

## First Experiments Part 3

1. "CCP" and PPS
2. "CCP and ADC"





# Contents

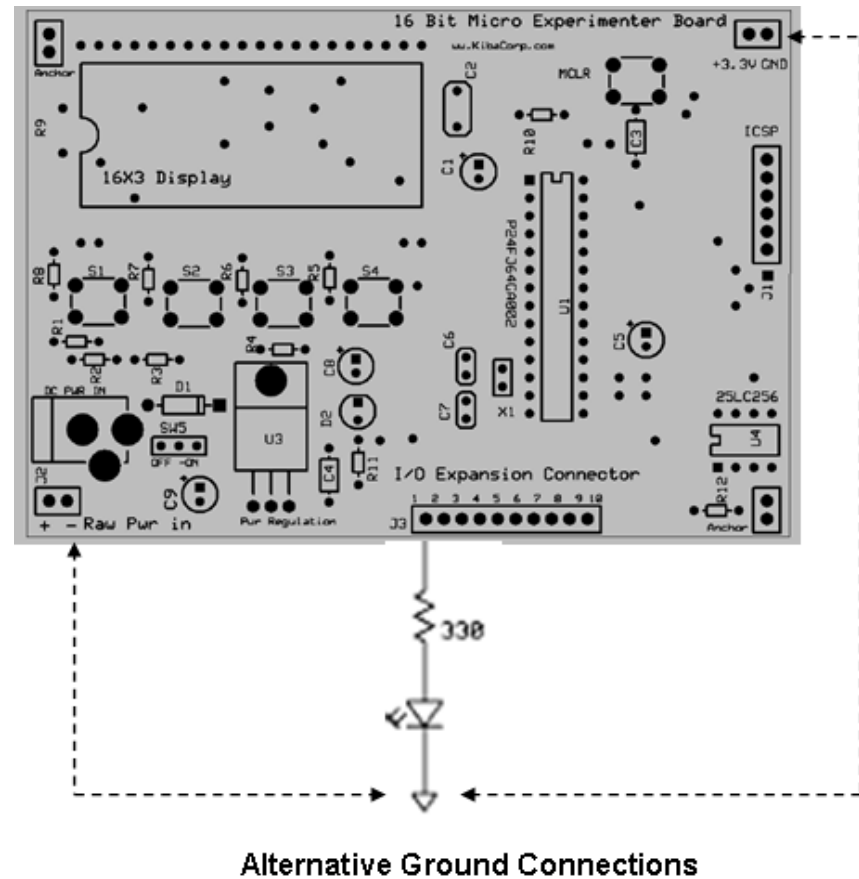
- Directions and descriptions for beginner 16 bit Experimenter starting experiments
  - Each experiment builds on another
  - Theme has been LED light control
- Recommend review of “Quick Start Guide” ( available on [www.kibacorp.com](http://www.kibacorp.com)) to better understand tools installation
- Good references
  - Microchip datasheet for PIC24F64GA002  
<http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en026374#1>
  - Di Jasio Book “Programming 16 bit Microcontroller in C”  
<http://www.flyingthepic24.com/>





# LED Light Dimming using CCP

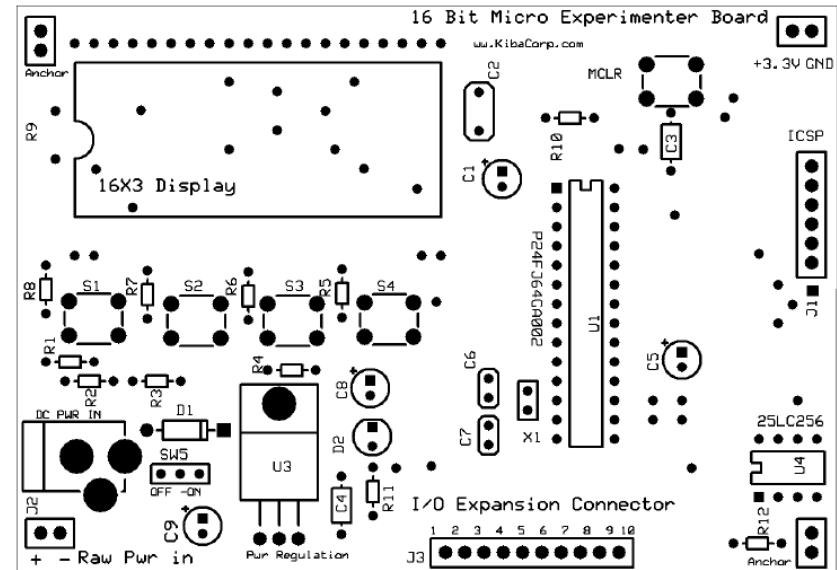
- Use a digital peripheral within the PIC24F to pulse width modulate +3.3V to 0 V DC to LED thereby controlling its brightness
- Similar hardware hook up to earlier experiments
- We will use software to change functionality of pin 1 of expansion Bus





