Programming Assignment #3 - String Manipulation and USB-UART
CSE30 - Computer Organization and Systems Programming
Winter 2011

Overview

The goal of this assignment is to extend the string builder application created in PA2 to communicate over USB through a UART Serial Bridge. The project will work as before with a few additions. Most notably, data for the string can now be inputed through USB via the HyperTerminal.

Project Description

You need to first understand how to use the USB-UART. This sets up a UART connection through USB. Information about how to set this up can be found at: http://www.cypress.com/?docID=26782. The required USB-UART component files are at: http://www.cse.ucsd.edu/~kastner/cse30/EP60246.zip.

First open the USB-UART example project to get setup and get somewhat familiar with the USB module. Even though most of the implementation has been abstracted away, go through the example code, documentation and data sheet very carefully to see what are the API calls used to perform I/O operations.

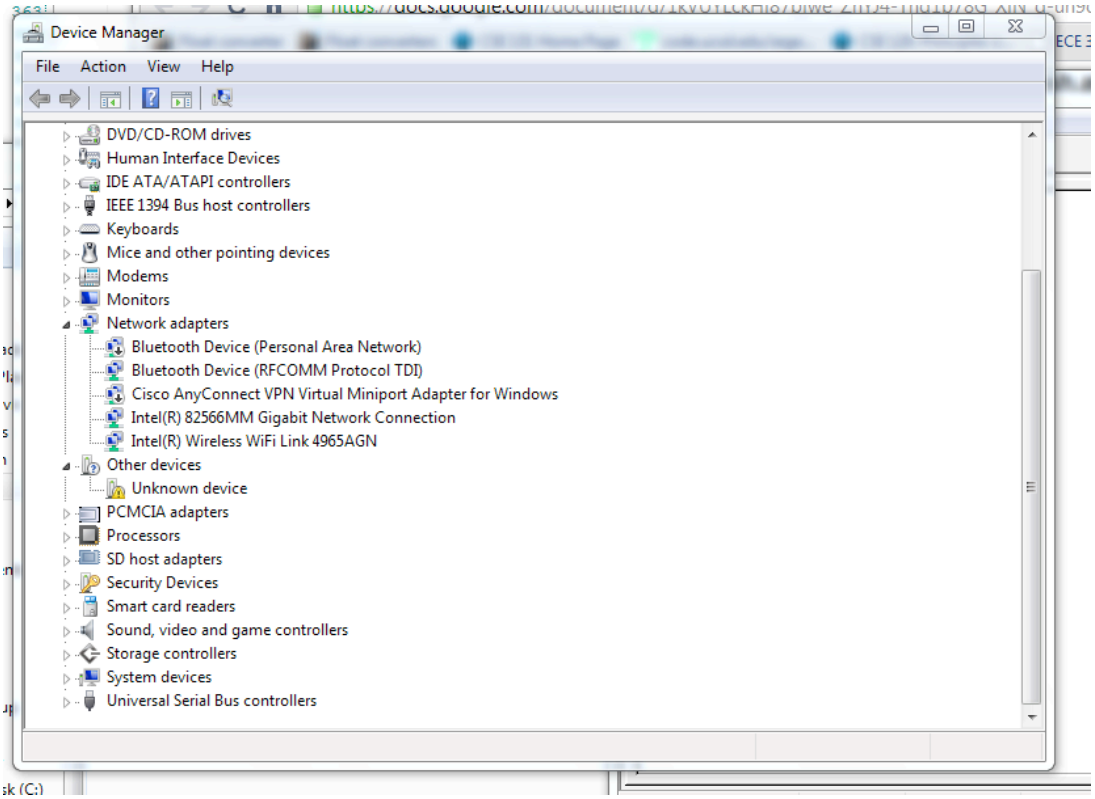
Choose and use available PSoC5 alternatives instead of the PSoC3 components. The PSoC5 components are available under an appropriately named directory. (Choose same device with the serial number CY8C5588AXI-06) . First you need to build and program this example to the board as you have done in PA 1 and PA 2. After you are done programming, unplug the Miniprog cable from PSoC, and plug the USB cable to the USB port of the board. Then, you will be prompted for device installation. Install the device driver as below.

