

# **ECEN 5023-001, -001B, -740**

## **Mobile Computing & IoT Security**

**Lecture #4**

**26 January 2017**

# Agenda

- Class announcements
- Review of Simplicity Exercise
- Managing Energy Modes Assignment
  - **Objective:** Become familiar with the Silicon Labs' Simplicity development system as well as learn the different Leopard energy modes and how to manage these energy modes.
- Documentation style sheet
- Wireless Networks - Infrastructure
- Mobile/Pervasive Adaptive Computing System Considerations

# Class Announcements

- Quiz #2 is due at 11:59pm on Sunday, January 29<sup>th</sup>, 2017
- Register for ESE lab card access at:
  - <https://goo.gl/forms/YaBXQHATHELA2Fik2>
- Slack is set up
  - Everyone that submitted the Simplicity Exercise has been invited to the MCIOTS slack team
  - The following Slack channels have been setup
    - general
    - managingenergymodes
    - letimer0
- Managing Energy Mode Assignment is due at 11:59pm on Wednesday, February 1<sup>st</sup>, 2017
- IDE debugging tips and idea workshop to be held on Monday the 30<sup>th</sup> and Thursday the 2<sup>nd</sup>
  - Time: TBD
  - Room: TBD
  - Once finalized will be announced via Slack and D2L

# Reading List

Below is a list of required reading for this course. Questions from these readings plus the lectures from August 23<sup>rd</sup>, 2016 onward will be on the weekly quiz.

- “Testing and Debugging Concurrency Bugs in Event-Driven Programs,” Guy Martin Tchamgoue, Kyong-Hoon Kim, and Yong-Kee Jim  
<https://www.silabs.com/Support%20Documents/TechnicalDocs/manage-the-iot-on-an-energy-budget.pdf>

Recommended readings. These readings will not be on the weekly quiz, but will be helpful in the class programming assignments and course project.

- “Silicon Labs’ Energy Modes App note – AN0007”  
<http://www.silabs.com/Support%20Documents/TechnicalDocs/AN0007.pdf>
- “Protothreads: Simplifying Event-Driven Programming of Memory-Constrained Embedded Systems,” Adam Dunkels, Oliver Schmidt, Tiemo Voigt, Muneeb Ali  
<http://muneebali.com/pubs/dunkels06protothreads.pdf>
- EFM32 CMU application note - AN0004  
<http://www.silabs.com/Support%20Documents/TechnicalDocs/AN0004.pdf>
- EFM32 GPIO application note - AN0012  
<http://www.silabs.com/Support%20Documents/TechnicalDocs/AN0012.pdf>
- EFM32 Low Energy Timer LETIMER application note - AN0026  
<http://www.silabs.com/Support%20Documents/TechnicalDocs/AN0026.pdf>

Important web link below. It will take you to the Silicon Labs’ application note home page for the Silicon Labs’ EFM32 family of products:  
<http://www.silabs.com/products/mcu/Pages/32-bit-mcu-application-notes.aspx>

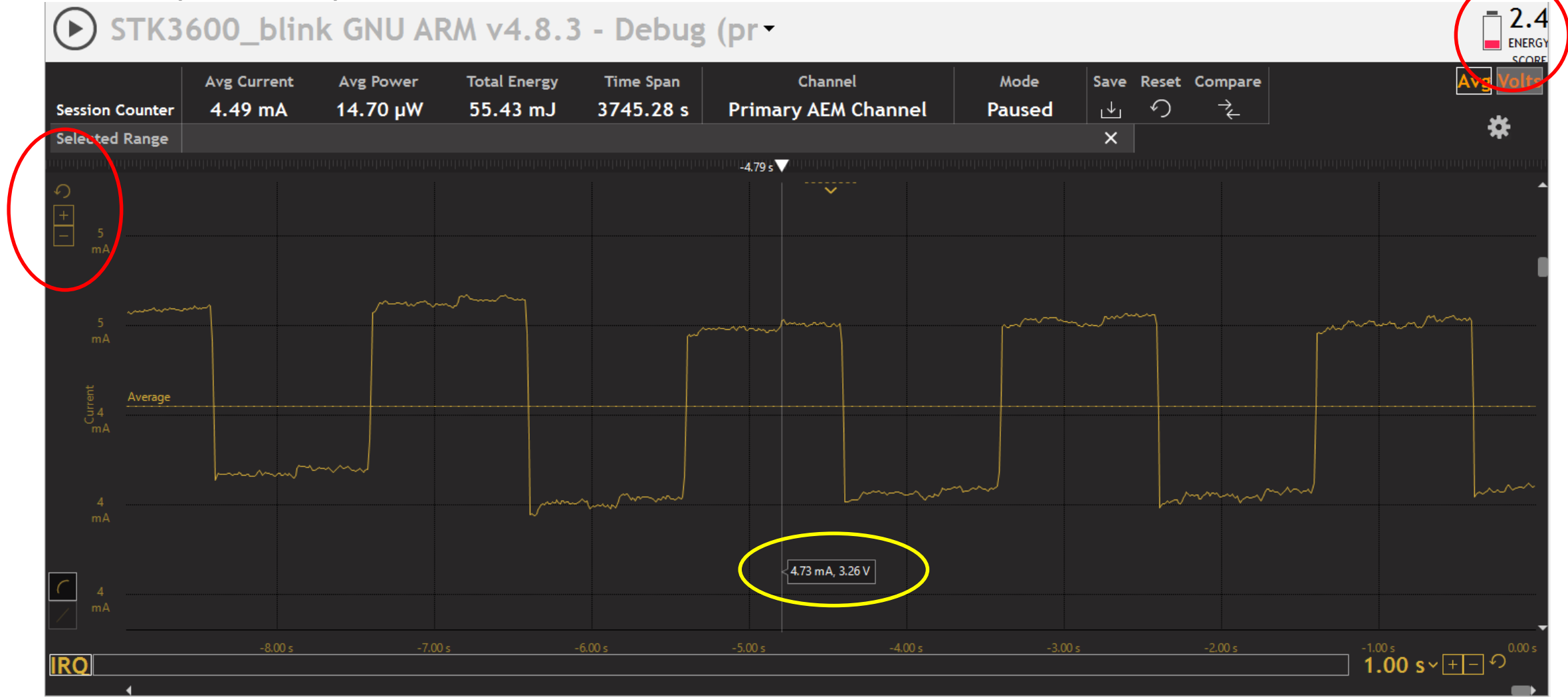
# Quiz 2

- Due by 11:59pm on Sunday, January 29<sup>th</sup>, 2017
- Questions will be from the required reading plus lectures from January 17<sup>th</sup>, 2017 onward

# Simplicity Exercise

- **Moving forward PDF code submission will not be accepted**
  - Transferring code from PDFs can cause errors in the transfer which may hurt your grade
  - Please submit the code in a text or word file
- The answers to the questions can be submitted in PDF format

# Simplicity Exercise Review



# Simplicity Exercise

1. Using the Energy Profiler, what is the instantaneous current measured once the program has begun without any modification to the sample code?
  - a. 4.90 – 5.40mA
2. What is the Energy Score after resetting the Energy Profiler and waiting 30 seconds without any modification to the sample code?
  - a. 2.3
3. After commenting out the code to toggle LED1, what is the instantaneous current measured while LED0 is off?
  - a. 4.40 – 4.95mA
4. After commenting out the code to toggle LED1, what is the instantaneous current measured while LED0 is on?
  - a. 4.85 – 5.55mA
5. After commenting out the code to toggle LED1, what is the Energy Score after resetting the Energy Profiler and waiting 30 seconds?
  - a. 2.3



# EMU – Energy Management Unit

- Will be EMU routines to enter sleep modes
  - `EMU_EnterEM1();`
  - `EMU_EnterEM2();`
  - `EMU_EnterEM3();`
  - `EMU_EnterEM4();`

# Managing Energy Modes Assignment

**Objective:** Become familiar with the Silicon Labs' Simplicity development system as well as learn the different Leopard Gecko energy modes and how to manage these energy modes.

Instructions:

1. In the Launcher view within Silicon Labs' Simplicity Studio 4, click / select the stk3600 dev board on the left
2. With the stk3600 dev kit selected, click on File in the menu bar, then click on New, click on Project, and select Silicon Labs MCU Project
3. From the top menu, select New. When you go to the right of New, select "Silicon Labs MCU Project"
4. Insure that the Leopard Gecko 3600 Starter Kit is selected for Kit and that the part number is EFM32LG990F256. Click on Next
5. Select if not already selected, Simplicity Configurator Program and click on Next
6. You can now specify the name of your project. No spaces. Click on Next
7. Insure that **Debug** is checked, and then click "Finish"

# Managing Energy Modes Assignment

8. Select the src folder in your project, and right click on main.c to rename the project to a name of your choice. No spaces are required, and you will need to add .c at the end of the project name of your choice
9. Go into your .c project source file, and delete all contents.
10. No go back to your project explorer window.
11. Add, cut and paste, all the emlib .c source file into the emlib folder of your project. The emlib .c source files can be found in the following directory:
  - a. C:\SiliconLabs\SimplicityStudio\v4\developer\sdk\exx32\v5.0.0.0\platform\emlib\src
    - i. Copy all files
  - b. Paste them into your emlib folder of your project in your IDE
  - c. When asked, select YES to Overwrite all existing libraries
  - d. This source file will compile during the next build/compile of the project
12. Develop a sleep() routine and the associated global variables and routines to manage which energy mode the Leopard Gecko can enter based on the Leopard Gecko peripherals that are being used?
  - a. Note: For EM3, the LETIMER0 will need to be associated with ULFRCO instead of LFXO
  - b. For EM0-EM2, associate the LETIMER0 with the LFXO oscillator



# Managing Energy Modes Assignment

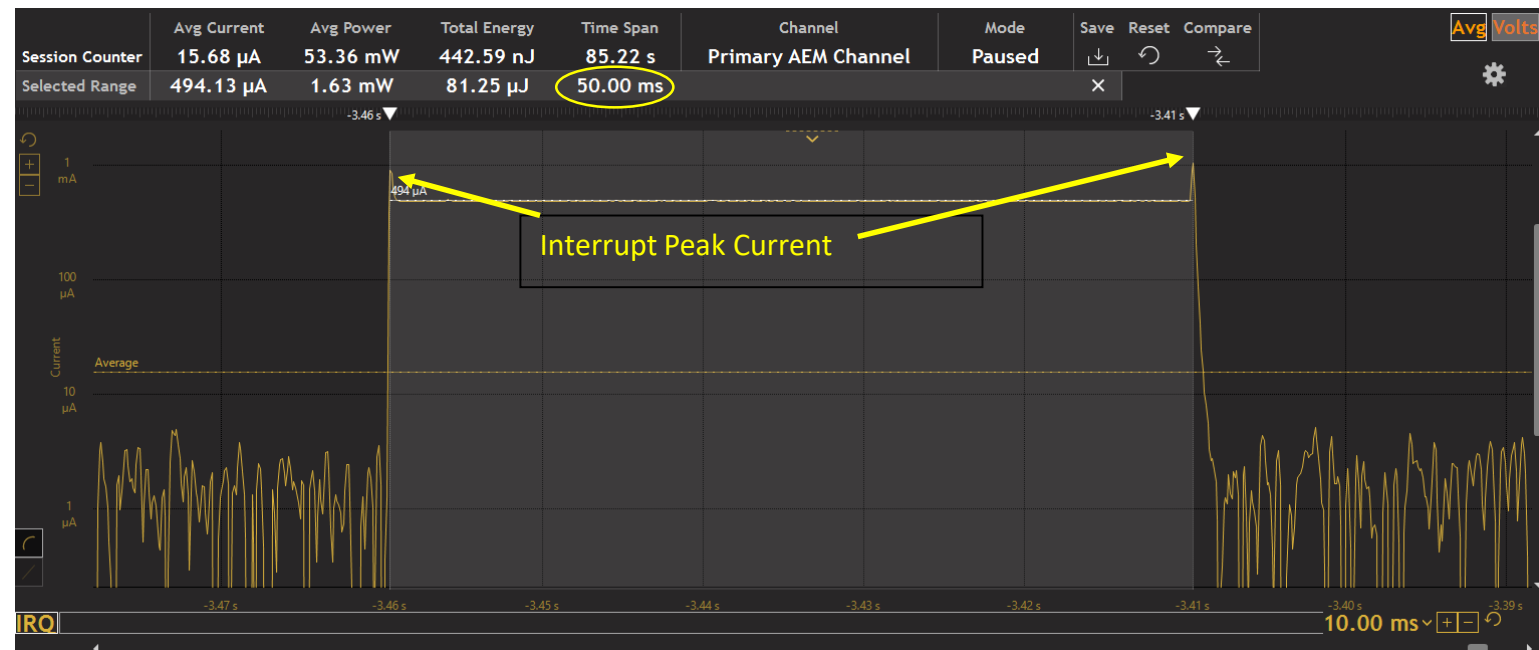
13. Develop the LETIMER0 interrupt handler routine to accomplish the next step
14. Program LETIMER0 peripheral to blink LED0 for 30mS every 1.75 seconds using the LETIMER0, LETIMER0 interrupt handler, and the sleep() routine.
  - a. Use this routine to obtain the Energy Score if LETIMER0 limits the MCU to enter the following modes while asleep: EM0, EM1, EM2, and EM3
  - b. A #define statement will be used to set the sleep mode limit
  - c. Due to the short duration of the LED0 being on, it will appear as a quick flash



# Managing Energy Mode Assignments

NOTE: TO HAVE SOME STANDARD TO DETERMINE WHERE TO PUT THE TIME CURSOR TO MEASURE TIME, YOU WILL BE MEASURING FROM INTERRUPT PEAK CURRENT TO INTERRUPT PEAK CURRENT. YOU WILL NEED TO ZOOM IN BOTH THE Y OR CURRENT AXIS AS WELL AS THE X OR TIME AXIS.

The below screen shot will show you where the measured time between the cursors is located, highlighted by the yellow circle, and where to fine the peak interrupt currents. The peaks are the result of the interrupt, and then the CPU going back to sleep but the current does not go all the way down due to the current draw of the LED.



# Managing Energy Modes Assignment

Questions:

In a separate document to be placed in the drop box with the program code, please answer the following questions:

**NOTE: All average currents should be taken at a time scale of 1s/div.**

1. What is the Energy Profiler score when the Leopard Gecko is only allowed to be in EM0 and what is the average current when the LED is off? What is the average current when the LED is off?
  - a. Wait 60 seconds after the Energy score is RESET
2. What is the Energy Profiler score when the Leopard Gecko can only go down to EM1 and what is the average current when the LED is off? What is the average current when the LED is off?
  - a. Wait 60 seconds after the Energy score is RESET
3. What is the Energy Profiler score when the Leopard Gecko can only go down to EM2 and what is the average current when the LED is off? What is the average current when the LED is off?
  - a. Wait 60 seconds after the Energy score is RESET
4. What is the period in milliseconds of the LED blinking using EM2 using the Energy Profiler “selected range” markers? What is the On-Duty Cycle in milliseconds of the LED using the Energy Profiler while limited to EM2?
5. What is the Energy Profiler score when the Leopard Gecko can go down to EM3 and what is the average current when the LED is off? ? What is the average current when the LED is off?
  - a. Wait 60 seconds after the Energy score is RESET
6. What is the period in milliseconds of the LED blinking using EM3 using the Energy Profiler “selected range” markers? What is the On-Duty Cycle in milliseconds of the LED using the Energy Profiler while limited to EM3?

# Managing Energy Modes Assignment


## Deliverables:

1. One document that provides the answers to Assignment #1
2. Another document that contains your .c project source file with LETIMER0 set to EM3 mode

## Please note:

1. There will need to be include statements for the following:
  - a. `#include <stdint.h>`
  - b. `#include <stdbool.h>`
  - c. `#include "em_device.h"`
  - d. `#include "em_chip.h"`
  - e. `#include "em_int.h"`
  - f. as well as for each peripheral used; CMU, EMU, LETIMER, and GPIO
2. There should be a `#defined` statement that can be used to switch the lowest energy mode available to the LETIMER0
3. In the `#defined` statements, the period between LED0 being turned on should be defined in seconds. By giving a new definition of time in seconds to this `#define` statement, the period should reflex the new period the next time the program is ran.

