Smart Capacitive Touch and Gesture Detection in sub-µA Deep Sleep Mode

Introduction

With the Low Energy Sensor Interface (LESENSE), the EFM32 microcontrollers are able to monitor a wide variety of sensors while continuously staying in energy saving deep sleep mode. Applications in need of wake on capacitive touch, sensing metallic objects or monitoring resistive sensors can be implemented with total consumption below 1.2 µA.

The sensors that LESENSE can interface range from simple thermistors, inductive or capacitive devices, to more complex analog sensors. In many applications such sensors are used to give input from the outside world to a microcontroller. As energy consumption and battery life become increasingly important, the microcontroller needs to be able to monitor these sensors by using as little energy as possible. The best way of doing this is staying in deep sleep as much as possible. This document will present three different measurement principles, capacitive, inductive and resistive, that can be implemented with the low energy sensor interface. References to the different application notes are also included.

The Challenge

Staying asleep for as high percentage of the time as possible and do the necessary measurements quickly and effectively, without energizing unnecessary circuitry, is the main challenge for a low power sensor application. Until now applications typically need to wake up the CPU to do sensor measurements (Figure 1). For a small microcontroller based system the CPU, when active normally consumes a lot of energy compared to the rest of the system. This means that reducing the amount of time the CPU is active, is one of the best ways to decrease energy consumption. Still, sensor measurements often require special excitation and sampling patterns. Wake up conditions may be complex and the required sampling rate might be high.

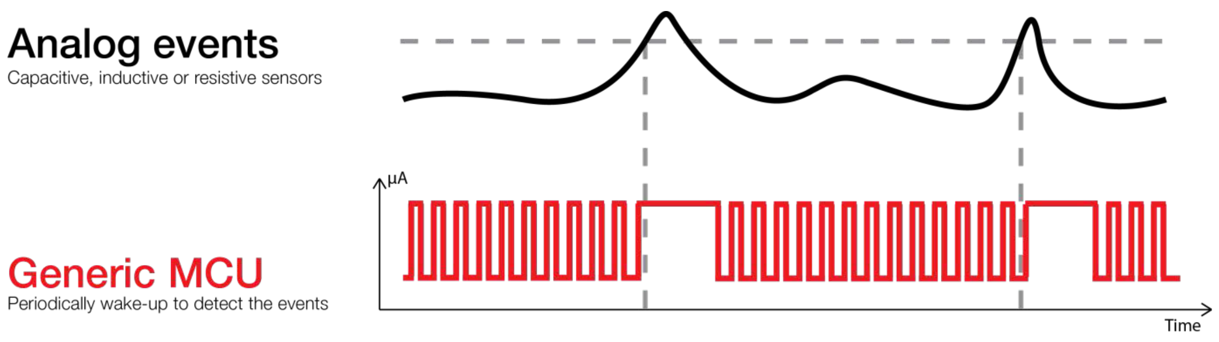
Analog events
Capacitive, inductive or resistive sensors
µA
Generic MCU
Periodically wake-up to detect the events
Time

Figure 1: High energy consumption with CPU active during every measurement.

Solution

A sensor measurement system that can operate autonomously even in deep sleep modes is the ideal solution. The autonomous system must be configurable for different external components and sensors. Different measurement principles should be easy to accommodate, and the system should incorporate energy friendly calibration features. How can all of this be done autonomously, saving precious energy for more complex tasks that really needs the energy hungry CPU? The answer: LESENSE.

LESENSE achieves this with a highly configurable sequencer and decoder that can also incorporate several of the ultra energy efficient peripherals in the EFM32. Efficient duty cycling of these resources means that a minimum of energy is used to monitor a sensor, (Figure 2). Further, the devices can be set to wake up only if certain conditions are met. This can be as simple as a finger touching a capacitive button, or it can be more complex transitions in the decoder, filtering sensor values before a wake up signal is issued. Either way, the Low Energy Sensor Interface will give the microcontroller a sense of the outside world without leaving its deepest beauty sleep.

