ELEC 327 (2014) Lab #2: PWM, Low Power Modes and PCBs

Part 1: Different Clocks, Low Power Mode, and PWM

So far, we’ve only used SMCLK sourced by the DCO. This, however, is a very fast clock, and we will need to use a different one if we want to have our interrupts occur at a lower frequency. For the following questions, we highly recommend testing each function as you go, then doing whatever you need to tie it together at the end. Again, you should have both the MSP430G2553 datasheet and user guide open while doing all of this, as you should need to use it extensively.

a) Using the DCO, what is the minimum frequency for a timer interrupt? What is the typical default frequency of the VLO? How do you turn on the VLO, and how do you change code from last week to run off the VLO instead of the DCO?

In all our code from last week, we had the MSP430 running full-time between each interrupt. To save on a lot of power, it is useful to put it into a low power mode (LPM) when certain clocks are not being used.

b) If we only want to monitor the VLO (ACLK) when not in an interrupt, what low power mode should we put the MSP430 into? What line of code should we add to the end of our main loop to do this? How can you change this one line to also enable interrupts?

To change the brightness of an LED (or most other analog devices), we use PWM. This essentially changes the brightness by altering the duty cycle of the output signal. Typically, for PWM we will set register 0 of the timer to be the total period, with the timer operating in up mode. Then, we can set various pins to be on for a fraction of that period, creating a PWM signal when it is pulsed fast enough.

c) Say we want to use Timer A1 for our PWM signal. Which pins is register 1 of Timer A1 capable of sending a PWM signal to? What should PxSEL, PxSEL2, and PxDIR be for these pins?

d) In up mode, the timer will continuously count up to the value in register 0, resetting to 0 every time it reaches it. If we want register 1 to be on then off for some portion of this time, repeating every time register 0’s count repeats, what mode should we put this register in? How do we actually set it to this mode?

Write some code to toss all this together. Source Timer A1 off the VLO clock, leaving it at its default frequency. Configure your registers such that the duty cycle of the LED’s input signal is 50%, and make sure to enable the LPM found in b). Since the PWM signals are automatically generated, you should not need an ISR.

e) If you were to set CCR0 to be 500, why does the LED flicker and not dim?

Set CCR0 such that you see no flicker. Remove the LED from the circuit and move your Vcc jumper to the MSP430 side of the board and use a multimeter to measure the amount of current the MSP430 is using (while it’s still on the breadboard). Measure this again without any LPM on, and comment on the difference.
f) What do you measure for each case, and why are they different? If you were using a 250mAh coin cell battery to power this circuit, how long could it run in each mode?

Turn in the code with LPM on as PWM_test.c.

**Part 2: Making a Mood Ring**

In this part, we’ll put together most of the concepts from Part 1 and add in ADC to make a “mood ring” with a RGB LED (ignoring the green part) and thermistor. We’ll start off with a few questions that should guide learning how to use ADC:

a) How would you turn on the ADC10 module and tell it to sample off P1.0? What is the default voltage range for conversion? How would you change that to instead be from 0-1.5V? What is the benefit of this smaller voltage range? What is a downside of it?

Throw this all together: code your “mood ring” and wire it on your breadboard. We’ll be using this thermistor an RGB LED (data sheet on class website). Make sure to comment your code in detail, such that any design decisions you made are easily available (which components are on which pins, label interval for timers in ms, etc). Follow these specifications:

- Control the brightness of the red/blue sections of your LED depending on temperature measured by the thermistor. Configure your circuit such that it measures temperature values roughly between 36C and 38C. This is best done using 2 external voltage references: Vref+ and Vref-.
- The hottest value you measure should be all red, the coldest all blue. Intermediate values should be a mix of the two.
- Sample at a fixed interval, not continuously. Use whatever clock you think would be best.
- For cleanliness, put large groups of your setup code (such as those setting up ADC or timers) into separate functions.

b) Suppose in lab that we only had an MSP430G2230. Would you be able to simultaneously control both the red and blue parts of the LED using the automatic PWM signals (i.e. not controlling them in an ISR)? Why or why not?

**Part 3: Making it on a PCB**

Once you have this working on a breadboard, the last thing we’re going to do is make a schematic for it and actually order / assemble it on the received PCB. We’ll be using Eagle for this part.

Eagle comes with a good number of parts, but not all that you need. You also don’t want to make every part that you need yourself. Learn how to import existing libraries, and add the MSP430 library provided from the class website. We also suggest that you add the Sparkfun library, as it contains many useful parts. You won’t be using these much in this lab since you’re provided with a base schematic that has all the parts on it, but in future labs you’ll have to make your own schematic (and populate it with parts you find).
Open the provided project file. It contains all the necessary parts, except for the resistors (on the PCB, you’ll be using a coin cell battery instead of USB power). Add those in, connect everything in the circuit, then everything in the schematic. Try to make the final board as small as you can reasonably get it. (Good designs should be 1.5”x1.5” or smaller.) Additionally, for this project, PCB designs should be rectangular. After routing, check to make sure that it passes the electrical rule check (ERC) and design rule check (DRC). Then run the “BareBones.cam” CAM job to create the necessary files for manufacturing. Include a zip-file with these files as part of your lab submission. We will submit a panel of the class’s designs for manufacturing, and in a subsequent lab, you will assemble and test your PCBs. Thus, late submissions of this portion of the assignment will mean that you lose points not only for this lab but also a subsequent one!!!