Laboratory Worksheet #07
PWM - Frequency and Pulsewidth Exercise

On LMS, in the Lab 3 folder, the *worksheet_07.c* code is provided to demonstrate the operation of Pulse Width Modulation (PWM). The Pulse signals are characterized by two attributes, the period (T) of one cycle which is controlled by PCA_Start in the program and the pulse width (PW) which is controlled by PW in the program. A shorter period corresponds to a higher frequency. A high duty cycle, $DC = \frac{PulseWidth}{Period} \times 100\%$, corresponds to a relatively large pulse width.

Exercise 1: PCA When answering the following questions, refer to the *worksheet_07.c* code.

1) What is the size of the PCA counter (in bits)?

16

2) What triggers a count in the PCA?

an overflow, after 65535 cycles

3) What is the interrupt priority of the PCA?

9

4) If $PCA\_Start = 47000$, how many counts will occur before the counter overflows? What is the period for this setting (time it takes to count from 47000 until it overflows)?

18535, 10 ms

5) Using the above start value, if PW = 3000, what is the pulse width in seconds? What is the Duty Cycle?

0.0016 s, 16.19%

Now, for a different example, determine PCA_Start and PW for a pulse train with a 3 ms period and a 35% Duty Cycle using a 16 bit counter triggered by SYSCLK/4.

PCA_Start = 48946

PW = 5806

Exercise 2: Hardware

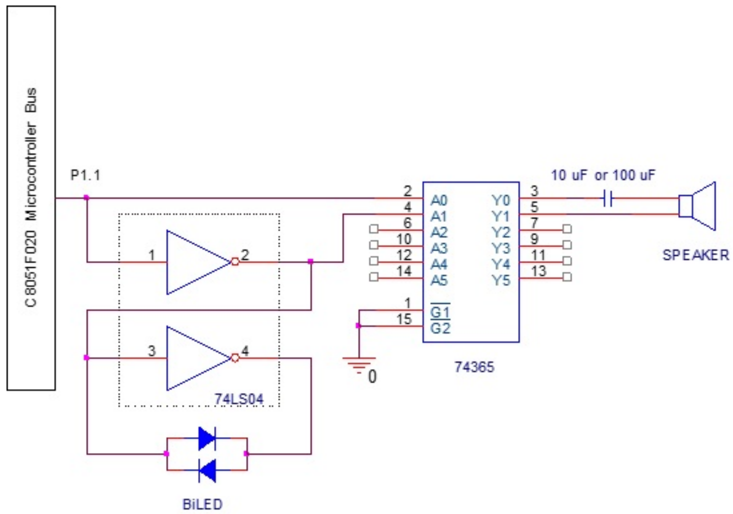

Figure 1: Potentiometer Circuit

1) Build the circuit as shown above. Note: you will need to obtain a speaker from the TAs. Speakers convert periodic electrical signals into corresponding tones. The buzzer you already have in your kits will NOT work since it only needs a voltage to provide a specific tone based on its internal circuit.

2) Download and run the sample program, Worksheet_07.c, from the LMS website.

a) Part A, changing duty cycle

a. Set *PCA_start* to 1000.

b. Change *PW*, the pulsewidth, and observe the effect on the LED..

At one extreme limit of the pulsewidth, the LED will be mostly green in color and at the other extreme limit, it will be mostly red in color. Explain this behavior.

As pulsewidth increases, the LED is more red.

As pulsewidth decreases, the LED is more green.

b) Part B, changing duty cycle

a. Set the pulsewidth, *PW*, to 4000.

b. Change the PCA start value, *PCA_start*h, and observe the effect on the speaker output.

At one extreme limit of PCA_start, the frequency will be low and at the other extreme limit, it will be high. Explain this behavior.

When the pulses are far apart (PCA_start is low), the frequency is lower.

When the pulses are close (PCA_start is high), the frequency is higher.

3) When you use the logic probe to test your PWM output, how does the indicator light behave?

blinking

When complete, include Worksheet 7 with your Laboratory 3.1 Pre-lab submission.

Laboratory Worksheet #08
Crossbar Configuration Exercise

This worksheet will help you configure the crossbar for Lab 3, part 1. Refer to the notes from the professor's lecture on the crossbar. Review the example the professor went over in class on the crossbar. Also refer to the Priority Crossbar Decode Table in the handout.

Exercise 1: Reserved pins and Crossbar initialization

This problem is an example only, do not confuse it with the Crossbar configuration for Laboratory 3 (and later laboratories).

