How to Design Capacitive Touch & Proximity Sensing Technology into Your Application

by Mike Salas, Director of Marketing MCUs
and Andres Marcos, Firmware Engineer, MCU Systems
Silicon Laboratories Inc., Austin, TX

The phrase "Human Interface" has suddenly become part of the lexicon of the technology industry. At its most basic level, this phrase is used to describe not only the aesthetics of a device, but, more importantly, the process by which a device responds to human interaction. While not a new concept, the availability of enabling technology that can revolutionize the way we interact with consumer electronics products and an urgency to incorporate this technology demonstrates the increased importance consumers are placing on interface design.

What defines good human interface design, and how can system designers implement a smarter, friendlier and more intuitive solution? To begin answering these questions, it is helpful to view a human interface simply as a set of functional interactions with end users and their surroundings. These interactions can be subdivided into two logical groupings: inputs and outputs.

Input events are those in which a user causes, either directly, indirectly or even inadvertently, a specific action to be performed. Examples of input events are:

- Touch detection – single finger touch, multi-finger touch, finger slides, taps, etc.
- External stimulus detection – proximity, motion, hand waving, voice, etc.
- Environmental detection – ambient light, temperature, etc.
- Physical detection – rotation, inclination, shock, vibration, etc.

With advances in fields such as touch sensors, proximity sensors, ambient light sensors and accelerometers, the ability and sophistication of devices to accept inputs has dramatically changed the entire human interface landscape.

However, it is equally important to tie input events to a tangible output event because the output event informs the user of an action that has taken place as a result of the input provided. (Sometimes input conditions result in a non-event.) Examples of output events include:

- Switching items on or off – screens, speakers, lights, safety features, etc.
- Adjusting controls – volume, backlight, brightness, stabilization, etc.
- Providing tactile feedback – auditory ("hear"), visual ("see"), haptic ("feel"), etc.

Human Interface Process Flow

The types of input events and output responses desired will vary greatly as they depend on the type of device being built. Figure 1 shows a visual representation of how a simplified human interface process flow can be envisioned.

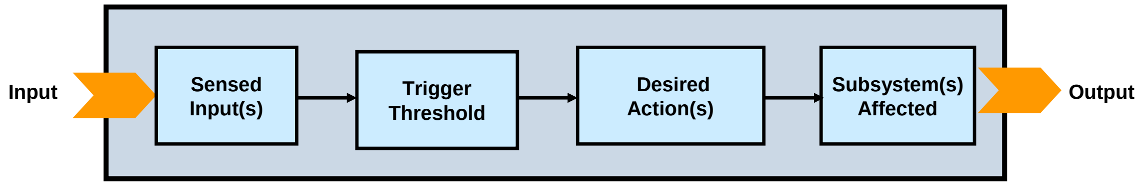


Figure 1 – Simplified Human Interface Process Flow

As shown in Figure 1, each of the input events is tied to a specific threshold level that must be met to trigger one or more output events. Similarly, each output event is tied to one or more subsystems that will be affected as a result of the input trigger. For example, a handset device may be kept in a sleep-mode for power-saving reasons. However, upon the detection of a touch that exceeds a certain pre-defined pressure threshold, the handset will turn on the screen, provide an auditory confirmation over a speaker and turn off the screen locking feature. In this example, a single input event has been directly tied to three different output events affecting two different subsystems (screen and speaker).

The good news for designers is that technological innovation has dramatically improved the ability of a device to offer a wide variety of creative input and output choices that can add a significant amount of appeal to an end product. However, this rapidly improving capability comes at a price: design complexity.

Challenges

The sheer numbers of input and output event possibilities are challenging even for the most experienced designer, and it is becoming exceedingly difficult to predict what users may find appealing not just today, but in the future as well. From an implementation perspective, what is really needed is the ability to create an interconnected framework that enables a tight coupling of these input/output interactions while still leaving flexibility to adjust for ever-changing market requirements. A possible solution is to have a set of sensing elements connected to a microcontroller (MCU), which can be used as a flexible, software-configurable platform. Depending on the needs and complexity of the end product, the MCU could be used as part of a larger system handling and processing all inputs/outputs and communicating the results to a master or host unit for further actions. The MCU also could be the brains of the whole system or product if the design requirements are relatively simple. An example of this type of human interface subsystem is shown in Figure 2.

