Code-Free Approach to Touch and Proximity for Embedded Systems

By Mark Lee, Principal Applications Engineer, Cypress Semiconductor Corp.

The capacitive sensing user interface has established itself in recent years as a practical and innovative upgrade for push buttons in many types of consumer goods. A light tap on a smooth surface or the proximity of a hand replaces physical actuation of a mechanical switch.

Capacitive sensing modules make it easy to add a touch and proximity interface to a system since there is no code to write or debug. Just take the module off the shelf and drop it into the system. Sensor modules however are inflexible in the functions they perform, and may limit the features that can be added to a product. A more flexible solution is desirable since it is often the little extras that get added at the last minute that make a product stand out from the competition.

This article shows how to incorporate capacitive sensing features into a simple embedded system without writing a single line of code. This Code-Free approach has the convenience of off-the-shelf modules, with the flexibility of a code-intensive embedded controller. The complete design process is demonstrated for a real-world product idea, from concept to working prototype.

The Code-Free approach to system design is made possible with PSoC Express software and the FirstTouch evaluation kit, both from Cypress Semiconductor. Creative people that brainstorm new products no longer need to throw the design over-the-wall to engineering to get a working prototype. If you can draw a block diagram of your system, you can create a real product using the graphical interface of PSoC Express. The whole process can easily be completed in a couple of hours for someone new to PSoC Express. Development time and project costs are both reduced in the process.

Building a Better Mousetrap

The classic inventor's quest is to build a better mousetrap. The design example in the following discussion comes close to that very result. Let's create an embedded system as a design example that combines two trends in electronics: capacitive sensing and digital photography. Our example project will be a smart proximity sensor that triggers a digital camera. Such a system is called a "camera trap". These cameras have been in science news headlines in recent years. Wildlife researchers have used camera traps to prove that animals once thought extinct are still on the prowl, like jaguars in Arizona [1]. In remote places like the dense forests of Borneo, scientists have also discovered totally new species with these cameras [2].

The existing technology for camera traps seems to be working just fine. Why reinvent the wheel? Here is a short comparison of old and new technology that makes the case for building a better trap.

Old Technology: Based on PIR motion sensor. At high ambient temperatures, animals in the target zone get lost in the background scene. Reflected sunlight gives false detects. The bulky hardware needs to be hidden from the animals and protected from elements, yet any blockage to optical path will disable trigger. The trigger zone is hard to customize with simple lenses.

New Technology: Based on Capacitive Proximity Sensor using a wire as the proximity sensor. Stealthy wire can be attached to branch, across a rock or by a watering hole. Trigger zone is very localized and fine tuned with a wire clipper. Works equally well in bright sunlight and starry night. Background heat has no effect on performance. The proximity sensor wakes up periodically, and wakes the camera only when an animal is in target zone, saving on battery power.

Comparing the two approaches to building a camera trap, it is seen that the capacitive sensing approach to camera traps has advantages over the existing approach in certain situations. The new approach should find a niche in the field of wildlife research.

Design Process Overview

There are six steps in the Code-Free design process.

1. Describe the system in words.

2. Draw system block diagram.

3. Define state machines, transfer functions and truth tables.

4. Simulate system.

5. Test the real system.

6. Fine-tune CapSense.

Step 1. Describe System in Words

We will take a top down approach to design. This means define in words at a high level how the system should operate, and then fill in the technical details as needed. Here is a short non-technical description of the system.

The system will constantly monitor for any animal. When an animal is detected, the camera is turned on and a picture snapped. The camera is off when no animal is detected. To prevent pictures from being taken of the researcher setting the camera trap, the detection system is disarmed for a period when the system is reset.

These few sentences are a good start, but more detail is needed before diving into PSoC Express. Here is a more technical description of how the system should work.

1. Set up the camera.

2. Hide a wire in the target zone. For example, attach the wire to a branch. Wire insulation should blend in with the environment.

