



MICROCHIP

Regional Training Centers

Section 11
Output Compare PWM

What's PWM

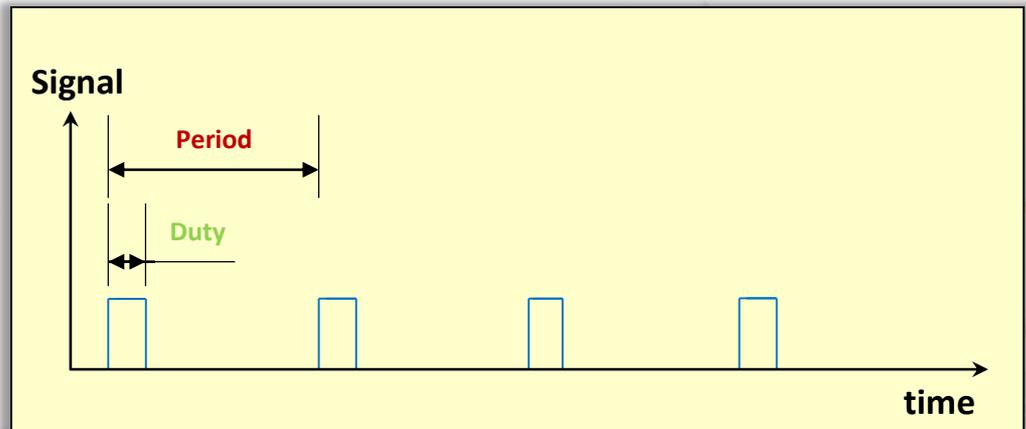
- The Pulse-width modulation (PWM) Waveform is a modulation technique used to encode a message into a pulsing signal.
- Its main use is to allow the control of the power supplied to electrical devices, ex. motor control, ballast, LED, H-bridge, power converters, and other types of power control applications.
- There have two important terms

- ◆ **Period :**

The single square wave duration.

- ◆ **Duty :**

The proportion of active time to the regular interval



PIC24F Output Compare

- ◆ PIC24FJGB provide below modes for Output Compare Module (OCM<2:0>):

Simple Compare Match Mode:

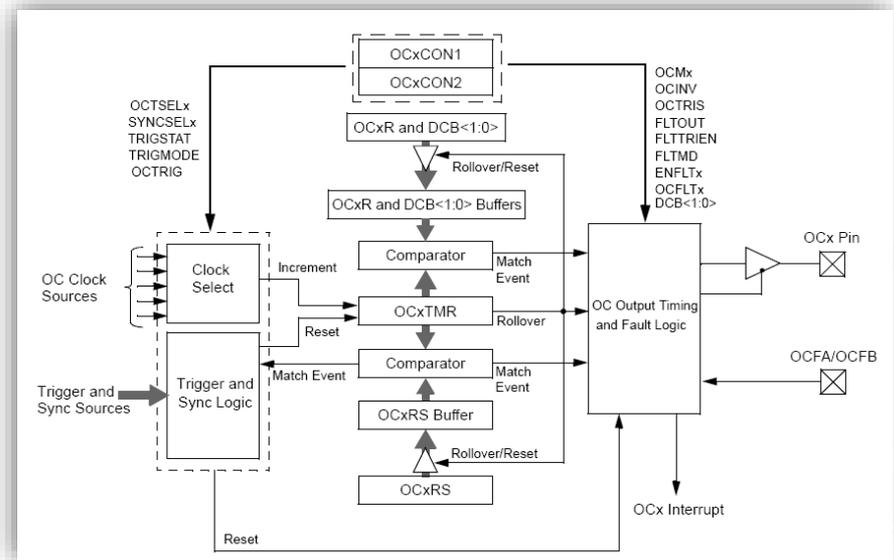
1. Normal Low, High on Match
2. Normal High, Low on Match
3. Toggle on Match

Dual Compare Mode:

