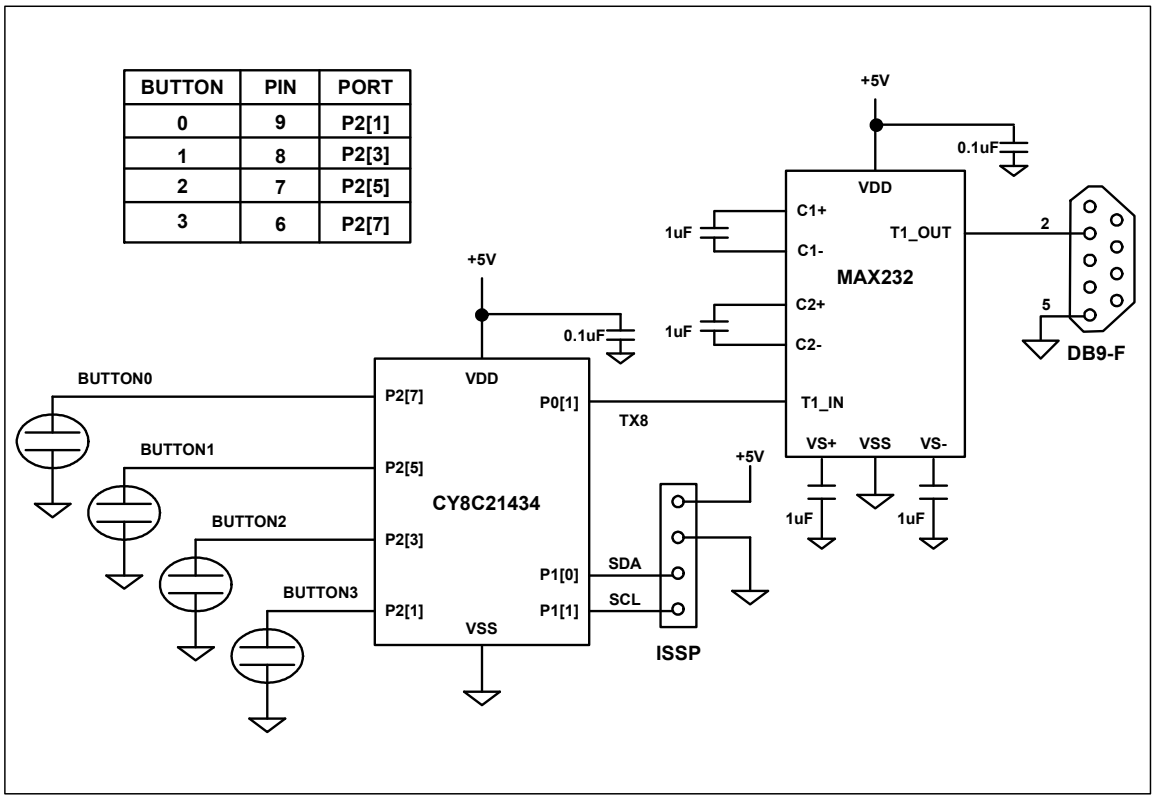
The output of the comparator is fed into the clock input of a PWM which gates a 16-bit counter clocked at 24MHz. A finger on the sensor increases the capacitance, thus increasing the counts. This is how a finger is sensed. Typical waveforms for this system are shown in figure 4.

Figure 4. Waveforms of the CapSense Relaxation Oscillator circuit



A schematic for an implementation of this project is shown in figure 5. For capacitive sensing and serial communication the circuit uses a Cypress CY8C21x34 series PSoC chip which contains a set of analog and digital functional blocks that are configured by firmware stored in on-board flash memory. A second chip handles RS232 level shifting to provide a communications link to a host computer, enabling data logging of capacitive sensing data at 115,200 baud. The pin assignments for the four CapSense buttons are shown in the table in figure 5. The PSoC is programmed through the ISSP header that contains power, ground, and the programming pins SCL and SDA. The host PC connects to the capacitive sense board through a DB9 connector.

Figure 5. Schematic for the capacitive sensing circuit



The PSoC is configured in firmware to operate from a 5V supply with an internally generated 24MHz system clock. The 24MHz clock is divided by 26 to provide a clock for the 115,200 baud TX8 module. The CapSense User Module is selected to run in Period Method, an operating mode in which counts are accumulated over a fixed number of relaxation oscillator cycles. In other words, the 16-bit counter value represents a period that is proportional to the capacitance of the sensor.

