High-Level USART Functions

- Transmit a string
- Read a line with edit functionality
- Set ANSI Colors
- Move the cursor home
- Clear the current line

High-Level USART Functions

- Transmit a String

```
void USART0_TransmitString( char *str )
{
        //send a string to the serial port
        int count;
        for (count=0; count<=(strlen(str)-1); count++)
        {
                if (str[count] != '\0')
                {
                  /* Wait for empty transmit buffer */
                  while ( !(UCSR0A & (1<<UDRE0)) )
                      ;
                  /* Start transmission */
                  UDR0 = str[count];
                } else
                {
                        count=strlen(str); //force exit of loop at end of string
                }
        }
}
```

High-Level USART Functions

- Read a line with edit functionality

```
char *USART0_ReceiveLine ( void )
{
//Pull a full line from the buffer
int count;
char outbuf[255];
count=0;
        while (count<255)
        {
                outbuf[count] = USART0_Receive();

                if (outbuf[count] == 10 || outbuf[count] == 13) //if we get a CR, exit the loop
                {
                        outbuf[count]='\0'; // terminate the string
                        count=254; // force exit of the loop
                } else
                {
                        USART0_Transmit(outbuf[count]); //give echo of keystrokes
                }

                if (outbuf[count] == 8) //if we get a backspace, handle it
                {
                        count--;
                } else
                {
                        count++;
                }
        }
        return strcat(outbuf,'\0'); //convert char[255] to a true char *
}
```

High-Level USART Functions

- Set ANSI Colors

```
/* The following subroutine will trigger an ansi attribute such as the following:
 *
 * 0    Reset all attributes
 * 1    Bright
 * 2    Dim
 * 4    Underscore
 * 5    Blink
 * 7    Reverse
 * 8    Hidden
 *
 */
void USART0_SetANSIColor( int ansicolor )
{
        //create a blank string to use for our buffer
        char tempbuffer[255];
        char secondtemp[255];
        tempbuffer[0]=27; //all ansi strings start with escape
        tempbuffer[1]='\0';
        strcat(tempbuffer,"[\0"); //don't ask...
        strcat(tempbuffer,itoa(ansicolor,secondtemp,10));
        strcat(tempbuffer,"m\0"); //see above
        USART0_TransmitString(strcat(tempbuffer,'\0'));
}
```

High-Level USART Functions

- Move the cursor home

```
void USART0_MoveCursorHome( void )
{
        //create a blank string to use for our buffer
        char tempbuffer[255];
        char secondtemp[255];
        tempbuffer[0]=27; //all ansi strings start with escape
        tempbuffer[1]='\0';
        strcat(tempbuffer,"[0;0H\0"); //don't ask
        USART0_TransmitString(strcat(tempbuffer,'\0'));
}
```

High-Level USART Functions

- Clear the current line

```
void USART0_ClearLine( void )
{
        //create a blank string to use for our buffer
        char tempbuffer[255];
        tempbuffer[0]=27; //all ansi strings start with escape
        tempbuffer[1]='\0';
        strcat(tempbuffer,"[K\0"); //don't ask
        USART0_TransmitString(strcat(tempbuffer,'\0'));
}
```

Ternary Conditional Assignments

- What are they?!
- Why would I use them?
- Code Sample Before
- Code Sample After

Ternary Conditional Assignments

- Ternary assignments allow you to assign a variable differently on the basis of a conditional statement

In other words...

If condition is true, variable = one value,
else variable = other value.

Ternary Conditional Assignments

- Shorter Code
- Less ambiguous statements (once you're familiar with them)
- Dazzle your friends with C statements they don't understand

Ternary Conditional Assignments

- Before

```
if (outbuf[count] == 8) //if we get a backspace, handle it
{
    count--;
} else
{
    count++;
}

7 Lines !
```

Ternary Conditional Assignments

- After

```
//if we get a backspace, handle it
count = (outbuf[count] == 8)?count--:count++;
```

2 Lines!