1. Single Pulse
2. Continuous pulse

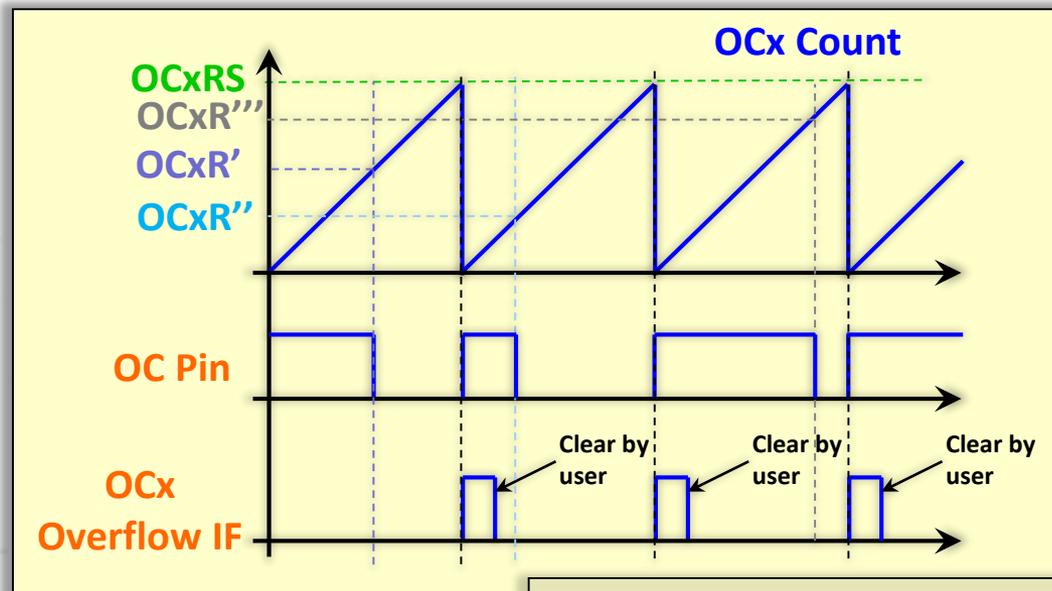
Simple PWM Mode:

1. Edge-Aligned PWM Mode
2. Center-Aligned PWM Mode



Edge-Aligned PWM Mode

- For Edge-Aligned PWM Mode, the period time (T) is controlled by the OCxRS register. OCx pin change stats on each OCx counter = OCxRS and OCxR condition.



Rule

1. OCx Counter = OCxR -> Output Set to low
2. OCx Counter = OCxRS -> Output Set to high

Lab14 Edge-Aligned PWM



Lab14 Edge-Aligned PWM

- ◆ Try to initial OC1 module to Edge-Aligned PWM Mode, than use PWM to control LED brightness.
- ◆ OC1 module clock source set to F_{CY} ($F_{osc} / 2$), Sync Self, Frequency is 1KHz, Duty cycle 10%.
- ◆ Please connect **RG6(OC1)** to **LED(D6)** to observe LED brightness.

◆ **Let's go!**

Lab14 Edge-Aligned PWM Hits

- *Primary Compare (OCxR, Duty)
- *Secondary Compare (OCxRS, Period)

The screenshot shows the OC1 - Editor window with the following settings:

