MICROCHIP

Regional Training Centers

Section 8
Interrupt

What's Interrupt ?

- In general, processor are executed in sequence according to the program's flow.
- However, if some peripherals or event needs immediate attention then an interrupt flag been set as high-priority and requiring to interrupt current process to serve it.
- Processor will jump to execute the interrupt service routine (ISR, Interrupt Service Routine) to service peripherals or events.

Which events need Interrupt ?

- **For General (maskable)**
  - **Time critical event**

    As soon as possible, when event occurred.
  - **Unpredicted event**

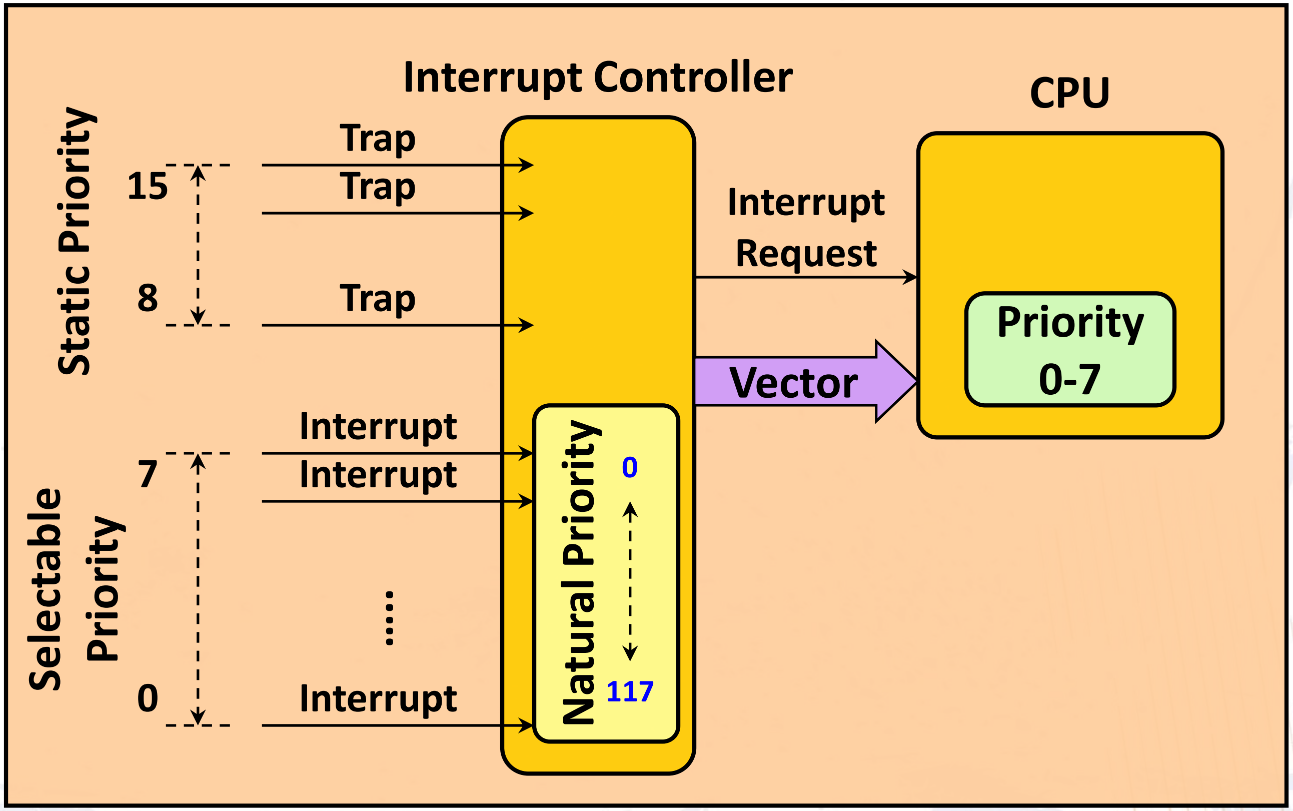
    Polling too waste computing time.

- **For Emergency (non maskable)**
  - Stack Overflow/Underflow
  - Math Error
  - Address Error
  - etc..

16-Bits Interrupt Architecture

- 16-Bits's Interrupt is Nested Vector Interrupt Controller.
- 8 processor exceptions and software traps
- Provide 7 (0~7) user-selectable priority levels.

  (7 is Highest priority)
- up to 118 vectors.
- Fixed interrupt entry and return latencies.



