

Techniques for Robust Touch Sensing Design

Author: *Burke Davison*
Microchip Technology Inc.

INTRODUCTION

The purpose of this application note is to describe the best design practices when developing capacitive touch applications for noisy environments. This application note will begin by defining the problems caused by noise, and explain how that noise typically affects systems. Hardware guidelines will then be provided to help maximize the natural signal-to-noise ratio (SNR) of the application. Software techniques are then covered to describe some of the common methods used to filter a sensor's signal to increase the SNR further, and then to make a decoding decision based on the behavior of the capacitive sensor.

The hardware design topics that will be covered are:

1. Button and slider pad design and spacing
2. Overlay material and thickness
3. Adhesive layer recommendations
4. Sensor trace layout and series resistance
5. Layout techniques for ESD protection
6. Power supply grounding scenarios
7. Choosing VDD and bypass capacitors

mTouch™ and RightTouch™ sensing solution systems have passed industry test standards in conducted and radiated susceptibility, and radiated emissions. This application note describes the important aspects of capacitive touch design which, when coupled with good printed circuit board (PCB) techniques, will allow these systems to continue performing in these extreme testing conditions.

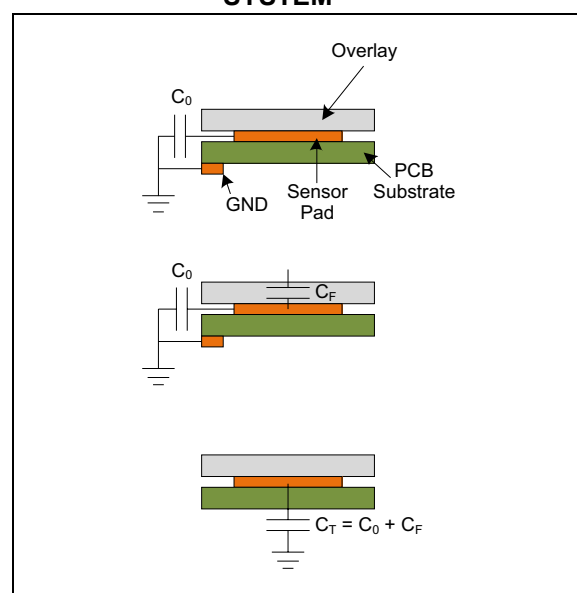
For information on the basics of capacitive touch sensing and other more advanced topics, visit the Microchip touch and input sensing solutions web site at <http://www.microchip.com/mTouch>.

Basic Capacitive Touch Review

Capacitive sensors are areas on a PCB that have been filled with copper and then connected back to the PIC® device using a trace. The PIC device will then measure the sensor in some manner that allows it to notice small shifts in capacitance typically caused by a user's finger approaching the sensor. The capacitance is continuously read in software and when a change occurs, the system will register a press on that sensor.

In [Figure 1](#), C_{BASE} is the capacitance value when no object is over the pad. This is referred to as the sensor's base capacitance. C_F is the capacitance change caused by a finger touch, and C_T is the total capacitance of the sensor.

FIGURE 1: CAPACITIVE SENSOR SYSTEM



The capacitances in this system can be calculated by the parallel plate capacitance shown in [Equation 2](#). It is important to understand that in real applications the capacitive sensor system is much more complex than [Equation 1](#). Generally speaking, the system could be considered as a network of capacitors, resistors, and inductors which are a result of the PCB, the overlay, the human body, and the environment. Therefore, it is very difficult to calculate the exact characteristics of a capacitive sensor system.

The recommendations in this application note are based on lab test results and are not absolute conditions. Users can always make their own adjustments based on the capacitance equation and their own application's needs.

EQUATION 1: PARALLEL PLATE CAPACITANCE

$$C = \epsilon_r \epsilon_0 \frac{A}{d}$$

Where:

ϵ_r = relative permittivity of the dielectric material

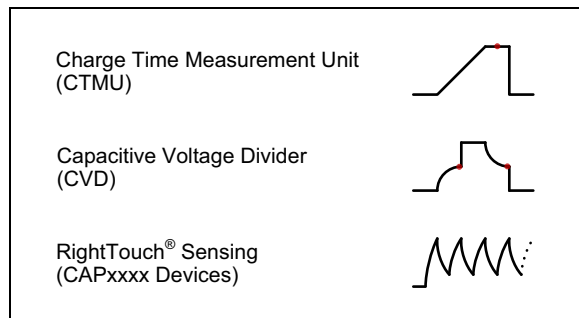
ϵ_0 = permittivity of space (8.854×10^{-12} F/m)

A = plate area in square meters (m^2)

d = distance between the plates in meters (m)

There are two fundamental methods for detecting a shift in capacitance using a microcontroller. The first is to use a voltage measurement where the system manipulates the pin of the sensor, to place a voltage based on the amount of capacitance on the pin, and looks for a shift in the voltage reading on the sensor. This includes methods such as Microchip's Charge Time Measurement Unit (CTMU) and Capacitive Voltage Divider (CVD). The alternative is to measure the sensor using a frequency approach, such as the RightTouch scanning method, which uses a pseudo-randomized frequency to sense changes in capacitance. The waveforms for all three scanning methods can be found below in Figure 2.

FIGURE 2: CAPACITIVE SENSING ACQUISITION WAVEFORMS



Acquisition Waveforms

This application note will focus on the hardware design of the system and the sections of firmware not involved in signal acquisition. For designs implementing the CVD or CTMU techniques, the source code available in the Microchip Library of Applications implements this for the designer. If using a RightTouch turnkey product, these techniques are built-in as part of the solution.

Noise Immunity vs. Low Power

When developing a capacitive touch system, it is important to know what your main goal should be from the very start of product development. For the majority of applications, how the system is powered will answer this question. For line-powered systems, conducted noise immunity is the main concern. For battery-powered systems, low power is the main concern.

It is also possible that some systems may overlap between these two regions. A cell phone that has the option of being powered through a USB cable is one example. The majority of the time, it would be concerned with low power; however, it needs to be careful of conducted noise when being powered through the main line. For this reason, only voltage-based acquisition methods should be used in these systems.

While noise immunity and low power are not mutually exclusive, focusing on one will require that design trade-offs be made to the other. For example, implementing a slew rate limiter filter to reduce susceptibility to conducted noise will require increasing the sample rate of the system, which will increase the overall power consumption. Lowering VDD is an excellent idea in low-power applications, but doing so will also decrease your noise immunity (see [Section, Power Supply Considerations](#)). This application note focuses on decreasing noise susceptibility and treats low power as a secondary goal. If low power is the main goal of the application, visit <http://www.microchip.com/XLP> for more technical details.

EFFECTS OF NOISE ON CAPACITIVE TOUCH SENSORS

Push Buttons vs. Capacitive Sensors

Before considering how to develop a robust capacitive touch application, it is important to understand the fundamental reason why noise is a concern. When using a mechanical button, the microcontroller's port circuitry decides whether the switch's pin is being pulled high or low and provides a single-bit digital result to the user. This result is then debounced to adjust for ringing, and the state of the button is based on the state of the debounce variable.

Capacitive touch sensor applications, however, are analog. The first clear difference is the need to manually perform the reading process. When using a mechanical switch, the microcontroller is able to read the pin using its internal hardware logic. For capacitive touch applications, separate hardware modules will need to be used to manipulate the sensor line. Whether it is using a voltage-based measurement or a frequency-based measurement, the analog result will be provided in the form of an integer value. This value is then typically filtered using different digital signal processing techniques to amplify the signal and attenuate the noise. The filter value is then sent through some form of debounce algorithm and a more complex decoding process. An extra layer of complexity is also added when the system is designed to perform in a closed loop manner, adjusting its behavior based on the sensor's current state.

The capacitive touch software process can be simplified into three distinct phases:

1. Acquisition

Using a voltage-based or frequency-based measurement technique to obtain a sample from the capacitive touch sensor.

2. Filtering

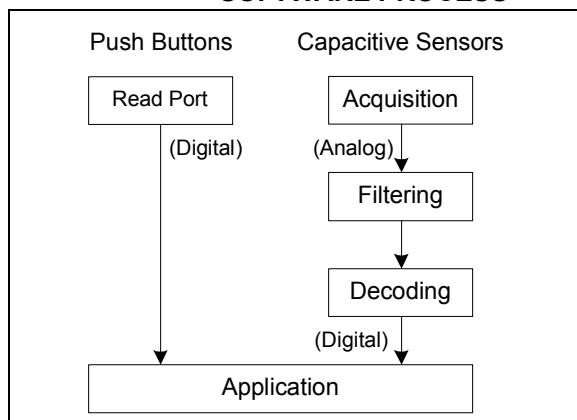
Manipulating the incoming sensor samples to increase the effective SNR of the system by attenuating the noise.

3. Decoding

Determining whether a sensor is pressed or released based on the current value of the sensor samples and the sensor's previous behavior.

Figure 3 illustrates the difference between push buttons and capacitive touch sensors, and labels the three main software stages of a capacitive touch system.

FIGURE 3: PUSH BUTTON VS. CAPACITIVE SENSOR SOFTWARE PROCESS



Conducted and Radiated Noise

'Conducted' and 'radiated' are the two main classifications of injected noise that can create instability in capacitive touch systems. Conducted noise is caused in systems that are powered externally from the device. This can include systems powered off the main-line power, desktop-powered USB devices, or any other situation that may mean the user is not sharing a ground with the application.

Radiated noise is a common challenge across all capacitive touch systems. In particular, if the capacitive touch sensor is a high-impedance input when being scanned, it essentially performs as a high-frequency antenna. Thus, electronic devices radiating electro-magnetic fields near the capacitive touch system will cause the readings to be affected. This can include cell phones, high-power communication lines, and fluorescent lights to name a few.

There are two main reasons why these two types of noise show up:

1. When a user presses on a capacitive touch sensor, he/she is becoming part of the system, so, if the user and the system are on different ground references, the system will interpret the user as an injected AC signal on the sensor.
2. Analog readings are susceptible to outside forces pushing them slightly in one direction or the other. The digital result of a mechanical switch is either high or low.

This application note will describe the different system design techniques that are recommended to overcome these two noise types. In addition to these guidelines, designers should be aware of the future working environment of the application and ensure there are no excessively noisy electronics nearby that may interfere with the system.

Capacitive Sensor Noise Behavior

Injecting noise on a capacitive touch sensor will cause the system to become more unstable. Voltage-based mTouch sensing solution reading methods such as CTMU and CVD will be affected differently than frequency-based reading methods, the RightTouch® scanning method. In voltage-based systems, the voltage of the sensor at a specific point in time is what determines the integer value of the reading. In frequency-based systems, the effect on the readings will vary based on the frequency of the injected noise and the frequency of the sensors oscillation.

NOISE BEHAVIOR: FREQUENCY-BASED ACQUISITION

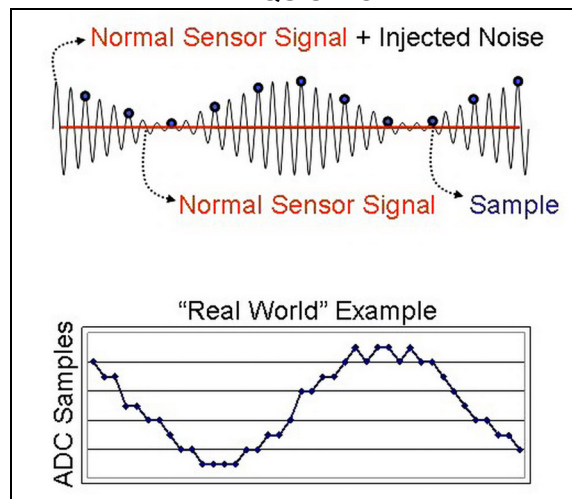
Frequency-based acquisitions methods must perform frequency-hopping techniques to eliminate harmonic noise concerns. This is handled automatically by the RightTouch turnkey products, and is always enabled. Additionally, multiple proprietary techniques are used to sense and adjust for noise in the device.

NOISE BEHAVIOR: VOLTAGE-BASED ACQUISITION

In voltage-based systems, injected noise can cause a positive or negative offset away from the natural sample value. If the sampling rate falls on a harmonic of the injected noise, resonance can occur. When this happens, the samples are falling on the peaks or valleys of the injected noise.

