AN0028: Low Energy Sensor Interface — Capacitive Sense

This application note covers the basics of capacitive sense and describes how to use the Low Energy Sensor Interface (LESENSE) to scan a number of capacitive sensors while remaining in EM2, achieving current consumption around 1.5 µA. Operation through several millimeters of plastic, glass or similar non-conductive overlay is also possible.

The software example simplifies LESENSE configuration for capacitive touch while operating with minimal energy consumption. It works on the EFM32 Tiny Gecko Starter Kit and the EFM32 Giant Gecko Starter Kit.

This application note focuses on how to implement capacitive touch with the EFM32 Series 0 microcontrollers. For hardware design of printed circuit boards for capacitive touch, please refer to the capacitive touch hardware design application note (AN0040).

For simplicity, EFM32 Gecko Series 0 is used throughout this document to represent the EFM32 Wonder Gecko, Gecko, Giant Gecko, Leopard Gecko, Tiny Gecko, Zero Gecko, or Happy Gecko MCU series.

KEY POINTS

- This application note includes:
  - This PDF document
  - Source files (zip)
    - Example C-code
    - Multiple IDE projects

1. Introduction

1.1 Capacitive Sensing

Capacitive sensing is a technology now widespread across different industries. High performance capacitive sensors are capable of high resolution measurements of proximity, position, humidity, fluid levels or acceleration of a conductive target. Lower cost capacitive touch sensors are less advanced and mostly used for human-device interface by measuring the capacitance change when a user's finger is near by. These kind of sensors are increasingly common in mobile devices of all sorts.

This application note will focus on the second type of sensors used to interface with users in a variety of applications. They can be implemented at very low cost and bring many advantages over mechanical switches, such as having no moving parts and less degradation over time, with usage, and environmental changes. The EFM32 capacitive touch feature is primarily made for implementing capacitive touch buttons and sliders with very low energy consumption, but it can also be adapted to other capacitive sensing applications.

EFM32 devices with the Low Energy Sensor Interface can use this peripheral to scan several touch pads and only wake the CPU when a touch is detected. To get the best performance possible, it helps to understand what affects performance and which parameters can be tuned to increase performance and decrease current consumption. Although understanding the possibilities and limitations of LESENSE really helps when configuring a custom capacitive touch application, the software example included should make it easy to implement capacitive touch with good performance, quickly.

1.2 LESENSE

The Low Energy Sensor Interface (LESENSE) is a peripheral state machine that utilizes other on-chip standard peripherals to perform sequenced measurement of a configurable set of analog sensors. LESENSE uses the analog comparators (ACMP) for measurement of sensor signals. The following figure gives an overview of the LESENSE peripheral and how it is connected to the analog comparators.

With LESENSE, almost every imaginable sensor interface tasks which uses analog comparators, DACs and counters can be automated and handled while the EFM32 is in EM2 (deep sleep mode). A wake up to EM0 (run mode) is only needed when sensor readings change and a trigger threshold is reached or when some higher level calibration is needed.

