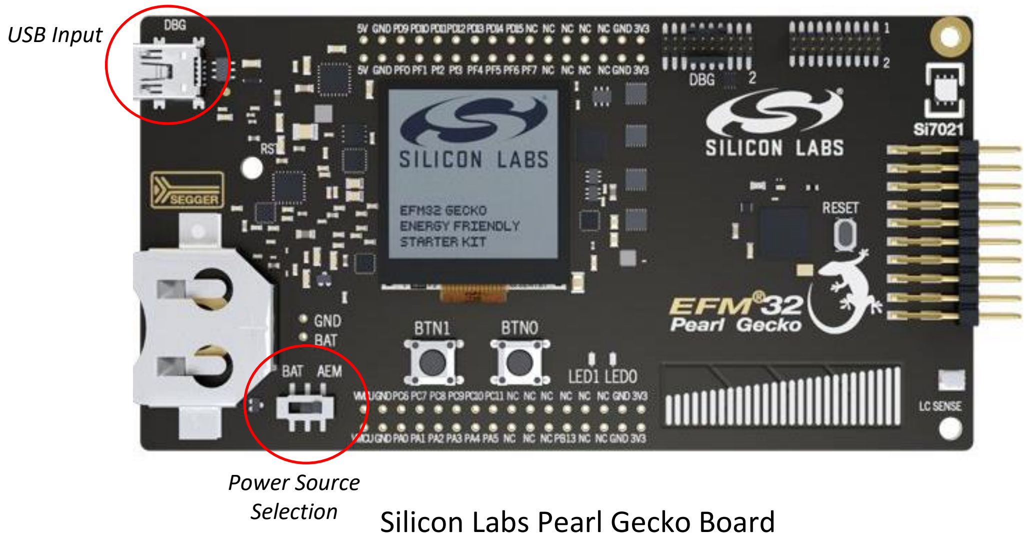
ECEN 2370f
Embedded Software Engineering Lab #1
Simplicity Exercise
Fall 2020

Objective: Install and become familiarized with the Silicon Labs' Simplicity Studio development environment

Due: Sunday, August 30th, 2020 at 11:59pm

I- Installing Simplicity and Importing the example project:

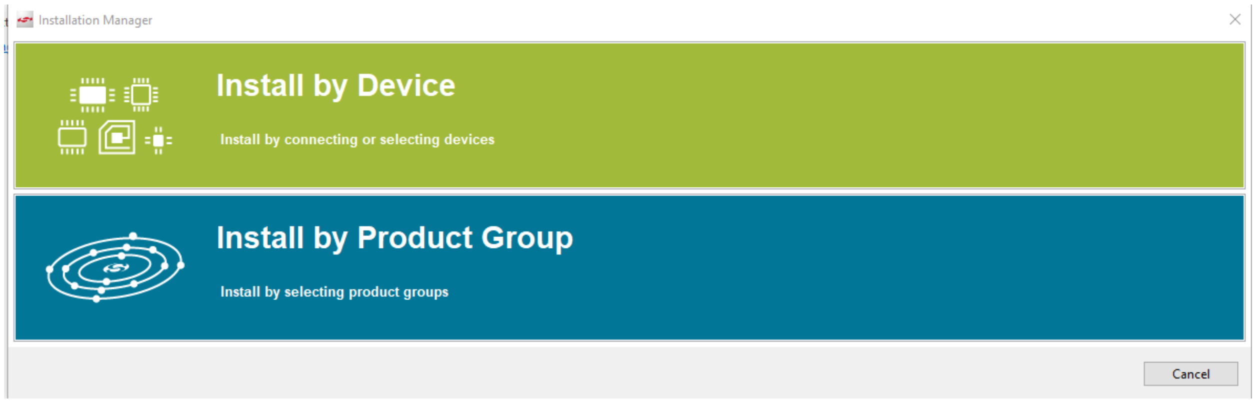
1. Connect your Silicon Labs Pearl Gecko to your computer via mini USB cable



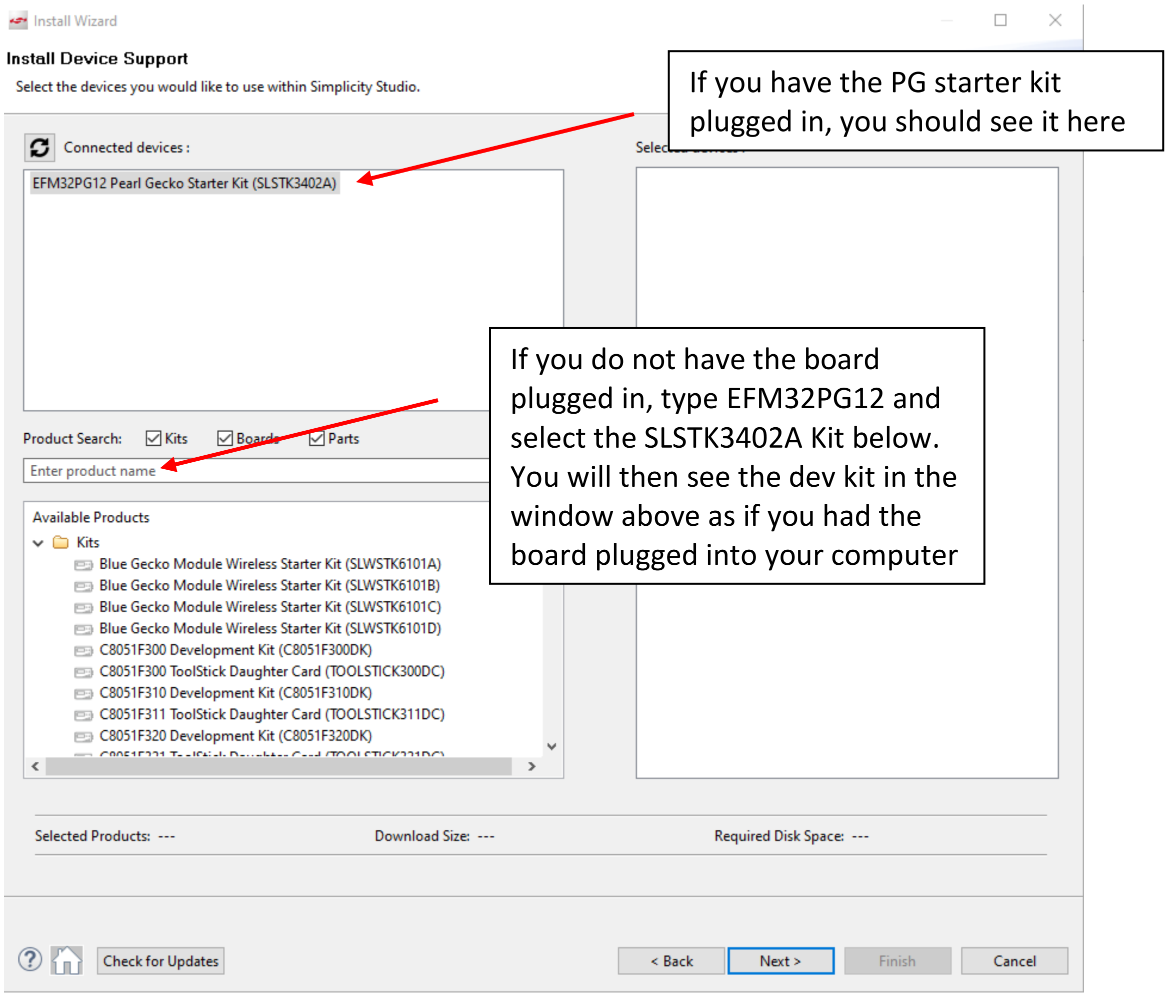
Silicon Labs Pearl Gecko Board

2. Ensure that the Power Source Select is to the DBG or AEM position

3. Download and Install Silicon Labs' Simplicity Studio 4 development environment. You can download the software from the following site and it will be easiest if you have your Pearl Gecko Start Kit plugged into your computer.
    - https://www.silabs.com/products/development-tools/software/simplicity-studio
    - You will be asked to create a Silicon Labs account
    - If you have your Pearl Gecko Starter Kit plugged into your computer, you should get a popup screen on how to install. Select by "Install by Device."


Installation Manager
Install by Device
Install by connecting or selecting devices
Install by Product Group
Install by selecting product groups
Cancel

- In the next popup, you should see a screen like the following.


Install Wizard
Install Device Support
Select the devices you would like to use within Simplicity Studio.
Connected devices :
EFM32PG12 Pearl Gecko Starter Kit (SLSTK3402A)
Selected devices :
Product Search: Kits Boards Parts
Enter product name
Available Products
Kits
Blue Gecko Module Wireless Starter Kit (SLWSTK6101A)
Blue Gecko Module Wireless Starter Kit (SLWSTK6101B)
Blue Gecko Module Wireless Starter Kit (SLWSTK6101C)
Blue Gecko Module Wireless Starter Kit (SLWSTK6101D)
C8051F300 Development Kit (C8051F300DK)
C8051F300 ToolStick Daughter Card (TOOLSTICK300DC)
C8051F310 Development Kit (C8051F310DK)
C8051F311 ToolStick Daughter Card (TOOLSTICK311DC)
C8051F320 Development Kit (C8051F320DK)
Selected Products: ---
Download Size: ---
Required Disk Space: ---
Check for Updates
< Back
Next >
Finish
Cancel

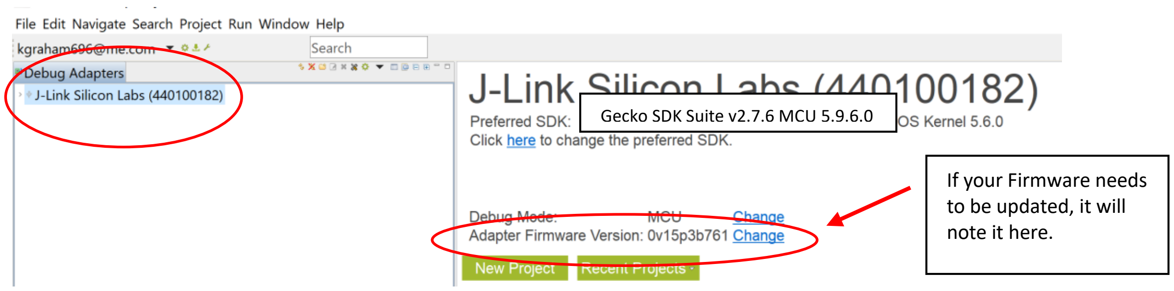
If you have the PG starter kit plugged in, you should see it here

If you do not have the board plugged in, type EFM32PG12 and select the SLSTK3402A Kit below. You will then see the dev kit in the window above as if you had the board plugged into your computer

- Double click the SLSTK3402A kit in the upper left window and it will then appear in the right window. Click on Next and follow the remaining prompts to install the software.

