BY SACHIN GUPTA, PRAKHAR GOYAL, AND KURIAN POLACHAN
CYPRESS SEMICONDUCTOR CORP

Designing reliable capacitive touch interfaces

UNDERSTAND HOW TO ENHANCE THE HARDWARE AND SOFTWARE RELIABILITY OF THIS INCREASINGLY POPULAR USER INTERFACE FOR APPLIANCES AND INSTRUMENTS.

Capacitive sensing offers an intuitive and robust interface that increases product reliability by eliminating mechanical parts in many appliances, or white goods, and instruments. Because of their experience with personal-electronics devices, many consumers are used to touch interfaces employing capacitive sensing, and they have come to expect these interfaces to be reliable and to operate accurately.

Environmental noise and other factors can affect capacitive technology, however, causing systems to be unresponsive to finger touches or to trigger false touches. Developers must fine-tune these sensors or risk a severe reduction of accuracy and reliability. By understanding how capacitive sensors work and how you can design them to be self-tuning to compensate for noise, you can build robust systems that make appliances more reliable, more cost-effective, and easier to use.

CAPACITIVE SENSING

To understand the challenges behind designing a robust user interface, it helps to first take a brief look at the technology behind a capacitive-measurement system (**Figure 1**). To sense the presence of a finger, a capacitive-sensing system must first know the sensor capacitance in the absence of a finger, or parasitic capacitance (**Figure 2a**). When a finger approaches or touches the sensor, the sensor's capacitance changes, resulting in another capacitance, the finger capacitance, in parallel to the parasitic capacitance (**Figure 2b**). The following **equation** yields the total sensor capacitance in the presence of a finger: $C_X=C_P \pm C_F$, where C_X is the total sensor capacitance, C_P is the parasitic capacitance, and C_F is the finger capacitance.

To analyze the sensor capacitance using a microcontroller, you must convert the sensor capacitance into a digital value. Several methods are available for measuring sensor capacitance. One type uses a switched-capacitor block that emulates the sensor capacitance using an equivalent resistance, a programmable current source, an external capacitor, and a precision analog comparator (**Figure 3**). The programmable current source continuously charges the external capacitor until its voltage crosses the reference voltage and the comparator's output is high. The programmable current source then disconnects, and the external capacitor discharges through the resistance until the capacitor's voltage drops below the reference voltage. The comparator's output is now low until the capacitor again charges to the reference voltage. The sen-

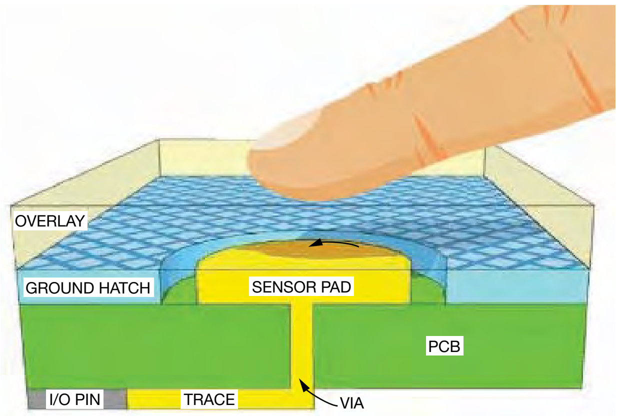

Figure 1 To understand the challenges behind designing a robust user interface, it helps to first take a brief look at the technology behind a capacitive-measurement system.

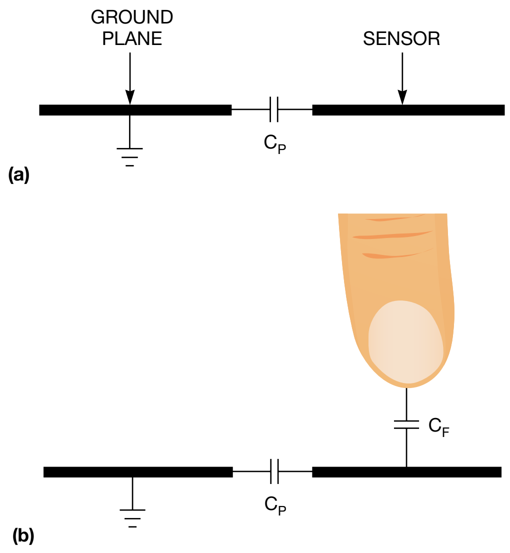

Figure 2 To sense the presence of a finger, a capacitive-sensing system must first know the sensor capacitance in the absence of a finger, or parasitic capacitance (a). When a finger approaches or touches the sensor, the sensor capacitance changes, resulting in another capacitance, the finger capacitance, in parallel to the parasitic capacitance (b).