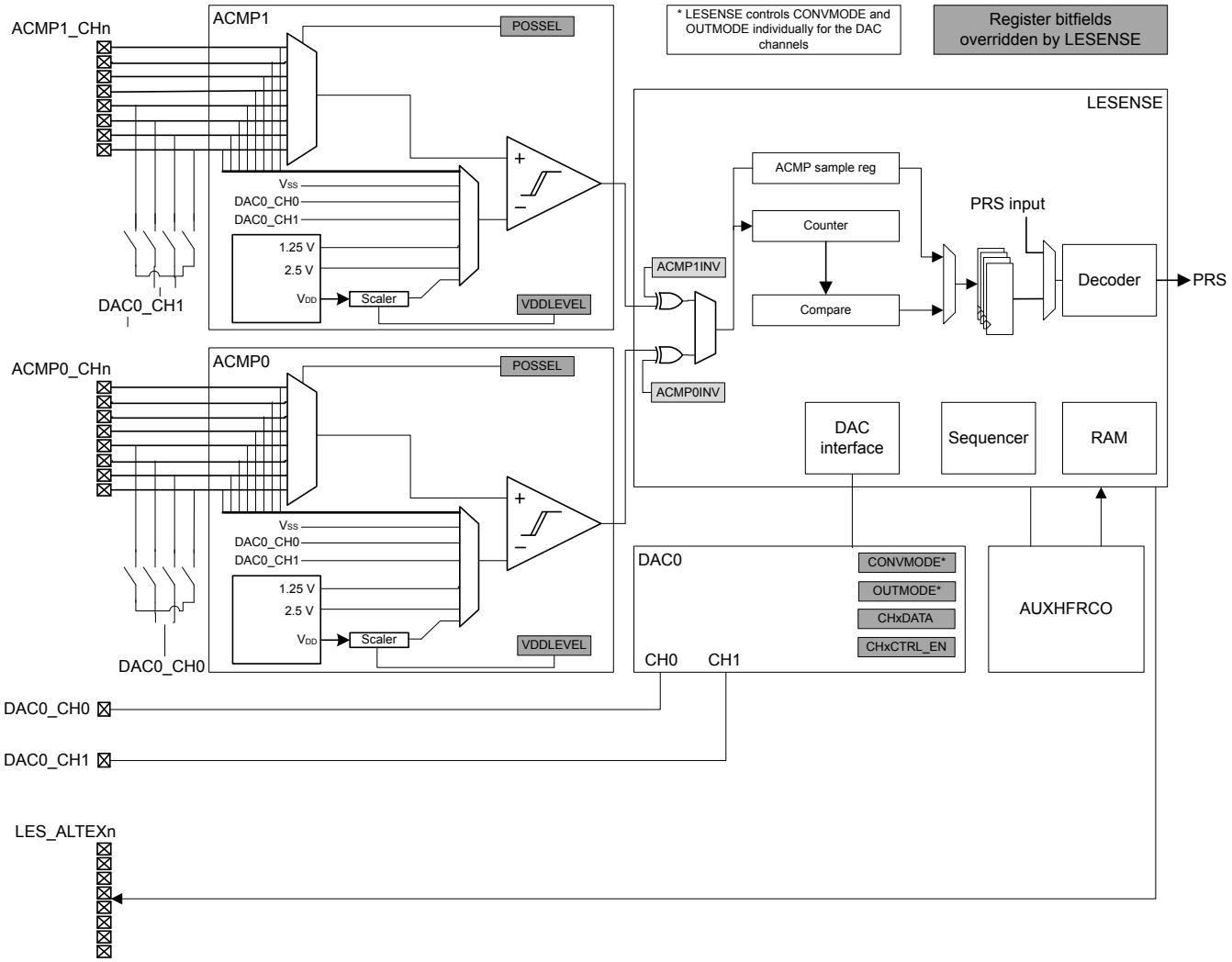


Figure 1.1. LESENSE Overview

LESENSE supports multiple sensor types: inductive (LC), capacitive, and general analog sensors. This application note focuses on the configuration of LESENSE and the analog comparators to scan capacitive touch sensors, and how the different settings affects the performance and power consumption.

2. Capacitive Touch Sensing

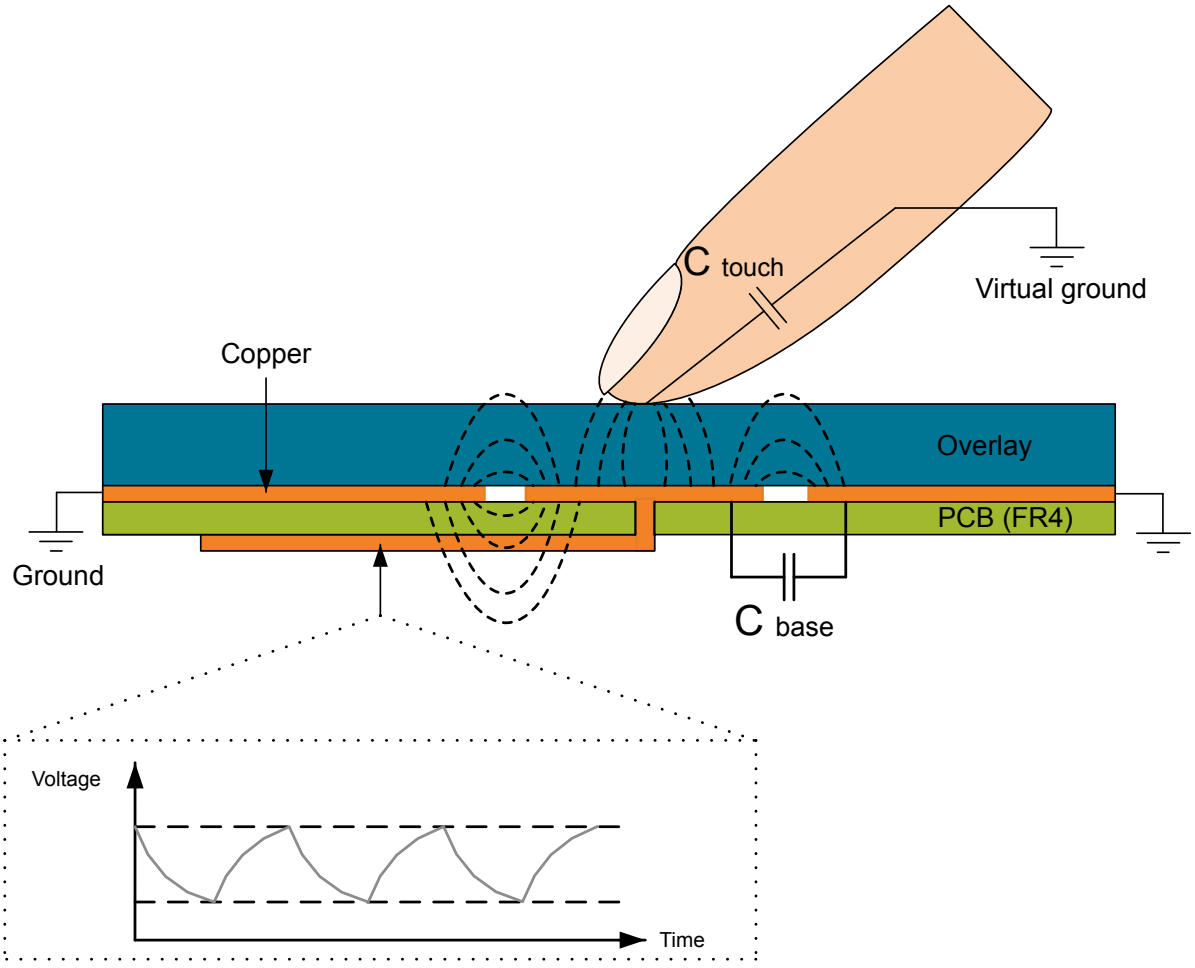


Figure 2.1. Capacitive Sense Overview

2.1 Theory

The working principle of a capacitive touch (or proximity) sensor is to measure the change in capacitance of a given, and otherwise constant, capacitor when approached or touched by a larger body such as a human finger or hand. The capacitive touch pad can be implemented with different technologies, ranging from a trace on a printed circuit board to various conductive coated surfaces like glass. The printed circuit board approach can of course be used with an overlay, typically made of plastic or glass.

