EFM®32

... the world's most energy friendly microcontrollers

Low Energy Sensor Interface

TM0010 - Hands-on Exercise

0 1 2 3 4

Introduction

In this exercise we will open a few different application note code examples for the Low Energy Sensor Interface (LESENSE) and try to modify different parameters and see the effects.

This exercise includes:

- This PDF document

ENERGY® micro

www.energymicro.com

1 Introduction

1.1 How to use this document

This document contains different tasks with explanations that will help the reader understand how the Low Energy Sensor Interface works and how to get started using it. Two application notes in particular will be needed when following the exercises, an0028 Lesense Capacitive Sense and an0036 Lesense Resistive Sense.

Projects for each of the following IDEs are provided with the respective application notes.

- IAR Embedded Workbench
- Keil µVision
- Rowley CrossWorks
- Eclipse/CodeSourcery

Since the projects are slightly different for the various kits, make sure that you open the project that is prefixed with the name of the kit you are using. If you are using eclipse, please see Application Note AN0023 on how to set up this.

1.2 Prerequisites

It is assumed you already have done the tutorials or are familiar with writing and compiling code for the EFM32. You should know the basic principles of how to use the Energy Aware Profiler and how to debug software using the register viewer in your IDE.

The exercises require that you have an EFM32 starter kit with LESENSE at hand (STK3300, STK3600 or STK3700). Before you start working on this tutorial you should make sure you have installed an IDE that supports EFM32. See the following webpage for a list of supported IDEs:

http://www.energymicro.com/tools/third-party-ide-compiler

You also need to have installed Simplicity Studio on your computer and *at least* the following packages:

- Application Notes
  - AN0028 Low Energy Sensor Interface - Capacitive Sense
  - AN0036 Low Energy Sensor Interface - Resistive Sense
- Documentation
  - Device Family Reference Manual (e.g. EFM32GG)
  - Cortex-M3 Reference Manual
- Software
  - CMSIS
  - CMSIS Documentation (if working offline)
  - BSP Package for your kit(e.g. EFM32GG_STK3700)
- Third Party
  - JLink ARM Drivers

2 LESENSE Resistive Sense

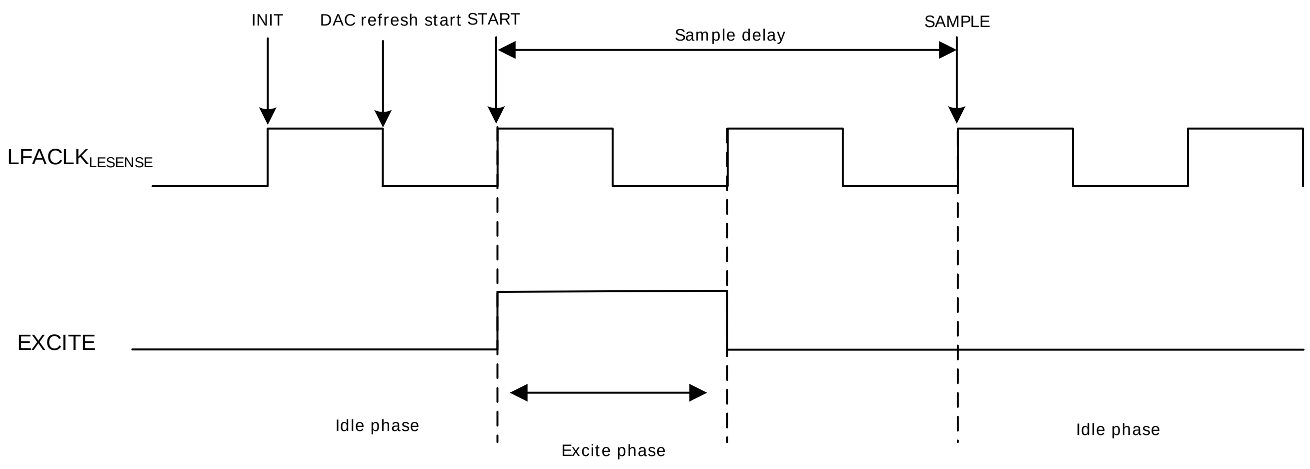
In this exercise you will play around with the lesense configuration in an example that wakes up on changes in ambient light level. Through out this hands-on document, in addition to the hands-on tasks, short questions are given that will help in understanding important terms and concepts. Answers are given at the end of this document.