Device Installation

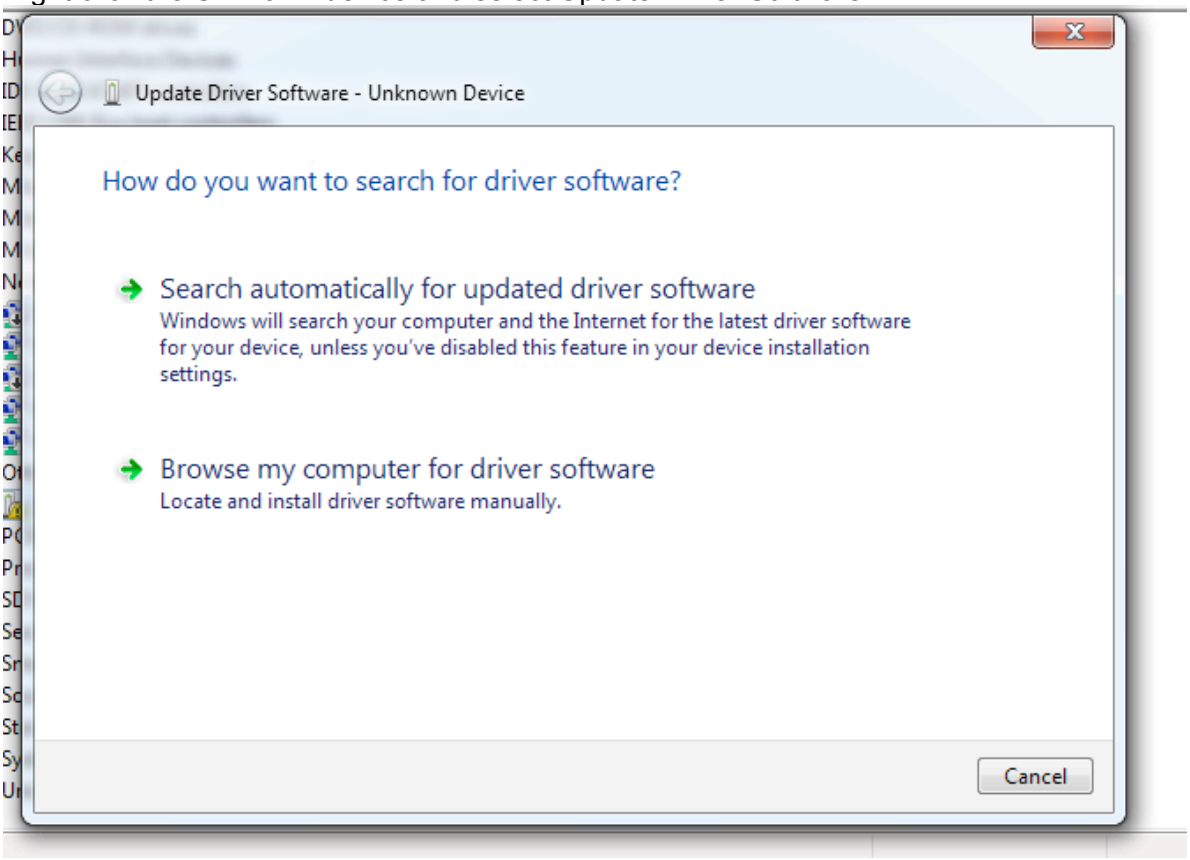
Once this is installed, you should connect your PSoC board via USB mini interface on the side of the board (you can use the same USB cord that connects to your programmer). This will require setting up your computer to register your PSoC board as a COM port.

When windows prompts for installing a new hardware, choose select the driver to install by navigating to wherever you downloaded this .inf file (http://www.cse.ucsd.edu/~kastner/cse30/USBUART_1_CDC.inf). In case the device installation fails manually install the driver by following the steps below (example given for Windows 7) :