Interrupt Controller

CPU

Static Priority

15

8

Trap

Trap

Trap

Interrupt Request

Vector

Priority 0-7

Selectable Priority

7

0

Interrupt

Interrupt

⋮

Interrupt

Natural Priority

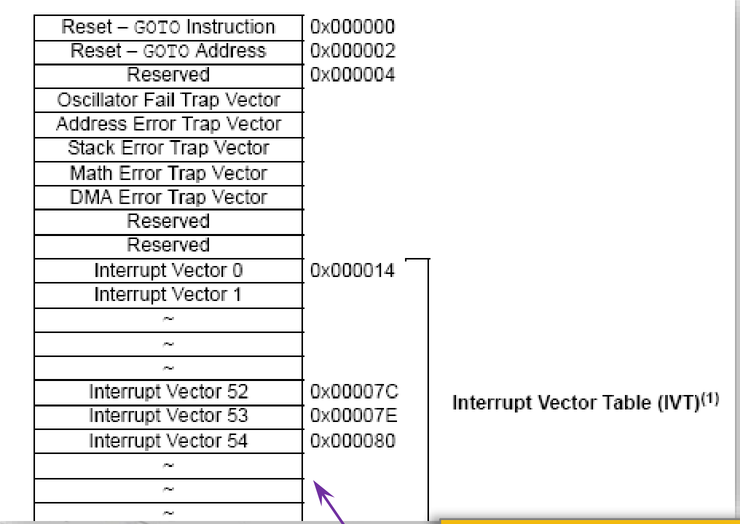
0

117

Natural Priority

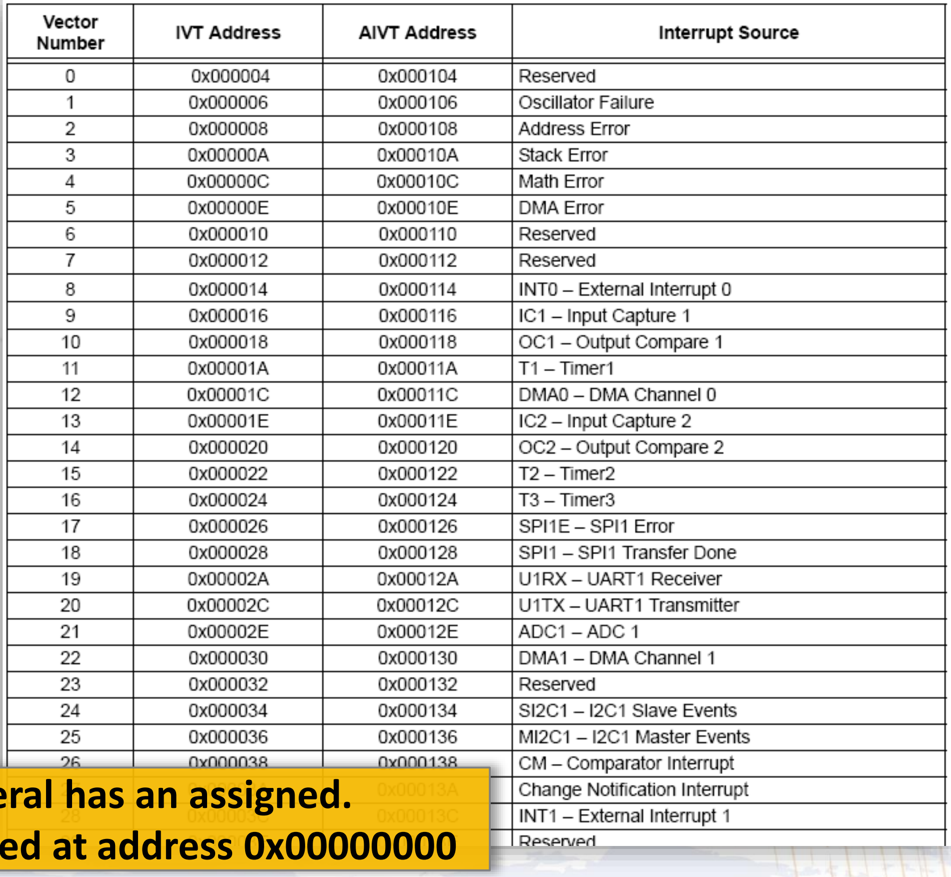
- **If two pending interrupts share the same priority, priority is given to the interrupt with the lowest exception number**

  (lowest interrupt vector address).



Reset – GOTO Instruction	0x000000
Reset – GOTO Address	0x000002
Reserved	0x000004
Oscillator Fail Trap Vector	
Address Error Trap Vector	
Stack Error Trap Vector	
Math Error Trap Vector	
DMA Error Trap Vector	
Reserved	
Reserved	
Interrupt Vector 0	0x000014
Interrupt Vector 1	
~	
~	
~	
Interrupt Vector 52	0x00007C
Interrupt Vector 53	0x00007E
Interrupt Vector 54	0x000080
~	
~	
~	

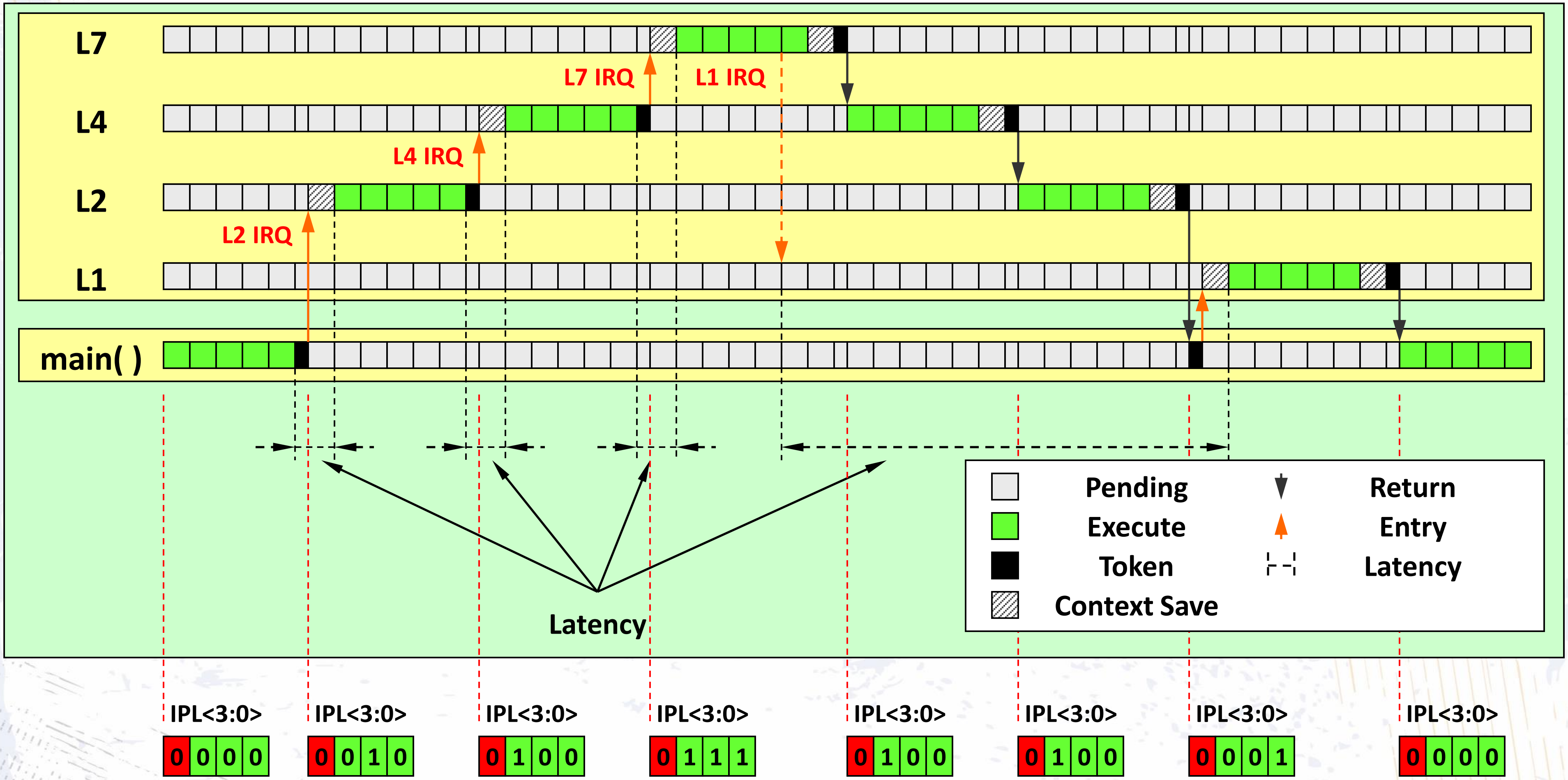
Interrupt Vector Table (IVT)[1]