# Many possible configurations for PIN 1?

- Pin 1 can have many configurations
- Normal PIC24F Chip level Pin assignments
  - Analog:
    - AN14 analog ADC input channel 14
    - Comparator input (C1IN)
  - Digital
    - SDA2 (I2C data in)
    - CN16 –change detect 16
    - Simple Digital Port I/O Port B pin 2 (RB2)
- Peripheral programmable Assigned using RP2
  - **Redirect in/out of any other on chip digital peripherals (UART,SPI, CCP) to this pin**
  - **We are interested in using RP2 as a CCP OC digital PWM function**



J3 I/O Expansion Bus		
J3 I/O bus PIN	Normal Assigned PIN	Programmable PIN
PIN 1	AN4,C1IN,-SDA2,CN6,RB2	RP2
PIN 2	AN5,CIN+,SCL2,CN7,RB3	RP3
PIN 3	SDA1,CN27,RB5	RP5
PIN 4	INT0,CN23,RB7	RP7
PIN 5	SCL1/CN22/RB8	RP8
PIN 6	SDA1/CN21/RB9	RP9
PIN 7	AN12/CN14,RB12	RP12
PIN 8	AN11/CN13,RB13	RP13
PIN 9	AN10,CVREF,RTCC,CN12,RB14	RP14
PIN 10	AN9,CN11,RB15	RP15



# Rules for Pin Priorities

- Both analog and PPS are high priority but require a SW configuration
    - We will examine how this is done configuring RP2 PIN 1 of Exp 16 Expansion I/O bus as a CCP OC PWM output
  - Next priority is fixed digital peripherals—again they need to be SW configured
  - Digital I/O is enabled during power up *–but is configured as input*
- **Peripheral priorities:**
    1. **Analog Functions** ANx, Vref+/-
    2. **PPS Outputs** UART TX, SDO, OC
    3. **PPS Inputs** UART RX, SDI, IC
    4. **Fixed Digital Peripheral Outputs** I<sup>2</sup>C™, CN, I/O Ports
    5. **Fixed Digital Peripheral Inputs** I<sup>2</sup>C, CN, I/O Ports



# Uses of Capture Compare Peripheral (CCP)

- Multiple Capabilities

- Non-software intensive

- Capture and Input Compare (IC)

- Measure the period of an external square wave with or without averaging

- Measuring Pulse width of external signals

- – Output Compare (OC)

- Can generate PWM and other specific modulation formats



# Output Compare and PWM

- PIC24F64GA002 has 5 CCP peripherals
- Timer 2 or Timer 3 as time base
- 16 Bit Compare
- • We will configure CCP1 OC1 for PWM generation



# Uses of PWM

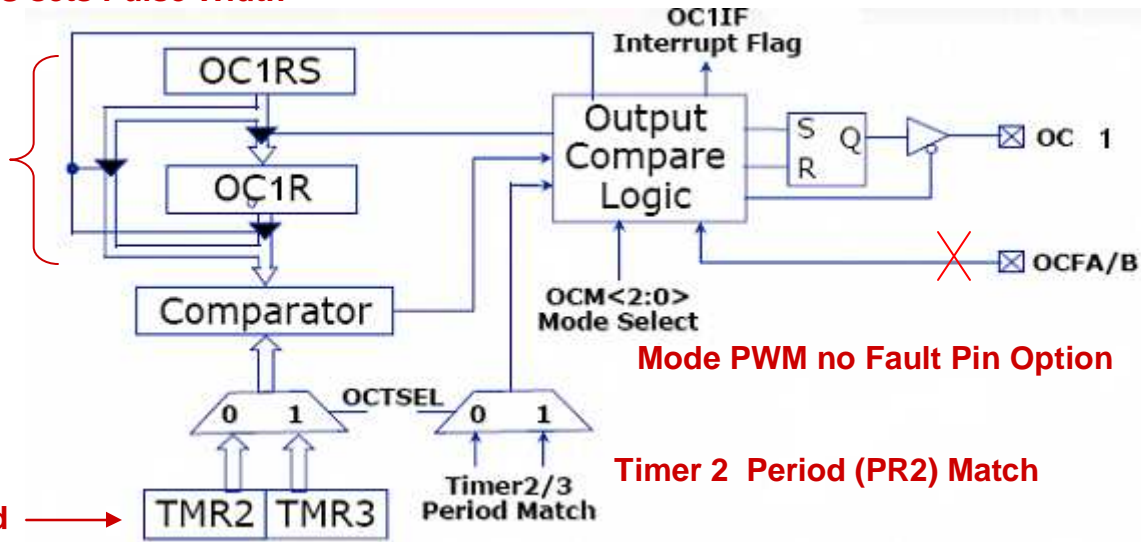
- Can generate controls for motors
- Can generate through addition of a LOW PASS Filter analog signals
- Can generate tones with a buzzer
- • Can work in light dimming



