



MICROCHIP

Regional Training Centers

**Section 6
Timer**

Timer or Counter ?

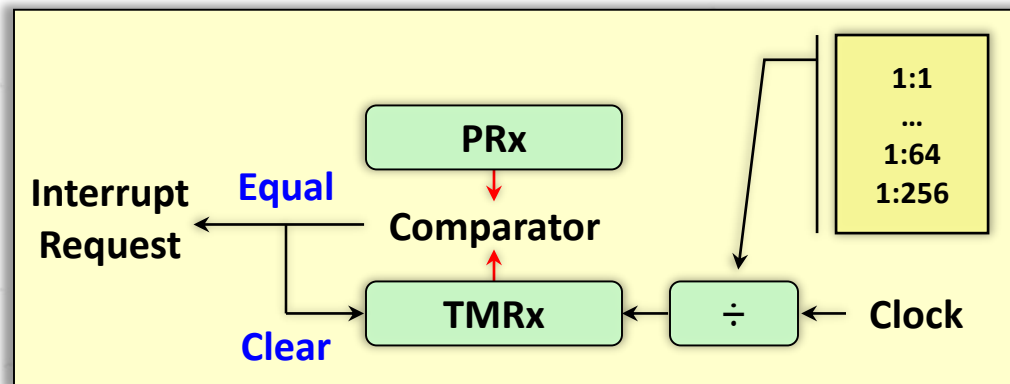
- ◆ **Counter or Timer is a clock counter, use to count how many clock into module after start.**
- ◆ **There are two kinds of generally called Timer or Counter. In fact, it's same.**
 - ◆ The Input Clock unknown : Just know how many count, called Counter.
 - ◆ The Input Clock known : Can use count and clock period to calculate time, called Timer.
- ◆ **Timer can set a notify condition, like overflow, equal some value etc.**
 - ◆ If interrupt disable : user must check flag, as soon as possible.
 - ◆ If interrupt enable : Timer will notify CPU, if condition is true.

PIC24FJGB Timer

- The PIC24FJGB provide **five** Timer module
- The Timer1 module is a **16 Bits** timer which can serve as the time counter for the **Real-Time Clock (RTC)**, or operate as a free-running, interval timer/counter. Timer1 can operate in **Timer**, **Counter** and **Gate Timer** mode.
- The Timer2/3 and Timer4/5 modules are **32 Bits** timers, which can also be configured as four independent. **16 Bits** timers with selectable operating modes. Timer2/3/4/5 can operate in **Timer**, **Counter** and **Gate Timer** mode.