If the two metal plates are placed with a gap between them and a voltage is applied to one of the plates, an electric field will exist between the plates. This electric field is the result of the difference between electric charges that are stored on the surfaces of the plates. Capacitance refers to the "capacity" of the two plates to "hold" this charge, i.e. a large capacitance is able to hold more charge than a small capacitance. The amount of existing charge determines how much current must be used to change the voltage on the plate.

When implementing a capacitive touch sensor there are a few different possible configurations which can be categorised in two different types, self- and mutual-capacitance. The difference is what the other side of the measured capacitance is. It can either be another electrode with different potential than ground, which is the mutual-capacitance type of setup. Or it can be ground, which is the self-capacitance type of setup. The EFM32 is designed for self-capacitance measurements where the absolute capacitance to ground is the measured entity.

2.2 Capacitive Sense for the EFM32 Series 0

The analog comparators in the EFM32 Gecko Series 0 include specialized hardware for capacitive sensing of capacitive pads. The capacitance is measured by including the pad as a capacitance to ground in a free-running RC oscillator (see the following figure).

By keeping the resistor, drive voltage, and reaction speed of the analog comparator constant, the frequency produced will ideally only be dependent on the pad to ground capacitance. Because the user's finger represents both a change in the dielectric material between the plates (pad and ground) and a parallel capacitor to (virtual) ground, it will be seen as a change in the pad capacitance upon touch, see Figure 2.1 Capacitive Sense Overview on page 3.

The frequency produced will thereby decrease when the button is touched, compared to when it is not touched. This frequency change can be measured. LESENSE does this by counting analog comparator pulses within a fixed time window. A change in count value is used as an indication of a button being touched or not.

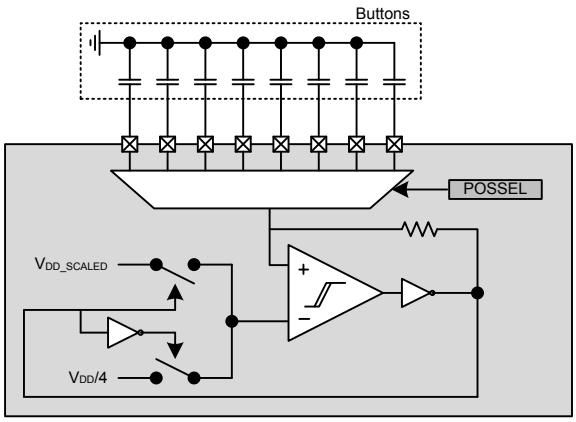


Figure 2.2. Capacitive Sense in the EFM32

The analog comparator contains a complete feedback loop with an adjustable resistor through which the capacitive pad is charged and discharged. The output of the analog comparator drives the switches that select the reference for the negative input of the comparator. This feedback will in turn cause the capacitor to charge or discharge as depicted in the following figure. The voltage will oscillate between $V_{DD}/4$ (lower voltage threshold) and V_{DD_SCALED} (upper voltage threshold).

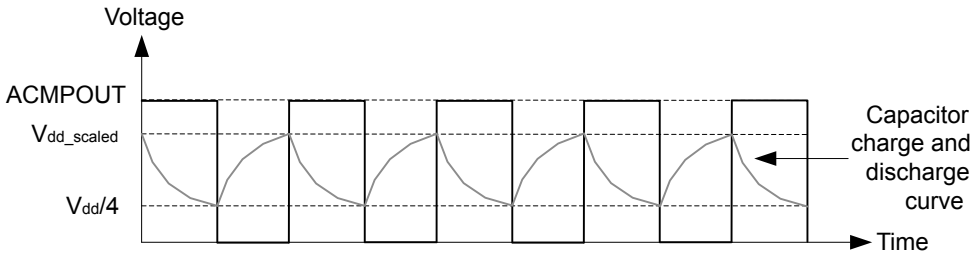


Figure 2.3. Capacitor Charge and Discharge

When the output of the ACMP is high, $V_{DD}/4$ is selected as reference, and the capacitor starts to discharge due to the inverter. When the capacitor voltage drops below $V_{DD}/4$ the ACMP output toggles. V_{DD_SCALED} is selected as reference and the capacitor starts charging. When the capacitor voltage again reaches V_{DD_SCALED} the ACMP output toggles and the cycle repeats. The closer V_{DD_SCALED} and $V_{DD}/4$ are the shorter the oscillation period will be. The V_{DD_SCALED} reference can be adjusted using the VDDLEVEL bitfield in ACMPn_INPUTSEL register. The resulting voltage is given by:

$$V_{DD\_SCALED} = V_{DD} \times \frac{\text{VDDLEVEL}}{63}$$

The frequency of the oscillation is mainly determined by V_{DD_SCALED}, the resistor, the capacitance, and the delay through the analog comparator. This makes calculating the exact frequency difficult. Fortunately, it is only the change in frequency that is important when detecting a touch.

2.3 External Disturbances

Touch detection is based on detecting a difference from a "normal" capacitance value. Because of this, it is necessary to keep track of what the untouched capacitance value really is. The problem is that this value will change with the environment. Changes in humidity, for example, will change the base capacitance.

Since the measurement itself is a frequency count during a fixed time window, jitter in measurement-period also affects the untouched reading of a button in addition to the button capacitance. For example, when the supply voltage, V_{DD} changes, the frequency will also change because of a change in pull-level for the RC-oscillator. Another effect that will cause noise in the readout is changes in the sample time window. This can for example be caused by low accuracy oscillator or changes in temperature.

