EFM32
... the world's most energy friendly microcontrollers

IR Sensor Monitoring Using LESENSE

AN0053 - Application Note

Introduction

This application note covers the basic theory of monitoring infrared (IR) photo sensors with EFM32 microcontrollers. It describes how to set up the Low Energy Sensor Interface (LESENSE) and create a circuit both for photo interruption and proximity detection while remaining in EM2, achieving current consumption around 5 µA.

The software examples, which is made for the EFM32 Giant Gecko Starter Kit (EFM32GG-STK3700), shows how to configure LESENSE to implement the photointerrupter or proximity sensor.

This application note includes:

- This PDF document
- Source files (zip)
  - Example C-code
  - Multiple IDE projects

ARM ZERO TINY GECKO LEOPARD GIANT WONDER

SILICON LABS

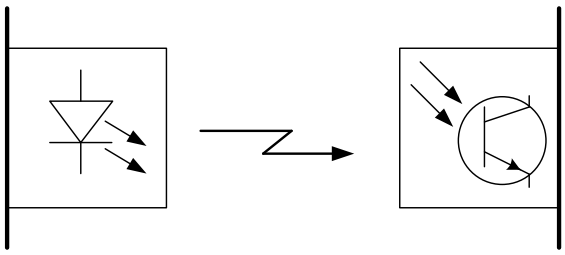
1 Introduction

This application note focuses on the configuration of LESENSE and the analog comparators to excite the IR emitter in a given pattern and to check if the same pattern is received by LESENSE, waking up the processor if a match is detected. Applications include rotary counters, limit switches, pellet dispensing, etc.

1.1 IR Photointerrupter

An IR photointerrupter is composed of an IR emitter and an IR detector facing each other, see Figure 1.1 (p. 2) . By emitting a beam of IR light the sensor can detect when an object passes inbetween the emitter and detector, breaking the beam.

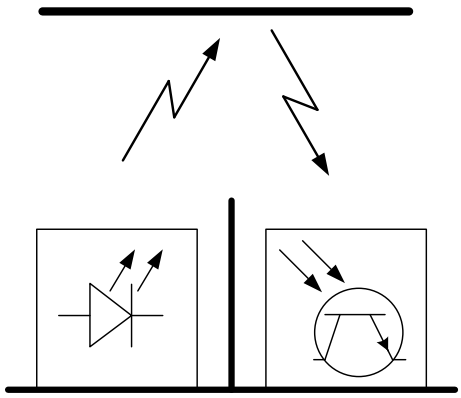
Figure 1.1. Photointerrupter



1.2 IR Proximity Sensor

The IR proximity sensor have the IR emitter and detector facing the same way, separated by a barrier blocking the IR light from reaching the detector directly. When an object is placed in front of the sensor some of the IR light will be reflected off the object, hitting the detector. See Figure 1.2 (p. 2) .

Figure 1.2. Proximity sensor



The amount of IR light reflected off an object into the detector is a function of distance from the detector, the surface of the object and the emitting strength of the IR sender.

1.3 Signal Measurement

Since some IR energy is present at all times, from lighting, as heat from surfaces, from the sun or from other electronics equipment, the sensors must be able to differentiate between ambient IR and IR from the emitter. The proximity sensor is more sensitive to ambient conditions because the signal reflecting

off an object loses most of its energy. There are several ways to make the sensor more robust to false detections.

One option is modulating the IR emitter with a given frequency, then on the detector side filter out everything not carrying that frequency. This gives a strong noise immunity, but requires dedicated hardware.

