MICROCHIP

Regional Training Centers

Section 7
Nested Vector Interrupt Controller Architecture

What's Interrupt ?

- In system programming, an interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention.
- An interrupt alerts the processor to a high-priority condition requiring the interruption of the current code the processor is executing.
- The processor responds by suspending its current activities, saving its state, and executing a function called an interrupt handler (or an Interrupt Service Routine, ISR) to deal with the event.

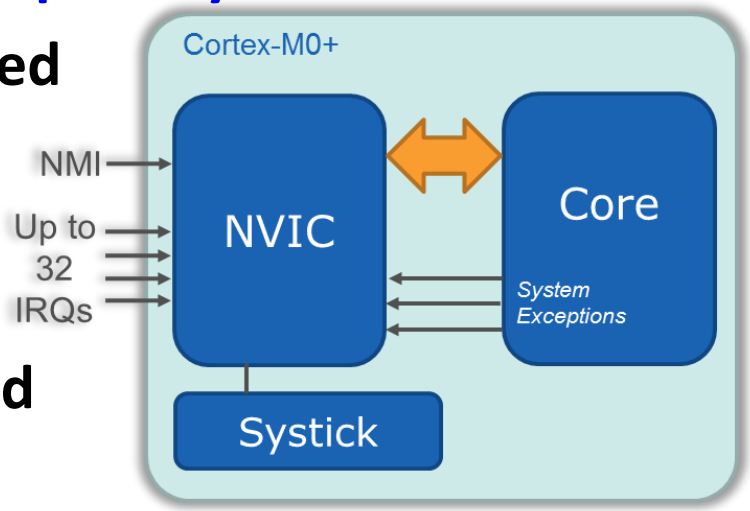
(wiki reference)

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..

Nested Vector Interrupt Controller

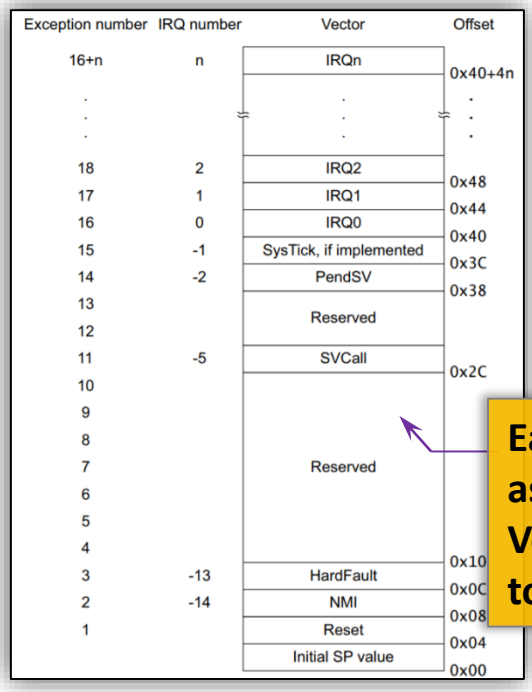
- NVIC provides an interface between interrupt sources (peripherals and external pins) and the core.
- The priority (Level 0 ~ 3) for each interrupt source is programmable, Level 0 is highest priority.
- Each interrupt lines has connected to one peripheral instance and have one or more interrupt flags.



- An interrupt request is generated from the peripheral when the interrupt flag has been set and the corresponding interrupt is enabled.

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).**



Each peripheral has been assigned to one of Vector table and based to address 0x00000000

TC3 (18)

TC4 (19)