Because a touch event is assumed to be rather quick in nature, slow changes in capacitance and count values are generally easy to filter out by low pass filtering or taking the median value over time. Fast changing disturbances are more difficult to filter out since a fast change in capacitance could either be a touching finger or just noise.

As a general rule, slow changes like changes in battery voltage when the battery is discharged, changes in humidity or oscillator frequency changes caused by temperature can easily be filtered out. Disturbances that can look like a finger touch on the other hand needs to be minimized, this means that noise on V_{DD}, oscillator jitter and radiated noise from other signals on the PCB should be minimized.

2.4 Signal to Noise Ratio

The change in capacitance when a finger is placed near the sensor element is generally very small. This means that the noise level in the count values needs to be even smaller to be able to reliably detect touches. As a rule of thumb, the signal to noise ratio should be above 5. The "signal" refers to the difference between the average count for touch and no touch. The ratio between this difference and the noise-jitter for an untouched button is then the signal to noise ratio. See the figure below for an example of signal to noise ratio.

Note: As a rule of thumb, the signal to noise ratio should be above 5 for good performance.

When developing a capacitive touch application one should check that the signal to noise ratio criterion is fulfilled for the actual PCB with the final overlay (same material and thickness that will be used). For evaluating the signal level the register view can be used in debug mode, or the actual values can be sent over a serial connection to a computer for analysis.

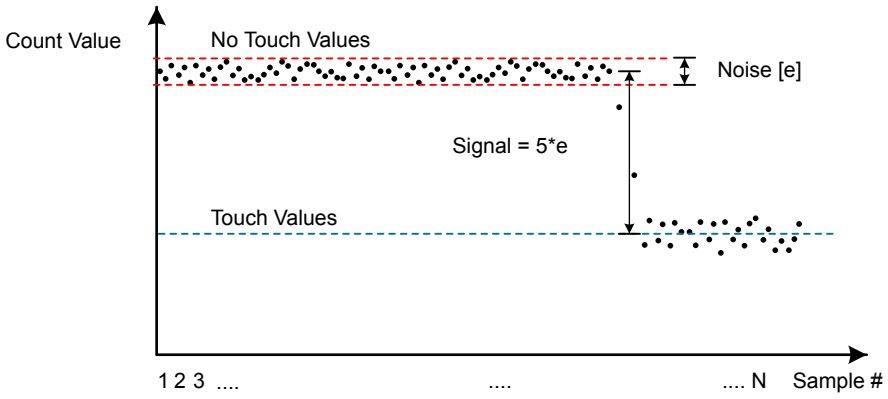


Figure 2.4. Signal to Noise Ratio

3. LESENSE for Capacitive Touch

3.1 Capacitive Touch Sequence

For capacitive touch, there is no need for many of the advanced features of LESENSE like excitation phase, measure delay, or start delay configurations (see the relevant reference manual). Only the sample delay phase is necessary, and the following list indicates the flow of a measurement for one capacitive touch button. See the following figure for a typical capacitive touch measurement sequence.

1. The analog comparators are switched on in capacitive touch mode. This starts the RC-oscillations on the selected channel.
2. The analog comparator output is gated to the LESENSE counter which starts counting. The sample delay counter is started at the same time.
3. When the sample delay counter reaches the configured sample delay, the counted value is stored and compared to a threshold value.
4. If the count compare is true, an interrupt request is sent. Either way, LESENSE turns off the analog comparators again and sleeps until the next scan cycle. Notice that the CPU itself is never woken up during a measurement cycle.

If LESENSE is configured to scan more than one channel the sequence will repeat itself for each channel in rapid succession before going to sleep. The scan period and sample delay should be configured so there is time to complete the measurement for all channels before a new scan is started. If the desired scan frequency is 100 Hz for example, each scan should not take more than 10 ms, which again must be divided between all the channels that is enabled.

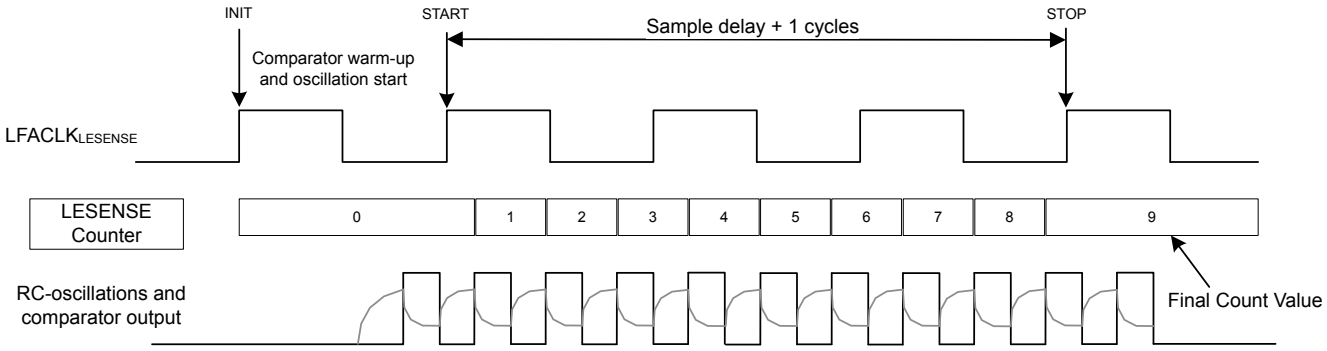


Figure 3.1. LESENSE Sequence for Capacitive Touch

3.2 LESENSE Configuration

To get the lowest noise level and thereby highest signal to noise ratio, it is important to configure both the analog comparators and LESENSE correctly. The following section lists the different settings, how they affect performance and what they should be set to or how to tune them. There are other settings that are not directly connected to capacitive touch, the configuration of those can be found in the software example.