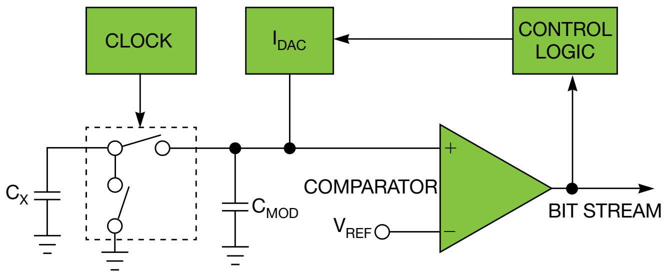

Figure 3 One method for measuring sensor capacitance uses a switched-capacitor block that emulates the sensor capacitance using an equivalent resistance, R_{EQ}; a programmable current source, I_{DAC}; an external capacitor, C_{MOD}; and a precision analog comparator.

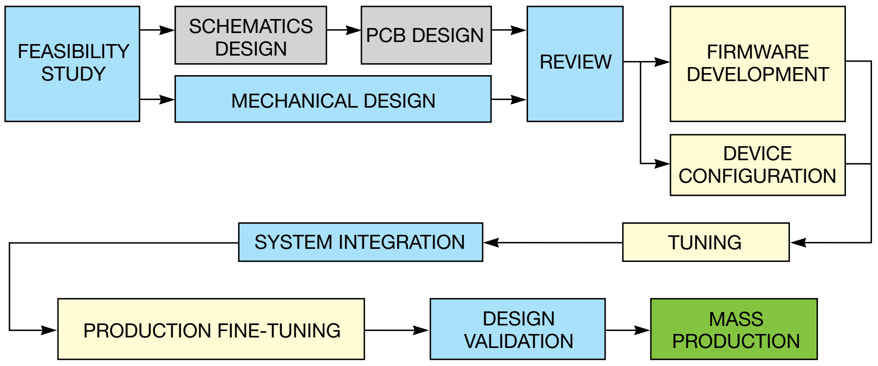

Figure 4 The design flow for a capacitive-sensor touch interface must account for real-world conditions, in which variations occur in components, environmental operating conditions, and noise.

sor's capacitance is greater in the presence of a finger, and the emulated resistance is lower, according to the following **equation**: $R_{EQ}=1/F_SC_X$, where R_{EQ} is the equivalent resistance and F_S is the switching frequency of the switched-capacitor block.

Thus, when a finger is present, the external capacitor discharges faster, and the comparator's output stays high for a shorter time, meaning that a higher capacitance value corresponds to a shorter high time for the comparator. You can feed the resulting bit stream to a counter for a fixed amount of time. This counter value, or raw count, provides an indication of the magnitude of the sensor's capacitance. The fixed amount of time for which the counter counts also determines the number of raw counts, or resolution. When you increase the resolution, the counter counts for a longer period, thus increasing the raw count. In other words, resolution is also the highest possible number of raw counts.

TUNING

Figure 4 shows the design flow for a capacitive-sensor touch interface. However, capacitive sensors must operate in the real world, in which variations in components, environmental operating conditions, and noise can affect sensor performance and reliability. Tuning is a critical process for ensuring that a sensor functions correctly and consistently. You achieve this goal by identifying and determining optimum values for a set of sensor parameters to maintain a sufficient SNR (signal-to-noise ratio) and finger threshold. In general, a 5-to-1 SNR is the minimum requirement for a robust sensor design (**Figure 5**). To avoid false triggering due to changes in capacitance resulting from atmospheric changes, a finger threshold of 65 to 80% of the signal strength ensures reliable finger detection.

Although sensor-controller manufacturers provide guidelines to aid engineers in tuning, achieving the ideal tuning parameters for the system involves iteration. For a sensor controller with a capacitive-sensing algorithm, the tuning procedure follows a particular set of steps (**Figure 6**).

Developers can implement tuning parameters either by writing code specific to the operation of the sensors in firmware, through external components, or by configuring the controller. With a firmware approach, developers have flexibility; however, whenever tuning parameters require changes, developers must also modify and update the firmware.

Alternatively, designers can simplify system-firmware development by using a fixed-function, nonprogrammable capacitive-sensor controller. In this case, designers must implement tuning parameters either by using external components on the board or by sending configuration data over a communication interface, such as an I^2C (inter-integrated circuit). With this approach, whenever tuning parameters require changing, the developer must either rework the user-interface board or update the configuration data. Developers must be aware that tuning can be time-consuming, especially if the PCB (printed-circuit board) or overlay requires changes between iterations.

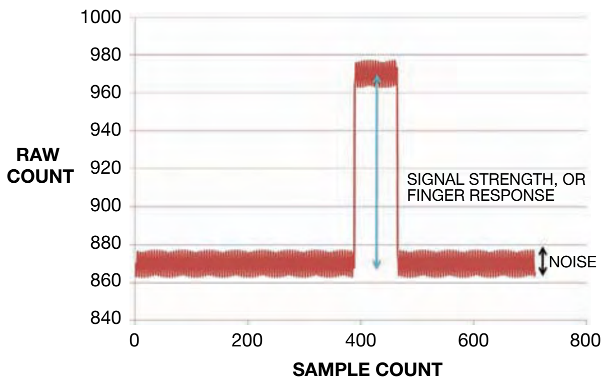

Figure 5 A 5-to-1 SNR is the minimum requirement for a robust sensor design.

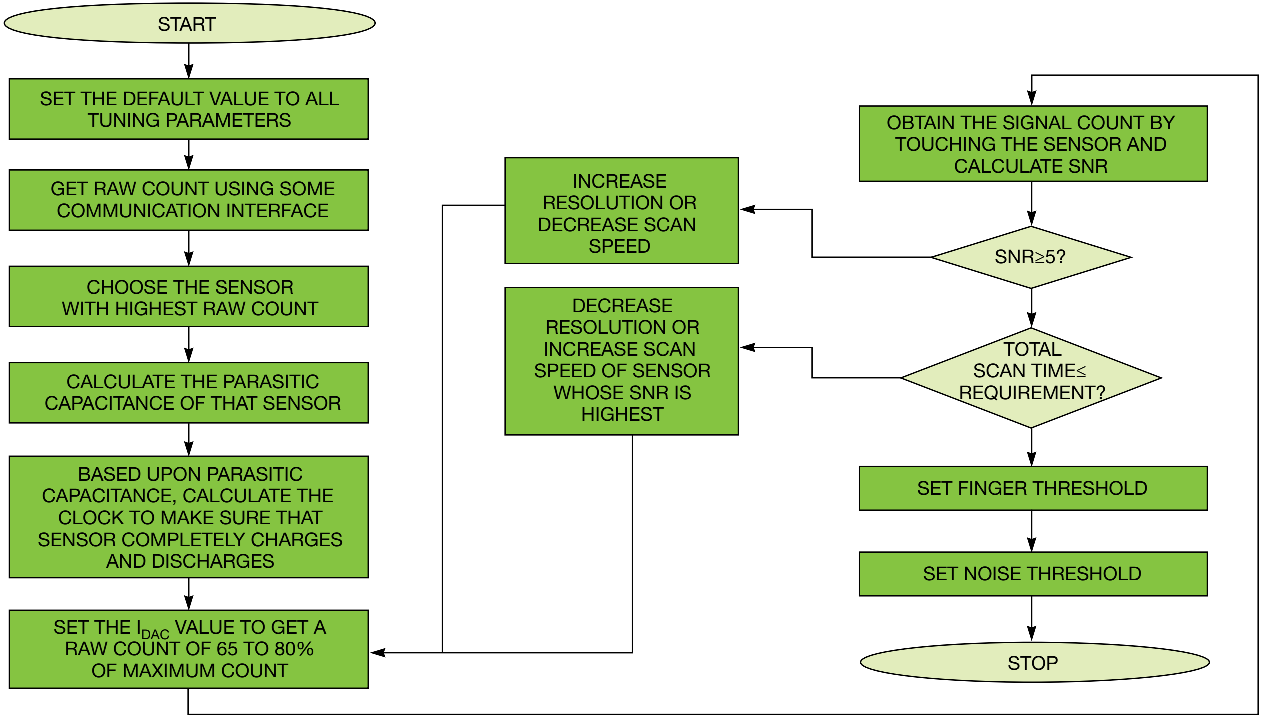
START
SET THE DEFAULT VALUE TO ALL TUNING PARAMETERS
GET RAW COUNT USING SOME COMMUNICATION INTERFACE
CHOOSE THE SENSOR WITH HIGHEST RAW COUNT
CALCULATE THE PARASITIC CAPACITANCE OF THAT SENSOR
BASED UPON PARASITIC CAPACITANCE, CALCULATE THE CLOCK TO MAKE SURE THAT SENSOR COMPLETELY CHARGES AND DISCHARGES
SET THE I_{DAC} VALUE TO GET A RAW COUNT OF 65 TO 80% OF MAXIMUM COUNT
INCREASE RESOLUTION OR DECREASE SCAN SPEED
DECREASE RESOLUTION OR INCREASE SCAN SPEED OF SENSOR WHOSE SNR IS HIGHEST
OBTAIN THE SIGNAL COUNT BY TOUCHING THE SENSOR AND CALCULATE SNR
SNR≥5?
TOTAL SCAN TIME≤ REQUIREMENT?
SET FINGER THRESHOLD
SET NOISE THRESHOLD
STOP