# Output Compare OC1

**OC1RS sets Pulse Width**

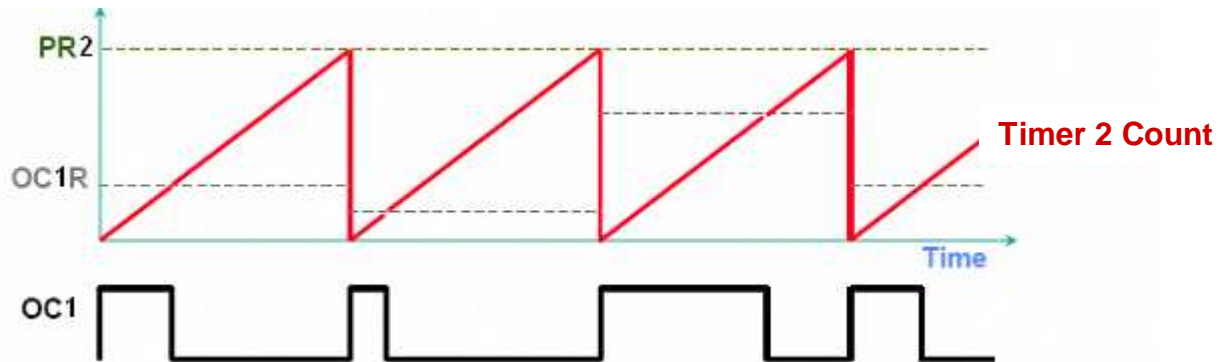
**OC1RS auto loads OC1R every PR2**



**Mode PWM no Fault Pin Option**

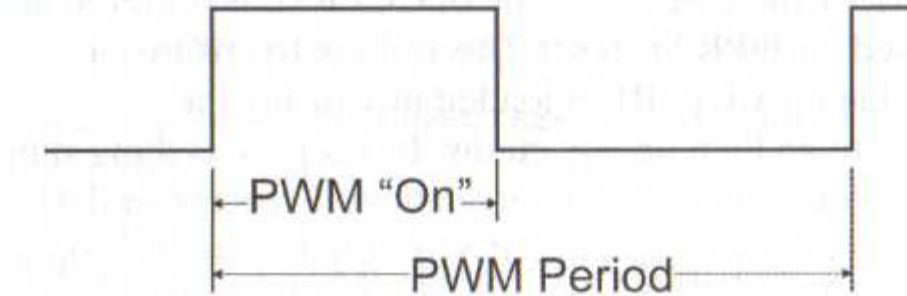
**Timer 2 Period (PR2) Match**

**PR2 sets period For Timer 2**





# OC 1 PWM Output



**Duty Cycle = (PWM "On" time) / PWM period**

**If PWM waveform swings between +3.3 to 0 Volts then the Duty Cycle determines the average DC delivered to a load**

***--so we can control DC voltage to LED  
thereby controlling its brightness!***



# Peripheral Pin select

- With smaller Chip packages ( with a large number of internal peripherals) not all the internal peripherals can be assigned to pins on the package
- Peripheral Pin Select of PPS is pin multiplexing that allows the user to select the pin out of digital functions
- Allows optimal usage of on-chip peripherals
- Allows pin redefinition via software