Task 1: Basic principle of operation.

Open the application note AN0036 pdf document from simplicity studio. The light sensor on the starter kits is connected as described in the chapter *Two-pin Resistive Sensor Reading*. Read through the application note and pay extra attention to the relevant "two-pin resistive sensor" -chapter. After you have read the application note, have a look at the timing diagram in Figure 2.1 (p. 3) .

(a): First consider the two timing periods in the timing diagram, sample delay and excitation period. How would you change the excitation period relative to the SAMPLE instant in the timing diagram for the "two-pin resistive sensor" to work? Hint: The sensor should have power when sampling occurs.

Figure 2.1. Timing diagram



Open up the example project and compile the "Lightsense_single" example for your kit, unmodified. Upload the binary to your kit. Disconnect the debugger and have a look at the current consumption with the eAProfiler. Remember that you need to reset the EFM32 once after exiting the debug session to get true EM2 operation. Scroll down to the software example chapter in the application note, try to verify the current consumption claims.

(b): What is the average current consumption and what is the scan rate?

Have a look at the source code again, all code we will look at in this exercise is located in the *lesense_lightsense_single.c* file. Locate the define for scan frequency and change the scan frequency to 50 Hz for example. Notice how the change in scan frequency affects current consumption.

Task 2: Change comparator trigger level.

(a): In the code; find out where the analog comparator threshold is configured. Why is it not in the analog comparator setup function?

Consider the figure of the lightsensor circuit in the software example chapter in the an0036 PDF. The light sensitive transistor increases conductivity with increasing light levels.

(b): In which direction must the analog comparator threshold be adjusted to make it trigger at higher ambient light levels?

In the code project, change the analog comparator threshold value *.acmpThres* to test if your assumptions were correct.

Play around with the threshold value in the register viewer while debugging. The setting is located in the LESENSE_CHx_INTERACT register. You should be able to find the correct channel by looking at which channel has already been configured in the register viewer after the LESENSE init function. Put a breakpoint in the LESENSE interrupt routine to know when it triggers and when it does not.

Task 3: Change lesense timing parameters.

For this task, please set the comparator threshold to the minimum value so it never wakes up.

In the source code, locate the exitation and sample delay configuration.

(a): Which clock is these delay timing values derived from? Can you also figure out if there is a prescaler involved, what is it configured to?

Change the *lesenseClkLF* prescaler so it divides the clock by 4, and set the excitation time to its maximum value. Tip: Have a look in the em_lesense.h file or bitfield description in the reference manual to figure out the maximum value for the extime field.

Now have a look with the profiler and notice the current consumption when you partly cover the light sensor with your hand. You should be able to see that the current consumption during the excitation period is dependent on light level.

3 LESENSE Capacitive Touch

In this exercise you will play around with the lesense configuration for capacitive touch wake up.