Peripheral Source	NVIC Line
EIC NMI – External Interrupt Controller	NMI
PM – Power Manager	0
SYSCTRL – System Control	1
WDT – Watchdog Timer	2
RTC – Real Time Counter	3
EIC – External Interrupt Controller	4
NVMCTRL – Non-Volatile Memory Controller	5
DMAC - Direct Memory Access Controller	6
USB - Universal Serial Bus	7
EVSYS – Event System	8
SERCOM0 – Serial Communication Interface 0	9
SERCOM1 – Serial Communication Interface 1	10
SERCOM2 – Serial Communication Interface 2	11
SERCOM3 – Serial Communication Interface 3	12
SERCOM4 – Serial Communication Interface 4	13
SERCOM5 – Serial Communication Interface 5	14
TCC0 – Timer Counter for Control 0	15
TCC1 – Timer Counter for Control 1	16
TCC2 – Timer Counter for Control 2	17
TC3 – Timer Counter 3	18
TC4 – Timer Counter 4	19
TC5 – Timer Counter 5	20
TC6 – Timer Counter 6	21
TC7 – Timer Counter 7	22
ADC – Analog-to-Digital Converter	23
AC – Analog Comparator	24
DAC – Digital-to-Analog Converter	25
PTC – Peripheral Touch Controller	26
I2S - Inter IC Sound	27

SAMD21G18A IVT Define

interrupt.c

```
__attribute__ ((section(".vectors")))
const DeviceVectors exception_table=
{
    /* Configure Initial Stack Pointer, using linker-generated symbols */
    .pvStack = (void*) (&_stack),

    .pfnReset_Handler              = ( void * ) Reset_Handler,
    .pfnNonMaskableInt_Handler     = ( void * ) NonMaskableInt_Handler,
    .pfnHardFault_Handler          = ( void * ) HardFault_Handler,
    .pfnSVCall_Handler             = ( void * ) SVCall_Handler,
    .pfnPendSV_Handler             = ( void * ) PendSV_Handler,
    .pfnSysTick_Handler            = ( void * ) SysTick_Handler,
    .pfnNVMCTRL_Handler            = ( void * ) NVMCTRL_Handler,
     …
    .pfnSERCOM5_Handler            = ( void * ) SERCOM0_Handler,
     …
    .pfnTCC2_Handler               = ( void * ) TCC2_Handler,
    .pfnTC3_Handler                = ( void * ) TC3_CompareInterruptHandler,
    .pfnTC4_Handler                = ( void * ) TC4_CompareInterruptHandler,
    …
    .pfnADC_Handler                = ( void * ) ADC_Handler,
    …
};
```

ISR Function Example

- Implement below ISR (Interrupt Service Routine) in TC function.

plib_tc3.c

```
#include "plib_tc3.h"
…

void TC3_CompareInterruptHandler( void )
{
    TC_COMPARE_STATUS status;

    status = TC3_REGS->COUNT16.TC_INTFLAG;

  /* clear period interrupt */
    TC3_REGS->COUNT16.TC_INTFLAG = TC_INTFLAG_OVF_Msk;

    // Getting starter add your code here ?????
}
```

Now ? Hands-on Lab ?

No!

Callback Function

- MH3 module provide the interrupt mode driver.
- All flag and event handling by MH3 interrupt interface routine automatically, user just focus on application. It's more powerful and easy to understand than operating a NVIC ISR function directly.
- The interrupt mode provide "callback" function to hook up your customer function to specific interrupt source and event.
- For Example,
  You can hook the LED toggle callback function in TC overflow event. Callback function will be executed automatically while TC overflow event has occurred.

TC Callback Hook Example

```
TCn_CompareCallbackRegister( TCn_Overflow, (uintptr_t )NULL );
TCn_CompareStart();
```

Hook (Register)

