BIEN4220: Lab 1: Introduction to the eZ430-MSP430F2013

Introduction

This lab introduces you to the eZ430-F2013 Development Kit, which provides a complete MSP430 development toolset with all the hardware and software needed to develop applications with the MSP430F2013. It is packaged in a convenient USB stick form factor. You will use the Texas Instrument's Code Composer Studio (CCS) Integrated Development Environment (IDE) to program and debug an embedded application. The USB port provides power to operate the ultra-low-power MSP430 so no external power supply is required. All 14 pins of the MSP430F2013 are accessible on the MSP-EZ430D target board for easy debugging and interfacing with peripherals. The MSP430 also has a built-in LED tied to one of the digital pins just like an Arduino.

In this lab, we will compare/contrast the usage of a simple program written in C and Assembly on our MSP430.

Throughout this lab you will need to have a TA or instructor sign-off on objectives of the lab. Be prepared to explain what you are showing the evaluator. Simply saying "Here is my signal on the scope" is not sufficient.

Lab Preparation:

1. Read this document.
2. Figure out how you can measure the **frequency** and **duty cycle** of a square wave signal using an oscilloscope.

Procedure:

Part I: Creating a new CCS project and programming the MSP430

In this part of the lab we will upload C code to the MSP430 which will blink the onboard LED. Just as programmers usually start their journey of learning a new programming language with a program that types "Hello World" to the standard console, embedded systems developers nearly always begin testing their embedded systems development toolchain with a program that turns an LED on and off. Here, blinky.c will have all the code you need to toggle the state of an LED.

1. Download the file blinky.c from the class D2L site.
2. Using the Code Composer Studio IDE. Create a new Project by selecting File -> New -> CCS Project.
3. In the Target settings filter dropdown you can select "MSP430x2xx Family" which will narrow down the right dropdown so you can select "MSP430F2012" or MSP430F2013" (you can check which one you have by looking at the microcontroller chip).
4. Enter a project name such as "Lab1_C".
5. Select "Empty Project" and click finish.
6. Add the blinky code to the project by clicking Project->Add Files…
    a. Select the blinky.c file that you downloaded. You may be prompted to copy or link the file. I suggest you copy the file so that it is saved with the project. Linking will keep blinky where it is and reference it (so if you downloaded it and then delete it later, it also gets deleted from your project).
7. Double click bliny.c in the Project Explorer to read what the program does!
8. To compile the code, download the application to the target device, and start the debugging session, click the little green bug icon. You should then see your first line of code (usually labeled "RESET") highlighted. Your CPU has been halted by the debugger. Click the green triangle "Resume" button in the toolbar or press F8. Your CPU will now execute the program. Note, you can "suspend" execution by clicking the parallel lines next to the Resume button. While suspended, you can step through your code – line by line – using the yellow arrow icon "Step Into".
9. Use the oscilloscope to measure the frequency & duty cycle of P1.0.

Blink Frequency: 1 Hz **Duty Cycle:** 50%

10. Pause execution of your code. Set a breakpoint at the line where P1.0 is "toggled". Set breakpoints by double clicking on the line numbers. Resume execution. Resume it again. Resume it a third time. **What happens?**
11. To terminate the debug session, click the red Terminate button on the toolbar.

Part II: C vs. Assembly on the MSP430

In this part of the lab we will upload Assembly code to the MSP430 which will blink the onboard LED.

1. Create a new project in CCS called "Lab1_asm". Under project settings, select "assembly only" project.
2. Add the blinky.asm code to your project. If there is another .asm file automatically generated for you, you may have to "Exclude file(s) from build" by right-clicking that file in the left hand "Projects" pane and selecting the exclude option from the pop-up menu.
3. Assemble and download the program to the target device. Using the oscilloscope, measure the frequency and duty cycle of the signal on P1.0. Note that the MaxCount is the same value in Assembly as it is in C.

Blink Frequency: 4 Hz **Duty Cycle:** 50%

4. Stop executing code. To better understand the logic of the program, let's step through it when MaxCount isn't a ridiculously large number. Change the value specified for the variable MaxCount to the value 5 (10,000x smaller than the original), re-upload your code, and run it without breakpoints. Using the oscilloscope, you should see the blink rate increased by about 10,000 times.
5. Now let's look at the logic of this program and the registers involved. Pause execution of your code. Set a breakpoint at the line corresponding to the label "DelayLoop". Expand the "Core Registers" item in the "Registers" window. Right-click register R4 in the Register window. Change its format to decimal. Resume execution of your code. Resume it again. Resume it a third time. What happens?
6. Pause execution of your code and step through it (Step Into button), paying particular attention to R4. What is happening?
7. Expand the "SR" register item in the Register window. You might want to increase the size of the "description" column in the Register window.
8. Single step through the code, watching the flow of instruction execution until the LED toggles again. **What pattern do you observe with the Z and N flags in the SR?**
9. Reset MaxCount to its original value (50,000). **Be prepared to describe line-by-line what the .asm and .c programs do.**
10. Compare the size and speed of the C program to the Assembly program and be prepared to **discuss with an evaluator why they may be different.** You can observe the memory usage in the Console panel after you upload your program.

	FRAM Usage (Program)	RAM (Data)	Blink Frequency (MaxCount = 50,000)
C Code	98 Bytes	52 Bytes	1 Hz
Assembly	64 Bytes	0 Bytes	4 Hz

For this signature, be prepared to:

- Compare and contrast the speed and code size of assembly and c.
- Describe the logic of the assembly program (referencing the SR, jump statements, R4, and the LED toggling)

Evaluator Signature: