ECE2049 – Embedded Computing in Engineering Design

Lab #0 – Introduction to the MSP430F5529 Launchpad-based Lab Board and Code Composer Studio

In this lab you will be introduced to the Code Composer Studio development environment that we will be using to program and debug our custom MSP430F5529 Launchpad based Lab Board. Code Composer Studio 7.2.0 is part of the ECE lab image and is on all of the computers in AK-113 and AK-227. If you want, you can download a free version of Code Composer Studio (*make sure it is version 7.2.0!*) to your own computer by following the links under Useful Links on the class website. This lab is straightforward and tutorial in nature and does not have a pre-lab. You are expected to sign-off each section of the lab and to answer all the questions highlighted in yellow in your lab report. You and you partner will be given a lab IO board to go with your MSP430F5529 Launchpad (available at bookstore or on-line). You are responsible for the board. Please treat it (and it's box) with care. You break it, you buy it!

Getting Started

1. Slide your board out of the protective box. DO NOT RIP THE BOX OPEN! You are responsible for your lab board and you must return it in good working order. You will need the box to protect the board over the term. Check the bottom of the board to be sure it has little rubber feet on each corner. It should but please: **DO NOT PLACE BOARD ON THE METAL COMPUTER CASE if it does not have the feet!** Ask the TA or tutor for some feet. In the mean time place the board on the cardboard box to prevent the possibility of shorting the board.

2. Assemble the board as shown below and in the Introductory Slides (on the Course Website Lab page). ***Double check that BOTH parts are oriented correctly and are seated properly before you power up the board!*** Plug the USB cable into the board and into the computer. With the board connected via the USB cable, double click on the desktop link to Code Composer Studio (CCS) or navigate to it through Start menu under Texas Instruments->Code Composer Studio 7.2.0. It is important for the cable to be connected when you open CCS this first time. If you get a warning pop-up about Java, just hit cancel.

3. When asked to enter a workspace, enter something like the following to create a workspace for this lab. Although you don't have to, you can create a new workspace for each lab. Create your workspace on your WPI CCC network drive *not* on the local machines. The local machines can be re-imaged at any time and you will lose your work!

R:\ECE2049_labs or R:\ECE2049_labs\Lab0

4. If the Welcome screen is not visible after CCS opens, click on Help and select Getting Started. The TI Resource Manager Explorer window should now be visible.

5. Watch the tutorial video, Getting Started with Code Composer Studio. This video contains a lot of very helpful tips on using the CCS debugger.

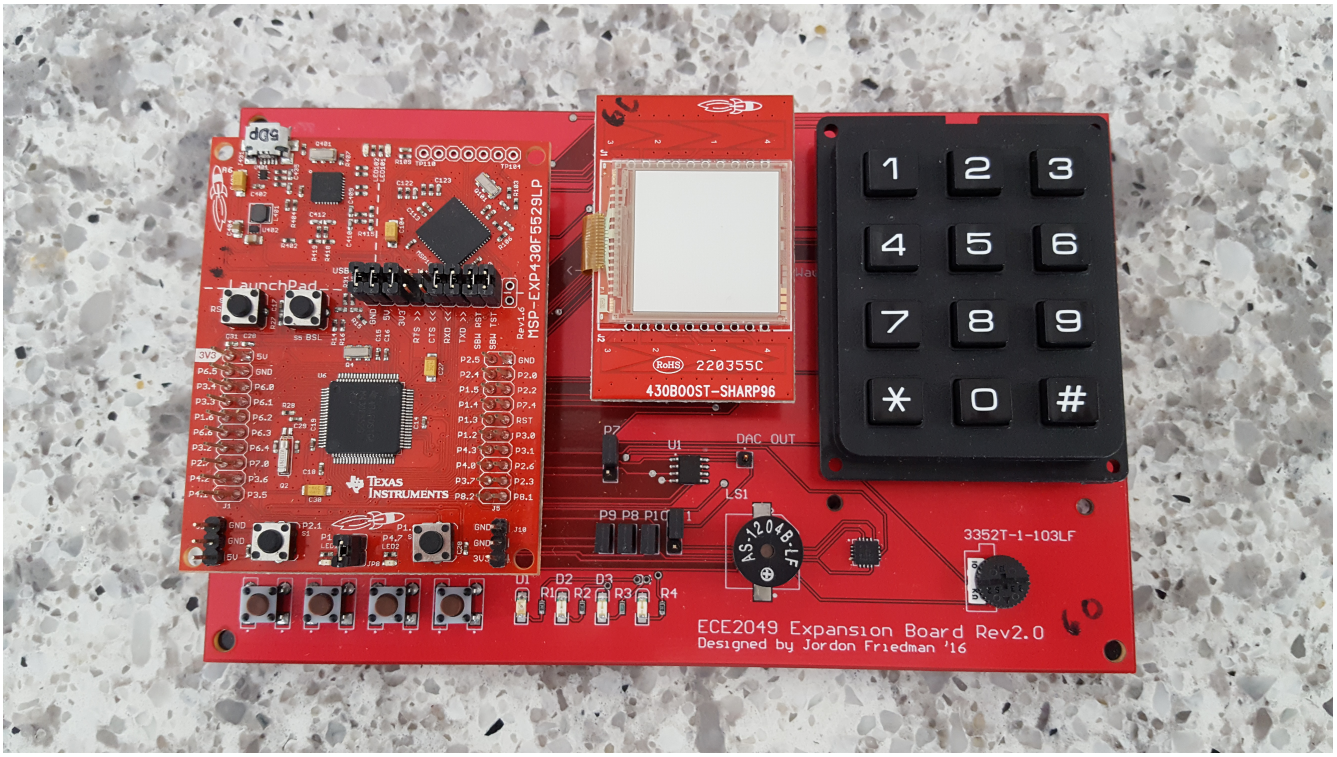
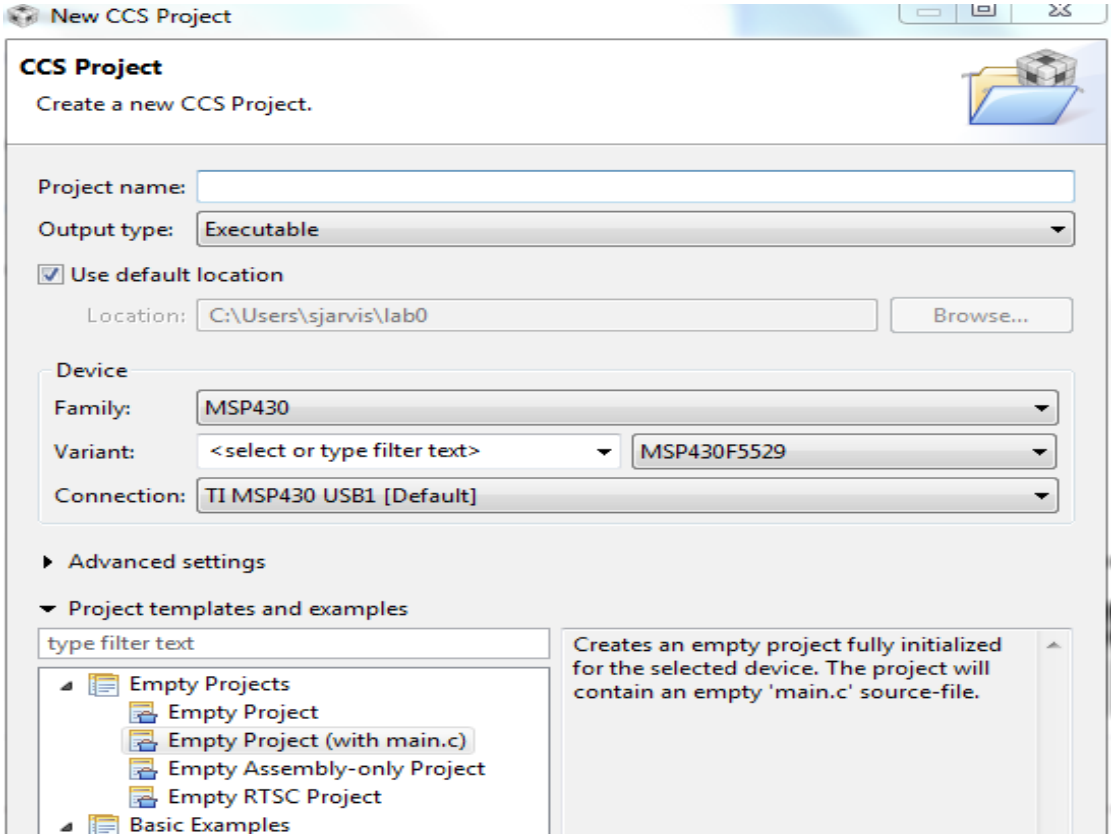


Figure 1: ECE2049 lab board with MSP430F5529 Launchpad & Sharp LCD screen properly mounted.

Building your first project – Don't Blink!

Download the file *blink.c* from the class website to the local Desktop then under Projects tab in CCS select New CCS Project. The window below will open. Enter a project name and set the other fields as shown below to match our processor, the MSP430F5529. It is critical that you select the right processor. Your project will not work if the wrong processor is given here. Select the Empty Project and then click Finish. This will return you to the main editor view and your new project should appear in **bold** the Project Explorer side pane. If it is not **bold**, click on it to select it as the active project and it should then be bold. Expand your project. If a skeleton *main.c* file was created you should delete it.



Now under the Project tab, select Add Files and add *blink.c* from the Desktop (choose the Copy option to copy the file to your new project directory). If *blink.c* doesn't open in the editor window. Click on the file in the side pane and it will open. Build your project (hammer icon) and enter the debugger (bug icon). If this is the first time you've run these steps, CCS can take a long time (several minutes). It's building cable drivers and libraries, etc. This is stuff it only needs to do once. If you get a pop-up window with MSP430 Ultra Low Power (ULP) warnings, you can dismiss it. You may also get a pop-up saying that there was an error in initialization and asking to update drivers. Click the UPDATE button. This, too, can take a few minutes. Be patient and DO NOT UNPLUG your MSP430 during the update.

Once the Debug control panel appears hit the go button (green arrow). This was covered in the tutorial video. The red LED on the custom lab board should now be blinking. Blinking lights are cool. *Demonstrate your first embedded programming success to a TA for sign-off*. Optionally, take a video and send it to your mom to show her what you do at school.

IMPORTANT NOTE: *Never* (ever!) unplug the board while it is programming or while in the Debug Window. *Always* exit to Editor or exit CCS entirely before unplugging the board

Hit the Pause button on the debugger and then the reset (these were in the tutorial video) and then start the blinking again. Hit pause and this time use the Step button to step through the code. You can watch the value of P6OUT in the registers window (under View/Registers). Try setting a breakpoint on the line *i = 50000;* and repeatedly running to that breakpoint

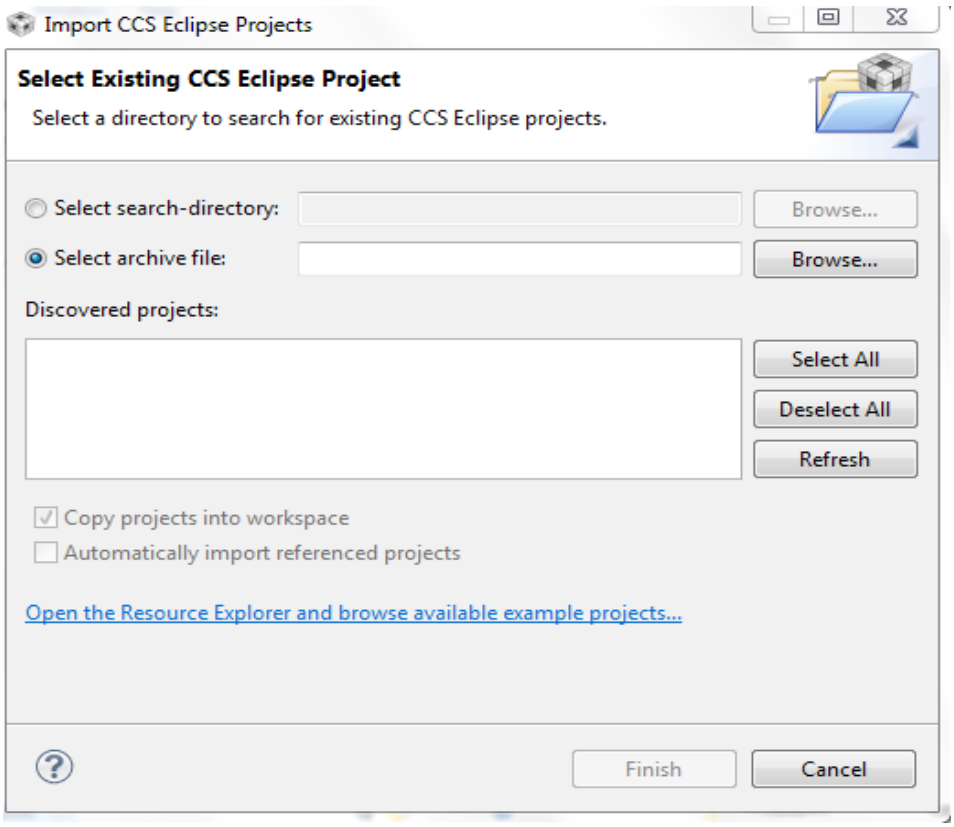
As shown in the comments in the code and in the schematic for the MSP430F5529 lab board, the RED is connected to digital I/O Port 6 Pin 2 (P6.2). The LED flashes on and off depending on whether a logic 1 or a logic 0 is written to that pin. In *blink.c* P6.2 is toggled by XOR'ing (^) the Port 6 Out register P6OUT with BIT2. BIT2 is a constant defined in *msp430f5529.h* as 0x0004. *Explain how and why this works in your report.*