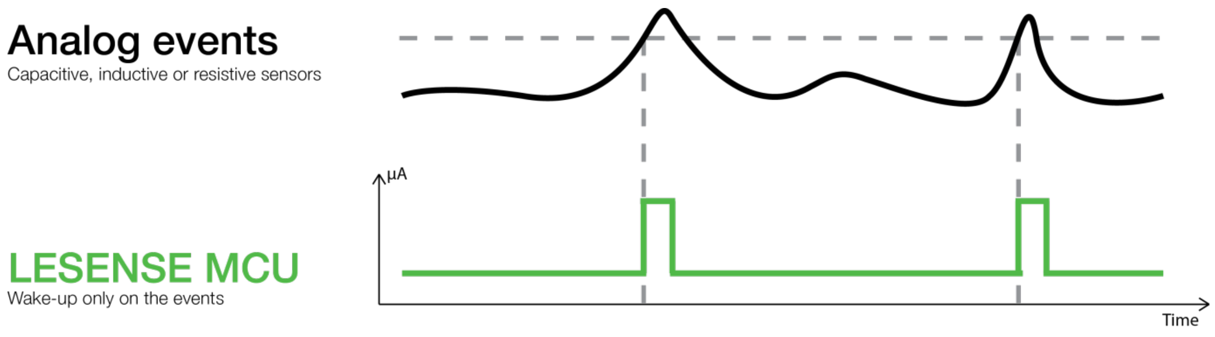
Analog events
Capacitive, inductive or resistive sensors
µA
LESENSE MCU
Wake-up only on the events
Time

Figure 2: Reduced energy consumption with LESENSE, waking up the CPU only when necessary.

More specifically the Low Energy Sensor Interface consists of analog comparators and a DAC (Digital to Analog Converter) which can be controlled by a very low power sequencer-module. The sequencer is able to operate in sub-µA modes running from a 32 kHz clock source. In addition to the sequencer, the comparator outputs can be counted, compared or passed on directly as interrupts. For accurate measurements, the digital-to-analog converter (DAC) can be selected as comparator reference and excitation source.

The sequencer controls which pins are connected to the comparator, how long the comparator is active and when the comparator output should be passed on to be counted or compared. Excitation with a DAC voltage or GPIO-pins can also be executed before or while the comparator is active. After a measurement, the counter or comparator output is buffered and stored for later processing. Interrupts or signals can also be passed on directly (through the peripheral reflex system) when the counter or output crosses pre-configured thresholds.

After a scan is finished, the results can be passed on to the low power decoder which is part of LESENSE. Each state in the decoder has configurable next states and trigger conditions. This makes it possible to interpret several sensor-readings and only wake up the CPU if they match a pattern over time (Figure 3).

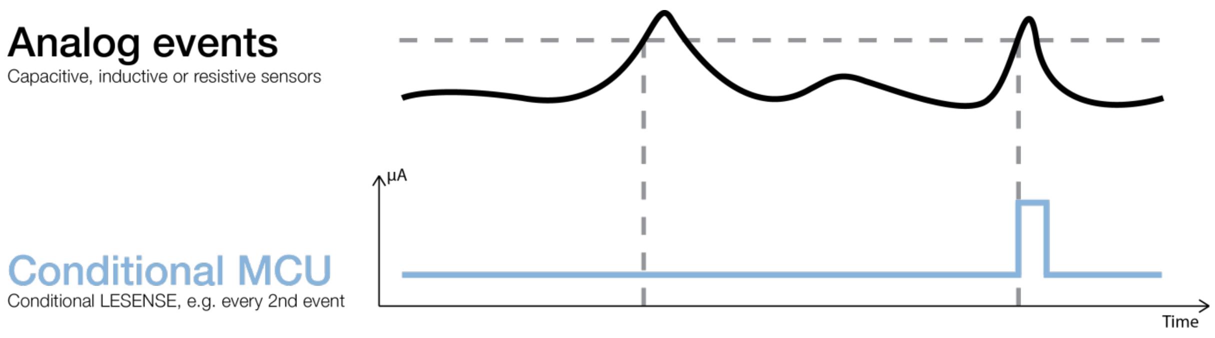

Figure 3: Conditional wake up on analog events.