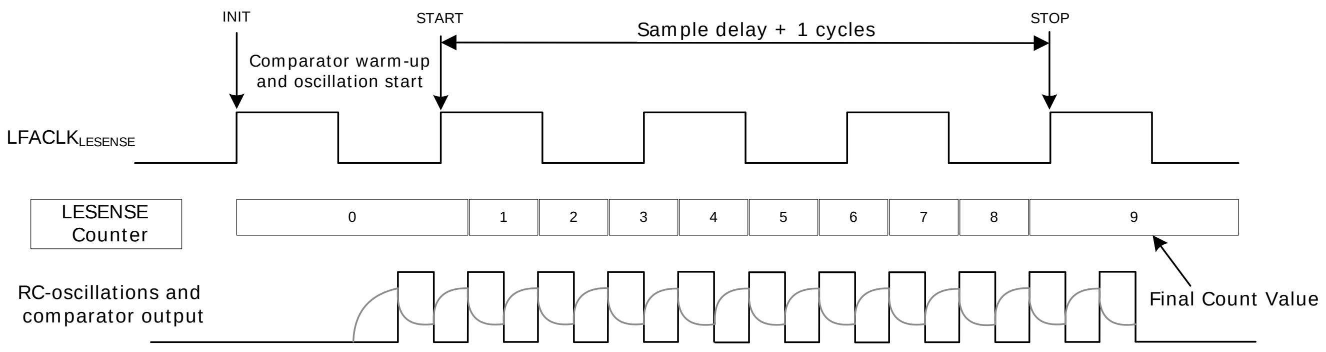
Task 1: Basic principle of operation.

Open the AN0028 pdf document from simplicity studio. Read through the theory chapter and try to understand the basic principle of how capacitive sensing works. Then have a look at the timing diagram in Figure 3.1 (p. 5) .

(a): How will the count value change when a finger comes close to the touch pad?

(b): How will significant change in frequency of the low frequency clock (LFACLK_lesense) affect the count value?

Figure 3.1. LESENSE Sequence for Capacitive Touch



Open up the example project and compile the "letouch_single" example for your kit, unmodified. Upload the binary to your kit. Disconnect the debugger and have a look at the current consumption with the eAProfiler. Remember that you need to reset the EFM32 once after exiting the debug session to get true EM2 operation. Scroll down to the "Current Consumption" section in the application note, try to verify the current consumption claims.

(c): What is the average current consumption and what is the scan rate? Remember that STK3700 with giant gecko will have a bit higher consumption than the STK3300.

Have a look at the source code again, all code we will look at in this exercise is located in the following files:

- *lesense_letouch.c*
- *lesense_letouch_main.c*
- *lesense_letouch_config.h*

Locate the define for scan frequency in *lesense_letouch_config.h* and change the scan frequency to 20 Hz for example. Notice how the change in scan frequency affects current consumption.

You might also notice a spike in current consumption every 5 seconds, that is the touch calibration routine executing. It executes in an RTC interrupt in the code.

Task 2: Change sample delay.

First modify the example to only scan one button. This is done by setting the sensitivity to zero for all but one channel as described in the an0028 application note.

(a): Change the *SAMPLE_DELAY* define in *lesense_letouch_config.h* to a higher number, for example 100. Compile, then start a debug session. Open the register viewer and locate the result buffer registers for LESENSE (LESENSE_BUFx_DATA). There should be 16 values in this result buffer. What seems to be the count value stored here?

If more than one lesense channel is enabled, this result buffer will contain values for each different channel in sequential order.

Now, still in the debug session, try to change the sample delay for the channel that is active. The sample delay is located in the LESENSE_CHx_TIMING register. You can locate which channel looking at the different channel config registers and find the one with different values than the rest (already configured channel). Notice how the count results changes when changing the sample delay. Try to put a finger on the pad corresponding to the active channel, notice how the count value changes.

(b): For a high sample delay, ~100; What seems to be the ratio between count values for a touch versus an untouched pad? Try to put some plastic between the pad and the touching finger. How does the ratio change?

Task 3: Change sensitivity setting.

For this task, please leave the sample delay at 100.

Notice how sensitive the capacitive touch is with the default 1.0 sensitivity setting. The 1.0 setting is appropriate for approximately 2-3mm of plastic between the touch pad and the touching finger. For a bare pcb without overlay, a lower sensitivity is needed. The numbers in this array represents the percentage of change needed in count value before a pad is considered to be touched by a finger. 1.0 means 1 percent change in count value. The values will typically be in the range 0.5 to 50.

Try to change the sensitivity down (increasing the percentage number) until the EFM32 wakes up when the finger touches the pcb, not before.

Locate the *compthres* value in the register viewer (LESENSE_CHx_EVAL register) and notice how it changes when changing the sensitivity threshold.