# Managing Energy Modes Assignment

4. In the #defined statements, the On-Duty Cycle between LED0 being turned on and off should be defined in seconds. By giving a new definition of time in seconds to this #define statement, the On-Duty Cycle should reflex the new period the next time the program is ran. 
5. The program should contain as a minimum, the following routines:
  - a. Sleep mode routines
    - I. Going to sleep
    - II. Selecting which sleep mode the system is limited to while the peripheral is required
    - III. Removing the restriction of the block sleep mode when the peripheral is no longer required
  - b. Setup routines
    - I. CMU
    - II. GPIO
    - III. LETIMER0
  - c. An interrupt handler for LETIMER0
  - d. A routine to turn-on and off the LED
6. Int main(void) should have the following as its first two lines of code:
  - a. `CHIP_Init(); /* Sets the part for any known errata */`
  - b. `Your block mode sleep routine set to EM3 /* Want to insure at this point that the program can never go into EM4. If your board goes to EM4, it will lock up for a period of time from 8 to 12 hours */`





# Now accepting assignments with header files (.h and .c)

- We now believe we can accept program assignments using .h and .c files
  - Please use the following rules:
    - The .h and .c files must reside in the /src folder of your project
    - The main program must be named main.c
    - Please add a main.h file that will be used for system wide parameters such as
      - Upper Temperature Limit
      - Lower Temperature Limit
      - Enabling or turning off DMA to the ADC
      - Etc.

# Program documentation style sheet

- Can be found in the course D2L Content Section under Course Content
  - [MCIOTS Style document – Spring 2017.pdf](#)
- The purpose of this document is to set a minimum set of rules of program code documentation to enable:
  - Understanding of code
  - Enable the course instructing team to assist with code when requested
  - Grade the programming assignments
- Not following this style sheet will result in **reduction** of programming assignment scores

# MCIOTS Style Sheet – Fall 2017

1. At the top of the program, please include your name, date, and the assignment information.
2. `#include` statements
3. `#define` statements
  - a. macros, Typedefs, constants
  - b. constants should be enumerated here and not using number constants
  - c. example:

<code>#define</code>	<code>EM0</code>	<code>0</code>
<code>#define</code>	<code>EM1</code>	<code>1</code>
<code>#define</code>	<code>EM2</code>	<code>2</code>
<code>#define</code>	<code>EM3</code>	<code>3</code>
<code>#define</code>	<code>EM4</code>	<code>4</code>
4. Global variables
5. function prototypes

# MCIOTS Style Sheet – Fall 2016

## 6. Suggested order of functions

- a. Sleep routines
- b. Peripheral functions
- c. Interrupt Handlers
- d. Other functions
- e. Main ()

## 7. Before each function, a brief description should state the purpose of the function, key variables, and any IP notices. Pseudo code example below:

- `/* **** */`
- `/* Routine to show an example of how it should be documented */`
- `/* before the function. */`
- `/* */`
- `/* Input variables: int_example is used as an input to this */`
- `/* to show an example of the suggested documentation. */`
- `/* Global variables: State_Machine_Example[] is used to update */`

# MCIOTS Style Sheet – Fall 2016

```

/*    current state of this example routine.    */
/*    Returned variables: return_value is the value that is returned    */
/*    Include any IP statement indicating where the below routine    */
/*    came from and any legal requirements that must be stated.    */
/*******/

```

```

Int Example_Routine (unsigned int int_example) {
    Int return_value;

    return return_value;
}

```

8. Must properly give copy right credit where due.

Note: Where code is used from another source such as Silicon Labs' application examples, please read the appropriate code header and add appropriate comments. For example, the below header specifies that the sleep code can be used commercially as long as its origins are not misrepresented. Please add a comment before the Sleep Routines that this code is originally Silicon Labs and copy righted by Silicon Labs' in 2015 and Silicon Labs' grants permission to anyone to use the software for any purpose, including commercial applications, and to alter it, and redistribute it freely subject that the origins is not misrepresented, altered source version must be plainly marked, and this notice cannot be altered or removed from any source distribution.

# MCIOTS Style Sheet – Fall 2016

```

/*****
 * @file sleep.c
 *****/
 * @section License
 * <b>(C) Copyright 2015 Silicon Labs, http://www.silabs.com</b>
 *****/
 *
 * Permission is granted to anyone to use this software for any purpose,
 * including commercial applications, and to alter it and redistribute it
 * freely, subject to the following restrictions:
 *
 * 1. The origin of this software must not be misrepresented; you must not
 *    claim that you wrote the original software.
 * 2. Altered source versions must be plainly marked as such, and must not be
 *    misrepresented as being the original software.
 * 3. This notice may not be removed or altered from any source distribution.
 *
 * DISCLAIMER OF WARRANTY/LIMITATION OF REMEDIES: Silicon Labs has no
 * obligation to support this Software. Silicon Labs is providing the
 * Software "AS IS", with no express or implied warranties of any kind,
 * including, but not limited to, any implied warranties of merchantability
 * or fitness for any particular purpose or warranties against infringement
 * of any proprietary rights of a third party.
 *
 * Silicon Labs will not be liable for any consequential, incidental, or
 * special damages, or any other relief, or for any claim by any third party,
 * arising from your use of this Software.
 *****/

```

# MCIOTS Style Sheet – Fall 2016

For the instructing team to provide help and suggestions with programming assignments, the following documentation must be included. Without this documentation, it is very difficult and time consuming to review and provide help.

1. No use of Init TypeDef defaults. The constructs must be fully specified by one of the following methods. The instructing team does not memorize or know the defaults of the Init TypeDef defaults.

```

LETIMER Init_TypeDef      LETIMER0_init;
int  intFlags;
int  Comp0_init;
int  Comp1_init;
int  LETIMER0_prescaler;
int  ULFRCO_count_calibrated;

LETIMER0_init.bufTop = false;
LETIMER0_init.comp0Top = true;
LETIMER0_init.debugRun = false;
LETIMER0_init.enable = false;
LETIMER0_init.out0Pol = 0;
LETIMER0_init.out1Pol = 0;
LETIMER0_init.repMode = letimerRepeatFree;
LETIMER0_init.rtcComp0Enable = false;
LETIMER0_init.rtcComp1Enable = false;
LETIMER0_init.ufoa0 = letimerUFOANone;
LETIMER0_init.ufoa1 = letimerUFOANone;

LETIMER_Init(LETIMER0, &LETIMER0_init);
  
```

Or,

# MCIOTS Style Sheet – Fall 2016

```

/* Set configurations for LETIMER 0 */
const LETIMER_Init_TypeDef letimerInit =
{
    .enable           = true,
    .debugRun         = false,
    .rtcComp0Enable   = false,
    .rtcComp1Enable   = false,
    .comp0Top         = true,
    .bufTop           = false,
    .out0Pol          = 0,
    .out1Pol          = 0,
    .ufoa0            = letimerUFOAPwm,
    .ufoa1            = letimerUFOAPulse,
    .repMode          = letimerRepeatFree
};

/* Initialize LETIMER */
LETIMER_Init(LETIMER0, &letimerInit);

```

2. No register bit manipulation with direct bit or hex values where appropriate and possible. Use of using the enumerations provided by the MCU/SOC provider to address bit names and fields. Similar to the use of Init TypeDef defaults, the instruction team does not memorize the value of each bit field in a register.

<del>LETIMER0-&gt;IEN  = 0x00000004;</del>	<del>Not acceptable</del>
LETIMER0->IEN  = LETIMER_IEN_UF;	Acceptable



# Energy Profiler providing erratic results

- If your Energy profiler is measuring current in the sleep mode  $<$  than 1nA which is not possible, you may need to reset the Energy profiler and Simplicity to obtain correct results. Below is the suggested process to reset your environment:
  1. Quit Studio.
  2. Unplug your device from PC, then re-plug it.
  3. Start Studio. That should solve the problem.

This problem usually happens when you plug in your device while Studio is running.