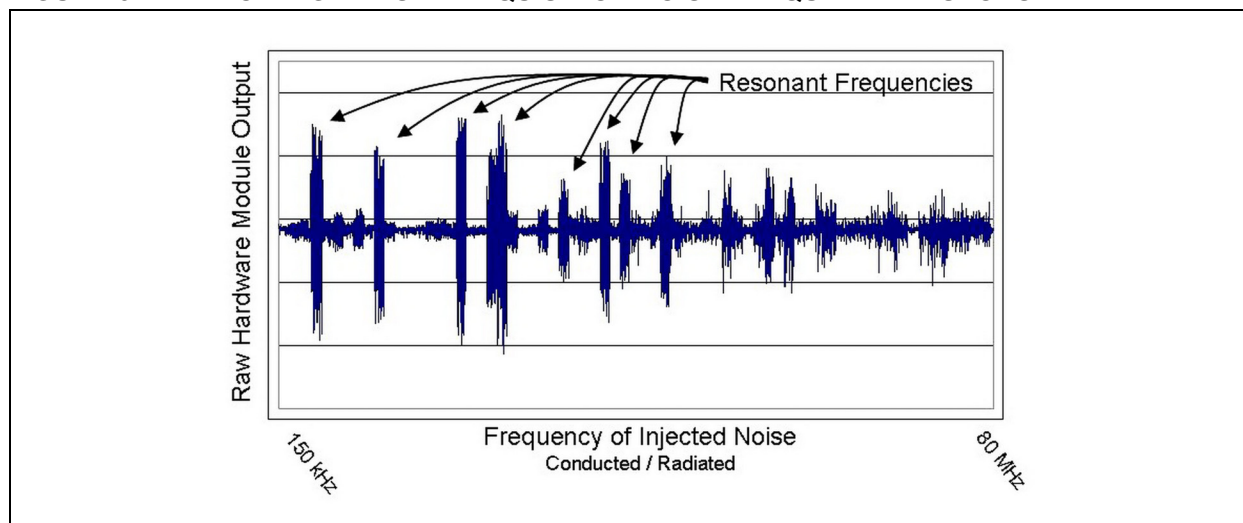
An example of this behavior can be seen in Figure 4. When sampling at one of these harmonics, the readings may all fall on the peaks of the noise or somewhere in the middle based on the starting time of the acquisition. Because of this, multiple readings at the same frequency will show a large amount of noise. This can be seen in Figure 5 where some of the noise frequencies are harmonics of the sampling rate and others are not.

FIGURE 4: VOLTAGE-BASED HARMONIC FREQUENCY ACQUISITION



The Microchip Library of Applications, available at <http://www.microchip.com/mla>, implements acquisition and filtering techniques to eliminate this behavior from the sensors' outputs.

FIGURE 5: VOLTAGE-BASED ACQUISITION NOISE FREQUENCY RESPONSE



Signal-to-Noise Ratio

In order to understand how hardware and software changes affect the system, it is necessary to have a way of measuring the signal's current performance. Sensitivity, or the amount that the system shifts, is not a sufficient measurement to define if a system is stable. For example, in a system where the average sensor output value is 20000 and a shift of 2000 is reached, you could simply subtract 18000 from each reading and claim a 100%, 2000 count shift was achieved. In reality, however, the shift or "signal" must be compared with the amount of noise. If noise was causing the sensor to drift by 1000 counts at any point in time, the system is in trouble.

One of the easiest ways to determine how stable a system is, or how much the system is affected by noise, is to look at its Signal-to-Noise Ratio (SNR). Just as it sounds, this is a way of measuring how strong the signal is when compared to unwanted disturbances of noise.

For the purpose of this application note, the SNR formula being used is defined in [Equation 2](#).

EQUATION 2: SIGNAL-TO-NOISE RATIO

$$SNR = \frac{|\mu_U - \mu_P|}{\sigma}$$

Where:

μ_U = mean value when not pressed

μ_P = mean value when pressed

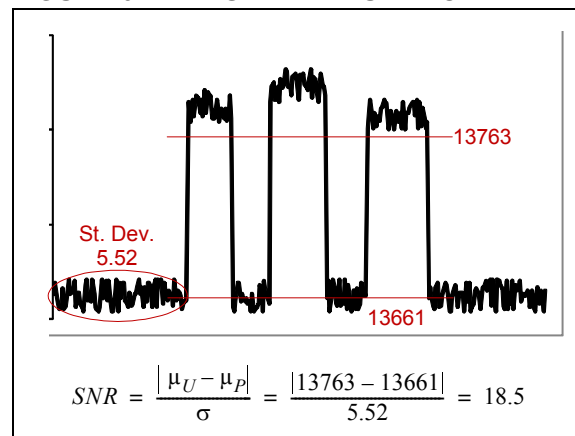
σ = standard deviation of the signal

The numerator of the equation is the amount that the system will shift when pressed or the 'signal'. The denominator is a measure of how much the noise is able to affect the readings. Using these as a ratio, a single number can be used to describe the quality of the sensor's signal by answering the question: How much shift do you require compared to the amount of noise you are trying to avoid?

If conducted noise is present on the system, the SNR will change when the sensor is pressed versus released. It may also change based on the frequency of the injected noise.

An example SNR calculation is shown below ([Figure 6](#)).

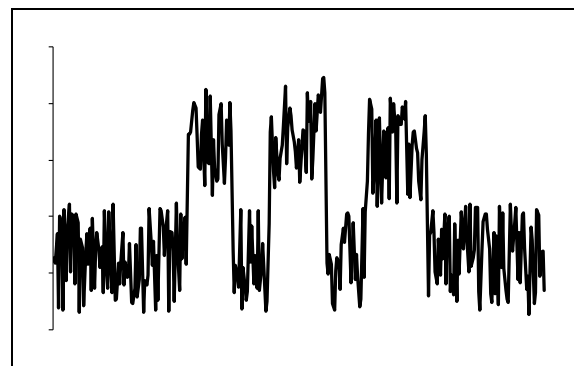
FIGURE 6: SNR CALCULATION



There are other formulas to calculate the SNR of a system. The important thing is to choose a method that provides consistent numbers across multiple measurements so informed decisions can be made about which of the hardware and software changes made are good and which are bad.

For reference, [Figure 7](#) is provided to show an example of what a signal-to-noise ratio of 3.5 would look like using [Equation 2](#). Note that, since the standard deviation of the noise and not the peak-to-peak value is being used, an SNR of 3.5 leaves little room to place a threshold. To be able to place a fixed threshold on the system, so that the pressed section plus its noise is completely separated from the unpressed section plus its noise, a system with an SNR of at least 7 is needed. In real applications, system SNRs should ideally be at least 15 to provide a higher level of reliability.

FIGURE 7: SIGNAL-TO-NOISE RATIO = 3.5



HARDWARE DESIGN

The hardware design of a capacitive sensor application is crucial to the system's overall success. The decisions made in this step of the process will determine how difficult it is to get a working, robust application. If the hardware design guidelines are followed, the sensor's sensitivity will be increased and it will be significantly easier to pass industry noise standards. Likewise, not following the guidelines will make success much more difficult and, in some cases, impossible. Keep this in mind while deciding which of these guidelines to follow in your designs.

Parallel Plate Capacitance Equation

The most important thing about hardware design is to remember that the basic capacitance equation, shown in [Equation 1](#), defines the relationship between hardware design decisions and the resulting sensitivity of the system.

For example, if the distance between the finger and the sensor is decreased by half, the sensitivity will double. If the area of the sensor is doubled (assuming it is still smaller than the area of a finger's press) then the sensitivity will also double.

Another important characteristic of capacitive touch sensors is the existence of parasitic capacitance which determines the sensors base capacitance, C_{BASE} . [Equation 3](#) explains how C_{BASE} can affect a system's sensitivity. You are only able to take a measurement of the total capacitance on the sensor, C_T , so the stronger the effect of C_{BASE} , the less you may be able to see C_F , the change in capacitance due to a finger.

An illustration of this system was previously provided in [Figure 1](#).

EQUATION 3: TOTAL SENSOR CAPACITANCE

$$C_T = C_{BASE} + C_F$$

Where:

C_T = total capacitance / measured value

C_{BASE} = sensor's base capacitance

C_F = finger's capacitance

[Equation 1](#) and [Equation 3](#) will be the basis of the hardware design guidelines for capacitive touch. The equations are simply physics. The guidelines are recommendations that will attempt to maximize your system's SNR and should be followed whenever possible. In some cases, an application may require that some of the guidelines not be followed. For example, a system might have size constraints or may require a thick covering material to protect it from damage. If this is the case, extra care should be taken to ensure a quality signal-to-noise ratio by closely following the other recommendations.

Design Goals

In general, to optimize performance of a capacitive sensor system using the mTouch or RightTouch solutions, designers should strive to:

- Achieve a large change in capacitance, C_F , relative to noise
- Minimize the base capacitance of the sensor, C_{BASE}
- Avoid conductive overlay material unless it is a metal-over-capacitive system
- Minimize overlay thickness

PCB Design Considerations

Board Material – No special requirements

Layer Thickness – No special requirements

Recommended two-layer PCB stack-up:

- **Layer 1** (top): Capacitive sensor pads and some sensor traces if they cannot be routed on the other layer.
- **Layer 2**: All components, any LED signal traces, power traces, and communication traces.

Recommended four-layer PCB stack-up:

- **Layer 1** (top): Capacitive sensor pads.
- **Layer 2**: Capacitive sensor traces.
- **Layer 3**: GND plane, except under capacitive sensor pads. Every effort should be made to keep this ground plane contiguous. This is especially true for the area under the capacitive sensing controller.
- **Layer 4** (bottom): All components, any LED signal traces, power traces, and communication traces.

For a PCB with more than four layers, always route the capacitive sensing traces on a layer close to the capacitive sensing pad layer, and place the GND or guard layer between the capacitive sensing traces and other signal layers.

Button Pad Design Considerations

Shape – No special requirements.

Size – 15x15 mm (0.6"x0.6") is recommended.

Pad-to-Pad Distance – 10mm (0.4") or 2-3x the overlay's thickness is recommended.

BUTTON PAD SHAPE

mTouch and RightTouch capacitive touch sensors work well with any button shape, including the most commonly used ones: square, rectangular, round, and oval. When designing a rectangular or oval sensor pad, a length-to-width ratio of less than 4:1 is recommended.

BUTTON PAD SIZE

'A' in Equation 1 is defined as the overlapping area. For capacitive touch applications, this means that you are limited by the smallest capacitive plate. If the sensor is smaller than a finger's press, the sensor's area is the limiting factor. If the sensor is larger than a finger's press, the finger is now the limiting factor.

You cannot change the user's finger size, but you can adjust the sensor size to maximize the sensitivity. The larger the sensor is, the higher the sensor's base capacitance will be. This will lower the sensitivity and allow more conducted noise to be injected into the system when a user presses. The smaller the sensor, the greater the chance that it is the limiting factor on sensitivity instead of the user's finger size. For this reason, the ideal sensor size is about the area of a finger press.

Best Option: The sensor size should be the same as an average user's finger press (15x15 mm or 0.6x0.6 inch).

Option 2: Design sensors to be smaller than optimal.

- Overlapping area, 'A' in Equation 1, is limited which reduces the maximum sensitivity.
- Adequate sensor separation will become more important to minimize the amount of sensor cross-talk.
- Use a thin cover overlay to gain some extra sensitivity.

Option 3: Design sensors to be larger than optimal.

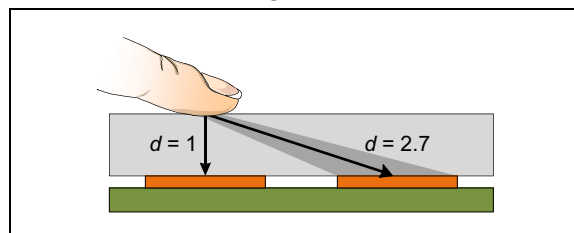
- Sensor base capacitance, C_{BASE} in Equation 3, can increase because of the increased proximity to ground which reduces sensitivity.
- Conducted noise disturbance is increased.
- Press shifts will vary by larger degrees because small fingers will cause less of a shift than large fingers due to less overlapping area, 'A' in Equation 1.
- Proximity sensing capability is increased.

PAD-TO-PAD SEPARATION DISTANCE

Crosstalk can become a challenge in capacitive touch applications if overlays are too thick or sensors are too closely placed together. Crosstalk is the unwanted shift of a different sensor from the one you are intending to press. If crosstalk is a problem in an application, the software must compare the two sensors and determine which sensor is "more pressed". This adds an extra step to the decoding process, increases the likelihood of error, and (depending on how it is implemented) can limit your system to one touch at a time. Following our guidelines will allow you to avoid implementing a system that must compare each sensor with every other sensor.

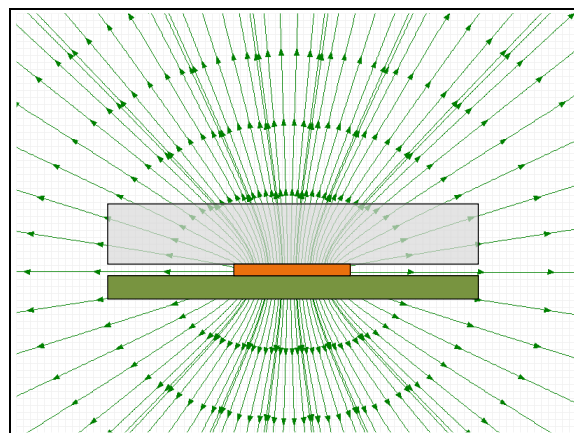
Figure 8 shows how a finger's press can affect the sensors located around the target sensor. By separating the sensors by 2-3 times the cover's thickness, the strength of the finger-to-sensor coupling is limited to a low and manageable amount. An alternative way of thinking about this relationship is to focus on the distance variable, 'd', in Equation 1. By separating the sensors, as shown in Figure 8, the crosstalk press is equivalent to pressing through an overlay that is much thicker. This results in a decreased crosstalk response from the system which increases the effective signal.

FIGURE 8: CROSSTALK CAUSED BY FINGER



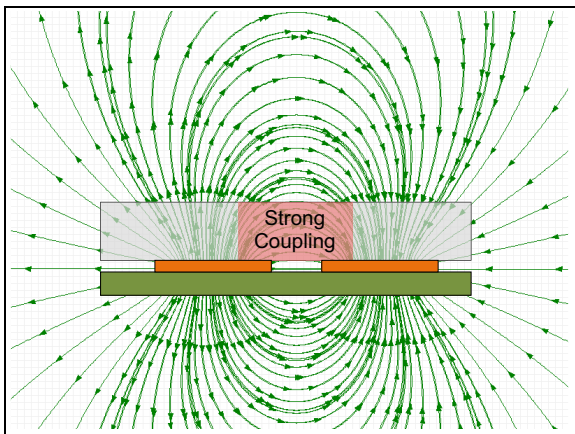
Sensor-to-sensor coupling is the other form of crosstalk that can negatively impact a design. Figure 9 shows how field lines will radiate from a capacitive sensor. The ability of those field lines to affect a neighboring sensor is based on the distance it travels and the material it is traveling through.

FIGURE 9: IDEAL SENSOR ELECTRIC FIELD



If the field lines are able to propagate only through the cover as shown in [Figure 10](#), the effect will be strong. The amount of crosstalk will be significantly reduced if the field lines must travel through the overlay, exit into free space, and then return through the overlay in order to affect a neighboring sensor, the amount of crosstalk will be significantly reduced. This is shown in the figure as the difference between the strong and weak coupling field lines. By following the first hardware design guideline, the field lines will be forced to travel through free space to reach a neighboring sensor and so the crosstalk caused by sensor-to-sensor coupling will be insignificant.

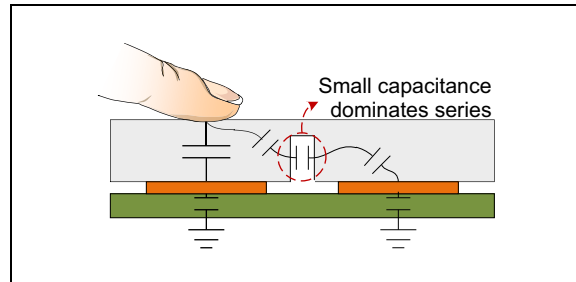
FIGURE 10: SENSOR-TO-SENSOR COUPLING



Alternatively, air gaps could be placed in the cover or PCB that will require the field lines to travel through free space. [Figure 11](#) illustrates this possibility. Notice how the crosstalk path now has a very small capacitor in series with the normal parasitic capacitances. This small capacitor will dominate the others and will result in an overall crosstalk shift that is very low. The larger the air gap, the smaller the capacitor, and the better this method will perform.

Finally, another option is to limit the sensitivity of the sensor by using nearby ground traces to block the field lines. Before using this technique, review [Section, Layout Design Considerations](#) to understand the recommended use of ground near sensors. Reducing the sensitivity of a system should not be a design decision that is made lightly and should only be used when the other possibilities have been exhausted.

FIGURE 11: AIRGAP CROSSTALK SOLUTION



By following this hardware design guideline, your designs will have reduced finger-to-sensor coupling and sensor-to-sensor coupling. This will result in a system that sees very little crosstalk which will allow the response time to speed up due to decreased processing overhead and the reliability of the system will increase as the sensors' signals become more immune to these negative effects.

Best Option: *Separate sensors as much as possible.*

Ideal minimum separation is 2-3 times the cover's thickness.

- The distance, 'd' in [Equation 1](#), between the sensors is kept high compared to the distance between the finger and the sensor, which results in reduced sensor crosstalk.
- Parasitic capacitance, C_{BASE} in [Equation 3](#), is kept low compared to the finger's capacitance, C_F , which results in increased sensitivity.

Option 2: *Create slotted air gaps in the cover.*

- The relative permittivity, ϵ_r in [Equation 1](#), between the sensors is lowered to "1", which results in decreased coupling between the sensors which decreases sensor crosstalk.

Option 3: *Use guard traces between the sensors.*

- The guard creates a low-impedance shield between the two sensors. Since the guard is at the same potential as the sensor being scanned, the field lines of the sensors are forced to move away from the guard and the neighboring sensor. This reduces the effect of crosstalk.

ACTIVE GUARD DRIVES

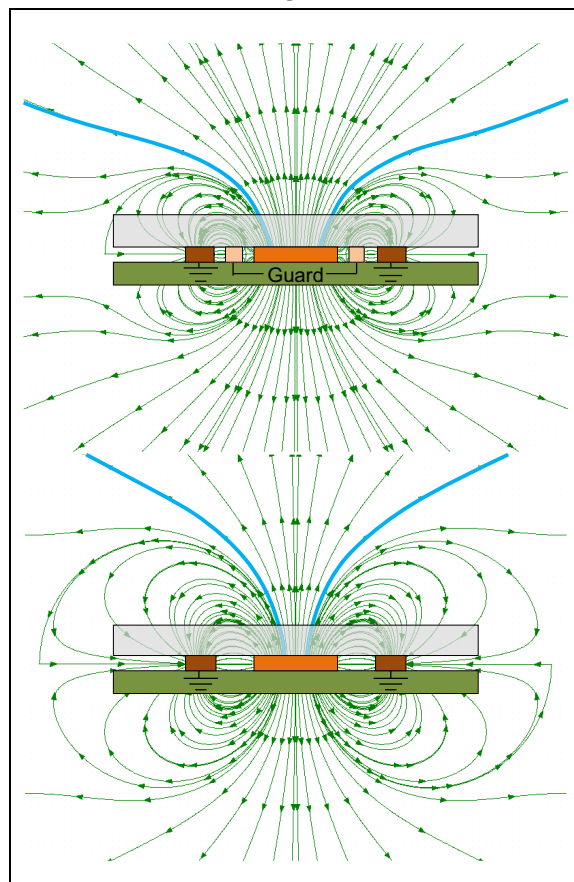
The base capacitance of a sensor determines how much sensitivity it will have. This capacitance can be lowered by making the environment around the sensor have the same voltage potential as the sensor while it's performing its waveform.

How exactly the guard is driven depends on the mTouch method or RightTouch device being used, but the voltage does not have to perfectly match the waveforms. Efficiencies of 60-70% can be achieved by simply driving an I/O in phase with the sensor.

- One guard trace can be used for all sensors. Sensors are scanned sequentially, so the guard can be actively driven for the sensor being currently scanned without affecting the others.
- Any power planes or low-impedance traces should be guarded from the sensor.
- Around the sensor's pad, the guard's trace should be about 1 mm thick and separated from the sensor by 2-3 mm.
- Following the sensor's trace back to the PIC device's pin, the guard's trace can be the same thickness as the sensor's trace: 0.1-0.3 mm. The separation of the guard trace from the sensor trace can be as small as 0.5 mm.

Figure 12 shows how an active guard can shape a sensor's field lines to increase its sensitivity.

FIGURE 12: ACTIVE GUARD FIELD LINES



MUTUAL DRIVES

Mutual drives are any I/O pin that is driven in phase with the mTouch waveform with the intention of measuring changes in the relative permittivity between the mutual drive (Tx) and the sensor (Rx).

When implementing a mutual sensor, there will be a base amount of coupling and then a change in the coupling when a new material is placed in the coupling path. In order to ensure maximum sensitivity, the mutual waveform should be driven in phase with the sensing waveform to ensure that the voltage shift due to additional capacitance is in the same direction as the shift due to the permittivity/mutual-coupling change.

There are two main situations where a mutual drive signal is advantageous to a design:

- If a piece of metal has the possibility of physically touching the sensor, driving the metal with a mutual signal will eliminate glitches on the sensor's reading when the short occurs.

(For example, the metal layer on a metal-over-capacitive system. This is not required, but beneficial if the metal layer can short to the sensor.)

- If the target being detected is isolated from the sensor's ground reference, placing a mutual drive near the sensor will allow the application to detect changes in the permittivity between the sensor and the mutual drive.

Depending on the type of capacitive sensing being used, the mutual drive may be driven differently. The differential CVD waveform requires that mutual drives be driven in phase with the waveform, but CTMU's single-ended waveform allows any board ground to behave like a mutual drive.

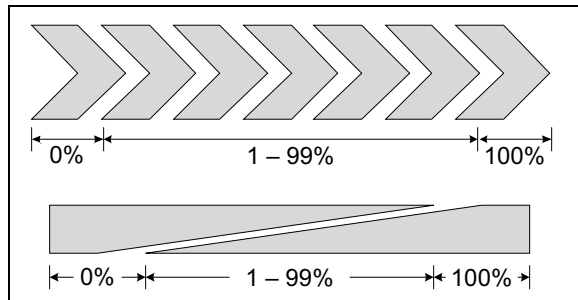
SLIDER SENSOR DESIGN

Sliders can be implemented using the mTouch Framework and Libraries, available in the Microchip Library of Applications, and on some RightTouch capacitive sensing devices, such as the CAP1114.

A typical slider shape is shown in [Figure 13](#). Similar to the individual capacitive sensing pad design, the distance between pads should be greater than 1.3 mm (~50 mils).

Theoretically, any pad shape used for a button pad can be used for a slider pad. However, the chevron shape shown in [Figure 13](#) will provide more linear (smoother) responses when a finger crosses from one pad to the next. It will also provide a clear direction indicator for schematic design, PCB layout, and assembling processes.

FIGURE 13: RECOMMENDED SLIDER DESIGNS



The seven-pad slider shown in [Figure 13](#) will provide good sensitivity, as well as enough accuracy for most applications, but a slider with less than seven pads could also be used for some applications. If the pad's size cannot meet the minimum requirement and reduced accuracy is acceptable, the number of pads in the slider can be reduced down to two.

The width of each pad and the distance between pads will usually be limited by the total length of the slider. The height of the slider will usually be limited by the physical dimensions of the machine.

Overlay Material Considerations

Thickness – As thin as possible. Less than 3mm, ideal

Material – Glass and plastics, typical. Dielectric constant between 2.0 and 8.0 recommended.

Adhesive – Thin, high permittivity, no air bubbles

In most applications, the capacitive sensor pads will be covered with an overlay to protect them as illustrated in [Figure 1](#). The material and thickness of the overlay, as well as the adhesive used, will affect the performance of the system.

OVERLAY THICKNESS

The thickness of the covering material is very important in affecting the sensitivity of capacitive touch systems. Product designers will try to make the covering material as thick as they can to increase the durability of the end product, but thick covers will decrease the system's sensitivity. [Equation 1](#) helps to explain why thick covers are such a concern for capacitive touch applications. As the distance between the PCB and the finger is increased, the expected capacitance shift is decreased.

The relationship between overlay thickness and sensitivity can be seen in [Figure 14](#). One thing to note about the graph is the importance that the permittivity of the material plays in defining the curves. A high permittivity material will allow for a larger sensitivity shift than an equally thick, but lower permittivity material. High permittivity has the negative effect of increasing the amount of crosstalk, however. If the covering material is highly conductive, the capacitive system may fail.

[Figure 8](#) shows the effect that a finger can have on a neighboring sensor. As the overlay's permittivity increases, so does this coupling.

If your application requires a thick covering material, consider creating a slot in the covering material where the sensor will be so the sensor can be placed closer to the user's finger. Conductive foam products are also available that can be used to fill the gap if the whole PCB cannot fit in the slot.

Best Option: Keep the overlay as thin as possible. Ideally, cover thickness should not exceed 3 mm to maximize sensitivity.

See [Figure 15\(a\)](#).

Option 2: Overlay is thicker than optimal, but the sensors' areas are increased to provide additional sensitivity.

See [Figure 15\(b\)](#).

Option 3: Overlay is thicker than optimal, but slots in the covering material are created to allow the sensor closer to the surface.

See [Figure 15\(c\)](#).

Option 4: Overlay is thicker than optimal, but slots in the covering material are created to allow EMI gaskets or springs to bridge the gap between the PCB and the finger.

See Figure 15(d).