Since the sensor results are available through the PRS (Peripheral Reflex System), they can be passed on to other low energy peripherals like serial ports or counters. The pulse counter, for instance, is able to take in PRS-signals and count them in quadrature counter mode. This enables the system to only wake up after a certain number of rotations. Think of a water-meter with a rotating vane, the rotations can be measured with LESENSE and counted with the quadrature counter. After, for example 10 rotations the CPU can be woken up to update the display and usage statistics. With a traditional MCU, the CPU would be needed for all the sequencing and control of the comparator. With the EFM32 all of this can be handled by LESENSE, allowing the chip to stay in deep sleep.

Since many applications must handle varying environmental conditions, continuous or periodic calibration of sensors-thresholds and measurement-timing is often critical for the performance. Since LESENSE stores several results in memory, the CPU does not need to wake up all the time to capture measurement results. These results can be used for periodic calibration and updates of trigger thresholds and measurement-timing.

Capacitive Sensor Example

Capacitive sensing is a technology based on measuring a change in capacitance which can be used in many different types of sensors. The most common application area is within Human Interface Devices (HID) like control panels and remote controls.

The Low Energy Sensor interface uses the analog comparators in the EFM32 to measure the capacitance between the sense-pin and the microcontrollers ground. Specifically it measures capacitance by including the sense-pin to ground capacitance in an RC-oscillator circuit. The frequency will change depending on the capacitance seen on the sense-pin. A touch sensor can simply be implemented by connecting the sense-pin directly to the touch pad area of the circuit board. No external components are needed.

This oscillating signal coming from the output of the comparator is then passed on to the LESENSE peripheral where each rising edge is used to increase a counter value, (Figure 4). LESENSE will wait for a configurable amount of time before capturing the counter value to the result buffer and clearing the counter to be ready to read the next sensor. The increased capacitance of a finger touching the sensor will result in a lower frequency and smaller count value. Once a counter value has been captured in the buffer, the value is compared against a threshold level which is individually configurable for each sensor. LESENSE then proceeds to wake up the EFM32 when the counter value is lower than the threshold. The sensor to be measured is controlled by the input switch of the analog comparators, which is also controlled by LESENSE.

In most cases, capacitive sensors are only touched for a very small percentage of the time compared to the overall idle periods. Hence, it is crucial to minimize the power consumption while the device is waiting to be touched. This can be achieved by only measuring the sensor periodically at longer intervals of several seconds. However, when a user actually touches the sensor, this strategy will lead to lagging and a less user friendly interface. As the capacitive sensing method implemented by the Low Energy Sensor Interface requires a minimum of resources, the sampling frequency can be increased without affecting performance, so that the responsiveness is improved while a user interacts with the device.

