Homework Assignment #3

Due 3/25/2015 @ 5PM

Instructions

Your team will turn in a single .zip file named hw3_team_xx.zip (where xx is your team number) that contains the following information to the drop box located on the ECE353 homepage:

Key Points

- All code should have comments. A 5 point penalty will be levied against each problem that is not commented sufficiently.
- Any code that does not assemble/compile will receive no points.
- Fill out your name and team information in main.c
- All code must be generated by your group either individually or as part of an in-class exercise. Any evidence of sharing code between groups will be treated as a case of academic misconduct and result in a failing grade for the course.

The Annoying Sound Generation System

1. Overview

   This homework will have you develop an application that generates mildly to severely annoying sounds using the Tiva Launchpad and the ECE 353 Carrier card. The ECE353 carrier card has a small speaker that is driven by a simple resistor ladder DAC (Digital to Analog Converter). The DAC is controlled by 6 GPIO pins on the Tiva Launchpad. We will output square waves on the GPIO pins connected to the DAC to produce sounds. The frequency and duty cycle of the square waves will be set using the PS2 joystick. You will also be able to control the volume of the speaker using the rotary potentiometer (P300). The UP directional button will be used to mute the speaker and re-enable sound.

   A demo of the working project has been provided in the demo directory. Do NOT recompile the project. Simply hit the load button in Keil and then hit the reset button on the Launchpad.

2. Peripheral Requirements
    a. Digital GPIO Pins

       Configure the UP directional button as <u>digital input</u>. This pin will need to its internal pull-up resistor enabled.

       Configure the GPIO pins connected to DAC_5 through DAC_0 as <u>digital outputs</u>. Do NOT enable pull-up/down resistors. Examine the ECE353 carrier card schematics to determine which pins these signals are connected to.

    b. Analog GPIO Pins

       Configure the GPIO pins connected to the PS2 joystick and the rotary potentiometer as <u>analog input pins</u>.

    c. ADC0

       Configure ADC0 to use <u>Sample Sequencer **#2**.</u> The sample sequencer will be initiated by the software (Processor). Configure the sequence so that the flag bit in the RIS register is set AFTER ALL three analog channels are converted.

    d. Timer0

       Timer0 should be configured as two 16-bit, periodic, count down timers (Timer0A and Timer0B). The timers should generate an interrupt when the timers expire.

       Timer0A will be used to generate the portion of the square wave that is logic 1.

       Timer0B will be used to generate the portion of the square wave that is logic 0.

    e. SysTick Timer

       The Systick Timer will be configured to generate an interrupt every 1millisecond.

    f. Interrupts

       Timer0A, Timer0B, and the SysTick timer should all generate interrupts. You will need to write interrupt service handlers for each of these interrupt sources.

3. Functional Description

a. PS2 Joystick

The PS2 joystick is used to determine the duty cycle and frequency of the square wave. Moving the joystick in the Y direction changes the frequency of the square wave.

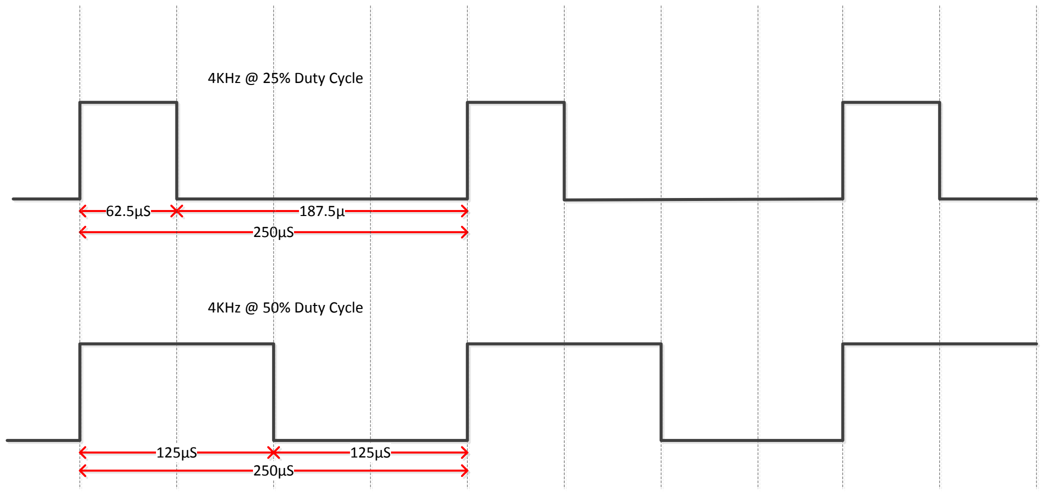
The frequency of the square wave should be adjusted according to the table below. The Y Direction Reading is listed in terms of the percentage of the maximum value of the 12-bit ADC.

Y Direction Value	Frequency
Y Reading < 20%	1.0 Khz
20% ≤ Y Reading < 30%	1.5 Khz
30% ≤ Y Reading < 40%	2.0 Khz
40% ≤ Y Reading < 50%	2.5 Khz
50% ≤ Y Reading < 60%	3.0 Khz
60% ≤ Y Reading < 70%	3.5 Khz
70% ≤ Y Reading < 80%	4.0 Khz
Y Reading ≥ 80%	4.5 Khz

The X direction sets the duty cycle of the square wave.

X Direction Value	Duty Cycle
X Reading < 20%	25%
20% ≤ X Reading < 80%	33%
X Reading ≥ 80%	50%

The following diagram shows two example waveforms. The waveform should be generated on the active DAC GPIO pin.



b. Rotary Potentiometer

The rotary potentiometer is used to control the volume of the speaker. The volume of the speaker will be set by allowing only a single DAC GPIO pin to be active at a given time. The following table can be used to determine which DAC GPIO pin is active based on the voltage output by the rotary potentiometer.

Potentiometer Value	Active DAC GPIO Pin
POT Reading < 15%	DAC_0
15% ≤ POT Reading < 30%	DAC_1
30% ≤ POT Reading < 45%	DAC_2
45% ≤ POT Reading <60%	DAC_3
60% ≤ POT Reading < 75%	DAC_4
POT Reading ≥ 75%	DAC_5

c. Timers

Timer0A and Timer0B are used to count the number of clock cycles required to generate the square wave. Timer0A is used to count the number of clock cycles when the active DAC GPIO pin is high. Timer0B is used to count the number of clock cycles when the active DAC GPIO pin low.

Each timer should generate an interrupt indicating that the active DAC GPIO pin's value should be toggled. Only one of these timers should ever be enabled at a given time. For example, when Timer0A generates an interrupt, Timer0A should be disabled and Timer0B should be enabled. When Timer0B generates and interrupt, Timer0B should be disabled and Timer0A should be enabled. The value of TAILR and TBILR are set based on the selected frequency and duty cycle input using the PS2 joystick.

The Systick Timer will be configured to generate an interrupt every 1millisecond. This interrupt indicates that an ADC sequence should be initiated and the UP button should be sampled.

d. UP Button

The UP button toggles the sound on and off. When the board powers up, the sound should be enabled. A button should be de-bounced for 16mS on the falling edge of the button.

4. Source Code Requirements

   The HW3 distribution contains an include directory that defines the required functions for the ADC, Timer0, and GPIO pins.   Examine adc.h, timer0.h, and gpioPort.h to see which functions you are required to implement.

   For the three header files above, you will need to implement the functions at the end of each file in the associated .c files in the drivers directory.

   There are other header files in the include directory, but you do not need to implement any of the functions in those files.