3. Connect sensor wire and camera control cable to the controller board.

4. Turn on the power to the controller board.

5. A red LED flashes for 30 seconds allowing the researcher to move away from the system without triggering a picture. The camera trigger is disarmed during this time and the camera is off.

6. The red LED stops flashing and stays off.

7. The trigger to the camera is now armed, waiting for an animal to move near the proximity sensor.

8. When an animal triggers the proximity sensor, a green LED turns on. The camera is turned on and pictures are taken.

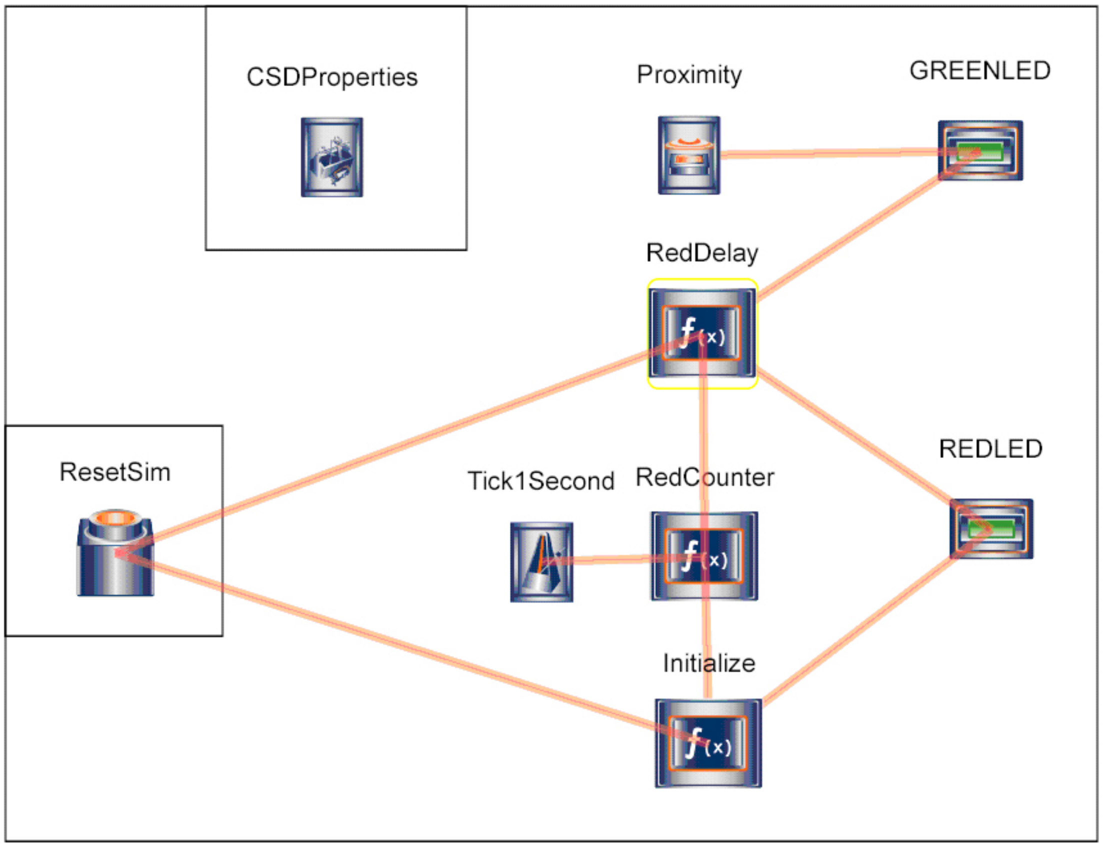
9. When the animal moves out of range, the green LED turns off. The camera turns itself off after a period of no input.