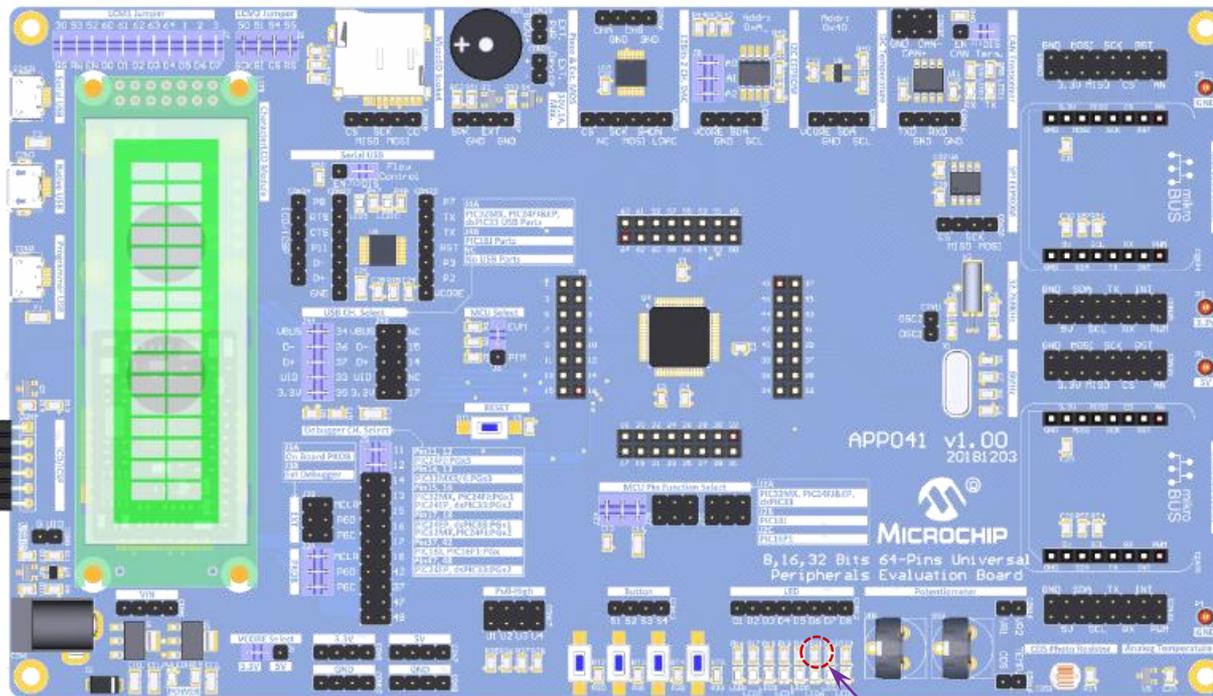
- Mode:** Edge-Aligned PWM mode
- Clock Source:** FOSC/2
- Trigger/Sync Source:** Self
- OCxR, Duty:** 16000
- OCxRS, Period:** 16000

$$PWM_{Duty} = \left(\frac{OCxR}{OCxRS} \right)$$

$$PWM_{freq.} = \left(\frac{Clock\ Source}{OCxRS + 1} \right)$$

Lab14 Edge-Aligned PWM

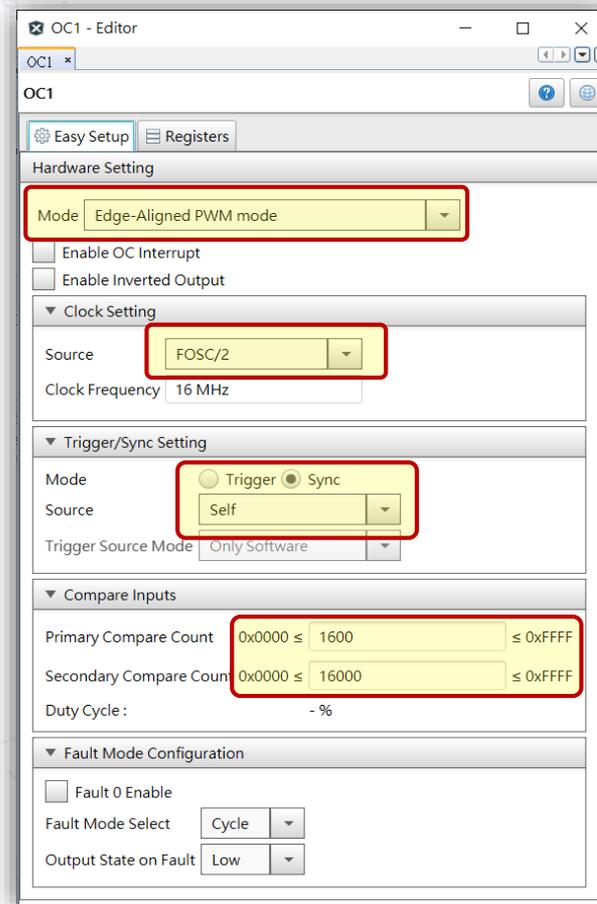
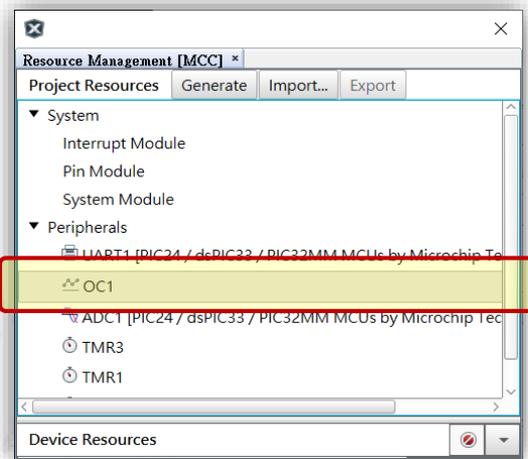
Result