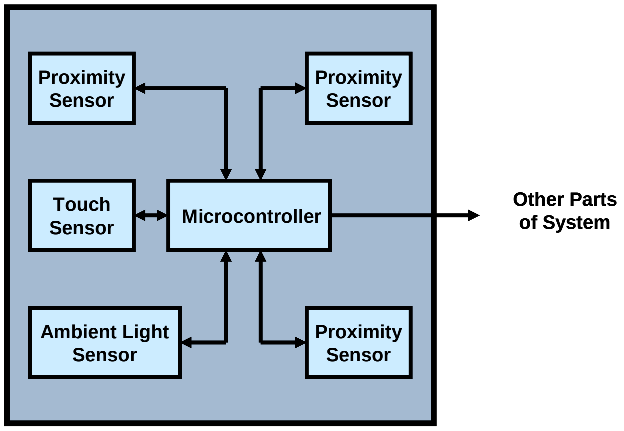


Figure 2 – Human Interface Subsystem

To better understand the role of the microcontroller as the orchestrator of a human interface system let's review each of the technologies that comprise the block diagram shown in the picture above.

Ambient Light Sensor (ALS)
At the core of an ambient light sensor is the photodiode, which is a semiconductor element that produces a current proportional to the incident light when a bias voltage is applied. The transfer function might be linear or logarithmic depending on the sensor. The produced current might be then digitized by an ADC and fed its output to the MCU or measured by the sensor and converted into a data format that a processing unit can easily interpret, such a PWM, I2C or a simple change in the state of an I/O line indicating the presence or absence of light.

Some ambient light sensing devices available in the market (such as Silicon Labs' Si1120 sensor) present not only the capability to obtain a measurement of the intensity of the incident light but also a way to distinguish a predominant light source. This is accomplished by having two different photodiodes in the same package, each having a different wavelength detection range (spectral sensitivity) and switching measurement readings between them. As shown in Figure 3, the energy contents in different light sources deliver varying characteristics for a specific subset of the spectrum; for example, the light emitted by an incandescent source has significant energy in both the visible and infrared spectrums while the light from a fluorescent source has most of its energy concentrated in the visible spectrum. Therefore, a device with both a photodiode having only visible light sensitivity (380 to 750 nm) and another photodiode with infrared spectral sensitivity (750 to 2500 nm) can be used to take a measurement from each and find out the type of the source from the ratio of these readings.

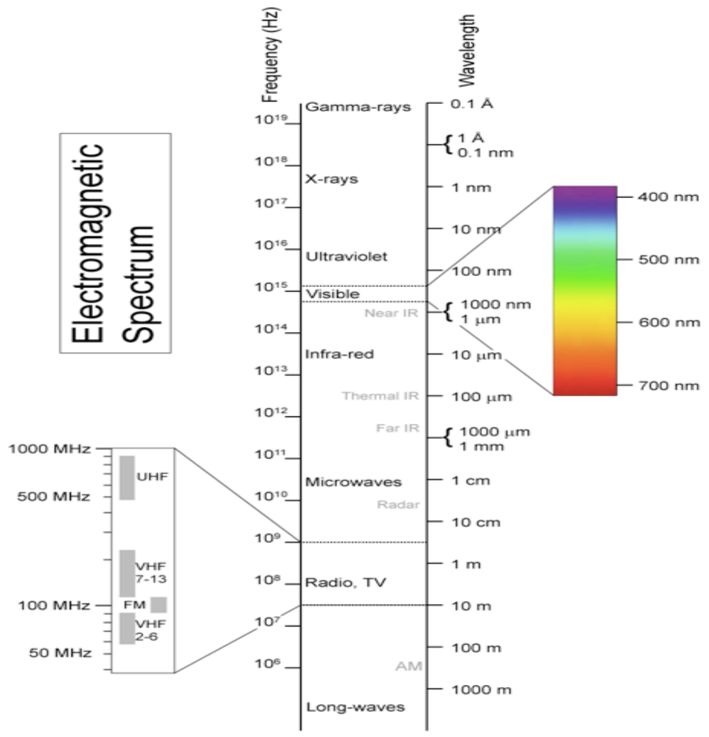


Figure 3 - Electromagnetic Spectrum

Ambient light sensors can be used to make adjustments based on the external light conditions present, thus improving the end user experience when interacting with a product. For example, a portable electronic device may use this kind of sensor to automatically dim the screen backlight as a function of the brightness of the room to save power. In an automobile, an ALS could be used to turn the headlights on or off to improve safety. A camera could adjust the shutter speed, the lens aperture and the white balance depending on the kind of light and luminescence level present.

Optical Proximity Sensor

As shown in Figure 4, an active optical proximity sensor system comprises two operating units: the transmitter and the receiver. The receiver works in the same way as an ALS but typically in the infrared range. The transmitter is an infrared source such an IR LED, which could be driven by the sensor itself or by the microcontroller depending on the type of sensor and/or the design requirements. Its basic operating principle is reflectance: Drive the transmitter to illuminate a target and measure the intensity of the reflected light on the receiver.