Vector Number	IVT Address	AIVT Address	Interrupt Source
0	0x000004	0x000104	Reserved
1	0x000006	0x000106	Oscillator Failure
2	0x000008	0x000108	Address Error
3	0x00000A	0x00010A	Stack Error
4	0x00000C	0x00010C	Math Error
5	0x00000E	0x00010E	DMA Error
6	0x000010	0x000110	Reserved
7	0x000012	0x000112	Reserved
8	0x000014	0x000114	INT0 – External Interrupt 0
9	0x000016	0x000116	IC1 – Input Capture 1
10	0x000018	0x000118	OC1 – Output Compare 1
11	0x00001A	0x00011A	T1 – Timer1
12	0x00001C	0x00011C	DMA0 – DMA Channel 0
13	0x00001E	0x00011E	IC2 – Input Capture 2
14	0x000020	0x000120	OC2 – Output Compare 2
15	0x000022	0x000122	T2 – Timer2
16	0x000024	0x000124	T3 – Timer3
17	0x000026	0x000126	SPI1E – SPI1 Error
18	0x000028	0x000128	SPI1 – SPI1 Transfer Done
19	0x00002A	0x00012A	U1RX – UART1 Receiver
20	0x00002C	0x00012C	U1TX – UART1 Transmitter
21	0x00002E	0x00012E	ADC1 – ADC 1
22	0x000030	0x000130	DMA1 – DMA Channel 1
23	0x000032	0x000132	Reserved
24	0x000034	0x000134	SI2C1 – I2C1 Slave Events
25	0x000036	0x000136	MI2C1 – I2C1 Master Events
26	0x000038	0x000138	CM – Comparator Interrupt
[illegible]	[illegible]	0x00013A	Change Notification Interrupt
28	0x00003C	0x00013C	INT1 – External Interrupt 1
[illegible]	[illegible]	[illegible]	Reserved

Each peripheral has an assigned.
Vector table is based at address 0x00000000

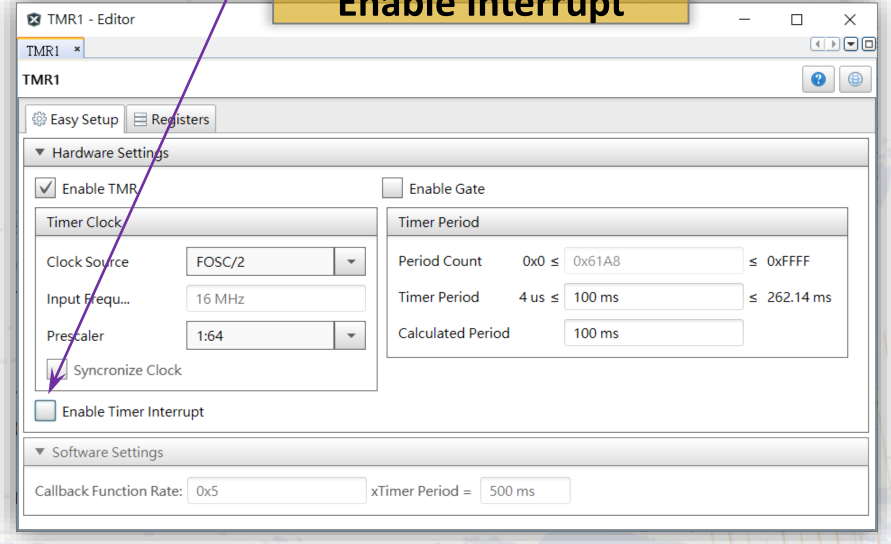
Flowchart of Interrupt



Coding for Interrupt

- **A complete interrupt code segment include below 3 parts.**
  - Interrupt Service Routine (ISR)
  - Enable Interrupt and Set Interrupt Priority
  - Initial Peripheral to generate Interrupt Request.
- **This is very difficult for rookie, if you use bare metal style to build your code.**
- **MCC can help you to build most all code segment. So you can easy to handle interrupt even you are beginner.**
- **Just one click all interrupt relate get ready.**

