Ali Kedwaii – mak3799
Zachary Pope – zhp76
EE445L 15660

Lab 1 Report

Objectives

The main objective of this lab is to refamiliarize ourselves with using the TM4C microcontroller, the ST7735 LCD, and the Keil environment. The lab is also an exercise in understanding fixed-point notation and creating scatter plots on the LCD. We wrote four functions: one that translated and displayed decimal fixed-point numbers, one that translated and displayed binary fixed-point numbers, one that created a drawing field on the LCD, and one that drew on the screen with given plot points. To write these, we had to understand the components of fixed point notation and how to map given coordinates to the dimensions of the screen.
For the extra credit, we had to use the simulator and logic analyzer within the environment to test four snippets of code that implemented either floating point or fixed point, and were written in assembly or C.

Extra Credit Measurement Data

Test 1 FP, C
Start: 0.118063 ms
End: 0.153728 s
Execution Time: 0.15361 s

Test 2 Fixed, C
Start: 0.118063 ms
End: 7.799313 ms
Execution Time: 7.68125 ms

Test 3 FP, ASM
Start: 0.118063 ms
End: 3.702812 ms
Execution Time: 3.58475 ms

Test 4 Fixed, ASM
Start: 0.118063 ms
End: 3.447375 ms
Execution Time: 3.329312 ms

When comparing Test 1 to Test 2 and Test 3 to Test 4, the fixed-point functions were faster than the floating-point functions. This is because when using the FPU, floating-point values are passed to the unit's registers, calculations are carried out, and the result is passed back to the normal core registers. These extra steps add a lot of time to the execution of the code.
Overall, the assembly functions are much faster than the C functions, because the C code first needs to be compiled and assembled.

Analysis and Discussion

1) **In what way is it good design to minimize the number of arrows in the call graph for your system?**
Having less arrows means you have less dependency on other methods which could slow down your overall execution time.

2) **Why is it important for the decimal point to be in the exact same physical position independent of the number being displayed? Think about how this routine could be used with the ST7735_SetCursor command.**
Having the decimal point in the same place makes it possible to right justify the displayed values, making it easier to read.

3) **When should you use fixed-point over floating point? When should you use floating-point over fixed-point?**
We would use fixed-point over floating point if the values we are dealing with have the same precision and have a low precision that is easy to calculate.
We would use floating point over fixed-point if the values we are dealing with have different precisions, or if we require very precise calculations.

4) **When should you use binary fixed-point over decimal fixed-point? When should you use decimal fixed-point over binary fixed-point?**
We would use binary over decimal if the chip we use does not support decimal arithmetic, or if binary operations would be faster.
We would use decimal over binary if the chip does support decimal arithmetic, or if we want it to be easier to understand for users.

5) **Give an example application (not mentioned in the book) for fixed-point. Describe the problem, and choose an appropriate fixed-point format. (no software implementation required).**
One example of using fixed-point to solve a problem would be reading the temperature using a thermistor. Given the resistance of the thermistor and the type of thermistor, we can produce a formula to calculate the temperature with a precision of 0.01. Since all values we calculate will have the same precision, we can store the values in fixed-point.

6) **Can we use floating point on the ARM Cortex M4? If so, what is the cost?**
Yes, we can use floating point on the Cortex M4 because there exists a Floating Point Unit (FPU). The cost of using it is that you will need the power to turn on the unit, and the process of moving data from the main core to the FPU to carry out the calculations and back will greatly add to the execution time of the program.