# Application notes

+ Software examples

## EFM32 Application Notes

 [Getting Started with EFM32](#)

<http://www.silabs.com/products/mcu/Pages/32-bit-mcu-application-notes.aspx>



Title	App Note	Software
Hardware Design Considerations	<a href="#">AN0002</a>	<a href="#">AN0002SW</a>
UART Bootloader	<a href="#">AN0003</a>	<a href="#">AN0003SW</a>
Clock Management Unit	<a href="#">AN0004</a>	<a href="#">AN0004SW</a>
Real-Time Counter	<a href="#">AN0005</a>	<a href="#">AN0005SW</a>
Real-Time Counter Calendar	<a href="#">AN0006</a>	<a href="#">AN0006SW</a>
Energy Modes	<a href="#">AN0007</a>	<a href="#">AN0007SW</a>
USART Synchronous Mode	<a href="#">AN0008</a>	<a href="#">AN0008SW</a>
EFM32 Getting Started	<a href="#">AN0009</a>	<a href="#">AN0009SW</a>
I <sup>2</sup> C Master and Slave Operation	<a href="#">AN0011</a>	<a href="#">AN0011SW</a>
GPIO	<a href="#">AN0012</a>	<a href="#">AN0012SW</a>
Direct Memory Access	<a href="#">AN0013</a>	<a href="#">AN0013SW</a>
TIMER	<a href="#">AN0014</a>	<a href="#">AN0014SW</a>



Launcher - Simplicity Studio™

File Edit Navigate Search Project Run Window Help

Sign In

Search

Tools

Launcher Simplicity IDE Energy Profiler Resources

• Device • **Solutions**

New Solution Add Devices

Custom Solution

EFM32 Leopard Gecko Starter Kit (EFM32LG-STK360

EFM32 Leopard Gecko Starter Kit board (BRD220

EFM32LG990F256

# EFM32 Leopard Gecko Starter Kit board (BRD2201A Rev A02)

Preferred SDK: Gecko SDK v4.4.1 Click [here](#) to change the preferred SDK.

New Project

Recent Projects

Getting Started

Documentation

Compatible Tools

Resources

EFM32 Leopard Gecko Starter Kit board (BRD2201A Rev A02)

BRD2201A\_A02\_assy\_draw

EFM32G Starter Kit (STK) Bill of Materials

EFM32LG Starter Kit (STK) Schematic

EFM32LG990 Data Sheet

EFM32LG Reference Manual

EFM32LG990 Rev E Errata

Gecko SDK Documentation

Electrical, Computer & Energy Engineering

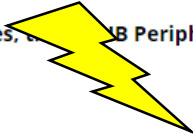
UNIVERSITY OF COLORADO BOULDER

Silicon Labs' Simplicity Studio

27



# Gecko SDK Documentation

Welcome to the documentation for the Gecko SDK for Silicon Labs 32-bit MCUs and SoCs. The Gecko SDK contains device header files,  ARM Peripheral Library, EnergyAware Driver library, middleware and support for Starter Kits and Development Kits. Please select your device family from the list below.

## EFM32 Gecko

- [Software Documentation](#)

## EFM32 Wonder Gecko

- [Software Documentation](#)

## EFM32 Jade Gecko

- [Software Documentation](#)

## EFR32 Blue Gecko

- [Software Documentation](#)

## EFM32 Tiny Gecko

- [Software Documentation](#)

## EFM32 Zero Gecko

- [Software Documentation](#)

## EZR32 Happy Gecko

- [Software Documentation](#)

## EFR32 Mighty Gecko

- [Software Documentation](#)

## EFM32 Giant Gecko

- [Software Documentation](#)

## EFM32 Happy Gecko

- [Software Documentation](#)

## EZR32 Leopard Gecko

- [Software Documentation](#)

## EFR32 Flex Gecko

- [Software Documentation](#)

## EFM32 Leopard Gecko

- [Software Documentation](#)

## EFM32 Pearl Gecko

- [Software Documentation](#)

## EZR32 Wonder Gecko

- [Software Documentation](#)

## ARM CMSIS Documentation

- [ARM CMSIS Documentation](#)

## ARM mbed TLS Documentation

- [ARM mbed TLS Documentation](#)



# EFM32 Leopard Gecko Software Documentation

efm32lg-doc-4.4.0

[Main Page](#)[Mo](#)[Files](#)[Documentation Home](#)[silabs.com](#)[EFM32 Leopard Gecko Software Documentation](#)[► Modules](#)[► Files](#)[Documentation Home](#)[silabs.com](#)

## EFM32 Leopard Gecko Software Documentation

Welcome to the software documentation for the EFM32 Leopard Gecko. Here, you will find documentation for

- The **CMSIS-CORE Device headers** for the EFM32 Leopard Gecko
- The **EMLIB Peripheral Library**
- The **EnergyAware Driver** library
- The **Board Support Package**
- The **Kit Drivers** library
- The **USB Host and Device** stack

Please also see [Simplicity Studio](#) for precompiled demo applications, application notes and software examples.



# EFM32 Leopard Gecko Software Documentation

efm32lg-doc-4.4.0

[Main Page](#)[Modules](#)[Files](#)[Documentation Home](#)[silabs.com](#)**EFM32 Leopard Gecko Software Documentation**

## EFM32 Leopard Gecko Software Documentation

Welcome to the software documentation for the EFM32 Leopard Gecko. Here, you will find documentation for

- The **CMSIS-CORE Device headers** for the EFM32 Leopard Gecko
- The **EMLIB Peripheral Library**
- The **EnergyAware Driver** library
- The **Board Support Package**
- The **Kit Drivers** library
- The **USB Host and Device** stack

Please also see **Simplicity Studio** for precompiled demo applications, application notes and software examples.

### ▼ Modules

- ▶ BSP
- ▶ Devices
- ▶ EMDRV
- ▶ EMLIB
- ▶ Kit Drivers
- ▶ USB

### ▶ Files

[Documentation Home](#)  
[silabs.com](#)

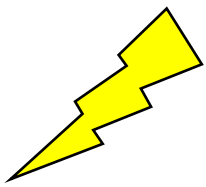


# EFM32 Leopard Gecko Software Documentation

efm32lg-doc-4.4.0

Search

- EFM32 Leopard Gecko Software Documentation
- Modules
  - BSP
  - Devices
  - EMDRV
  - EMLIB
    - ACMP
    - ADC
    - AES
    - BITBAND
    - BURTC
    - BUS
    - CHIP
    - CMU
    - COMMON
    - DAC
    - DBG
    - DMA
    - EBI
    - EMU
    - GPIO
    - I2C
    - INT



## EFM32 Leopard Gecko Software Documentation

Welcome to the software documentation for the EFM32 Leopard Gecko. Here, you will find documentation for

- The **CMSIS-CORE Device headers** for the EFM32 Leopard Gecko
- The **EMLIB Peripheral Library**
- The **EnergyAware Driver** library
- The **Board Support Package**
- The **Kit Drivers** library
- The **USB Host and Device** stack

Please also see **Simplicity Studio** for precompiled demo applications, application notes and software examples.





# EFM32 Leopard Gecko Software Documentation

efm32lg-doc-4.4.0

Main Page

Modules

Files

Documentation Home

silabs.com

Search

GPIO\_IntClear  
GPIO\_IntConfig  
GPIO\_IntDisable  
GPIO\_IntEnable  
GPIO\_IntGet  
GPIO\_IntGetEnabled  
GPIO\_IntSet  
GPIO\_Lock  
GPIO\_PinInGet  
GPIO\_PinModeGet  
GPIO\_PinModeSet  
GPIO\_PinOutClear  
GPIO\_PinOutGet  
GPIO\_PinOutSet  
GPIO\_PinOutToggle  
GPIO\_PortInGet  
GPIO\_PortOutClear  
GPIO\_PortOutGet  
GPIO\_PortOutSet  
GPIO\_PortOutSetVal  
GPIO\_PortOutToggle  
GPIO\_Unlock

► I2C

## GPIO

EMLIB

Enumerations | Functions

### Detailed Description

General Purpose Input/Output (GPIO) API.

This module contains functions to control the GPIO peripheral of Silicon Labs 32-bit MCUs and SoCs. The GPIO peripheral is used for pin configuration and direct pin manipulation and sensing as well as routing for peripheral pin connections.