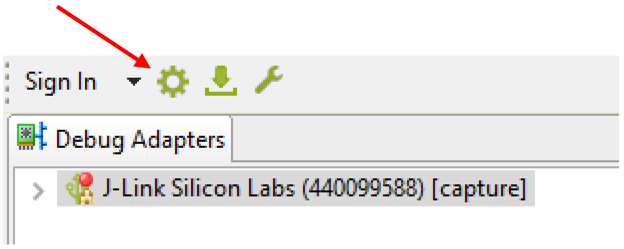
4. YOU WILL NEED YOUR Pearl Gecko dev kit plugged in for the next two steps to ensure that all the proper software was installed into Simplicity Studio.

   a. Make sure the J-Link or adapter firmware is up to date or installed. It may require a two-step process of download and then click install. You can determine and execute the download by being in the launcher perspective and clicking on the J-Link symbol to the left and check the middle of the screen whether the adapter is installed and to the latest revision. You will know if you do not have the latest firmware by seeing a prompt to download or install the latest version. If you do not have the latest version, download, and install the update. The picture below is what it should look like when you complete the operation.
      i. Please note that the latest version has changed since this screen shot


File Edit Navigate Search Project Run Window Help
kgraham696@me.com
Search
Debug Adapters
J-Link Silicon Labs (440100182)
J-Link Silicon Labs (440100182)
Preferred SDK:
Gecko SDK Suite v2.7.6 MCU 5.9.6.0
OS Kernel 5.6.0
Click here to change the preferred SDK.
Debug Mode: MCU Change
Adapter Firmware Version: 0v15p3b761 Change
New Project
Recent Projects

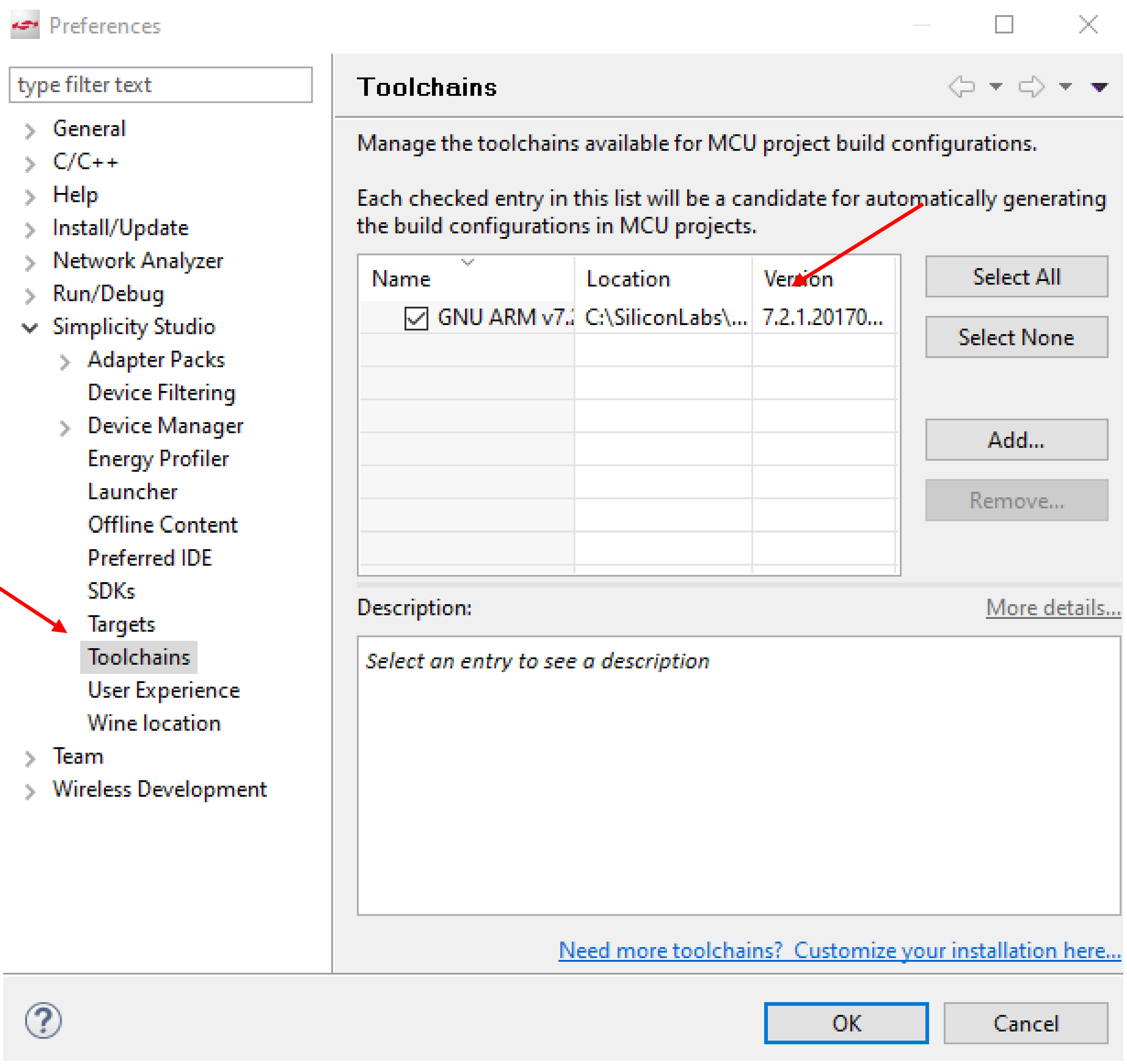
If your Firmware needs to be updated, it will note it here.

   b. To ensure that the proper toolchain is installed, GNU toolchain v7.2.1, go to the upper left window and click on the gear icon to open the Simplicity Preference Window.



   c. In the Simplicity Preference window, select toolchain in the left window and validate that it shows the most recent GNU toolchain, GNU toolchain v7.2.1.

   d. To ensure that all the software toolchains are downloaded and installed, find “update software” with your board installed and select install by device. Select your kit to move it to the right side when asked to select the device. Once you get to the screen to select the different software packages, scroll down to the bottom and if you use see packages for GNU toolchain v7.2.1, click on it to install.


Preferences

type filter text

- General
- C/C++
- Help
- Install/Update
- Network Analyzer
- Run/Debug
- Simplicity Studio
  - Adapter Packs
  - Device Filtering
  - Device Manager
  - Energy Profiler
  - Launcher
  - Offline Content
  - Preferred IDE
  - SDKs
  - Targets
  - Toolchains
  - User Experience
  - Wine location
- Team
- Wireless Development

Toolchains

Manage the toolchains available for MCU project build configurations.

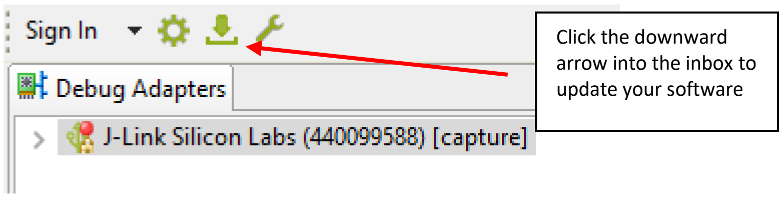
Each checked entry in this list will be a candidate for automatically generating the build configurations in MCU projects.

Name	Location	Version
☑ GNU ARM v7.2	C:\SiliconLabs\...	7.2.1.20170...

Description:

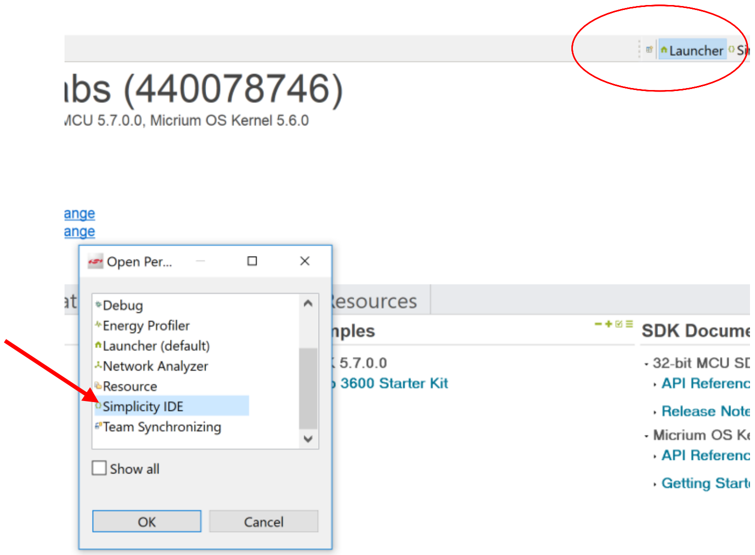
Select an entry to see a description

e. If the toolchain GNU toolchain v7.2.1, is not present, you will need to update your software. Find "update software" with your board installed and select install by device. Select your kit to move it to the right side when asked to select the device. Once you get to the screen to select the different software packages, scroll down to the bottom and if you use see packages for GNU toolchain v7.2.1, click on it to install.


Sign In
Debug Adapters
J-Link Silicon Labs (440099588) [capture]
Click the downward arrow into the inbox to update your software

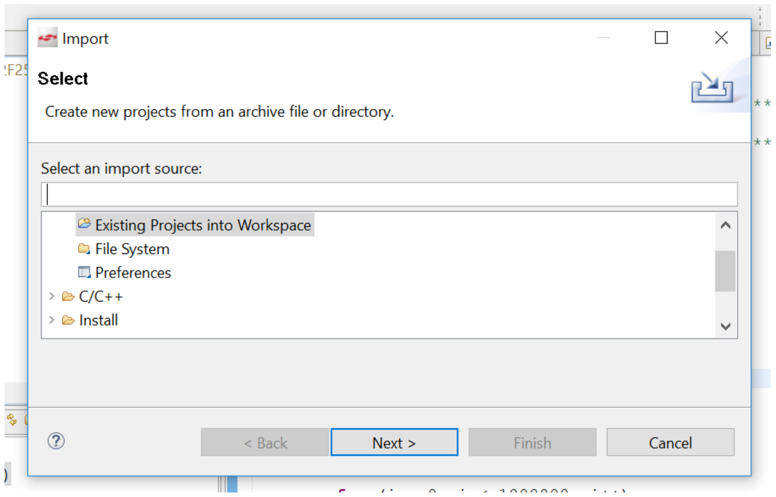
5. Download the example project from Canvas in the Lab 1 folder -> “Pearl_Blinky_F2020.zip”

6. When you open Simplicity, you can open the Simplicity IDE by clicking on the menu just to the left of the Launcher selection on the menu bar.



7. Now go into the Simplicity IDE by clicking on Simplicity IDE in the Menu bar

8. Select Import under File and then select “More Import Options” -> “Existing Projects into Workspace” and click on Next



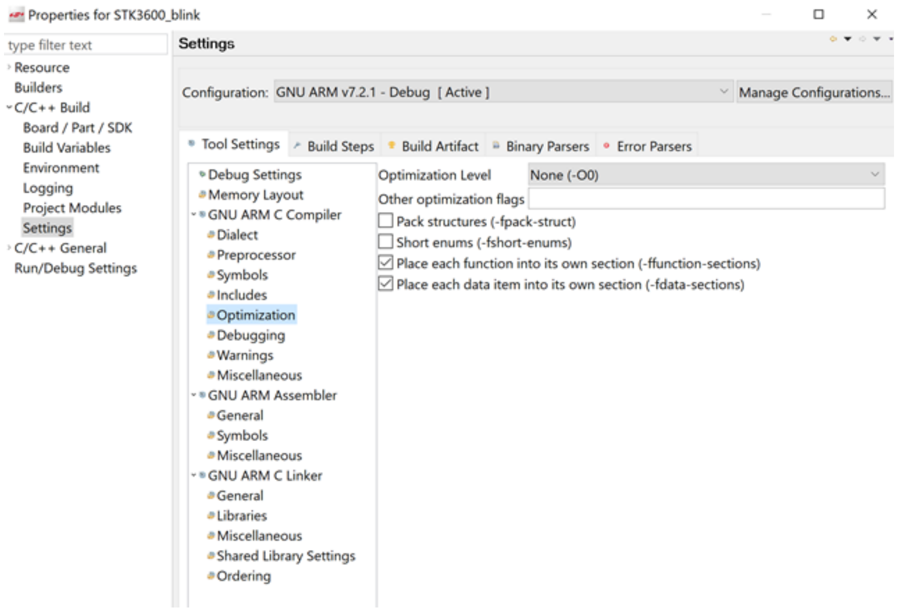
9. Select the “Select Archive File” and browse to select the downloaded archive project file for this lab “Pearl_Blinky_F2020.zip”. Then click on Finish. Now, the demo / example project should be loaded into your workspace.