3.2.1 Fixed Settings

These are settings that can be set to a fixed value which should be optimal for most capacitive touch applications.

Table 3.1. Fixed Settings for Best Capacitive Touch Performance

Setting	Optimal value	Description
LF oscillator	LFXO	By using a crystal oscillator the length of the sample window will be much more stable. The noise level can be reduced by a factor of 30-40 by using the LFXO instead of the LFRCO. Keep in mind that the LFXO needs an external crystal or clock input to operate.
LESENSE clock prescaler	0: Divide by 1	This is how much the LF clock is prescaled before clocking the LESENSE module. To get best performance the LESENSE module can just be run directly off of the LF clock for capacitive touch applications.
Analog comparator bias, register values	• Full Bias: 1 • Half Bias: 1 • Bias Prog: 0x5-0x7	The bias current affects the speed of the analog comparator. If the comparator bias current is too low the comparator will cause a significant delay for each oscillation. This delay will reduce the frequency, but more importantly the delay is variable and introduces noise-jitter in readings. As long as the bias current is high enough these effects are minimized.
RC-oscillator resistor, register value	0: Smallest resistor	The resistor affects the charge/discharge time of the capacitive touch element. Higher resistor values corresponds to higher time-constant for the RC-oscillator. The lowest possible value gives the fastest oscillations and reduce the needed sample delay.
Analog comparator hysteresis, register value	0x5	The hysteresis needs to be high enough to not cause false triggering.
Analog comparator V_{DD_SCALED}, register value	0x30	This value determines when the comparator triggers on rising edges. The low threshold is fixed at 0.25 * V_{DD}. By changing the V_{DD_SCALED} value, the oscillation speed can be modified. A value of 0.75 * V_{DD} ensures a stable frequency without jitter. If this value is to close to either V_{DD} or the low trigger threshold, cycle to cycle jitter will increase, which in turn decreases the SNR.
Start, excitation and measure delay in clock cycles for the selected clock source.	• Start Delay: 0 • Excitation Delay: 0 • Measure Delay: 0	These delays do not affect capacitive touch applications and should be set to 0.

3.2.2 Adjustable Settings

These settings can and should be tuned for different capacitive touch applications. A good starting point for a general capacitive touch application is given.

Table 3.2. Adjustable Settings for Best Capacitive Touch Performance

Setting	Starting point	Description
LESENSE scan frequency	5-20 Hz	This is how often a scan is started. Each enabled channel is scanned once for every scan start. A good approach to save energy can be to have a lower setting when waiting for the first touch and then adjust the value for subsequent touches to give a more responsive user experience.
Sample delay	10-100 LESENSE clock cycles	This is the time window measured in LESENSE clock cycles that LESENSE gates the analog comparator output to the counter. The counted value will jitter with 1 LSB even with 100% stable input frequency because the comparator signal is asynchronous with respect to the LESENSE clock. The delay should be high enough to allow for a difference of at least a few LSB's between touch and no touch. Reducing this value is the main contribution to lower the energy consumption.
Count threshold	1.0% - 50% change from average count value.	This is the threshold that LESENSE compares the count value against in order to determine if the channel in question is touched or not. It should be set so that a touch is detected at the moment a finger touches the overlay surface. The value is the percent of change from average untouched count. The minimum of 1.0% is based on empirical results using the EFM32 Tiny Gecko Starter Kit with LFXO and observing the noise level. This is then multiplied by 5 for a signal to noise ratio of 5. For a custom PCB this value might be even lower. By using percent instead of a fixed value the sample delay can be changed without also changing this threshold manually, given that the code calculates the actual value during initialization and calibration. In any case it should be kept above 5 times the noise level which results in a signal to noise ratio above 5 as mentioned earlier. In the software example included, these values are configured through the sensitivity[] array.

3.3 Performance Trade-off

The performance trade-off for a capacitive touch application is typically between current consumption and sensitivity. The sensitivity determines how thick the overlay can be. These two parameters are inversely correlated, for thick overlays more sensitivity is needed and the current consumption goes up. For thin overlays, or even touch buttons directly on the PCB, the current consumption can be brought down because a shorter measurement window is sufficient to determine if a touch has occurred.

For LESENSE the sensitivity is increased by increasing the sample delay. This increases the count value and this in turn gives higher resolution in the measurements. While increasing the sample delay, the time the analog comparators are kept running also increases. This increases the current consumption.

The approach to get the correct trade off between low power and good performance is to start out with a large sample delay and first configure the sensitivity by setting the threshold value. The count threshold should be set to a value which gives a touch event at the moment a finger is touching the overlay. Then the sample delay can be decreased until the count value only changes by a few LSBs when a touch occurs.

3.3.1 Current Consumption

The software example included, configured for scanning 4 pads at 5 Hz per pad through 1.8 - 2.0 mm of plastic, consumes around 3.4 µA. Scanning just 1 pad with the same settings reduces the consumption to approximately 2.1 µA. The current consumption scales linearly with scan-frequency, sample delay and the number of pads scanned.

If the PCB layout is made more ideal than the starter kit, with larger pads (1.5 - 2.0 cm), a thicker overlay can be implemented. With the same settings as above, the total consumption with 6mm overlay for 4 buttons (2.0 cm) at 5 Hz per channel is 3.1 µA. Some examples of achievable current consumption for different configurations are listed in the table below.

