Kyle Han, 113799110

ESE280 Laboratory 11

Section 3

Bench 9

Sensors, Basic Analog-to-Digital Conversion, and the 12bit ADC

Design Task 2 Verification Strategy:

Because the pin of PE3 is now an analog input, we can just directly test this on the microcontroller, without pulling up or down the pin. Thus, the pin is floating, and it’s analog reading will change with time due to the environment. Thus, we only need to step to make sure the logic is correct.

With the potentiometer and the multimeter, we can specify an analog voltage that PE3 should be at. Once maintained, we can step forward to make sure that the ADC correctly digitizes the number (Following the format of a specific fraction of our ref voltage, 2.5V)

We can check what was analog voltage was read via the debugger. As we step through the program, we can see what was once in the res register of the ADC will now be set in the bcd_entries and led_display arrays in .dseg. Each step of the way, we verify our calculations.

The final result in bcd_entries should be the hand calculated value as hex.

We can repeat this several times to ensure that everything is working correctly.

Design Task 4 Verification Strat:

We can do the same as task 2, hand calculating whatever value that was read from PE3. However, the final result in bcd_entries should be a proper BCD entry that matches the decimal math of the equation ((res*2500)/4096)-500

1. What is the significance (particular usefulness) of the internal 2.5V reference voltage?

Because it is hard to store decimals (Especially analogous "When do they stop!" voltages, that refer to a very specific analog value, we can use a different method from directly storing them. We instead store them as a discrete fraction of a certain reference voltage.

Therefore, instead of saying "1.25V", we can say "½VREF", where Vref is 2.5V. This fraction is infinitely easier to store, over calculating the binary representation of a floating point value, and storing that decimal point.

Essentially, it turns a continuous analog voltage into a discrete voltage that our microcontroller can actually use via a scaling process.

2. When you single step your program in Task 2, is the analog-to-digital conversion process single stepped? Explain how you know this.

No, the ADC is not single stepped. We know this because each time we return to poll the ADC, it is immediately ready with a value (Since a practically infinite amount of time has elapsed since last checked, at least according to the microcontroller).

This also makes sense because the ADC is a different module than the CPU. Via single stepping, we only pause the CPU to look at the values, but all other modules are not stopped.

3. When your program polls to determine the end of a conversion, which flag bit must it poll and why?

It must poll bit 0 of the "ADC0_INTFLAGS" register. According to the datasheet, this register will be turned to 1 when a result is ready. Thus, to see if a conversion is done, just poll this bit.

4. When measuring a fixed voltage in Task2, do any of the least significant bits of the output change? If so, to what do you attribute these changes.

Yes, they change quite a bit. I attribute these changes to noise in the system, since nothing is perfect. Even the multimeter shows us it's least significant value changing slightly. Thus, our microcontroller's value will be changing, and so will the value being read.

5. What is the range of temperatures that can be measured using the MCP9700A and internal 2.5V reference? What is the smallest temperature change that can be detected. Show your calculations.

The temp range that can be measured is -38C-125C. The datasheet tells us that these voltages are between 0V and 2V, both voltages that our ADC/Microcontroller are built to handle. Thus, we can access this whole range.

