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

$\boxed{41} \longleftrightarrow \boxed{60}$

	Port Bit	Bit Addresses & Labels
1.		
2.		
3.		
4.		
5.		
6.		
7.		
8.		
9.		
10.	P1.2	Motor
11.	P1.3	LED
12.	P1.0	Steering
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

P1MDOUT |= 0x0D

B) Timers

C) Interrupts

EA = 1
EIE1 |= 0x08

D) A/D

E) PCA

PCA0MD = 0x81;
PCA0CPM1 = 0xC2;
PCA0CN |= 0x40;

F) XBAR

XBR0 = 0x27

G) I2C


```
Complier directives
        #include<c8051_SDCC.h>
        #include <stdio.h>
        #include<stdlib.h>
        #define PW_MIN _____
         #define PW_MAX _____
        #define PW_NEUT _____
Function Prototypes
        Void Port_Init(void);
        Void Timer_Init(void);
        Void Interrupt_Init(void);
        Void Timer0_ISR(void) __interrupt 1;
        void PCA_Init (void)
        void XBR0_Init(void)
        void drive_motar(void)
        void steering servo(void)
        void LEDblink(void)
Global variables
        Sbit LED0 BUZZER SLDSW
        unsigned int MOTOR_PW = 0;
        unsigned int steering-servo
        unsigned int LED brightness
Main function
        Declare local variables
                (none)
        Initialize function
        Sys_Init();
        putchar(‘ ‘); //the quotes in this line may not format correctly
        Port_Init();
        XBR0_Init();
         PCA_Init();
        Print some message to indicate start
        Begin infinite loop
        Drive motor
        End main function
Void drive motor(void){
        Initialize speed controller , need to leave it at that value for one second(use counter at PCA ISR)
        USE BUZZER during initialization time
        Get an input char from keyboard
        If it is s change the pulsewidth signal and reduce the engine power
        If it is f change the pulsewidth signal and increase the engine power

        Remember The period must be 20ms and the pulsewidth must be initialized to be 1.5ms.
        And use sysclk/12 and 16 bit counter
        Pulsewidth should bigger than 1.1ms and smaller than 1.9ms
}
void steering servo(void){
        Initialize steering servo
        USE BUZZER during initialization time
        Get and input char from keyboard
        If it is r increase the steering pulsewidth by 10 (turn right more)
        If it is l decrease the steering puslswidth by 10(turn left more)

        Remember The period must be 20ms and the pulsewidth must be initialized to be 1.5ms.
        use sysclk/12 and 16 bit counter
        Initial estimates of the left and right limits are 0.9 [ms] and 2.1 [ms].

}
```

```
Void LED(void){
    Initialization (almost same as the 2 function before)
    USE BUZZER during initialization time
    Get and input char from keyboard
    If it is b turn the LED BRIGHTER
    If it is d turn the LED DIMMER

    //Remember The period must be 20ms and the pulsewidth must be initialized to be 1.5ms.
    //use a Crossbar setting XBR0=0x27 with a pulsewidth signal on CEX3
    //use sysclk/12 and 16 bit counter
    //The pulsewidth should never be less than 1[ms] or greater than 19[ms].
}
Void Port_init(void)
{
    Set up port pin 1
}
Void Timer_Init(void){
    Set up Timer
}
Void Interrupt_Init(void) __interrupt 9{
    Set up interrupt
}
```

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?

SYSCLK/12 overflow

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, 10.06ms

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

0.001628s, 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

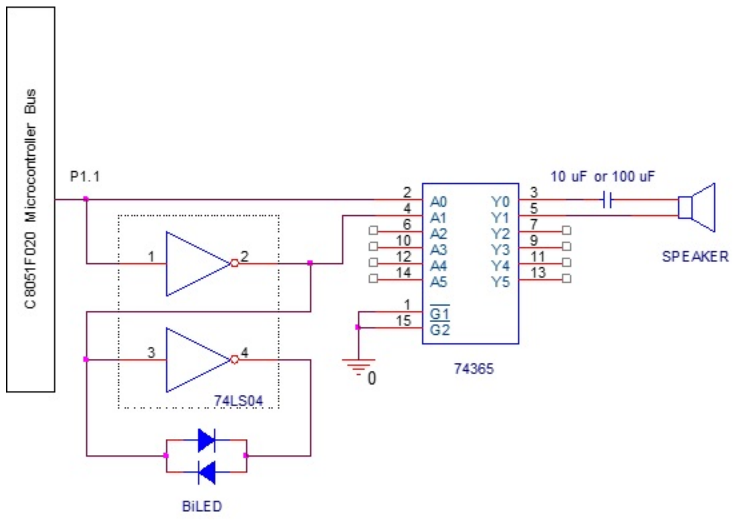

Figure 1: Potentiometer Circuit

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 the BILED is wired to display the amount of high or low output of CEX1, the amount of one color will change when the duty cycle (PW/period) moves from 0% to 100%

b) Part B, changing duty cycle

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

b. Change the PCA start value, *PCA_starth*, 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.

Since the speaker vibrate accordingly to the output frequency of CEX1, when PCA_start increase, the period gets shorter therefore the frequency gets higher (f=1/T).

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

Its blinking rate changes accordingly.

When complete, include Worksheet 6 with your Laboratory 2 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 ;
SDL	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	CEX2 ;
Steering Servo	CEX0 ;
LED	CEX3 ;

When complete, include Worksheet 6 with your Laboratory 2 Pre-lab submission.