CSE466 LAB 1 - "HELLO WORLD"

AUTUMN 2012

OBJECTIVES

In this lab you will DO the following:

- Blink a light-emitting diode (LED) at a rate of 1Hz
- Read the value of an analog photosensor

In this lab you will LEARN the following:

- Basics of the MSP430 microcontroller architecture
- Use of Code Composer Studio and the MSP430 C compiler
- Use of the MSP430 timer and ADC (analog to digital conversion) peripherals
- Programming and debugging with the eZ430 programming dongle

DELIVERABLES

At the beginning of the next lab period you will demo your working system to the TA. Be prepared to demonstrate any and all functionality developed throughout the lab assignment.

Prior to the beginning of your next lab period, each group should turn in the following to the course dropbox (one set per group):

- **A PDF** containing answers to all questions posed in this lab prompt
- Your **fully commented and neatly presented** code, in a zipped format.
  NOTE: Only source files (*.c, *.h, etc) are necessary. Make it obvious which files were used in which portion of the lab.

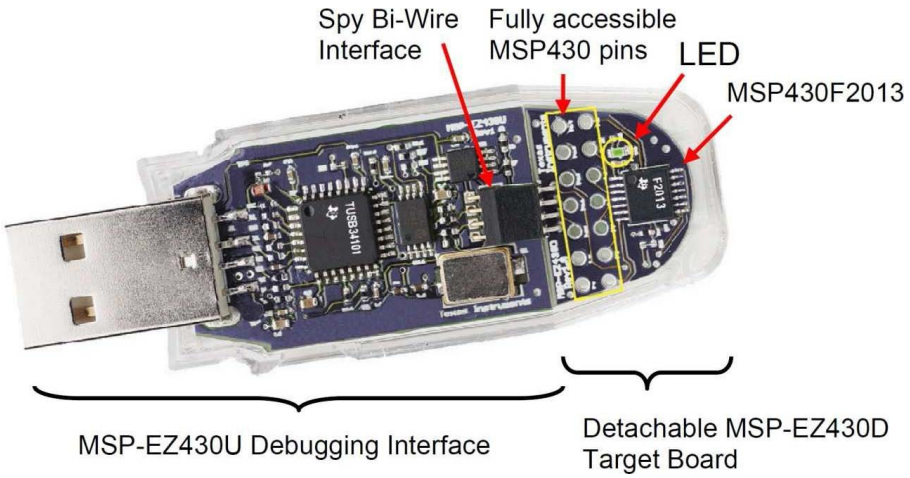
RESOURCES

These documents and web resources will be useful in completion of the lab and/or in answering the questions posed.

- MSP430F2013 Datasheet
- MSP430F20XX Family User's Guide
- MSP430F20XX Code Examples
- MSP430 Optimizing C/C++ Compiler User's Guide

PART 1: USING CODE COMPOSER STUDIO AND THE EZ430-F2013 DEVELOPMENT TOOL

We will use the eZ430-F2013 development kit in this portion of the experiment. This kit includes both an MSP430 programming tool and a detachable MSP430F2013 microcontroller. This section demonstrates how to set up a Code Composer project for the eZ430-F2013 and download the application to the MSP430F2013. The sample program will blink the LED on the MSP-EZ430D target board.



HINTS:

- Read the MSP430F20XX Family User's Guide section on Digital I/O.
- Make sure you understand the purpose of each line in the example code.
- Look at Table 5-1 (C/C++ Data Types) of the MSP430 Optimizing Compiler Guide.

PROCEDURE:

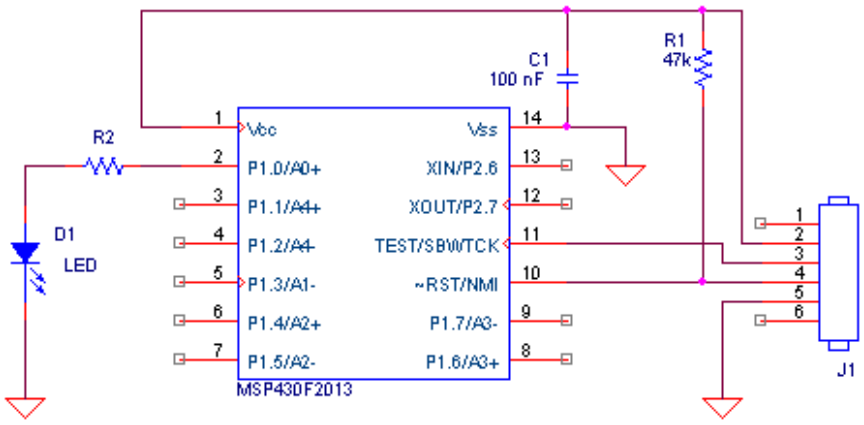
1. Plug in the MSP-EZ430U Debugging Interface with the target board attached.
2. Open CCS and File-> Switch Workspace to the directory in which you'd like your projects to reside.
3. File -> New -> CCS Project.
4. Name the project "Blink" in default workspace. Click Next.
5. Project type: MSP430. Next. Next. (no additional project settings).
6. Set Device Variant = MSP430F2013. Click Finish.
7. Download and unzip the MSP430 Code Examples for the MSP430F20XX series
8. In CCS, select the Blink project and click Project-> Add Files to Active Project…
9. Navigate to the MSP430 Code Examples C code directory and find "msp430x20x3_1.c"
10. Click Open to add the file to your project.
11. In the file you just added, change #include "msp430x20x3.h" to #include "msp430f2013.h" and save.
12. Use Target -> Debug Active Target to begin the debug session.
13. Use Target -> Run to start the application.
14. Observe blinking LED.
15. Use Target -> Terminate All to exit debug session.