Figure 6 Although sensor-controller manufacturers provide guidelines to aid engineers in tuning, achieving the ideal tuning parameters for the system involves iteration. For a sensor controller with a capacitive-sensing algorithm, the tuning procedure follows these steps.

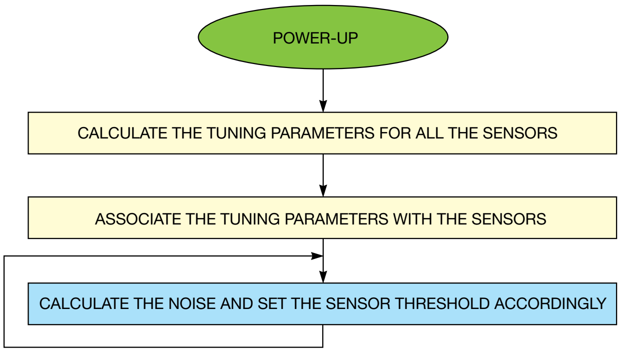
POWER-UP
CALCULATE THE TUNING PARAMETERS FOR ALL THE SENSORS
ASSOCIATE THE TUNING PARAMETERS WITH THE SENSORS
CALCULATE THE NOISE AND SET THE SENSOR THRESHOLD ACCORDINGLY

Figure 7 To cost-effectively handle design constraints and market needs, it is best to implement tuning in the appliance itself. This ideal self-tuning system performs this task.

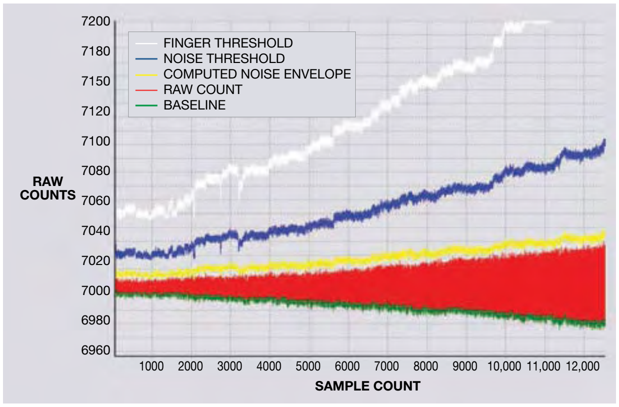

Figure 8 Adjust the threshold value for finger detection depending on detected noise in the sensor's raw counts. A self-tuning system can adjust the finger threshold depending on system noise.

PRODUCTION TUNING

Capacitive-sensor performance depends heavily on the physical properties and characteristics of the sensor board and environmental and operating conditions. For example, sensor capacitance can change due to variations in the PCB-manufacturing process, in overlay material or thickness, or in the PCB vendor. All of these variations can affect sensor performance, and the challenges do not stop there. Parasitic capacitance also varies with environmental conditions, such as noise, temperature, and humidity. Thus, a board you tune in the Swiss Alps may not work in the hot and humid climate of Hong Kong, resulting in more time and labor to retune the board. To minimize yield losses due to process variation or vendor change, tuning must take into account expected differences based on statistical analysis.

You may need to redo the board layout for various reasons, including changing a button's size, moving traces to incorporate minor changes in the schematic, and resizing to address EMC (electromagnetic-compliance) and EMI (electromagnetic-interference) issues. All of these modifications require that you retune the sensors. Moreover, tuning requires a communication protocol and a host-side application to observe and analyze the raw sensor data. Tuning also requires extra I/Os because it must take place after the final board is complete, creating potential issues for systems with pin constraints.

Tuning is a difficult job, requiring significant expertise with the chips and an understanding of capacitive-sensing effects at low signal levels. Couple these issues with the time-to-market constraints of the appliance market, and tuning can impose significant delays and increase system cost. To cost-effectively handle design constraints and market needs, it is best to implement tuning in the appliance itself. An ideal self-tuning system performs this task (**Figure 7**).