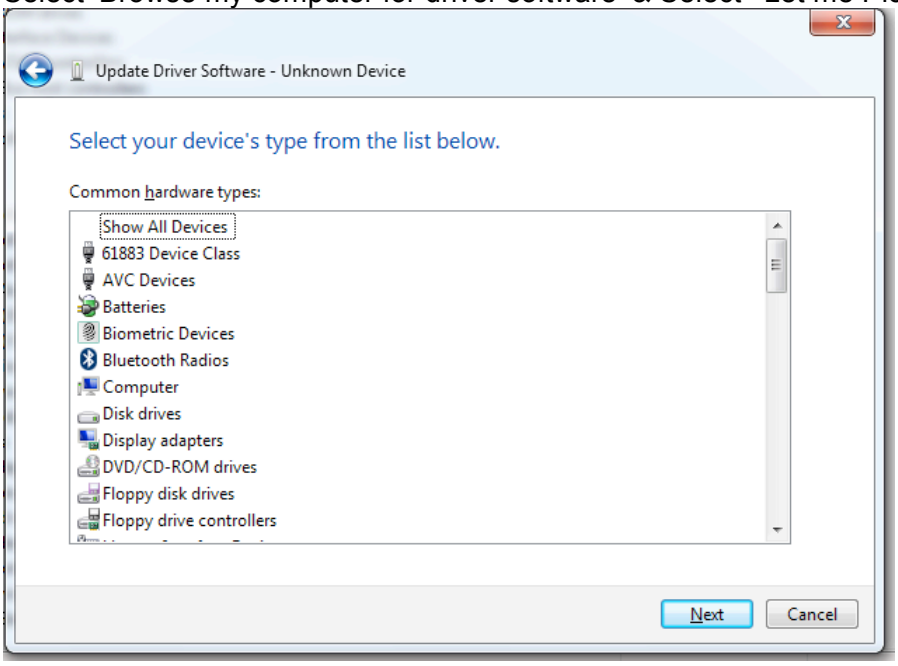
Navigate to device manager :



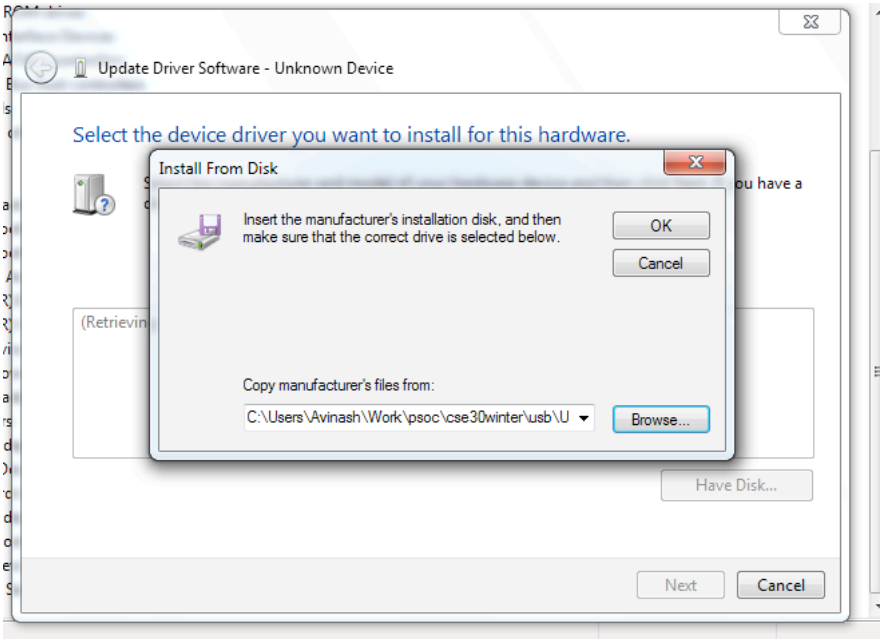
Right click the Unknown device and select Update Driver Software