**Slightly bright !
(10% Duty)**

Lab14 Edge-Aligned PWM

MCC's Setting & Code Example



MCC's O.C.PWM Function

- ◆ MCC Provide below common functions:

```
void OC1_Initialize(void)
```

```
    // Initial OC.
```

```
void OC1_Start(void); void OC1_Stop(void);
```

```
    // OC Enable & Disable.
```

```
void OC1_SingleCompareValueSet(value)
```

```
    // Single Compare Mode Setting ( OC1R <- value)
```

```
void OC1_DualCompareValueSet(priVal, secVal)
```

```
    // Dual Compare Mode Setting ( OC1R <- priVal, OC1RS <- secVal)
```

```
void OC1_CentreAlignedPWMConfig(priVal, secVal)
```

```
    // Centre Aligned Mode Setting ( OC1R <- priVal, OC1RS <- secVal)
```

```
void OC1_EdgeAlignedPWMConfig(priVal, secVal)
```

```
void OC1_PrimaryValueSet( uint16_t priVal)
```

```
void OC1_SecondaryValueSet( uint16_t secVal )
```

```
    // Edge Aligned PWM Mode Setting ( OC1R <- priVal, OC1RS <- secVal)
```

100% Duty Issue

◆ How to generate 100% duty PWM ?

◆ Rule Review

1. OCx Counter = OCxR -> Output Set to low
2. OCx Counter = OCxRS -> Output Set to high

◆ 100% is means OCxR = OCxRS must be set for 100% duty output.

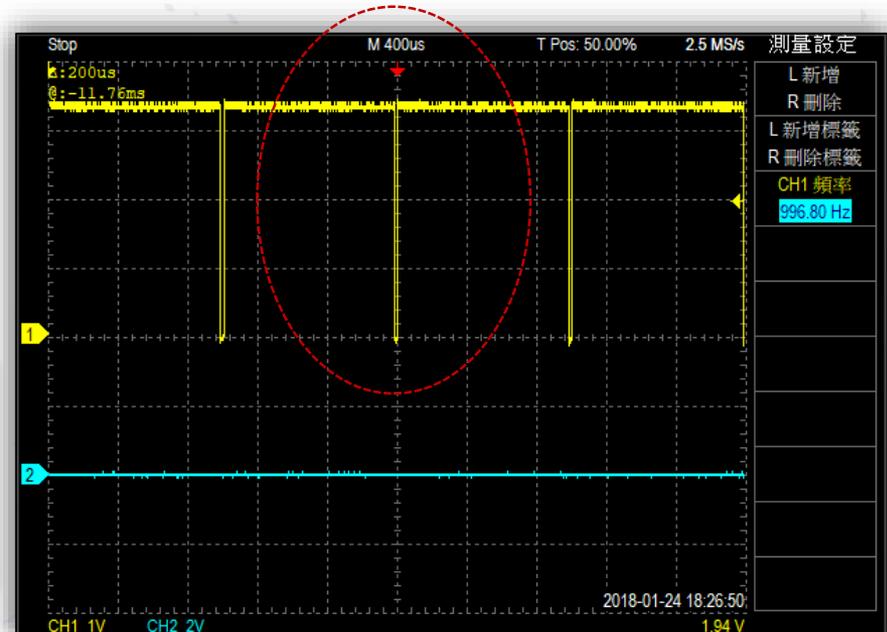
Which rule be executed ?

