Panel 1

Prior to Le17

Pulse Width Modulation

ME430 Mechatronics

1

Panel 2

Pulse Width Modulation (PWM)

What is Pulse Width Modulation?

Why?

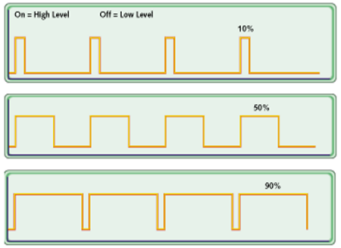


Figure 1. PWM signals of varying duty cycles

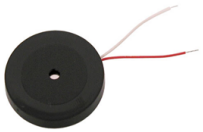
Two parts of the PWM:

- Frequency - 1/(time between rising edges)
- Duty Cycle - % of time PWM is high

2

Panel 3

Piezoelectric speakers



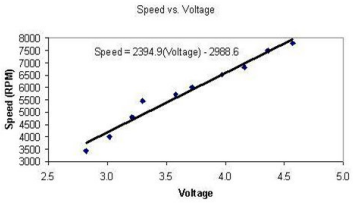
Cheap

Easy to use

3

Panel 4

Speed Control for a DC motor



Pulse Width Modulation

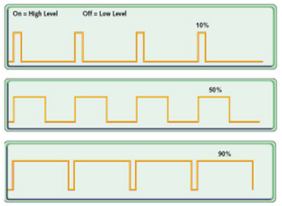


Figure 1. PWM signals of varying duty cycles

4

Panel 5

TABLE 2-9: PWM FUNCTIONS

Function	Description
OpenPWMx	Configure PWM channel **x**.
SetDCPWMx	Write a new duty cycle value to PWM channel **x**.

Note 1: The enhanced PWM functions are only available on those devices with an ECCPxCON register.

5

Panel 6

Duty cycle range

SetDCPWM1
SetDCPWM2
SetDCPWM3
SetDCPWM4
SetDCPWM5
SetDCEPWM1

Function: Write a new duty cycle value to the specified PWM channel duty-cycle registers.

Include: `pwm.h`

Prototype:
```
void SetDCPWM1( unsigned int dutycycle );
void SetDCPWM2( unsigned int dutycycle );
void SetDCPWM3( unsigned int dutycycle );
void SetDCPWM4( unsigned int dutycycle );
void SetDCPWM5( unsigned int dutycycle );
void SetDCEPWM1( unsigned int dutycycle );
```

Arguments: ***dutycycle***
The value of *dutycycle* can be any 10-bit number. Only the lower 10-bits of *dutycycle* are written into the duty cycle registers. The duty cycle, or more specifically the high time of the PWM waveform, can be calculated from the following formula:
PWM x Duty cycle = (DCx<9:0>) x TOSC
where DCx<9:0> is the 10-bit value specified in the call to this function.

Remarks: This function writes the new value for *dutycycle* to the specified PWM channel duty cycle registers.
The maximum resolution of the PWM waveform can be calculated from the period using the following formula:
Resolution (bits) = log(FOSC/Fpwm) / log(2)

File Name:
```
pw1setdc.c
pw2setdc.c
pw3setdc.c
pw4setdc.c
pw5setdc.c
ew1setdc.c
```

Code Example:
```
SetDCPWM1(0);
```

6

Panel 7

Simple Open function even

OpenPWM1
OpenPWM2
OpenPWM3
OpenPWM4
OpenPWM5
OpenEPWM1

Function: Configure PWM channel.

Include: `pwm.h`

Prototype:
```
void OpenPWM1( char period );
void OpenPWM2( char period );
```

Arguments: ***period***
Can be any value from 0x00 to 0xff. This value determines the PWM frequency by using the following formula:
PWM period =[(***period***) + 1] x 4 x TOSC x TMR2 prescaler

Remarks: This function configures the specified PWM channel for period and for time base. PWM uses only Timer2.
In addition to opening the PWM, Timer2 must also be opened with an **OpenTimer2(...)** statement before the PWM will operate.

