EENG18020 Ultrasonic Lab
Week 4 - PWM

1 Introduction

In this section you will test the PWM generator (Pin DIO21, Figure1) on the CC1350 micro-controller board. You will be using the **pwm** and **pwm_adc** project.

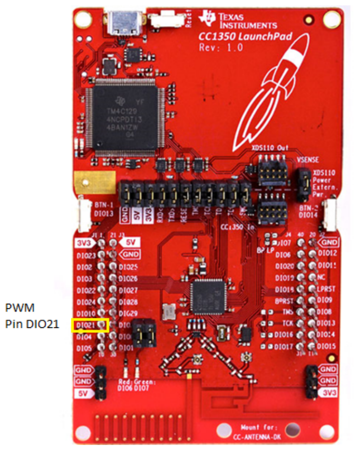

Figure 1: PWM Pin DIO21

2 Generate Pulsed PWM Signal with Micro-Controller

In order to detect the time taken for an ultrasonic pulse to reach a destination and return, we must know the time at which the transmission and reception started, and then calculate the difference between these two. Therefore we drive the transducer with a micro-controller.

While it is possible for the micro-controller to manually toggle the output pin between on and off, it is inefficient as it requires a lot of CPU time for a relatively simple task. The superior method of driving the transducers is the built-in pulse-width modulation (PWM) generator.

PWM signals are square waves with variable pulse-width. The CC1350 board has 4 General-Purpose Timer Modules, which are capable of generating 4 PWM signals without any intervention from the micro-controller's CPU. This results in much more reliable and stable operation at high frequencies (as there is no time spent waiting for an interrupt to be serviced). The purpose of the PWM generator is to generate signals with a variable duty cycle (hence variable power delivery for generating analogue voltages). However, by setting the duty cycle to 50%, the PWM period to the same as the resonant period of the ultrasonics, and then turning the PWM on and off, we can create a pulsed square-wave.

There are two ways to turn the PWM signal on and off. One option is to enable and disable the PWM counter, however, this takes time and hence introduces more uncertainty into the time measurement of the start signal. Instead, a more effective option is to toggle the duty cycle between 0% and 50%, as a 0% duty cycle signal is always low, and hence is indistinguishable from turning the PWM off, except for the fact that it takes effect immediately.

With CCS we can use CC1350 to generate PWM signal by setting the following parameters as a C-type structure. An example is given in **pwm.c**.

Figure 2 shows example code to generate a pulsed PWM.

Table 1: C-type structure to configure a PWM signal

Type	Name	Options
PWM_Period_Units	periodUnits	PWM_PERIOD_US: Microseconds
		PWM_PERIOD_HZ: Hertz
		PWM_PERIOD_COUNTS: Timer Count
uint32_t	periodValue	Integer value
PWM_Duty_Units	dutyUnits	PWM_DUTY_US: Microseconds
		PWM_DUTY_FRACTION: Percentage of Period
		PWM_DUTY_COUNTS: Timer Count
uint32_t	dutyValue	Integer value

```
void pwmLoop()
{
    while(1)
    {
        //Sleep for 500ms
        Task_sleep(SLEEP_1MS * 500);
        //Set duty to 0
        UART_printf(uart, "PWM = 0\r\n");
        PWM_setDuty(pwm1, 0);
        //Sleep for 500ms
        Task_sleep(SLEEP_1MS * 500);
        //Set duty to 50%
        UART_printf(uart, "PWM = 50%%\r\n");
        PWM_setDuty(pwm1, PWM_INIT_DUTY);
    }
}
```

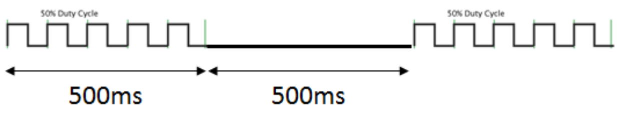

Figure 2: Pulsed PWM signal

3 Pulsed Ultrasonics

Our ultrasonic signal has a velocity of $300ms^{-1}$ and a frequency of 40kHz. As a result, the wavelength of the signal must be 7.5mm. This is problematic as a wave that has travelled a multiple of its wavelength is indistinguishable from a wave that hasn't. Therefore, once the signal travels 7.5mm (3.75mm back and forth), it becomes ambiguous with itself. To solve this, we pulse the ultrasonics and measure the time at which the pulse returns. While the precision of this process is dependent on the sampling rate (and hence not particularly high in our system), it gives excellent range. When combined with phase detection, the two methods complement each other to give a large range and high precision.

Task 1: Read and run the PWM project, answer the following questions

1. When generating the PWM signal, why might we want to use the **PWM_setDuty** (pwm_ex.c) function?

2. When generating the PWM signal, why might we want to use the **Task_sleep** (pwm_ex.c) function?

3. You have been given the code to generate a pulsed PWM signal. Draw a block diagram showing how you would connect your CC1350 micro-controller to the analogue section of the circuit. Make sure to note down pin number(s) and draw the main components of the analogue section.

4. Connect your CC1350 to the input pin of your transmitter amplifier, show a lab demonstrator the output of the CC1350 before and after amplification on the oscilloscope. Show your receiver amplifier's output.

5. Modify the code to produce a PWM signal at the fundamental frequency of your transducer. Check the signal again. Make sure your output from receiver amplifier does not exceed 3.3V.

6. Load the **pwm_adc** project, connect receiver amplifier's output to the pin DIO24 of the board, measure the received signal from the receiver amplifier with the ADC and graph a small section of it inside CCS.

7. You have now transmitted a signal with the PWM peripheral and received the signal through the ADC peripheral. Suggest some pseudo code to measure the distance to an object.