10. Change the name of the project by adding your two initials in front of the project name Pearl_Blinky.zip. You can rename your project by right clicking the project while in the Simplicity IDE perspective and select rename
    a. Ex. For Keith Graham, the name would change to KG_Pearl_Blink.zip

II- Changing Code Optimization Level:

1. Right click the project name and scroll all the way down and open Properties

2. Within properties, select Settings within C/C++ Build and set the Optimization Level to "None (-O0)" for Windows and "None" for MACs.
    a. Do not input anything in "Other Debug Flags"



Configuration: GNU ARM v7.2.1 - Debug [Active]

Optimization Level: None (-O0)

Other optimization flags:

☐ Pack structures (-fpack-struct)

☐ Short enums (-fshort-enums)

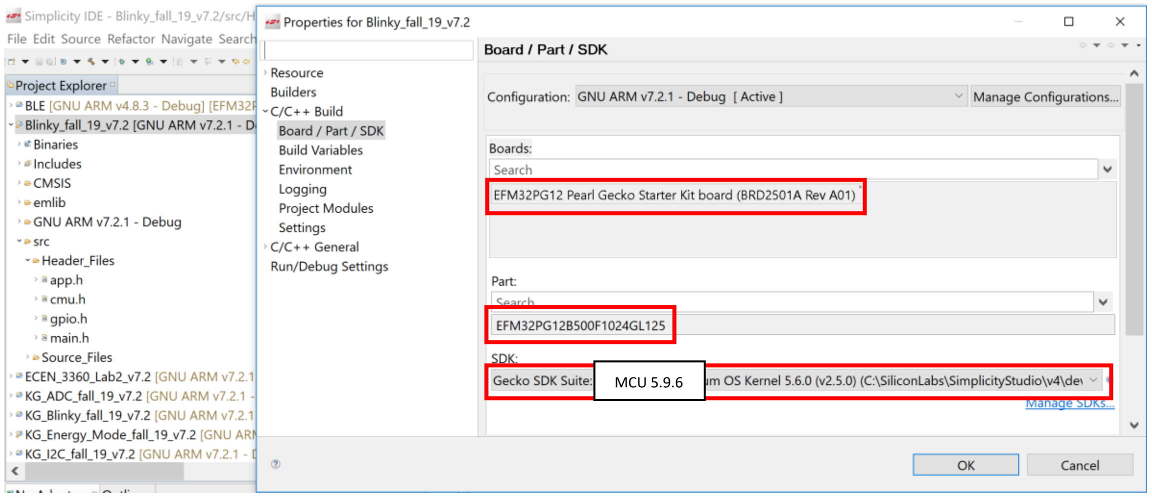
☑ Place each function into its own section (-ffunction-sections)

☑ Place each data item into its own section (-fdata-sections)

III- Verify board SDK – MCU 5.9.6.0

1. As you did previously to check and possibly change the optimization setting, right click your project and then click on Board / Part / SDK.

2. Verify that the setting's "snapshot" below is similar to your configuration. The SDK should be Gecko SDK Suite: MCU 5.9.6.0.


Simplicity IDE - Blinky_fall_19_v7.2/src/H
File Edit Source Refactor Navigate Search
Project Explorer
BLE [GNU ARM v4.8.3 - Debug] [EFM32
Blinky_fall_19_v7.2 [GNU ARM v7.2.1 - D
Binaries
Includes
CMSIS
emlib
GNU ARM v7.2.1 - Debug
src
Header_Files
app.h
cmu.h
gpio.h
main.h
Source_Files
ECEN_3360_Lab2_v7.2 [GNU ARM v7.2.1
KG_ADC_fall_19_v7.2 [GNU ARM v7.2.1 -
KG_Blinky_fall_19_v7.2 [GNU ARM v7.2.1
KG_Energy_Mode_fall_19_v7.2 [GNU ARM
KG_I2C_fall_19_v7.2 [GNU ARM v7.2.1 -
Properties for Blinky_fall_19_v7.2
Resource
Builders
C/C++ Build
Board / Part / SDK
Build Variables
Environment
Logging
Project Modules
Settings
C/C++ General
Run/Debug Settings
Board / Part / SDK
Manage Configurations...
Search
Manage SDKs...
OK
Cancel

Configuration: GNU ARM v7.2.1 - Debug [Active]

Boards: EFM32PG12 Pearl Gecko Starter Kit board (BRD2501A Rev A01)

Part: EFM32PG12B500F1024GL125

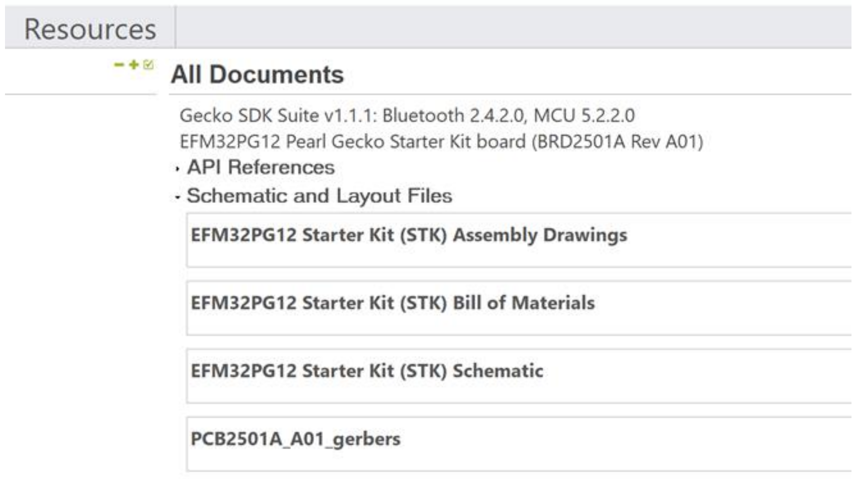
SDK: Gecko SDK Suite: MCU 5.9.6 m OS Kernel 5.6.0 (v2.5.0) (C:\SiliconLabs\SimplicityStudio\v4\dev

3. If your system does not match the above, search these parameters and select the item that matches the above.

4. The most likely item you may not find here is the SDK. If you cannot find MCU 5.9.6 by clicking on Manage SDKs, you will need to update your Simplicity to this SDK.
    a. To learn how to locate MCU 5.9.6 to download, it is the same process as the instruction to possibly fix your build issue by downloading a different GNU toolchain described further in this document. Instead of locating the GNU tools, scroll down to find the SDK files. You may need to click on view all previous versions as well.
    b. If you have any difficulty, please contact the instructing staff immediately.

IV- Finding and setting the LED Pins:

1. Go to the dev kit's schematic and find which pins of the Pearl Gecko are connected to the LED0 and LED1 of the dev kit. You can find the schematic in Simplicity from the Launcher perspective, under the Documentation Tab.

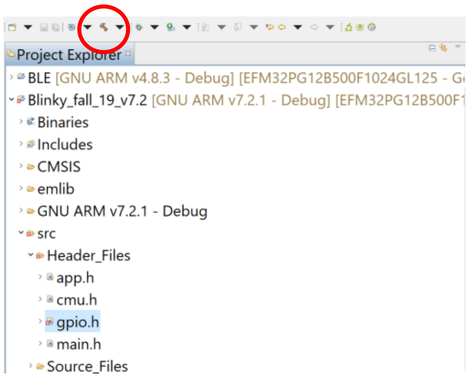


All Documents

Gecko SDK Suite v1.1.1: Bluetooth 2.4.2.0, MCU 5.2.2.0
EFM32PG12 Pearl Gecko Starter Kit board (BRD2501A Rev A01)
- API References
- Schematic and Layout Files
    - EFM32PG12 Starter Kit (STK) Assembly Drawings
    - EFM32PG12 Starter Kit (STK) Bill of Materials
    - EFM32PG12 Starter Kit (STK) Schematic
    - PCB2501A_A01_gerbers

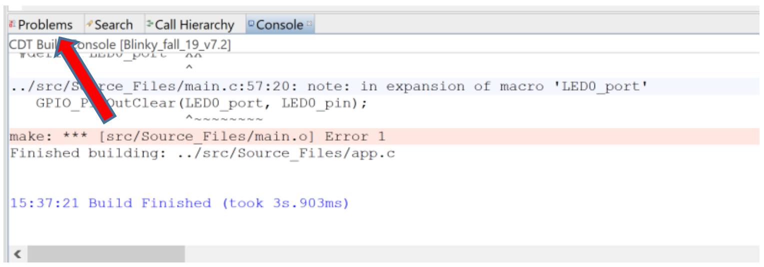
2. DEBUG HINT:

a. Now try to build your project by first selecting a file in your project and then clicking the build icon, the hammer, in the top menu bar



Project Explorer
BLE [GNU ARM v4.8.3 - Debug] [EFM32PG12B500F1024GL125 - G
Blinky_fall_19_v7.2 [GNU ARM v7.2.1 - Debug] [EFM32PG12B500F1
Binaries
Includes
CMSIS
emlib
GNU ARM v7.2.1 - Debug
src
Header_Files
app.h
cmu.h
gpio.h
main.h
Source_Files

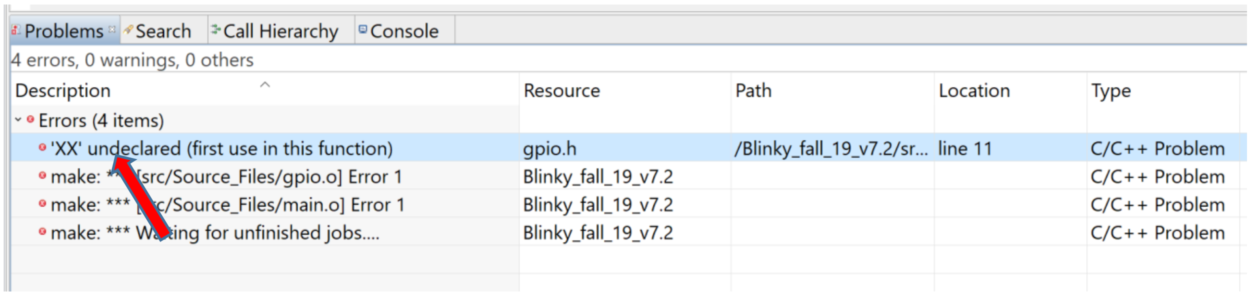
b. The build will fail, and you should see something similar to the below screen shot in the console window:



```
 #define LED0_port  XX
                    ^
../src/Source_Files/main.c:57:20: note: in expansion of macro 'LED0_port'
   GPIO_PinOutClear(LED0_port, LED0_pin);
                    ^~~~~~~~~
make: *** [src/Source_Files/main.o] Error 1
Finished building: ../src/Source_Files/app.c


15:37:21 Build Finished (took 3s.903ms)
```

c. To get to the compile error quickly, click on the Problem Tab to get to a list of all the make / build errors