10. Go back to step 7 and repeat loop.

Step 2. Draw System Block Diagram

A block diagram of this system is drawn using PSoC Express, as shown in Figure 1. The diagram is defined through a graphical drag-and-drop process. PSoC Express has a library of parts called a driver catalog that includes such functional blocks as temperature sensors, light sensors, and accelerometers. Blocks are dragged from the catalog as needed onto the block diagram using a mouse. To keep things simple, the only outputs to the system in this design example are the red and green LED. It is a simple matter to add the "camera on/off" and "snap picture" outputs to the camera by grabbing two more digital outputs from the driver catalog.

Figure 1. System block diagram.



The system for this design example is composed of the following blocks.

Proximity - A proximity sensor based on the CSD method of CapSense.

Green LED - turns on when the proximity sensor has been triggered.

Red LED - flashes during the 30 second Disarm period after reset.

Tick1Second - The timebase that outputs a pulse train at a rate of 1 pulse per second.

RedCounter - A counter that increments once per second.

RedDelay - A state machine that changes state after 30 seconds.

Initialize - A state machine that resets the counter when the system is reset.

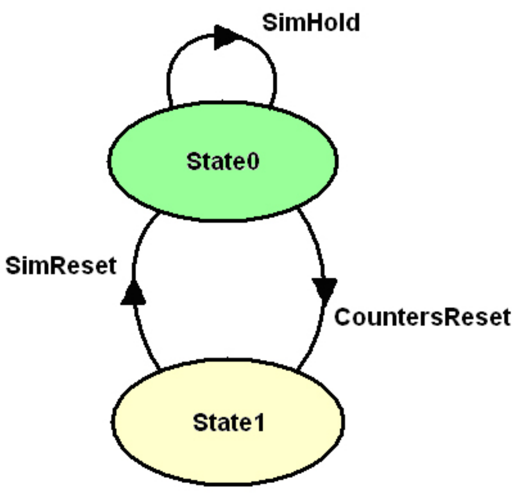
State 3. Define State Machines, Transfer Functions and Truth Tables.

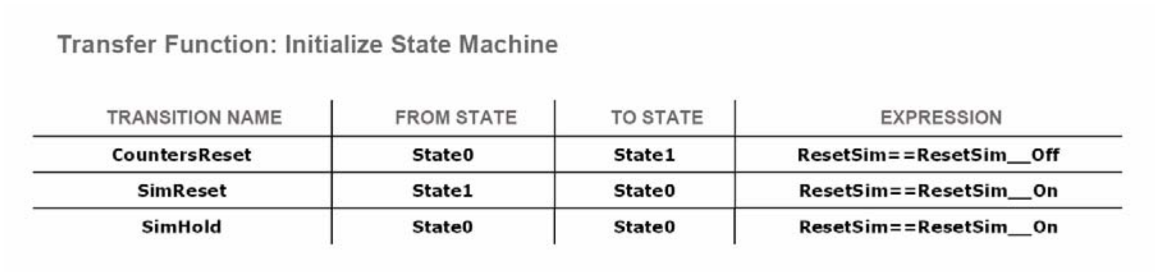
Although no code is written with PSoC Express, a logical descritption of he system needs to be defined, and this is done through state machines, transfer functions, and truth tables.

There are two simple state machines in the design, and they have the same basic structure. The state diagram is drawn in PSoC Express, as shown in Figure 2 and 3. The transition logic is shown in the tables that are included in each figure.

The state machine called Initialize is included in the system to make simulations more user friendly under the control of the reset switch called ResetSim. As shown in the transfer function table of Figure 2, this state machine is held in State0 whenever the reset switch is ON. Otherwise the state machine is in State1, allowing the counter to increment.

Figure 2. State diagram for the state machine called Initialize.



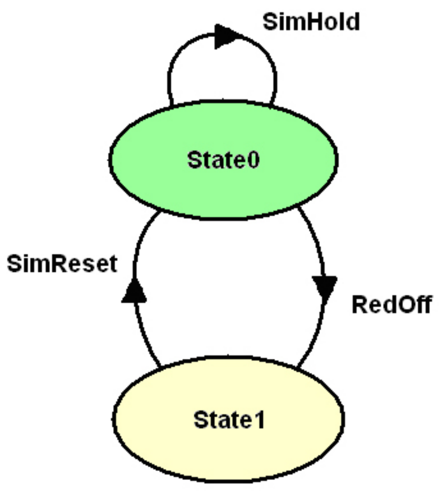


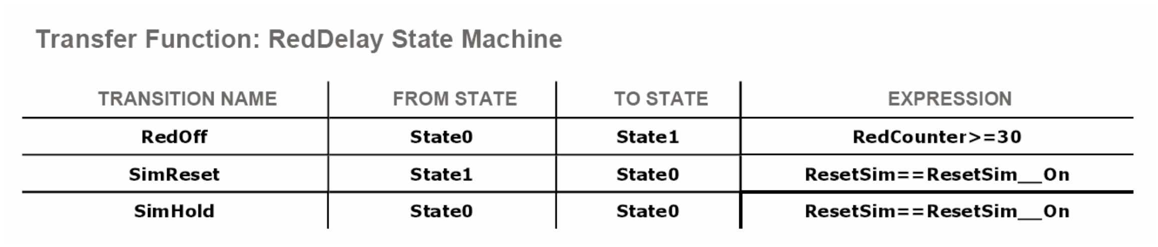
Transfer Function: Initialize State Machine

TRANSITION NAME	FROM STATE	TO STATE	EXPRESSION
CountersReset	State0	State1	ResetSim==ResetSim__Off
SimReset	State1	State0	ResetSim==ResetSim__On
SimHold	State0	State0	ResetSim==ResetSim__On

The state machine called RedDelay is added to the system to indicate when 30 seconds have elapsed since the system was reset. The transfer function table in Figure 3 is self-explanatory.

Figure 3. State diagram for the state machine called RedDelay.



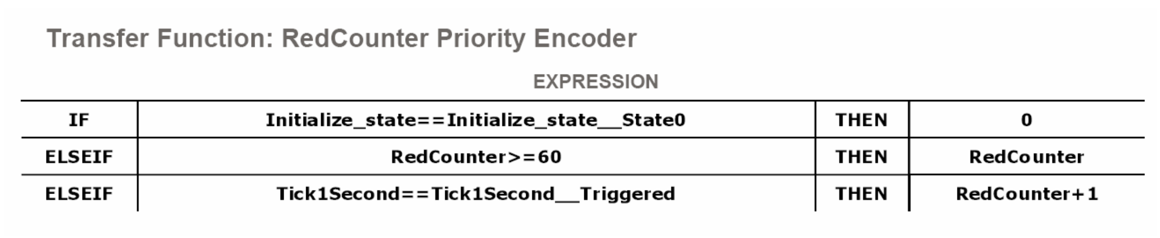


Transfer Function: RedDelay State Machine

TRANSITION NAME	FROM STATE	TO STATE	EXPRESSION
RedOff	State0	State1	RedCounter>=30
SimReset	State1	State0	ResetSim==ResetSim__On
SimHold	State0	State0	ResetSim==ResetSim__On

There are three truth tables in the design. These tables are a special type called a priority encoders. The encoder has a single output, and its output setting is controlled by the logic expressions entered in the table. These logic rules are tested in sequence until all the conditions for one of the rules are met. These three tables are used to increment the counter and turn the LEDs on and off, as shown in Figure 4, 5, and 6.

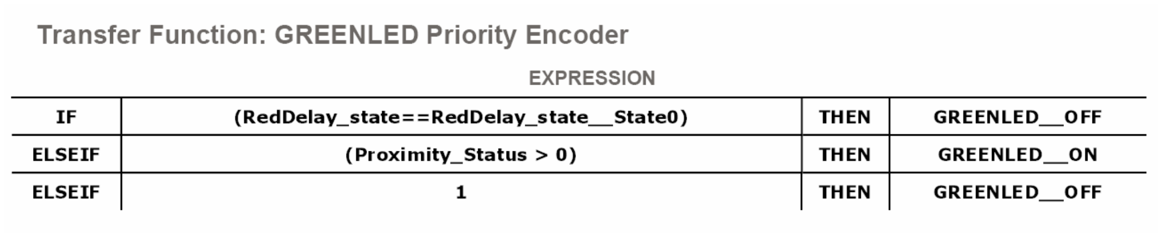
Figure 4. RedCounter definition.



Transfer Function: RedCounter Priority Encoder

	EXPRESSION		
IF	Initialize_state==Initialize_state__State0	THEN	0
ELSEIF	RedCounter>=60	THEN	RedCounter
ELSEIF	Tick1Second==Tick1Second__Triggered	THEN	RedCounter+1

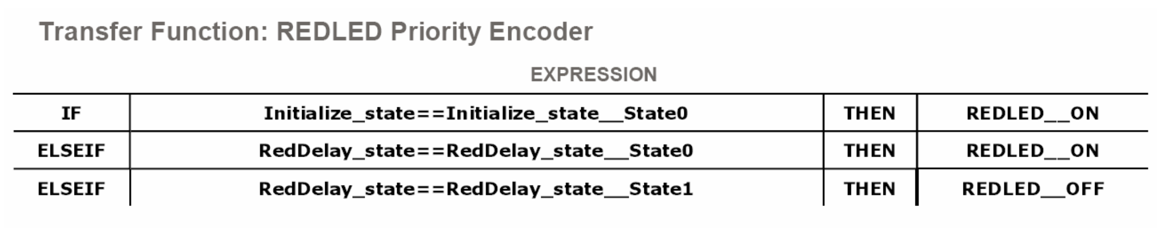
Figure 5. GreenLED definition.



Transfer Function: GREENLED Priority Encoder

	EXPRESSION		
IF	(RedDelay_state==RedDelay_state__State0)	THEN	GREENLED__OFF
ELSEIF	(Proximity_Status > 0)	THEN	GREENLED__ON
ELSEIF	1	THEN	GREENLED__OFF

Figure 6. RedLED definition.



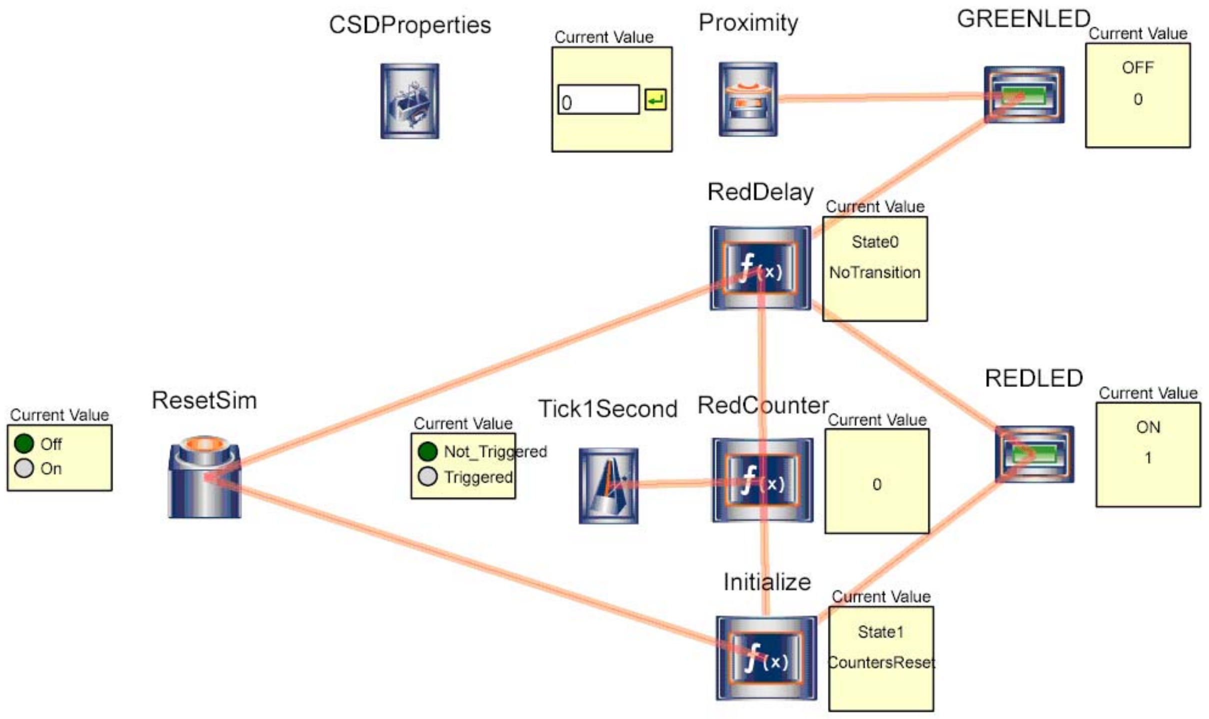
Transfer Function: REDLED Priority Encoder