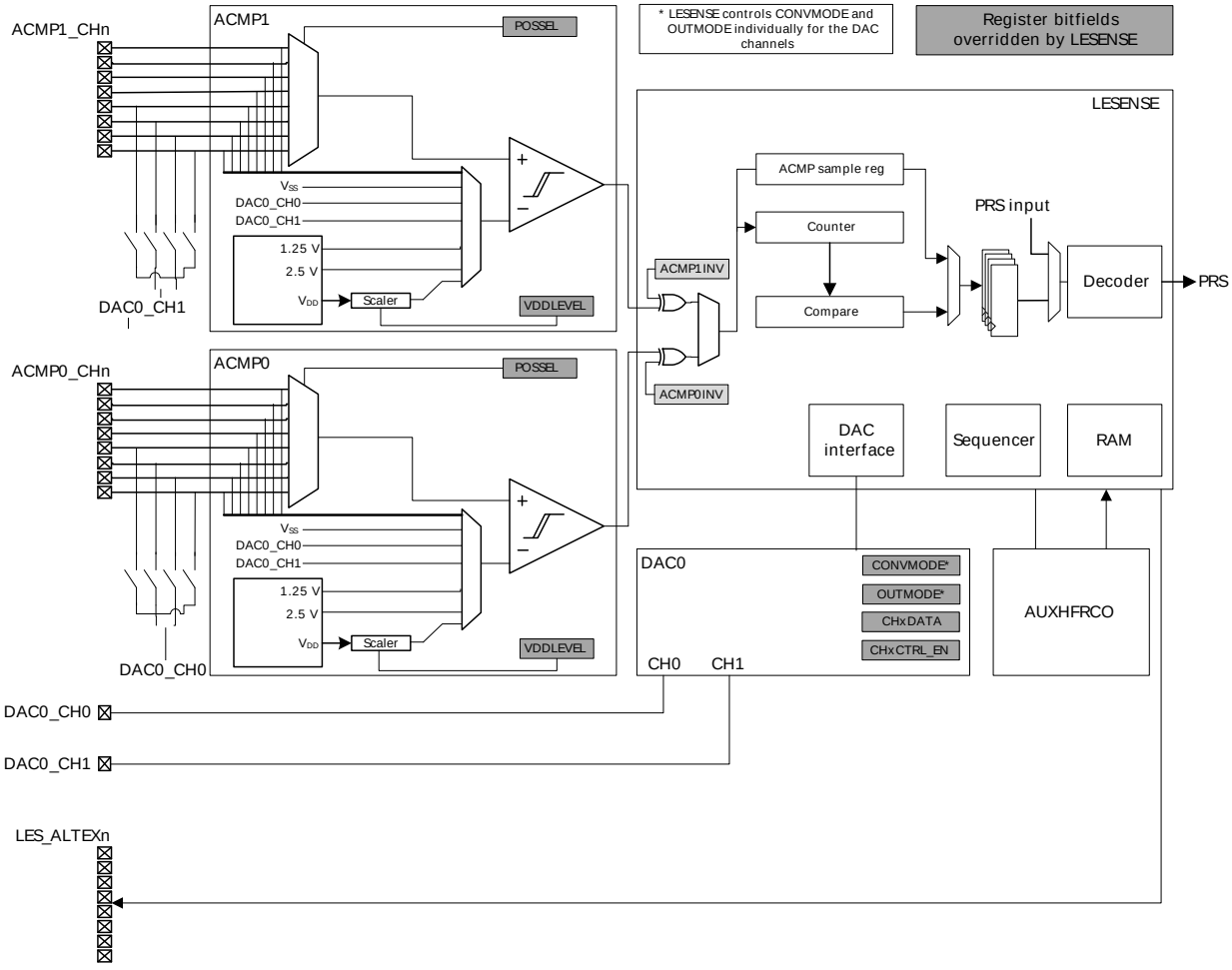
Another solution is to encode the emitted IR signal by toggling the emitter on/off and see if the same code is measured at the detector. A simple coding scheme would be to set the IR emitter low for one period and high for the second. Only if the same signal is received is the sensor triggered. This can be achieved using LESENSE. This method is dependent on calibrating the sensor often enough to counter shifting ambient conditions, for example someone turning on a light in the room.

1.4 LESENSE

The Low Energy Sensor Interface (LESENSE) is a peripheral state machine which 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. Figure 1.3 (p. 3) 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 1.3. LESENSE overview



For a more thorough discussion on how to setup and use LESENSE, see application note AN0036 Low Energy Sensor Interface - Resistive Sense.