### Enumerations

```
enum GPIO_DriveMode_TypeDef {
    gpioDriveModeStandard = GPIO_P_CTRL_DRIVEMODE_STANDARD,
    gpioDriveModeLowest = GPIO_P_CTRL_DRIVEMODE_LOWEST,
    gpioDriveModeHigh = GPIO_P_CTRL_DRIVEMODE_HIGH,
    gpioDriveModeLow = GPIO_P_CTRL_DRIVEMODE_LOW
}

enum GPIO_Mode_TypeDef {
    gpioModeDisabled = _GPIO_P_MODEL_MODE0_DISABLED,
    gpioModeInput = _GPIO_P_MODEL_MODE0_INPUT,
    gpioModeInputPull = _GPIO_P_MODEL_MODE0_INPUTPULL,
    gpioModeInputPullFilter = _GPIO_P_MODEL_MODE0_INPUTPULLFILTER,
    gpioModePushPull = _GPIO_P_MODEL_MODE0_PUSHPULL,
    gpioModePushPullDrive = _GPIO_P_MODEL_MODE0_PUSHPULLDRIVE,
    gpioModeWiredOr = _GPIO_P_MODEL_MODE0_WIREDOR,
    gpioModeWiredOrPullDown = _GPIO_P_MODEL_MODE0_WIREDORPULLDOWN,
    gpioModeWiredAnd = _GPIO_P_MODEL_MODE0_WIREDAND,
    gpioModeWiredAndFilter = _GPIO_P_MODEL_MODE0_WIREDANDFILTER,
    gpioModeWiredAndPullUp = _GPIO_P_MODEL_MODE0_WIREDANDPULLUP,
    gpioModeWiredAndPullUpFilter = _GPIO_P_MODEL_MODE0_WIREDANDPULLUPFILTER,
}
```







# EFM32 Leopard Gecko Software Documentation

efm32lg-doc-4.4.0

Main Page Modules Files Documentation Home silabs.com

Search

GPIO\_ExtIntConfig  
GPIO\_InputSenseSet  
GPIO\_IntClear  
GPIO\_IntConfig  
GPIO\_IntDisable  
GPIO\_IntEnable  
GPIO\_IntGet  
GPIO\_IntGetEnabled  
GPIO\_IntSet  
GPIO\_Lock  
GPIO\_PinInGet  
GPIO\_PinModeGet  
**GPIO\_PinModeSet**  
GPIO\_PinOutClear  
GPIO\_PinOutGet  
GPIO\_PinOutSet  
GPIO\_PinOutToggle  
GPIO\_PortInGet  
GPIO\_PortOutClear  
GPIO\_PortOutGet  
GPIO\_PortOutSet  
GPIO\_PortOutSetVal  
GPIO\_PortOutToggle

```
void GPIO_PinModeSet ( GPIO_Port_TypeDef port,  
                      unsigned int pin,  
                      GPIO_Mode_TypeDef mode,  
                      unsigned int out  
                      )
```

Set the mode for a GPIO pin.

#### Parameters

[in] port The GPIO port to access.  
[in] pin The pin number in the port.  
[in] mode The desired pin mode.  
[in] out Value to set for pin in DOUT register. The DOUT setting is important for even some input mode configurations, determining pull-up/down direction.

Definition at line 265 of file `em_gpio.c`.

References `GPIO`, `GPIO_PinOutClear()`, `GPIO_PinOutSet()`, and `gpioModeDisabled`.

Referenced by `ADC0_IRQHandler()`, `BSP_BccPinsEnable()`, `BSP_BusControlModeSet()`, `BSP_EbiInit()`, `BSP_McuBoard_DeInit()`, `BSP_McuBoard_Init()`, `BSP_McuBoard_UsbVbusPowerEnable()`, `CAPSENSE_setupGPIO()`, `DBG_SWOEnable()`, `eZRADIO_hal_GpioInit()`, `I2CSPM_Init()`, `KSZ8851SNL_SPI_Init()`, `MICROSD_DeInit()`, `MICROSD_Init()`, `MSDD_Init()`, `RETARGET_SerialInit()`, `setupI2C()`, `SPI_TFT_Init()`, and `TFT_DirectGPIOConfig()`.

```
_STATIC_INLINE void GPIO_PinOutClear ( GPIO_Port_TypeDef port,  
                                       unsigned int pin  
                                       )
```

Set a single pin in GPIO data out port register to 0.

#### Note

In order for the setting to take effect on the output pad, the pin must have been configured properly. If not, it will take effect whenever the pin has been properly configured.

#### Parameters





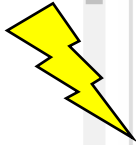
# EFM32 Leopard Gecko Software Documentation

efm32lg-doc-4.4.0

Main Page Modules Files Documentation Home silabs.com

Search

- ▶ BURTC
- ▶ BUS
- ▶ CHIP
- ▶ CMU
- ▶ COMMON
- ▶ DAC
- ▶ DBG
- ▶ DMA
- ▶ EBI
- ▶ EMU
- ▼ GPIO
  - ▶ GPIO\_DriveMode\_TypeDef
  - ▶ **GPIO\_Mode\_TypeDef**
  - GPIO\_Port\_TypeDef
  - GPIO\_DbgLocationSet
  - GPIO\_DbgSWDCLKEnable
  - GPIO\_DbgSWDIOEnable
  - GPIO\_DbgSWOEnable
  - GPIO\_DriveModeSet
  - GPIO\_EM4DisablePinWakeup
  - GPIO\_EM4EnablePinWakeup
  - GPIO\_EM4GetPinWakeupCause
  - GPIO\_EM4SetPinRetention



## enum GPIO\_Mode\_TypeDef

Pin mode. For more details on each mode, please refer to the reference manual.

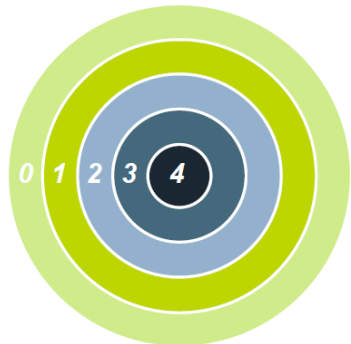
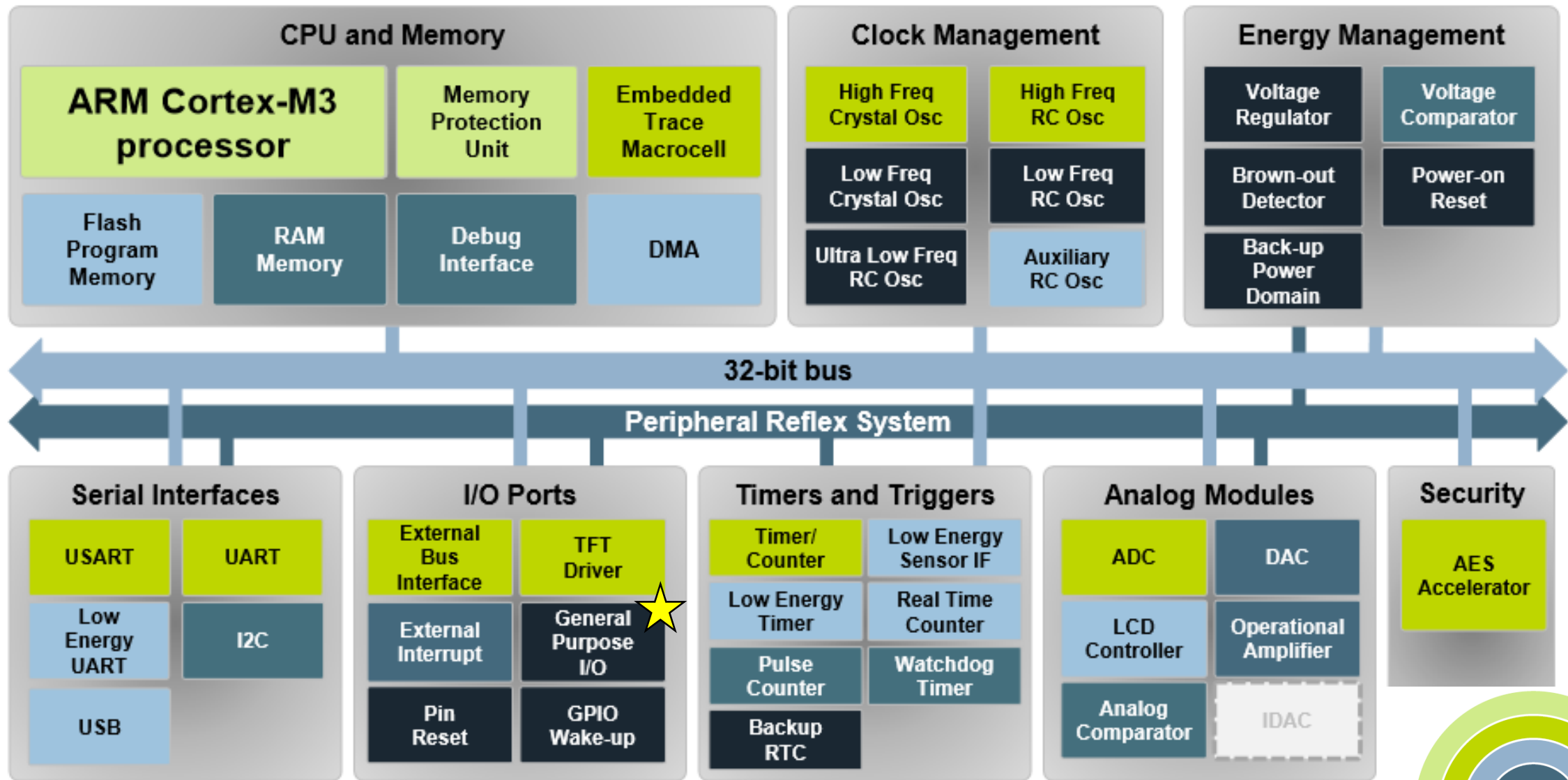
### Enumerator