Table 3.3. Current Consumption for Different Software and Hardware Configurations

Hardware	Overlay Thickness (Plexiglass)	Sample Delay	Count Threshold	LESENSE Scan Frequency Per Channel	Buttons Scanned	Current Consumption (at 3.3V)
STK3300	0 mm	1 LFCLK-cycles	40%	5 Hz	1	1.7 µA
STK3300	0 mm	1 LFCLK-cycles	40%	5 Hz	4	1.8 µA
STK3300	0 mm	1 LFCLK-cycles	40%	10 Hz	4	1.9 µA
STK3300	2 mm	30 LFCLK-cycles	1%	5 Hz	1	2.1 µA
STK3300	2 mm	30 LFCLK-cycles	1%	5 Hz	4	3.4 µA
STK3300	2 mm	30 LFCLK-cycles	1%	10 Hz	4	5.0 µA
Custom PCB, 2 cm buttons	2 mm	10 LFCLK-cycles	10%	5 Hz	1	2.0 µA
Custom PCB, 2 cm buttons	2 mm	10 LFCLK-cycles	10%	5 Hz	4	2.5 µA
Custom PCB, 2 cm buttons	2 mm	10 LFCLK-cycles	10%	10 Hz	4	3.1 µA
Custom PCB, 2 cm buttons	6 mm	20 LFCLK-cycles	4%	5 Hz	1	2.1 µA
Custom PCB, 2 cm buttons	6 mm	20 LFCLK-cycles	4%	5 Hz	4	3.1 µA
Custom PCB, 2 cm buttons	6 mm	20 LFCLK-cycles	4%	10 Hz	4	4.3 µA

3.3.2 Temperature and Voltage Variations

The actual count value will vary with voltage and temperature because the analog comparators performance changes with supply voltage and temperature. Because of this, continuous software calibration is necessary when the count value only changes a few percent upon a touch event. How often a calibration is necessary depends on how quickly the temperature and voltage is expected to change. The actual count value decreases linearly with increasing temperature, about 15% over the temperature range from -40°C to 85°C.

The change over supply voltage is also linear with respect to the change in voltage. The count value increases with increasing supply voltage, about 20% over the supply range from 2.0 V to 3.6 V.

4. Capacitive Sense Library

The Capacitive Sense Library (cslib) supports all EFM32 devices. More information about the Capacitive Sense Library can be found in *AN0828: Capacitive Sensing Library Overview*.

Most devices have examples using the cslib library available in the Gecko SDK v4.4.0 (v4.4.1 in Simplicity Studio) or later in Simplicity Studio v4. To access these examples for the STKs:

Note: Not all STKs have cslib examples due to pin or tool conflicts. However, the library is supported on all EFM32 devices.

1. Connect the STK to the PC where Simplicity Studio v4 is installed. To open and build the example without having a kit connected:
   a. Click the [**Solutions**] tab.
   b. Click the [**New Solution**] button.
   c. Select [**Custom Solution**] and click [**OK**].
   d. Click the [**Add Devices**] button.
   e. Search for the device family (i.e. Wonder Gecko) and select the Starter Kit associated with that family (i.e. EFM32 Wonder Gecko Starter Kit). Click [**OK**].
2. In the [**Getting Started**] area, select an SDK if the preferred SDK is not specified by clicking the [**Change Preferred SDK**] link. The examples are available in v4.4.1 or later.
3. Under [**Demos**], click [**View All**].
4. Select the cslib example and continue through the wizard to open a project based on this example.