◆ Result : Glitch !

◆ To prevent this issue

◆ set **OCxR > OCxRS** for 100% duty.

```
if( DutyVlaue >= OCxRS ) /* 100% duty */  
    OC1_PrimaryValueSet(OCxRS+ 1);  
else  
    OC1_PrimaryValueSet(DutyVlaue);
```



Lab15 PWM ADC Duty Adj



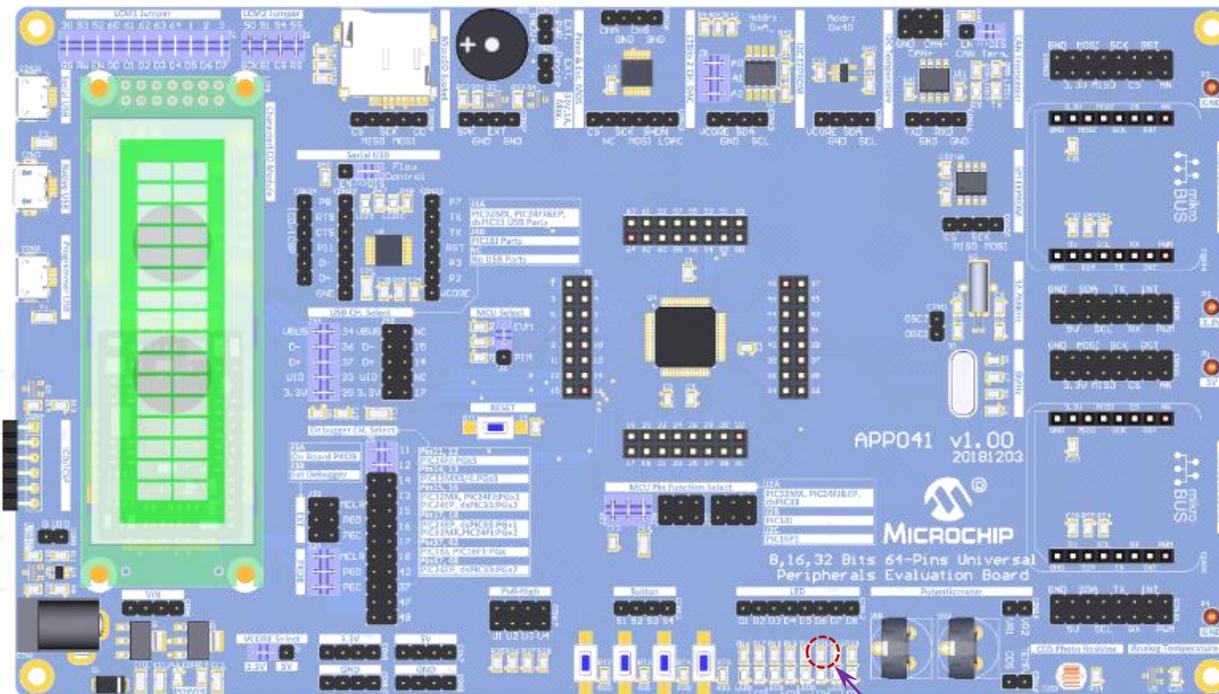
Lab15 PWM ADC Duty Adj

- ◆ Try to use **Potentiometer(VR1)** to control **LED(D6)** brightness.
- ◆ VR Value(0 ~1023) <-> PWM Duty (0% ~ 100%)
- ◆ OC1 Setting same as lab14.