When overlays are thicker than optimal:

- The distance, 'd' in Equation 1, between the sensor and the user's finger increases, which causes the capacitance between the finger and the sensor, C_F in Equation 3, to decrease which results in decreased sensitivity.
- Sensor-to-sensor crosstalk increases, as shown in Figure 10, due to additional sensor field lines being able to travel through the high-permittivity cover compared to the low-permittivity air.

FIGURE 14: OVERLAY EFFECT ON SENSITIVITY

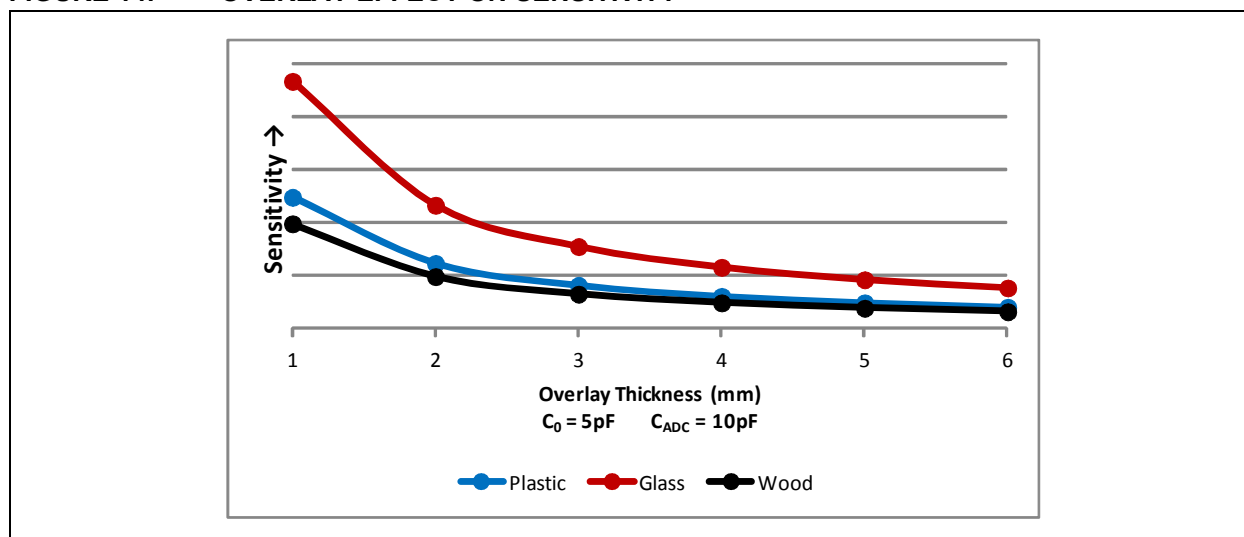
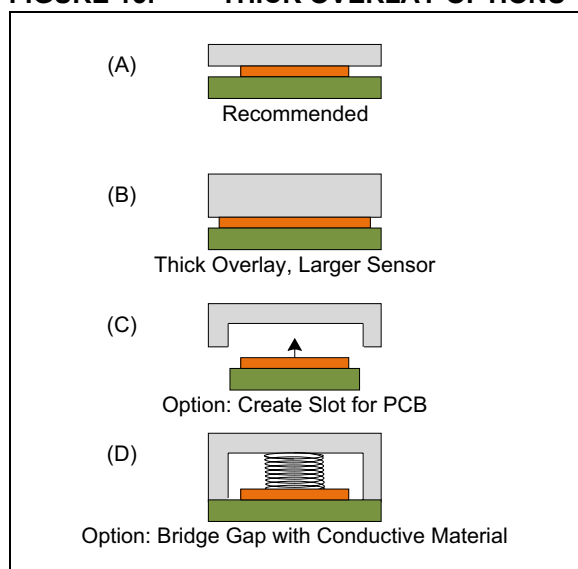


FIGURE 15: THICK OVERLAY OPTIONS



ADHESIVE SELECTION

Adhesive is used to secure the covering material to the PCB and is another important element to a robust capacitive touch system. Equation 1 will help to explain the necessity of a good connection. The relative permittivity of air is about one. Plastics are usually between two and three. Glass is about four. If you have

air separating the cover and PCB, your effective permittivity, ϵ_R , will be significantly decreased. For example, a 1 mm air gap will decrease your sensitivity to a half or a quarter of what it was. Remember that when three capacitors are in series, the smallest will dominate.

For systems using the metal-over-capacitive technique, it is especially important that a good adhesive is found for the application. Distances of tens of microns (10 micron \approx 0.4 mil) can make a significant difference in these designs. Talking to a representative from 3M or another adhesives manufacturer is recommended to ensure your choice is the best for your custom application.

There are several other important factors to keep in mind when choosing or working with a commercial adhesive:

1. Keep the adhesive thin in order to keep your sensitivity high. For most regular capacitive touch systems, 2 mil (50 micron) is a good thickness.
2. Always read the bonding instructions for the adhesive. Some data sheets specify a required amount of pressure, temperature, and time to achieve a secure, lasting grip.

3. Check the temperature limitations of your adhesive. In some environmental conditions, the glue can fail which will lead to unpredictable behavior from your capacitive touch application.
4. Be careful of air bubbles when applying the adhesive. If there are bubbles in the glue, your sensitivity will suffer the same as if you had an air gap between the cover and the PCB.
5. Make sure the adhesive type matches well with the covering material. Different adhesives are made for low surface energy and high surface energy plastics. Most adhesives will adhere to glass and PCB with few problems.

Some example adhesives that may perform well are:

High Surface Energy Plastics:

Example: ABS or Polycarbonate

3M's Adhesive Transfer Tape 467MP

Low Surface Energy Plastics:

Example: Polypropylene

3M's Adhesive Transfer Tape 9626

3M's Adhesive Transfer Tape F-9752PC

3M's Adhesive Transfer Tape 9122

3M's Optically Clear Adhesive (OCA):

8211, 8212, 8213, 8214, 8215

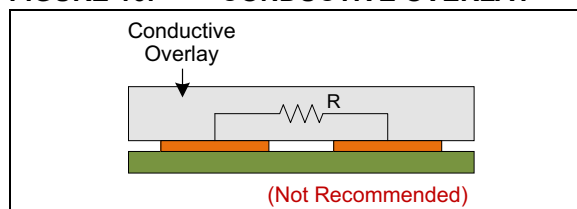
All of these will adhere to PCB and glass.

CONDUCTIVE OVERLAYS

Unless you are designing a metal-over-capacitive solution, highly conductive overlays are never recommended. However, in some capacitive sensor applications, the overlay may be conductive due to a conductive coating over the non-conductive plastics or due to filling with carbon to darken the overlay's color.

With conductive material over the capacitive sensor, the resistance of the material will add an equivalent resistor between neighboring sensors. When a finger touches a sensor, the capacitance change on one pad will affect the other untouched sensor to a varying degree. As the resistance decreases, the change in capacitance on untouched sensor pins becomes larger. If the resistance is too small, the signals can become too similar to determine the difference between a press on one sensor or the other. This is illustrated in Figure 16.

FIGURE 16: CONDUCTIVE OVERLAY



Using a conductive overlay is not recommended because:

- The amount of carbon in the plastic may change and then make the resistance change over time.
- The resistance from location to location could vary.
- The resistance from different product groups (different date codes) could vary.

Any of the above changes will affect the capacitive sensor input values and cause the device settings (such as the thresholds) to be incorrect.

If a design must use conductive materials, always test the overlay samples and determine the allowable range of conductivity.

Layout Design Considerations

The following rules will ensure a successful capacitive sensor PCB design. However, these are recommendations, not requirements.

- LED Output Traces should be isolated from capacitive sensor pads on different layers with a GND or guard plane between them.

Note: The same isolation should be observed for the capacitive sensing pads/traces and any other switching signals on the PCB, including signals generated by sources other than the mTouch or RightTouch device.

- Neighboring capacitive sensor traces appear as GND from the sensor's perspective. Routing two sensors traces in parallel is equivalent to routing both parallel to GND.
- Sensor traces cannot be parallel with LED output traces on the same layer or on adjacent layers. If a sensor trace must cross an output signal on adjacent layers, they must cross at a 90 degree angle.
- Sensor trace width: 0.1-0.2 mm
- Sensor trace length: Minimized, or guarded
- Sensor series resistance:
 - CVD :: 4.7 - 10 kΩ
 - CTMU :: 1 - 2.5 kΩ
 - RightTouch :: None required
- Minimum sensor trace separation: 0.1 mm
- Minimize via usage in sensor traces. This increases base capacitance.
- Unused RightTouch sensor and LED pins should be terminated either with a pull-down resistor or tied directly to GND.

Note: Ensure unused LED/GPIO pins shorted to GND are not driven by controlling firmware.

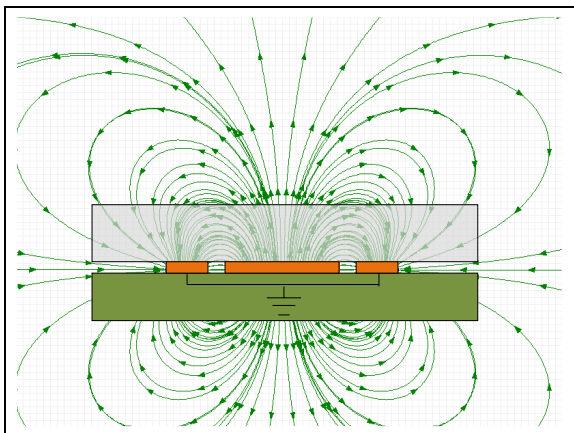
- Bypass capacitor(s) should be placed close to the capacitive sensing controller's VDD pin(s).

CAPACITIVE SENSOR COUPLING

The field lines of an ideal capacitive sensor would radiate away from the pad without coupling into any nearby components or low-impedance trace. When a user approaches the sensor, field lines able to reach out further will detect changes more quickly. In practice, coupling will always occur and will distort the ideal field line pattern in some manner. This behavior is shown in Figure 17.

The goal of any design should be to minimize the degree of coupling that is seen between the pad and its surrounding environment so that the sensor's field lines are able to behave as ideally as possible.

FIGURE 17: CAPACITIVE SENSOR FIELD LINES



CAPACITIVE SENSOR TRACES

Best Option: Keep sensor traces thin and short.

There are two main reasons to follow this advice. First, keeping trace lengths short will minimize CP which will increase the sensitivity of the system. Long traces are also more susceptible to behaving like antennas which will increase the noise floor of the application.

Communication lines should be kept away from sensor traces if at all possible. If not, run them perpendicular to the sensor traces to minimize their disturbance. You can also guard them with ground traces to couple the communication lines to ground instead of the more sensitive capacitive sensor traces. Avoid running capacitive sensor traces parallel with any noise-causing lines and keep them separated from ground and other capacitive sensor lines to reduce parasitic capacitance.

To keep sensitivities high and noise low, make sensor trace lengths short.

CAPACITIVE SENSOR SERIES RESISTANCE

Adding a series resistor to CVD and CTMU sensor pins will stabilize the sensor readings in high frequency noise environments. The resistor works with the internal pin capacitance to create a low pass filter. As the resistance increases, the cutoff frequency of the filter decreases. However, if the resistance becomes too large, the settling time of the waveform will increase which may allow low-frequency noise to be injected on the signal.

If using the CTMU acquisition method, do not exceed the module's maximum input impedance of 2.5 kΩ, or the scan rate will decrease. The lowest recommended series resistance is 1 kΩ.

If using the CVD acquisition method, the typical resistor value is 4.7 kΩ, but can vary from 1-10 kΩ based on your applications requirements. The settling delay of the sensor's waveform may need to be adjusted to allow the capacitors to fully settle before beginning the ADC conversion.

If using a RightTouch turnkey product, no series resistance is required.

ELECTROSTATIC DISCHARGE PROTECTION

Microchip's capacitive touch sensors are capable of withstanding high levels of electrostatic discharge (ESD) without physical damage. In addition, operational immunity from electromagnetic interference (EMI) is minimized through hardware and software filtering. However, excessive environmental conditions can produce false triggers, activate internal ESD protective clamps, or affect VDD and ground resulting in a device Reset. As such, it is important for electromagnetic compatibility (EMC) to be considered as early as possible in the system design process.

ESD typically has two distinct points of entry into a system:

1. Transient charge entering through a board-to-board connection, requiring circuit design solutions.
2. Transient charge coupling to the PCB, requiring layout and system solutions.