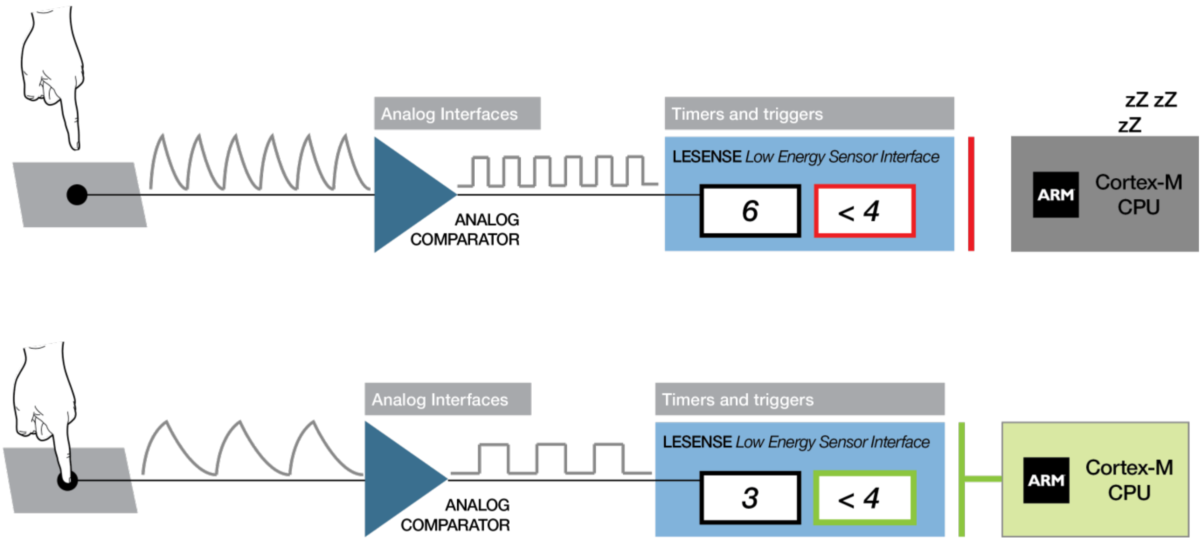

Figure 4: Capacitive sensor

Current consumption

The current consumption of capacitive sensing is influenced by a several factors. Sampling frequency and thickness of the overlay are the two main contributors to current consumption. The frequency at which the sampling occurs is proportional to the added current consumption, if one sample at twice the sampling rate, the dynamic current consumption will also be doubled. Regarding overlay thickness, thicker overlay results in higher current consumption. An accurate relation between increased current consumption and thickness of the overlay is not trivial to establish. For thicker overlays the difference between touch and no touch is smaller. To differentiate the two, LESENSE needs to keep the relaxation oscillator running longer. Exactly how much longer for a given overlay depends on the size of the pads, the dielectric material in the overlay and the actual PCB-design.

Empirical results have shown that for a 5mm acrylic overlay with sampling frequency at 5 Hz, the added consumption for each touch pad is approximately 500 nA. This results in approximately 3 µA total consumption for a 4 button touch application sampling at 5 Hz. The static consumption without sampling is less than 1 µA. For improved user experience the sampling speed could be increased to 10 Hz after the first touch event, resulting in 5 µA total consumption.

Conditional Wakeup

Since LESENSE also includes a fully functional and configurable decoder, many different conditional wake-up triggers can be implemented. The decoder can be utilized to only wake up from deep sleep when several sensor conditions are met at once. It can also track consecutive measurements to issue a wake up only after a special sequence of threshold crossings has occurred. This can for instance be that a temperature and humidity sensor both has reached their thresholds, or a pressure sensor must trigger 10 times in a row before a wake up event occurs.

Since the inputs to the decoder can be connected to the peripheral reflex system, almost any peripheral can interact with it. This makes it possible to logically combine several GPIO-pins to trigger a wake up, or use the decoder to decode serially transmitted data. The possibilities with a fully configurable decoder are truly end-less.

Application Example: Slide to Wake

By configuring the decoder in LESENSE to continuously evaluate up to four capacitive touch inputs, an application can implement gesture detection in addition to simple touch detection. An interrupt can be issued only when the correct gesture is detected across up to four sensors. A familiar application is the intuitive “slide to wake” functionality often found in smart phones. For a similar user experience with LESENSE, the decoder is configured to only issue an interrupt after the correct sequence of touches is detected from the touch pads. The added current consumption is negligible; this kind of gesture detection can be implemented without much overhead in current consumption. A four segment capacitive slider, waiting for a “slide to wake”-event, would only consume 3-4uA.

