Lab #4: Programming in C & Adding Serial RAMs (via DSP SPI)

**Purpose**

The purpose of this lab is to become proficient in using Code Composer Studio’s ‘C’ compiler and to learn how to interface serial SRAM to the DSP. Part II consists of re-doing Lab #3’s Floating Point Unit (FPU) algorithm/problem in ‘C’ that was previously implemented in assembly. Part III consists of re-writing the Lab #3 LCD code in ‘C’ and then Part IV will be interfacing a serial SRAM to the DSP SPI port.

**Part I. In-Lab Quiz.**

Your TA will give you a simple in lab quiz that should take approximately 20 minutes to complete. It will be very similar to the material in the following sections.

**Part II. Serial LCD Implementation in ‘C’**

You’re Serial LCD should be attached to your Launchpad (LP) main board as follows:

- LCD Vcc => LP +5V
- LCD GND => LP GND
- LCD SDA => GPIO104
- LCD SCLK => GPIO105

Using our LCD driver (OneToOneI2CDriver.c) and your ‘C’ code, create a main program that sends out Your Name (first and last) followed by EEL4511 to your onboard LCD. i.e. Jane Doe (1st LCD line), EEL4511 (2nd LCD line). You should create a function that can write commands down to the LCD in C using our driver and another function that writes a string of characters terminated by zero (null character) in C.

**Part III. Interfacing Serial SRAM to the DSP (SPIB)**

Using SPIB (port specified later in class) on the DSP, interface the serial SRAM contained in your parts kit as follows:

- GPIO63 => SPISIMOB, GPIO64 => SPISOMIIB, GPIO65 => SPICLKB, Any (2) GPIO (66/67?) => CS0/CS1

1. Print out and review the necessary pages in our data sheet found online next to this lab. i.e. device pin-out, timing diagrams, suggested wiring circuits, etc.

2. Memory Map: RAM0/CS0/256K X 8, RAM1/CS1/256K X 8, we will use two addresses in RAM to store a word.

3. SPI Initialization – We’ll be using the SPIB module’s High-Speed pins to interface with the dual SRAMs. Find which pins these are and initialize them for your SPI driver. Read the SPI portion of the F28379D manual and identify any registers that need to be changed. Connect the SPI bus to your DAD board’s logic analyzer and send known data to determine if your initializations are working.

4. SRAM Signal Connection - Match up the SPI bus pins to the Dual SRAM Breakout board, as well as the CS0 and CS1 signals. Before powering your board up, make sure no data bits are grounded or tied to +3.3V before powering on the board. Use your multi-meter and check all signals wired to headers. Wiring mistakes at this point can destroy your DSP ($$)!  

5. SRAM Software Testing – Create your own drivers to interface with the dual SRAM breakout. It is recommended to first write and read a single byte to each SRAM at a known address, and then make sure this works for every address in the memory map. You should then create functions that can write and read 16-bit words to the SRAMs: these functions can be byte-addressable. Start with a low SPI clock speed and increment up until the tests below no longer work. Record the final speed you get, which should be on the order of 10’s of MHz. Also, record how long your functions take to write/read a 16-bit word. Include this data in your lab report.
Lab #4: Programming in C & Adding Serial RAMs (via DSP SPI)

Note1: See the CLKPHASE documentation for the SPI module to make sure you have the correct phase/edge selected for your SPI clock with respect to your SPI data. Recommended: use twisted pair (with GND wire) on the SPI clock signal to minimize noise.

Test1. Write 0xAA to all locations in SRAM and then read all locations and check if 0xAA is present. If this passes write "0xAA OK!" to your LCD display, else write "0xAA Error!" and place the first error data value and address in memory for future viewing.

Test2. Write 0x0000 to the first memory location, next write 0x0001 to the next location and then 0x0002 to the next and so on to all remaining memory locations. Read them back and verify no errors are found. If no errors are found, display "Inc Test No Error" else display "Inc Test Error" and again store the first defective address and data value.

Note2: For the tests above, write your main program such that it waits for a push button pressed to begin writing to your RAM and then you must push the button again to begin reading from your RAM.

Lab Point Break-down
Part I: In-Lab Quiz (20 minutes) 30%
Part II: Print out/bring code to lab 5% & working ‘C’ program 25%
Part III: Print out/bring code for all three tests to lab 5% & all three tests functional 35%