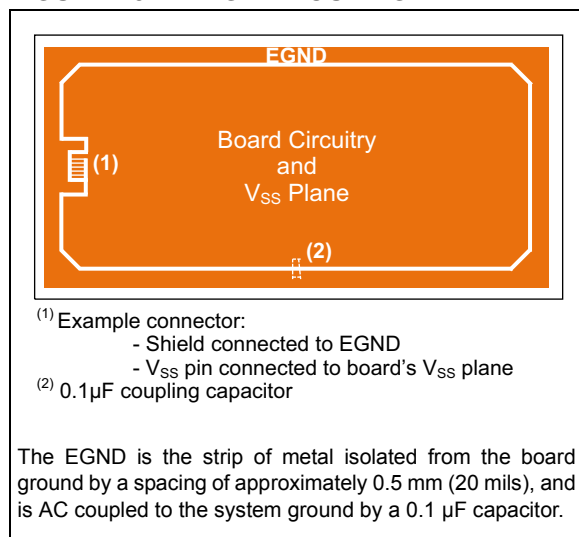
For transient charge entering through a board-to-board connection, there are several approaches that may help resolve this issue:

- Increase the impedance to high frequencies using ESD protection device(s) such as a series resistor, ferrite bead, or common-mode choke on the VDD and ground lines.
- Add these ESD protection devices to the communication lines.
- Add Transient Voltage Suppression diodes (TVS), also known as avalanche breakdown diodes, between VDD and ground to shunt the ESD current.

For transient charge coupling to the PCB, there are several approaches that may help resolve this issue:

- The ESD charge point(s) of entry must be determined. These can be visible air gaps in the covering material, areas where two pieces of covering material come together, around the edges of the covering material, etc.
- Specific metal on the PCB may be designed as an ESD-Ground (EGND) for conducting the ESD charge. The EGND should be exposed as a metal ring around the outer edge of the PCB to conduct ESD current to the chassis. This EGND should be terminated directly to the system chassis using conductive foam when possible. The EGND ring should be routed on all layers of the PCB.
- Route the signal ground between the EGND and all other traces on the PCB, using a spacing of 0.5-1 mm (20-40 mils).

FIGURE 18: GND ROUTING EXAMPLE



There are several additional points to remember:

- Direct coupling of ESD to sensor device pins should be minimized by directing energy to safe areas of the system, such as to chassis ground via a ground strap or similar means.
- Metal covering materials and surfaces are usually more difficult to protect against ESD than those made only of plastic. Additionally, plastics may only require "air discharge" ESD compliance (i.e., IEC 61000-4-2) whereas metal overlays and buttons can require "direct contact discharge" compliance. It is generally simpler to provide a safe ESD environment inside a plastic enclosure than one with exposed metal parts.
- Other PCBs, in particular ones mounted on the same overlay and directly connected to the capacitive sensing PCB, should be considered potential sources of ESD.

Example: A mechanical power button board.

Typically, these boards utilize the same V_{DD} as the capacitive sensor PCB. It is recommended to separate the voltage supply between the two PCBs by a small value resistor (~50 ohms).

- Radiation of ESD energy from nearby metal chassis plating should be considered a possible coupling mechanism. A useful mitigation technique for this type of coupling is to either shield the plate or shield the bottom PCB layer with conductive tape over a thin insulating material.
- If ESD energy cannot be redirected by mechanical means, TVS diodes can be connected to V_{DD} , MCLR, and long LED traces (especially those going off-board). The voltage rating of these TVS devices should match V_{DD} with board placement ideally being in between the ESD source and the capacitive sensor controller.
- A full layer board grounding and internal conductive coatings on plastic enclosures are among the best mitigation techniques to minimize the effects of EMI.
- Small bypass capacitors (5 pF-15 pF) can be placed on the sensor traces, located as close as possible to the controller pins to help shunt excess energy to ground rather than having it enter the device.

Note: Bypass capacitors on sensors are not recommended unless other techniques have failed to produce sufficient ESD protection. This technique may limit the sensitivity of the sensors and should not be used on proximity sensors.

Power Supply Considerations

Better, cleaner power supplies lead to better, cleaner capacitive sensor readings. The following topics should be considered when evaluating a power supply for your design.

GROUNDING SCENARIOS

There are three coupling paths that are affected when a user approaches a capacitive sensor. There is a change in coupling between the capacitive sensor and the board's ground due to the close proximity of the finger. This is a localized effect and will not be affected too greatly by the power supply or board layout.

The second coupling path connects the sensor and earth ground through the user's body. It adds capacitance to the sensor and is one of the primary paths for conducted noise injection. If this is an open circuit because the user is not sharing a ground with the board, the capacitance added by this path is negligible.

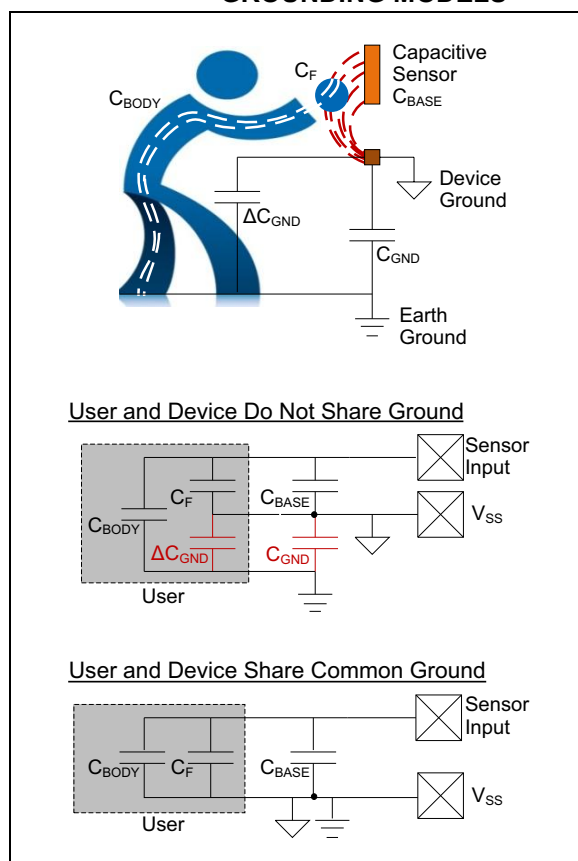
Finally, the third coupling path is the amount of additional capacitance to earth ground that is added to the board's ground due to the user's body. If the board-to-earth capacitance increases, the effect of the sensor-to-earth coupling will be seen more strongly on the sensor.

This model, illustrated in Figure 19, leads to some very significant conclusions in order to best design a quality capacitive system.

1. The best sensitivity can be achieved by having a common ground between the user and the sensor.
2. If this is not possible, the capacitance of the body-to-earth coupling path should be maximized.

Shared-ground systems typically have twice as much sensitivity as systems that do not share a ground with the user.

FIGURE 19: CAPACITIVE SENSING GROUNDING MODELS



CHOOSING V_{DD} TO MAXIMIZE NOISE IMMUNITY

Best Option: Keep V_{DD} as high as possible to maximize noise immunity.

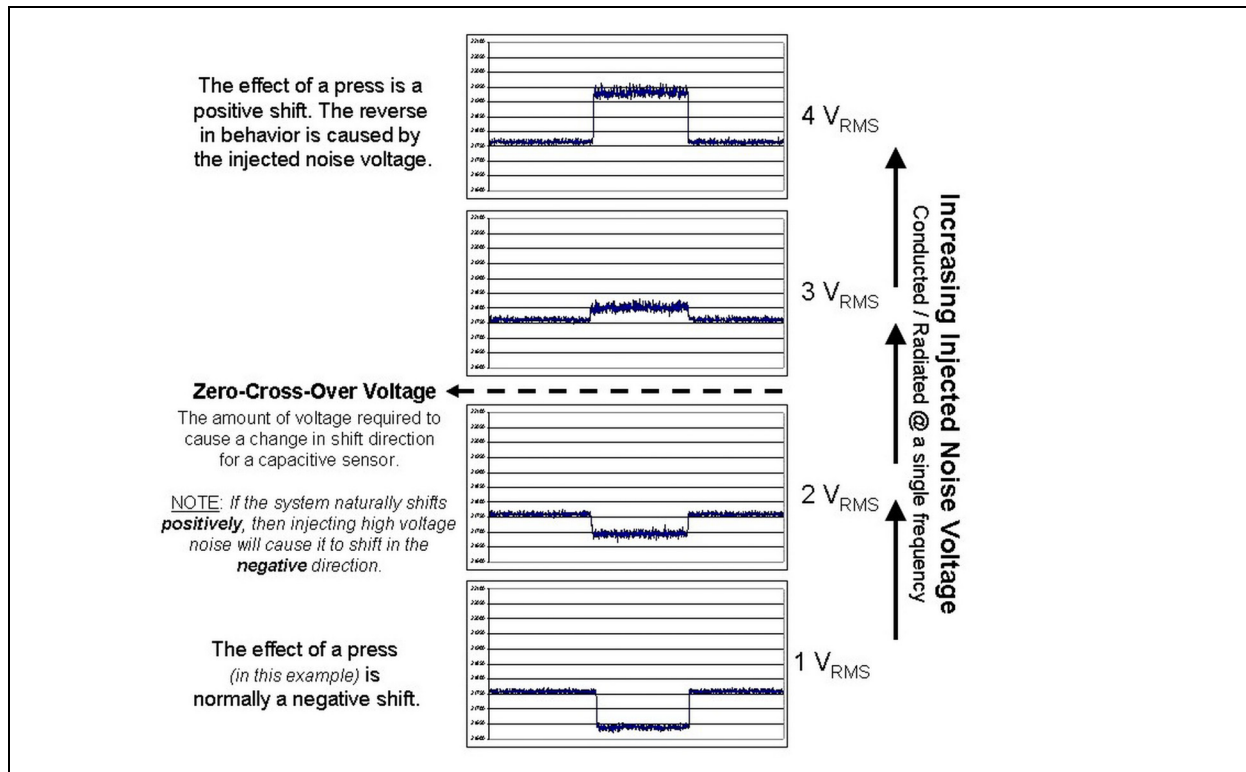
Although not ideal for low-power applications, high V_{DD} systems will perform better in conducted noise environments than low V_{DD} systems. This is due to the fact that all capacitive touch systems will eventually be overpowered by the injected noise as the voltage level of the noise is increased. Making V_{DD} a higher value will require a higher voltage level of injected noise before this happens.

In voltage-based acquisition systems, the behavior you are attempting to delay is the reverse press phenomenon. Figure 20 shows a regular sensor being injected with noise of increasing voltage levels. The noise begins adding voltage to the reading which eventually overpowers the normal capacitive sensing behavior and causes a positive shift when pressed.

BY-PASS CAPACITORS

At least one by-pass capacitor to ground ($\sim 0.1\mu\text{F}$) should be connected to and placed near each V_{DD} pin on the controller. Additional filtering is possible by placing another smaller value, smaller package capacitor in parallel.

FIGURE 20: REVERSE SHIFT BEHAVIOR WHEN INJECTING CONDUCTED NOISE ON A VOLTAGE-BASED ACQUISITION METHOD



SOFTWARE TECHNIQUES

The software methods described in this section of the application note are a subset of the techniques already implemented in the mTouch Framework, mTouch Library, and RightTouch capacitive sensing controllers. This section is provided for educational purposes. It is not necessary to implement these if using any of Microchip's capacitive sensing solutions.

Consider Your System Requirements

Your system's limitations are based on two main factors. First, your hardware design decisions will result in a base SNR for the application. If the base SNR is high, the software will not need to filter the signal as much and the detection process will be fast and easy.

If the base SNR is low, the software will need to heavily filter the signal and the detection process will need to go through more steps to make sure no false triggers are registered. The second factor to determining your limitations is the application's performance requirements. If the product must have a specific response time, or if there is a limited amount of memory available, then some software techniques may not be applicable.

For example, in gaming systems speed is the most important requirement. Care should be taken when choosing a software filtering technique that the response time does not suffer significantly. The code size should be kept small to allow for fast execution, and the sampling rate of our system may need to be increased.

When considering any of the techniques described in this section, remember that all of them have a cost in time, power, and memory usage. The benefits should always be weighed against the costs.

Sampling Rates

One of the first decisions when creating capacitive touch firmware is whether to trigger a new scan from the main loop or from the Interrupt Service Routine. For applications concerned with noise, the second approach is recommended.

Fixed-time sampling rates are important to the correct operation of filters and detection decisions should be based off a concept of how long, in real-world time, a new behavior has been measured. If the sample rate of the system is based on a non-fixed interval, like a function call from the main loop, other applications in the system could change the sampling rate. For example, a system that is actively controlling a power supply's output voltage will have priority over mTouch sensing due to its special timing requirements. If the power supply control application does not allow mTouch sensing to regularly scan, the system could miss a press or release.

The second decision that must be made is: What will the fixed sampling rate be? This is largely dependent on the acquisition method that has been chosen as well as the specific requirements of the system.

Voltage-based acquisition methods scan often and very quickly, while frequency-based acquisition methods scan over a longer period of time at a slower rate. Gaming systems that require a very fast response time may scan over 100 times a second, while a battery-powered proximity sensor may only scan three times a second until it notices a user nearby.

For many systems concerned about noise, the sampling rate and the decoding rate may be different. For example, a system could scan a sensor once every 50 μ s, continuously updating the filter, but only run the decoding sequence every 10 ms. This can reduce the amount of processing overhead while still allowing the system to adjust constantly to the changing environment.