Congratulations, you've just built and tested your first MSP430 application.

Question 1.1: Why is the variable i declared as volatile in the example code? What happens if you take out the volatile keyword?

Question 1.2: What are the minimum and maximum values that the variable i can represent? Describe what happens if you attempt to assign a value larger than the maximum (try it!).

PART 2: WIRING UP THE BREADBOARD



HINTS:

- Most LEDs have a forward voltage of about 2.0 Volts and require about 5 mA of current for adequate brightness.
- Read the MSP430F2013 datasheet section on the electrical characteristics of outputs.

PROCEDURE:

1. Construct the circuit shown using a breadboard.
2. The LED's current limiting resistor value is intentionally missing from the schematic. Calculate the required resistor value.
3. Before powering on the breadboard, verify that there are no shorts between power and ground, and that power and ground are connected to the proper MSP430 pins.
4. Turn on the eZ430U debugging interface, which will power up your breadboard.
5. To verify that your hardware is working properly, program your wired up breadboard with the same code used in Part
6. Modify and re-compile your code so that the LED blinks four times faster than before.

Congratulations, you've just wired and tested your first MSP430 circuit.

Question 2.1: What is the power source for your circuit? What is voltage of this source?

Question 2.2: Using the device datasheet for the MSP430F2013, find out how much current you can source from:
(A) A single output pin, and
(B) An entire output port combined (usually 8 pins)

PART 3: USE TIMER_A PERIPHERAL FOR BLINKER TIMING

Now you must modify your program to use the hardware timer peripheral "TIMER_A" to blink the LED.

HINTS:

- Read the Family User's Guide section on TIMER_A.
- Read the Family User's Guide section on Basic Clock Module operation.
- Read the MSP430F2013 datasheet section on DCO Frequency selection.
- Skim through the Family User's Guide section on Interrupts.

PROCEDURE:

1. Create and configure a new CCS project as in Part 1.
2. Add the MSP430 code example file “msp430x20x3_ta_02.c” to your project.
3. In the file you just added, change #include "msp430x20x3.h" to #include "msp430f2013.h" and save.
4. Start the debugger and run the application. Observe blinking LED.
5. Modify your code to toggle the LED every 1.0 seconds.
6. Use Target -> Terminate All to exit debug session.

Question 3.1: What does the following line do, exactly, in the sample application?

```
TACTL = TASSEL_2 + MC_1;                  // SMCLK, upmode
```

Question 3.2: What is the clock frequency of the processor in the sample application? Of the timer peripheral? How did you find this information?

Question 3.2: What is the clock source in this example? How accurate is this source? What are some other possible clock sources when using the MSP430 (There are at least two others)?

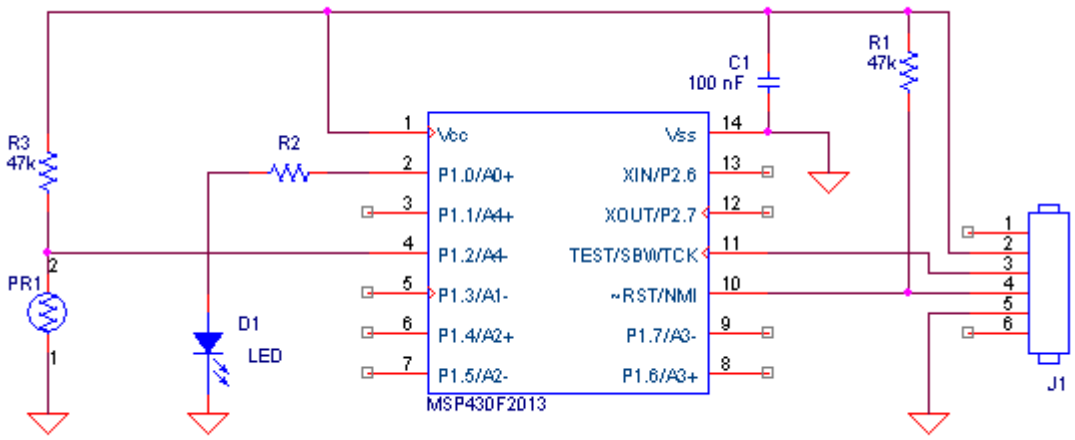
PART 4: USE THE ADC TO READ A PHOTO-RESISTOR AND CHANGE THE BLINK RATE

You will now enhance your design by adding a light sensor. The goal is to change the blink rate of the LED based on light intensity. To interface the analog photo-resistor with the microcontroller, you will use the MSP430’s Analog-to-digital converter (ADC) input.

HINTS:

- Skim through the Family User’s Guide section about the ADC10 peripheral
- Reference the code example file “msp430x20x3_sd16A_01.c” for an interrupt-driven method of reading the ADC value.

PROCEDURE:



1. Connect the photo-resistor and pull-up resistor to pin P1.2/A4+ of the MSP430F2013, as shown.
2. Create and configure a new CCS project as in Part 1. Create a new C source file for your code.
3. Write and test an application which:
    a. Flashes an LED using the TIMER_A peripheral.
    b. Reads the value produced by the photosensor at a regular interval.
    c. Changes the LED flash rate based on light intensity.

Question 4.1: What is the sampling rate of the ADC in your application? Describe two possible ways in which you can determine the ADC sampling rate when designing or testing this application.