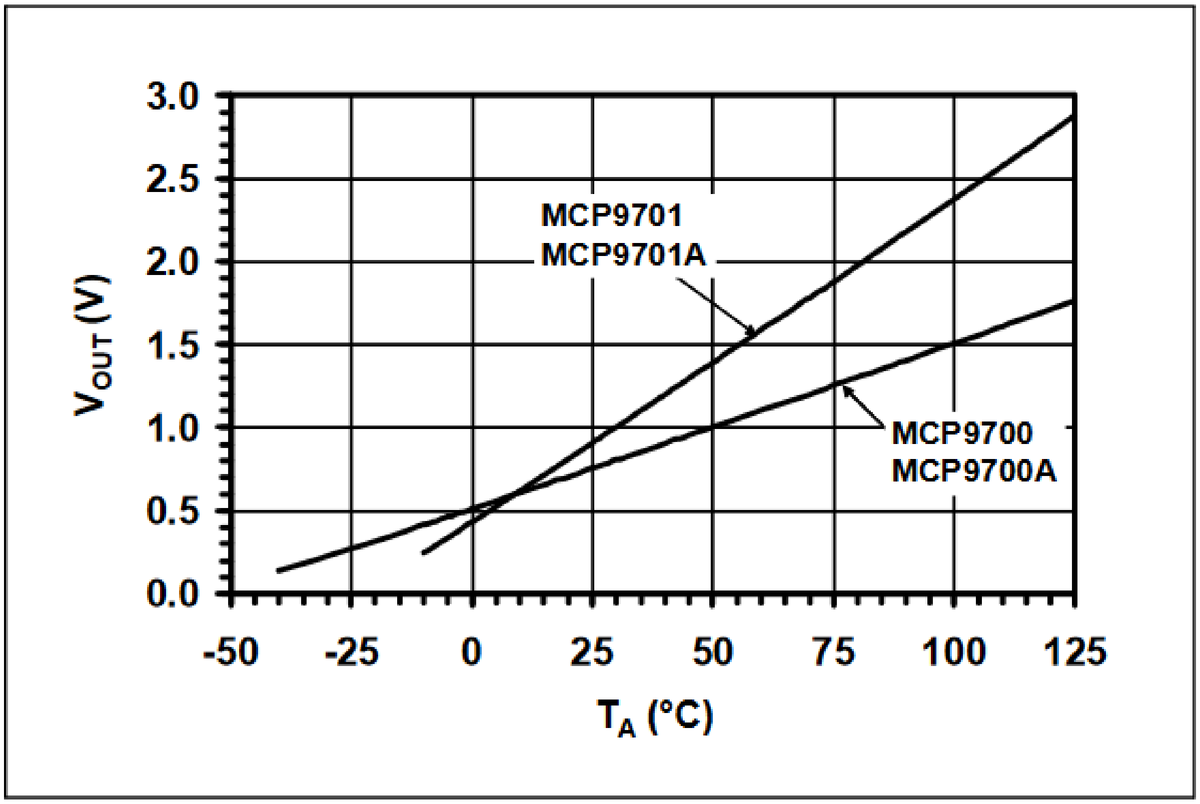
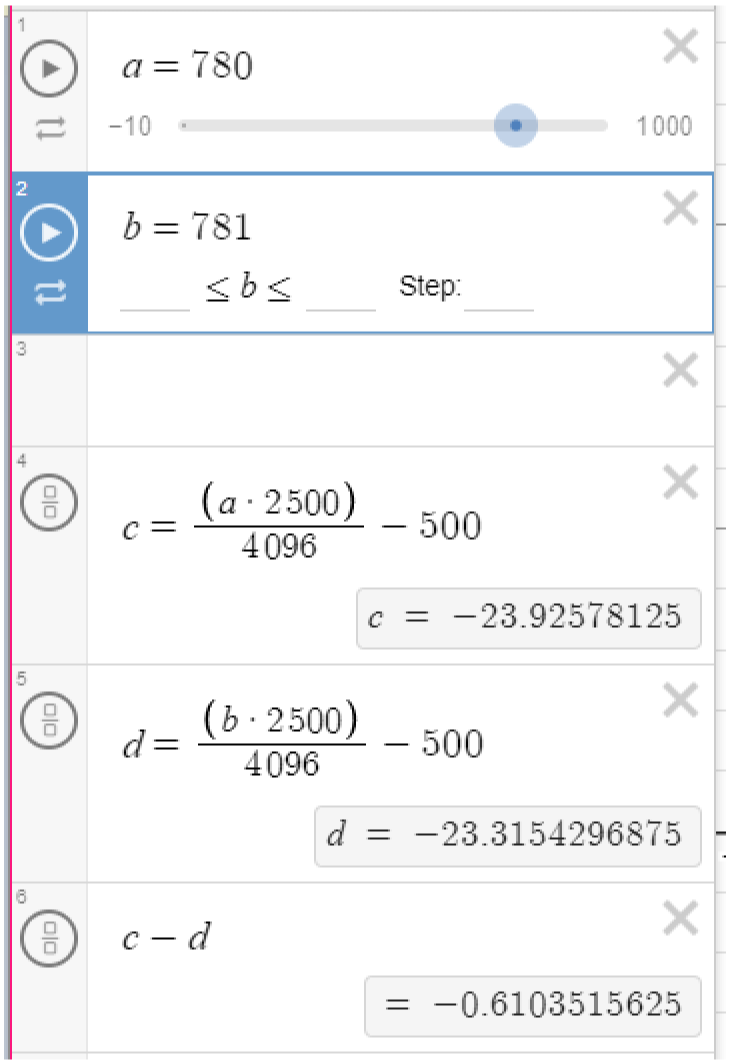


FIGURE 2-16: *Output Voltage vs. Ambient Temperature.*

The smallest temp change we can detect is from two consecutive hexs. Thus, plugging this into the equation that we have, we can compute the smallest temperature change to be .61C.



1 $a = 780$

−10 1000

2 $b = 781$

$\leq b \leq$ Step:

3

4 $c = \frac{(a \cdot 2500)}{4096} - 500$

$c = -23.92578125$

5 $d = \frac{(b \cdot 2500)}{4096} - 500$

$d = -23.3154296875$

6 $c - d$

$= -0.6103515625$

6. Write the basic equation for the computation you must perform to scale the ADC binary output that represents the measured voltage to a binary representation of the measured temperature. Are there any limitations in the order in which the operations in the equation must be performed. If so, explain the limitations and why they exist.

((RES*2500)/4096)-500

Yes, the limitation is that our divide function will cause us to lose data on the decimal side. Thus, even though in the real world, we can do multiplication and division in any order, in this case, we should do multiplication first, on order to not lose any data.