Software filters and the decoding algorithm will need to be designed with the sampling rate of the sensors in mind. If not, the filters will become too fast and suffer from too little noise reduction or they will become too slow and may cause the signal to be dampened or slowed. Decoding algorithms that do not consider the sampling rate of the system may have problems achieving a required response time or may change states (on/off) too quickly.

Jittering the Sample Rate

VOLTAGE-BASED ACQUISITION METHODS ONLY

One of the problems that can occur in systems that use the Interrupt Service Routine to trigger a new scan is that the scans are then vulnerable to noise being injected at a harmonic of the sampling rate. Jittering will help to dampen high-frequency noise being injected on to the system either through radiated or conducted noise. [Figure 13](#) shows an example of what this can look like.

The simplest solution is to change the sampling rate by a very small amount each time a sample is taken using the "jittering" technique. For example, if you are scanning a sensor once every 400 μ s, you may decide to delay the reading by an extra 0-10 μ s (an amount that changes each time a sample is taken) to make sure you are not hitting any harmonics. Although this will technically change our sample rate, it will not change the average sampling rate and the change is insignificant compared to the total sampling interval.

In the jittering implementation below ([Example 1](#)), the Least Significant bits from our last ADC sample are used to create the random, short delay. The value is masked with `0x0F` to limit the maximum amount of delay that is possible.

EXAMPLE 1: JITTERING THE SAMPLE RATE

```
void interrupt ISR()
{
    if (T0IE & T0IF)
    {
        // Short Delay
        jitter = ADRESL & 0x0F;
        while(jitter--);

        mTouch_Service();
    }
}
```

FREQUENCY HOPPING

Frequency-Based Acquisition Methods Only

This is the frequency-based acquisition equivalent to jittering the sample rate on a voltage-based acquisition. It is important to change the sampling frequency of the waveform to avoid issues with harmonic noise. To keep the signal offset value consistent, the hopping procedure should be the same for every sampling period.

RightTouch capacitive sensing controllers will perform this automatically based on the level of noise in the system.

OVERSAMPLING

Oversampling is the process of using more than one acquisition sample per “sensor reading”. For example, in most systems, the Interrupt Service Routine determines when a new reading should take place. When this occurs, the system acquires a value and saves it as the new reading. To increase the stability of your readings, you could have the system acquire two samples off the same sensor by scanning it twice and then adding the two samples together to create one “sensor reading”.

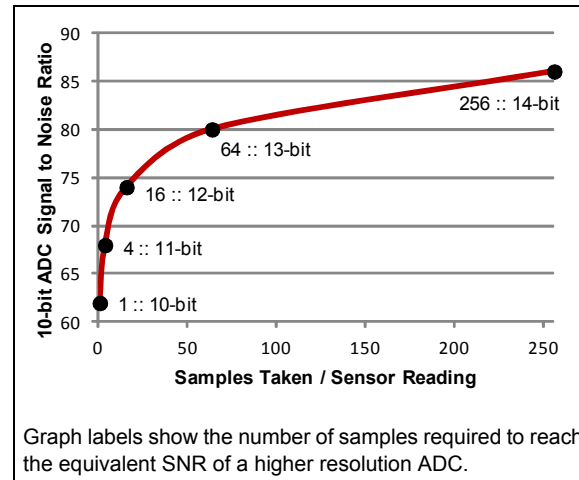
Note: ‘Sample’ is used to refer to a single scan of a sensor using a hardware module (e.g., ADC). ‘Reading’ is used to refer to a group of samples that have been added together, which are then sent as-is to the filtering and decoding routines.

There are several reasons why this technique is helpful to a system:

1. Sampling errors caused by impulse noise will not affect the system as much since each sample is only part of a full reading value. The system effectively averages out some of the errors during the acquisition process.
2. Samples do not have decimal values. By allowing the system to scan multiple times per reading, it is able to gain additional resolution on the signal.

The benefit of this method to the effective SNR of the system is shown in Figure 21. The clear diminishing return of oversampling should be considered against the system's time and power consumption requirements.

FIGURE 21: OVERSAMPLING TRADE-OFF: TIME VS. SNR



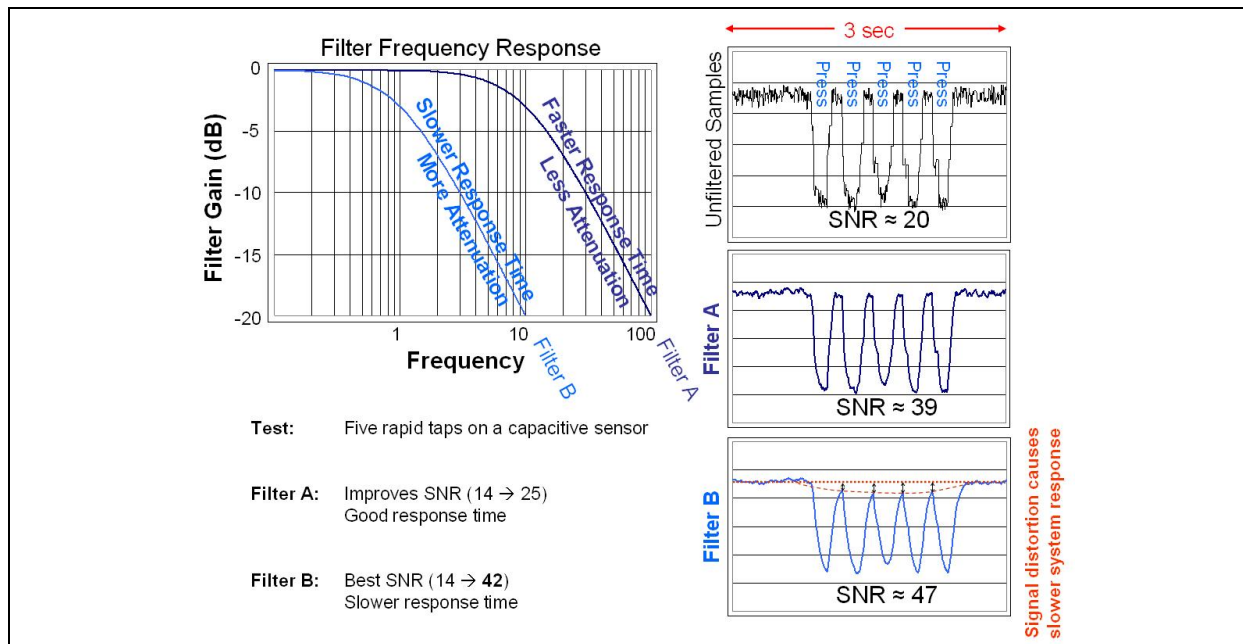
For more information about the effects of oversampling on an ADC conversion, refer to AN1152, “Achieving Higher ADC Resolution Using Oversampling,” available on the Microchip web site at <http://www.microchip.com>.

Software Filtering

Filters are algorithms that take an input signal and output a modified version of the signal. The function it performs is based on the type of filter it is. The bandwidth of a filter also plays a large role in how it performs. From a firmware perspective, the function of the filter is defined in the code structure and operations that are performed. The bandwidth is usually set by constants in the implementation that determine what number a value should be divided by, how many times to bit-shift left or right, and what coefficient should be used to multiply the result by.

With filters, there is a trade-off between noise reduction and response time delay. This trade-off can be visualized in the bandwidth of the filter as shown in Figure 22.

FIGURE 22: NOISE REDUCTION VS. RESPONSE TIME TRADE-OFF



If the filter's bandwidth is narrow, less noise will be able to pass through, but it may take a long time for the filter to follow the signal. On the other hand, if the filter's bandwidth is wide, more noise will be able to pass through, but it will follow the signal more closely. Combining multiple filters can allow the designer to get the benefits of each while limiting the negative impact.

There are three types of filters that are commonly used in mTouch sensing solution applications. More filters could easily be added to this list and may be more appropriate for a specific application, but these have been chosen because most designs will be able to use one or more of them.

The three filter types are:

1. **Slew Rate Limiter**

Used as the first input filter on new incoming samples to reject impulse noise and smooth the signal.

Implemented in the acquisition routine.

2. **L-Point Running Average**

Used to create a slow-updating (high time constant) baseline ("average") for each sensor as a reference point during decoding. Allows the system to track environmental changes such as temperature and humidity.

Implemented in the filtering routine.

3. **Low Pass Butterworth**

Used to reject white noise on the sensor readings while still maintaining a fast response time (low time constant).

Implemented on the 'reading' variable in the filtering routine before sending it to the decoding algorithm.

FIR FILTERS VS. IIR FILTERS

Finite-Impulse Response (FIR) filters take a fixed number of previous inputs and use them to create the next output.

Finite-Impulse Response Filter Benefits:

- Simple implementations
- Better filter stability
- Fewer concerns about integer precision

Infinite-Impulse Response (IIR) filters take the input and use it in combination with the previous output of the filter to determine the next filter output.

Infinite-Impulse Response Filter Benefits:

- Low memory requirements
- Low processing requirements

Ultimately, both filter types can be useful to an application. For capacitive touch systems, IIR filters can be used for slowly-updating environmental baselines. The filter should be designed so that it can handle impulse noise without becoming unstable. FIR filters can be used for quickly-updating sensor variables like the current reading.

FILTER: SLEW RATE LIMITER

The Slew Rate Limiter (SRL) filter's main design goal is to reject impulse noise from sensors' readings. Sometimes referred to as a 'decimation' filter, implementing the SRL filter requires a specific scanning technique that will possibly change the sample rate of your design.

The concept of the SRL filter is simple. The PIC device is maintaining a “current reading” variable for each sensor. In most systems, when a new sensor reading is created, the “current reading” variable is replaced by the new value. In an SRL filtered system, when a new reading value is generated, the “current reading” value is then either decremented or incremented by one, based on whether the latest reading is higher or lower than the “current reading” variable. For example, if a sensor’s current reading is 200 and the next acquisition results in a value of 300, the system will update the “current reading” to 201. In order for the system to reach a current reading of 300, the next 99 scans must be higher than the current reading.

This behavior is very beneficial because it limits the influence of each sample. If impulse noise is affecting the system, a single impulse-noise-affected reading will only cause 1 bit of noise on the reading variable. On the other hand, because you are updating the current reading variable so slowly, you need to update the rate of the samples. When a user presses on a sensor, the current reading variable needs to be able to move with the finger’s capacitance at a fast rate. There is also no need to access the decoding function of the system after each individual reading, since it can only shift by 1 each time.

There will be several parts to the SRL filter implementation to take care of these special requirements. First, the system will scan based on a timer’s interrupt at a fast rate. After each of these scans, it will run the SRL filter to increment/decrement the “current reading” variable. After the Nth sample, a flag is set that allows the decode function to run. An example code implementation of this filter can be seen below in [Example 2](#).

EXAMPLE 2: SLEW RATE LIMITER FILTER

```
#define SCANS_PER_DECODE 100

uint16_t reading;
uint16_t counter;

void main(void)
{
    // Main Loop
    while(1)
    {
        if (counter >= SCANS_PER_DECODE)
        {
            mTouch_decode();
            counter = 0;
        }
    }
}

void interrupt ISR(void)
{
    uint16_t newReading;

    if (TMR0IE && TMR0IF)
    {
        // Take a reading, store the value
        newReading = mTouch_getReading();

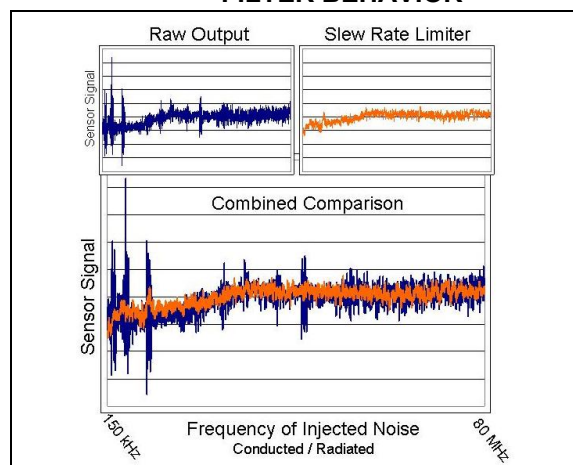
        // Initialize
        if (reading == 0)
            reading = newReading;

        // Slew Rate Limiter
        if (newReading > reading) reading++;
        else reading--;

        counter++;
    }
}
```

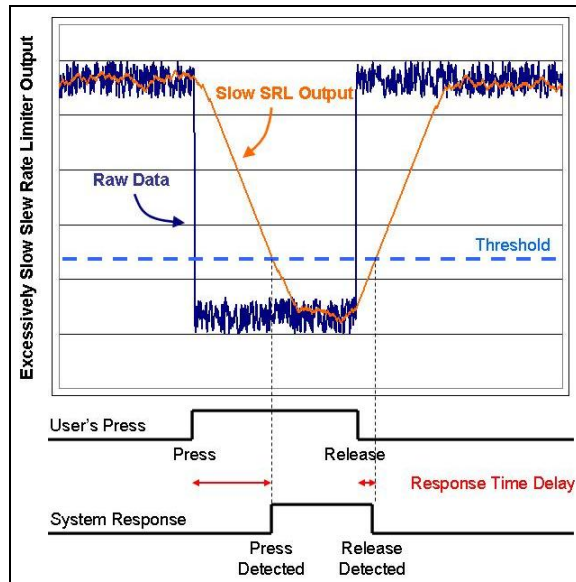
The benefits of this filter can be seen in [Figure 23](#). The impulse noise on the system has been rejected and the signal is now more easily decoded.