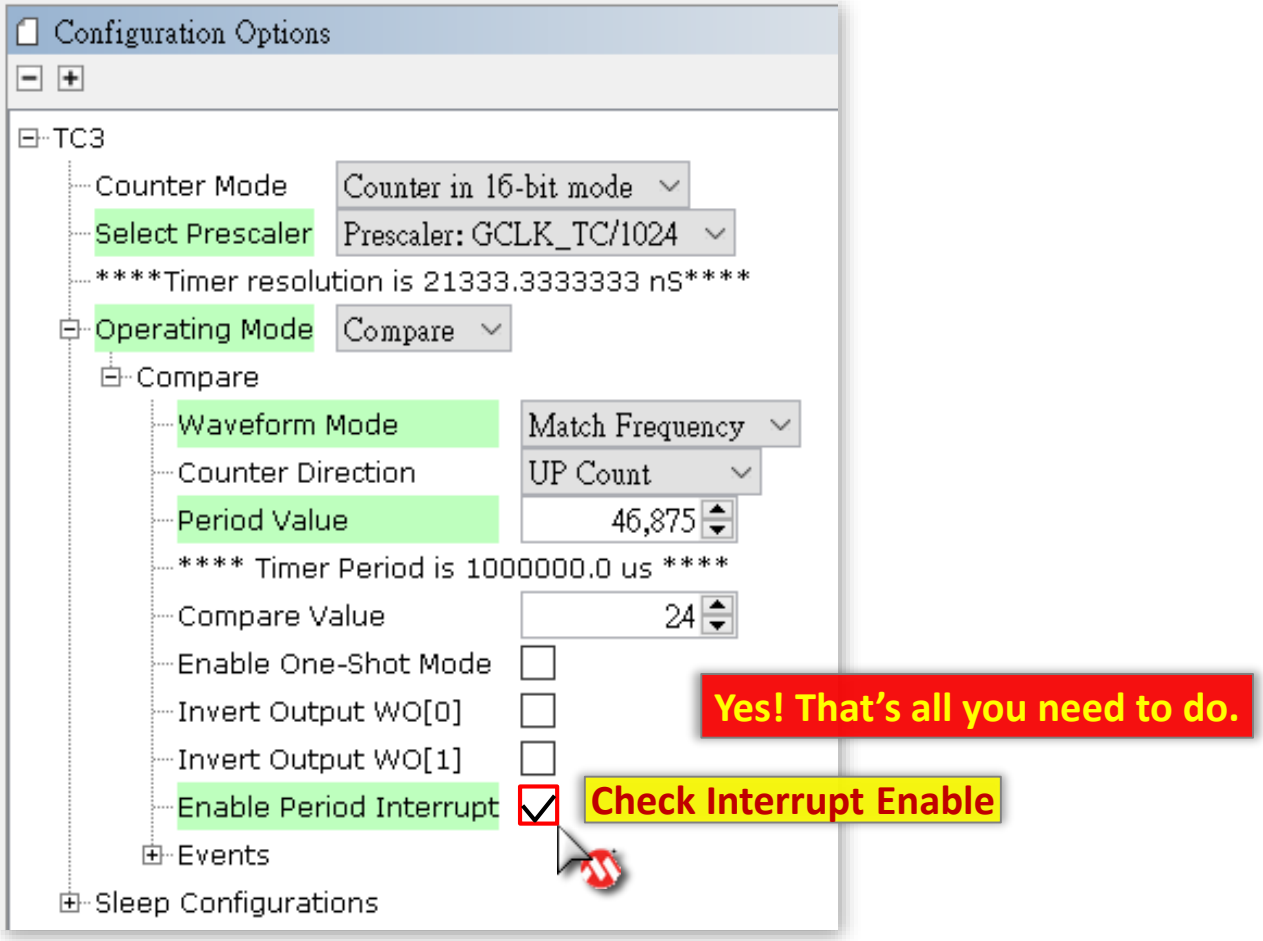
```
void TCn_CompareInterruptHandler( void )
{
   TCn_COMPARE_STATUS status;
   status = TCn_REGS->COUNT16.TC_INTFLAG;
   /* clear period interrupt */
   TCn_REGS->COUNT16.TC_INTFLAG = TC_INTFLAG_OVF_Msk;
   if( TCn_CallbackObject.callback != NULL )
   {
      TCn_CallbackObject.callback( status, TCn_CallbackObject.context );
   }
}
```

Callback

```
void TCn_Overflow(TC_COMPARE_STATUS status, uintptr_t context)
{
   if( status & TC_INTFLAG_OVF_Msk )
      LED_Toggle();
}
```

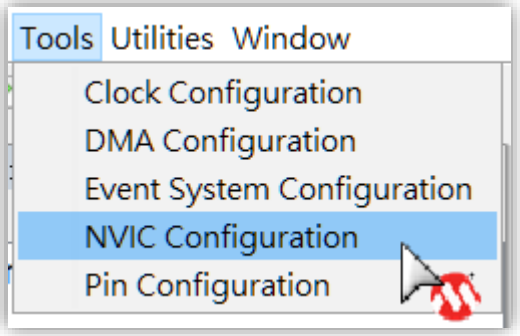
MH3 Interrupt Example

- **Enable Interrupt in TC configuration windows.**



NVIC Priority Settings in MH3

- Open NVIC Settings windows in Tools ▸ NVIC Configuration.