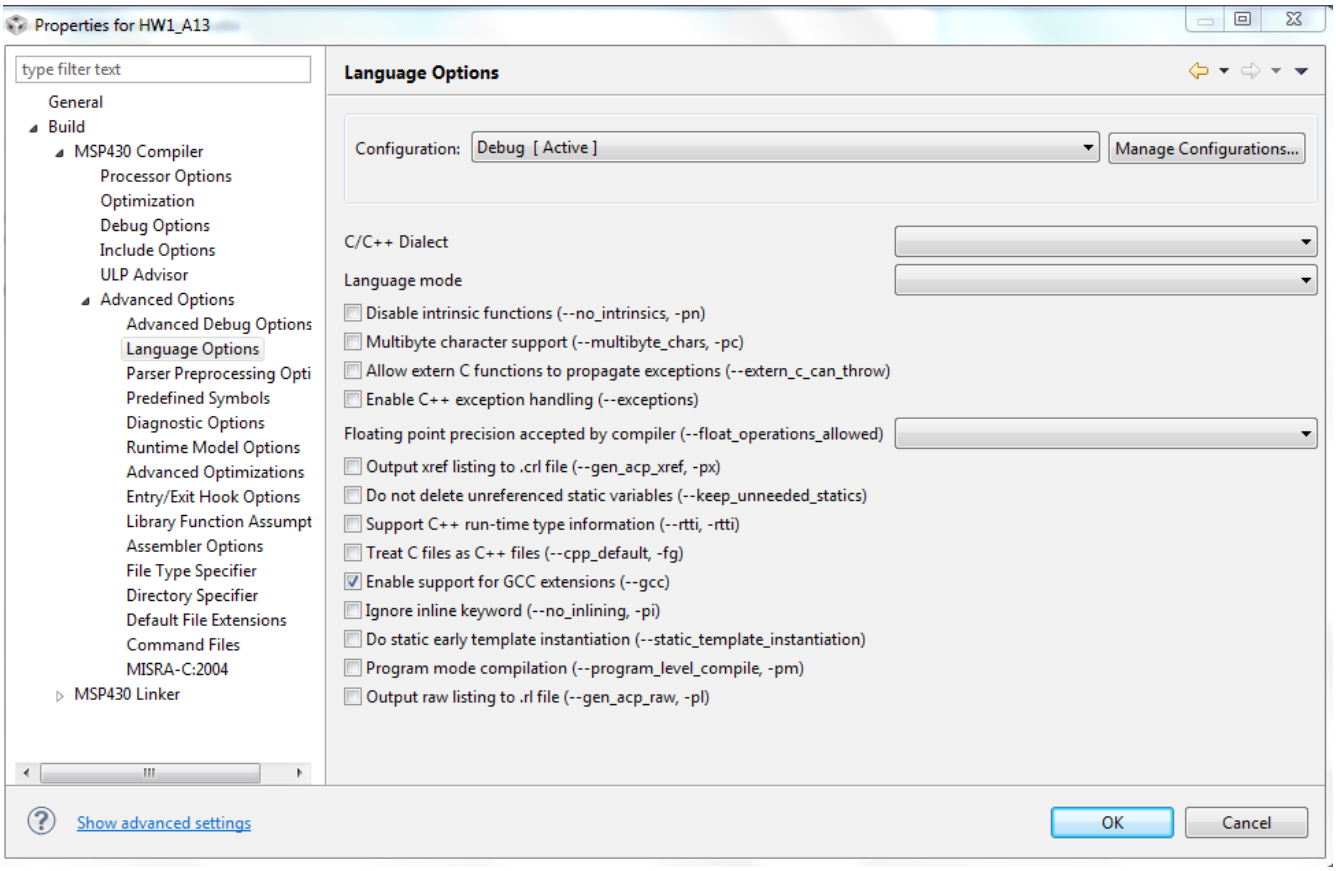
In *blink.c* there are 2 commented lines of code below the exclusive OR (XOR) statement. Test whether logic 1 or logic 0 lights the LED by commenting out the toggle statement and then successively un-commenting each of these lines of code. You can also determine which logic value lights the LED by stepping through the code and watching the value of P6OUT. *Document which logic level (0 or 1) lights LED1 in your report.*