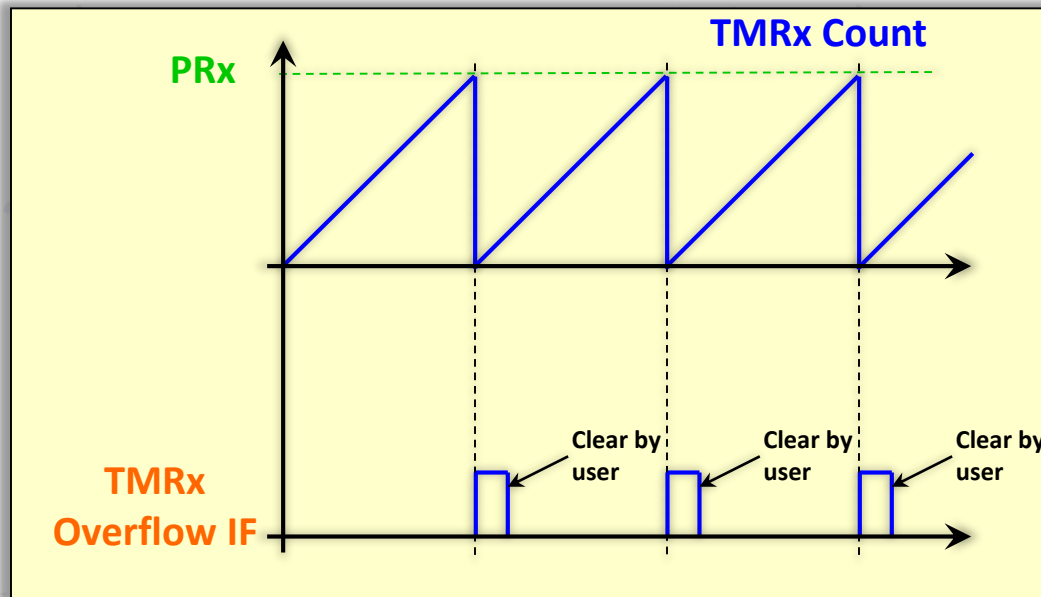
Timer/Counter Operation Mode

- Below block diagram, Just a concept.
- Clock into TMRx after prescaler. Comparator continuously compares the values of TMRx and PRx.
- Asserts an interrupt request while TMRx and PRx equal and then clear TMRx to zero.
- You can fill a value to PRx to determine period you want.



Timer/Counter Operation Mode

- ◆ TMRx increase by input clock source.
- ◆ TMRx clear automatically when counter TMRx = PRx
- ◆ set the TxIF interrupt flag.
- ◆ interrupt will be generated if enabled.



Prescaler

- Timer is 16 Bits counter , counting range from 0 to 65,535. If you need more than the count. You must adjust prescaler to expand the count range.
- For example:
if the clock in is 0.25uS, to reach a 500mS period, the PRx must be set to 2,000,000 (2,000,000 * 0.25uS = 500mS).
However, this value has exceeded the acceptable range of PRx.
At this point you can set the prescaler to reduce the count value to expand the count range.

$$(2^n - 1) \leq PRx = \frac{\text{Peroid}}{\text{Clock} \times \text{Scale}}$$

$$\times PRx = \frac{500mS}{0.25uS \times 1}, PRx > 65535$$

$$\checkmark PRx = \frac{500mS}{0.25uS \times 64}, PRx < 65535$$

Lab4 Timer1 Polling



Lab4 Timer1 Polling

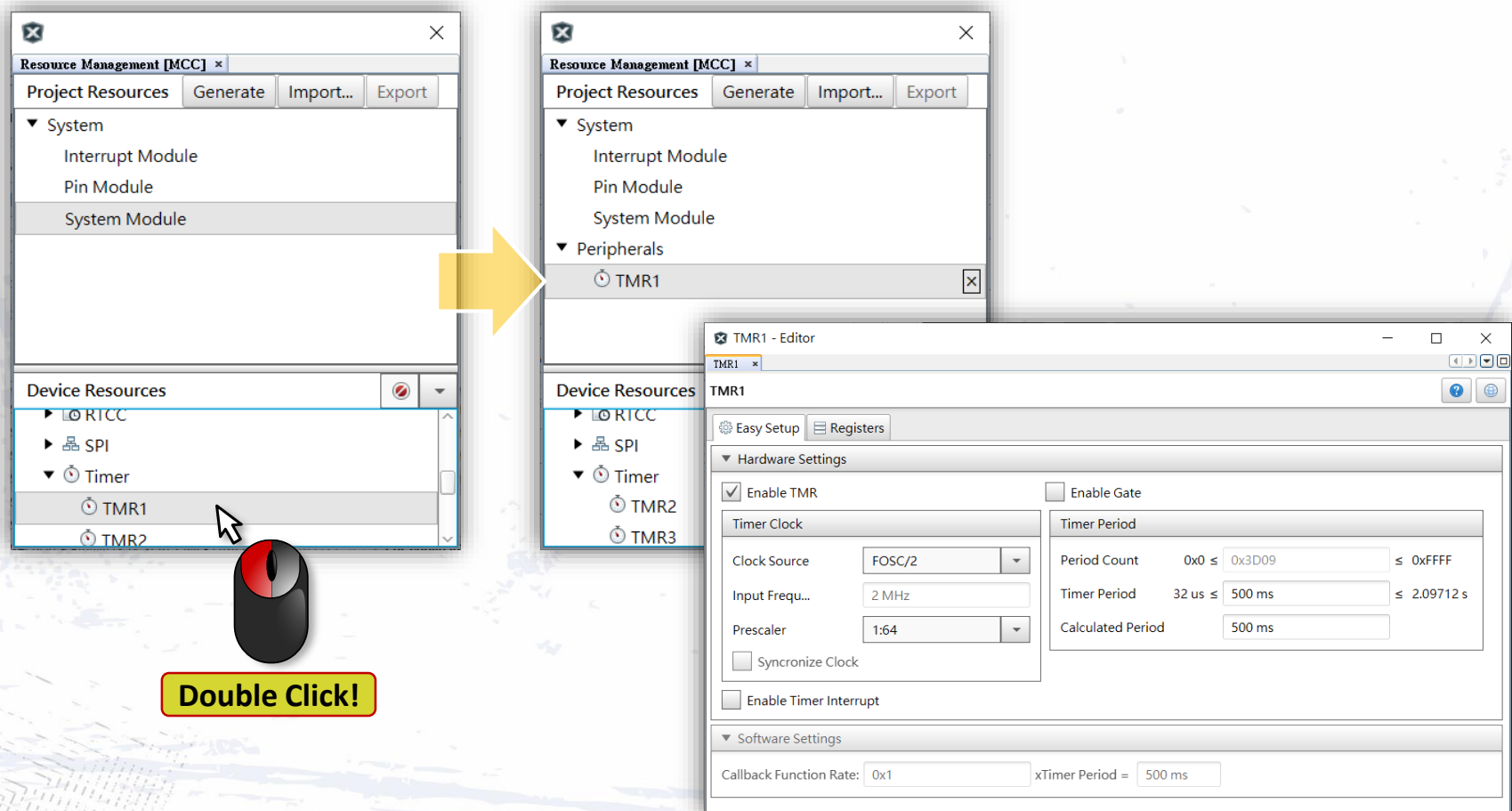
- ◆ Try to use Timer1 to replace software delay to control LEDs continues toggle.
- ◆ The Timer1 setup to timer mode, timer period is 500mS.

