**CMPE-250 Laboratory Exercise Nine**

**Serial I/0 Driver**

By submitting this report, I attest that its contents are wholly my individual writing about this exercise and that they reflect the submitted code. I further acknowledge that permitted collaboration for this exercise consists only of discussions of concepts with course staff and fellow students; however, other than code provided by the instructor for this exercise, all code was developed by me.

Joel Yuhas

Performed 3/24/16

Submitted 4/4/16

Lab Section L2

Instructor: Shaaban, Muhammad

TA: Peter Muller

Stephen Moore

Connor Goss

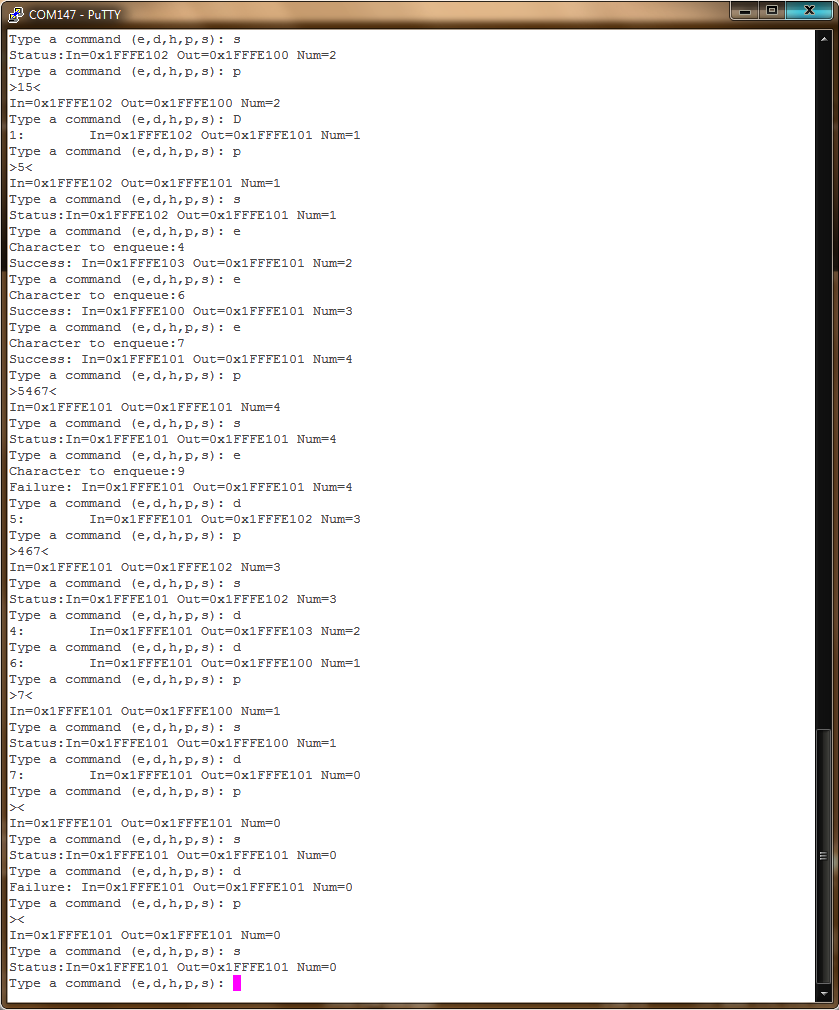
Lecture Section 01

Professor: Sarra, Alessandro

**Documentation:**

Screen Capture With:

* Uppercase and lower case commands
* Commands with an empty queue
* Commands with a partially full queue
* Commands with a full queue
* Commands with a circular buffer operation



Screen Capture Discussion:

This screen capture shows the inputs that were entered by the user and how the program responded to those inputs. The user can be seen enqueueing items and then enqueueing items, all while the program displays in-pointer, out-pointer, and number of characters that were enqueued at the time. These items were useful debugging tools that allowed the user to tell if the queue structure was progressing through as expected.

The circular motion of the queue can also be seen as the in-pointer goes from 0x1FFFE103 back to 0x1FFFE100, as such with the outpointer. Capital letters can be seen entered in, such as when ‘D’ is placed in spot of ‘d’, but the program still executes the same.

List exact memory range (starting address and the address of the last byte) for these components:

* Executable code
* UART0 ISR code
* Constants in ROM (e.g., prompt string)
* RAM used
  + Program queue (buffer and record structure)
  + Receive queue (buffer and record structure)
  + Transmit queue (buffer and record structure)

**Memory Map**

|  |  |  |
| --- | --- | --- |
| **Type** | **Starting Address** | **Last Byte** |
| Executable Code | 0x00000410 | 0x00008FB |
| UART0 ISR code | 0x0000007EF | 0x000007EF |
| Constants in ROM | 0x00000204 | 0x0000002B3 |
| Program Queue (Buffer)  Program Queue (Record) | 0x1FFFE100  0x1FFFE104 | 0x1FFFE103  0x1FFFE115 |
| Receive Queue (Buffer)  Receive Queue (Record) | 0x1FFFE118  0x1FFFE168 | 0x1FFFE167  0x1FFFE17A |
| Transmit Queue (Buffer)  Transmit Queue (Record) | 0x1FFFE17C  0x1FFFE1CC | 0x1FFFE1CB  0x1FFFE1DE |

Memory Map discussion:

The memory map shows the address location of almost all elements in the code. The contents provided above, show the starting address and the last byte of the desired elements. Most of these elements are not determined by the user, but can still be used all the same. Specifically, Addresses such as addresses of all of the program queue buffers and records. These addresses had to be called numerous times in the code, as they store the location of the sections of memory that were used by the queue, which was manipulated by the program.

The constant memory addresses were important as well and were called by the program in order to print out the desired data to the terminal screen. Overall, these memory addresses were important for the user to know because they could be used while debugging, for example, the queue addresses could be placed into the memory map and then their contents could be read off by the user while the program traversed through the queue. This would allow the user the ability to see how the program was manipulating the queue in memory and would allow the user the ability to anticipate and locate errors if need be.