FIGURE 23: SLEW RATE LIMITER FILTER BEHAVIOR



Care should be taken to make sure the SRL filter is not moving too slowly. If the sampling rate is too low, the current reading value will not update fast enough and can cause problems with response times. An example of what this too-slow behavior will look like is shown in Figure 24.

FIGURE 24: EXCESSIVELY SLOW SLEW RATE LIMITER FILTER EXAMPLE



To solve this issue, either:

1. Reduce the amount of time between timer interrupts.
2. Change the increment/decrement amount to a value larger than 1. However, the larger this value is, the less the filter will be able to reject impulse noise.

FILTER: L-POINT RUNNING AVERAGES

This filtering technique is used in a vast number of applications and has been documented thoroughly. Its behavior is defined in Equation 4. The current value, $x[n]$, is averaged with the last $L-1$ values. When deciding on a value for L , keep in mind the division operation. If a power of 2 is chosen for the value of L , the division operation can be simplified to a series of bit-shifts, reducing the complexity of the filter.

EQUATION 4: L-POINT AVERAGE (FIR)

$$y[n] = \frac{1}{L} \sum_{k=0}^{L-1} x[n-k]$$

Where:

$y[n]$ = output at time 'n'

$x[n]$ = input at time 'n'

L = memory of the filter

k = counter variable

Also notice that, as implemented in Equation 4, this is an FIR filter which means that a single reading can only affect the output for a specific number of samples, L . After this period, the sample no longer has an influence on the system's behavior. For noise-injected situations, this is a very beneficial characteristic for our filters to have. However, the memory cost of this filter will make any filters with a large window difficult to implement.

To solve this limitation of the filter, its behavior can be changed from FIR to IIR, as shown in Equation 5. This creates a fixed, low-memory cost for the filter but degrades some of its features. The IIR version of the L-Point Running Average will become more distorted from the original as L gets larger.

EQUATION 5: L-POINT AVERAGE (IIR)

$$y[n] = y[n-1] + \frac{x[n] - y[n-1]}{L}$$

Where:

$y[n]$ = output at time 'n'

$x[n]$ = input at time 'n'

L = memory of the filter

EXAMPLE 3: FIR L-POINT AVERAGE FILTER

```
#define HISTORY 8 // 'L'

uint16_t reading[HISTORY];
uint8_t index;

uint16_t FIR_Average(uint16_t new)
{
    uint16_t average = 0;

    // Replace oldest reading
    reading[index] = new;

    // Sum all reading values
    for (uint8_t i = 0; i < HISTORY; i++)
    {
        average += reading[i];
    }

    // Divide by the history window size
    // NOTE: This operation is efficient
    // as long as HISTORY is a power of 2.
    average = (uint16_t)(average/HISTORY);

    // Update index for next function call
    index++;
    if (index >= HISTORY) index = 0;

    return average;
}
```

EXAMPLE 4: IIR L-POINT AVERAGE FILTER

```
#define HISTORY 8 // 'L'

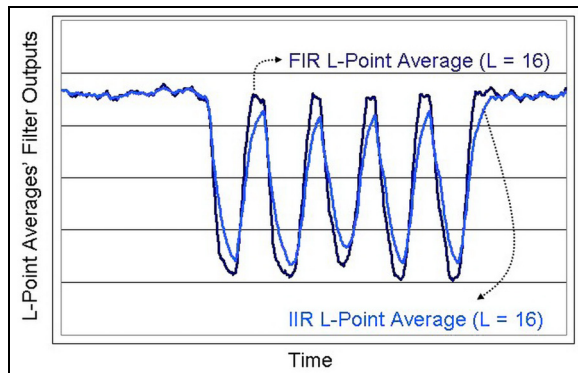
uint16_t average;

uint16_t IIR_Average(uint16_t new)
{
    // Update Average
    average -= (uint16_t)(average/HISTORY);
    average += new;

    // NOTE:
    // This filter implementation has a gain
    // of 'HISTORY'. For that reason, we
    // can divide the average by HISTORY
    // or multiply the reading by HISTORY
    // before we compare the two.
    return (uint16_t)(average/HISTORY);
}
```

Figure 25 shows the difference between the FIR L-point running average and the IIR L-point running average estimation. Notice that the estimation introduces a response time delay in the system. For this reason, it is not recommended to use this filter directly on the reading signal. However, the time delay of this filter is not a problem for the baseline calculation. Since the baseline should be moving very slowly, the time delay will not affect its behavior negatively.

FIGURE 25: FIR VS. IIR AVERAGES



FILTER: LOW PASS BUTTERWORTH

This filter implementation is an alternative to the L-point running average. While both are low pass filters, the digital low pass filter in this section is based on the digital implementation of a Butterworth filter, and can be seen defined in Equation 6. Notice that the only complex operation in this filter is the multiplication between K and the previous output of the filter. If K is chosen wisely, this filter can be easily implemented with the use of bit-shifts.

EQUATION 6: DIGITAL BUTTERWORTH LOW PASS FILTER

$$y[n] = x[n] + x[n-1] + Ay[n-1]$$

Where:

$y[n]$ = the output at time 'n'

$x[n]$ = the input at time 'n'

A = the filter's coefficient ($0 \leq A < 1$)

Smart values for A include: 0.8125, 0.8750, 0.9375, and 0.9688 to name a few. These factors can be multiplied easily using only bit-shift operations on the $y[n-1]$ value. An example of this can be seen below:

As the value of 'A' gets closer to 1, the cutoff frequency of the Butterworth low pass filter approaches zero. This increases the effectiveness of the filter, but will slightly increase the filter's settling time. Also, as 'A' gets closer to 1, the integer value of $y[n]$ will increase quickly. This puts an upper limit on the value of 'A' if you want to continue representing each sensor's signal with the typical 16-bit integer value.

The benefit of this type of filter when compared with the L-point running average can be seen in Figure 26. The increase in the effective SNR of the sensor is much higher when implementing the Butterworth than the running average; however, the running average estimate is very inexpensive to implement and performs the job well as a filter for the sensor's baseline.

**EXAMPLE 5: DIGITAL BUTTERWORTH
 LOW PASS FILTER**

```
#define       FILTER_GAIN       3

typedef struct
{
    uint16_t y;
    uint16_t x;
} filter_t;

filter_t     filter;

uint16_t LP_Butterworth(uint16_t reading)
{
    // Pointer to the correct filter
    // variables for our sensor...
    filter_t* h = &filter;

    // Temporary variables
    uint16_t x1, y1, ay1;
    uint16_t temp0, temp1;

    // Populate temporary variables
    x1 = (h -> x);
    y1 = (h -> y);

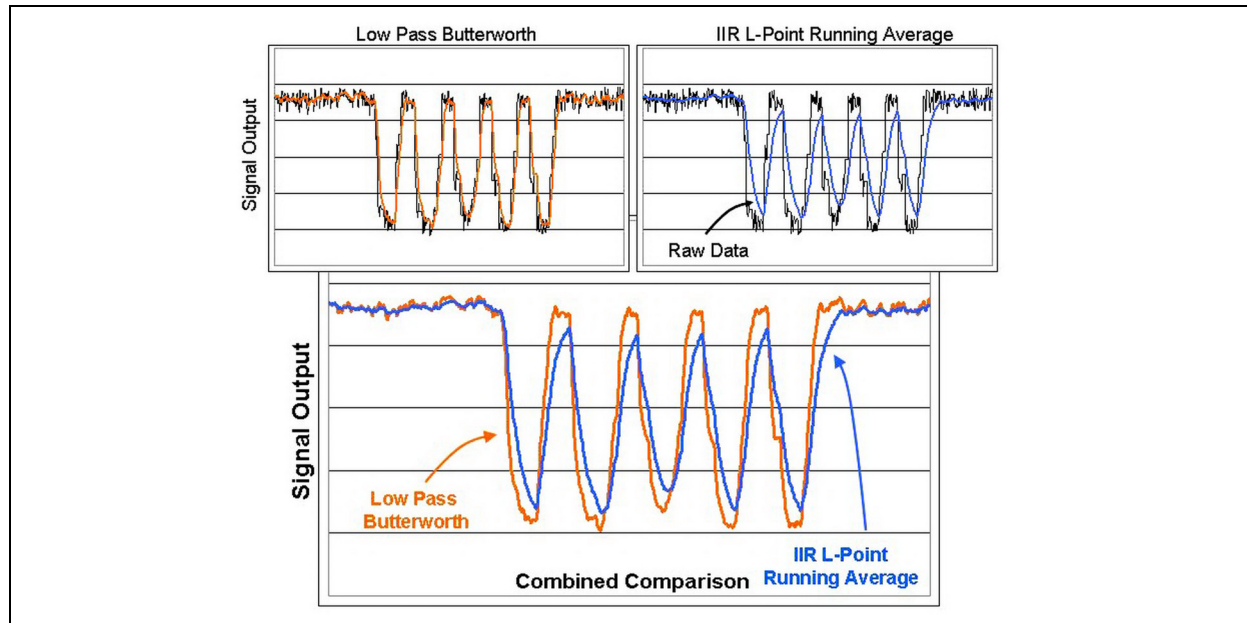
    // Calculate: ( a * y[1] )
    // Where, a = 0.8125.
    temp0 = y1 >> 2;
    temp1 = y1 >> 4;
    ay1 = y1 - temp0 + temp1;

    // 1st Order Filter Equation:
    // y1 = x[0] + x[1] + (a * y[1])
    y1 = reading + x1 + ay1;

    // Store values for next time
    (h -> y) = y1;
    (h -> x) = reading;

    // Return the filter's new result
    return y1 >> FILTER_GAIN;
}
```


FIGURE 26: LOW PASS FILTER COMPARISONS



SETTING SENSOR THRESHOLDS

After the sensor signal has been read and filtered, the decoding process should begin. During this step, the sensor's signal will be compared against a pre-determined threshold to decide if there is a press.

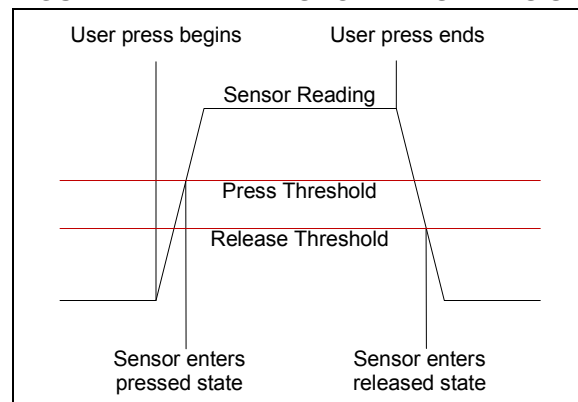
When establishing a threshold, there are two main options: setting a fixed threshold or calculating the threshold at run-time. Setting a fixed threshold is the easiest and least time/memory intensive, but it may not work in every situation. For mass production systems, it is possible there will be slight differences in the sensor's default values across boards. If the application is using a fixed threshold, there is a possibility that the threshold may not work for all of them. Calculating the threshold at run-time is the other option. When this is implemented, the system will take several readings on power-up and will determine where the threshold should be set based on the system's samples. For the majority of applications, a calculated threshold will not significantly affect the system's behavior and is preferred for its flexibility.

In some cases, two thresholds may be used in the same way as hysteresis – one used to enter the pressed state and one used to enter the released state. A diagram showing this behavior is seen in [Figure 27](#).

Choosing where to place the thresholds for a system can be one of the most important and difficult tasks of getting a robust solution working. Assuming the sensor's signal moves down when pressed, if the threshold is too high, false triggers can become a risk. If the threshold is too low, the system may be unable to detect a press in all situations.

It is important to remember that your system will be used by a wide variety of people. People with big hands will cause more of a shift than those with small hands. Make sure you are setting the thresholds and testing them with a cross-section of individuals so that everyone will be able to activate the sensor.

FIGURE 27: THRESHOLD HYSTERESIS



ENVELOPE DETECTOR

The Envelope Detector is a decoding technique that uses an extra variable to track the average deviation from the sensor's baseline (or "average"). This can be particularly effective in systems that experience a large amount of injected noise. [Figure 27](#) illustrates an example of a system that would benefit from this decoding technique.