gpioModeDisabled	Input disabled. Pullup if DOUT is set.
gpioModeInput	Input enabled. Filter if DOUT is set
gpioModeInputPull	Input enabled. DOUT determines pull direction
gpioModeInputPullFilter	Input enabled with filter. DOUT determines pull direction
gpioModePushPull	Push-pull output
gpioModePushPullDrive	Push-pull output with drive-strength set by DRIVEMODE
gpioModeWiredOr	Wired-or output
gpioModeWiredOrPullDown	Wired-or output with pull-down
gpioModeWiredAnd	Open-drain output
gpioModeWiredAndFilter	Open-drain output with filter
gpioModeWiredAndPullUp	Open-drain output with pullup
gpioModeWiredAndPullUpFilter	Open-drain output with filter and pullup
gpioModeWiredAndDrive	Open-drain output with drive-strength set by DRIVEMODE
gpioModeWiredAndDriveFilter	Open-drain output with filter and drive-strength set by DRIVEMODE
gpioModeWiredAndDrivePullUp	Open-drain output with pullup and drive-strength set by DRIVEMODE
gpioModeWiredAndDrivePullUpFilter	Open-drain output with filter, pullup and drive-strength set by DRIVEMODE

Definition at line 318 of file **em\_gpio.h**.

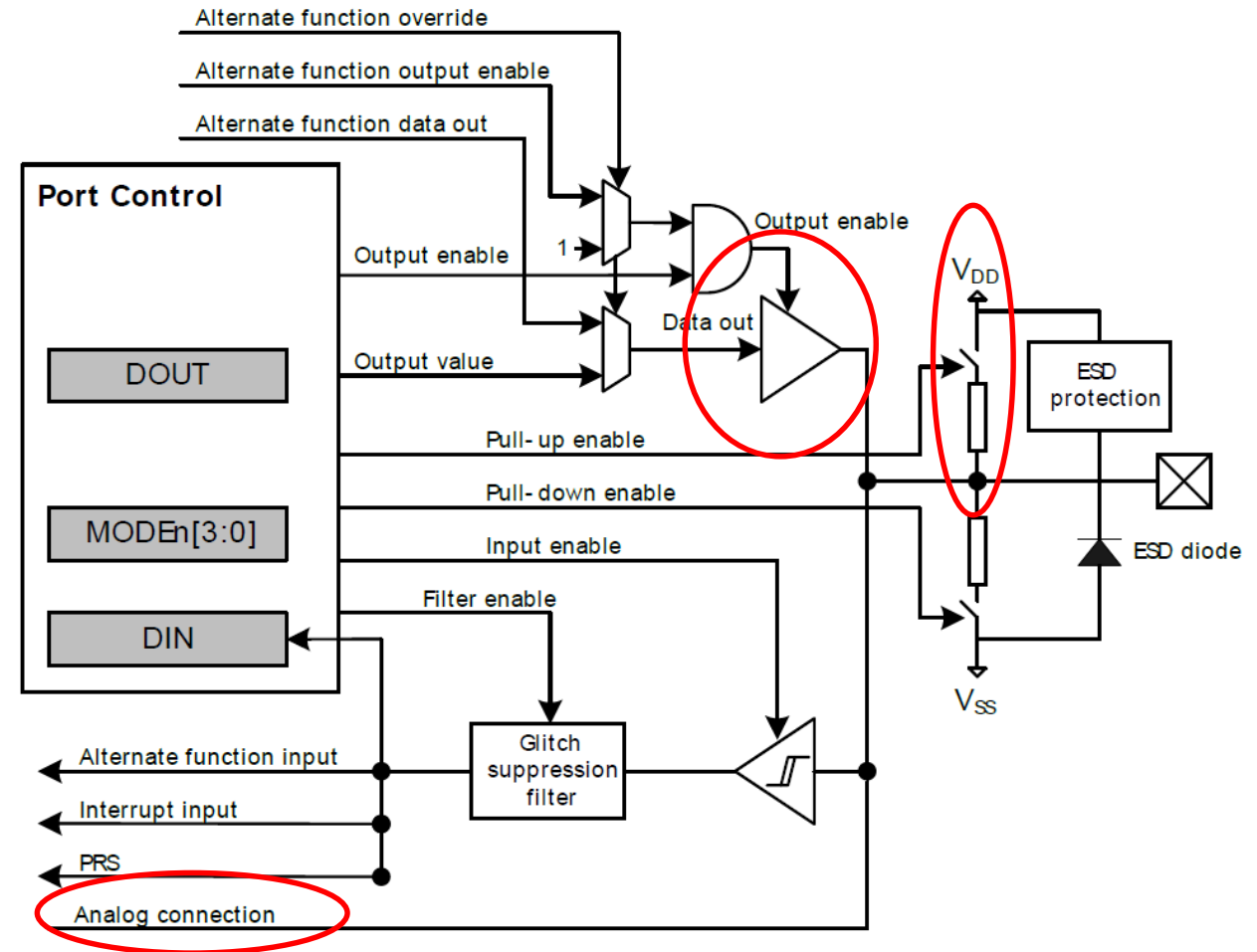
## enum GPIO\_Port\_TypeDef





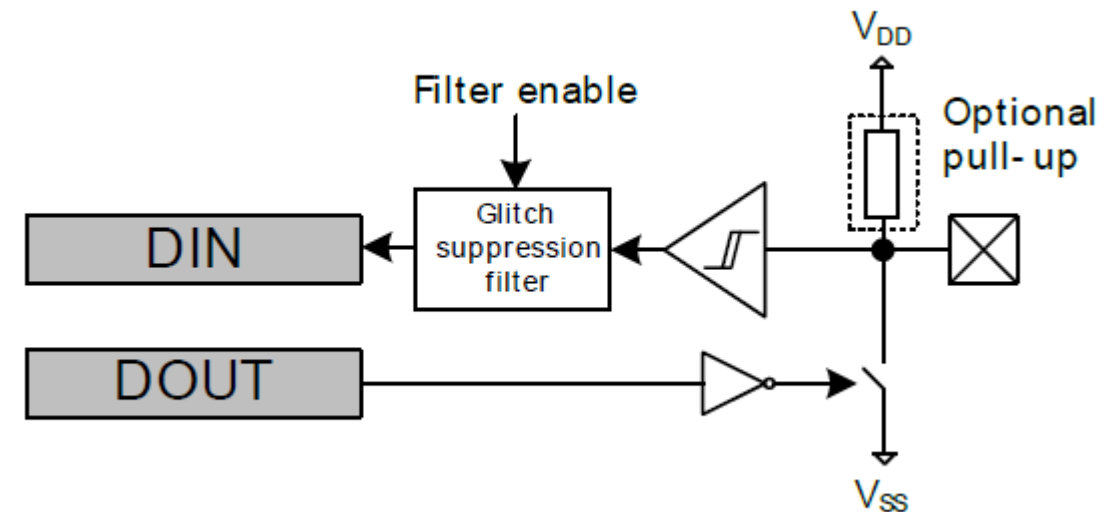
# GPIO Peripheral

- Individual configuration for each pin
  - Tristate (reset state)
  - Push-pull
  - Open-drain
  - Pull-up resistor
  - Pull-down resistor
  - Four drive strength modes
    - HIGH
    - STANDARD
    - LOW
    - LOWEST