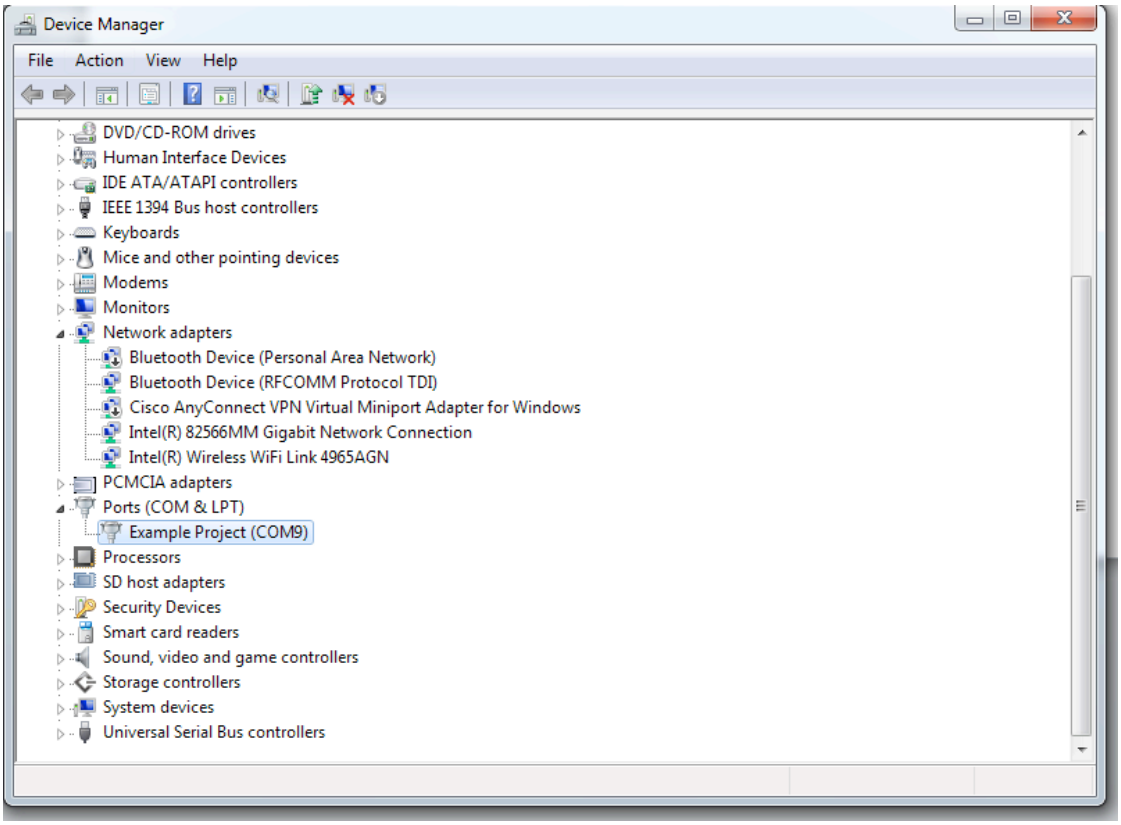
Select ‘Browse my computer for driver software’ & Select ’ Let me Pick …’ -> Show all Devices



In the next screen, select Have Disk and select the .inf file provided from wherever it has been downloaded to.



Ignore any warnings about unsigned drivers and complete the installation. The device should now be detected as a serial device named Example Project