In systems with self-tuning capacitive sensing, numerous algorithms achieve a workable touch-sense system. At a basic level, appliance-implemented self-tuning is the same as manual tuning. In one-time compensation, some tasks occur once at power-up; in dynamic compensation, the appliance must continuously perform some tasks.

Self-tuning capacitive-sensing systems must calculate the best parameter settings for the sensors depending on the appliance and expected operating environment. The capacitive-sensing systems in this article use switched-capacitor theory. In contrast, a physical sensor capacitor forms a resistor by charging and discharging the sensor capacitor in consecutive cycles. Emulated resistance is proportional to the sensor capacitor's value, and you measure it using a current source with an analog-to-digital-conversion stage to compute the value of the sensor's capacitance.

Proper emulation of resistance requires the sensor capacitor to fully charge and discharge at a frequency that provides enough time for this operation. Therefore, you should adjust the switching frequency in accordance with the absolute sensor capacitance and reduce the frequency if the sensor's capacitance is higher.

Because the system converts the capacitance of the sensor to counts, the smallest change in capacitance that you can measure depends on the system's resolution. You can calculate the required resolution using the parasitic capacitance and required sensitivity.

Scan time is one of the most important tuning parameters from a system-specification point of view. However, increasing the resolution of scanning adds increased noise to the system. To compensate for this increase in noise, the scan time of sensors must stretch to integrate the noise and reduce its effect on capacitance measurements. A self-tuning algorithm must not exceed the scan time beyond system requirements. The best way to deal with scan time is to lay out the board to keep parasitic capacitance as low as possible.

Upon selecting the scanning resolution, you must automatically adjust a baseline measurement of the sensors—that is, the raw count when a finger is not present—to nearly 80% of the maximum count. This approach ensures that variations in neither environmental conditions nor silicon parameters adversely affect the sensor's measurement precision or its ability to accurately detect a finger touch.

Noise, by its nature, is a random function of time. It is not the same on power-up, a moment after power-up, or an hour after power-up. As a result, you should adjust the threshold value for finger detection depending on detected noise in the sensor's raw counts. A self-tuning system can adjust the finger threshold depending on system noise (**Figure 8**).

SNR and scan time are the main factors determining the robustness, reliability, and efficiency of appliance-implemented self-tuning. Ensure that the SNR from self-tuned sensors is higher than the minimum requirement of 5-to-1 across the parasitic-capacitance range to preserve robustness and reliability. Scan time affects the power efficiency of the self-tuning algorithm; the more time a sensor takes to scan a sensor, the more power it consumes. Even though a longer scan may fit application requirements, the self-tuning algorithm should optimize power consumption by cutting the scan time as much as possible without compromising the SNR.

Designers often overlook one of the most important aspects—layout—during the initial design stage of a design; board layout affects the whole system's performance. Parasitic capacitance affects the amount of effort necessary for tuning, product yields, scan time, and other system characteristics. You should follow the guidelines from controller manufacturers when designing the layout to minimize the parasitic capacitance of the sensor. These guidelines can be used to improve the performance of the system through self-tuning, which helps you meet changing market needs. Cypress, for example, provides the SmartSense self-tuning-based capacitive sensor, which automatically optimizes the scan speed to be as high as possible, minimizes power consumption, and maintains an SNR of greater than 5-to-1 to avoid any false triggering.

Self-tuning controllers eliminate the overhead of repeatedly tuning capacitive-sensing-based appliances as specs and operating conditions change. In some cases, a sensor's parasitic capacitance may be high, requiring external components and some manual tuning to maintain the capacitance within a typical range.

Manual tuning can impose significant design challenges for developers implementing capacitive sensing in appliances and other systems. Tuning is sometimes necessary for different lots due to process variations; following board redesigns to accommodate changes in overlay thickness, button size, or other requirements; or due to noise and interference issues. Tuning improves performance and reliability, but manual tuning can add cost and delay the product's release. Self-tuning controllers eliminate these costs and delays, allowing developers to quickly implement reliable systems without becoming capacitive-signaling experts.EDN

ACKNOWLEDGMENT

This article originally appeared in two parts on EDN's *sister site, Medical Designline (http://bit.ly/rhsD8E and http://bit.ly/npv5hq).*

AUTHORS' BIOGRAPHIES

Sachin Gupta is a senior applications engineer on the global-applications team at Cypress Semiconductor.

Prakhar Goyal is an application engineer at Cypress Semiconductor.

Kurian Polachan is a senior applications engineer in Cypress Semiconductor's consumer and computation division, where he focuses on CapSense applications.

Ultra Wide 9~72VDC Input
Switching Regulators