Exercise 2 – Due Thursday 9/26 before noon

In last week's exercise you got hands on experience with actual hardware and some simple sample code. This week we expand on that by improving the cUartEcho application with printf debugging functionality. You can use either the original cUartEcho application or your modified version.

A) Embedded programming is often about bridging the gap between hardware and software in the form of writing device drivers. This means reading data sheets and understanding the special function registers.

   1. In the **MSP430F1611 User Guide**, read the following pages well enough to complete the rest of the exercise:

      13-1 to 13-4
      13-8 to 13-11
      13-16 to 13-18
      13-21 to 13-30 (cursorily)

   2. What values should the registers **U1TCTL**, **U1BR0**, **U1BR1**, **U1MCTL** be set to in order for **USART1** to run at 115200 baud rate, while using **SMCLK** as the baud rate clock source and with **BRCLK** = 1MHz? Answer with hexadecimal assignments in a text file.

B) **printf** can be too resource intensive to be deployed in production code. However, for debugging purposes it can be an invaluable tool.

   1. Include **stdio.h** in the **cUartEcho** application and insert a **printf("hello world\n\r")** line before the while-loop. Don't compile.

   2. Since **printf** is platform independent it requires a platform dependent function to handle the actual printing of characters. This function:

      int putchar(int character)

      is the undefined reference the compiler complained about above.

   3. Implement the **putchar** function using a buffer to hold the characters passed and with a buffer size that is easily changeable. **putchar** should return the character it was passed if the buffer wasn't full or **-1** otherwise.

4. Add the interrupt handler for the **UART1TX_VECTOR** (the syntax is completely analog to the **UART1RX_VECTOR**) and use it to process the buffer, i.e., print the next character in the buffer when the interrupt fires. Remember to enable the TX interrupt if not already done and only process the buffer if it is not empty.

5. In the **putchar** function, make it so that the first character triggers the interrupt enabled processing of the buffer.

6. Change the **UART1RX** interrupt service routine (ISR) to use **putchar** instead of writing directly to **U1TXBUF**.

7. Verify your implementation by testing the output of **printf**, both with strings shorter and longer than the buffer can hold.

8. Insert comments (a couple of sentences) about the race conditions present and how you avoid them. Upload the c-file as your answer.