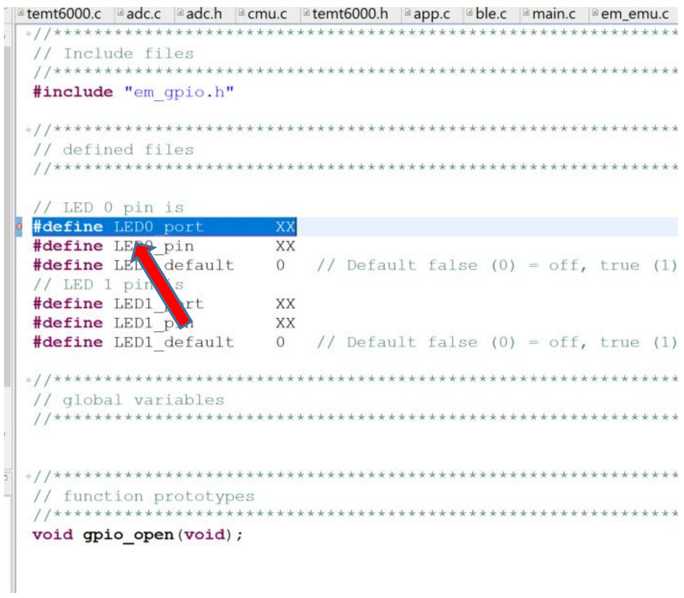
   i. You many need to expand "Errors" to see the list



4 errors, 0 warnings, 0 others

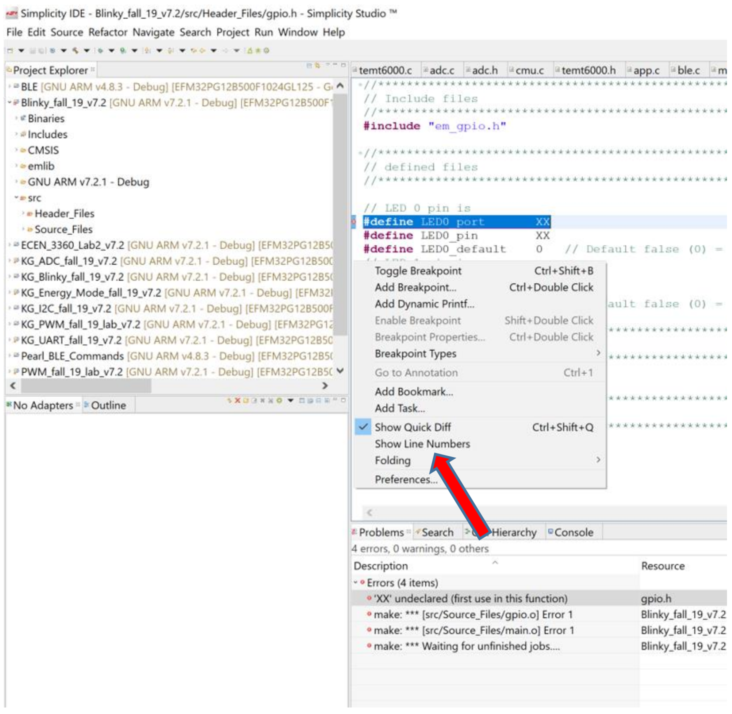
Description	Resource	Path	Location	Type
Errors (4 items)				
'XX' undeclared (first use in this function)	gpio.h	/Blinky_fall_19_v7.2/sr...	line 11	C/C++ Problem
make: *** [src/Source_Files/gpio.o] Error 1	Blinky_fall_19_v7.2			C/C++ Problem
make: *** [src/Source_Files/main.o] Error 1	Blinky_fall_19_v7.2			C/C++ Problem
make: *** Waiting for unfinished jobs....	Blinky_fall_19_v7.2			C/C++ Problem

d. For all the non "make" errors, you can double-click the error and it will open the "Resource" file and take you to the error



```
//*************************************************************
// Include files
//*************************************************************
#include "em_gpio.h"

//*************************************************************
// defined files
//*************************************************************

// LED 0 pin is
#define LED0_port          XX
#define LED0_pin           XX
#define LED0_default       0    // Default false (0) = off, true (1)
// LED 1 pin is
#define LED1_port          XX
#define LED1_pin           XX
#define LED1_default       0    // Default false (0) = off, true (1)

//*************************************************************
// global variables
//*************************************************************


//*************************************************************
// function prototypes
//*************************************************************
void gpio_open(void);
```

e. Sometimes it is helpful to know the line numbers in a file. To view line numbers in a file, right click the left edge/bar of the .c or .h file to open the menu. Select “Show Line Numbers.”

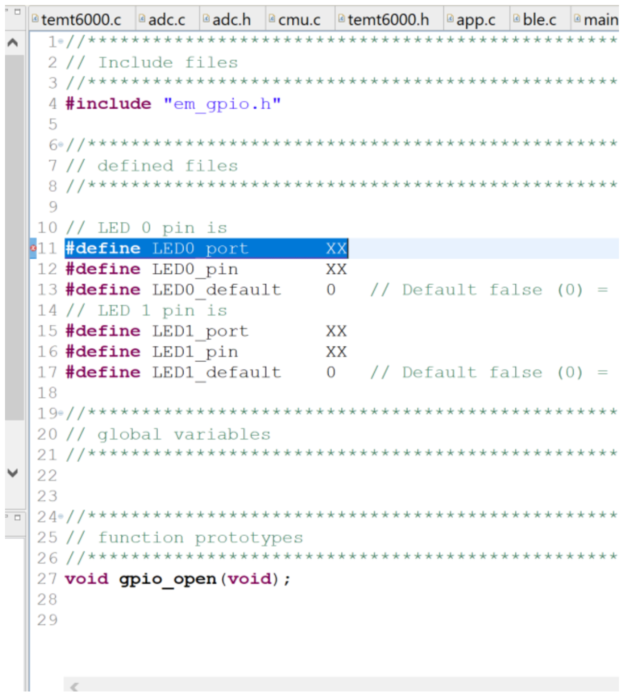

Simplicity IDE - Blinky_fall_19_v7.2/src/Header_Files/gpio.h - Simplicity Studio ™
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
BLE [GNU ARM v4.8.3 - Debug] [EFM32PG12B500F1024GL125 - G
Blinky_fall_19_v7.2 [GNU ARM v7.2.1 - Debug] [EFM32PG12B500F
Binaries
Includes
CMSIS
emlib
GNU ARM v7.2.1 - Debug
src
Header_Files
Source_Files
ECEN_3360_Lab2_v7.2 [GNU ARM v7.2.1 - Debug] [EFM32PG12B5
KG_ADC_fall_19_v7.2 [GNU ARM v7.2.1 - Debug] [EFM32PG12B50
KG_Blinky_fall_19_v7.2 [GNU ARM v7.2.1 - Debug] [EFM32PG12B5
KG_Energy_Mode_fall_19_v7.2 [GNU ARM v7.2.1 - Debug] [EFM32
KG_I2C_fall_19_v7.2 [GNU ARM v7.2.1 - Debug] [EFM32PG12B500F
KG_PWM_fall_19_lab_v7.2 [GNU ARM v7.2.1 - Debug] [EFM32PG12
KG_UART_fall_19_v7.2 [GNU ARM v7.2.1 - Debug] [EFM32PG12B50
Pearl_BLE_Commands [GNU ARM v4.8.3 - Debug] [EFM32PG12B5
PWM_fall_19_lab_v7.2 [GNU ARM v7.2.1 - Debug] [EFM32PG12B5
No Adapters
Outline
temt6000.c
adc.c
adc.h
cmu.c
temt6000.h
app.c
ble.c
Toggle Breakpoint Ctrl+Shift+B
Add Breakpoint... Ctrl+Double Click
Add Dynamic Printf...
Enable Breakpoint Shift+Double Click
Breakpoint Properties... Ctrl+Double Click
Breakpoint Types
Go to Annotation Ctrl+1
Add Bookmark...
Add Task...
Show Quick Diff Ctrl+Shift+Q
Show Line Numbers
Folding
Preferences...
Problems
Search
Hierarchy
Console

```
//*************************************************
// Include files
//*************************************************
#include "em_gpio.h"

//*************************************************
// defined files
//*************************************************

// LED 0 pin is
#define LED0_port          XX
#define LED0_pin           XX
#define LED0_default       0    // Default false (0) =
```

4 errors, 0 warnings, 0 others

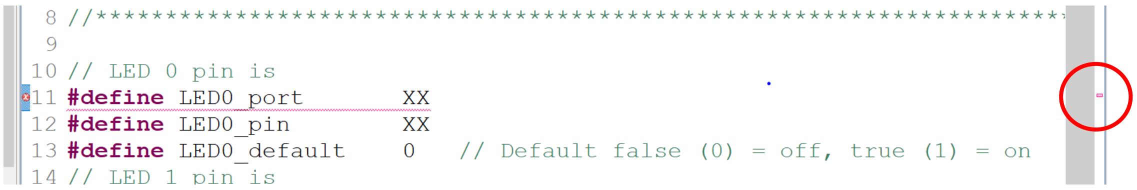
Description	Resource
Errors (4 items)	
'XX' undeclared (first use in this function)	gpio.h
make: *** [src/Source_Files/gpio.o] Error 1	Blinky_fall_19_v7.2
make: *** [src/Source_Files/main.o] Error 1	Blinky_fall_19_v7.2
make: *** Waiting for unfinished jobs....	Blinky_fall_19_v7.2

f. After selecting / clicking on “Show Line Numbers,” your view should now look like the below screen shot



```
//*****************************************************
// Include files
//*****************************************************
#include "em_gpio.h"

//*****************************************************
// defined files
//*****************************************************

// LED 0 pin is
#define LED0_port         XX
#define LED0_pin          XX
#define LED0_default      0    // Default false (0) =
// LED 1 pin is
#define LED1_port         XX
#define LED1_pin          XX
#define LED1_default      0    // Default false (0) =

//*****************************************************
// global variables
//*****************************************************


//*****************************************************
// function prototypes
//*****************************************************
void gpio_open(void);


