ECGR 4101/5101, Fall 2012: Lab 5

Reading and Writing from the UART and Using Queues

Learning Objectives:

This lab will test your ability to apply the knowledge you have obtained from the last lab, while adding the use of a queue.

Laboratory Assignment:

1. Create a new HEW workspace by creating a new tutorial project.

2. Create a function for initializing your UART. You may use whatever setting you want, just make sure that your terminal program is configured properly to receive the data.
3. Create a function for sending data from the UART.
4. Create a function for transmitting data from the UART.
5. From the class webpage, download the queue.h and queue.c files, and add them to your project. This is done by copying the files into the project directory with all the other c and h files. Once this is done be sure to include queue.h in your main.c file.
6. Take a moment and consider the queue structure. Take a look at the functions available in queue.c for manipulating your queue. Now, in your main function, add a queue which we will be using to hold received bytes from the UART.
7. In your UART receive function, add code that takes the received data and place it in the queue that was created in the previous step.
8. The end goal of our queue is to store every character entered by the user, and echo back everything when the user hits enter on the terminal. Create a conditional statement that checks for the return character, '\n'. If the return character is detected, dequeue everything stored in the queue and transmit it back to the terminal.

   Remember, when passing a queue to a function, such as the ones provided by the queue.c file, the QUEUE data structure is a defined data type that cannot be simply passed, so we must use pointers for manipulating the data with subroutines. When a pointer is expected to be passed to a function, it is expected the memory address of the queue instead of the queue itself. To pass the address of a variable, add an "&" to the front of the variable. So, adding data to the queue may look something like this:
   Queue_Enqueue(&q, data);
   Where &q is the address of queue q.
9. To complete the transmission and reception functionality, we will need to add code that polls for incoming data inside the while(1) loop.
10. Also within the while(1) loop, add code that checks the current size of the queue, and turn on the LEDs in the LED ring according to the size. If the queue size is 3, then 3 consecutive LEDs should be turned on. If your queue size exceeds 10, in which there are no more LEDs to turn on, begin turning them off consecutively. So if your queue size is 11, LED 1 should be off and the rest should be on. Also, be sure to turn all LEDs off when the queue is emptied after receiving a return character.

Lab Report:

1	UART is properly initialized and transmission and reception work properly.	
2	Strings of characters are properly returned to the terminal when a return is entered.	
3	The LEDs properly reflect the queue size.	

Include in your lab report observations and procedure like the following:

The general learning objectives of this lab were . . .

The general steps needed to complete this lab were . . .

Some detailed steps to complete this lab were . . . .

1. *Step one*
2. *Step Two*
3. *. . . .*

Code generated or modified to complete this lab...

No need to include all the files for the lab. Just include the modified code.

Some important observations while completing/testing this lab were . . .

In this lab we learned . . . .