Lab4 Timer1 Polling Hints

- ❖ At this time we not to enable interrupt, so we need polling timer interrupt flag to measure timer period.
- ❖ **Interrupt Flag must clear manually.**
E.g. : `if(_T1IF == 1) { _T1IF = 0; ...; }`
- ❖ **MCC provide timer status maintain & check functions,**
 - ❖ `// Status maintain function, must execute at main loop or ISR.`
`void TMR1_Tasks_16BitOperation(void);`
 - ❖ `// Status Check functions.`
`bool TMR1_GetElapsedThenClear(void);`
`int TMR1_SoftwareCounterGet(void);`
`void TMR1_SoftwareCounterClear(void);`

Lab4 Timer1 Polling Hints

❖ Add Timer resource for Device Resources



Double Click!

TMR1 - Editor

Hardware Settings

- ☒ Enable TMR
- ☐ Enable Gate
- Timer Clock**
 - Clock Source: FOSC/2
 - Input Frequency: 2 MHz
 - Prescaler: 1:64
 - ☐ Synchronize Clock
- Timer Period**
 - Period Count: 0x0 ≤ 0x3D09 ≤ 0xFFFF
 - Timer Period: 32 us ≤ 500 ms ≤ 2.09712 s
 - Calculated Period: 500 ms
- ☐ Enable Timer Interrupt

Software Settings

Callback Function Rate: 0x1 xTimer Period = 500 ms

Lab4 Timer1 Polling Hints

The screenshot shows the 'TMR1 - Editor' window with the 'Easy Setup' tab selected. The 'Hardware Settings' section is expanded, showing the following configuration:

- Enable TMR:** ☒ (Callout: **Module Enable**)
- Enable Gate:** ☐
- Timer Clock:**
 - Clock Source:** FOSC/2 (Callout: **Clock Source**)
 - Input Frequ...:** 2 MHz
 - Prescaler:** 1:64 (Callout: **Prescaler**)
 - ☐ Synchronize Clock
- Enable Timer Interrupt:** ☐
- Timer Period:**
 - Period Count:** 0x0 ≤ 0x3D09 ≤ 0xFFFF
 - Timer Period:** 32 us ≤ 500 ms ≤ 2.09712 s (Callout: **Period (PRx)**)
 - Calculated Period:** 500 ms