```

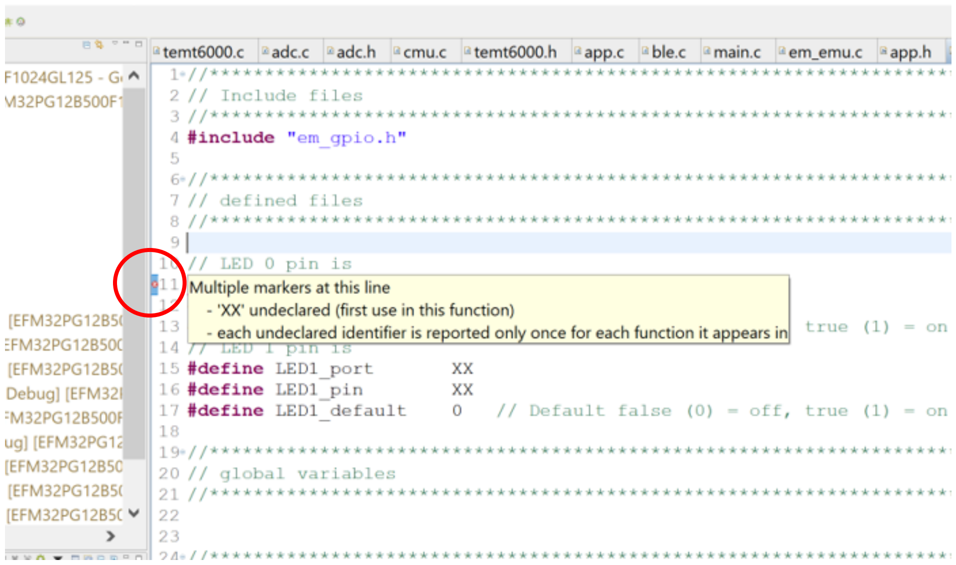
g. Here is a third way of locating a compile error. On the right side of the .c or .h file, if you see a "red dot," there is an error on the line of code. Scroll up or down until the error appears in the file window



```
//***********************************************************************

// LED 0 pin is
#define LED0_port         XX
#define LED0_pin          XX
#define LED0_default      0    // Default false (0) = off, true (1) = on
// LED 1 pin is
```

i. If you see a "yellow dot" instead of a "red dot," the yellow dot indicates a warning instead of a problem

h. To learn more about the error, you can hoover the cursor over the red circle with the white X to the left of the line of code to obtain a helpful popup window



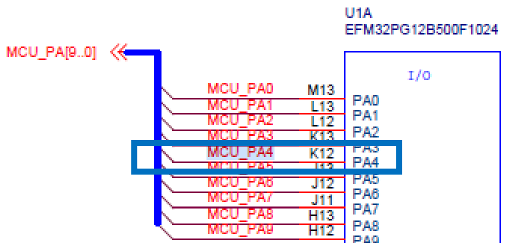
```
//*****************************************************************************
// Include files
//*****************************************************************************
#include "em_gpio.h"

//*****************************************************************************
// defined files
//*****************************************************************************

// LED 0 pin is
```

Multiple markers at this line
- 'XX' undeclared (first use in this function)
- each undeclared identifier is reported only once for each function it appears in

```
                                                          true (1) = on
// LED 1 pin is
#define LED1_port          XX
#define LED1_pin           XX
#define LED1_default       0    // Default false (0) = off, true (1) = on

//*****************************************************************************
// global variables
//*****************************************************************************


//*****************************************************************************
```

3. You can find the Port and Pin information for the LED0 and LED1 in several different documents:
   a. The Pearl Gecko Dev Kit Starter Kit documentation:
      i. UG257: EFM32 Pearl Gecko PG12 Starter Kit User's Guide
      ii. https://www.silabs.com/documents/public/user-guides/ug257-stk3402-usersguide.pdf
   b. The Pearl Gecko Dev Kit Schematic:
      i. You can locate the LED0 and LED1 devices using ctrl-F and then backtrack the signal back to the Pearl Gecko to find its Port and Pin information
      ii. https://www.silabs.com/documents/public/schematic-files/EFM32PG12-BRD2501A-A01-schematic.pdf

4. The Pearl Gecko pins are described with the letter P for Port, a letter for the port specified, and then the port pin numbers:
   a. For example, in the below schematic snapshot. The signal MCU_PA4 connects to the Pearl Gecko I/O port PA4



   b. PA4 represents the following:
      i. P = Port
      ii. A = A port whose firmware enumeration is gpioPortA
      iii. 4 = pin 4 of Port A

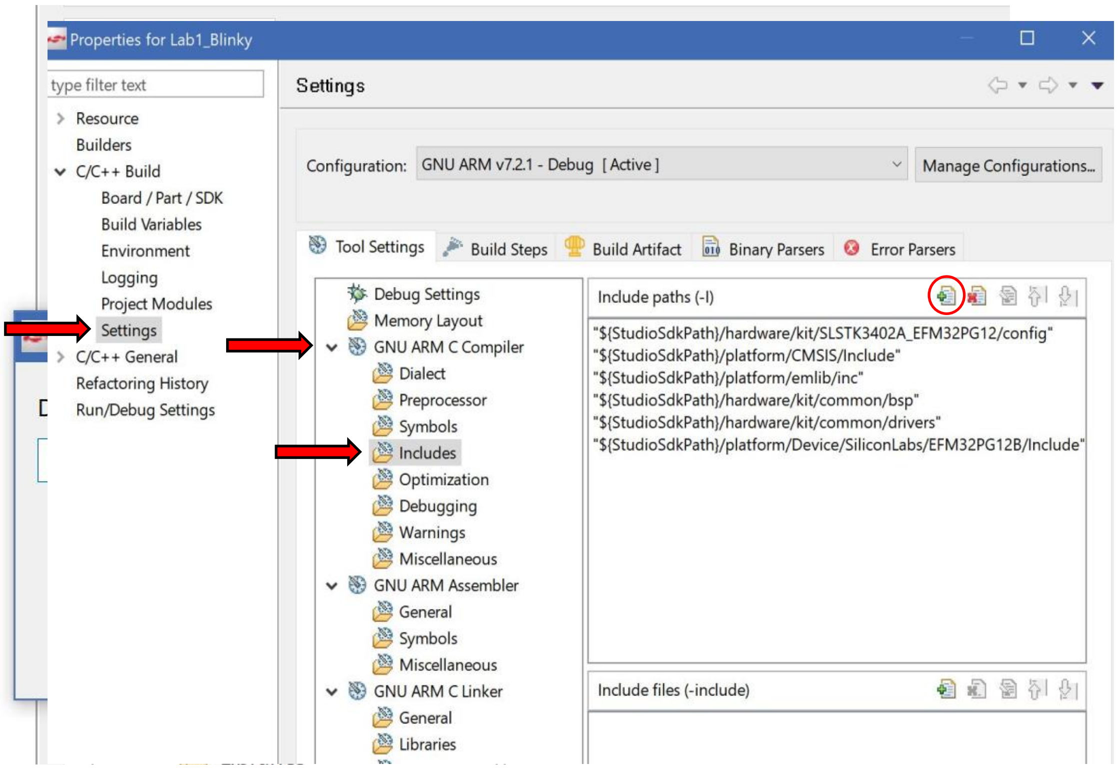
5. Once you find the Port and Pin of the LEDs, expand the Pearl_Blinky project and then open the /src/Header_Files folder. Next, open the brd_config.h file and complete the following #define statements by tracing the connection from the LEDs to the Pearl Gecko. **Ports are defined using the enumerations of gpioPortA, gpioPortB, etc.. Pins should be unsigned integers.**

• //LED 0 pin is		//Example: PZ23
• #define LED0_Port	XX	//Example: for port Z, use gpioPortZ
• #define LED0_Pin	XX	//Example: 23u (u = unsigned)
• #define LED0_Default	0	//Default false (0) = off, true (1) = on
• //LED 1 pin is		//Example: PX14
• #define LED1_Port	XX	//Example: for port X, use gpioPortX
• #define LED1_Pin	XX	//Example: 14u (u = unsigned)
• #define LED1_Default	0	// Default false (0) = off, true (1) = on

NOTE 1:
If you have header file inclusion errors, follow these steps:

1. Right click on your project and click on Properties
2. Expand C/C++ Build and click on Settings
3. Under GNU ARM C Compiler, click on Includes and then Add:

4. You should see this dialog box, click on Workspace:

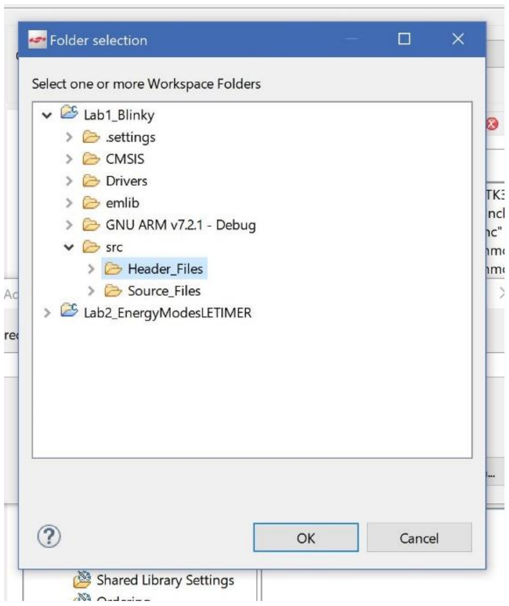


Configuration: GNU ARM v7.2.1 - Debug [Active]

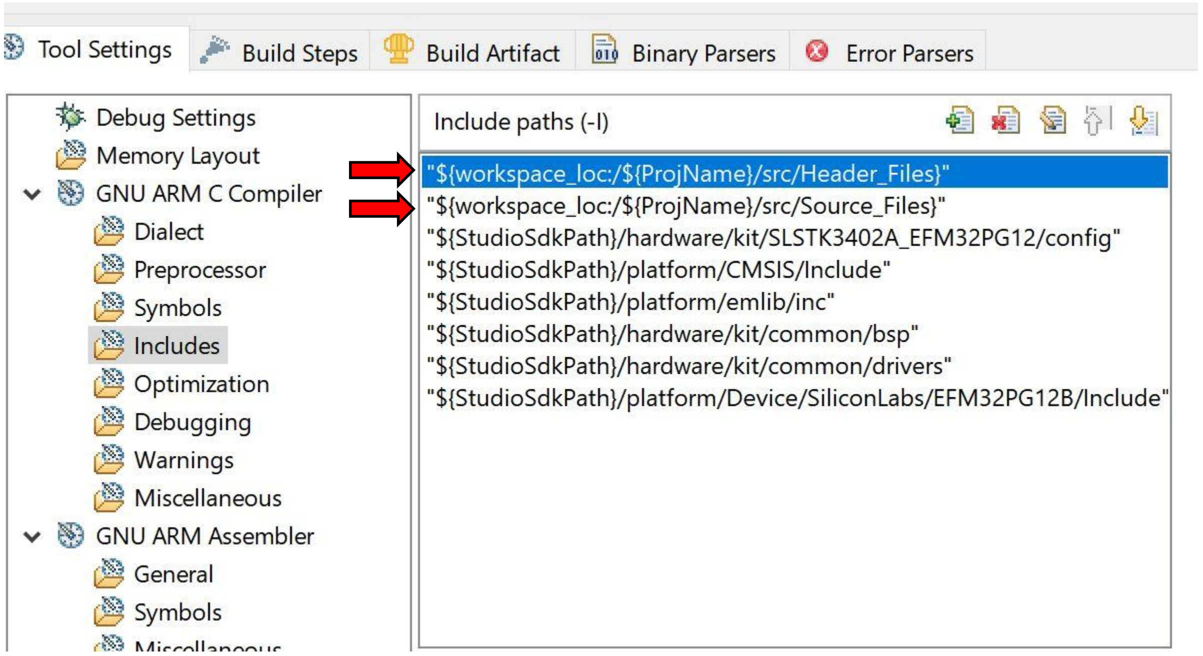
Include paths (-I)

```
"${StudioSdkPath}/hardware/kit/SLSTK3402A_EFM32PG12/config"
"${StudioSdkPath}/platform/CMSIS/Include"
"${StudioSdkPath}/platform/emlib/inc"
"${StudioSdkPath}/hardware/kit/common/bsp"
"${StudioSdkPath}/hardware/kit/common/drivers"
"${StudioSdkPath}/platform/Device/SiliconLabs/EFM32PG12B/Include"
```