***CCP 1 OC1 output will need to be mapped to Pin 1 I/O Exp Bus using PPS***

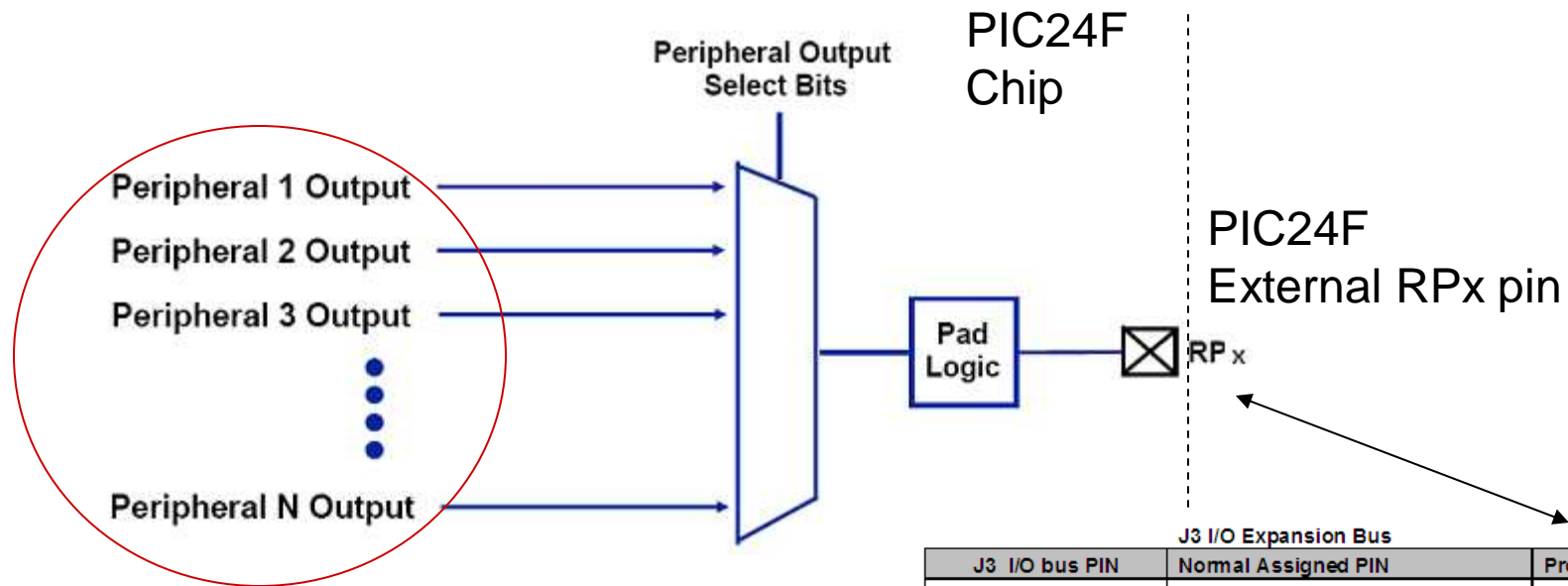


# PPS Implementation Details

- **Any function can remap to any RP pin**
  - Multiple functions on one pin is supported
- **Inputs vs. Outputs**
  - Inputs assign a pin to a specific peripheral
  - Outputs assign a peripheral to a specific pin
- **Pinout is set in software**
  - Allows on-the-fly configuration or one-shot configuration



# Remappable Outputs



1 of 5 CCP Out , 1 of 2 UART TX ,  
 1 of 2 SPI CLK or SDO,  
 1 of 2 Comparator Out

J3 I/O Expansion Bus

J3 I/O bus PIN	Normal Assigned PIN	Programmable PIN
PIN 1	AN4,C1IN-,SDA2,CN6,RB2	RP2
PIN 2	AN5,CIN+,SCL2,CN7,RB3	RP3
PIN 3	SDA1,CN27,RB5	RP5
PIN 4	INT0,CN23,RB7	RP7
PIN 5	SCL1/CN22/RB8	RP8
PIN 6	SDA1/CN21/RB9	RP9
PIN 7	AN12/CN14,RB12	RP12
PIN 8	AN11/CN13,RB13	RP13
PIN 9	AN10,CVREF,RTCC,CN12,RB14	RP14
PIN 10	AN9,CN11,RB15	RP15

16 bit Exp Expansion Bus



# Experimenter 16 Expansion I/O

- Each of the 10 pins has both fixed and programmable functionality
- RPx indicates the PPS designator for the Pin