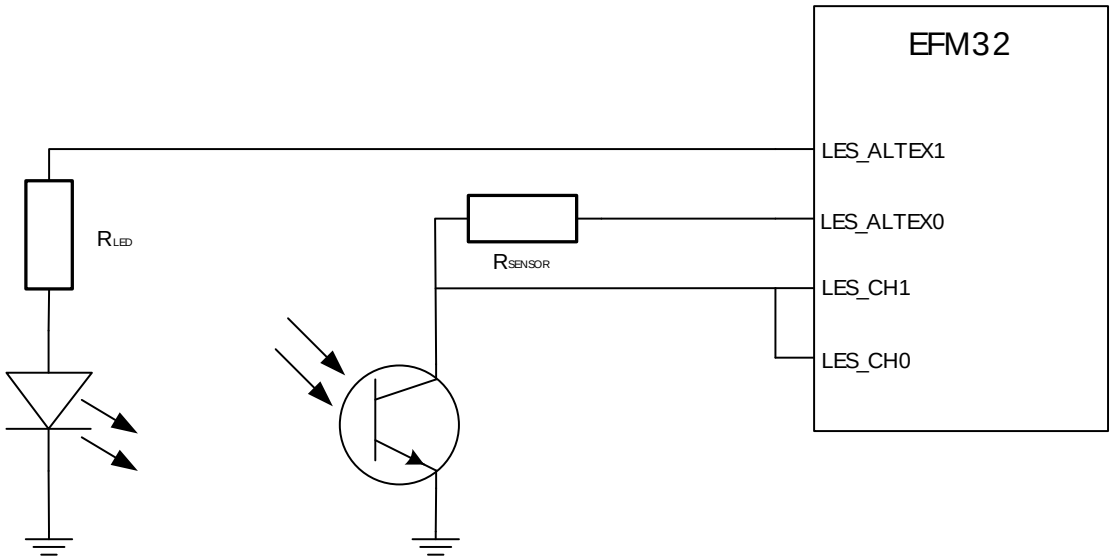
2 Circuit

The circuit consists of an IR emitter and detector. The emitter is an IR LED and the detector is a phototransistor connected in a common-emitter configuration. The IR LED and phototransistor can be bought as a single package, both in photointerrupter and proximity sensor configuration.

2.1 Schematic

The detector and IR LED get their power from the alternative excitation channel 0 and channel 1 pin, respectively. Sensor measurements are done on the phototransistors collector. In this case it is connected to LESENSE CH0 and CH1. See Figure 2.1 (p. 5) for schematic. Normally, each LESENSE channel control a seperate sensor. Since both channels here use the same sensor, LES_CH0 and LES_CH1 are connected to the same pin.

Figure 2.1. Circuit schematic



2.2 Component Values

To get the best performance, use a phototransistor with a peak sensitivity wave length that matches the transmitter IR LED wave length, 940 nm is a common value.

Choosing resistor value for R_{LED} is choosing a balance between sensor range and energy consumption. The proximity sensor requires more current to work reliably than the photointerrupter. The GPIO pins on EFM32 can source/sink up to 20 mA. If increased range is needed, the IR LED pin can be used for controlling a transistor giving the possibility of increasing the drive strength of the IR LED.

The common-emitter amplifier circuit gain is controlled by the value of R_{SENSOR}. Higher value of R_{SENSOR} give higher gain. The gain should be set high enough so that the phototransistor goes into saturation when it is exposed to IR from the IR LED. The final value should be chosen after experimenting with what best fit the application.

3 Working Example

This application note comes with two software examples for the EFM32 Giant Gecko Starter Kit (EFM32GG-STK3700). One for each sensor configuration, photointerrupter and proximity sensor. The examples use the external circuit described in Chapter 2 (p. 5) . When an object passes in front of the sensor the LESENSE interrupt will increment a counter and write the value to the LCD segment display.

3.1 Circuit BOM

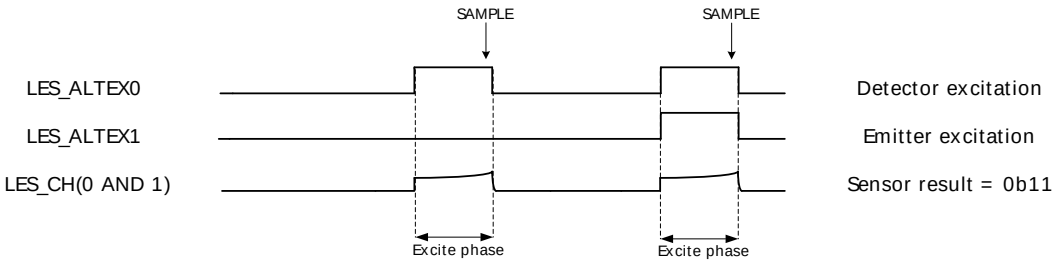
The proximity sensor software has been tested using the following components, giving it a range of 20 mm:

- R_{SENSOR} = 46 000 ohm
- R_{LED} = 50 ohm
- Reflective Optical Sensor with Transistor Output - TCRT5000
- EFM32GG-STK3700