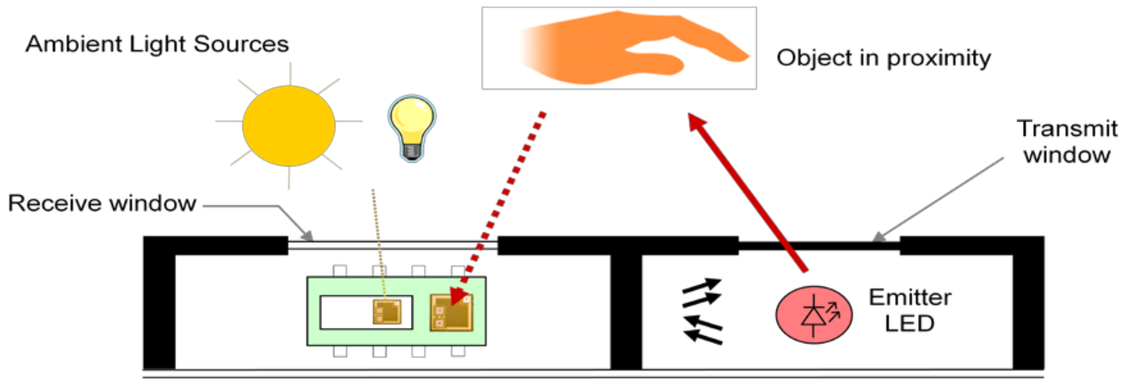


Figure 4 – Optical Sensing System

A proximity sensor is an excellent aid for products that require keeping an eye on power consumption. Examples include a cell phone that turns off the display and the backlight when held close to the ear or a monitor that goes into low-power mode when no viewers are close to it.

With an MCU available for reading the signal level (output) of the receiver, the functionality of the sensor can be expanded to determine not only whether an object is close to the system but also how close it is, opening up a whole new world of human interfaces: gesture recognition (in its simplest form, one dimensional). But the story does not end there. Simply add another transmitter to the design, use the MCU to switch between the two transmitters (turn on one transmitter at a time taking a measurement for each) and you can obtain a relative position of the object parallel to the system, in addition to the depth or distance to the object reflecting the IR energy. Repeat this process over time and the composite output becomes a set of positions that can be converted into a motion pattern. In turn, one or several motion patterns become a gesture. Examples of these kinds of gestures include left or right swipes (the speed of the swipe can be detected if sampling fast enough), pausing at a particular segment of the system and detecting the length of a pause gesture. These gestures could be useful when implementing a volume control, a page scroll control (in an electronic reader), turning pages in documents or adjusting settings on an electronic device.

Now that we have an idea of what kind of application functionality can be achieved by using an active optical proximity sensor, let's take a look at some of the design considerations needed and the challenges that can be found when developing a product that employs this technology.

When adding an active optical proximity sensor to a system, the first aspect that needs consideration is the range of operation of the end product. Range is affected by the current supplied to the IR transmitter, the emission angle of the transmitter (the narrower the angle, the longer the range), the peak output (emission) wavelength of the transmitter, the peak input (emission) wavelength of the receiver (to reach the maximum range the peak wavelength of the transmitter should match the peak wavelength of the receiver), external filters or lenses and the noise present in the environment, such the ambient light and the object's reflectivity.

Bear in mind that the distance that the energy emitted by the transmitter needs to travel is twice the desired range (from the system to the reflecting object and back). Since the decrease in light is inversely proportional to the square of the extent it travels, the energy the receiver sees is reduced by the fourth power of the range.

Another consideration is the optical isolation between the transmitter and the receiver in the system (see Figure 5). Poor isolation will add noise to the measurements, resulting in a reduced dynamic range of the signal perceived by the system, thus adversely affecting the detection range.

