

# ME 461 Laboratory #1 (Two Week Lab)

## Soldering and Introduction to the Hardware and Software

### Goals

---

1. Learn to solder by soldering the base set of components to the breakout (green) board.
2. Start to become familiar with the TMS320F28379D LaunchPad and Code Composer Studio.
3. Build and run your first microcontroller program.

### Exercise 1

---

In this exercise, you are going to solder components to your breakout board. Your instructor will distribute the printed circuit boards. You will also receive a TMS320F28379D Launchpad. Your instructor will give a short soldering demo and show you where to find wire and other components in the lab. In future labs you will be given the schematic files for the green breakout board you will be soldering. Follow the instructions of your instructor to solder your board.

### Exercise 2





---

First follow the other Lab 1 document “Using the ME461 Repository” and its first section “Clone the Repository” to check out the ME461 repository to your lab PC and/or your personal laptop. If you are going to be using both the lab PC and your laptop for development, you will want to create the same path on both your laptop as on the lab PC. Partners create a folder using a combination of your NetIDs at the root C:\drive (don’t use -, \*, or spaces) and keep all your files there. It is also a good idea for you to make backups of your lab files outside of your Git repository as you progress through the semester. Making this extra backup will save you when we can’t figure out what went wrong with Git. Git is awesome and normally works but every once in a while, I have seen issues that caused students to lose versions of their files. This is mainly due to us being beginner Git users, but I am getting better each semester.

### Exercise 3

---

1. Open Code Composer Studio 12 (CCS 12) and select the “workspace” folder in your repository. For example, if your netIDs were “jonesz3” and “smithd5”, you would have checked out your repository in c:\jonesz3\_smithd5\ME461\_Repo. The workspace folder then to select would be c:\jonesz3\_smithd5\ME461\_Repo \workspace.

2. Once CCS 12 is done loading your workspace, you need to load the “LABstarter” project. When you perform this load, the project is copied into your workspace. Therefore, if you rename this project, you will be able to load the “LABstarter” project again if you would like to start another minimal project. I purposely located this “LABstarter” project in the same folder that has many of the example projects you will be studying this semester. These examples are part of the software development stack that TI calls C2000ware. I have copied the needed parts of C2000ware into our repository so that if you accidentally modify something you can easily get it back. Again, now if your netIDs were “jonesz3” and “smithd5”, perform the following to load your starter project. In CCS select the menu Project->Import CCS Projects. Click “Browse” and explore to “C:\jonesz3\_smithd5\ME461\_Repo\C2000Ware\_4\_01\_00\_00\_F28379D\device\_support\F28379D\examples\cpu1\LABstarter” and finally press “Finish”. Your project should then be loaded into the CCS environment. Let’s then rename the project “lab1” by right clicking on the project name “LABstarter”. In the dialog box change the name to “lab1”. Also explore into the project and find the file “LABstarter\_main.c”. Right-Click on “LABstarter\_main.c” and select “Rename”. Change the name to “lab1\_main.c”. To save you some headaches in the future, I recommend you perform one more step after you have created a new “LABstarter” project. Right click on your project name in the Project Explorer and select “Properties.” Select “General” on the left-hand side. Then in the “Project” tab find the “Connection” drop down and select “Texas Instruments XDS100v2 USB Debug Probe.” “Apply and Close”.
  
3. Now that you have the project loaded you can build the code and download it to the LaunchPad board. Plug in your LaunchPad to the USB of your PC and then in CCS hit the green “Debug” button  and in the dialog box that pops up, select CPU1 only. This will compile the code and load it to your LaunchPad board.
  
4. Click the Suspend button  to pause the code and the Resume (or Play) button  to resume or start your code’s execution Use both of these to prove to yourself that both the blue and red LEDs are blinking on and off. Use one more button in CCS 12, the Restart  button. Run your code and then press the pause button to stop your code. If you hit the Resume button the code starts up again from where you stopped it. If you press the Restart button, CCS will take you back to the beginning of your code and then you can click the Resume button to start your code again from the beginning. This saves time if you just need to restart the same code. If you need to make changes to your code, Restart does not download the new code. You have to re-Debug your code to get the new changes downloaded to the processor.

