CPE 301L
Microcontroller Based System Design
Laboratory Exercise #7
USART in AVR
Department of Electrical and Computer Engineering
University of Nevada, at Las Vegas

Goals:
The goal of this lab is to transmit data to a PC using the Serial communication standard RS232.

Equipment Usage
For this lab the following equipment will be used:

- Atmega328P
- MAX232/SP232
- Female to Female Serial Cable (Null Modem)
- Terminal Program(TeraTerm or Hyper Terminal)

Background:

RS232
RS232 is the name for a set of standards used for serial communications between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE). Originally the DB-25 used to be the standard connection. The DB-9 connection later replaced the DB-25. This was later replaced by USB connections because the RS232 limited speed and large voltage swings. For this lab we will be using the DB-9 connector. Normally connection is done with a DTE connecting to a DCE. For this lab we will use a DTE to DTE connection which is otherwise referred to as a null modem connection.
USART

AVR supports 4 Universal asynchronous receiver/transmitter modes: Normal asynchronous, double speed asynchronous, master synchronous and slave synchronous. For PC connections we will focus on using only the asynchronous modes. There are 5 registers that control the settings for USART in AVR.

UBRR: This register controls the baud rate at which the signal will be transmitted. The baud rates must be the same on both the receiving and transmitting end. The baud rate selected for the microcontroller must match the baud rate of the serial port settings. This register is split into high and low components to accommodate higher baud rates.

UDR: This is the data register which stores the data the will be transmitted or the data that has been received.

UCSRA/B/C: These registers deal with controlling the various settings for the USART connection. This includes the parity bit, character size, mode (asynchronous or synchronous) and switching between transmit mode and receive mode.

Formulas:

Calculating the Baud rate is determined by which USART mode the microcontroller currently has active. The table below shows the formulas for calculating the baud rate and the UBRR register value that is desired.

<table>
<thead>
<tr>
<th>Operating Mode</th>
<th>Equation for Calculating Baud Rate</th>
<th>Equation for Calculating UBRRn Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous Normal mode(U2Xn = 0)</td>
<td>BAUD = ( \frac{f_{osc}}{16(UBRRn + 1)} )</td>
<td>UBRRn = ( \frac{f_{osc}}{16 \times BAUD} - 1 )</td>
</tr>
<tr>
<td>Asynchronous Double Speed mode (U2Xn = 1)</td>
<td>BAUD = ( \frac{f_{osc}}{8(UBRRn + 1)} )</td>
<td>UBRRn = ( \frac{f_{osc}}{8 \times BAUD} - 1 )</td>
</tr>
<tr>
<td>Synchronous Master mode</td>
<td>BAUD = ( \frac{f_{osc}}{2(UBRRn + 1)} )</td>
<td>UBRRn = ( \frac{f_{osc}}{2 \times BAUD} - 1 )</td>
</tr>
</tbody>
</table>

Baud Rate and UBRRn calculation table

Microcontroller Fuses:

In the past we have spoken on internal features such as clock speed. However, most of these features cannot be accessed via code. To adjust settings such as clock speed you must access and modify the fuses of the microcontroller. You can access these fuses by opening up the AVR programming menu and clicking on the tab labeled fuses. Most of the settings are default configurations and do not need to be modified. For the purpose of this lab you should know how to change the internal clock speed. By clicking on the box marked CKDIV8 you can change the internal clock from 1MHz to 8Mz and vice versa.
**Prelab:**

**Design 1:** Modify the transmit code so that it will work for the Atmega328 instead of the Atmega32. Use an internal clock of 8MHz.

**Code for transmitting data**

```c
#include <avr/io.h>
void usart_init (void) {
    UCSROB = (1<<TXEN0);
    UCSROC = ((1<<UCSZ01)|(1<<UCSZ00)|(1<<UMSEL00));
    UBRRL = 0x33;
}
void usart_send (unsigned char ch) {
    while(! (UCSR0A & (1<<UDRE0)));
    UDR0 = ch;
}

int main(void) {
    unsigned char strLenght = 30;
    unsigned char i = 0;

    usart_init();
    while(1) {
        usart_send(str[i++]);
        if (i>=strLenght) 
            i = 0;
    }
    return 0;
}
```

**Code for receiving data**

```c
//Program the AVR to receive bytes of data serially and place them on Port B
//Set baud rate at 9600, 8-bit data, and I stop bit
int main(void) {
    DDRB = 0xFF;
    UCSROB = (1<<RXEN0);
    UCSROC = (1<< UCSZ01)|(1<<UCSZ00)|(1<<UMSEL00);
    UBRRL = 0x33;
    while(1) {
        while(! (UCSR0A & (1<<RXC0)));
        PORTB = UDR0;
    }
    return 0;
}
```
**Lab Experiments:**

**Experiment 1:** Modify the above code to work for the Atmega328P so the microcontroller will transmit data to the terminal. Set your Baud rate to 9600. Use the schematic below as a guide for your connections.

**Experiment 2:** Modify your code to receive data and transmit the collected data back.

**Experiment 3:** Write a program where the microcontroller will take any letter and transmit the opposite case of it back to the terminal. For example, if the letter ‘T’ is sent from the terminal, the microcontroller should send back the letter ‘t’ to the terminal.

**Post-Lab Deliverables:**

1) Submit your completed project report including your working code.
2) Submit a PCB layout Design of your circuit using Altium