A) Objectives (1/2 page maximum)

The objective of lab 1 is to familiarize ourselves with the TM4C123 launchpad and the ST7335 LCD panel. Additionally, we will be exploring the basic modular structure of projects on the TM4C123. For this lab, we will be implementing four functions.

1. ST7735_sDecOut2: takes in a fixed point number with resolution 0.01 and outputs it in a human-readable format on the LCD.
2. ST7735_uBinOut6: takes in an unsigned 32-bit binary fixed-point number with resolution 1/64 and outputs it in a human-readable format on the LCD
3. ST7735_XYplotInit: Specify the X and Y axes for an x-y scatter plot. Draw the title and clear the plot area.
4. ST7735_XYplot: Plot an array of (x,y) data

B) Hardware Design (none for this lab)

Done

C) Software Design (upload **`fixed.h fixed.c main.c`** files to Canvas as instructed by your TA)

See submitted files

D) Measurement Data (none for this lab)

E) Analysis and Discussion (1 page maximum). In particular, answer these questions

1) In what way is it good design to minimize the number of arrows in the call graph for your system?

First off, it makes the system more straightforward and easier for debugging. Secondly, the more calls between modules, the higher the latency will be generally.

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.

This aligns the numbers nicely and makes it much easier to compare printed numbers, therefore making the printed data more readable.

3) When should you use fixed-point over floating point? When should you use floating-point over fixed-point?

fixed-point over floating point: consumes less power.

floating-point over fixed-point: can be more precise.

4) When should you use binary fixed-point over decimal fixed-point? When should you use decimal fixed-point over binary fixed-point?

Binary fixed-point: rescaling operations is done faster via bit shifts.

Decimal fixed-point: when fractional powers of ten is required.

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).

Example: cashier machines.

Cashier machines do not need to be super precise in order to do its job. They show results with only 2 decimal digits, so a decimal fixed format is desired. Manufacturing these machines without a floating-point unit can also save money.

6) Can we use floating point on the ARM Cortex M4? If so, what is the cost?

Yes. The cost is that the arithmetic operations will be slower and consumes more power. Also, there is only single precision (“float”) and no double precision (“double”)

Bonus:

1. C Floating Point: 2.407 ms

2. C Fixed Point: 1.485 ms

3. Assembly Floating Point: 3.585 ms

4. Assembly Fixed Point: 3.330 ms

I tested the first two in a project derived from Lab 1 (by enabling the floating point capabilities in startup.s). I tested the last two in a project derived from Float_4FC123asm. The code can be found in Lab1.c in the bonus folder.

From the results we can tell that fixed point arithmetic operations are faster than using floating point on the TM4C123. What surprised us was that the two C test ran faster than the two written in assembly. We think that this may be the result of compiler optimization when it compiles the C code.