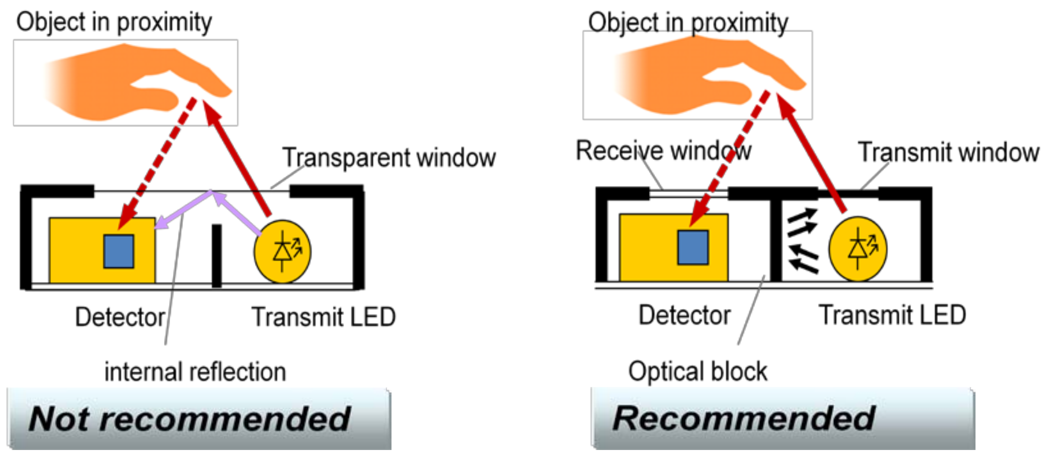


Figure 5 – Optical Isolation between Transmitter and Receiver

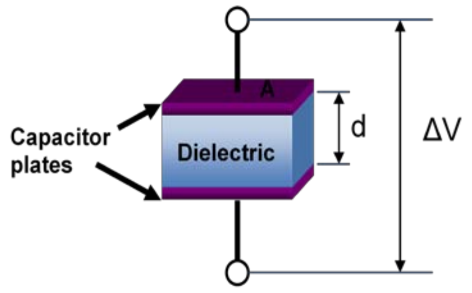
As mentioned earlier, an important source of noise for an optical proximity sensor is ambient light since it may contain considerable energy in the IR spectrum. A measurement technique that mitigates the effect of ambient light noise is to turn the transmitter off, take a measurement (to obtain the IR signal floor), turn the transmitter back on and take another measurement. The difference between the second and the first measurement represents data without the AL noise.

When designing the product housing, it is imperative to take into account the material to be used as the window for the transmitter and receiver to the outside world. Materials have a transmission coefficient for certain wavelengths, and you should pick a material that allows most of the IR light to pass through it. The manufacturer of the material should be able to provide such coefficient. Sometimes the transmission coefficient for a particular material is specified in terms of the visible light range so make sure to obtain the coefficient for the IR range.

Although in most applications there is not much to do in this regard, it is important to consider the characteristics of the reflective object. The size and shape of the object determines how much of the emitter’s beam width is reflected, and the shape determines the angle of reflection. Some objects (such as a human hand) absorb some of the incident IR energy, so consider this fact during the design process.

Capacitive Touch Sensor
When two conductive objects are placed with a dielectric between them and a potential difference is applied to these conductive objects, an electric field is created between them.

The capacitance of a parallel plate capacitor is determined by formula shown in Figure 6:



$$C = \epsilon_0\epsilon_r * (A / d)$$

where:
ϵ_0 is the permittivity of free space
ϵ_r is the relative permittivity of the dielectric material
A is the area of the plates
D is the distance between the plates

Figure 6: Capacitance of Plate Capacitor

If we take a PCB trace and a human finger (the human body is about 65 percent water) as the two conductive elements and air as the dielectric, the way a capacitive touch sensor works becomes clear from the equation to compute capacitance in the figure above: a finger approaching the PCB trace effectively decreases the distance between the two conductive elements, thus increasing the capacitance.

Several Methods for Measuring Capacitance
Relaxation oscillator: A capacitor is charged and discharged when its voltage crosses a low and a high threshold, respectively. An MCU with an integrated comparator can be used to implement such thresholds. Based on the fact that for a capacitor $\tau = RC$, the frequency of the oscillator will decrease when the capacitance increases. Remember that the capacitance increases as a finger gets closer to the PCB. This frequency is then measured by the MCU to determine the state of the capacitive switch (pressed versus not pressed).

Successive approximation (SAR): A couple of current DACs drive the currents on two capacitors (one used as a reference and the other one serving as the sensing element). The voltages produced on both capacitors, which are directly proportional to the current and inversely proportional to the capacitance, are compared against a voltage reference. This operation is performed several times to complete a measurement. A bit of the capacitance reading is obtained with each comparison, and, depending on the result of each cycle, the unit controlling the current

DACs will increase or decrease the current going through the sensor capacitor to attempt to get the same ramp rate as the reference capacitor. The advantages of this method include immunity to DC offsets, reduced noise susceptibility and minimal requirements for external components (which will be the case if the silicon contains the reference capacitor, DACs and controlling unit).

Charge transfer: A voltage is applied to the sensor, which causes it to accumulate some charge. Then this charge is transferred to an integrated capacitor. This process is repeated until a voltage threshold is reached. The capacitance of the sensor is determined by the voltage in the integrated capacitor (and the ADC, or a simple comparator could be used to extract the threshold information) and the number of cycles it took to reach the predetermined voltage level.