	EXPRESSION		
IF	Initialize_state==Initialize_state__State0	THEN	REDLED__ON
ELSEIF	RedDelay_state==RedDelay_state__State0	THEN	REDLED__ON
ELSEIF	RedDelay_state==RedDelay_state__State1	THEN	REDLED__OFF

Step 4. Simulate System

One of the powerful features of PSoC Express is simulation. The simulation view of the design example is shown in Figure 7. This tool allows you the test whether the output of the system transitions at the appropriate times. The simulation allows you to run what-if experiments without ever programming a part. Once everything is working as planned, it is time to program a part and check out how well the system works in the real-world.

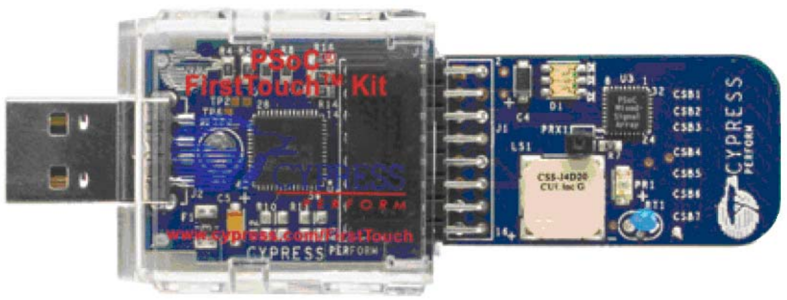
Figure 7. Simulation of the system.



Step 5. Test the Real System

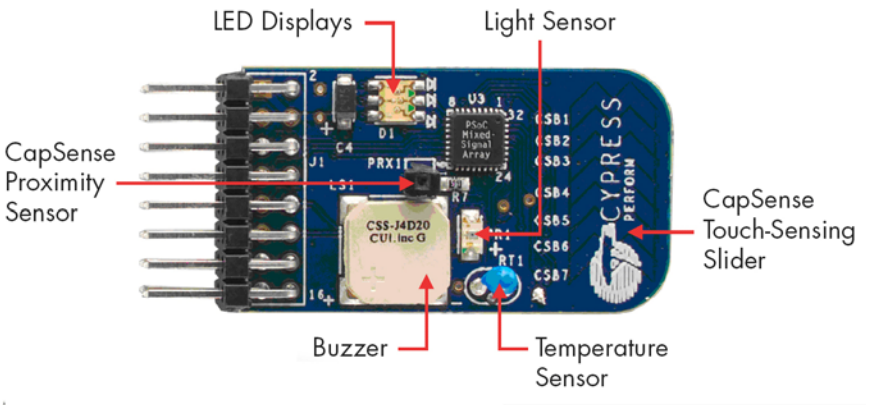
The FirstTouch evaluation kit is low cost (approx. US$30). It is packaged in the format of a USB memory stick as shown in Figure 8 and 9. Everything needed to evaluate the design example is included on this tiny development system.