5. Expand your project, and src. You should see Header_Files and Source Files



6. Click on Header_Files and then OK
7. Click on OK again to close the dialog box
8. Repeat the same steps to add the Source_Files folder
9. Your screen should look like this:



```
"${workspace_loc:/${ProjName}/src/Header_Files}"
"${workspace_loc:/${ProjName}/src/Source_Files}"
"${StudioSdkPath}/hardware/kit/SLSTK3402A_EFM32PG12/config"
"${StudioSdkPath}/platform/CMSIS/Include"
"${StudioSdkPath}/platform/emlib/inc"
"${StudioSdkPath}/hardware/kit/common/bsp"
"${StudioSdkPath}/hardware/kit/common/drivers"
"${StudioSdkPath}/platform/Device/SiliconLabs/EFM32PG12B/Include"
```

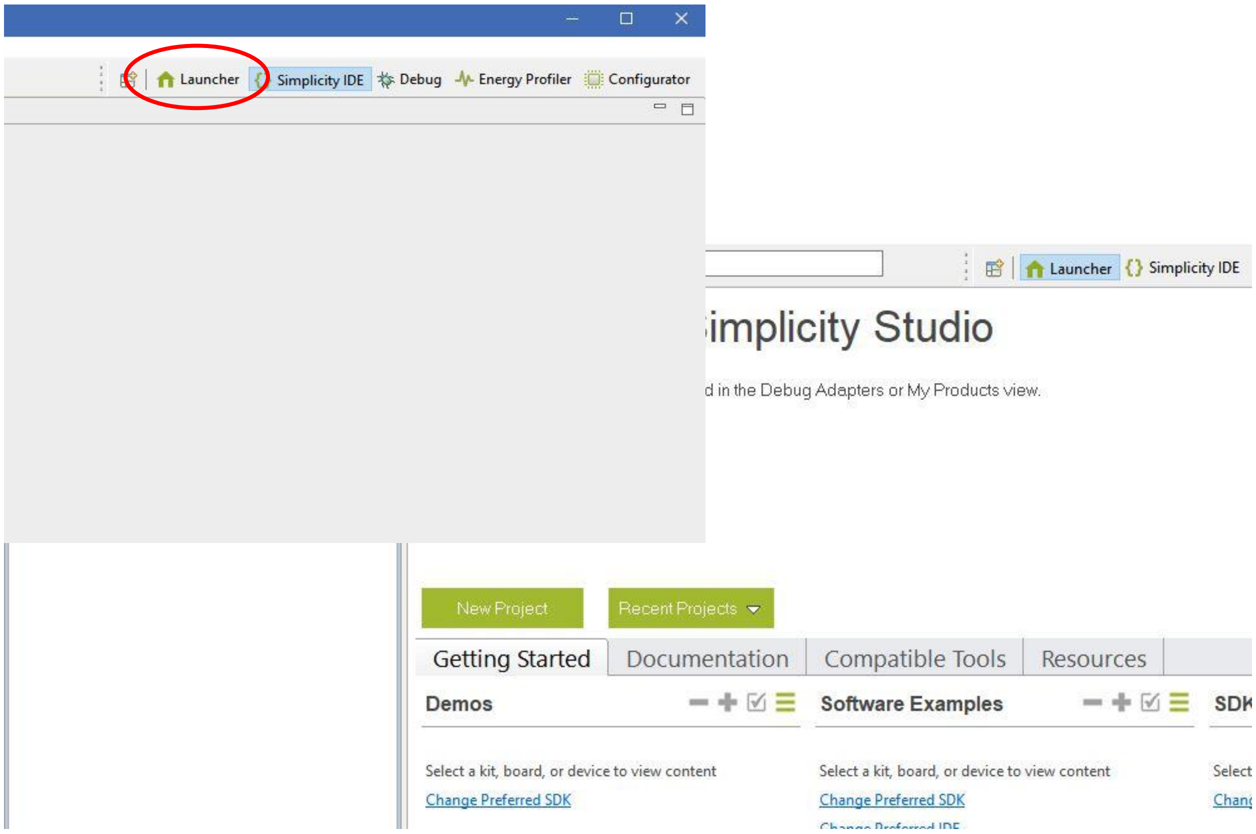
Do not perform the below instructions of adding the header and source includes under the assembler section. There are instructions how to remove them below.

a. Please check if you have included "Header_Files" and "Source_Files" <u>under GNU ARM Assembler</u> folders such as Includes. If yes, select these links and delete them.
   - Right click your project, and click on Properties
   - Expand C/C++ Build and click on Settings
   - Click on "GNU ARM Assembler"
     - Check each folder
     - If you have references to Header_Files and Source_Files, click on each one and then click on the delete icon found above it.
   - The Header_Files and Source_Files includes mentioned above should only be within the Settings "GNU ARM C Compiler/Includes" folder

10. Click on Apply then OK
11. Your project should now build without include errors

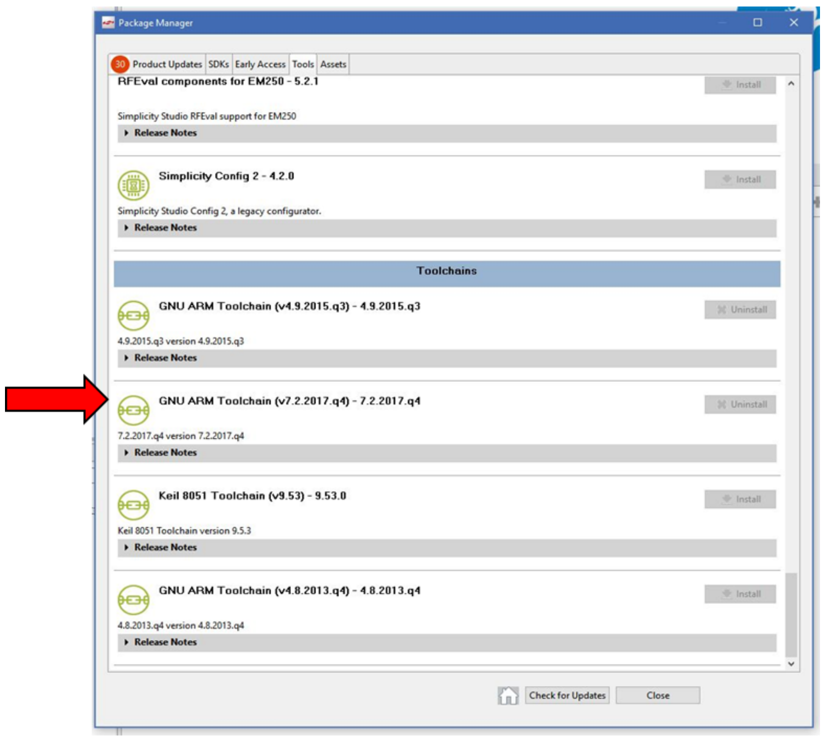
NOTE 2:
If you get a "MAKE" error or similar, you might be missing a toolchain:
To install the missing toolchain (such as GNU ARM v7.2.1) follow these steps

1. Click on the launcher view if you are not already on it:



At the top right, you should see an icon like the one shown below "Update Software":

3. Once the Package Manger window appears, click on the "Tools" tab and scroll towards the end. You should see a toolchains section



RFEval components for EM250 - 5.2.1

Simplicity Studio RFEval support for EM250

Simplicity Config 2 - 4.2.0

Simplicity Studio Config 2, a legacy configurator.

Toolchains

GNU ARM Toolchain (v4.9.2015.q3) - 4.9.2015.q3

4.9.2015.q3 version 4.9.2015.q3

GNU ARM Toolchain (v7.2.2017.q4) - 7.2.2017.q4

7.2.2017.q4 version 7.2.2017.q4

Keil 8051 Toolchain (v9.53) - 9.53.0

Keil 8051 Toolchain version 9.5.3

GNU ARM Toolchain (v4.8.2013.q4) - 4.8.2013.q4

4.8.2013.q4 version 4.8.2013.q4

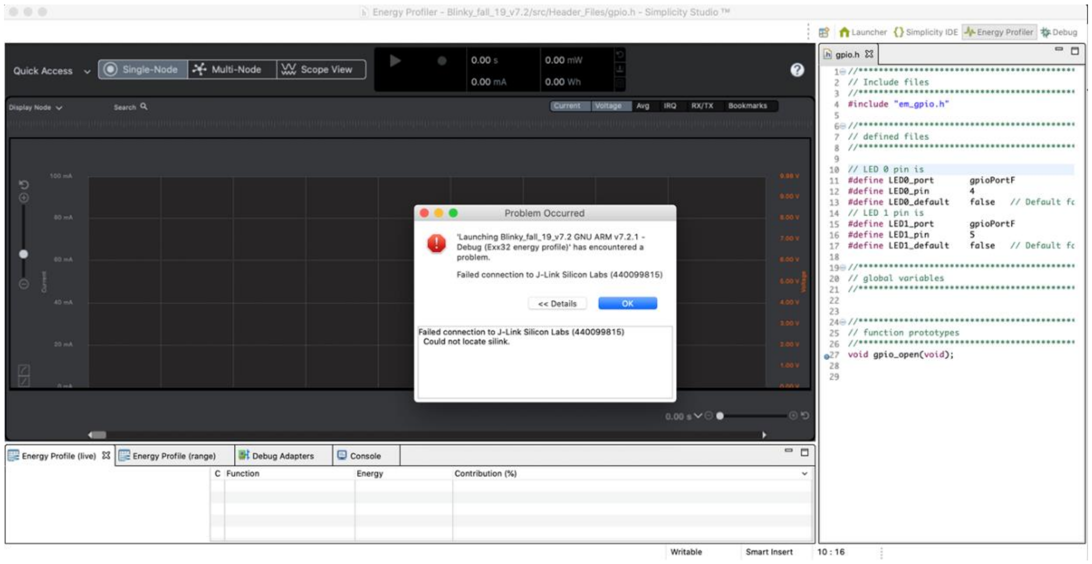
4. Find GNU ARM Toolchain v7.2 and simply click on Install if it is not already installed:

5. That's it! Your project should now build without errors.

V- Building and Profiling:

1. Now, click on build to build the project and under the Run pull down menu, select "Profile."
    b. Simplicity IDE should begin to compile the project, and then download and flash the code onto the Pearl Gecko dev kit
    c. After flashing the microcontroller, the code should begin to run, and the LEDs on the dev kit should begin to flash
    d. You should always power cycle your board after programming to reset the Energy profiler by switching the power switch from AEM/DBG to BAT and back

If you are using a MacBook and received the below screenshot error message while running the Energy Profiler, please refer to the Appendix at the end of this Lab Assignment.