**J3 I/O Expansion Bus**

J3 I/O bus PIN	Normal Assigned PIN	Programmable PIN
PIN 1	AN4,C1IN-,SDA2,CN6,RB2	RP2
PIN 2	AN5,CIN+,SCL2,CN7,RB3	RP3
PIN 3	SDA1,CN27,RB5	RP5
PIN 4	INT0,CN23,RB7	RP7
PIN 5	SCL1/CN22/RB8	RP8
PIN 6	SDA1/CN21/RB9	RP9
PIN 7	AN12/CN14,RB12	RP12
PIN 8	AN11/CN13,RB13	RP13
PIN 9	AN10,CVREF,RTCC,CN12,RB14	RP14
PIN 10	AN9,CN11,RB15	RP15

A large curly bracket on the left side of the table groups all 10 rows. An arrow points from the bottom of the table to the text "RPx indicates the PPS designator for the Pin" in the list above.



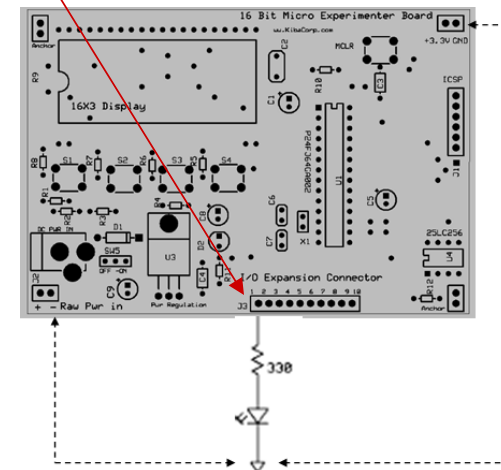
# Remappable Peripheral Outputs

Function	Output Function Number <sup>(1)</sup>	Output Name
NULL <sup>(2)</sup>	0	NULL
C1OUT	1	Comparator 1 Output
C2OUT	2	Comparator 2 Output
U1TX	3	UART1 Transmit
U1RTS <sup>(3)</sup>	4	UART1 Request To Send
U2TX	5	UART2 Transmit
U2RTS <sup>(3)</sup>	6	UART2 Request To Send
SDO1	7	SPI1 Data Output
SCK1OUT	8	SPI1 Clock Output
SS1OUT	9	SPI1 Slave Select Output
SDO2	10	SPI2 Data Output
SCK2OUT	11	SPI2 Clock Output
SS2OUT	12	SPI2 Slave Select Output
OC1	18	Output Compare 1
OC2	19	Output Compare 2
OC3	20	Output Compare 3
OC4	21	Output Compare 4
OC5	22	Output Compare 5

J3 I/O Expansion Bus		
J3 I/O bus PIN	Normal Assigned PIN	Programmable PIN
PIN 1	AN4,C1IN-,SDA2,CN6,RB2	RP2
PIN 2	AN5,CIN+,SCL2,CN7,RB3	RP3
PIN 3	SDA1,CN27,RB5	RP5
PIN 4	INT0,CN23,RB7	RP7
PIN 5	SCL1/CN22/RB8	RP8
PIN 6	SDA1/CN21/RB9	RP9
PIN 7	AN12/CN14,RB12	RP12
PIN 8	AN11/CN13,RB13	RP13
PIN 9	AN10,CVREF,RTCC,CN12,RB14	RP14
PIN 10	AN9,CN11,RB15	RP15

