ECEN-361 Lab-07:Queues and IPC

# NAME: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Introduction and Objectives of the Lab

This lab will introduce the concept of interprocess communication mechanisms, particularly queues and task messages. This will be done in the framework of FreeRTOS middleware as supplied by the STM32 ecosystem.

Queues

Queues are a well-known data structure, serving a FIFO strategy to store/forward data. The student has likely programmed with queues in other classes. We take this data structure and move it into an environment where access to the queue is available multiple producers (tasks that add to the queue), and consumers (tasks that take first-off data from the queue.) Simple everyday analogous systems utilizing this type of structure can be seen in many operations: A tech-support call-center producing many incoming calls at a time, putting them in the queue to be help for service, while many operators pull the calls off to consume and process them.

In FreeRTOS, the queue structure is global in nature, and available to be written-to and read-from most any other process. In general processes will be dedicated to handing one or the other operation – either producing or consuming. Events (interrupts) can be setup to signal on various states of the queue: EMPTY, FULL, FILLED-to-a-LEVEL, etc. can a be the source of interrupts and exceptions in processing.

For this lab we will build a small queue pipeline with three producers, and 1 consumer. Elements that go into the queue pipeline from the superset of all ASCII characters, but each producer has the unique quality that they deliver only subsets of the full character set, so:

Producers

1. Random lowercase letters [a..z] One character issued per 400mS
2. Random symbols, punctuation, etc. [!..=] One character issued per 700mS
3. Typed Uppercase Letters (from keyboard – PuTTY) One character per keystroke – about 200mS

Consumer:

1. Reads the queue once every second and displays:
   1. Number of bytes left on the queue or empty “----” or FuLL”
   2. The state of the queue on the TTY Terminal (PuTTY)

Follows is a simple diagram of a queue being fed by all three sources. It’s not quite full and no priority scheme is implemented between the producing tasks.

A diagram of a computer code

Description automatically generatedColor-coding of the item shown in the queue corresponds with the source of that item.

There is also control on the production:. A dedicated button start/stops each of the following producers.

* The Random Character Producing Task Button\_1🡪 Start/Stop
* The Random lowercase number Production Task Button\_2 🡪 Start/Stop

The consumer is an outputting process who waits for data availability then sends that data to the USART2 out. (USB TTY via PuTTY).

Note that keystrokes typed into the UART keyboard (via the TTY Terminal) are ‘filtered’ before being put onto the queue. This type of pre-processing Is typical and could be likened to a real-world example of a tech-support site sending customers with different types of issues into different waiting queues.

In this code, all keystrokes typed are checked to see that they are strictly alphabetic [‘A’ … ‘z’] and then added to the queue only as uppercase. This allows us to determine the source of the item in the queue, by the type of data: uppercase, lowercase, and special-characters all can come only from their unique source producer.

kj

# Lab Instructions

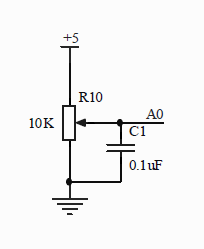
## Overview of System

## Part 1: Understand the ADC, Read/Output the voltage

### Accept the Assignment, Download the repo, Run it

Add the project to STM32CubeIDE, Clean/Compile/Run. Open a TTY-emulator (PuTTY [Windows], screen [Mac]), and review the output on the serial emulator. The pre-built project, cloned from the repo, has three concurrent tasks running.

configured the ADC to take readings from the potentiometer and display them on the 7-segment LED display as a voltage. The ADC result is sampled once a second and output on the serial/USB monitor. You should see an output like this:



As seen in the schematic, the Multifunction board has a potentiometer on it that is wired to swipe between 0V and 5V. This trim pot is blue, on the left side. See the schematic in the Documentation folder.

Using a small screwdriver (or fingernail!), adjust the potentiometer on the Multifunction board and look at the output in the serial monitor. The default DisplayMode is the Voltmeter.

## 4 Pts.

1. What is the minimum sample resolution (change) you can see by adjusting the potentiometer?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Can you predict this change, given the current settings of the ADC?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Change the sampling accuracy of the ADC (Use the **.ioc** file & STM32 GUI), to be the maximum of 12-bit: Re-generate, build, and compile.

A screenshot of a computer

Description automatically generated

Now predict and verify the smallest sample resolution being detected.

1. Can you predict this change (from the math), given the current settings of the ADC?  
     
   Predicted: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Actual Seen: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### PWM

## 2 Pts.

1. Using the Digtal input on the Saleae Analyzer and the DisplayMode and StartStop button – Measure the D2\_LED output at a different duty-cycle points. The duty cycle is shown on the 7-Segment LED or the TTY terminal output when stopped. Does the duty cycle shown match the waveform? List the measured times (high vs low)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### DAC

## 4 Pts.

1. Using a voltmeter (or the Analog input on the Saleae Analyzer) and the DisplayMode and StartStop button – Measure a few points on the D1\_LED output. Do they match with estimation? (12-bit DAC, 5V range). You can also look at the value printed on the TTY Terminal output when the cycle is stopped. List a couple of the measurements.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. In a DAC controlled LED, the LED can’t truly be dimmed to zero because the Vforward of the diode isn’t high enough to turn on the diode and use the current to generate photons. Using the DisplayMode button and the StartStop button, empirically determine the voltage at which the D1\_LED appears to go (almost) out? (Remember this is negative true logic).

For an LED like this, which approach to ‘dimming’ makes more sense? PWM or DAC?  
  
Would this apply to a motor? (Think about this …)  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Which approach would work for approximating a sine-wave output? PWM or DAC? Why?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Extra Credit (5 pts.max for any of these completed)

1. This code uses the ADC in a blocking mode. Change the ADC to sample via an interrupt instead. Show some of the code changes that need to be made:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. What is the maximum sample rate, given the current clocking scheme, for: the least-precise (6-bit) and most precise (12-bit) conversions?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. What is the maximum sample rate, given the current clocking scheme, for: the least-precise (6-bit) and most precise (12-bit) conversions?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. The Seven-Segment display using a type of PWM by refreshing each of the four segments in turn. As given you, it goes fast enough that it looks to be 100% brightness. Change the parameters of the refresh Timer (Timer17) with a parameter so that the apparent intensity of this display can be dimmed.