'Launching Blinky_fall_19_v7.2 GNU ARM v7.2.1 - Debug (Exx32 energy profile)' has encountered a problem.

Failed connection to J-Link Silicon Labs (440099815)

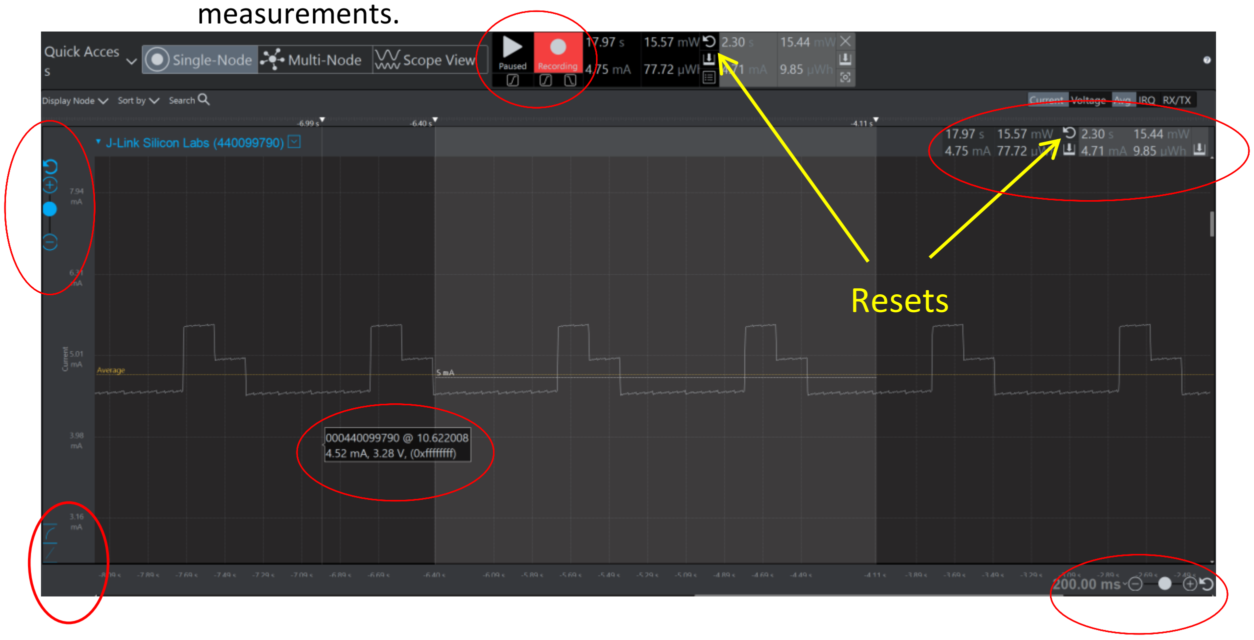
```
Failed connection to J-Link Silicon Labs (440099815)
  Could not locate silink.
```

gpio.h

```
//**********************************************
// Include files
//**********************************************
#include "em_gpio.h"

//**********************************************
// defined files
//**********************************************

// LED 0 pin is
#define LED0_port        gpioPortF
#define LED0_pin         4
#define LED0_default     false   // Default fc
// LED 1 pin is
#define LED1_port        gpioPortF
#define LED1_pin         5
#define LED1_default     false   // Default fc

//**********************************************
// global variables
//**********************************************


//**********************************************
// function prototypes
//**********************************************
void gpio_open(void);


```

2. In the Energy Profiler, click the pause button towards the top center.

3. Reset the measurements by clicking on their respective resets. See next figure.
    a. Click the play button towards the top center to restart the Energy Profiler measurements.



<span style="color:red">4.</span> Click on the lower left of the vertical access to change the scale from logarithmic to linear. Linear is indicated by a straight line at a 45 degree angle while logarithmic is indicated by an arc. <u>Use the linear mode for all measurements in this course</u>. WE WILL BE USING THE LINEAR MODE FOR ALL MEASUREMENTS IN THIS COURSE UNLESS OTHERWISE SPECIFIED!

5. To get accurate current readings, after each code download, you must move the Power Select Switch on the Pearl Geck dev kit completely to the left and then back to the right.
    a. Failure to perform this power reset can affect your Lab worksheet scores

6. You can zoom in and out the vertical axis, current measurement, via the + and – buttons located on the left side approximately ¼ down from the top. The circular arrow will reset the scale to the default current / division setting.

7. You can zoom in and out the horizontal axis, time, via the + and – buttons located at the bottom right corner. The circular arrow will reset the scale to the default time / division setting.

8. Pausing the Energy Profiler, with zooming in while using the linear scale, take the average current measurement to determine the following:
    e. Total current drawn when both LEDs are OFF
    f. Total current drawn when one LED is ON
    g. Total current drawn when both LEDs are ON

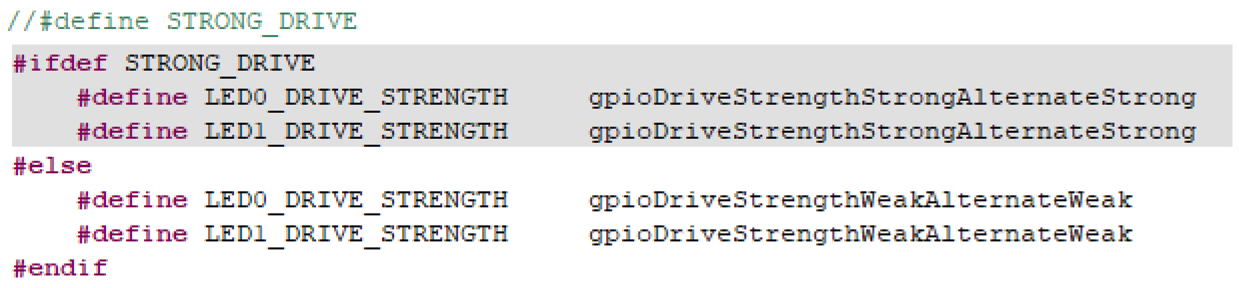
9. How much current is one LED consuming?

10. The Pearl Gecko's outputs can be configured to provide different amount of currents to its output pins. This feature can be used to reduce signal noise by reducing the transition time of the signal or to safe power.
    a. In brd_config.h, you will be switching the drive strengths used by gpio.c by altering the definition of the #define STRONG_DRIVE
    b. In gpio.h, you will find the commands to set the drive strength of the LED0 and LED1 pins
        i. GPIO_DriveStrengthSet(LED1_PORT, LED0_DRIVE_STRENGTH);
           GPIO_DriveStrengthSet(LED1_PORT, LED1_DRIVE_STRENGTH);
        ii. LED0_DRIVE_STRENGTH and LED1_DRIVE_STRENGTH are definitions defined in brd_config.h.
    c. The compiler directive in brd_config.h will define these constants as STRONG if the definition STRONG_DRIVE is defined

```
#define STRONG_DRIVE

#ifdef STRONG_DRIVE
    #define LED0_DRIVE_STRENGTH     gpioDriveStrengthStrongAlternateStrong
    #define LED1_DRIVE_STRENGTH     gpioDriveStrengthStrongAlternateStrong
#else
    #define LED0_DRIVE_STRENGTH     gpioDriveStrengthWeakAlternateWeak
    #define LED1_DRIVE_STRENGTH     gpioDriveStrengthWeakAlternateWeak
#endif
```

    d. For the next part of this assignment, you will want to explore how changing the output drive of the Pearl Gecko's outputs affect your previous measurements.

By commenting out the definition of STRONG_DRIVE, you will see that the else clause is now active, not shaded out, indicating that the definitions LED0_DRIVE_STRENGTH and LED1_DRIVE_STRENGTH will now be defined as Weak.



```
//#define STRONG_DRIVE
#ifdef STRONG_DRIVE
    #define LED0_DRIVE_STRENGTH     gpioDriveStrengthStrongAlternateStrong
    #define LED1_DRIVE_STRENGTH     gpioDriveStrengthStrongAlternateStrong
#else
    #define LED0_DRIVE_STRENGTH     gpioDriveStrengthWeakAlternateWeak
    #define LED1_DRIVE_STRENGTH     gpioDriveStrengthWeakAlternateWeak
#endif
```

11. Rebuild/compile and run the updated program. ***Remember to always switch the power connector from AEM/DBG to BAT and back to reset the Energy Profiler***

11. Pausing the Energy Profiler, with zooming in with the linear scale, take the average current measurement to determine the following:
    a. Total current drawn when both LEDs are OFF
    b. Total current drawn when one LED is ON
    c. Total current drawn when two LEDs are ON

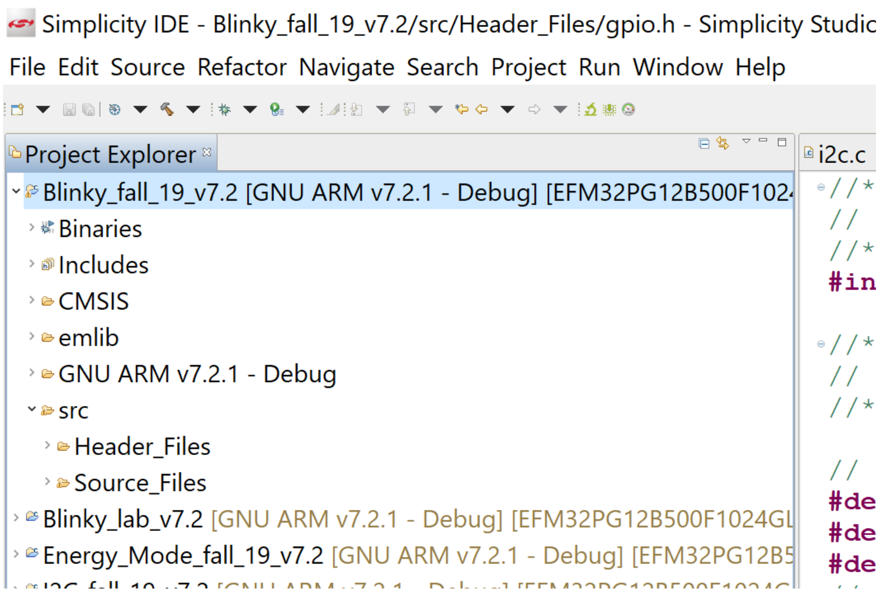
12. How much current is one LED consuming?