This is the pin that We want to map it to OC1 of using RP2

This is the peripheral we select for our experiment Output function # 18



Alternative Ground Connections



# Making it Happen

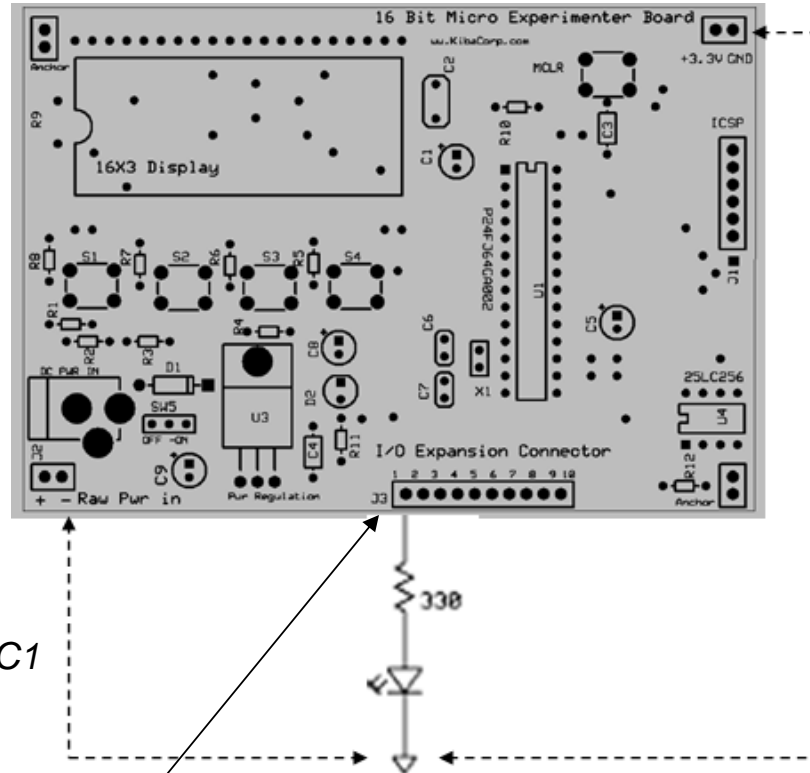
//Required Code Statement  
// 'C' structure

RPOR1bits.RP2R = 18;

SFR RPOR1  
Controls RP2/RP3  
PPS assignments

RP2  
Configure  
Bits with  
RPOR1

PPS assignment  
Function # 18 ->OC1



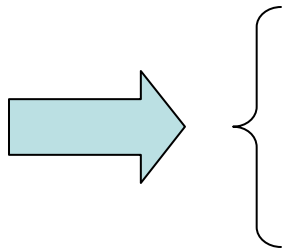
Alternative Ground Connections

**This now transforms from a digital output to an CCP module generating PWM**



# Initializing a PPS Application

- Initialize the pin out by mapping pins to desired peripheral input/output functions
  - Unlock
  - Map RPx pins -> Peripheral Input Functions
  - Map RPx pins -> peripheral Output Functions
  - Lock the RPx SFR(s) using 'lock sequence
- Configure peripherals later in your code before use.



```
45 main()
46 {
47
48     __builtin_write_OSCCONL(OSCCON & 0xbf) ; //clear bit 6  unlock
49
50     //input//  PPS first -there are none
51
52     //output // PPS second
53     //do PPS on RB2 pin 1 of I/O EXP  to make it OC1 output
54     RPOR1bits.RP2R = 18;          // Make Pin 1 I/O EXP bus (RP2)  OC1
55
56     __builtin_write_OSCCONL(OSCCON | 0x40) ; //set bit 6 lock
57
58     int setting =DutyCycle100percent ;
59
60     setPWM(setting);
61
62     while(1);
63 } // main
64
```



# CCP Code


```
16 #define NoDutyCycle      0
17 #define DutyCycle12percent 0x03ff/8
18 #define DutyCycle25percent 0x03ff/4
19 #define DutyCycle50percent 0x03ff/2
20 #define DutyCycle100percent 0x03ff
21
22 void turnoff(void) {
23     //turn off pin 1 I/O EXP RB2, Timer2 and any OC1 configuration on this pin
24     T2CON =0;
25     OC1CON =0x0000;
26     TRISBbits.TRISB2 = 1;
27 }
28
29 void turnon(int setting) {
30     //turn on pin 1 I/O EXP RB2, Timer2 and any OC1 configuration on this pin
31     PR2=DutyCycle100percent ;
32     TMR1 =0;
33     OC1R =setting;
34     OC1RS =setting;
35     OC1CON = 0x0006; //timer2 and edge aligned PWM
36     T2CON =0x8000;
37 }
38
39 void setPWM(int setting) {
40     turnoff();
41     turnon(setting);
42 }
43
44
45 main()
46 {
47
48     __builtin_write_OSCCONL(OSCCON & 0xbf) ; //clear bit 6 unlock
49
50     //input// PPS first -there are none
51
52     //output // PPS second
53     //do PPS on RB2 pin 1 of I/O EXP to make it OC1 output
54     RPOR1bits.RP2R = 18; // Make Pin 1 I/O EXP bus (RP2) OC1
55
56     __builtin_write_OSCCONL(OSCCON | 0x40) ; //set bit 6 lock
57
58     int setting =DutyCycle100percent ;
59
60     setPWM(setting);
61
62     while(1);
63 } // main
```