4 Hands-on Solution

In this section you will find answers to the questions given in the hands-on training material. Please try to find the answers yourself before looking at the answers given here.

LESENSE Resistive Sense

Task 1: Basic principle of operation.

(a): The excitation period should be the same length as the sample delay. This ensures that the resistive divider is powered while sampling. It is OK that the excitation ends at the same instant the sampling happens because there will always be a small but finite delay through the analog comparator.

(b): The consumption should be approximately 2 uA, remember that tiny gecko and giant gecko has a bit different consumption. Also remember that AEM accuracy can vary, +- 0.5uA is reasonable deviation. The scan frequency should be approximately 20 Hz. Since this is running from the LFRCO, the frequency can vary with some Hz.

Task 2: Change comparator trigger level.

(a): The comparator threshold for the analog comparator is configured within the LESENSE module. Specifically the .acmpThres in the channelIinit struct. This enables LESENSE to use different thresholds for different channels, even though the different channels use the same analog comparator. This value maps directly to the LESENSE_CHx_INTERACT_ACMPTHRES register bit field.

(b): Since the light sensitive transistor is at the top of the resistive divider and its resistance decreases with higher ambient light levels, the output voltage will increase for increasing light levels. By adjusting the trigger threshold up, the analog comparator triggers at higher ambient light levels, making the sensor more sensitive.

Task 3: Change lesense timing parameters.

(a): The delay timing is derived from the internal timer in LESENSE, it can either be clocked from the LFCLK or AUXHFRCO clk. It is prescaled by the LESENSE_TIMCTRL_LFPRESC or LESENSE_TIMCTRL_AUXPRESC bitfields respectively.

LESENSE Capacitive Touch

Task 1: Basic principle of operation.

(a): A finger touching the pad will cause the RC-oscillations to run slower, resulting in a decreased count value.

(b): Change in the LFclk frequency will change the timebase for lesense which will cause the count value to change because the sample delay time-window changes. For the highest sensitivity the LFclk should be derived from a crystal oscillator to keep the frequency stable enough to catch even the smallest changes in RC-oscillation frequency.

(c): Around 3.5 - 4.5 uA depending on the kit type.

Task 2: Change sample delay.

(a): Should be count values around 2000 give or take a couple of 100.

(b): Depending on how close the finger is to the pad, the count value can change as much as 80%.

5 Revision History

5.1 Revision 1.00

insert_date

Initial revision.

A Disclaimer and Trademarks

A.1 Disclaimer

Energy Micro AS intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Energy Micro products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Energy Micro reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Energy Micro shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products must not be used within any Life Support System without the specific written consent of Energy Micro. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Energy Micro products are generally not intended for military applications. Energy Micro products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

A.2 Trademark Information

Energy Micro, EFM32, EFR, logo and combinations thereof, and others are the registered trademarks or trademarks of Energy Micro AS. ARM, CORTEX, THUMB are the registered trademarks of ARM Limited. Other terms and product names may be trademarks of others.

B Contact Information

B.1 Energy Micro Corporate Headquarters

Postal Address	Visitor Address	Technical Support
Energy Micro AS P.O. Box 4633 Nydalen N-0405 Oslo NORWAY	Energy Micro AS Sandakerveien 118 N-0484 Oslo NORWAY	support.energymicro.com Phone: +47 40 10 03 01

www.energymicro.com
Phone: +47 23 00 98 00
Fax: + 47 23 00 98 01

B.2 Global Contacts

Visit **www.energymicro.com** for information on global distributors and representatives or contact **sales@energymicro.com** for additional information.

Americas	Europe, Middle East and Africa	Asia and Pacific
www.energymicro.com/americas	www.energymicro.com/emea	www.energymicro.com/asia

Table of Contents

List of Figures

ENERGY micro

Energy Micro AS
Sandakerveien 118
P.O. Box 4633 Nydalen
N-0405 Oslo
Norway

www.energymicro.com