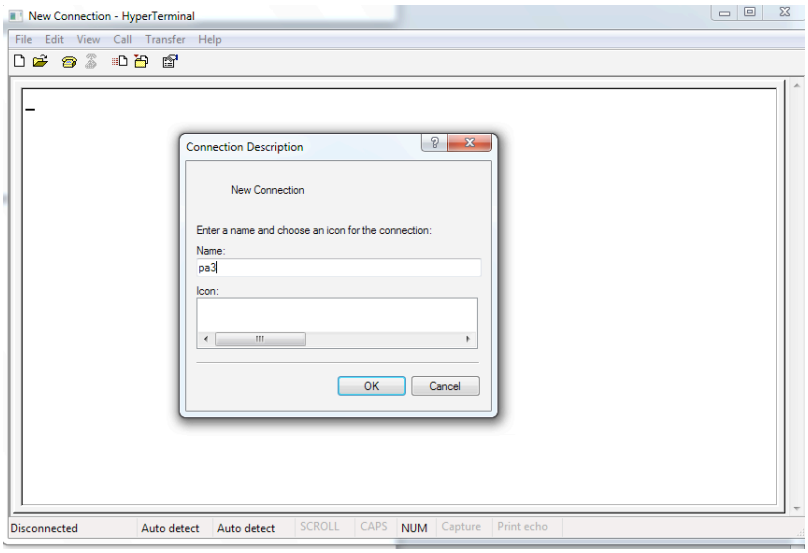


Setting up Hyperterminal

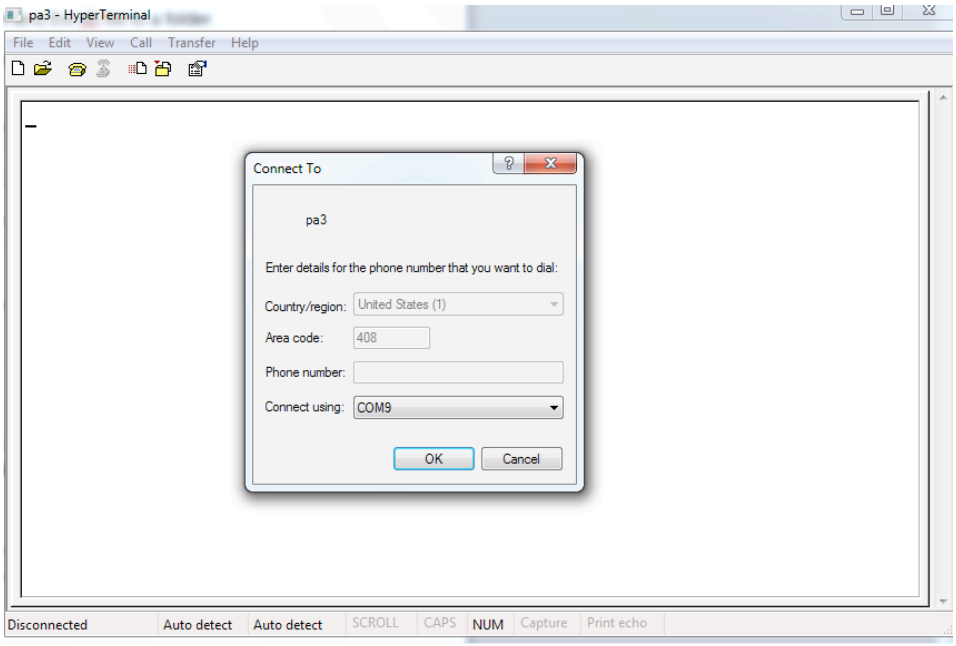
You will use Hyperterminal to talk from your keyboard to the PSoC board over a COM port. You should be able to use any hyperterminal program. You can find the one below at: http://www.cse.ucsd.edu/~kastner/cse30/hypertrm.zip

Extract the attached hyperterminal program and the dll file to a folder.

Open Hyperterminal and proceed to create a new connection – name it whatever you like.



In the next screen, select the COM port matching the port Example Project was assigned