# CCP Experiment

- Map red led to output compare 1 to perform PWM on led thereby controlling its brightness
- Open Loop.mcp
- Examine source
- What is the RPX pin are we configuring?
- Where does the PPS occur?
  - Note lock and unlock sequences
- What is the initial duty cycle of the PWM?
  - Compare PR2 with OC2RS
- Select Debug to PICKIT2
- Build, download and Run
- Verify operation
- Change duty cycle, build, download, run and notice effects on LED
  - (Brightness should be changing)

```
#define NoDutyCycle          0
#define DutyCycle12percent  0x03ff/8
#define DutyCycle25percent  0x03ff/4
#define DutyCycle50percent  0x03ff/2
#define DutyCycle100percent 0x03ff
```

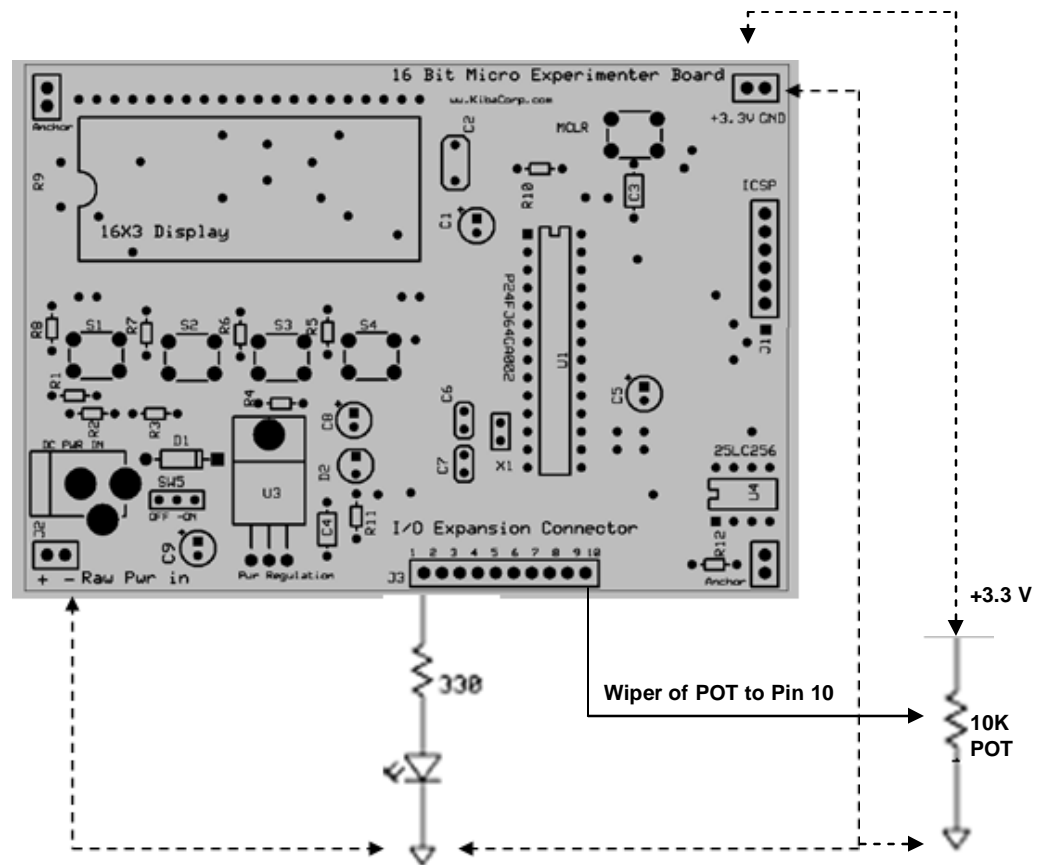


```
int setting =DutyCycle100percent ;
setPWM(setting) ;
```



# LED Light Dimming using CCP and ADC

- Use a digital peripheral within the PIC24F to pulse width modulate +3.3V to 0 V DC to LED thereby controlling its brightness
- Similar hardware hook up to earlier experiments but we are now adding a 10K pot as shown
- We will use Pin 1 as a OC1 for PWM of LED and Use Pin 10 as a analog input to control duty cycle of PWM through POT setting
- As you move wiper CW or CCW ADC readings will be placed in OC1 OC1RS to control pulse width. Reading should be controlled between 0-0x3ff or 0 to 100% duty cycle



Alternative Ground Connections



# ADC

- The PIC24 Analog to Digital Converter (ADC) is a 10 bit converter that has up to 11 input channels and performs conversion up to 500K per sec.
- With the Experimenter 5 of these 11 channels are available to as ADC inputs with the I/O Expansion Bus.
- They are I/O expansion pins 1, 7, 8, 9, and 10.