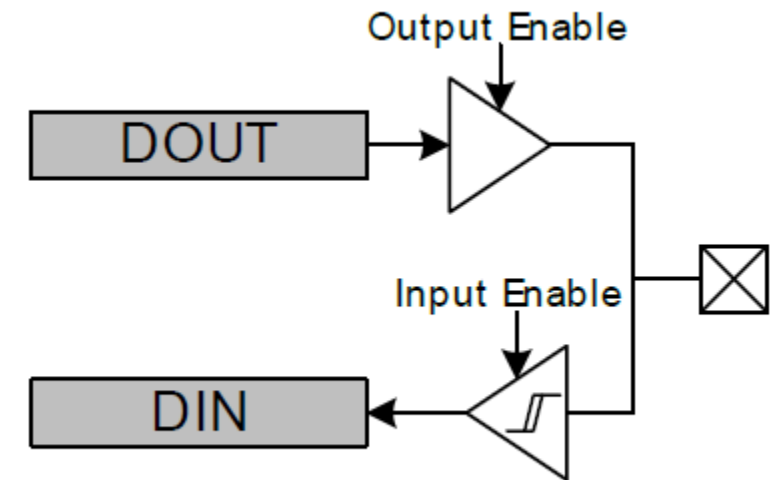
# GPIO – Open drain

- Common configuration when multiple sources may drive the bus
  - I2C



# GPIO – Push/Pull configuration

- Common Output or input configuration when a single source is driving the net
- In output mode, it can be used to:
  - Source current to an LED by driving the positive node of the LED
  - Or, sink current from an LED by connecting to ground the negative/ground node of the LED



# Setting up the GPIO

- First, the clock tree to the GPIO must be established
  - Without establishing the clock tree, all writes to the GPIO registers will not occur
  - Pseudo code in the CMU setup routine to enable the GPIO clock tree:
    - Lastly, enable the GPIO clocking using the [CMU\\_ClockEnable](#) for the GPIO

# Setting up the GPIO

- Second, the GPIO must be set up
  - Specifying the pins
    - Both the Port and Pin # is required
    - For the Leopard Gecko, use the schematic from Simplicity
  - What is the function of the pins?
    - Push-Pull
    - Open drain
    - Etc.
  - Program the functionality of the GPIO pin using [GPIO\\_PinModeSet](#)
  - Program drive strength of the GPIO pin using [GPIO\\_DriveModeSet](#)



Launcher - Simplicity Studio™

File Edit Navigate Search Project Run Window Help

Sign In Search Tools

Launcher Simplicity IDE Energy Profiler Resource

• Device • **Solutions**

New Solution Add Devices

Custom Solution

- EFM32 Leopard Gecko Starter Kit (EFM32LG-STK360)
- EFM32 Leopard Gecko Starter Kit board (BRD2201A Rev A02)
  - EFM32LG990F256

# EFM32 Leopard Gecko Starter Kit board (BRD2201A Rev A02)

Preferred SDK: Gecko SDK v4.4.1 Click [here](#) to change the preferred SDK.

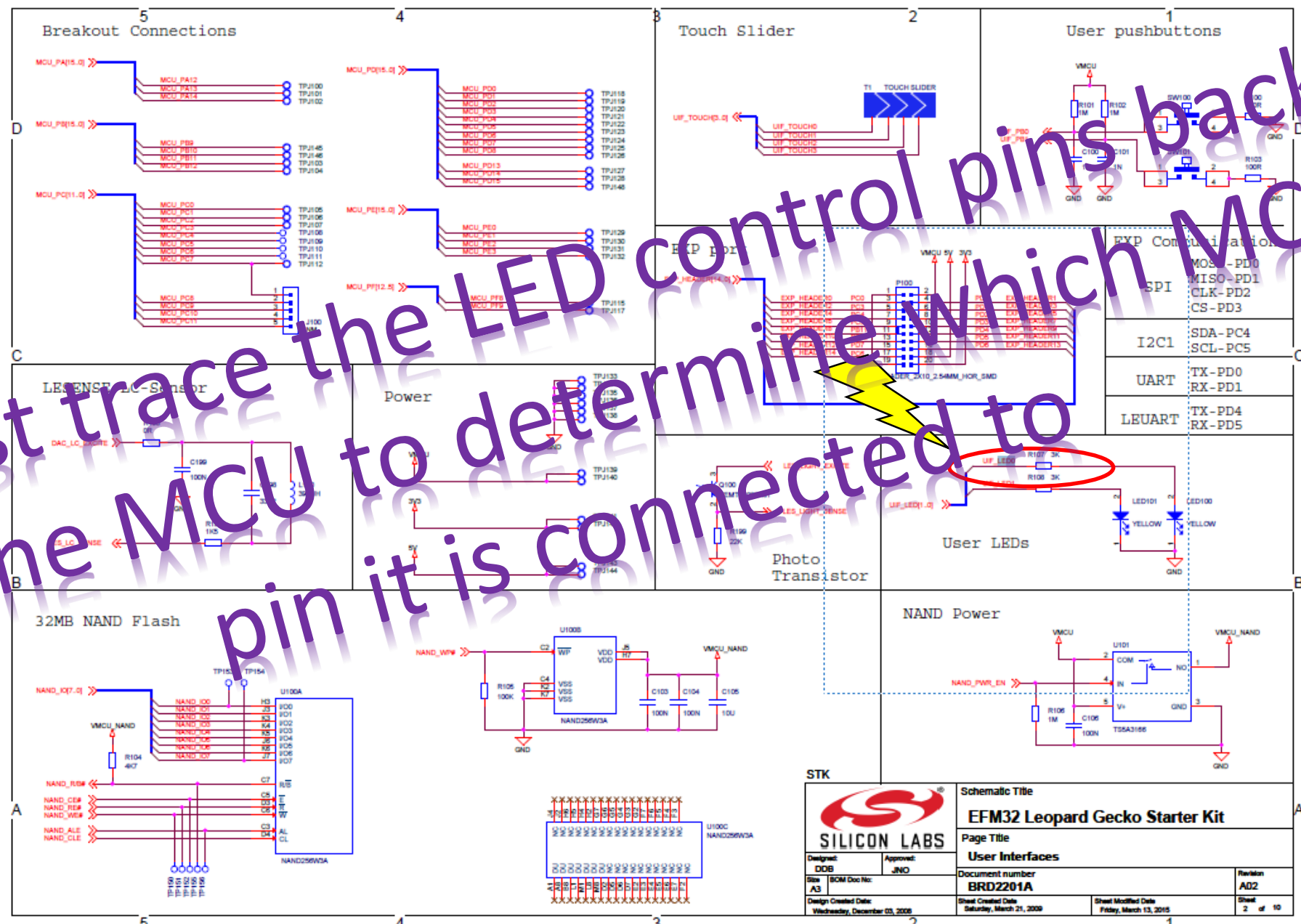
New Project Recent Projects

## Getting Started Documentation Compatible Tools Resources

EFM32 Leopard Gecko Starter Kit board (BRD2201A Rev A02)

BRD2201A_A02_assy_draw	EFM32G Starter Kit (STK) Bill of Materials	EFM32LG Starter Kit (STK) Schematic
EFM32LG990 Data Sheet	EFM32LG Reference Manual	EFM32LG990 Rev E Errata
Gecko SDK Documentation		





Must trace the LED control pins back to the MCU to determine which pin it is connected to

# Setting up the GPIO

- Third, the GPIO interrupts must be enabled if needed
  - Clear all interrupts from the GPIO to remove any interrupts that may have been set up inadvertently by accessing the `GPIO->IFC` register or the emlib routine
  - `GPIO_IntConfig` emlib command to set GPIO interrupts
  - Enable the appropriate GPIO interrupts by setting the appropriate bits in the `GPIO->IEN` register or using an emlib routine
    - There are two interrupt vectors (handlers) for the GPIO
      - Even GPIO pins
      - Odd GPIO pins
  - No need to set `BlockSleep` mode since GPIO pins work EM0 thru EM3
  - A subset of GPIO pins works down to EM4
  - Enable interrupts to the CPU by enabling the GPIO in the Nested Vector Interrupt Control register using `NVIC_EnableIRQ(GPIO_EVEN_IRQn);` or `NVIC_EnableIRQ(GPIO_ODD_IRQn);`