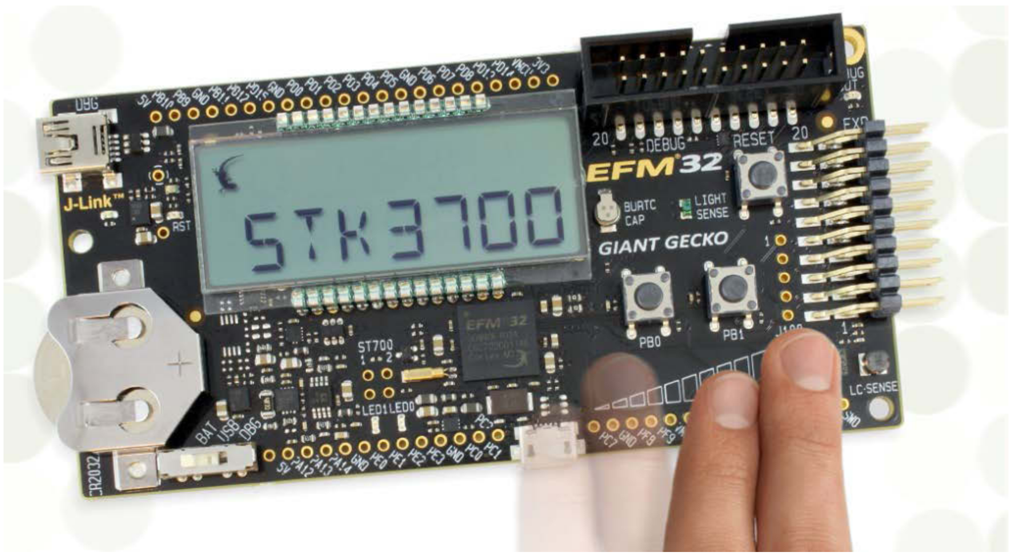
Figure 5: Slide to wake functionality.

Calibration

Sensor systems are implemented in a wide variety of environmental conditions and must be able to operate reliably while parameters such as temperature, humidity, supply voltage, permeability and conductivity change constantly. Often it is not easy to determine if the change in sensor readout is due to a change in the physical quality sought to be monitored or fluctuations in the climate surrounding the device. For capacitive sensors for instance, the increased permeability of humidity near a sensor could give the same increase in readings as a human finger. In such cases it is important to collect sensor data over time to correct for fluctuations. The Low Energy Sensor interface allows you to buffer a series of up to 16 sensor readout values. This allows the device to wake up infrequently and adjust the sensor thresholds according to the buffered data. The filtering schemes must then be adjusted according to the application and the expected rates of drift. For instance, the temperature in a living environment usually changes relatively slowly, whereas humidity on a capacitive touch sensor can appear instantaneously. The offset that the factors introduce must therefore be treated differently when filtering.

Summary

The Low Energy Sensor Interface (LESENSE) enables the EFM32 microcontroller to monitor many different kinds of analog sensors while staying in deep sleep mode. Running from the low frequency clock source, LESENSE can monitor up to 16 sensors in sub-µA sleep modes. Typical average current consumption is around 1.2 µA. Applications include any kind of capacitive, inductive or resistive sensing, rotation counting, GPIO-state decoding or similar. LESENSE also has a fully configurable decoder to decode sensor outputs. The decoder can evaluate sensor states and wake up the CPU when a special combination of sensor outputs or patterns over time occurs. By combining the sensor interface with the built in decoder, the possibilities are only limited by the imagination of the designer.

For further information about LESENSE, please refer to the following application notes:

- an0028 Low Energy Sensor Interface - Capacitive Sense
- an0029 Low Energy Sensor Interface - Inductive Sense
- an0036 Low Energy Sensor Interface - Resistive Sense
- an0040 Hardware Design for Capacitive Touch
- an0053 IR Sensor Monitoring Using LESENSE