J3 I/O bus PIN	Normal Assigned PIN	Programmable PIN
PIN 1	AN4,C1IN-,SDA2,CN6,RB2	RP2
PIN 2	AN5,CIN+,SCL2,CN7,RB3	RP3
PIN 3	SDA1,CN27,RB5	RP5
PIN 4	INT0,CN23,RB7	RP7
PIN 5	SCL1/CN22/RB8	RP8
PIN 6	SDA1/CN21/RB9	RP9
PIN 7	AN12/CN14,RB12	RP12
PIN 8	AN11/CN13,RB13	RP13
PIN 9	AN10,CVREF,RTCC,CN12,RB14	RP14
PIN 10	AN9,CN11,RB15	RP15

**We will be configure pin 10 to be analog input 9 of ADC  
We will leave PIN 1 as an OC1**



# KibaCorp ADC Library

- An ADCDEMO program using the ADC library is made available
- Using our ADC library we are able to make any of these pins act as an ADC input and perform an ADC conversion.
- The conversion automatically converts voltage signal at the input of the pin between 0 volts to +3.3 Volts to a binary ten bit value 0 to 1023 representing that voltage
- The ADC library functions that are used are:
- **InitADC (IOpin)** – initializes the designated Expansion bus pin for ADC operation as an input channel. There are only five possible pins available for this use on the I/O expansion and they are designed in the code as :
  - **pin1, pin7, pin8, pin9, pin10** (make sure to use exact syntax for pin shown here). This function must be called first before any ADC library function. Only one pin can be designated as an input at a time.
- **ReadADC (IOpin)** - this function performs an ADC conversion on the input voltage present on the pin (see pin designations shown above) and returns an integer value from 0 to 1023 representing the measured voltage.



# CCP with ADC

```
17 #define DELAY 8000
18 int adcvalue =0; //creaet adc output variable
19
20 void turnoff(void) {
21     //turn off pin 1 I/O EXP RB2, Timer2 and any OC1 configuration on this pin
22     T2CON =0;
23     OC1CON =0x0000;
24     TRISBbits.TRISB2 = 1;
25 }
26 void turnon(int setting) {
27     //turn on pin 1 I/O EXP RB2, Timer2 and any OC1 configuration on this pin
28     PR2=0x03ff ;
29     TMR1 =0;
30     OC1R =setting;
31     OC1RS =setting;
32     OC1CON = 0x0006; //timer2 and edge aligned PWM
33     T2CON =0x8000;
34 }
35 void setPWM(int setting) {
36
37     turnoff();
38     turnon(setting);
39 }
40 main()
41 {
42     __builtin_write_OSCCONL(OSCCON & 0xbf) ; //clear bit 6 unlock
43     //input// PPS first -there are none
44     //output // PPS second
45     //do PPS on RB2 pin 1 of I/O EXP to make it OC1 output
46     RPOR1bits.RP2R = 18; // Make Pin 1 I/O EXP bus (RP2) OC1
47
48     __builtin_write_OSCCONL(OSCCON | 0x40) ; //set bit 6 lock
49
50     T1CON = 0x8030; // TMR1 on, prescale 1:256 Tclk/2 --use timer1 for delay
51     initADC(pin10); //make pin 10 an analog input a9
52
53     while( 1)
54     {
55         //1. change PWM to ADC value
56         adcvalue = readADC(pin10); //read ADC
57         setPWM(adcvalue); // set PWM
58
59         // 2. delay for min time using timer1 read before next change
60         TMR1 = 0;
61         while ( TMR1 < DELAY)
62         {
63             ;
64         }
65     } // main
```



# CCP with ADC Experiment

- Hook up hardware as shown in earlier slide
- Open Loop.mcp
- Examine source –examine where ADC value is used to change PWM setting
- Note that Timer 1 is used as a delay
- Select Debug to PICKIT 2
- Build, download and Run
- Verify operation -Change POT set CW and CCW and note instantaneous LED brightness change