Ronald Macmaster and Parth Adhia

EID: Rpm953 and Pda375

09/05/2016

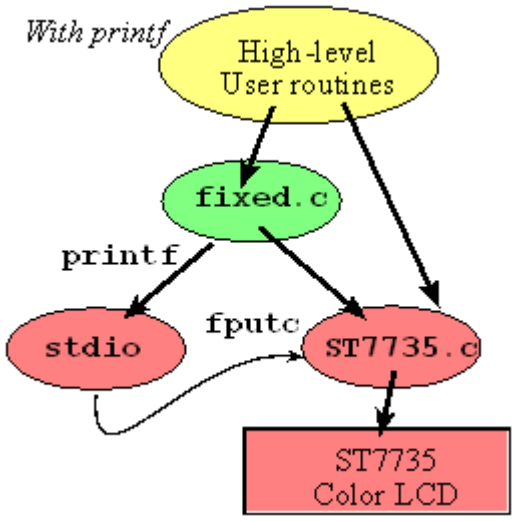
EE 445L – Lab 1: Fixed-Point Output

1.0: Objectives

We hope to get acquainted with the lab equipment and lab environment. Additionally, we will familiarize ourselves with the Keil uVision 4 IDE and the ARM Cortex M microprocessor. Our project involves developing a set of fixed-point output routines for interfacing with the ST7735R LCD screen. Thus, we will also strengthen our background in graphics driver development.

2.0: Software Design

We have uploaded the fixed.h, fixed.c, and main.c files to Canvas prior to implementing the software on our hardware. Below is the high level call graph designed by Professor Valvano.



3.0: Analysis and Discussion:

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

We minimize the arrow count to **reduce the functional overhead** of calling our fixed-point library. The LCD routines are called repeatedly and abundantly as a low level driver. The more call arrows, the more memory and time our system will need to operate.

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 would like to format our output to a consistent height and width. Allowing a flexible decimal point position would cause uncertainty in the output format. Thus, we would be unable to make predictable calls to the SetCursor method.

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

We use fixed-point when our range of values is **small and predictable**. Also, we choose fixed-point over floating-point when our values require a **higher amount of precision**. We use floating-point when the range of values is very large and unpredictable, or we can gain additional benefits from a native FPU.

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

We use binary fixed-point when the calculations / interpretations are to be done by a computer. We use decimal fixed-point when the calculations / interpretations are to be done by a human.

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).

We wish to measure the diameter and length of various bacteria to estimate their size in volume. Bacteria range from 0.2-2.0 um in diameter and are less than 1.0 um in length. The value will be calculated in decimal fixed-point with a resolution of 1×10^{-9} meters and 32-bit precision.

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 ARM Cortex M4. However, we must also push the floating-point registers onto the stack (in addition to the general purpose registers) upon servicing an Interrupt Service Routine.

Extra Credit:

We performed an empirical study on four different implementations of fixed/floating-point processes on our Cortex M processor. The implementations were in either C or Assembly respectively. We measured the execution time in SysTick counts on a processor running at 80MHz.

Here are the results:

Test 1 (Floating-point in C): 2625098 ticks

Test 2 (Fixed-point in C): 163876 ticks

Test 3 (Floating-point in ASM): 65566 ticks

Test 4 (Fixed-point in ASM): 49188 ticks

(Conclusions on the next page)

We can conclude that high-level floating point arithmetic operations are very expensive, even while utilizing the FPU (Test 1 vs. Test 3). However, there is not much difference between the low level routines in fixed/floating-point implementation. Finally, high-level fixed-point operations perform significantly better than high-level floating point operations for simple arithmetic.