When a high level of injected noise is present on capacitive touch sensors, press detection can be difficult if the decoding algorithm is only looking for a shift in sample values. Some frequencies of noise may cause the press behavior shown in Figure 28. When this occurs, an envelope can be created to track the noise level of the system and this can be used with a threshold to make press decisions. An example implementation of this decoding technique has been provided in Example 6.

Keep in mind while implementing this technique that the delta value is the absolute value of the difference between the current sensor reading and the average. This means that shifts in both the positive and negative direction will cause the envelope to increase. If this behavior is not desirable for your system, the absolute value section of the code example can be removed. For more information about the envelope detector decoding algorithm, see application note AN1317, “mTouch™ Conducted Noise Immunity Techniques for the CTMU”.

EXAMPLE 6: ENVELOPE DETECTOR

```
// Speeds from Fastest-Slowest:
//      2, 4, 8, 16, 32
#define      SPEED      4

uint16_t envelope;
uint16_t baseline;

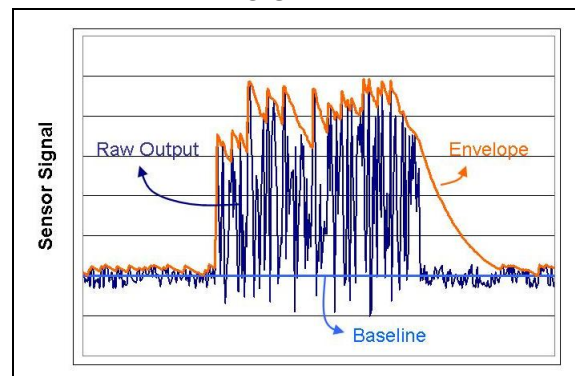
uint16_t UpdateEnvelope(uint16_t reading)
{
    int16_t delta = baseline - reading;

    // Absolute Value
    if (delta < 0)
        delta = -delta;

    // Update envelope
    envelope -= (uint16_t)(envelope/SPEED);
    envelope += (uint16_t)(delta);

    return envelope;
}
```

FIGURE 28: ENVELOPE DETECTOR UNDER HIGH INJECTED NOISE



COMMON CHALLENGES

There are several behaviors of capacitive touch systems that commonly appear. In this section, the main issues seen in these applications will be discussed and the different possible solutions described that can be used.

The challenges and solutions covered in this section are:

- Crosstalk
- Impulse Noise
- Unresponsive Buttons
- Flickering Buttons
- Reversed Operation

Common Challenges: Crosstalk

Crosstalk is the undesired shift of a sensor when an adjacent sensor is pressed. It is mainly a side effect of placing sensors too close together, but can be also significantly affected based on the thickness and relative permittivity of the covering material.

Best Option: Increase the amount of separation between sensors.

If possible, reducing the amount of crosstalk using hardware design techniques is preferred to the software adjustments that can be made. Crosstalk will also become more of a problem in systems with thick covers, so reducing the overlay thickness may also be a required step.

For more information on the causes of crosstalk and the available hardware changes to reduce this effect, see the [Section “Pad-to-Pad Separation Distance”](#) of this application note.

Option 2: Adjust the Thresholds

If the above hardware solutions are not adequate or cannot be used by your specific application, adjusting the software may be your only alternative. Changing the threshold by increasing the amount of shift required to detect a press might be the best option if the system has more sensitivity than needed.

Eliminate the ability of crosstalk to activate a sensor by requiring that the sensor's reading shifts more than the maximum shift caused by crosstalk in the system. Keep in mind when doing this, however, that noise on the system may increase the maximum crosstalk shift which could cause false triggers in the future.

Option 3: Implement the "Most Pressed" Algorithm

Alternatively, the system's decoding algorithm can be changed to compare each sensor's shift with the other sensors' in order to determine which one is the "most pressed."

Implementing this algorithm will limit your system's capabilities. First, the system will only be able to support one press at a time since only the highest shift sensor is always picked. To minimize the effect of this limitation, only compare nearby sensors to each other. The second limitation this algorithm places on a design is on response time. The extra execution time required by the processor and the requirement that all sensors must be scanned before the decoding process can begin will both slow the system down. Depending on the application, this may or may not be an acceptable design trade-off.

Common Challenges: Impulse Noise

Impulse noise appears as individual or small groups of readings that behave in a significantly different manner than the readings before and after them due to a noise source and not a finger's press. These spikes can quickly destabilize a system if not handled correctly. Additionally, if noise spikes are a common problem for the system, they can also lower the effective SNR.

Software filters are usually the easiest method for removing this problem. The key is to adjust the filter to reduce the affect of the spikes, but not affect the response time of the system. For example, if the decision is made to take the average and just slow it down to minimize the impact a single reading can have, it will reduce the noise spikes in the system. However, it will also slow down the response time since reliance is now on a much slower average.

Best Solution: Implement the Slew Rate Limiter filter.

Infinite Impulse Response (IIR) filters are not going to be as effective as Finite Impulse Response (FIR) filters will be in this case. Since IIR filters allow one reading to affect the signal for a (theoretically) infinite period of time into the future, the effect of a noise spike could cause the filter to become unstable. The best option in this case is to use the Slew Rate Limiter (also referred

to as 'Decimation') filter described in the software section of this application note. It will eliminate all impulse noise from the system with a minimal impact on the overall behavior so long as the sample rate is increased to adjust for the decrease in response time.

Common Challenges: Unresponsive Buttons

Unresponsive buttons are sensors that will not change state when a finger is placed on them. The two main software features that will affect your sensors to possibly make them unresponsive are thresholds and debounce values. Sensor sensitivity can change as noise is injected on a system. This change can reduce the sensitivity to the point that the threshold is no longer crossed. Increased noise on the system will increase the probability of a single sensor reading falling outside of the acceptable range to change state. If the maximum debounce value is high and the noise is high, it is possible the sensor will never respond.

There are several things that can be done to fix this problem in a system.

Best Solution: Adjust the Thresholds

First, the threshold can be made easier to cross by modifying the software. When noise reduces the sensitivity of the system, the easier threshold will decrease the possibility of an unresponsive button. This may not be a solution if making the threshold easier to cross will introduce false triggers into the system. The best way to make sure there is plenty of sensitivity to work with in this situation is to follow the hardware design guidelines described in this application note.

Option 2: Check Both Directions for Shifts

Another possibility is that the system is being injected with a specific amount of noise that has caused it to reverse its press behavior. When this happens, pressing on the sensor will cause a shift in the opposite direction as expected. The solution to this problem is to place the threshold on both sides of the readings. This may not be an option, however. In some systems, the direction of a shift indicates a specific event. For example, in some water-resistant systems a shift in one direction is a press while a shift in another direction indicates the presence of water. Implementing a shift-check in both directions would eliminate the water resistance of the application.

Option 3: Use Hardware to Increase Sensitivity

Finally, if adjusting the thresholds does not solve the problem, hardware changes may need to be made to increase your system's base sensitivity. First, make sure the application is following all of the design guidelines. The next step is to begin increasing sensor area, decreasing crosstalk through sensor separation, decreasing the cover thickness, or changing its material. All of these suggestions can be traced back to their origins in [Equation 2](#), which defines the relationship between capacitance and hardware design.

Common Challenges: Flickering Buttons

The term "flickering button" refers to a sensor that will rapidly toggle its state while a finger is present. This is distinctly separate from a false trigger, which is a sensor changing state while there is no finger. Flickering buttons are caused in the same way as unresponsive sensors. As noise is injected on a sensor, the sensitivity of the sensor can change. If the sensitivity is reduced to the point that it approaches the threshold, noise can sometimes cause the readings to jump above and below it. This will result in the sensor rapidly changing states.

Best Solution: Implement Threshold Hysteresis

The best and most effective solution to this problem is to adjust your algorithm so that it implements a large hysteresis on the thresholds. Separate press and release thresholds will be able to tolerate a much larger amount of system noise. If sensitivity levels are high, the thresholds will be able to be placed further apart, increasing the noise resistance of the system. Similarly, as the thresholds are placed closer together, the system begins to return to its single-threshold behavior and can once again experience flickering buttons. For these reasons, a combination of hysteresis and debouncing should be the default solution to this problem in applications.

Option 2: Increase the Debounce Requirement

The easiest way to reduce this problem is to increase the maximum debounce value, but this method can have mixed results. Increasing the debounce count will decrease the response time of the system and will only decrease the probability of flickering. The smaller probability of flickering will translate to a slower flicker, but most likely not the complete elimination of it.

Option 3: Adjust the Thresholds

A third option is to make the threshold easier to cross or to increase the sensitivity of the system through hardware changes. However, if there is a large amount of noise on the system, this still may not be enough by itself to solve the problem.

Common Challenges: Reversed Operation

Reversed operation can occur in systems with a high amount of injected noise. The noise can cause the system to shift in the opposite direction as the no-noise behavior. If the system was shifting down on each press, it is now shifting up. This can be a problem in systems that only look for a shift in one direction. Figure 24 shows how a system looking for a "down" shift can enter a reversed operation state if an "up" shift occurs. There are two main ways of solving this problem. The baseline's speed can be decreased and thresholds can be placed on both sides of the baseline.

Best Solution: Check Both Directions for Shifts

The more robust solution to this problem is to create a threshold on either side of the average so that a shift in either direction will result in a detected press. Not all systems may work well with this solution, however. For example, in some systems designed to work in wet conditions, a shift in one direction means a finger is present while a shift in the other direction corresponds to the effect of water. In a situation such as this, it may be wiser to ignore any shifts in the wrong direction. Ultimately, the decision should be based on the specific application's requirements.

Option 2: Adjust the Filters' Time Constants

Slowing down the baseline's speed would require that a longer opposite-direction-from-normal shift occur before the system enters the reversed operation state. This would cause unresponsive buttons when specific types of noise are injected onto the application, but may be the best option for a given system. Just slowing down the average will not eliminate the problem, however. If a user were to press on a sensor for a very long period of time, the slow average will eventually follow the finger and, when it is finally removed, reversed operation could still occur. This is a performance trade-off that must be considered when using any baseline in a system. Determining the difference between the environment naturally shifting the readings and a finger artificially shifting the readings can sometimes be difficult.

Option 3: Increase the VDD of the system

While this will not eliminate the problem, raising VDD will fight against the reversing behavior by increasing the amount of noise that must be injected on the system before it flips. This option is placed last because increasing VDD results in a higher current consumption for the system.

CONCLUSION

Good hardware design choices are the foundation to a robust capacitive touch design. By following the provided design guidelines, your custom application will have a high base SNR which will decrease your required software overhead, speed up your system's overall response time, and allow your system to perform well even in noisy conditions.

Although there are many opportunities to adjust a design to meet a specific application's needs, the best design choices for a robust system are summarized in Table 1.

Acquisition techniques such as jittering the sample rate and implementing a slew rate limiter will reduce the amount of noise that is introduced through the readings. Digital filters such as the L-Point Average and the low pass Butterworth allow the system to track environmental changes and further increase the SNR of the signal. Finally, specific algorithm implementations such as threshold hysteresis, debouncing, and the "most pressed" algorithm will allow the system to react to even the most rare of noise disturbances.

From start to finish, capacitive touch systems should be developed with the key focus of increasing the SNR of the sensors' signal. Following the given hardware suggestions and implementing the provided software algorithms will make the system both cost effective and robust.

For information on the basics of mTouch sensing and other more advanced topics, visit the Microchip web site at <http://www.microchip.com/mTouch>.

TABLE 1: IDEAL HARDWARE DESIGN RECOMMENDATIONS

Button Pads	
Shape	No specific requirement.
Size	15 x 15 mm
Pad-to-Pad Distance	10 mm
Overlay	
Thickness	< 3 mm
Material	$2.0 \leq \epsilon_r \leq 8.0$
Adhesive	Thin, high ϵ_r , no bubbles
Layout	
Sensor traces	Thin, short
Series resistance	CVD 4.7 - 10 k Ω
	CTMU 1 - 2.5 k Ω
LED/Comm. traces	Avoid sensor traces
Power Supply	
Shares GND w/User	Improve sensitivity by 2x
Best VDD for Noise	Higher is better
Bypass Capacitors	0.1 μ F on all VDD pins

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, dsPIC, FlashFlex, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, PIC³² logo, rPIC, SST, SST Logo, SuperFlash and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MTP, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.


Analog-for-the-Digital Age, Application Maestro, BodyCom, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Omniscent Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICtail, REAL ICE, rLAB, Select Mode, SQI, Serial Quad I/O, Total Endurance, TSHARC, UniWinDriver, WiperLock, ZENA and Z-Scale are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

GestIC and ULPP are registered trademarks of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2010-2013, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

ISBN: 9781620772126

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
== ISO/TS 16949 ==

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
<http://www.microchip.com/support>
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hangzhou
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880-3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7828
Fax: 886-7-330-9305

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Druenen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820