Importing and building your second project

Download *ece2049_demo_CCS7_A18.zip* from the class website to the local desktop. Inside CCS, return to the Editor window (i.e. press the red square) and select Project/Import Existing CSS Eclipse Project. In the window that appears (below) chose Select archive file and navigate to the Desktop and enter *ece2049_demo_CCS7_A18 .zip*. Check the Copy Projects option and click finish. The project should now appear in the side pane. Click on it to expand the project and to select it as active.

Now right click on this project and select Show Build Settings. Under MSP430 Compiler/Advanced Options/Language Options double check that the Enable Support for GCC Extensions option is selected. It should be but if it is not the project will not build and it's nasty to find! *Build* this project and enter the debugger. You may get warnings about some unused variables. You can ignore them for now. *Run the project and demonstrate it to the TA for sign-off*. Once you are done playing with the demo, stop the debugger and open *main.c* in the editor view. This project is actually quite a bit more complex than *blink.c* because it includes the libraries required to use the LCD screen. Most, if not all, of your labs this term will require the LCD screen, therefore, your labs will start by importing this demo. You will use the given graphics library functions as is and expand upon the other functions as needed. You will also add new capabilities.


Import CCS Eclipse Projects
Select Existing CCS Eclipse Project
Select a directory to search for existing CCS Eclipse projects.
Select search-directory:
Browse...
Select archive file:
Browse...
Discovered projects:
Select All
Deselect All
Refresh
Copy projects into workspace
Automatically import referenced projects
Open the Resource Explorer and browse available example projects...
Finish
Cancel


Properties for HW1_A13
type filter text
General
Build
MSP430 Compiler
Processor Options
Optimization
Debug Options
Include Options
ULP Advisor
Advanced Options
Advanced Debug Options
Language Options
Parser Preprocessing Opti
Predefined Symbols
Diagnostic Options
Runtime Model Options
Advanced Optimizations
Entry/Exit Hook Options
Library Function Assumpt
Assembler Options
File Type Specifier
Directory Specifier
Default File Extensions
Command Files
MISRA-C:2004
MSP430 Linker
Language Options
Configuration: Debug [Active]
Manage Configurations...
C/C++ Dialect
Language mode
Disable intrinsic functions (--no_intrinsics, -pn)
Multibyte character support (--multibyte_chars, -pc)
Allow extern C functions to propagate exceptions (--extern_c_can_throw)
Enable C++ exception handling (--exceptions)
Floating point precision accepted by compiler (--float_operations_allowed)
Output xref listing to .crl file (--gen_acp_xref, -px)
Do not delete unreferenced static variables (--keep_unneeded_statics)
Support C++ run-time type information (--rtti, -rtti)
Treat C files as C++ files (--cpp_default, -fg)
Enable support for GCC extensions (--gcc)
Ignore inline keyword (--no_inlining, -pi)
Do static early template instantiation (--static_template_instantiation)
Program mode compilation (--program_level_compile, -pm)
Output raw listing to .rl file (--gen_acp_raw, -pl)
Show advanced settings
OK
Cancel

Exploring the demo project

Starting with example code and modifying it to perform desired tasks is an often used development strategy in embedded systems. Experiment with and modify *main.c* within the demo project to do at least the following.

1) Play with the LCD write commands to move the text to different positions on the screen. *Document what controls the position of text on the screen in your report*.

2) Right click on a function and select expand declaration. This should open the declaration and is a great tool during debugging.

3) Declare an array of char and initialize it to contain your name. Write your name to the LCD. *Note*: this is an array of char. Do you need to include a NULL terminator ('\n') as the last element of your name array or not?

4) Step through the code until you get to the assignment for `a_flt`, `X` and `tst`. Answer the questions asked in the comments in your report. (Hint: Use the Variables window.)

```
// What size does the Code Composer MSP430 Compiler use for the
// following variable types? A float, an int, a long integer and a char?

// What value stored in myGrade (i.e. what's the ASCII code for "A")?
// What is the new value of tst? Explain?
```

5) When does the buzzer sound? More importantly, what turns it off?