# Setting up the GPIO

- Fourth, the GPIO interrupt handler must be included

- Routine name must match the vector table name:

```
Void GPIO_EVEN_IRQHandler(void) {  
}
```

Or

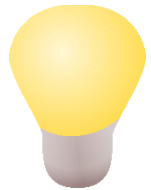
```
Void GPIO_ODD_IRQHandler(void) {  
}
```

- Inside this routine, you add the functionality that is desired for the GPIO interrupts

# GPIO – Input / Output



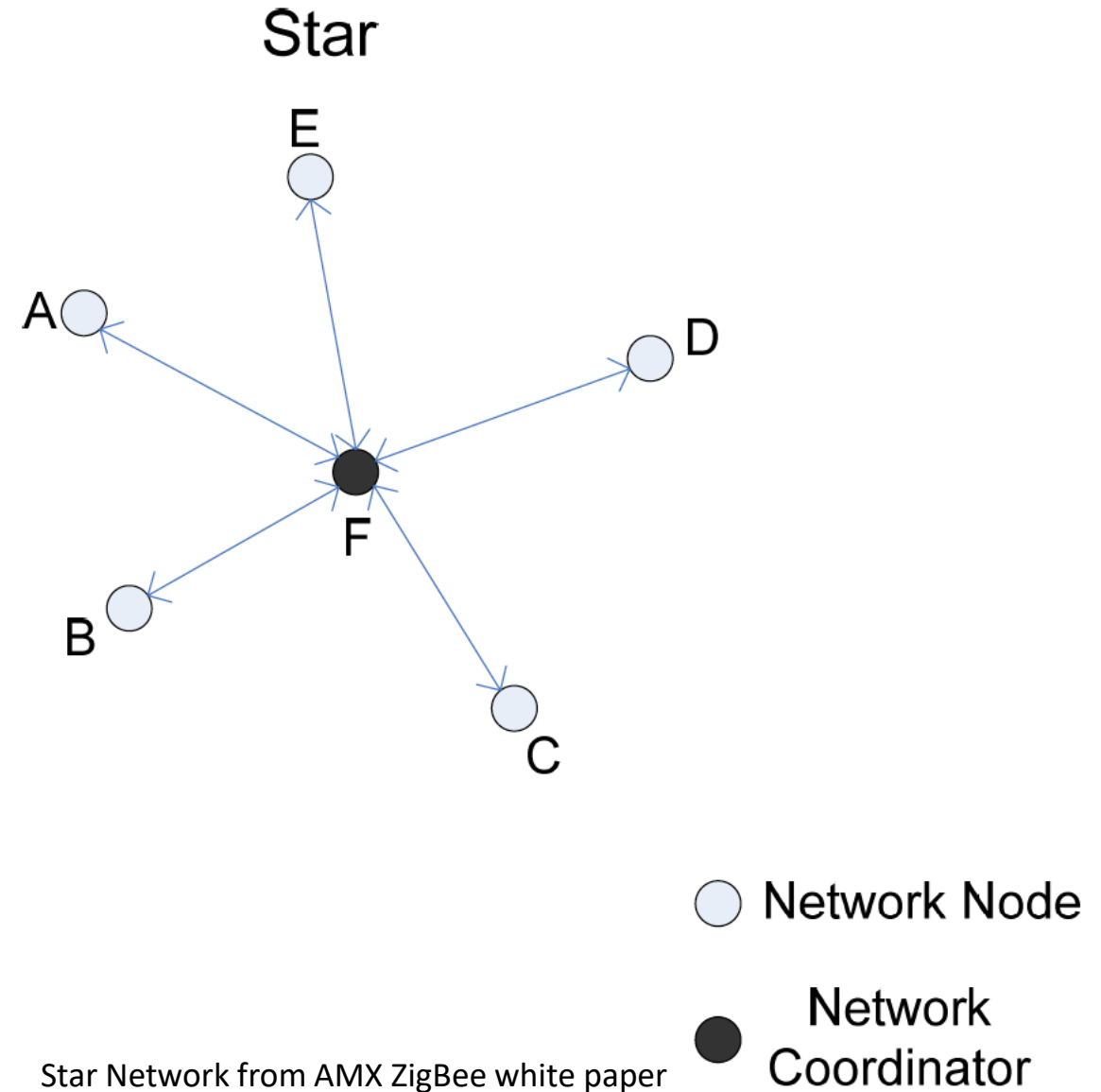
- To read a GPIO pin, direct register access can be used or emlib routine `GPIO_PinInGet`



- Setting a GPIO pin output:
  - Programming a one or “high” with `GPIO_PinOutSet`
  - Programming a zero or “low” with `GPIO_PinOutClear`

# Wireless Networks

- Infrastructure networks provide typically these functions
  - Bridge to other networks
  - Forwarding functions
  - Medium access control
- In Infrastructure Networks, communications typically goes from wireless nodes to a wireless access point
  - Star Network



Star Network from AMX ZigBee white paper

# Infrastructure Networks

- These networks are simpler due to the network functions are placed into the access point and the wireless clients can remain quite simple
- Collisions may occur if the medium access of the wireless nodes and the access point is not coordinated.
  - With the Access Point controlling medium access, no or very little collisions are possible
  - A useful feature for maintaining and controlling Quality of Service (QoS)
- Infrastructure wireless networks lose some of the flexibility of wireless networks due to their reliance on the infrastructure.
  - Single Point of failure
  - Cannot be set up quickly – infrastructure must be in place

# WiFi is an example of a Infrastructure Network

- Access point is required to **coordinate medium access**
- Access point to **bridge** to other networks
- Access point to **forward** packets upstream and downstream
- Coordinates **Quality of Service**
  - Audio
  - Video
  - Games, etc.
- **Star Network**
  - Wireless clients cannot communicate with each other directly



# Wireless LAN advantages:

- Flexibility
  - Within radio range coverage, nodes can communicate without further restriction
  - Radio waves can penetrate walls, senders and receivers can be placed anywhere within radio coverage
- Design
  - Wireless networks allow for the design of small, independent devices
  - Cables not only restrict users access to the network, but to the physical design of the device
- Robustness
  - Wireless networks can survive disasters such as a cellular network providing services while the wired network of a building is down
- Cost
  - After the cost of installing an access point, adding additional users does not increase the cost

# Wireless LAN disadvantages

- Quality of Service
  - Typically WLANs offer lower quality than their wired counterparts due to lower bandwidth limitations of radio transmissions
- Restrictions
  - All wireless products have to comply with national and potentially international regulations
- Safety and Security
  - Radio waves for data transmission may interfere with other high-tech equipment in hospitals or radar installations
  - Open radio interfaces make eavesdropping much easier than wired LAN such as fiber optics

# Wireless LAN design goals

- Global operations
  - Mobile product can be taken from one country to another and should be made to operate legally in each country
- Low Power
  - WLAN clients are typically mobile and run on batteries. The WLAN design should take into account the requirements of low power devices
- License-free operation
  - WLAN operators do not want to apply for a license to use their equipment
- Robust transmission technology
  - Radios must be able to operate in potentially “noisy” RF environments such in a home with a hairdryer, vacuum cleaner, or RF obstacle

# Wireless LAN design goals (continued)

- Simplified spontaneous cooperation
  - Should not require complicated setup routines, but should operate spontaneously after power-up
- Easy to use
  - These WLANs should not require complex management, but rather work in a “plug-and-play” concept
- Protection of investment
  - Huge investment has been made into wired LAN for performance, reliability, and security
  - Wireless LANs should protect this investment by bridging their wireless networks onto the wired networks

# Wireless LAN design goals (continued)

- Safety and Security
  - Wireless products should have radios that are safe to be used with people and sensitive equipment
  - Encryption mechanisms should be integrated into the wireless network to provide privacy of data
- Transparency for applications
  - Existing applications should work in both wired and wireless LAN environments
  - In the wireless LAN environment, there could be higher latency and lower bandwidth available to the application