Listing 1 provides a listing of the system firmware. Most of the work involved in setting up a capacitive sensing system has been coded into a set of standard CSR routines that are called from a C program. For example, CSR_1_Start() configures the internal routing of the PSoC so that the current source DAC is connected to the analog mux, and the comparator is connected to the properly initialized PWM and 16 bit counter.

Listing 1: Firmware for a capacitive sensing system

```
//------------------------------start of listing-------------------------------------------------
//-----------------------------------------------------------------------------
// main.c, a CapSense program in C
//            A demonstration of Capacitive Sensing with PSoC
//            with a 10mm glass overlay
//-----------------------------------------------------------------------------
#include <m8c.h>                // part specific constants and macros
#include "PSoCAPI.h"        // PSoC API definitions for all User Modules
void main()
{
  //a flag that is set when a finger is on any buttons
            int bBaselineButtonFlag;

            CSR_1_Start();  //initialize CapSense user module
            TX8_1_Start(TX8_1_PARITY_NONE); //initialize TX8 module
```

```
            M8C_EnableGInt;  //enable global interrupts

            CSR_1_SetDacCurrent(200,0); //set current source to 200 out of 255
                                                    //use low range of current source
            CSR_1_SetScanSpeed(255); //set number of osc cycles to 255-2=253
                        while(1)
            {
                        CSR_1_StartScan(1,1,0);  //scan one button only, button 1 on P2[3]
           //wait for scanning of button to complete
                        while (!(CSR_1_GetScanStatus() & CSR_1_SCAN_SET_COMPLETE));

  //update baseline if required, set flag if any button pressed
                        bBaselineButtonFlag = CSR_1_bUpdateBaseline(0);

//data log the raw counts on button 1
                        TX8_1_PutSHexInt(CSR_1_iaSwResult[1]);
                        TX8_1_PutChar(',');

//data log switch mask... which switch is on?
                                                TX8_1_PutSHexInt(CSR_1_baSwOnMask[0]);
                                                TX8_1_CPutString(",");

//data log switch difference = raw counts - baseline
                        TX8_1_PutSHexInt(CSR_1_iaSwDiff[1]);
                        TX8_1_PutChar(',');

//data log update timer as a teaching aid
                        TX8_1_PutSHexInt(CSR_1_bBaselineUpdateTimer);
                        TX8_1_PutChar(',');

//data log the baseline counts for button 1
                        TX8_1_PutSHexInt(CSR_1_iaSwBaseline[1]/4);


                                                TX8_1_PutCRLF();


            }
}
//------------------------------end of listing---------------------------------------------------------
```

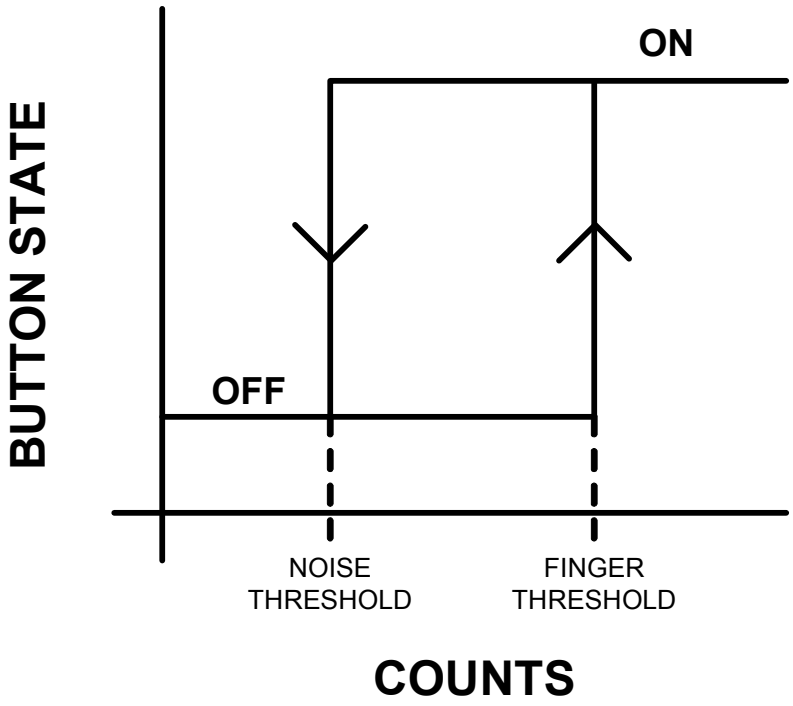
Turning the Sensor

Every time the function CSR_1_StartScan( ) is called in the program listed above, the capacitance of Button1 is measured. The raw count values are stored in the CSR_1_iaSwResult[] array. The user module also tracks a baseline for the raw counts. The baseline value of each button is an average raw count level computed periodically by an IIR filter in software. The update rate for the IIR filter is programmable. The baseline enables the system to adapt to drift in the system due to temperature and other environmental effects.

