Willis Allstead
CPE 301-1001
Lab #8
November 11, 2016

Assignment Description:

In this lab we were supposed to learn how to use the serial functions such as kbhit, getchar, putchar. We were then supposed to use these to return the hex value of an input char. We were supposed to print “0x” then the input char converted to hexadecimal then a newline character.

Problems Encountered:

The number one problem I had in this lab was converting from an unsigned char to hexadecimal representation. What seemed to be a simple problem turned into a multi-hour attempt at this. I eventually ended up not being able to do this, but did see that I was on track to doing it. For example when I type “a”, I get back “0x`”.  “`” in hex representation is 60. That 6 in 60 is actually the first digit of the hex-number that represents “a” in ascii, 61. This pattern continued with other characters. I just could not get to a final product in the time I had. I believe that with a little more time I could figure it out.

Lessons Learned:

I basically just learned that I need to understand ascii-hex-binary conversion a little bit better when it comes to actually physically using them. Sure, I know how to convert them all on paper and how converting between them works there, but I have never had to actually do this in the real world so it felt a bit foreign.

Description of Completed Lab: (compiled code below)

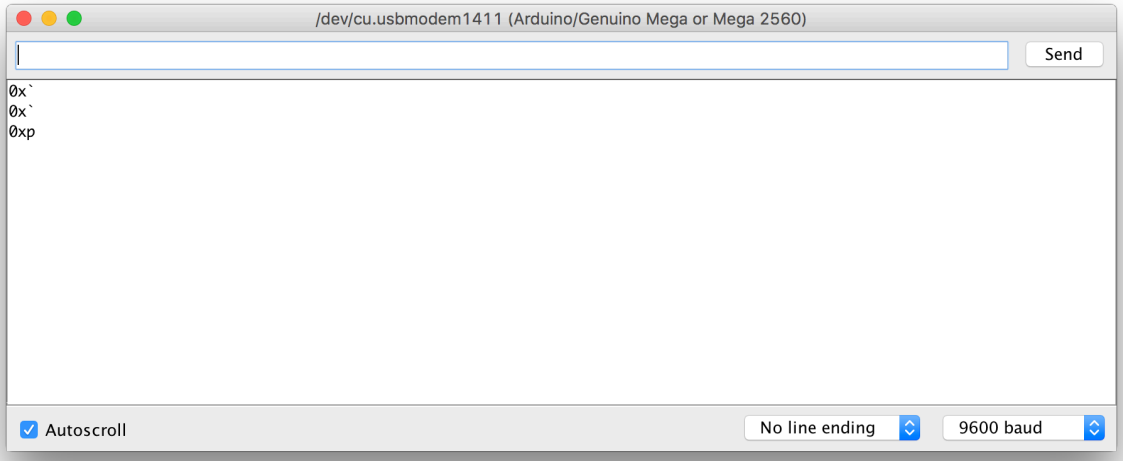
ANSI-C_Lab_8 §

```
#define RDA 0x80
#define TBE 0x20
volatile unsigned char *myUCSR0A = (unsigned char *) 0x00C0;
volatile unsigned char *myUCSR0B = (unsigned char *) 0x00C1;
volatile unsigned char *myUCSR0C = (unsigned char *) 0x00C2;
volatile unsigned int  *myUBRR0  = (unsigned int *)  0x00C4;
volatile unsigned char *myUDR0   = (unsigned char *) 0x00C6;

void U0init(int U0baud);
unsigned char U0kbhit();
unsigned char U0getchar();
void U0puthex(unsigned char U0ptohex);
void U0putstr(unsigned char *string);
void U0putchar(unsigned char U0pdata);

void setup()
{
 // initialize the serial port on USART0:
 U0init(9600);
}
void loop()
{
  unsigned char cs1;
  while (U0kbhit()==0){}; // wait for RDA = true
  cs1 = U0getchar();    // read character
  U0puthex(cs1);     // echo character
}

// function to initialize USART0 to "int" Baud, 8 data bits,
// no parity, and one stop bit. Assume FCPU = 16MHz.

void U0init(int U0baud)
{
 unsigned long FCPU = 16000000;
 unsigned int tbaud;
 tbaud = (FCPU / 16 / U0baud - 1);
 *myUCSR0A = 0x20;
 *myUCSR0B = 0x18;
 *myUCSR0C = 0x06;
 *myUBRR0  = tbaud;
}
//
// Read USART0 RDA status bit and return non-zero true if set
//
unsigned char U0kbhit() {
  unsigned char rdaIsSet = *myUCSR0A & RDA;
  return rdaIsSet;
}
//
// Read input character from USART0 input buffer
//
unsigned char U0getchar() {
  while (U0kbhit() == 0) {}; // wait until data available (double-checking for safety)
  return *myUDR0;
}

void U0puthex(unsigned char U0ptohex) {

  while ((*myUCSR0A & TBE) == 0) {};
  U0putchar('0');
  while ((*myUCSR0A & TBE) == 0) {};
  U0putchar('x');

  unsigned int firstChar = U0ptohex & 0b11110000;
  while ((*myUCSR0A & TBE) == 0) {};
  U0putchar(firstChar);
  while ((*myUCSR0A & TBE) == 0) {};
  U0putchar('\n');

}


//
// Wait for USART0 TBE to be set then write character to
// transmit buffer
//
```

```
void U0putstr(char *string) {
  while(*string) {
     U0putchar(*string);
     string++;
  }
}

void U0putchar(unsigned char U0pdata) {
  unsigned char transmitterBufferEmpty = *myUCSR0A & TBE; // "If UDREn is one, the buffer is empty, and therefore ready to be written."
  while (transmitterBufferEmpty==0) {}; // wait until it is ready
  *myUDR0 = U0pdata; // write character
}
```

```
Done uploading.

Sketch uses 972 bytes (0%) of program storage space. Maximum is 253,952 bytes.
Global variables use 19 bytes (0%) of dynamic memory, leaving 8,173 bytes for local variables. Maximum is 8,192 bytes.

83
```

Below is what the Serial Monitor looked like after entering a, a, s.



As you can tell I was on my way there. I even had a printStr function lined up and ready for use when I had correctly decoded the binary number. I would have then probably gotten rid of the putHex function and done everything using the putStr.