Lecture/Lab 12-project

In the final assignment of the class, we will be writing a program to play some holiday music.

Before beginning the program itself, we will being writing some of the functions that will ultimately be used in the program.

Mainly, we need to control the pulse width modulator to produce tones, as well as have a means to delay by a specific amount of time.

Functions you should write:

1) A function that enables the PWM0 and sets its duty\_cycle to 0.

This function really has no parameters, and has no return value so the signature should look like:

void init\_pwm();

You can use the IOWR\_ALTERA\_AVALON\_PIO\_DATA(address,value); function to set values to the PWM. Remember to have the following includes:

**#include** <altera\_avalon\_pio\_regs.h>

**#include** <system.h>

The base address for PWM0 is in system.h as PWM0\_BASE

but remember that the PWM0 has three registers, at addresses BASE, BASE +4, and BASE +8. You can use the following form:

IOWR\_ALTERA\_AVALON\_PIO\_DATA(PWM0\_BASE, x);

// writes the value in variable x to PWM0\_base (the Clock\_Divider register)

IOWR\_ALTERA\_AVALON\_PIO\_DATA(PWM0\_BASE + 4, y);

// writes the value in variable y to PWM0\_base +4 (the Duty\_cycle register)

IOWR\_ALTERA\_AVALON\_PIO\_DATA(PWM0\_BASE + 8, z);

// writes the value in variable z to PWM0\_base + 8(the Enable register)

The function just needs to set enable to 1 and duty\_cycle to 0.

2) A function that takes an integer parameter of a frequency in Hz, and sets the PWM modulator to output a 50% duty cycle square wave at that frequency.

This function needs to take a frequency in hz, so the signature should look like:

void set\_pwm\_freq(int hz);

Review lecture/lab 3 for the equations for PWM setup. Essentially, set the clock divider to 50,000,000 divided by frequency in hz, and then set the duty\_cycle to half that value to create a square wave at a given frequency.

Consider first setting duty\_cycle to 0, then setting the clock\_divider to the desired value, then set the duty\_cycle to half that. Doing it in this order will avoid spurious signals.

3) A function that resets the pwm output to off.

Like the init, this takes no parameters so it should have a void return type and no params.

It should simply set the duty\_cycle register to 0, there is no need to set the clock\_divider to 0 also.

4) A function that takes an integer parameter of a time in milliseconds, and delays for that amount of time.

This function should take a time in milliseconds, so the signature should look something like:

void delay\_ms(int time);

Here is a structure for a delay that always delays 1 millisecond:

**void** **delay\_ms**()

{

**int** i;

**int** dummy;

**for**(i = 0; i < 447; i++)

dummy++;

}

Modify this function so it takes an integer parameter, and then make the function loop the correct number of times (calculate the correct number of loops if 447 gives 1 millisecond, and the parameter should wait X milliseconds).