Lab1 Report
Michael Park(mjp2853), Jack Zhao(jz6492)
01/27/16

Lab 1. Fixed Point Output

Objectives:

Our objective for Lab 1 was to experiment with Fixed Point calculations on the microcontroller. Then we would have to demonstrate these values as outputs onto the LCD screen. Specifically, we would have to deal with a wide variety of situations requiring fixed resolutions, to test the robustness of our program in dealing with the situations without the use of floating point calculations. The reason we would need to do this is that floating point processing requires much more resources and time for calculation than fixed point calculations. We would need to deal with both the situations of printing out specific fixed point values onto the screen as well as using fixed point values in the plotting of a graph. We would need to perform these scenarios in different methods than just standard multiplication of a resolution.

Analysis and Discussion:

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

It is a good design because a reduced number of arrows in the call graph will reduce the modules dependent on any portion of the overall program. This allows the program to be used in a wide variety of settings and on several different machines with the same efficiency.

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.

We implemented the feature of setting the decimal point to be fixed (ie. In the exact same physical position independent of the number). The reason that we do this to pretty-print values on LCD and keep the resolution of the software consistent. Since we will have a format for the display values, we can use ST7735_SetCursor function to set the fixed cursor position for the displays. Also the displaying will become flexible in that we can display digits in the value in any order once we have the output string built.

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

Fixed point can represent integer numbers with combination of different bit patterns. This becomes handy versus the use of floating point, when the calculations themselves are not too complex. A few key considerations developers make when using fixed point is Cost, Ease of Development and Performance. Fixed point processors can be manufactured at a much lower cost than floating point processors because they are simpler in nature. In addition, the performance speed of fixed point calculations is much quicker than floating point which allows for the mass processing of data for fixed point calculations.

On the other hand, floating point calculation is much more complex than fixed point calculations, but allows for more robustness in the processing of data. Since floating point calculations are similar to scientific notations, both the resolution and the precision can be altered to reflect better values. Though the cost of manufacturing a floating point processor is much greater than for a fixed point processor, developing an algorithm for floating point processors is much easier as there is less manipulation of the actual values. In addition, floating point can be used to calculate a much wider range of values than fixed point which allows for more precise calculations. All these variables allow for floating point processors to be more specialized in data processing (ie. Addressing specific scenarios that a fixed point processor would not be able to address).

Lab1 Report
Michael Park(mjp2853), Jack Zhao(jz6492)
01/27/16

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 is preferred in situations where speed of calculations matter. Binary fixed point is processed much quicker because bit shift to re-scaling numbers into fixed point is much quicker than standard multiplication. But decimal fixed point is preferred for usability especially when the resolution is a multiple of 10 because it is much easier to develop programs using decimal values.

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).

An example of a system in our lives that uses fixed point is a vending machine. A vending machine (that accepts US dollars) uses fixed-point numbers. The resolution only needs to be 0.01. Therefore, the numbers can be represented as integers internally.

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

We can use floating point on the ARM Cortex M4 with single precision FPU. One cost of performing floating point calculations is a time cost:
Without FPU: 1.25 / 1.52 / 1.91 DMIPS/MHz
With FPU: 1.27 / 1.55 / 1.95 DMIPS/MHz
Another cost is a precision cost. FPU on ARM Cortex M4 only has single digit precision whereas we can more than single digit precision if we use fixed point.

Conclusion:
Some final considerations as we performed this lab is that there are different instances in using fixed point versus floating point calculations and this can depend based on a variety of factors. At one time, floating point was not possible to be implemented due to hardware constraints and so people developed methods completing various tasks using fixed point implementations. Though our hardware has progressed significantly, there are still many applications of fixed point in our society today.

Extra Credit Conclusion:
We have determined a few observations of the behavior of the four different methods of implementation(float c, fixed c, float asm, fixed asm). To measure the execution time, we measured the number of clock cycles for each method using the Systick clock.

c float: 2543238*12.5ns = 31.790475 ms
c fixed: 163869*12.5ns = 2.0483625 ms
asm float: 65553*12.5ns = 819.4125 us
asm fixed: 49183*12.5ns = 614.7875us

From these measurements, we have determined that assembly code runs faster than c code. In addition, we have determined that fixed point implementations executed faster than floating point executions. The ordering based on speed of execution was: Assembly Fixed Point< Assembly Floating Point<C Fixed Point<C Floating Point.