5. Read through the `main()` function of your `lab1_main.c` file and see if you can find the function that changes the period of the timer functions. Change the period and see if the LEDs blink at a different rate.

## Exercise 4

---

1. I may make changes to the class repository so each week when you come into lab it is a good idea to see if there are any changes. First, to merge these changes, if there are any, from my repository that you earlier merged, perform the steps in the “Using the ME461 Repository” document section “Course File Updates”. This is a bit confusing the first time so please ask questions.
2. Create a new LABstarter project using the same steps as above. Rename this project something like “printtest” and remember to rename the `LABstarter_main.c` file to say `printtest_main.c` or whatever you would like.
3. Build and run this program. Look at the code and you will see that the function `serial_printf` is called every time the variable `UARTPrint` is equal to one. How often is `UARTPrint` getting set to one? **Show your answer to your TA.**
4. To see this printed text, you need to install a serial terminal on you PC. I like the program Tera Term on Windows. *On Mac do a web search for the Serial Terminal program “screen”.* Plug in your F28379D-Launchpad board to a USB port. We need to figure out what serial port COM number your USB serial port is using. The easiest way to find this is to run “Device Manager” in Windows and find the “Ports” item. Under ports find the COM number for the device titled “XDS100 Class USB Serial Port”. Run Tera Term and select the “Serial” item and find the XDS100 COM port in the list of COM ports. Final thing to do is change the Baud (or Bite) rate of the COM port. Still in Tera Term select the menu item “Setup” and then “Serial Port...”. Change the “Speed” to 115200 if it is not already.
5. Now you are ready to “bug/debug” you code in Code Composer Studio. Download and run your code and you should see text printed in the serial terminal.
6. Also click (or give focus) to Tera Term and type some text into the terminal. The typed text will not be shown in the window but those characters are being sent to the F28379D. Notice that when you type, the number of characters received increases “Num SerialRX: ”. Look through this starter code and find the `serial_printf` statement and notice that it is printing a variable `numRXA`. Go to the top of your C file and find the line `extern uint32_t numRXA`. The `extern` here is telling the C compiler that the global variable `numRXA` is defined in another C file as `uint32_t numRXA`. Highlight `numRXA`, right click on it, and select “Open Declaration”. This

will take you to the C file “F28379DSerial.c” where `numRXA` is defined. Perform a search in this file (Ctrl-F) for `numRXA`. You will find that this variable is incremented by one every time the function `RXAINT_recv_ready` is called. This function is the receive interrupt function for the serial port that is connected to Tera Term. This interrupt function is called every time Tera Term sends one character to the F28379D processor which is every time you type a character into Tera Term. The sent character is received in the global variable `RXData`. Inside this `RXAINT_recv_ready` function, write an if statement that checks if the character in `RXData` is equal to the character ‘a’. If it is ‘a’, turn on the board’s red LED. Write another if statement that checks if `RXData` is equal to ‘b’. If it is ‘b’, turn off the board’s red LED. Use the command `GpioDataRegs.GPBSET.bit.GPIO34 = 1;` to turn OFF the red LED. Use the command `GpioDataRegs.GPBCLEAR.bit.GPIO34 = 1;` to turn ON the red LED. In your main C file, where in future labs you will write most of you code, make sure to find in CPU\_Timer0’s interrupt function the line that toggles on and off GPIO34. Comment that line out so you can control the on/off of the RED LED by pressing the ‘a’ and ‘b’ keys.

## Lab Checklist

---

1. Demonstrate to your TA that you have successfully created, built, and ran your first DSP project.
2. Show your program that changed the period of the blink rate of the blue and red LED.
3. Show in lab that you can print text to a serial terminal. Also show that when you type text into the serial terminal the number of characters received changes in the print line and when you type ‘a’ The RED LED turns ON and when you type ‘b’ RED LED turns OFF.
4. For your lab submission, create a subfolder in your class Box folder and name it “Lab1”. Inside this folder submit:
  - a. A “How To” document that explains how...
    - i. to create a new LABstarter project in Code Composer.
    - ii. to pull up Tera Term and connect to the LaunchPads serial port.
    - iii. to check out your repository from github.com and comments that may help you remember this procedure.
    - iv. what `__interrupt void RXAINT_recv_ready(void)` is doing for us in the default project.