3.2 Working Principle

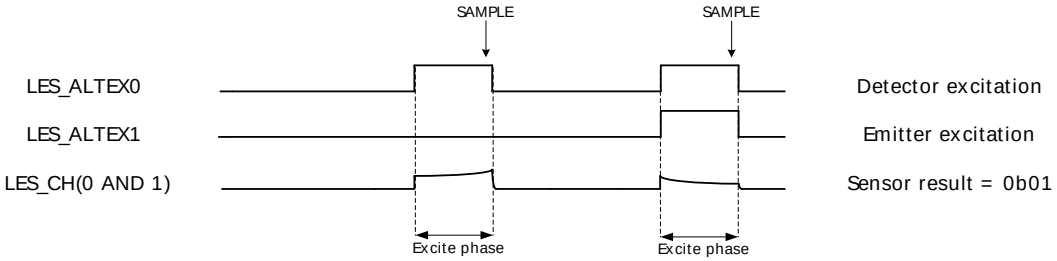
The software examples use LESENSE to control the circuit allowing the EFM32GG to stay in EM2, only waking up the CPU when an object is detected. LESENSE uses two of its channels to excite the detector and emitter on its alternative excitation pins. Measurement is done on the input pins on LES_CH(0 and 1). The detector is excited for both measurements while the emitter is excited only for the second measurement. If the two measurements both return 1, no IR light is hitting the detector. See Figure 3.1 (p. 6) for sensor timing of a single scan where no IR is hitting the detector.

Figure 3.1. Signal timing plot, no detection



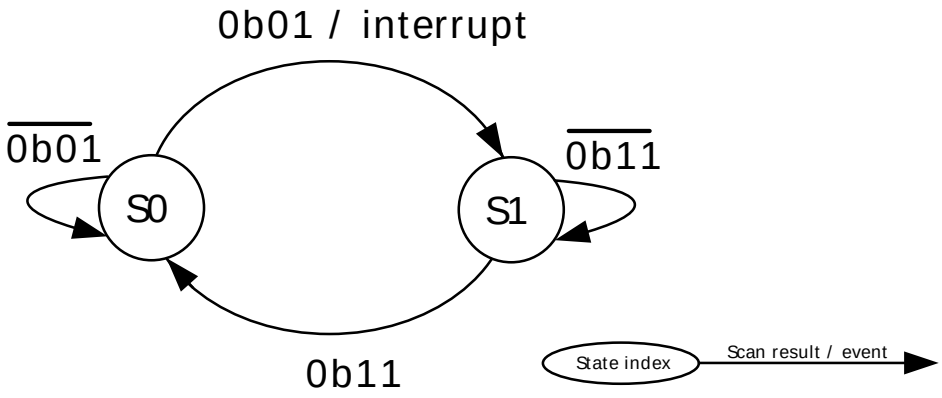
If the first measurement returns 1 and the second 0, IR light from the emitter is hitting the detector. See Figure 3.2 (p. 7) for sensor timing, now with IR hitting the sensor. The detector measurements are done with the LESENSE controlled analog comparator (ACMP). With the detector on the positive input and the negative input set by the calibration function.

Figure 3.2. Signal timing plot, with detection of object



The decoder in LESENSE is used to wake up the CPU when an object is detected. The decoder is implemented as a programmable state machine. Here two of the possible sixteen states are used to implement the photo sensor. After each scan the sensor result of the two measurements are stored in the LESENSE_SCANRES register. The result is compared against the predefined trigger values in the decoder, if a match is detected the state machine transition to the next state. See Figure 3.3 (p. 7) for the proximity sensor state machine.

Figure 3.3. Proximity sensor state machine



3.3 Configuration

Configuration of the sensor depends on the application and which type of sensor is used, photointerrupter or proximity sensor. The behaviour of the sensor can be changed by altering the LESENSE state machine or altering the following defines.

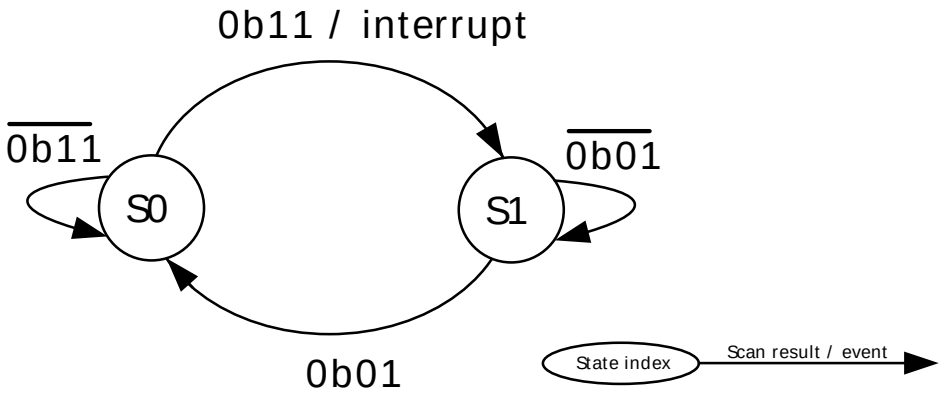
```
/* How many times per second the sensor scans for objects.
   Energy consumption increases linearly with frequency. */
#define OBJECT_SCAN_FREQ          4
/* How often to calibrate the analog comparator threshold value */
#define SENSOR_CALIB_PERIOD_SEC   5
/* How many levels below the actual threshold should ACOMPTHRES be set to */
#define SENSOR_SENSITIVITY        2
```