Figure 8. FirstTouch kit showing programmer with USB interface, and development board.



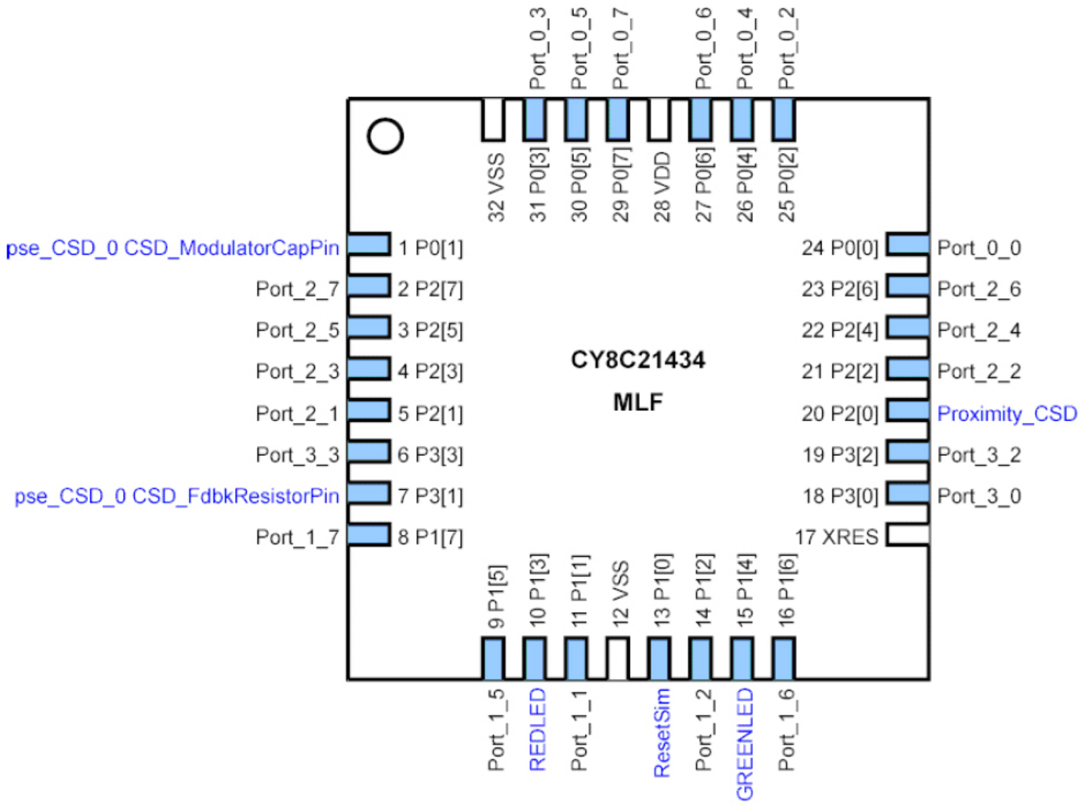
[+] Feedback

Figure 9. Detailed view of FirstTouch kit development board, including connection point for proximity sensor wire.



The process of programming the FirstTouch kit includes selecting the correct PSoC, CY8C21434, and assigning the correct pins for each function, as shown in Figure 10. The hex file for project was created within PSoC Express, and programmed into the FirstTouch kit. The behavior of the real system mirrors the prediction from the simulation, which in turn matches the system requirements defined earlier. The project is a success, and the whole process only took a couple hours for a beginner. For experienced users of PSoC Express, the design process is even faster.