13. Now go back into the Simplicity IDE, and comment out the following line of code in the main.c routine
    a. GPIO_PinOutSet(LED1_port, LED1_pin); => //GPIO_PinOutSet(LED1_port, LED1_pin);

14. Now click on Run for Simplicity Studio to compile, flash, and start the updated program.

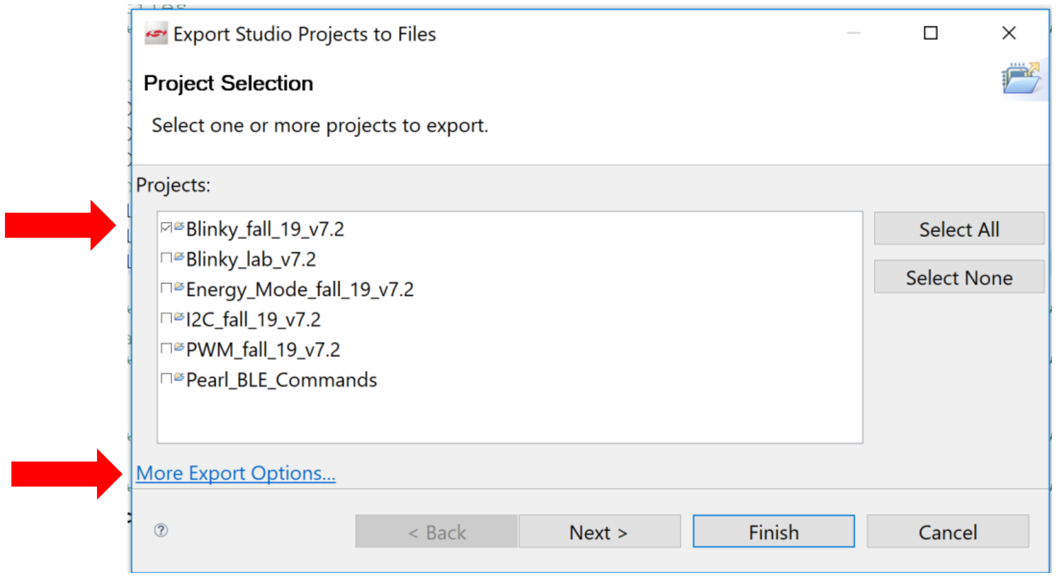
15. To complete the assignment, you must export your project and upload it into the Lab 1 Assignment in the course Canvas website.

    a. Export the project by right clicking your project in the IDE perspective and then selecting Export from the main menu File pulldown


Simplicity IDE - Blinky_fall_19_v7.2/src/Header_Files/gpio.h - Simplicity Studio
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
i2c.c

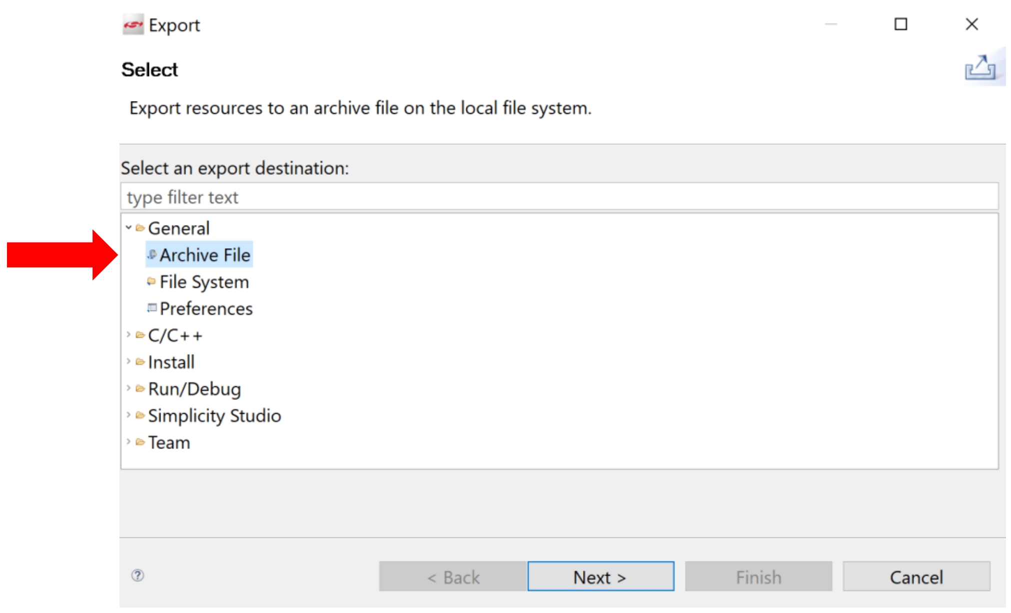
Blinky_fall_19_v7.2 [GNU ARM v7.2.1 - Debug] [EFM32PG12B500F102
Binaries
Includes
CMSIS
emlib
GNU ARM v7.2.1 - Debug
src
Header_Files
Source_Files
Blinky_lab_v7.2 [GNU ARM v7.2.1 - Debug] [EFM32PG12B500F1024GL
Energy_Mode_fall_19_v7.2 [GNU ARM v7.2.1 - Debug] [EFM32PG12B5

b. In the Export popup window, insure your project is select and then click on More Export Options.

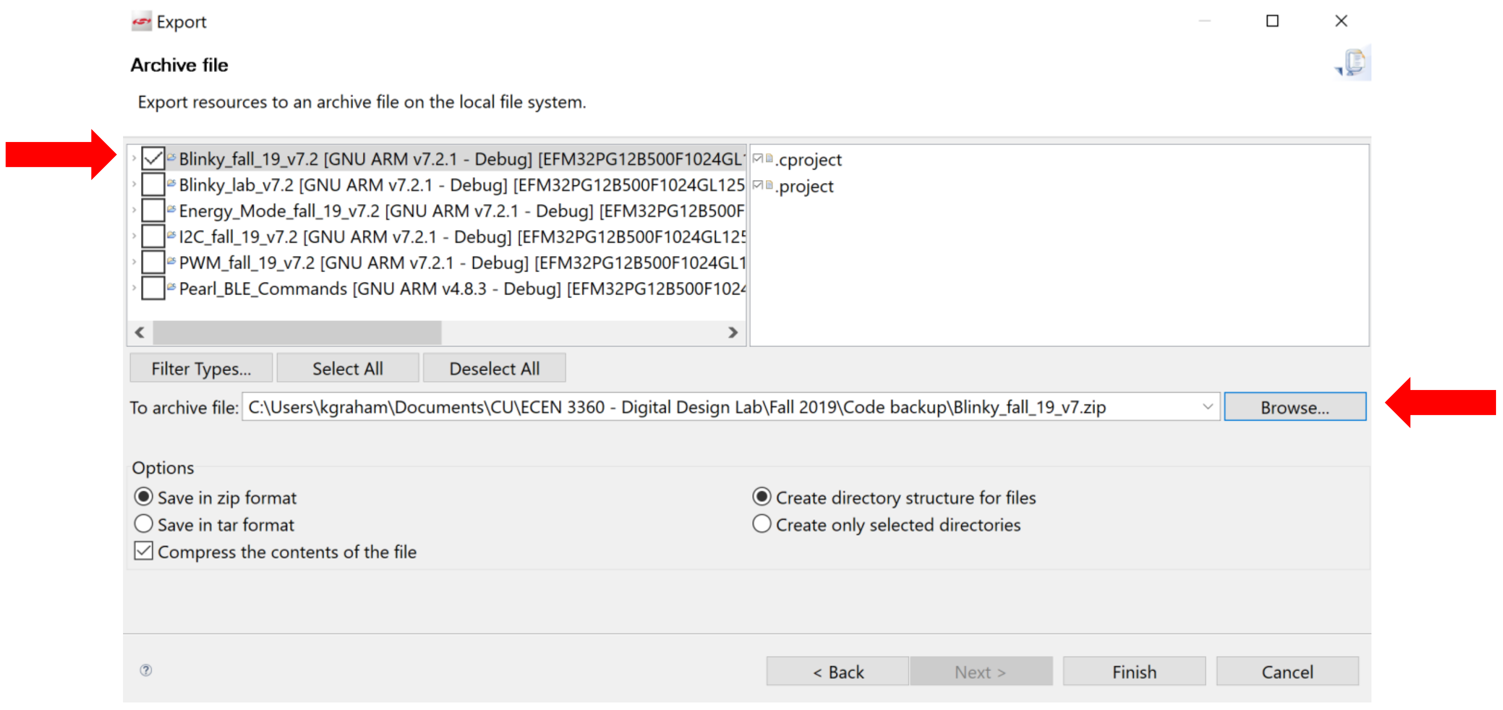


☑ Blinky_fall_19_v7.2
☐ Blinky_lab_v7.2
☐ Energy_Mode_fall_19_v7.2
☐ I2C_fall_19_v7.2
☐ PWM_fall_19_v7.2
☐ Pearl_BLE_Commands

c. In the More Export Options, you may need to expand the window to see all the options. Select Archive File and then click on Next.



d. In the next Export popup window, make sure that only your project is selected and that you have selected a folder and name for the exported .zip project. Once ready, click Finish.


Export
Archive file
Export resources to an archive file on the local file system.
Blinky_fall_19_v7.2 [GNU ARM v7.2.1 - Debug] [EFM32PG12B500F1024GL
Blinky_lab_v7.2 [GNU ARM v7.2.1 - Debug] [EFM32PG12B500F1024GL125
Energy_Mode_fall_19_v7.2 [GNU ARM v7.2.1 - Debug] [EFM32PG12B500F
I2C_fall_19_v7.2 [GNU ARM v7.2.1 - Debug] [EFM32PG12B500F1024GL12
PWM_fall_19_v7.2 [GNU ARM v7.2.1 - Debug] [EFM32PG12B500F1024GL1
Pearl_BLE_Commands [GNU ARM v4.8.3 - Debug] [EFM32PG12B500F1024
.cproject
.project
Filter Types...
Select All
Deselect All
Browse...
< Back
Next >
Finish
Cancel

To archive file: C:\Users\kgraham\Documents\CU\ECEN 3360 - Digital Design Lab\Fall 2019\Code backup\Blinky_fall_19_v7.zip

Options
[✓ Save in zip format]
Save in tar format
[✓ Compress the contents of the file]
[✓ Create directory structure for files]
Create only selected directories

e. Now locate your .zip exported project and upload it into the Lab 1 assignment located in this course Canvas course website.

Reference link: Hardware Abstraction Level (HAL)

- https://docs.silabs.com/mcu/latest/efm32pg12/

Questions:

Complete the Lab 1 worksheet for this Lab. The Lab 1 worksheet can be found in the Canvas Course Quiz section.

Point breakdown:

Lab 1

• Program execution (exported project)	10 pts

Late Penalty deduction:

- Exported program

○ Due day + 1 day	max score is 8 pts
○ 1 day late to 3 days late	max score is 6 pts
○ After 3 days late	max score is 0 pts

Appendix A (Possible MAC O/S fix for j-link silink error):

If you are on a MacBook and have issues with the Energy Profiler, please try the following recommendation from Silicon Labs. It has been validated by several students.

Here is the feedback from Silicon Labs regarding the j-link/silink error on MAC O/S based systems. If you are having this issue, can you please read the following and implement the recommended work around?

I just verified the bug on my Mac. It is a bug occurring on the newest SDK. The bug fix is provided in the KBA provided by Craig and I have verified it works on Mac.

Craig's knowledge based posting:

https://www.silabs.com/community/software/simplicity-studio/knowledge-base.entry.html/2019/12/20/issue_with_networkanalyzerandenergyprofileron-uqTQ (Links to an external site.)

Here is a bit more specific solution:

cd /Applications/Simplicity\ Studio.app/Contents/Eclipse/developer/adapter_packs/silink (this is the default path on the Mac, it could be different if customized installation path is used, just cd into the silink folder inside the SDK)

```
rm libjlinkarm.6.dylib
ln -s libjlinkarm.6.22.4.dylib libjlinkarm.6.dylib

rm libjlinkarm.dylib
ln -s libjlinkarm.6.22.4.dylib libjlinkarm.dylib
```

5 lines into terminal, and it should fix the problem.

Hope this resolved your problem! *Go Buffs*!

Regards,

Silicon Labs