◆ **Let's go!**

Lab15 PWM ADC Duty Adj

Result



dimming Up & Down
by VR1

Lab15 PWM ADC Duty Adj MCC's Setting & Code Example

```
int main(void)
{
    ...
    while (1)
    {
        if (AD1Flag)
        {
            AD1Flag = 0;
            ...

            if (ADCResult[0] >= 1023)
                OC1_PrimaryValueSet(OC1RS + 1);
            else
                OC1_PrimaryValueSet(((long) ADCResult[0] * OC1RS) / 1023);
        }
    }
}
```



```
if( DutyVlaue >= OCxRS ) /* 100% duty */
    OC1_PrimaryValueSet(OCxRS+ 1);
else
    OC1_PrimaryValueSet(DutyVlaue);
```

OC's Duty & Period update

- OC's duty (OCxR) can update at any time, hardware will auto reload the new value at new PWM cycle moment.
- OC's period (OCxRS) only allow update at new PWM cycle moment.
- **Enable OC interrupt more easy to update PWM period.**
- For example:

```
void OC2_CallBack(void)
{
    OC2_SecondaryValueSet(NewPeriodValue);
    OC2_PrimaryValueSet(OC2RS >> 1); /* 50% Duty */
    ...;
}
```

Lab16 PWM Buzzer Tone



Lab16 PWM Buzzer Tone

- ◆ Try to initial OC2 to export special frequency (Tone) to Buzzer, change the tone every second.
 - ◆ `Tone[] = {262,294, 330, 349, 392, 440, 494}; /*Do, Re, Mi, etc.. */`
- ◆ Enable OC2 interrupt, priority set to **7** (highest).
Tyr to change OC2 frequency at OC2 callback function.
- ◆ Please connect **RG9(OC2)** to **Buzzer(SPK)**.

