829H1 Real-Time Embedded Systems Exercise 3

Candidate No: 105936

12th January 2019

Contents

1. Introduction

This report outlines the work completed during the laboratory sessions, what equipment was used and what was learnt. Code listings for some of the created programs can be found in section 3.

2. Experiments

2.1. PWM

For this experiment I created a PwmOut object and set the period of it to 0.01 seconds and the duty cycle to 0.5 to make identifying the period easy.

2.1.1. Changing the duty cycle

I then went on to change the duty cycle to 80% and verified my results on the oscilloscope. This shows that the wave was on for 0.008 seconds and off for 0.002 seconds at a time. The code for this can be found in section A.1

2.1.2. Changing the period function

I also changed the period function and used the period_ms() and pulsewidth_ms() functions and I was able to recreate the same wave with each. For the period_ms() function I had to convert the time to milliseconds. For the pulsewidth_ms() function I had to halve the millisecond time to get the program to output the same wave. The code for the pulsewidth_ms() example can be found in section A.2.

2.2. Generating PWM in software

The given example manually implements the PWM function which generates the wave on the oscilloscope.

2.2.1. Modifying the wait times to increase the duty cycle

Firstly I changed the wait times so that the low duty cycle was now a medium high duty cycle of 60% and the high duty cycle is now a very high duty cycle of 95%. This was done by changing the wait times accordingly. I also change the time each of the duty cycles were running to 3 seconds to make it a bit quicker to test. The code for this experiment can be found in section B.1

2.2.2. Modifying the program to use PwmOut

This was quite simple as all I had to change was the DigitalOut to a PwmOut. Set the period in us to 1000 and then remove the for loops and set the desired duty cycle followed by a wait. The code for this can be found in section B.2

3. Conclusion

In conclusion, this was a useful quick introduction into how to develop using the PWM class to manage power output to a pin and therefore a external device.

Appendices

A. Lab Exercise 1

A.1. Part 1

```
#include "mbed.h"

PwmOut PWM1(PTC10); //create a PWM output called PWM1 on pin PTC10

int main()
{
    PWM1.period(0.010); // set PWM period to 10 ms
    PWM1=0.8; // set duty cycle to 80%
}
```

A.2. Part 2

```
#include "mbed.h"

PwmOut PWM1(PTC10); //create a PWM output called PWM1 on pin PTC10

int main()
{
    PWM1.pulsewidth_ms(5); // set PWM period to 10 ms
    PWM1=0.5; // set duty cycle to 50%
}
```

B. Lab Exercise 2

B.1. Part 1

```
/*2 PWM values generated in turn, with full on and off included for
comparison. for not using PWM function directly */
#include "mbed.h"
DigitalOut motor(PTD3);
int i;
int main() {
   while(1) {
      motor = 0; //motor switched off for 3 secs
      wait (3);
      for (i=0;i<3000;i=i+1) { //3000 PWM cycles, medium-high duty cycle
        motor = 1;
        wait_us(600); //output high for 600us
        motor = 0;
        wait_us(400); //output low for 400us
      }
      for (i=0;i<3000;i=i+1) { //3000 PWM cycles, high duty cycle
        motor = 1;
        wait_us(950); //output high for 950us
        motor = 0;
        wait_us(50); //output low for 50us
      }
     motor = 1; //motor switched fully on for 3 secs
     wait (3);
```

```
  }
}
```

B.2. Part 2

```
/*2 PWM values generated in turn, with full on and off included for
    comparison. */
#include "mbed.h"
PwmOut motor(PTD3);

int main()
{
    while(1) {
        motor.period_us(1000);
        motor = 0; //motor switched off for 3 secs
        wait (3);
        motor = 0.6; // 60% duty cycle
        wait(3);
        motor = 0.95; // 95% duty cycle
        wait(3);
        motor = 1; //motor switched fully on for 3 secs
        wait (3);
    }
}
```