CSC211
Laboratory D: Serial Communication

Background Information

In this laboratory, you will write programs that allow two AVRs to communicate with one another.

**USART serial communication:**

**USART** (Universal Synchronous and Asynchronous Receive and Transmit) is a serial communication protocol that allows computers to send data to one another. **Serial** communication means that data is transmitted one bit at a time. The bits are organized into **frames** that contain a few control and error-correction bits in addition to the data that is intended for transmission. For two computers to communicate this way, they must agree on several parameters regarding the makeup of the frames and the rate at which data will be transmitted.

Serial communication is accomplished by shift registers in each unit. Suppose we have two computers, and one is to send data to the other. The sender places a data frame in a shift register, and then shifts the bits of the frame out of the register and onto the communication line one at a time. The receiver receives the bits one at a time from the communication line and shifts them into a receiving shift register.

This process is controlled via interrupts in both computers. The USART unit in the sender generates a **TXC** (transmission complete) interrupt when all of the bits of the current frame have been sent, indicating that it is ready to be given another byte to send. The USART unit in the receiver generates a **RXC** (receive complete) interrupt when all the bits of a frame have arrived, indicating that the data is correct and ready for use.

**USART registers:**

There are several registers in the AVR's I/O Register space that control USART communication, as follows. For more details, see pages 155-163 of the datasheet.

1. **UBRRH/UBRRL**: This pair of registers is used to set the transmission speed, called the baud rate. The sender and receiver programs must agree on this. To set both for 9600 baud:
   - \( UBRRH = 00 \)
   - \( UBRRL = 06 \)

2. **UCSRC**: This register specifies the makeup of each data frame, so both the sender and receiver should again be given the same values.
   - \( bit7 = 1; UCSRC \) and \( UBRRH \) share space, when \( bit7=1 \) we access UCSRC.
   - \( bit6 = 0; \) asynchronous
   - \( bit5 \) and \( bit4 = 1; \) odd parity
   - \( bit3=0; \) 1 stop bit
   - \( bit2 \) and \( bit1=1; \) 8-bit data
   - \( bit0=0; \) clock parity bit=0

3. **UCSRB** (for sender program):
   - \( bit6 = 1; \) enable TXC (transmission complete) interrupt
   - \( bit3 = 1; \) enables transmit functionality of the USART unit
   - all other bits should be set to 0

4. **UCSRB** (for receiver program):
   - \( bit7 = 1; \) enable RXC (receive complete) interrupt
   - \( bit4 = 1; \) enables receive functionality of the USART unit
   - all other bits should be set to 0
5. **UCSRA** (for sendor program):
   - bit5 = 1; enables UDR register (described next)
   - all other bits should be set to 0

6. **UDR** (USART I/O Data Register):
   This register is used in both programs to store the data that is to be sent or has just been received.

   In the receiver, when an incoming byte is complete the USART unit automatically places it in the **UDR** register and generates an **RXC** interrupt. You can then read the data with a instruction like “IN r4, UDR”.

   In the sendor, to send a byte of data you just need to place it in the **UDR** register with an appropriate “OUT” instruction. This triggers the USART unit to begin transmission, and when the transmission is complete it will generate a **TXC** interrupt. This is your cue to place the next byte in **UDR**.

**NOTE:** The AVR datasheet indicates that the USART settings should be initialized BEFORE global interrupts are enabled.

**USART Interrupts:**
To use the **RXC** and **TXC** interrupts, you will need to write interrupt services routines and jump to them from the corresponding positions of the interrupt vector. See page 54 of the ATmega 8515L Datasheet to locate the appropriate position in the interrupt vector for each of these interrupts.

**AVR pins associated with USART Rx and Tx:**
The receive (RXD) and transmit (TXD) signals use the same pins as PD0 and PD1. When these functions are enabled as described above, they override the normal function (as bits of Port D) for these pins. The pins can be accessed on the STK500 development board as follows:

   - RXD: Expansion header 1, pin 38
   - TXD: Expansion header 1, pin 37

To allow one AVR to communicate with another, we connect the ground pins on the two AVRs together and then connect the RXD pin on the receiver to the TXD pin on the sender.

**Laboratory Exercises**

1. Write a program that causes your AVR to receive data via the USART unit and display that data on the LEDs as it is received. To do this:
   - Set up an interrupt vector that recognizes and responds to the USART RXC interrupt, located at address $009 of program memory.
   - Initialize the USART for receiving data. (See background information above.)
   - Set up the stack and enable global interrupts. Per the AVR datasheet, global interrupts should be enabled AFTER initializing the USART.
   - Write a main loop that does nothing but spin its wheels while waiting for interrupts. You are writing an event driven AVR program!
   - Write an interrupt service routine that is triggered by the USART RXC interrupt. This routine should read the value just received in register UDR and output it to the LEDs for display. You do not need to introduce a delay here; that will be done by the sending program.

I will set up a test station where I can help you test your program. (I have already completed the programs below, which send data via USART.)
2. Write a program that displays a recognizable repeating sequence of lights and also transmits that sequence via USART. To do this:

- Set up the various registers needed to control USART transmission via interrupts. (This will be similar to your USART receive program, but some of the settings will differ. See the background information above for the necessary settings.)
- To initialize the transmission process, write the initial value in your light sequence to the UDR register and display it on your LEDs as the last step of your setup code. (This is needed since the TXC interrupt is generated when the USART unit finishes transmitting each value. Thus, we need to load the first value into UDR without being prompted to by an interrupt signal.)
- Write a main loop that does nothing but spin its wheels while waiting for interrupts.
- Write an interrupt service routine that is triggered by the USART TXC interrupt. This routine should write a new byte to the UDR register, display it on the LEDs, and delay for an appropriate length of time to allow the light pattern to be visible.

Again, I can help you test your program when it is ready.

3. Write a third program that both transmits and receives data via USART. Your program should generate a sequence of bytes (that represent a light pattern) and transmit that sequence via USART. However, the program should not display this pattern on the local LEDs. Rather, it should display the data that it receives via USART.

This should allow two AVRs that are both programmed this way to generate light patterns and display them on the LEDs of the other development board.