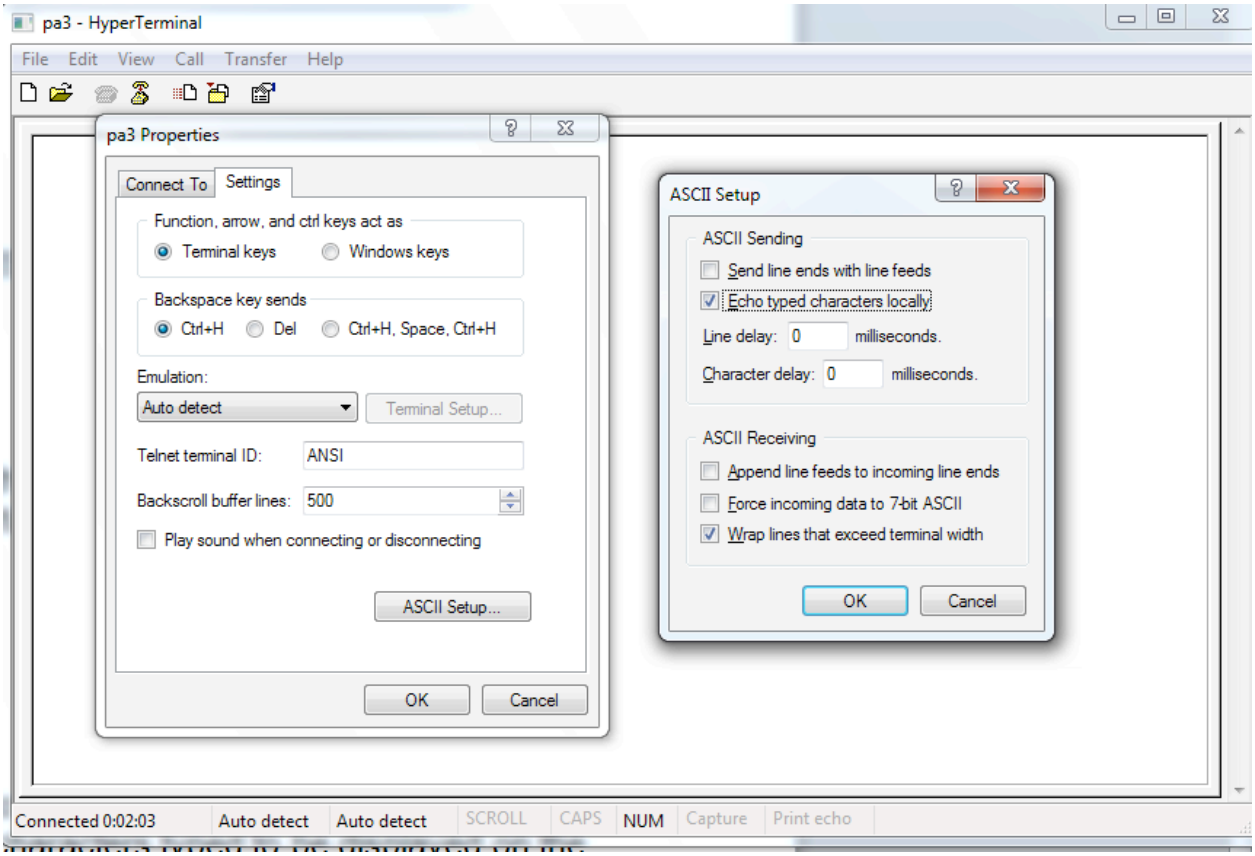


In the port properties window ensure the following settings are entered :

- Bytes per second : Leave as default
- Data Bits : 8
- Parity : None
- Stop Bits : 1
- Flow Control : None

Now the connection is setup.

For the example project the ‘local echo’ setting needs to be turned off, which is the default. The local echo causes characters typed to be displayed on the local terminal. Local echo should be enabled when you are doing the assignment. To enable it navigate to File->Property->Settings->ASCII settings and check the box that says ‘Echo typed characters locally’



Programming Assignment

Now that you are setup and have seen the basic workings of the USB-UART module, extend the functionality of the ‘string builder’ project created in Programming Assignment 2.

Now in addition to being able to enter strings through the CapSense inputs as defined in PA2, pressing both CapSense buttons P0_5 and P0_6 simultaneously should now read in the first 16 characters from the USB input buffer into the string currently being edited (and display it on the LCD as well). Once this is done, the program should exit from edit mode.

Events that accompany button presses should happen only once per press, as was defined in PA2. Similarly when in edit mode, pressing buttons corresponding to other

strings can either cause the program to exit into display mode or continue to edit the other strings.

In addition to taking input from the keyboard, you must also determine some properties of the strings. For this you need to wire up LED 2, 3 and 4 (LED1 still indicates if we are in Edit Mode). These can use any port that you choose.

LED2 should now light up if any of the strings are a permutation of any other string currently in memory (excluding the empty string). This means that any two of the strings have the same length and exactly the same characters. For example, if String1 = “ABC” and String2 = “BCA” then LED2 should light up. Write a function turnLED2On that returns TRUE/FALSE depending on the values of the three strings. The function prototype should be:

```
int turnLED2On(char * string1, char * string2, char * string3);
```

You must write all of the code for this function, i.e., you must not use any library functions (e.g., from <string.h>). This is because you will translate these functions to ARM in a future programming assignment.

LED3 should light up if any of the strings are equal to any other strings in memory (again excluding the empty string). Write a function turnLED3On that returns TRUE/FALSE depending on the values of the three strings. The function prototype should be:

```
int turnLED3On(char * string1, char * string2, char * string3);
```

Again you must write all code without any library functions.

LED4 should light up if any of the strings are a substring of any of the other strings. For example, if String2 = “123” and String3 = “ab123c”, then String2 is a substring of String3. Write a function turnLED4On that returns TRUE/FALSE depending on the values of the three strings. The function prototype should be:

```
int turnLED4On(char * string1, char * string2, char * string3);
```

Last time, no library functions.

Hints:

- Avoid using the Enter / Return key when interacting with the hyperterminal. Transmit the carriage return and line feed (0x0D, 0x0A) characters once you read from the buffer instead. This will advance the hyperterminal to the next line and make it easier to understand what is going to be entered.
- Follow the example lab and instructions to get setup very carefully
- Ignore buffer underflow and overflow conditions. If the buffer overflows, the result should be similar to the string wrapping around (though this will not be tested it is helpful to keep in mind while programming)