The 'Software Settings' section is also visible, showing:

- Callback Function Rate:** 0x1
- xTimer Period =** 500 ms

Lab4 Timer1 Polling Hints

Timer Polling Example (Base on MCC Function)

```
● while (1)
{
    // Check Timer 1 Status.
    if (TMR1_GetElapsedThenClear( ) )
    {
        // LEDs Toggle;
    }

    // maintain function, must execute at main loop.
    TMR1_Tasks_16BitOperation( );
}
```

Lab4 Timer1 Polling



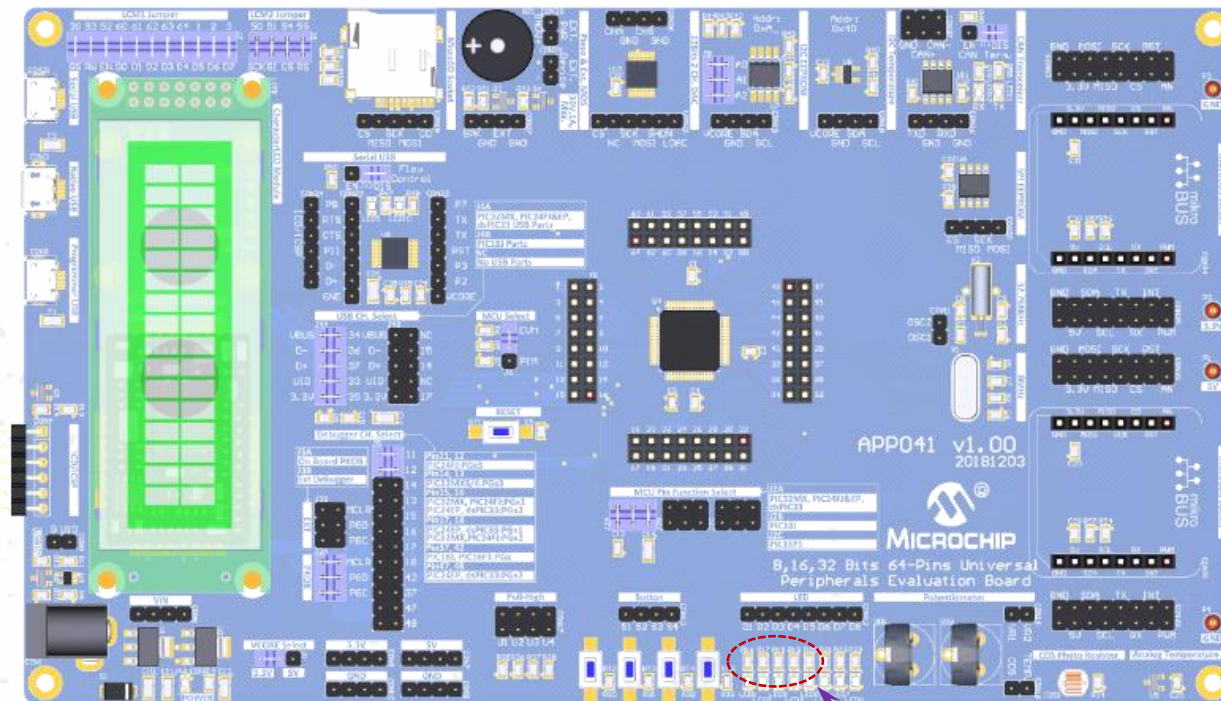
Lab4 Timer1 Polling

- Try to initial Timr1 to control LEDs continues toggle.
- Timer1 setup to timer mode, timer period is 500mS.