NVIC Settings

Vector Number	Vector	Enable	Priority (0 =	Handler Name
-15	Reset (Reset Vector)	☑	-3	Reset_Handler
-14	NonMaskableInt (Non-maskable Interrupt)	☑	-2	NonMaskableInt_Handler
-13	HardFault (Hard Fault)	☑	-1	HardFault_Handler
-5	SVCall (SuperVisor Call)	☑	0	SVCall_Handler
-2	PendSV (Pendable SerVice)	☑	0	PendSV_Handler
-1	SysTick (System Tick Timer)	☐	0	SysTick_Handler
0	PM (Power Manager)	☐	3	PM_Handler
1	SYSCTRL (System Controller)	☐	3	SYSCTRL_Handler
2	WDT (Watchdog Timer)	☐	3	WDT_Handler
3	RTC (Real Time Counter)	☐	3	RTC_Handler
4	EIC (External Interrupt Controller)	☐	3	EIC_Handler
11	SERCOM2 (Serial Communication Interface 2)	☐	3	SERCOM2_Handler
12	SERCOM3 (Serial Communication Interface 3)	☐	3	SERCOM3_Handler
13	SERCOM4 (Serial Communication Interface 4)	☐	3	SERCOM4_Handler
14	SERCOM5 (Serial Communication Interface 5)	☐	3	SERCOM5_Handler
15	TCC0 (Timer/Counter for Control Applications 0)	☐	3	TCC0_Handler
16	TCC1 (Timer/Counter for Control Applications 1)	☐	3	TCC1_Handler
17	TCC2 (Timer/Counter for Control Applications 2)	☐	3	TCC2_Handler
18	TC3 (Timer/Counter 3)	☑	3	TC3_CompareInterruptHandler
19	TC4 (Timer/Counter 4)	☑	3	TC4_CompareInterruptHandler
20	TC5 (Timer/Counter 5)	☐	3	TC5_Handler
23	ADC (Analog-to-Digital Converter)	☐	3	ADC_Handler
24	AC (Analog Comparators)	☐	3	AC_Handler
25	DAC (Digital-to-Analog Converter)	☐	3	DAC_Handler
26	PTC (Peripheral Touch Controller)	☐	3	PTC_Handler
27	I2S (Inter-IC Sound Controller)	☐	3	I2S_Handler

18	TC3 (Timer/Counter 3)	☑	3	TC3_CompareInterruptHandler
19	TC4 (Timer/Counter 4)	☑	3	TC4_CompareInterruptHandler

TC Interrupt Interface Routines

```
/*** TC3 Interface Routines***/
void TC3_CompareInitialize( void );
void TC3_CompareStart( void );
void TC3_CompareStop( void );
uint32_t TC3_CompareFrequencyGet( void );
void TC3_Compare16bitPeriodSet( uint16_t period );
uint16_t TC3_Compare16bitPeriodGet( void );
uint16_t TC3_Compare16bitCounterGet( void );
void TC3_Compare16bitCounterSet( uint16_t count );
void TC3_CompareCallbackRegister
                ( TC_COMPARE_CALLBACK callback, uintptr_t context );
```

One new function than polling mode.