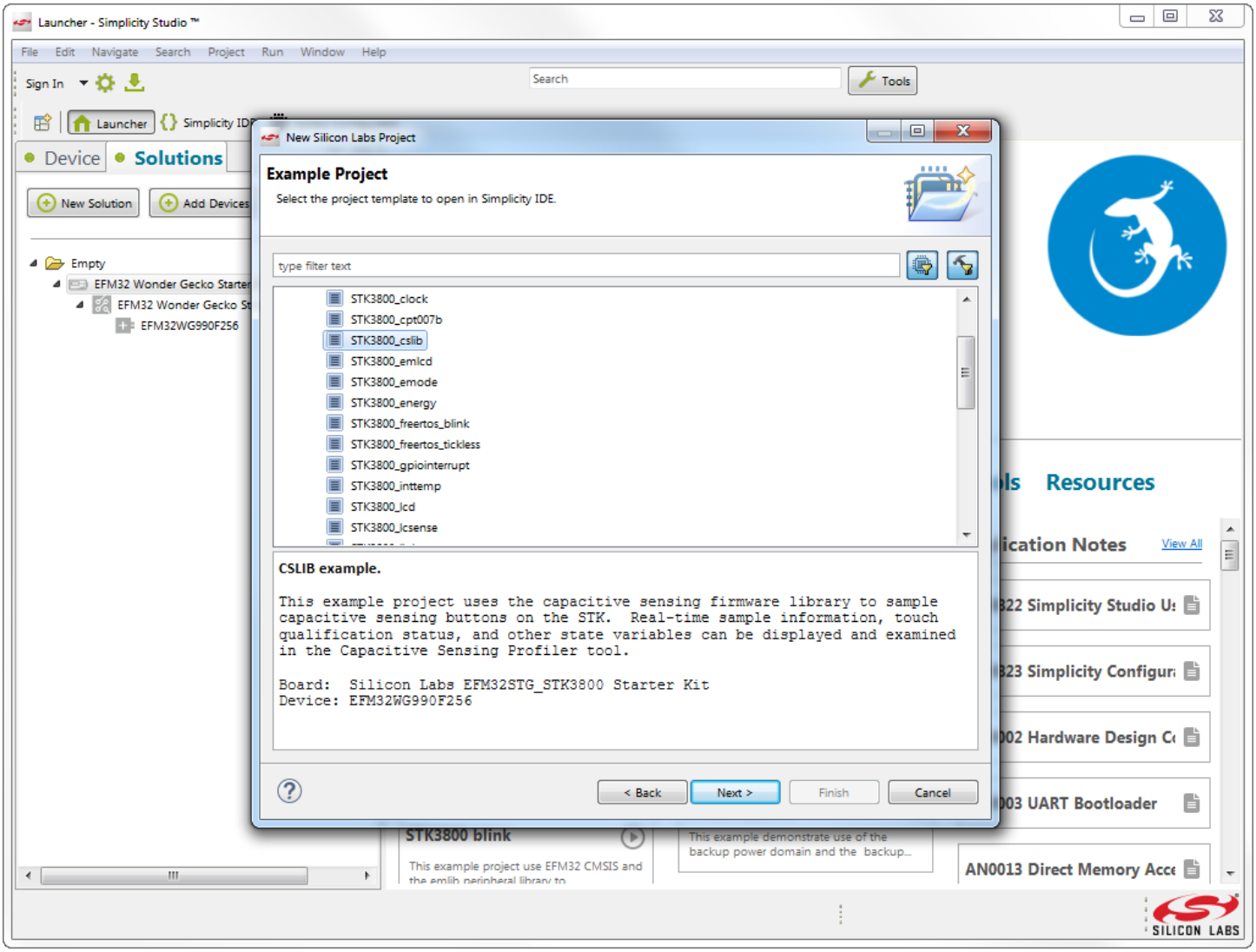


Figure 4.1. Capacitive Sense Library Examples for EFM32

5. Software Example

This application note comes with a software example for the EFM32 Tiny Gecko Starter Kit (EFM32TG-STK3300) and the EFM32 Giant Gecko Starter Kit (EFM32GG-STK3700). The example uses the capacitive touch slider on the lower right corner of the starter kit. By touching one of the slider elements, the user LED is lit during the touch event and turns off again upon no touch.

5.1 Working Principle

The software example simplifies the configuration of LESENSE for capacitive touch. LESENSE is configured to scan the enabled buttons with a certain scan-frequency and wake up if a button has a count value below a certain threshold. The touch threshold for each button is specified in percent of the average count for an untouched button as described in 3.2.2 Adjustable Settings.

The example uses an RTC interrupt to run a calibration routine. The interval between calibrations can be adjusted for different usage scenarios. The calibration routine tries to find the count values for the untouched buttons. It then reconfigures the wake-up threshold values for LESENSE based on the percent-values configured at initialization.

5.2 Calibration Routine

The calibration routine is executed at fixed intervals. It works by storing count values for each pad from the N previous calibration runs, it then takes the maximum count value seen in the previous N calibration events as the new untouched level.

Since a touch event is always characterized by a decrease in the count value, the system is allowed to quickly calibrate the threshold count upwards, but it should take longer to change the threshold down. With this approach it only takes one calibration event to change the threshold upwards, but it takes N calibration events to change the threshold down. Rising voltage and falling temperature will both contribute to an increase in the nominal count values, this will be corrected after one calibration event. Falling voltage and rising temperature will result in decreasing nominal count values, this will be corrected after N calibration events.

The calibration routine should be run frequently enough to catch small changes in the untouched count level before it changes enough to trigger a false touch. To save power it should not be run too frequently either.

The idea is that during the N last calibration events, there should at least be one calibration event that is executed when the user does not touch any buttons. The count value from that event will be the maximum count value encountered. In other words, the calibration count N multiplied by the interval between calibrations should be higher than the longest expected touch event. If a longer touch event occurs, it will only take one more calibration event to get back to the correct threshold value.

In the included software example, both the calibration frequency and N can be adjusted. The default value is 5 seconds between each calibration and N = 10. This will contribute less than 100nA to the average current consumption.

5.3 Initialization

To start the capacitive touch operation, an array containing the trigger threshold in percent for each of the LESENSE channels is declared and initialized. This is used by the calibration routine to configure the correct thresholds later on.

The calibration routine interval can also be configured depending on how fast supply voltage and other environmental factors are expected to change. Once every 10 seconds with N = 5 to 10 are reasonable values, supply voltage or temperature does not change significantly during a couple of minutes. If the application is moved between different environments, and thereby must withstand quicker changes in environment, the calibration routine can be run more frequently, every second for example.

At start up the calibration routine is executed N times in fast succession, during this period the buttons should not be touched for the calibration to work correctly. If a pad is touched during this initialization it will only take one calibration event after the touch event is over to correct the threshold value.

5.4 Touch Event

If a count value below the threshold for any channel is observed, LESENSE will trigger an interrupt. The interrupt routine stores and clears the interrupt flag, then it finds the channel that triggered the interrupt. The interrupt routine then reconfigures the scan frequency and checks the particular channel again a number of times (#define VALIDATE_CNT). This ensures that an actual touch event has occurred.

If all the validation measurements are positive, the evaluation criterion on that particular channel is inverted, this will result in a new interrupt on that particular channel when the touch event is over. By inverting the touch criterion in LESENSE, a minimum amount of cpu time is spent keeping track of touched and untouched buttons, LESENSE keeps track on its own.

Touch hysteresis is also implemented by changing the threshold value when the touch criterion is inverted. Touch hysteresis prevents the system from oscillating between touch and no touch events.

5.5 Further Improvements

The software example implements some of the basics for a robust touch application. Depending on the application, improvements can still be made. Some ideas are mentioned in the following section.

By making use of the LESENSE decoder, LESENSE can issue an interrupt for a touch only after a certain number of sequential count values are below the touch threshold. This can reduce the energy consumption further.