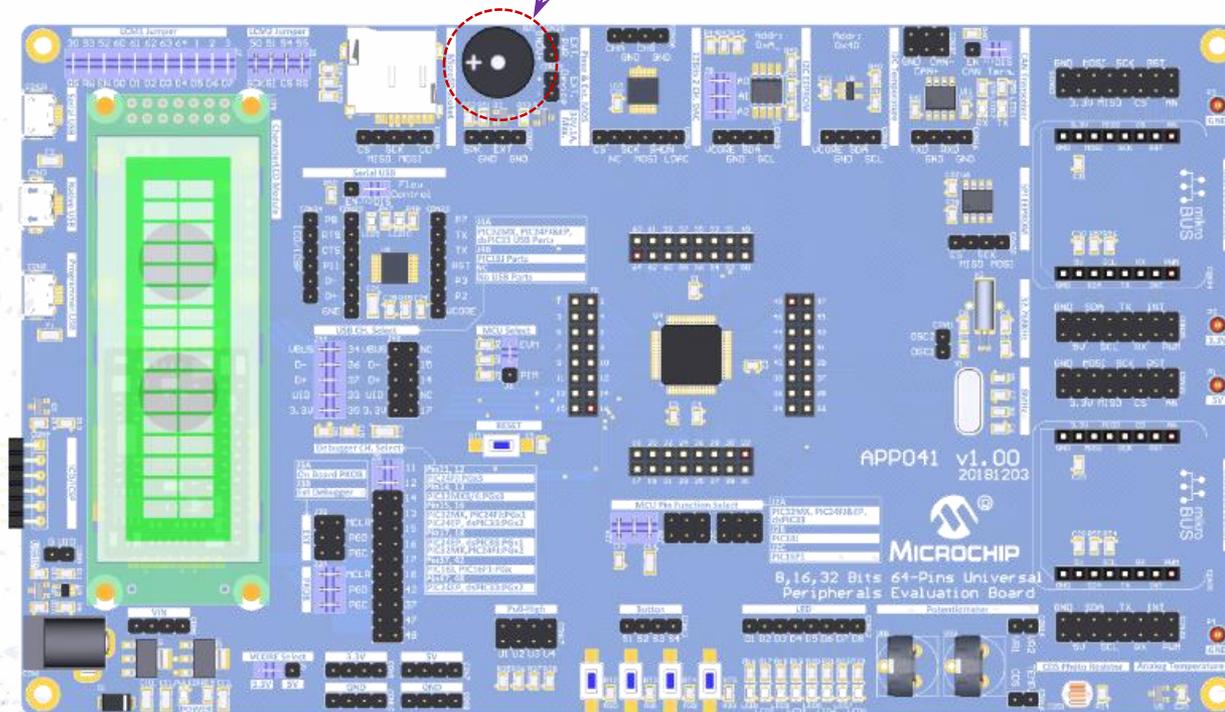
- ◆ **Let's go!**

Lab16 PWM Buzzer Tone

Result



Tone by your code



Lab16 PWM Buzzer Tone

MCC's Setting & Code Example

```
const unsigned int Tone[] = {262, 294, 330, 349, 392, 440, 494};
volatile unsigned char T2Flag = 0;
int main(void)
{
    ...
    while (1)
    {
        ...
    }
}
void TMR2_CallBack(void)
{
    ...
    T2Flag = 1;
}
void OC2_CallBack(void)
{
    static unsigned char i = 0;
    if (T2Flag)
    {
        T2Flag = 0;

        OC2_SecondaryValueSet(16000000 / Tone[i]);
        OC2_PrimaryValueSet(OC2RS >> 1);
        if (++i > 6)
            i = 0;
    }
}
```