■ **Let's go!**

Lab4 Timer1 Polling

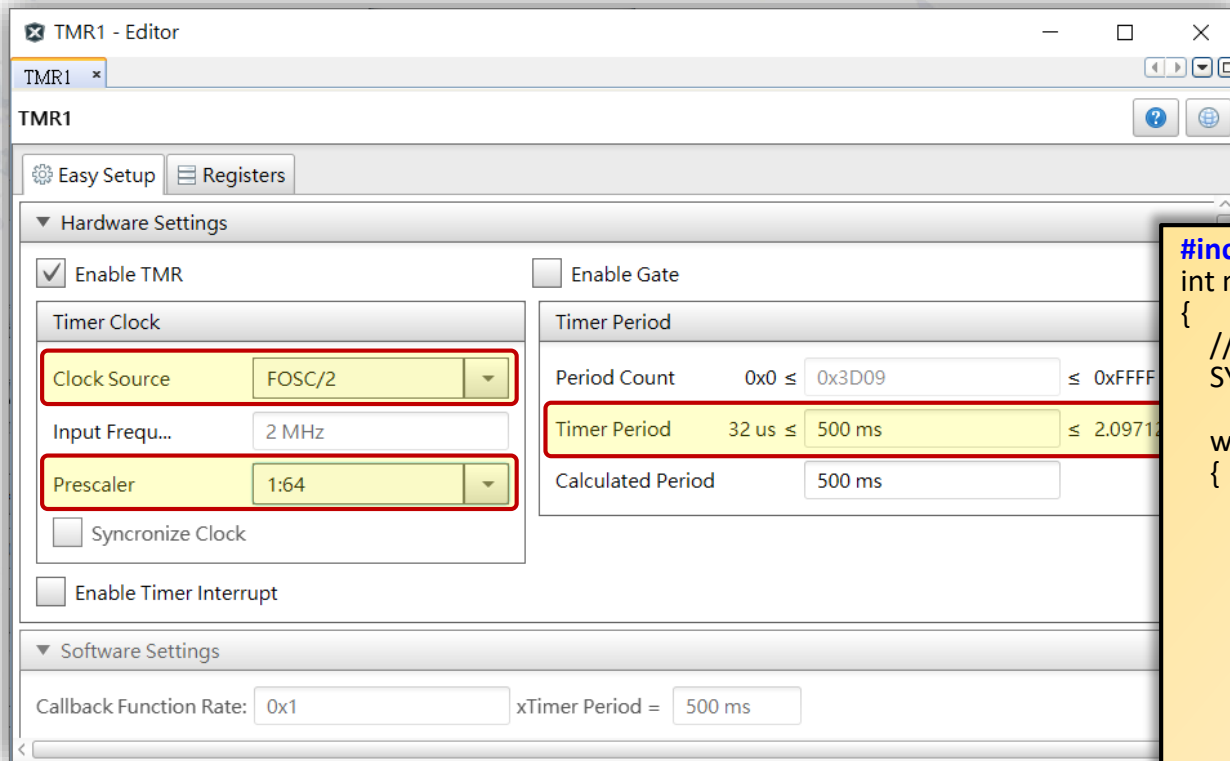
Result



**LEDs Control by TMR1 Polling,
Toggle every 500ms.**

Lab4 Timer1 Polling

MCC's Setting & Code Example



```
#include "mcc_generated_files/tmr1..h"
int main(void)
{
    // initialize the device
    SYSTEM_Initialize();

    while (1)
    {
        // Add your application code
        if(TMR1_GetElapsedThenClear( ))
        {
            ...
        }

        if(S1_GetValue())
            D8_SetLow();
        else
            D8_SetHigh();

        TMR1_Tasks_16BitOperation( );
    }
    return -1;
}
```