The switch difference array CSR_1_iaSwDiff[] contains the raw count values with the baseline offset removed. The current ON/OFF state of the buttons is determined using the switch difference values. This allows the performance of the system to remain constant even though the baseline may be drifting over time.

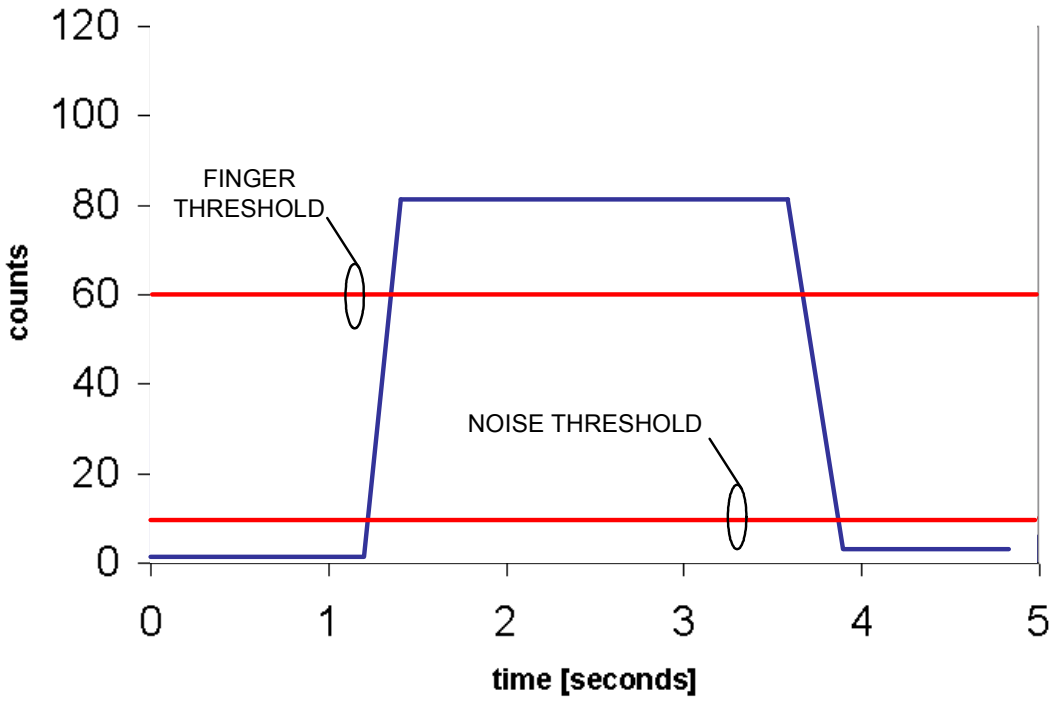
Figure 6 shows the transfer function between difference counts and button state that is implemented in firmware. The hysteresis in this transfer function provides clean transitions between ON and OFF states even though the counts are noisy. This provides a debounce function for the buttons. The lower threshold is called the Noise Threshold, and the upper threshold is called the Finger Threshold. Setting of the threshold levels determines the performance of the system. With very thick overlays, the signal-to-noise ratio is low. Setting the threshold levels in this kind of system is challenging, and this is part of art of capacitive sensing.

Figure 6. Transfer function between difference counts and button state



An idealized raw counts waveform for a 3 second button press is shown in figure 7. The threshold levels for this project are shown in the figure. The Noise Threshold is set to 10 counts, and the Finger Threshold is set to 60 counts. The noise component that is always present in real count data is not shown in figure 8 so that the threshold levels can be seen clearly.

