Lab 2 – Debugging and Serial Interfaces

Overview

In this lab, you will be enabling a 'serial debugging terminal' that you will be using in all later labs. To enable the serial port (USART1) on the AT90USB chip on the Wunderboard, you will need to read and understand the datasheet section about the USART.

Once the USART is enabled, you will write a simple program to demonstrate its function. Additionally you may need to install a terminal program to enable you to send and receive data over the serial port.

Prelab

1. What do the following register and bit names stand for in reference to the AT90USB USART (pages 181 to 205 of the datasheet found on the lab webpage)? Describe what you think each one does with one sentence for each.
   a. USCRA
   b. UDREn
   c. TXENn
   d. UCSZn1:0
   e. U2Xn
2. Explain in words what the bit UDREn in UCSRnA does. How could this be used to ensure that you do not overwrite USART data that has yet to be sent?
3. What does the following code do? Why would you write code in this fashion rather than using the { and } in while() loops?

   while (UDR1 == 0b10101010);

4. Complete the following table:

	Binary	Decimal
~(0b00000010)		
0b00010000 & 0b11101111		
(1 << 4) | 0b01010000		
~(1<<2) & 0b11111111		
0b01010101 | 0b11000011		

Procedure

1. Download the skeleton code for this lab from the lab webpage. In the main.c file you will find 3 function stubs that need to be filled out to allow for your program to write to value out over the USART. Read below to understand what each function does and to get some hints for each function:

   - unsigned char InitializeUART ()
     This function should setup the USART1 on the Wunderboard to transmit at 9600bps in 8N1 mode (8 data bytes, No Parity, and 1 Stop bit). This function should return a 1 if it fails and a 0 if it does not.

     HINT: The variables you NEED to manipulate are already listed in the skeleton code. Look each of them up in turn to understand them. You should set the U2X1 bit to get a better error percent (pg. 203).

   - unsigned char SendByteUART (unsigned char data)
     This function takes a single 8 bit value and transmits it over USART1. Before send the value over the USART, the function should check a flag to make sure that the USART is ready for a new byte to be sent. If the byte is not ready to be sent, the function should immediately return a 1. If the value is sent, it should return a 0. The value should be copied into UDR1 with in the function if the USART is ready for more data.

     HINT: Look at the UDREn bit in the UCSRnA register to see if you can write a new value.

   - unsigned char SendStringUART (unsigned char *data)
     This function takes a pointer to a string of single 8 bit values and transmits it over USART1. Before sending the value over the USART, the function should check a flag to make sure that the USART is ready for a new byte to be sent. If the byte is not ready to be sent, the function should immediately return a 1. If the value is sent, it should return a 0.

     HINT: Use the function strlen() (found in string.h) and SendStringUART() that you wrote.

2. Once you have written each of the functions, be sure to test them and show that you can see values on a computer connected to the Wunderboard 'Serial Connection' USB connector.

   To better understand the Wunderboard USART and how to view its data on different operating systems, refer to the Wunderboard Usage guide on the webpage.

3. Now write a simple main line that changes the pattern of lights on the LED array based on what the switches and buttons on Port A are set to. The value displayed on the LEDs should also be written to the USART with a short description. The output to the USART should look something like:

   Port A is: 'x'
   where 'x' is the value of PINA.

4. Finally, enclose all of the USART commands inside of an #ifdef structure based on the #define for a constant called DEBUG (all capital letters). This will make sure that the serial functionality will only be compiled into the program while the constant DEBUG is defined. It would look something like:

```
#define DEBUG

#ifdef DEBUG
    Serial code goes here...
#endif
```

Demonstrate and Submit code

Show your code and program to your TA. They will want to see the code running with the #define DEBUG inserted and the code with it commented out as well. You will need to turn in a printed copy of your source code when you demonstrate the working code.

Study Questions

1. You wrote a function for sending data that looked at the UDREn bit to decide if data could be sent. Write a new function that check if it is ok to send new data using the TXCn bit in UCSRnA instead. Why is this method not as good as the method you did use in lab?

Lab Summary

Task	Completed?
Prelab	2pts.
LED outputs match switches	4pts.
Outputs Serial	2pts.
#define works	2pts.
Study Questions	2pts.