Just One Click !
Enable Interrupt



TMR1 - Editor

TMR1

Easy Setup | Registers

Hardware Settings

☑ Enable TMR
☐ Enable Gate

Timer Clock
Clock Source: FOSC/2
Input Frequ...: 16 MHz
Prescaler: 1:64
☐ Syncronize Clock

Timer Period
Period Count: 0x0 ≤ 0x61A8 ≤ 0xFFFF
Timer Period: 4 us ≤ 100 ms ≤ 262.14 ms
Calculated Period: 100 ms

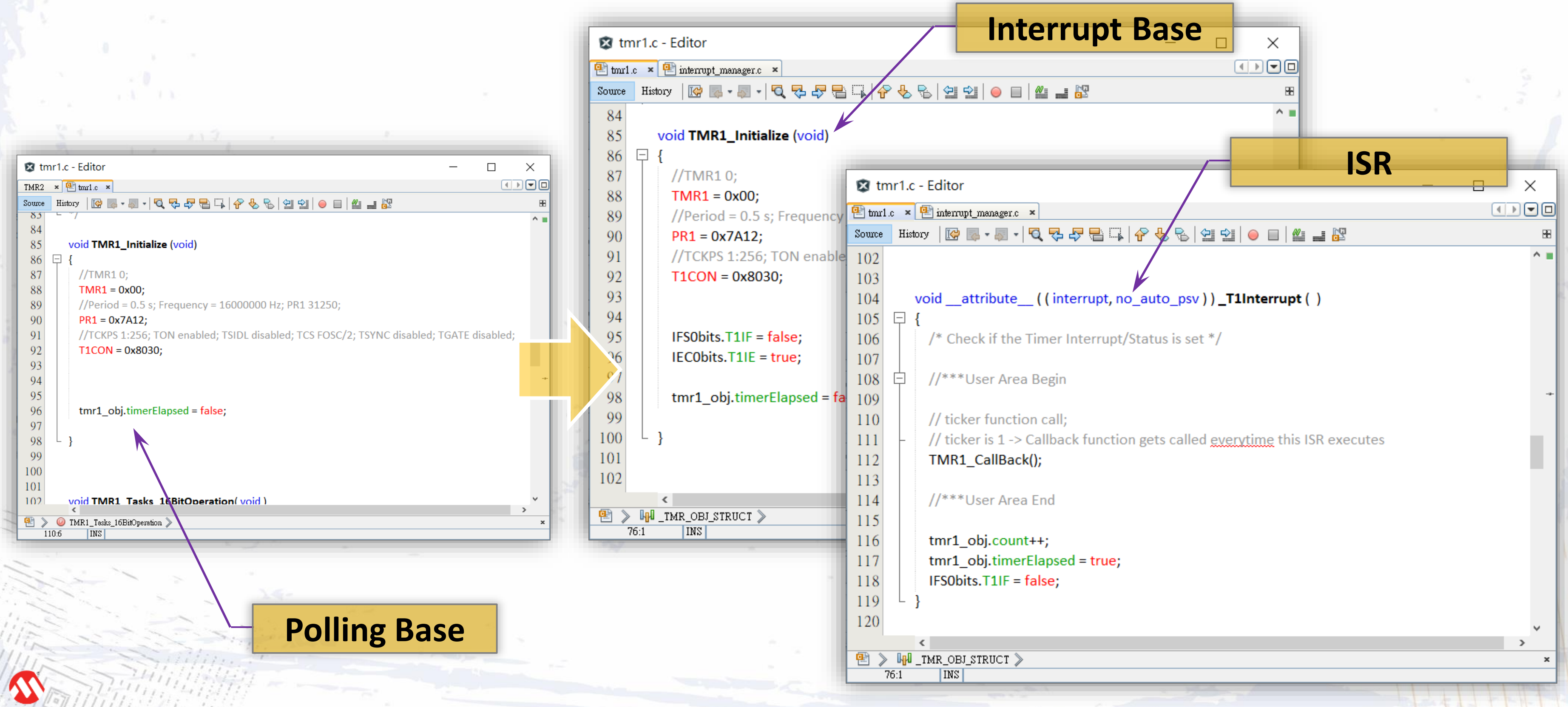
☐ Enable Timer Interrupt

Software Settings

Callback Function Rate: 0x5 xTimer Period = 500 ms

MCC’s Interrupt Function

- **MCC enable interrupt, set priority and create interrupt service routine automatically.**



Polling Base

```
void TMR1_Initialize (void)
{
    //TMR1 0;
    TMR1 = 0x00;
    //Period = 0.5 s; Frequency = 16000000 Hz; PR1 31250;
    PR1 = 0x7A12;
    //TCKPS 1:256; TON enabled; TSIDL disabled; TCS FOSC/2; TSYNC disabled; TGATE disabled;
    T1CON = 0x8030;



    tmr1_obj.timerElapsed = false;

}



void TMR1_Tasks_16BitOperation( void )
```