Figure 7. Placing the threshold levels on a plot of raw counts with the baseline removed

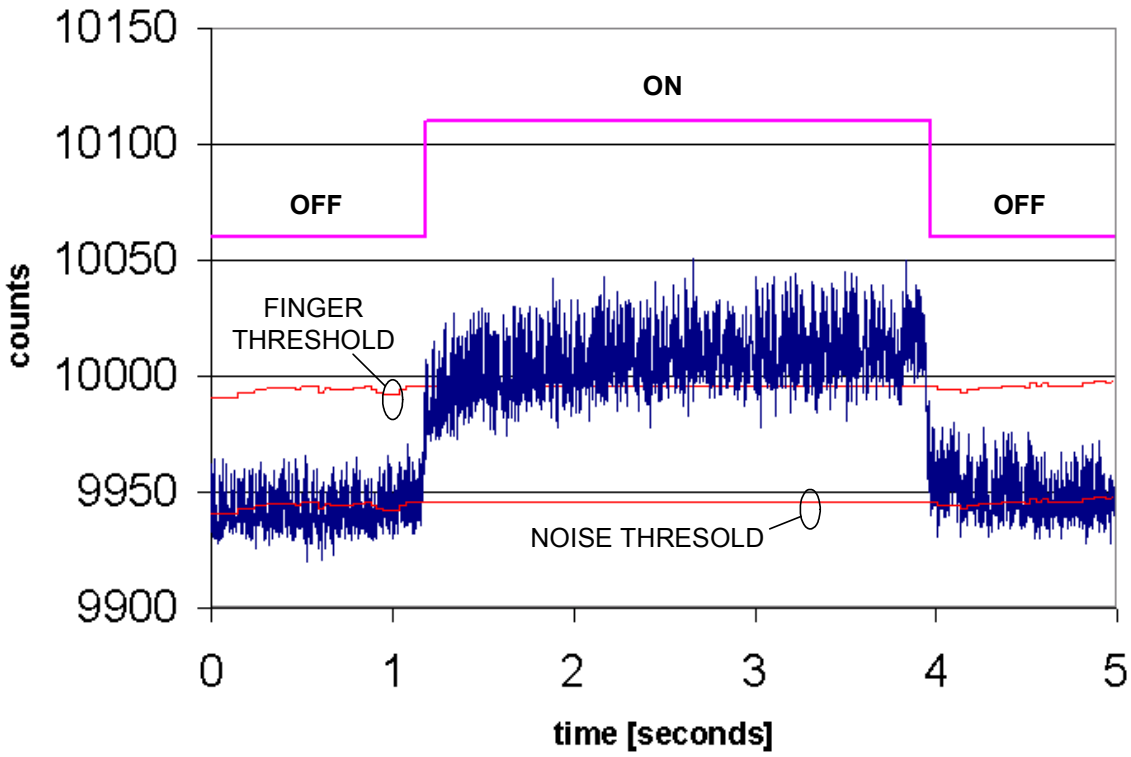


Part of the tuning process includes selecting the level of the current source DAC and setting the number of oscillator cycles to accumulate counts. In the firmware, the function CSR_1_SetDacCurrent(200,0) sets current source to level 200 out of 255 in low range of current source, about 14uA. The function CSR_1_SetScanSpeed(255) sets the number of oscillator cycles to 253 (255-2). Analysis of the raw counts and difference counts shows that the system has a parasitic trace capacitance, C_P, of around 15pF, and a finger capacitance, C_F, of around 0.5pF. The finger changes the total capacitance by around 3%. Acquisition of each raw count value only takes 500 microseconds per button.

Measured Performance

The measured performance of the capacitive sensing system is shown in figure 8. The difference counts were captured on the host PC via a terminal emulation program, and then plotted with the help of a spreadsheet. The finger is placed on the 10mm thick glass overlay for 3 seconds. The ON/OFF state of the buttons is superimposed on the raw counts. The button transitions cleanly between states even with the relatively noisy raw counts signal produced by the sensing through the thick glass. Note how the finger and button threshold are adjusted periodically as the baseline drifts. When a finger press is sensed, the baseline value locks its value until the finger is removed.

Figure 8. Measured performance of the sensor through 10mm of glass



Figures 9 and 10 show detail views of each state transistion. In figure 9, the button state is initially OFF. The first sample of the difference count over the finger threshold transitions the button state to ON. In figure 10, the button makes a transition to the OFF state with the first sample of the difference count below the noise threshold.

