Lesson 36 Worksheet – Analog-to-Digital Conversion

1. Before class, answer the following questions. Feel free to consult with the internet, your ECE 315 friends (they’ll want to recall lessons 17-18), or other resources, such as the tutorial on ADCs on the SparkFun website. Do not strictly copy your answers; ensure you understand the theory behind ADCs. This understanding may come in handy during our in-class exercise or on the final.
2. What is an analog-to-digital converter (ADC)?
3. What does resolution mean in regards to an ADC?
4. What is quantization error?
5. What is clipping?
6. What is aliasing?
7. What equation relates an ADC’s resolution with the maximum voltage expected and the voltage level measured for a particular sample?
8. How many bits does the ADC on our MSP430 use?
9. What is the highest hex value our ADC can return?
10. Why are ADCs useful?
11. What equations do you use to convert between Celsius and Fahrenheit?
12. In previous years, our robot used infrared sensors to detect the wall. Since we are using the ultrasonic rangefinder this year, we no longer need to use the ADC10 module for our robot. However, understanding how to use an ADC is still an important part of understanding embedded systems. Today, we will create a basic hot/cold indicator with our MSP430 and a temperature sensor, LM34CZ. Check out the tech doc for the sensor: <http://ece.ninja/382/datasheets/lm34.pdf>

<http://www.ti.com/lit/ds/symlink/lm34.pdf>

1. In short, we will connect the temperature sensor to our microcontroller. If the temperature detected is too hot or too cold, we’ll turn on the red or the green LED on our chip. Mr. Evans has ways of creating a temperature that is very warm or very cold to test out your program.
2. We are using the TO-92 package. ***NOTE THAT THE PINOUT VIEW IS FROM THE BOTTOM! DO NOT CONNECT THIS BACKWARDS OR YOUR CHIP WILL GET VERY HOT.***
3. What temperature range is this sensor good for detecting?

Let’s work on the code to use our ADC. Create a new C project in CCS. In our program, let’s create three global variables: InputVoltage, TempC, and TempF.

1. Let’s look at the ADC section again in the Family User’s Guide. Notice we have two control registers, ADC10CTL0 and ADC10CTL1. We will need to configure these registers to ensure we are most effectively using our ADC.
2. I chose input channel four. You are welcome to choose a different one; ensure you do not interfere with your LEDs. Let’s also divide our clock by eight. We need to choose our reference voltages; we can either provide an external voltage or use an internal voltage. Let’s use the internal reference voltage of 1.5 V (this also requires you to ensure the REFON bit is correct). We will let VR+=VREF and VR-=VSS. We are assuming that we will not need to measure temperatures above 150 °F.
3. What are the purposes of VR+ and VR-? Why is 1.5 V the better choice for VREF?
4. Let’s choose a long sampling time – in fact, let’s choose the longest sample and hold time available to us. Since we are using the ADC, let’s be sure to turn the subsystem on.
5. Now that the control registers are taken care of, we also need to specify the channel on which to expect an analog input. This is done in the ADC10AE0 register. Ensure your INCH selection matches the bit you set in this register.
6. Don’t forget that since we are using the LEDs on the MSP430 we need to set them up so they will turn on when we want them to.
7. Now that our initialization is complete, let’s move on to the meat of our program. Every time we want to take another sample from our sensor, we need to enable a conversion and tell the ADC to actually start doing the converting. This is also taken care of in the ADC10CTL0 register.
8. Do we ever need to disable the conversion enable?
9. After we take a sample, let’s have a short delay, then proceed to see what value our ADC came up with. This shows up in the ADC10MEM register. Let’s take that value and then figure out exactly what the voltage reading was the MSP430 received from the temperature sensor. Remember, we set our reference voltages at 1.5 V and 0V up above in part F. Given our binary number, how can we convert that to a voltage? (Hint: look at your answer to question 6.)
10. Let’s do math! Convert your ADC value to InputVoltage using your answer to question 6. Our sensor outputs 1mV per 0.1°F. So, if the temperature is 77°F, our voltage should be 770mV. Calculate your TempF, and use your equation from question 10 to calculate your TempC.
11. We are almost finished! Last thing: set your LEDs to turn on as a proper notification to the passersby. It is up to you to decide when to turn on the lights and what each color light should mean. Now we can test! (Hint: if you want to look at your three variables in CCS as you step through the program, highlight the variable name, wrong click, and select Add Watch Expression. This value will now show up under the expressions tab in Debug mode.)

What to turn in for this assignment:

* Answers to questions 1 – 13 (via BitBucket)
* Documentation statement
* A signature from your instructor showing that you completed the in-class exercise for today and that your program and circuit work as intended.