$$PWM_{freq.} = \left(\frac{Clock\ Source}{OCxRS + 1} \right)$$

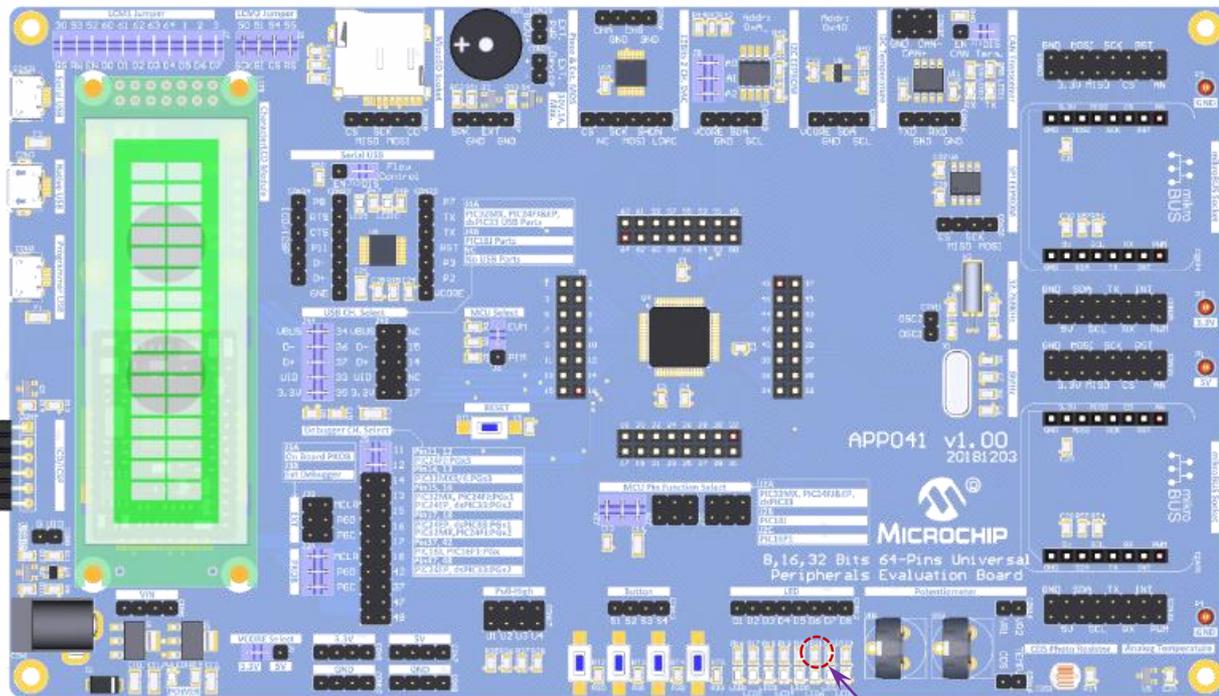
Lab17 PWM Auto Duty Adj



Lab17 PWM Auto Duty Adj

- ◆ Try to use timer to control OC1's PWM duty, automatically.
- ◆ Duty 0% -> Duty 100% -> Duty 0% -> ...
- ◆ Change duty one time per **50ms**.
(2% - 5% duty change every time suggest)

Lab17 PWM Auto Duty Adj Result



dimming up and down automatically



Lab17 PWM Auto Duty Adj

MCC's Setting & Code Example

The screenshot shows the 'TMR4 - Editor' window. The 'Easy Setup' tab is active. Under 'Hardware Settings', 'Enable TMR' is checked. The 'Timer Clock' section shows 'Clock Source' as FOSC/2, 'Input Frequ...' as 16 MHz, and 'Prescaler' as 1:64. The 'Bit Mode' is set to 16 Bit. The 'Timer Period' section shows 'Period Count' as 0x0 ≤ 0x30D4 ≤ 0xFFFF, and 'Timer Period' as 4 us ≤ 50 ms ≤ 262.14 ms. The 'Calculated Period' is 50 ms. Under 'Software Settings', 'Enable Timer Interrupt' is checked, and the 'Callback Function Rate' is 0x1, with 'xTimer Period =' set to 50 ms.

The screenshot shows the 'Interrupt Module - Editor' window. The 'Easy Setup' tab is active. The 'Interrupt Manager' table is displayed, with the 'TMR4' row highlighted in yellow.

Module	Interrupt	Description	IRQ Nu...	Enabled	Priority
Pin Module	CNI	CN - Change Notifica...	19	<input type="checkbox"/>	1
ADC1	ADI	ADC1 - A/D Converte...	13	<input checked="" type="checkbox"/>	5
UART1	UERI	U1E - UART1 Error	65	<input checked="" type="checkbox"/>	6
UART1	UTXI	U1TX - UART1 Trans...	12	<input checked="" type="checkbox"/>	6
UART1	URXI	U1RX - UART1 Receiver	11	<input checked="" type="checkbox"/>	6
OC1	OCI	OC1 - Output Compa...	2	<input type="checkbox"/>	1
TMR4	TI	T4 - Timer4	27	<input checked="" type="checkbox"/>	1
TMR4	TNI	T5 - Timer5	28	<input type="checkbox"/>	1
TMR3	TI	T3 - Timer3	8	<input type="checkbox"/>	1
TMR2	TI	T2 - Timer2	7	<input checked="" type="checkbox"/>	3
TMR2	TNI	T3 - Timer3	8	<input type="checkbox"/>	1
TMR1	TI	T1 - Timer1	3	<input checked="" type="checkbox"/>	4
OC2	OCI	OC2 - Output Compa...	6	<input checked="" type="checkbox"/>	7

Lab17 PWM Auto Duty Adj MCC's Setting & Code Example

```
volatile unsigned char T4Flag = 0;
unsigned int Duty = 50;
char DutyDistance = 2;
int main(void)
{
    ...
    while (1)
    {
        if (T4Flag)
        {
            T4Flag = 0;

            Duty += DutyDistance;

            if (Duty >= 100 || Duty <= 0)
                DutyDistance = -DutyDistance;

            if (Duty >= 100)
                OC1_PrimaryValueSet(OC1RS + 1);
            else
                OC1_PrimaryValueSet(((long) Duty * OC1RS) / 100);
        }
        ...
    }
}

void TMR4_CallBack(void)
{
    T4Flag = 1;
}
```