Figure 9. Close-up of transition to ON state

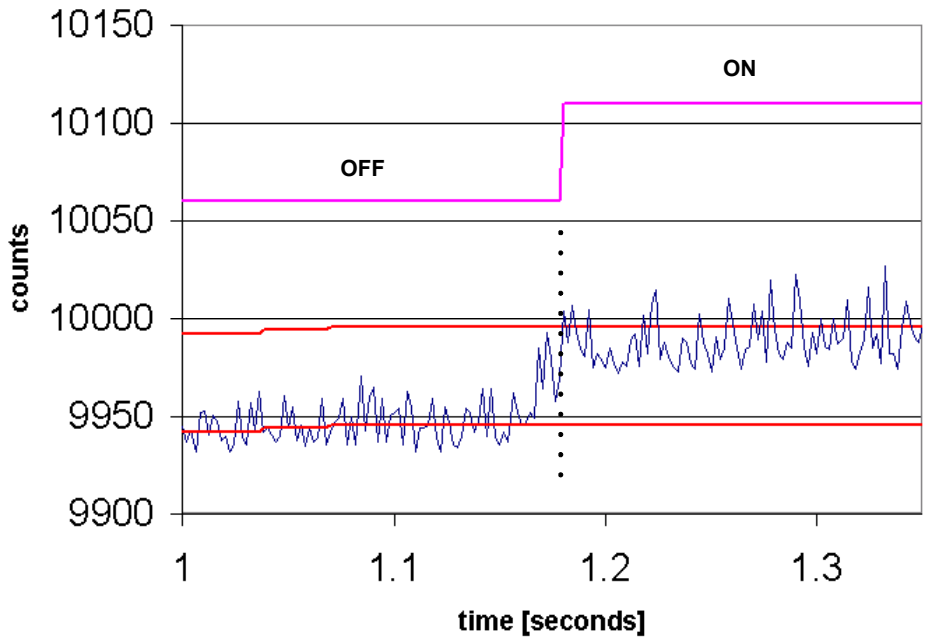
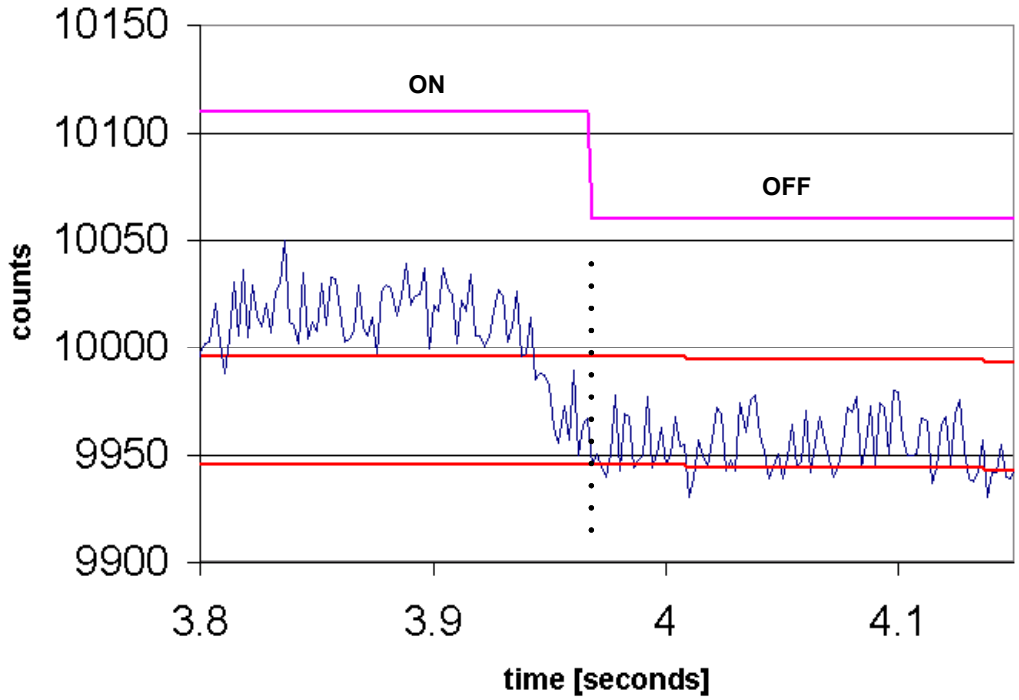


Figure 10. Close-up of transition to OFF state



Conclusion

The primary advantage of capacitance-based touch sensors over mechanical switches is that capacitance-based touch sensors do not wear out through long-term use, as do mechanical switches. Recent advances in mixed signal technology have not only brought down the expense of touch sensors to the point where they are cost-effective to implement in a wide range of consumer products, but also enable higher sensitivity and reliability of sensing circuits to increase overlay thickness and durability. Using the design techniques presented here, it is possible to sense a finger press through 10mm of glass, as well as achieve clean transitions between ON and OFF button states using the debounce method based on noise and finger thresholds, making capacitive touch sensors a practical alternative to mechanical components.

References

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com