ECE 5780/6780 Lab 3 – Spring 2024

Due Date: Friday, February 9 (20 points)

Objectives

The purpose of this lab is to learn how to translate PC keyboard presses sent over USB/UART into discrete audio tones using FreeRTOS, Keil µVision, and the STM32L476 Nucleo-64 Board.

Overview

In this lab you will create, download, execute, and debug a multi-task C program with two interrupt handlers in the STM32L476: A timer interrupt that controls the sine wave audio output on a speaker and a UART receive interrupt that handles characters sent from the PC keyboard.

Preparation

You will need your STM32L476 Nucleo-64 Board and ECE 5780/6780 lab kit.

Requirements

1. Make a copy of your Keil µVision project from Lab 2 to use as a starting point.

2. The user push button still controls the LED and the on/off functionality of the speaker using the software-generated, 64-entry-lookup-table sine wave. When the speaker is first toggled on, the output should be 220 Hz.

3. The STM32L476 will be interfaced via UART with a PC running a serial terminal such as PuTTY. Select Baud rate, number of data bits, parity scheme, and number of stop bits appropriately. Implement a UART receive interrupt to handle incoming characters.

4. Typing characters ‘a’ through ‘g’ on the PC serial terminal will result in a change in frequency on the speaker, according to Table 1. This implements a musical octave.

5. Your system must utilize an LED task, a button task, a timer interrupt (to generate the audio sine wave via the DAC), and a UART RX interrupt to handle incoming serial characters. Additionally, your system must implement queues to (1) pass information from the UART RX interrupt handler to the timer interrupt handler, (2) pass information between the button and LED tasks, and (3) pass information from the button or LED task to the timer interrupt.

Character	Frequency	Musical Note
‘a’	220.00 Hz	A (low A)
‘b’	246.94 Hz	B
‘c’	261.63 Hz	C (middle C)
‘d’	293.66 Hz	D
‘e’	329.63 Hz	E
‘f’	349.23 Hz	F
‘g’	392.00 Hz	G
‘h’	440.00 Hz	A (high A)

Table 1: Musical frequencies

Pass-off

Demonstrate the working system to the instructor, either in-person, via Zoom, or via a recorded video (emailed to the instructor) by playing a song such as “Twinkle Twinkle Little Star”. Also show that you are using queues for interrupt-to-task, task-to-task, and task-to-interrupt communication (no other global variables).

Hints

1. The following FreeRTOS API functions may prove useful:
    a. xQueueCreate()
    b. uxQueueMessagesWaiting()
    c. uxQueueMessagesWaitingFromISR()
    d. xQueueReceive()
    e. xQueueReceiveFromISR()
    f. xQueueSendToBack()
    g. xQueueSendToBackFromISR()

2. The CMSIS NVIC_SetPriority() function may be useful.