File Name:
```
pw1open.c
pw2open.c
pw3open.c
pw4open.c
pw5open.c
ew1open.c
```

Code Example:
```
OpenPWM1(0xff);
```

7

Panel 8

Example code for PWM

```
OpenTimer2(TIMER_INT_OFF & T2_PS_1_4);
    //PWM period =[(timer ticks) + 1] x 4 x TOSC x TMR2 prescaler
    // Fosc = 1 MHz (the default internal oscillator)
    // TMR2 prescaler = 4

OpenPWM1(141);
        // Resulting PWM Period = (141+1_* 4 * 1/1E6 * 4) = 0.00227 seconds -> 440 Hz

SetDCPWM1(300);
            // Set the duty cycle


// Later in your code if you want it off
SetDCPWM1(0);

// Later in your code if you want it always high
SetDCPWM1(1023);

// Later in your code if you want it at 50%
SetDCPWM1(512);
```

8

Panel 9

PWM shortcut formula:

Desired PWM period =[(timer ticks) + 1] x 4 x Tosc x TMR2 prescaler

Limitations:

timer ticks

Tosc

TMR2 prescaler:

9

Panel 10

PWM also depends on the OSCCON setting

bit 6-4 **IRCF2:IRCF0:** Internal Oscillator Frequency Select bits

111 = 8 MHz (INTOSC drives clock directly)
110 = 4 MHz
101 = 2 MHz
100 = 1 MHz(3)
011 = 500 kHz
010 = 250 kHz
001 = 125 kHz
000 = 31 kHz (from either INTOSC/256 or INTRC directly)(2)

10

Panel 11

Open Timer 2

OpenTimer2

Function:	Configure and enable timer2.
Include:	`timers.h`
Prototype:	`void OpenTimer2( unsigned char config );`
Arguments:	***config*** A bitmask that is created by performing a bitwise AND operation ('&') with a value from each of the categories listed below. These values are defined in the file `timers.h`. **Enable Timer2 Interrupt:** `TIMER_INT_ON` Interrupt enabled `TIMER_INT_OFF` Interrupt disabled **Prescale Value:** `T2_PS_1_1` 1:1 prescale `T2_PS_1_4` 1:4 prescale `T2_PS_1_16` 1:16 prescale

11

Panel 12

Sample calculation already done for you

```
/** Define Constants Here ****************************************/
//  Formula from the library function:
//  PWM period =[period + 1] x 4 x TOSC x TMR2 prescaler
//
// We have setup:
// Fosc = 1 MHz
// Tosc = 1/1 000 000
// TMR2 = 4
//
//  Want a frequency for Middle A (440Hz)
//  1/440 = (period + 1) * 4 * 1/1000000 * 4
//  period = 141.0454545
//   Therefore A_mid = 141
//
//  It's 1 equation and 1 unknown.  It's not that hard.
//   I used excel and setup the period for basic notes:
```

12

Panel 13

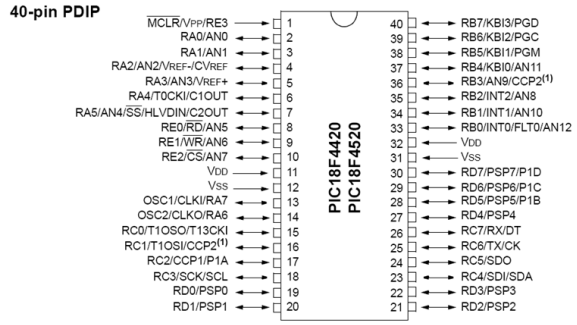
Example problem: Say we want 3 Hz. What is the Timer 2 prescaler, PWM period, and clock frequency?

13

162

Panel 14

Where does the PWM output?



14

Panel 15

Sample code:

Using the ADC program play me a nice middle F (349 Hz) when RB0 is pressed

Note to keep the LCD you must use the 4 MHz clock frequency (so use EC)

It works out nicely. What Timer 2 prescaler and Period did you use?

Write the C code needed to perform these tasks

15