ECEN 5823 Spring 2025
Assignment 2 - Managing Energy Mode Assignment

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

Note: This assignment will begin with the completed Assignment #1 - Simplicity Studio Exercise Assignment.

Instructions:

1. Start by creating your assignment repository using the assignment link at https://classroom.github.com/a/IA0OC_V6. You will not need to clone this from the GitHub repo you just created into a new local directory since you will be starting with your code from the previous assignment. Instead, follow these instructions and run these commands with git bash within your assignment directory. Ensure you are setup to use SSH keys before proceeding, see the instructions from Github at https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/ and be sure to use Git Bash For Windows or Cygwin if using a Windows environment. (Cygwin .ssh directory will be different than Git Bash.)

    a. On your local PC, duplicate the folder created in Assignment #1 to a new folder, name it something like ecen5823-assignment2-<username> where <username> is your GitHub username.
    b. Change your current directory to this new folder ecen5823-assignment2-<username> and execute the following git commands.
    c. git remote remove origin
    d. git remote add origin <url>
        i. Where <url> is the URL for the repository created with the link above
        ii. This adds the new submission repository created in the link above as your new origin.
    e. git push origin master
    f. Import ecen5823-assignment2-<username> into Simplicity Studio
        i. Remember to perform a **Clean…** immediately after you import the project into Simplicity Studio.
2. Develop timer code to configure a periodic timer with interrupt. This timer will be used to turn an LED on/off with a fixed period, configured via #define macros, and will operate in two possible modes, a low power mode (used with EM3 sleep state) and higher power modes (EM0/EM1/EM2).

a. The timer code shall use LETIMER0
b. LETIMER0 shall use one of two possible oscillators, depending on configuration
    i. In low power mode (EM3) the ULFRCO (1 KHz) should be used.
    ii. In higher power modes (EM0/EM1/EM2) the LFXO (32.768 KHz) should be used.
c. The default setting for the interrupt handler timing should configure the interrupt for the **LED0** on time of **175ms** with a period of **2.25 seconds**. **Use this setting for your measurements in the questions file.**
d. Create a #define statement in your app.h code to set:
    i. Lowest allowed EM mode, #define LOWEST_ENERGY_MODE
        1. Support setting this #define to either EM0, EM1, EM2 and EM3 (respectively 0, 1, 2, and 3). You will make 4 runs of your code, 1 run for each of these energy mode settings.
e. Create #define statements in your timers.h code to set:
    i. The LED on time in milliseconds, #define LETIMER_ON_TIME_MS
    ii. Total blink period in milliseconds, #define LETIMER_PERIOD_MS
f. The settings of these 3 #defines should control oscillator selection, LETIMER0 clock frequencies and the values used to load into the LETIMER0 registers.

3. Based on the #define values above, your code will have to set APP_IS_OK_TO_SLEEP to false for EM0, and to true for EM1, EM2 and EM3. Leave APP_SLEEP_ON_ISR_EXIT set to SL_POWER_MANAGER_IGNORE for now. In your app_init() function you will have to add a power requirement if the desired energy mode is EM1 or EM2. Do not add any power requirement to run in either EM0 or EM3. See routine sl_power_manager_add_em_requirement().

Questions:
Answer the questions within the questions directory for assignment 2 (Assignment2-ManagingEnergyModes.md) and include with your submission.

Deliverables:

- Submission via **github classroom** and your repository setup via the classroom link at above. Use the following git command to submit your assignment:
  git push origin master
- Verify your ecen5823-assignment2-<username> repository contains your latest/ intended code and answers to the questions. It is your responsibility to push the correct version of your project/code. Use a web browser to verify the correct version of your project/code has been uploaded to GitHub.
- In **Canvas**, submit the URL to your github repository and in **GitHub** create a tag of your GitHub submission prior to the due date with the text of A2-final. The date and time of the tag creation is what we use to determine whether your submission was on time or late. These 2 steps will complete your assignment submission.

Approach/Guidance:

Develop code that uses LETIMER0 to generate periodic interrupts. Get your oscillator and LETIMER0 interrupt code running first in EM0.