1) Assume the following are enabled: UART0, I2C (SMBus0), and the first four capture/compare modules associated with the PCA. Which port pins will be assigned to the following:

TX0	P0.0 ;
RX0	P0.1 ;
SDA	P0.2 ;
SCL	P0.3 ;
CEX0	P0.4 ;
CEX1	P0.5 ;
CEX2	P0.6 ;
CEX3	P0.7 ;

2) Determine the bit assignments for XBR0. Indicate assigned bits with a 0 or a 1, no bits will be unassigned (no X's).

XBR0 data sheet

bit	7	6	5	4	3	2	1	0
	0	0	1	0	0	1	0	1

3) Determine the command to initialize XBR0 based on the above bit assignments.

XBR0 = 0x25 ;

Exercise 2: Laboratory preparation

1) What is the XBR0 setting indicated in Laboratory 3? 0x27

2) For each Laboratory 3.1 version, which Capture Compare Module is assiged.

Speed Controller	2 ;
Steering Servo	0 ;
LED	3 ;

When complete, include Worksheet 8 with your Laboratory 3.1 Pre-lab submission.

EVB Pin	
1	2
3	4
5	6
7	8
9	10
11	12
13	14
15	16
17	18
19	20
21	22
23	24
25	26
27	28
29	30
31	32
33	34
35	36
37	38
39	40
41 ⟷ 60	

	Port Bit	Bit Addresses & Labels
1.		
2.		
3.		
4.		
5.		
6.		
7.		
8.		
9.		
10.		
11.		
12.		
13.		
14.		
15.		
16.		
17.		
18.		
19.		
20.		
21.		
22.		
23.		
24.		
25.		
26.		
27.		
28.		
29.		
30.		
31.		
32.		
33.		
34.		
35.		
36.		
37.		
38.		
39.		
40.		

Software Initializations

A) Port I/0

```
P0MDOUT |= 0xB0
```

B) Timers

C) Interrupts

```
EA = 1;
EIE1 = 0x40;
```

D) A/D

E) PCA

```
PCA0MD = 0x00
PCA0CPM0 = 0xC2;
PCA0CPM1 = 0x00;
PCA0CPM2 = 0xC2;
PCA0CPM3 = 0xC2;
```

F) XBAR

```
XBR0=0x27
```

G) I2C

Pseudocode

Lab 3-1
Partner 1

Compiler directives
Declare global variables
Function prototypes

Main function
    Declare local variables
    Initialize system, ports and PCA
    Begin infinite loop
        Set pulsewidth
End main function

Set pulsewidth
    Declare local variables
    Wait for keystroke
    If “f” increment motor_pw by a value of 10
        If pw is greater than the maximum value, make it limit down to the max value
    Else if “s” decrease motor_pw by 10
        If pw is less than the minimum value, limit it upwards toward the minimum value
    Else if another letter, give a reminder
    Update the speed command
    Print motor_pw

Update lo byte of CCMModule 1
Update hi byte of CCMModule 1

PCA ISR
    If PCA interrupt flag is set
        write lo byte of start_count in PCA0
        write hi byte of start_count in PCA0
        clear interrupt flag
    Else
        handle other PCA interrupts

Partner 2
Compiler directives
Declare global variables
Function prototypes

Main function
    Declare local variables
    Initialize system, ports and PCA
    Begin infinite loop
        Set pulsewidth
End main function

Set pulsewidth
    Declare Local Variables
    Set step size of 10 for changing pulse width
    Wait for keystroke
    If “l” move wheel left
    If “r” move wheel right
    Wait until steering not centered
        Center the steering
        Set this value
        Store value
    If steering not at the maximum value
        Bring steering to maximum value
        Set the value
        Store value as minimum pulse width
    If steering not at the right maximum value
        Bring steering to maximum value
    Set the value
        Store value as maximum pulse width

Update lo byte of CCMModule 1
Update hi byte of CCMModule 1

PCA ISR
    If PCA interrupt flag is set
        write lo byte of start_count in PCA0
        write hi byte of start_count in PCA0
        clear interrupt flag
    Else
        handle other PCA interrupts

Partner 3
Compiler directives
Declare global variables
Function prototypes

Main function
- Declare local variables
- Initialize system, ports and PCA
- Begin infinite loop
  - Set pulsewidth

End main function

Set pulsewidth
- Declare Global Variables
- Wait for keystroke
- If “d” dim the light
- If “b”, make the light more bright
- Set Step Size of 200
- If Led Not at dimmest
  - Find Pulse Width which will make the LED dimmest
  - Store this value
  - Print value
- If Led Not at brightest
  - Find Pulse Width which will make the LED brightest
  - Store this value
  - Print value
- Vary Led brightness as needed with range given

Update lo byte of CCMModule 1
Update hi byte of CCMModule 1

PCA ISR
- If PCA interrupt flag is set
  - write lo byte of start_count in PCA0
  - write hi byte of start_count in PCA0
  - clear interrupt flag
- Else
  - handle other PCA interrupts