EEET2250 – Week 5 Lab Task (due Week 6)
Using Pulse-Width Modulation (PWM) on the OUSB board

Total possible marks: 15 marks (3% of final mark)

Aim:
The Lab Tasks are aimed at helping you to build up code for the Lab Test 2 - the Week 5 Lab Task is directly related to your Lab Test 2 to be held during your Week 7 lab class (which will be just before the mid-semester break).

This Lab Task requires you to write a program to use and control the Pulse-Width Modulation (PWM) on the Open USB-IO (OUSB) microcontroller board. Your program will also be required to read from the analogue inputs on the board e.g., trimpot and Light Depended Resistor (LDR).

It is recommended that a generic OUSB access function is used for the Lab Tasks (and Lab Tests), where your code from the Week 2 Lab Tasks can be used to do this or you can refer to page 22 of the OUSB board manual for an example - just remember to use the safe Visual Studio version of the appropriate functions and correctly ***null terminate*** the command string.

The tasks specified below are aimed at helping you to break down the program into smaller functional 'modules': you should attempt to code in modules, checking the functionality of each module before continuing to code (i.e., do not write the whole program at once and expect it all to work!).

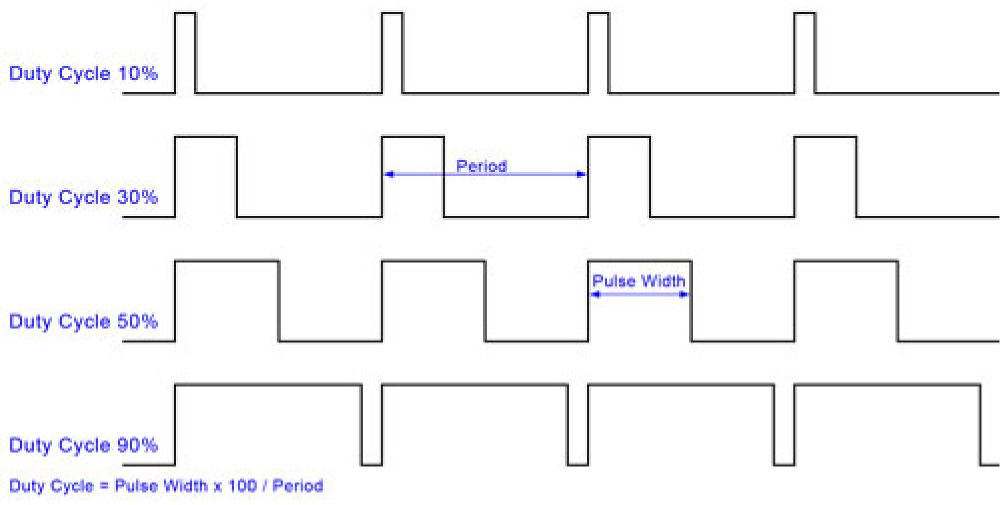
Once you have finished all of the tasks notify your tutor who will assess your work. Having task (4) marked also means that you have tasks (2) and (3) completed. The marking for these tasks is binary – it either works or does not.

Remember to use the lecture notes and OUSB board manual as references as well as the prescribed textbook. You also might find the OUSB Board Resources page on the EEET2250 Canvas shell a useful starting point too.

By completing the Week 5 and 6 Lab Tasks, you have effectively written most of your Lab Test 2 code, except for the required error checking, and error codes that are displayed on console output (which is used by autotester to assess your code). The Lab Test 2 proforma cpp file is available on EEET2250 Canvas Shell, and you may wish to code the week 5 and week 6 Lab tasks inside the provided Lab Test 2 proforma to gain familiarity with the Lab Test 2 requirements.

Background:
Pulse-Width Modulation (PWM) refers to a type of signal modulation where the signal is effectively a square wave that varies in how much of the square wave is 'on' (signal high) and 'off' (signal low) i.e., the width of the pulses changes (modulated). As the signal basically acts like a switch (usually operating at a high speed), PWM is an efficient way to deliver a low voltage from a higher voltage with minimal losses and this is why PWM is often used to drive electrical loads e.g., DC motors.