Lab5 Multi-Timer Polling



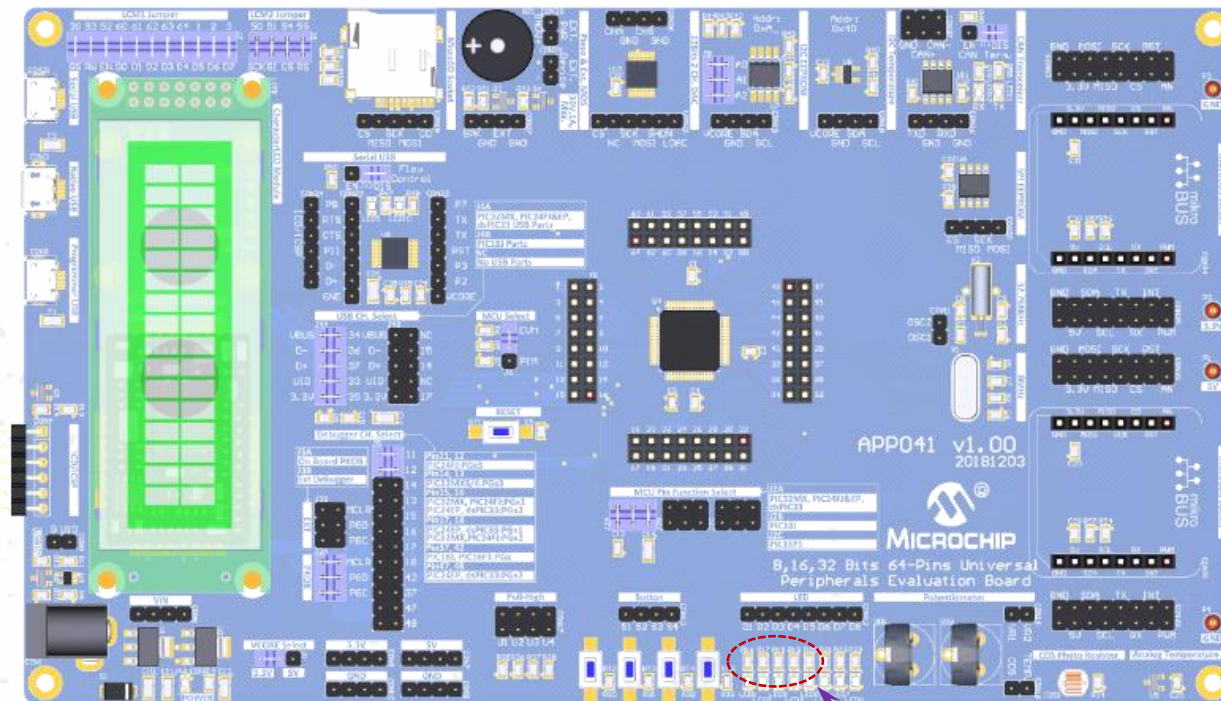
Lab5 Multi-Timer Polling

- Try to add Timer2 to control LEDs continues toggle.
- The Timer2 setup to timer mode, timer period is 1S.
- Timer1 control D1, D2, toggle period -> 500mS
Timer2 control D3, D4, toggle period -> 1S

Let's go!

Lab5 Multi-Timer Polling

Result



D1,D2 Control by TMR1 Polling,
Toggle every 500ms.
D3,D4 Control by TMR2 Polling,
Toggle every 1s.

Lab5 Multi-Timer Polling

MCC's Setting & Code Example

The screenshot shows the 'TMR2 - Editor' window. The 'Easy Setup' tab is active. Under 'Hardware Settings', 'Enable TMR' is checked. The 'Timer Clock' section shows 'Clock Source' set to 'FOSC/2' and 'Prescaler' set to '1:64'. The 'Timer Period' section shows 'Period Count' as '0x0 ≤ 0x7A12 ≤ 0xFFFF' and 'Timer Period' as '32 us ≤ 1 s ≤ 2.09712 s'. The 'Calculated Period' is '1 s'. Under 'Software Settings', 'Callback Function Rate' is '0x1' and 'xTimer Period' is '1000 ms'.

```
#include "mcc_generated_files/tmr1.h"
#include "mcc_generated_files/tmr2.h"
int main(void)
{
    // initialize the device
    SYSTEM_Initialize();

    while (1)
    {
        // Add your application code
        if(TMR1_GetElapsedThenClear( ))
        {
            ...
        }

        if(TMR2_GetElapsedThenClear( ))
        {
            ...
        }

        if(S1_GetValue())
            D8_SetLow();
        else
            D8_SetHigh();

        TMR1_Tasks_16BitOperation( );
        TMR2_Tasks_16BitOperation( );
    }
    return -1;
}
```