Timing of each measurement is controlled by the LESENSE_CHx_TIMING register. Excitation time is controlled by EXTIME in LESENSE_CHx_INTERACT, it must be long enough to give the phototransistor time to stabilize. Signal timing for a single scan is shown in Figure 3.1 (p. 6) .

3.3.1 Photointerrupter

The photointerrupter is made to detect when an object blocks the IR beam from hitting the detector. See Figure 3.4 (p. 8) for LESENSE state machine configuration. Since the IR emitter is placed facing the IR detector the difference on the detector output between object blocking and no object is much greater than with the proximity sensor configuration. This can in some cases remove the need for calibrating the sensor more than once. `SENSOR_SENSITIVITY` can also be set higher than with the proximity sensor.

Figure 3.4. Photointerrupter state machine



3.3.2 Proximity Sensor

The proximity sensor is made to detect when an object reflects the IR beam back to the detector. See Figure 3.3 (p. 7) for LESENSE state machine configuration. It is more sensetive to ambient IR and must be calibrated often enough to counter shifting conditions.

3.4 Calibration

The detector measurements are done with the analog comparator (ACMP). With the detector on the positive input and the negative input controlled by ACMPTHRES in LESENSE_CHx_INTERACT. See equation Equation 3.1 (p. 8) .

Negative input equation

$$\text{Negative input} = V_{DD} \times \text{ACMPTHRES}/63 \quad (3.1)$$

The value in ACMPTHRES is controlled by the calibration function `sensorCalib()`. `SENSOR_CALIB_PERIOD_SEC` is how often it should run, when set to 0 the calibration function will only run once. It takes the previously written value in ACMPTHRES for CH0 and CH1 and increments or decrements the value until the threshold is found. `SENSOR_SENSITIVITY` is how many levels below the actual threshold ACOMPTHRES is set to. The value can be set higher for the photointerrupter configuration since the difference in measurement with and without IR light hitting the detector will be much greater than with the proximity sensor.

3.5 Further Improvements

If a more robust sensor is needed there are several ways to do this both in hardware and software. Improving on the hardware has been discussed in Chapter 2 (p. 5) . By using additional LESENSE channels the pattern sent out and received could be made longer, every extra channel would require use of one or two additional pins on the EFM32GG.

Filtering in the LESENSE interrupt routine is a possibility, for example taking an additional 10 readings and checking if they are all the same. What the best alternative is depends on the application. Adding channels makes every scan use more energy, while filtering makes each interrupt longer.

The decoder state machine could be altered to change the behaviour of the sensor. For example it could be made to count number of times it detects an object, not waking up before a predefined limit is reached.

4 Revision History

4.1 Revision 1.04

2014-05-07

Updated example code to CMSIS 3.20.5

Changed source code license to Silicon Labs license

Added project files for Simplicity IDE

Removed makefiles for Sourcery CodeBench Lite

4.2 Revision 1.03

2013-10-14

New cover layout

4.3 Revision 1.02

2013-05-08

Added software projects for ARM-GCC and Atollic TrueStudio.

4.4 Revision 1.01

2012-11-12

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

4.5 Revision 1.00

2012-08-15

Initial revision.

A Disclaimer and Trademarks

A.1 Disclaimer

Silicon Laboratories 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 Silicon Laboratories 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. Silicon Laboratories 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. Silicon Laboratories 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 Silicon Laboratories. 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. Silicon Laboratories products are generally not intended for military applications. Silicon Laboratories 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

B Contact Information

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

Please visit the Silicon Labs Technical Support web page:
http://www.silabs.com/support/pages/contacttechnicalsupport.aspx
and register to submit a technical support request.

Table of Contents

List of Figures

List of Equations

silabs.com


ZERO
ARM Cortex-M0+
TINY
ARM Cortex-M3
GECKO
ARM Cortex-M3
LEOPARD
ARM Cortex-M3
GIANT
ARM Cortex-M3
WONDER
ARM Cortex-M4