Interrupt Base

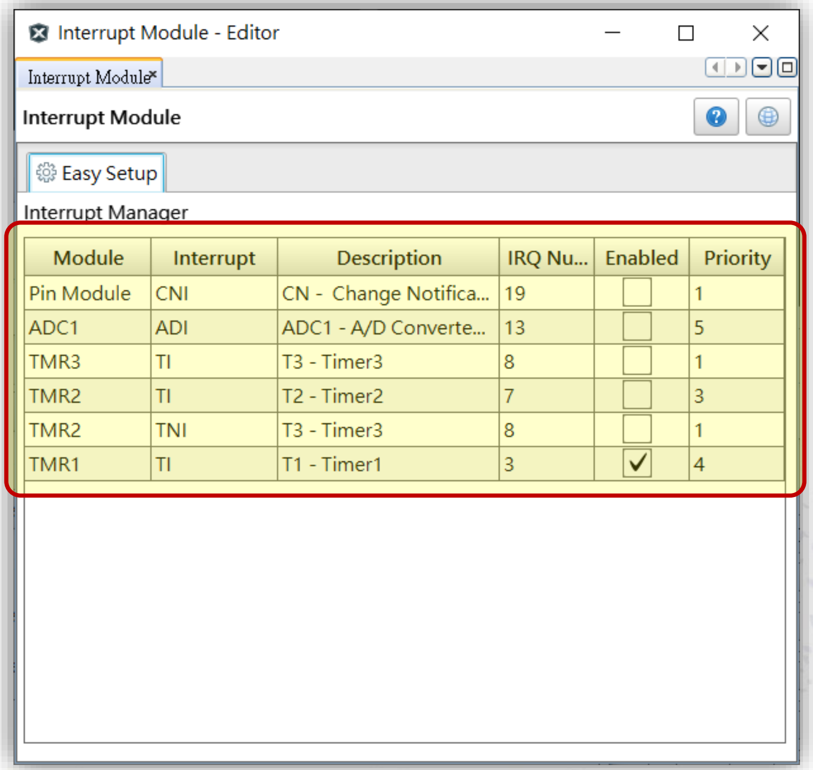
```
void TMR1_Initialize (void)
{
    //TMR1 0;
    TMR1 = 0x00;
    //Period = 0.5 s; Frequency
    PR1 = 0x7A12;
    //TCKPS 1:256; TON enable
    T1CON = 0x8030;


    IFS0bits.T1IF = false;
    IEC0bits.T1IE = true;

    tmr1_obj.timerElapsed = fa
}
```

ISR

```
void __attribute__ ( ( interrupt, no_auto_psv ) ) _T1Interrupt ( )
{
    /* Check if the Timer Interrupt/Status is set */

    //***User Area Begin

    // ticker function call;
    // ticker is 1 -> Callback function gets called everytime this ISR executes
    TMR1_CallBack();

    //***User Area End

    tmr1_obj.count++;
    tmr1_obj.timerElapsed = true;
    IFS0bits.T1IF = false;
}
```

MCC’s Interrupt Manager

- **MCC’s Interrupt Module allow user to manager interrupt enable/disable and priority.**



Interrupt Manager

Module	Interrupt	Description	IRQ Nu...	Enabled	Priority
Pin Module	CNI	CN - Change Notifica...	19	☐	1
ADC1	ADI	ADC1 - A/D Converte...	13	☐	5
TMR3	TI	T3 - Timer3	8	☐	1
TMR2	TI	T2 - Timer2	7	☐	3
TMR2	TNI	T3 - Timer3	8	☐	1
TMR1	TI	T1 - Timer1	3	☑	4

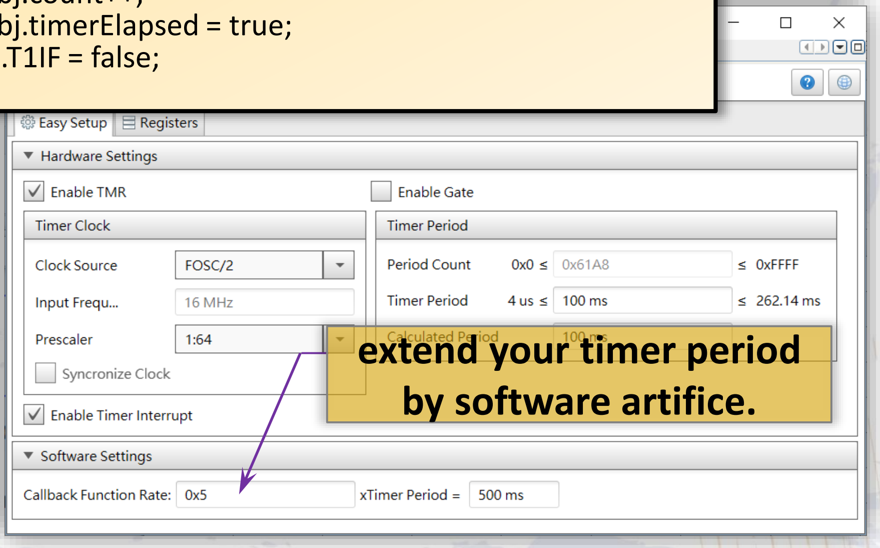
Callback Function

- Most interrupt handler function provide, named “callback” function for user.
- Flag and event handle by handler function automatically, user focus at application only. It’s more powerful and easy to understand than operating a bare metal style.
- The callback function own named “weak” qualification. This is mean user can redefine your same name function in your code to replace the original function.
- For Example,
  you can redefine timer callback function in your code. And put LED toggle code to your callback function.
  your callback function executing automatically, when Timer interrupt occurred.

Timer’s Callback Function Rate

- Timer’s callback function rate is a software counter.
- Hardware always Limited. Its can’t set a long period, E.g. 10 second.
- The “Rate” can be use to extend your timer period by software artifice.

```
void __attribute__((interrupt, no_auto_psv)) _T1Interrupt()
{
   static volatile unsigned int CountCallBack = 0;

   if (++CountCallBack >= TMR1_INTERRUPT_TICKER_FACTOR)
   {
      TMR1_CallBack();
      CountCallBack = 0;
   }
   tmr1_obj.count++;
   tmr1_obj.timerElapsed = true;
   IFS0bits.T1IF = false;
}
```



☑ Enable TMR
☐ Enable Gate

Clock Source: FOSC/2
Input Frequ...: 16 MHz
Prescaler: 1:64
☐ Syncronize Clock

Period Count: 0x0 ≤ 0x61A8 ≤ 0xFFFF
Timer Period: 4 us ≤ 100 ms ≤ 262.14 ms
Calculated Period: 100 ms

☑ Enable Timer Interrupt

Callback Function Rate: 0x5
xTimer Period = 500 ms

extend your timer period by software artifice.