6) What is the relationship between the keypad and the 4 colored LEDs? Does the keypad fuction return ASCII or integer values? Explain in your report.

7) Write a high quality lab report and submit your final (modified!) demo code on-line to practice the process.

Writing a High Quality Lab Report

Your lab should be written in a professional style. It should be an electronically prepared technical document like what you would submit to a fellow engineer or your boss. The report should include:

Introduction = 1-2 paragraphs (1/2 page tops) succinctly stating the objectives of the lab and giving an overview of what you accomplished.

Discussion and Results = As many pages as it takes (without padding!). In this section you should thoroughly discuss what you did in each part of the lab. You should describe the approach you took to solving any problems. Again, this is a technical document. It should present your work in a clear and concise fashion. Include pseudo code or flow charts to explain any code that YOU developed (you do not need to include code from tutorials, code presented in class, etc). Results should also be thoroughly discussed. Any measurements should be tabulated, questions should be stated as given and answered completely (and in complete sentences). In general any figures should not be hand drawn, snippets of code may be included where useful but full listing should be submitted on-line as instructed.

******* Be SURE to clearly indicate your answers to any questions asked in the lab assignment. The TA can't give you credit if they can't find your answer!***

Summary and Conclusion = 1-2 paragraphs (1/2 page tops). Wrap-up and summarize what you accomplished in the lab. This should be a "bookend" to the introduction.

Appendices = Include any relevant raw data sheets, or links to them.

YOU MUST SUBMIT YOUR ORIGINAL SIGN-OFF SHEET WITH YOUR LAB REPORT!!!

DON'T FORGET TO SUBMIT YOUR (modified) PROJECT ON-LINE (see below)! One submission per team

In industry, the FIRST view of YOUR work by *anybody* other than your immediate supervisor will see will probably be in WRITING!

Learning to be an effective communicator of technical information is probably THE MOST IMPORTANT job skill you can have.

This lab used code provided to you but in all FUTURE LABS you will be writing your own code to solve a problem and will be REQUIRED to submit your CCS project on-line. We will start that submission procedure with this lab. Submit your modified demo project

To submit your code for grading, you will need to create a zip file of your CCS project so that the TAs can build it. You can also use this method to create a complete backup copy of your project (perhaps to archive or send to your partner) for later. To do this:

1. Right click on *your last modified demo project* and select "Rename..."
2. If you are submitting your project, enter a name in the following format: *ece2049a18_lab0_username1_username2*, where username1 and username2 are the user names of you and your partner. (NOTE: Failure to follow this step will result in points deducted from your lab grade!)
3. Click OK and wait for CCS to rename your project.
4. Right click on your project again and select "Export..." then select "General" and "Archive file" from the list and click Next.
5. In the next window, you should see the project you want to export selected in the left pane and all of the files in your project selected in the right pane. Select all. You should not need to change which files are selected.

6. Click the "Browse" button, find a location to save the archive (like your R drive) and type in a file name using the EXACT SAME NAME used in Step (2).

7. Click "Finish". CCS should now create a zip file in the directory you specified.

8. Go to the Assignments page on the class ***Canvas*** website. Click the **Lab 0** assignment. Attach the archive file of your project that you just created and hit the Submit button. Only one code submission per team.

ECE2049 A-Term 2018
Lab #0 -- Sign-off Sheet

(***This sheet MUST be attached to Lab Report!***)

Report due: **Tuesday 9/4/2018**

Student 1: _____ ECE mailbox: _____

Student 2: _____ ECE mailbox: _____

Board #: _____

BOTH PARTNERS MUST BE PRESENT AT SIGN-OFF!

Task Max points	***Max points***	***TA's assessment***
Build and demonstrate blink.c tutorial	10	
Download, build and demonstrate MSP430 development board demo	10	
Demonstrate modified demo project	10	
Answer to TA Questions	5	Student 1 Student 2
Report (Include answers to all questions and code submitted on-line)	15	
Total points	**50**	

*****BOTH students MUST be present for ALL sign-offs!***

***Note*: Be sure to include your original sign-off sheet with your lab report!**

ECE2049 Lab Report Grading Rubric

Format -- 3 pts

Did you follow instructions given above as to the format of your report?

Is you code formatted, properly commented, etc.?

What is expected for the following parts was already described above.

Introduction – 1 pts

Discussion/Questions – 7 pts

Conclusion – 1 pts

Professionalism – 3 pts

Spelling, grammar, neatness, presentation, etc.