The ratio of 'on' to 'off' for the PWM square wave pulses is called the duty cycle (between 0 and 100%):



(Image source: http://www.protostack.com/blog/2011/06/atmega168a-pulse-width-modulation-pwm/)

The OUSB board uses the PWM command to generate a square wave on pin PB3 of the microprocessor, which is connected to the LED3 (on PORTB) and to the inverting buffer which outputs on connector J5 pin 27 (connected to the simulated DC motor load components soldered onto the prototype section of the OUSB board - we will use this in the Week 6 Lab Tasks and Lab Test 2).

The command "`ousb pwm-freq 1 ...`" sets the PWM frequency (we will use 46Hz in this lab so the human eye cannot see LED3 flicker on PORTB).

The command "`ousb pwm 1 ...`" sets the PWM duty cycle. For example:

```
ousb pwm-freq 1 46   // Sets PWM 1 to 46 Hz

ousb pwm 1 30     // Sets PWM 1 to 30% duty cycle, LED3 on PORTB glows at 30% brightness

ousb pwm 1 70     // Sets PWM 1 to 70% duty cycle, LED3 on PORTB glows at 70% brightness

ousb pwm 1 100    // Sets PWM 1 to 100% duty cycle, LED3 on PORTB glows at full brightness
```

Note that the `pwm-freq` command must be used first to set frequency but after that any number of PWM commands can be used to change the duty cycle. The '`1`' in '`ousb pwm 1 ...`' refers to the first PWM modulator on the OUSB board (there are three PWM modulators on the OUSB board), which is the PWM we use as this is connected to LED3 on PORTB (so you can see the effect of PWM) and to the simulated DC motor load components soldered onto the prototype area of the board. More details can be found in chapter 6 of the OUSB manual and in the Lab Test 2 proforma.

Tasks

1. When you attempt each of the three tasks below, remember the six (6) steps of engineering design: draw flowcharts and write pseudocode to represent the logical flow of your program for each of the tasks below.

[3 marks]

2. Write code for your program to use the PWM on the OUSB board (and test that it works!). You need to accept an integer parameter from the command line into your program (i.e., using `argc` and `argv`) as the duty cycle. Perform an error check to ensure that the duty cycle entered is between 0 and 100 inclusive.

   (Hint: remember to set the PWM frequency to 46Hz first before you set/change the PWM duty cycle!)

   [4 marks]

3. Write code for your program to read from the analogue ports on the OUSB board e.g., trimpot, LDR etc. (and test that it works!). You will be able to find information regarding the OUSB board analogue ports by referencing the OUSB reference manual and the schematics provided on the EEET2250 Canvas shell. Your code should accept an integer parameter from the command line into your program (i.e., using `argc` and `argv`) as the analogue port number to be read. Print only the read number returned from the board onto the screen. Perform an error check to ensure that the port number entered as part of the command line arguments is between 0 and 7 inclusive.

   [4 marks]

4. Using the code from tasks 2 and 3, modify your program to accept another parameter from the command line input (i.e., using `argc` and `argv`) before the integer value so that your code can decide to run the algorithm for task 2 or task 3 at runtime:
   - Accept a 'P' parameter followed by the duty cycle integer to run the code from task (2) and use the PWM
     e.g., `myProgram.exe P 50` // Sets a PWM at 50% duty cycle

   - Accept an 'A' parameter followed by the analogue port number to run the code from task (3) and read from analogue I/O
     e.g., `myProgram.exe A 5` // Read from analogue I/O port number 5

   Perform an error check to ensure that if letters other than 'P' or 'A' are entered onto the command line, an error message is printed onto the screen. You should also test the appropriate ranges for the integer value based on whether the P or A command is selected.

   [4 marks]

Once you have finished these tasks you are free to move on to working on the Week 6 Lab Task and/or Lab Test 2 practice solution.

You should not leave the laboratory after completing the tasks on this sheet. There is certainly enough work to continue on with for later tasks.