Lab7 Timer1 Interrupt

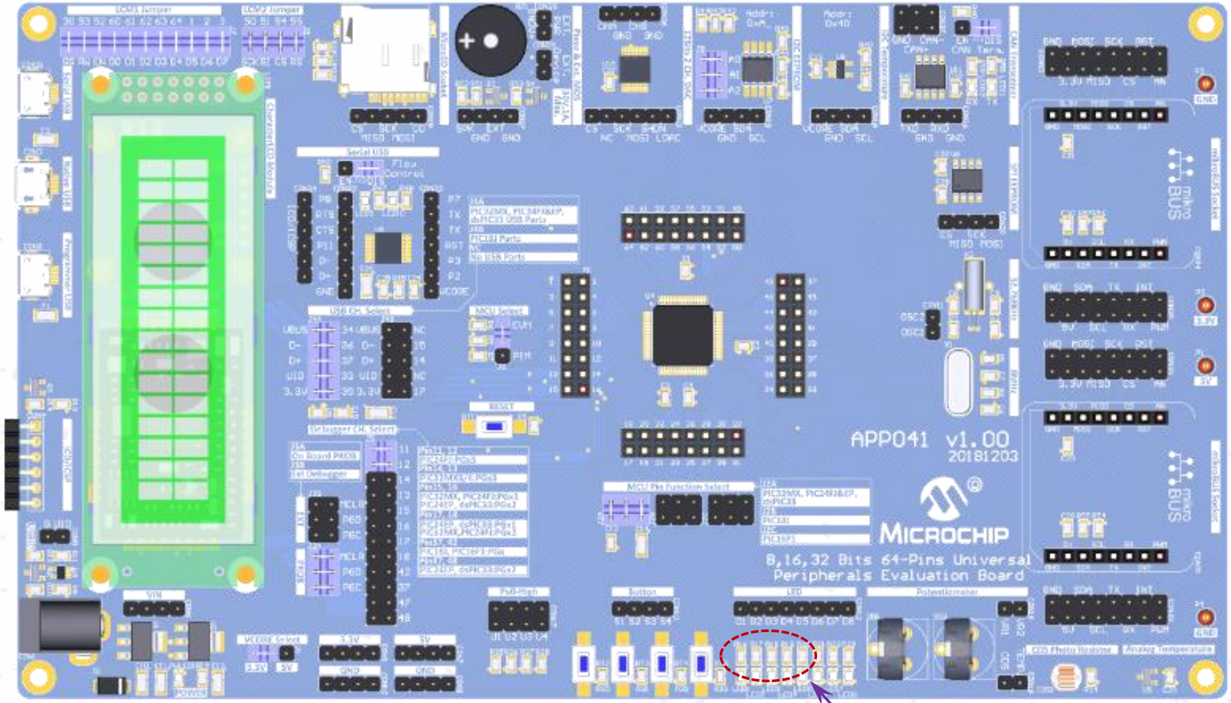
Lab7 Timer1 Interrupt

- **Try to modify code style to from polling to interrupt style.**
- **Please change Timer1, D1 and D2 toggle function to callback style.**
- **Timr1 interrupt priority set to level 4**
- **Enable timer callback rate, set period to 500mS(100mS*5)**