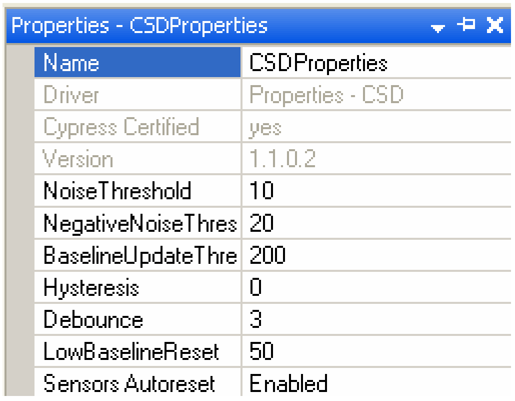
Figure 10. Pin assignments for the FirstTouch development board project.



Step 6. Fine-Tune CapSense

One feature not covered in the simulation is the fine-tuning of the CapSense sensor block. The CSD and Proximity property settings that work well with the FirstTouch kit are shown in Figures 11 and 12. More information on CSD and proximity can be found in the CapSense Best Practices Application Note from Cypress Semiconductor [3].

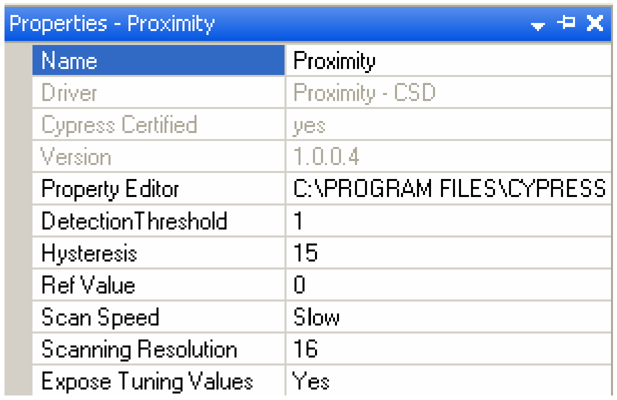
Figure 11. Properties of CSD Properties



Properties - CSDProperties

Name	CSDProperties
Driver	Properties - CSD
Cypress Certified	yes
Version	1.1.0.2
NoiseThreshold	10
NegativeNoiseThres	20
BaselineUpdateThre	200
Hysteresis	0
Debounce	3
LowBaselineReset	50
Sensors Autoreset	Enabled

Figure 12. Properties of Proximity Sensor



Properties - Proximity

Name	Proximity
Driver	Proximity - CSD
Cypress Certified	yes
Version	1.0.0.4
Property Editor	C:\PROGRAM FILES\CYPRESS
DetectionThreshold	1
Hysteresis	15
Ref Value	0
Scan Speed	Slow
Scanning Resolution	16
Expose Tuning Values	Yes

Going Further

Imagine a scenario where a researcher wants to photograph animals that are in proximity to the branch where the trigger is located, but ignore animals that actually touch the branch. For example, a snake or snail could set off the proximity sensor when crawling along the branch, and this researcher does not desire pictures of such creatures. To capture pictures of animals that pass near the proximity sensor but don't touch it, just place additional touch sensors along the stick. Making this change in PSoC Express involves selecting CapSense buttons from the driver catalog, and then changing the logic for turning on the Green LED (ON when Proximity_status >0 and Button_status <1). That's one smart mouetrap!

This was a brief introduction to PSoC Express. Detailed help can be found on-line at Cypress.com. One place to start with more in-depth study is Application Note AN2261, a Primer on PSoC Express [5].

References

[1] Will Rizzo, "Return of the Jaguar?", Smithsonian Magazine, December 2005

[2] Shaoni Bhattacharya, "Mystery Mammal Discovered in Borneo's Forests", NewScientist.com News Service, December 6, 2005

[3] Application Note AN2394, " CapSense Best Practices", Cypress Semiconductor

[4] Application Note AN2292, " Layout Guidelines for PSoC CapSense", Cypress Semiconductor

[5] Application Note AN2261, "Design Aids - PSoC Express™ 2.0 Primer: First Introduction", Cypress Semiconductor

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