Implementing use of DMA to transfer count values to memory and do more advanced filtering and processing can also be beneficial. This can be used to differentiate between a water splash or similar disturbance and actual touch events. The collected samples can also be used to continually adjust the compromise between energy consumption and sensitivity.

A reference pin can be implemented with an unused analog comparator pin. The reference pin should be left unconnected, a users finger should not be able to affect it. With a reference pin, environmental changes like temperature and supply voltage can be tracked more accurately and compensated by adjusting the threshold for the other capacitive touch pins. With this approach, the required assumption of untouched buttons in at least one of the N calibration events could be relaxed.

6. Revision History

6.1 Revision 2.10

2016-10-14

Updated formatting.

Minor text edits for clarification.

Added 4. Capacitive Sense Library.

6.2 Revision 2.05

2014-05-07

Updated example code to CMSIS 3.20.5

Changed to Silicon Labs license on code examples

Added project files for Simplicity IDE

Removed makefiles for Sourcery CodeBench Lite

6.3 Revision 2.04

2013-10-14

New cover layout

6.4 Revision 2.03

2013-05-08

Added software projects for ARM-GCC and Atollic TrueStudio.

6.5 Revision 2.02

2012-11-12

Adapted software projects to new kit-driver and bsp structure.

Added software support for the Giant Gecko STK.

6.6 Revision 2.01

2012-04-20

Adapted software projects to new peripheral library naming and CMSIS_V3.

Modified ACMP initialization to not use the low power reference. With LESENSE it is automatically not used, but now the ACMP config matches what actually happens.

6.7 Revision 2.00

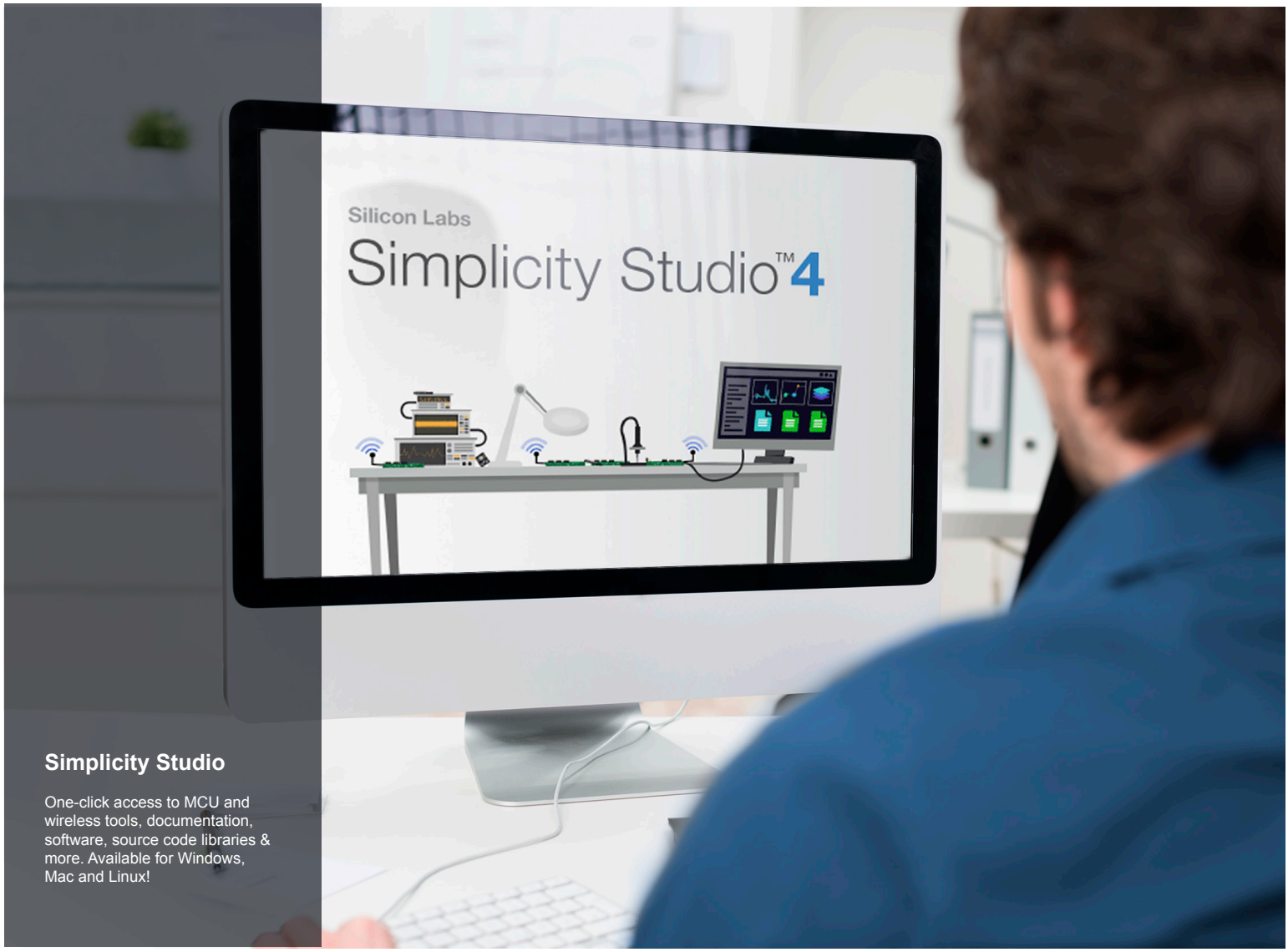
2012-03-07

Substantial update of both document and software example.

6.8 Revision 1.00

2011-05-26

Initial revision.



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS®

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com