- **Let's go!**

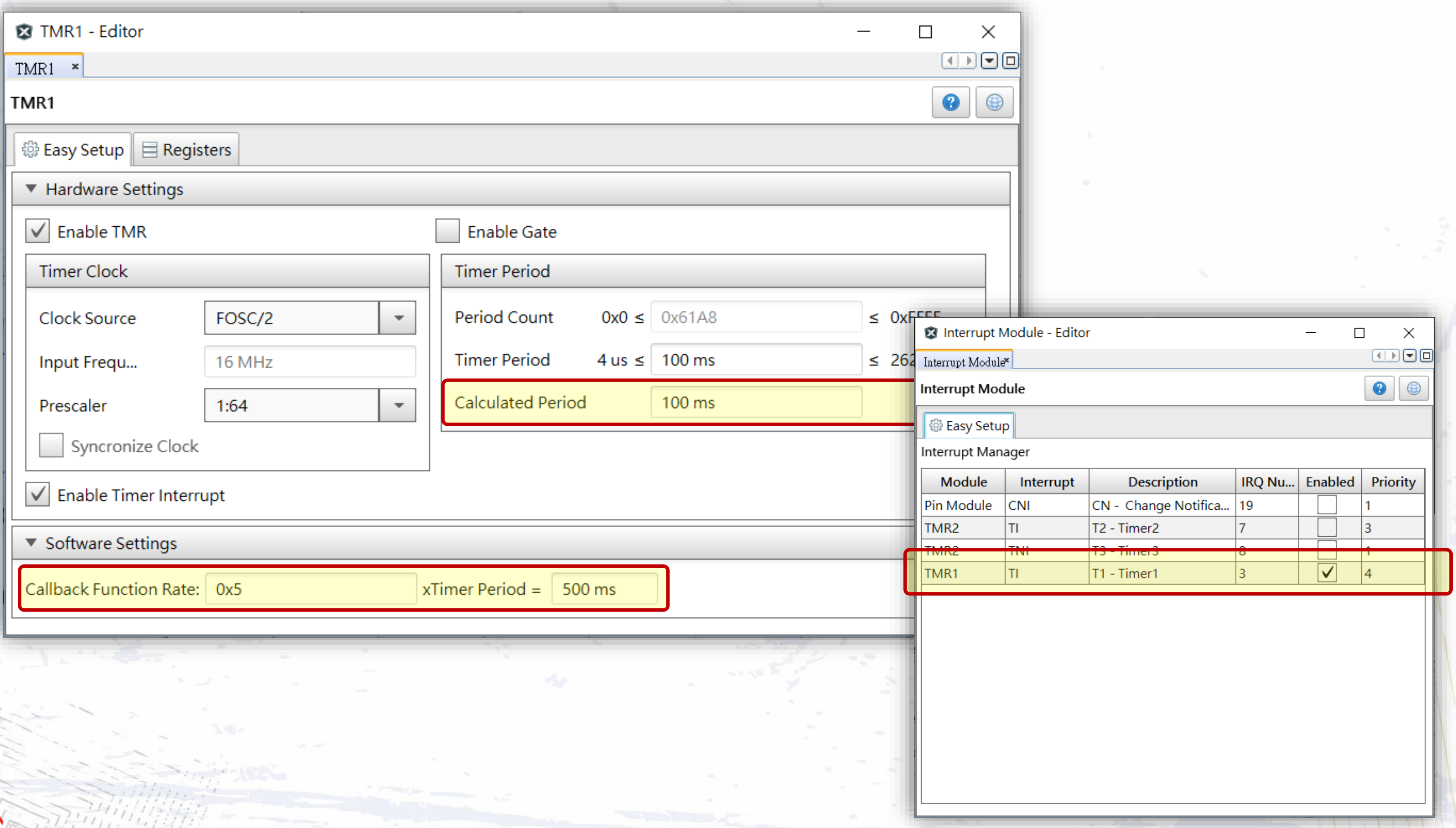
Lab7 Timer1 Interrupt

Result



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

Lab8 Multi-Timer Interrupt MCC's Setting & Code Example



TMR1 - Editor

TMR1

Easy Setup | Registers

Hardware Settings

☑ Enable TMR

☐ Enable Gate

Timer Clock

- Clock Source: FOSC/2
- Input Frequ...: 16 MHz
- Prescaler: 1:64
- ☐ Syncronize Clock

Timer Period

- Period Count: 0x0 ≤ 0x61A8 ≤ 0xFFFF
- Timer Period: 4 us ≤ 100 ms ≤ 262
- Calculated Period: 100 ms

☑ Enable Timer Interrupt

Software Settings

Callback Function Rate: 0x5 xTimer Period = 500 ms

Interrupt Module - Editor

Interrupt Module

Easy Setup

Interrupt Manager

Module	Interrupt	Description	IRQ Nu...	Enabled	Priority
Pin Module	CNI	CN - Change Notifica...	19	☐	1
TMR2	TI	T2 - Timer2	7	☐	3
TMR3	TNI	T3 - Timer3	8	☐	1
TMR1	TI	T1 - Timer1	3	☑	4

Lab7 Timer1 Interrupt

MCC's Setting & Code Example

```
int main(void)
{
    // initialize the device
    SYSTEM_Initialize();

    while (1)
    {
        // Add your application code
        if(TMR2_GetElapsedThenClear( ))
        {
            …
        }

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

        TMR2_Tasks_16BitOperation( );
    }
    return -1;
}

void TMR1_CallBack(void)
{
    D1_Toggle();
    D2_Toggle();
}
```

Lab8 Multi-Timer Interrupt

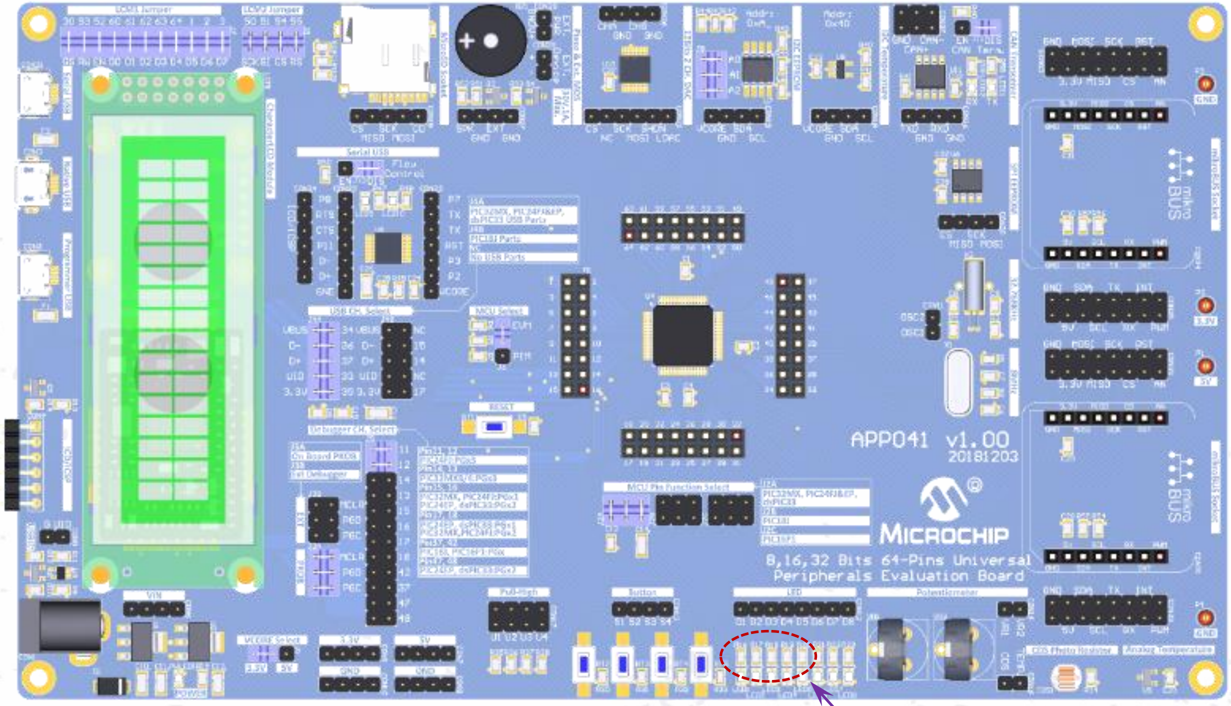
Lab8 Multi-Timer Interrupt

- Try to change Timer2, D3 and D4 toggle function to callback style, also.
- Timr2 interrupt priority set to level 3
- Enable timer callback rate, set period to 1S(500mS*2)