Resistor-capacitor charge timing: This method involves discharging a capacitor and then applying a voltage to an RC circuit. The capacitance is directly proportional to the time it takes the circuit to get to a particular voltage threshold.

"It's more than a button."

While a capacitive sensing element is an excellent replacement for a mechanical button, a set of these elements can be used to create more user-friendly interfaces such as scroll wheels and sliders, which are practical solutions for detecting linear movement along a line or a curve. A slider can be created from only a couple of elements and a wheel from four (Figure 7). Needless to say, an MCU is the perfect fit for a system implementing such human interfaces since it can be programmed to transform the composite measurements into linear or angular positions and communicate these to the rest of the system or to the user, or to take an action on an output subsystem if appropriate.

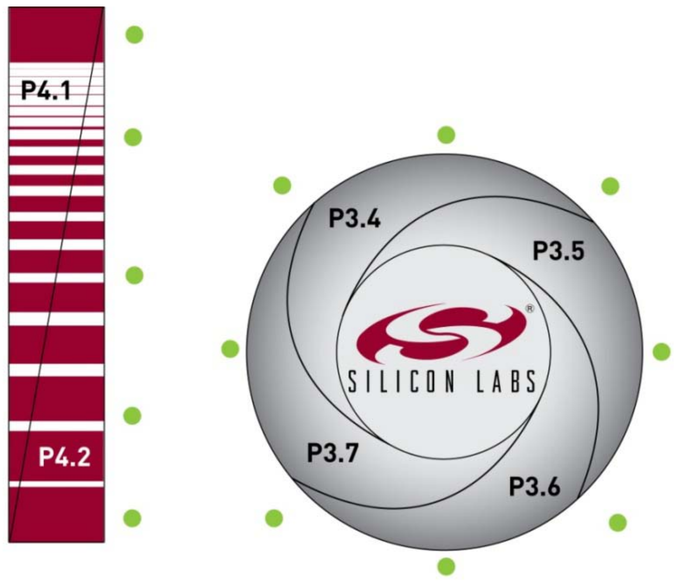


Figure 7 – Slider and Wheel Implementation Using Two and Four Sensors

Capacitive Proximity Sensor
A different end user experience can be achieved by employing the same principles that apply to capacitive sensing buttons by modifying the value of one of the parameters on the capacitance equation, i.e., the larger the area, the more sensitive a button becomes.

The button pad could be implemented in a large enough format to enable dependable detection of conductive objects moving nearby, thus turning the pad into a proximity sensor.

Design considerations
The goal is to build a sensor that presents the largest possible delta measurement between the button not pressed (or inactive) and button pressed (or active) states. The larger the delta, the more room the designer has to adjust the thresholds of each particular switch on the system, resulting in increased reliability. Covering the details of the PCB design is beyond the scope of this article, but some aspects to consider include:

- Capacitive sensing pad layout (proximity to other conductive objects and ground planes) and grounding
- Characteristics and thickness of the dielectric material
- External noise
- Changes in humidity and temperature.

Changes in environmental conditions can be minimized with the aid of an MCU by implementing baselines (a baseline is an average of measurement of a particular switch state over certain period of time) and digital filters.

To minimize the risk of false triggers, an implementation with two thresholds, one for determining the active state and another for determining the inactive state, is desired to take advantage of hysteresis.

Refer to the link below for notes on PCB design:
https://www.silabs.com/Support%20Documents/TechnicalDocs/AN447.pdf

Why incorporate capacitive sensors in my human interface design?
The answer to this question is straightforward: Low cost and high reliability. These benefits stem from the nature of the implementation of the capacitive sensor. It is nothing more than a PCB trace, thus reducing the number of parts, materials and assembly needed. In additions, the buttons implemented with capacitive sensors have no moving parts so the end product becomes more robust mechanically.

By using an interconnected subsystem closely tied together via a flexible, software-configurable platform, it is possible to create a cost-effective capacitive sensor-based human interface implementation that is both functional and scalable. As market needs change and as new ideas emerge, firmware can be adjusted to quickly and easily implement these changes without the need to re-architect the entire system. In addition, by leaving some GPIO pins on the MCU reserved for future use, it is also possible to quickly make hardware additions while leaving the core architecture the same.

One of the hardest challenges that system designers face is envisioning what a compelling human interface solution looks like. By breaking the problem into a series of inputs and outputs and by using a scalable architectural framework, designers can create solutions that go a long way toward meeting the ever-changing definition of human interface.

#   #   #