1. Create a single #define label that will set the lowest energy mode allowed, example: #define LOWEST_ENERGY_MODE 0
    a. This single #define value shall control the lowest energy mode entered/allowed AND select which oscillator is used to drive the clock for LETIMER0 AND determining what values to load into the LETIMER0 control registers.
    b. Put these new #defines in app.h.

Due to issues (bugs?) in the Simplicity Studio code editor, please define your LOWEST_ENERGY_MODE this way:

```
// Only define 1 of these to define the lowest energy mode
// 0 = highest energy mode, 3 = lowest energy mode
#define LOWEST_ENERGY_MODE 0
//#define LOWEST_ENERGY_MODE 1
//#define LOWEST_ENERGY_MODE 2
//#define LOWEST_ENERGY_MODE 3
```

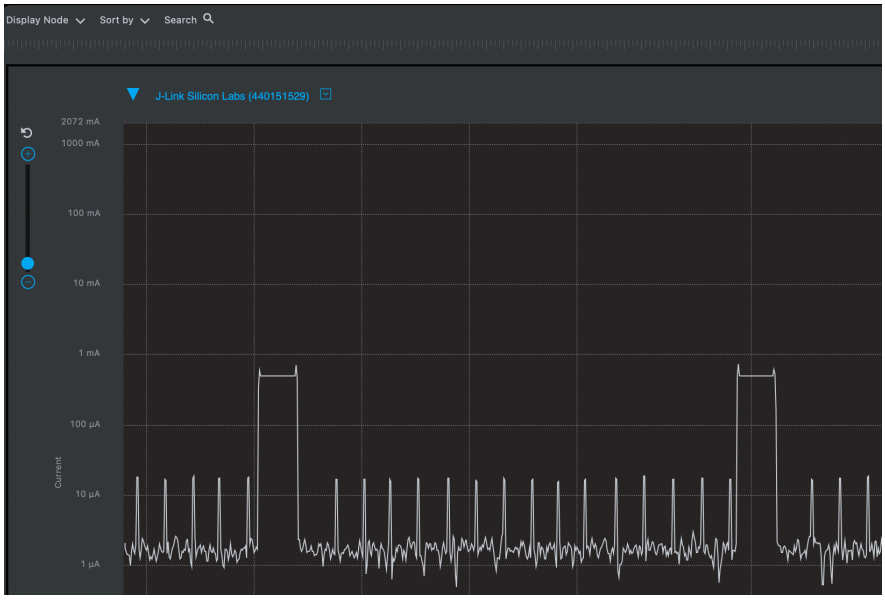
2. Create a new pair of files to configure oscillators and clock trees (all EMLIB CMU_***() function calls shall be in this file) called: oscillators.c/.h.
3. Create a new pair of files to configure timers (all LETIMER0 functions that are not interrupt service routines shall be in this file) called: timers.c/.h.
4. Create a pair of files to hold all of the code for interrupt service and related routines called: irq.c/.h.
5. After you get EM0 working in the Energy profiler, then proceed to debug EM1, EM2 and EM3.

Order of development:

1. Develop code to configure oscillators, clock trees and get the timer running without interrupts. After you believe you have the timer running, write some test code that reads the timer value to confirm it is running. Test both oscillators, perhaps use an LED to see that your code is functioning as you expect. There are routines you can use to read the frequency of the LETIMER0 clock.
2. Develop your IRQ service routine for LETIMER0 and configure the timer to generate interrupts. Test that code using both oscillators.
3. Lastly, develop the code that places the MCU into each of the 4 energy states, once for each energy mode per compile.
4. You will run your code 4 times, once in each energy mode and make current measurements using the Energy Profiler. Your steps are:

```
for (EM=0; EM<4; EM++) {
  Set LOWEST_ENERGY_MODE #define
  Compile code
  Run with energy profiler
  Measure results
  Answer A2 questions in the questions directory
}
```

Here is a screen shot of my implementation for EM3



Files in your src/ directory should look like (not showing some starter code files):