- **Let’s go!**

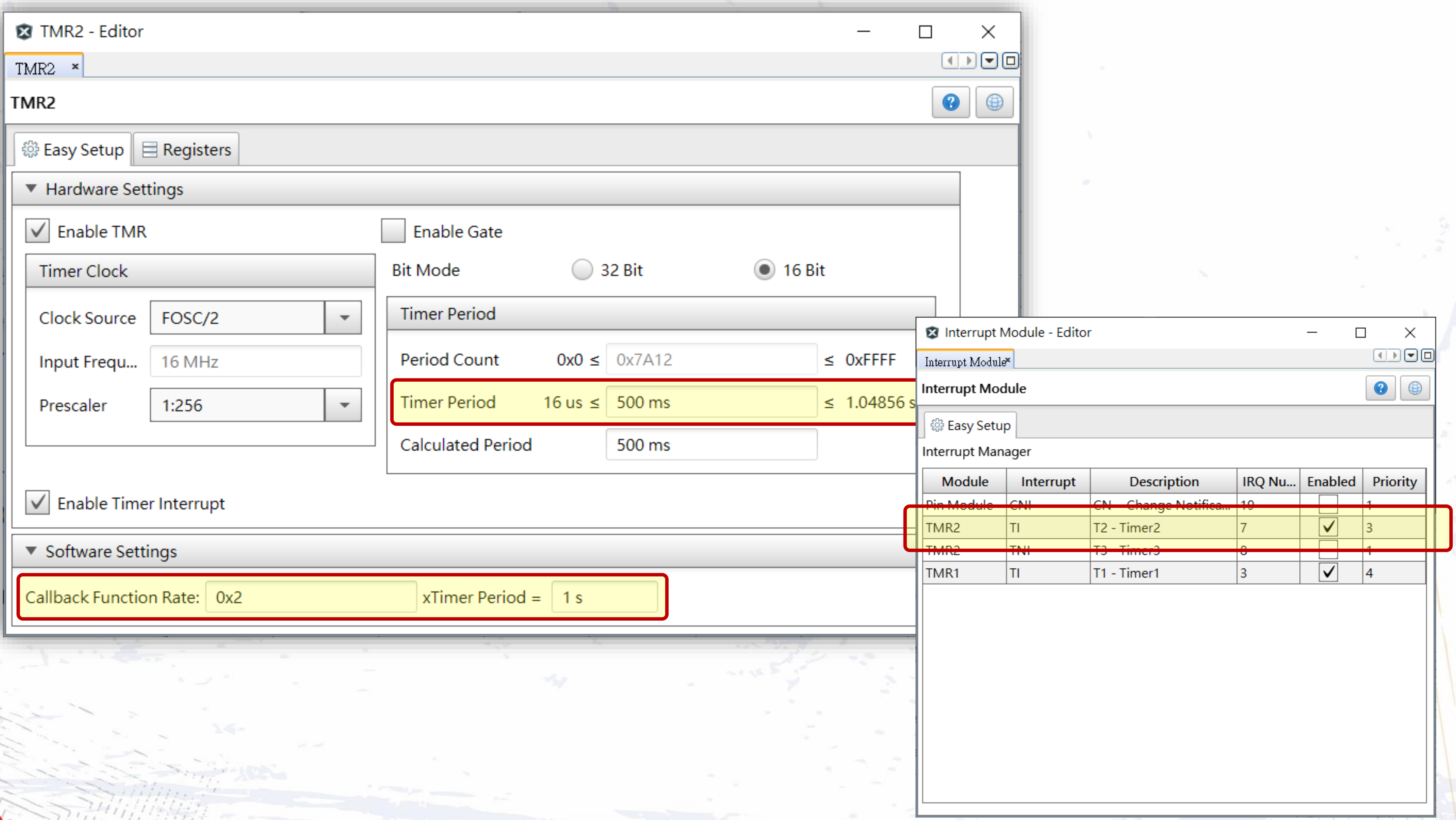
Lab8 Multi-Timer Interrupt

Result



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

Lab8 Multi-Timer Interrupt
MCC's Setting & Code Example



TMR2

Hardware Settings

☑ Enable TMR

☐ Enable Gate

Timer Clock

Clock Source: FOSC/2

Input Frequ...: 16 MHz

Prescaler: 1:256

Bit Mode: ○ 32 Bit ● 16 Bit

Timer Period

Period Count: 0x0 ≤ 0x7A12 ≤ 0xFFFF

Timer Period: 16 us ≤ 500 ms ≤ 1.04856 s

Calculated Period: 500 ms

☑ Enable Timer Interrupt

Software Settings

Callback Function Rate: 0x2 xTimer Period = 1 s

Interrupt Module

Interrupt Manager

Module	Interrupt	Description	IRQ Nu...	Enabled	Priority
Pin Module	CNI	CN - Change Notifica...	10	☐	1
TMR2	TI	T2 - Timer2	7	☑	3
TMR3	TNI	T3 - Timer3	8	☐	1
TMR1	TI	T1 - Timer1	3	☑	4

Lab8 Multi-Timer Interrupt
MCC's Setting & Code Example

```
int main(void)
{
   // initialize the device
   SYSTEM_Initialize();

   while (1)
   {
      // Add your application code
      if(S1_GetValue())
         D8_SetLow();
      else
         D8_SetHigh();
   }
   return -1;
}

void TMR1_CallBack(void)
{
   D1_Toggle();
   D2_Toggle();
}

void TMR2_CallBack(void)
{
   D3_Toggle();
   D4_Toggle();
}
```