Lab5 TC MFRQ Interrupt Callback

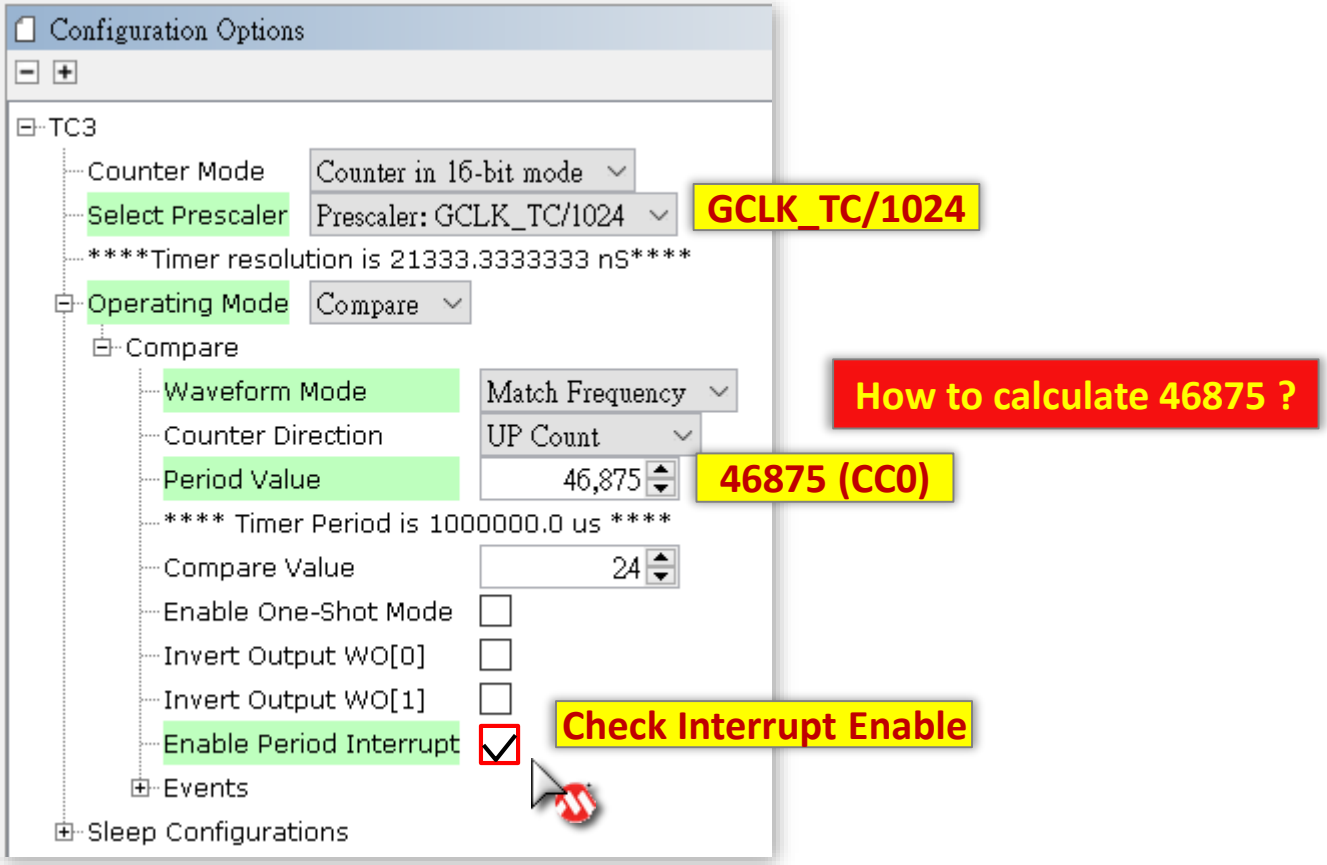
- Try to configure code style to interrupt callback style. LED1 toggle in callback while TC3 overflow interrupt.
- Modify TC3 period to 1 second in MH3 for easy observe.
- Modify code segment in main() follow below sequence.
  - Firstly, remove polling LED1 toggle from main loop.
  - Secondary, Create a callback function for LED1 toggle
  - Finally, hook up callback function to TC3 overflow event.

- Let's go!

Lab5 TC MFRQ Interrupt Callback

Step 1

- Configure TC3 Period to 1 second.
- Enable Period Interrupt of TC3 in configuration windows.

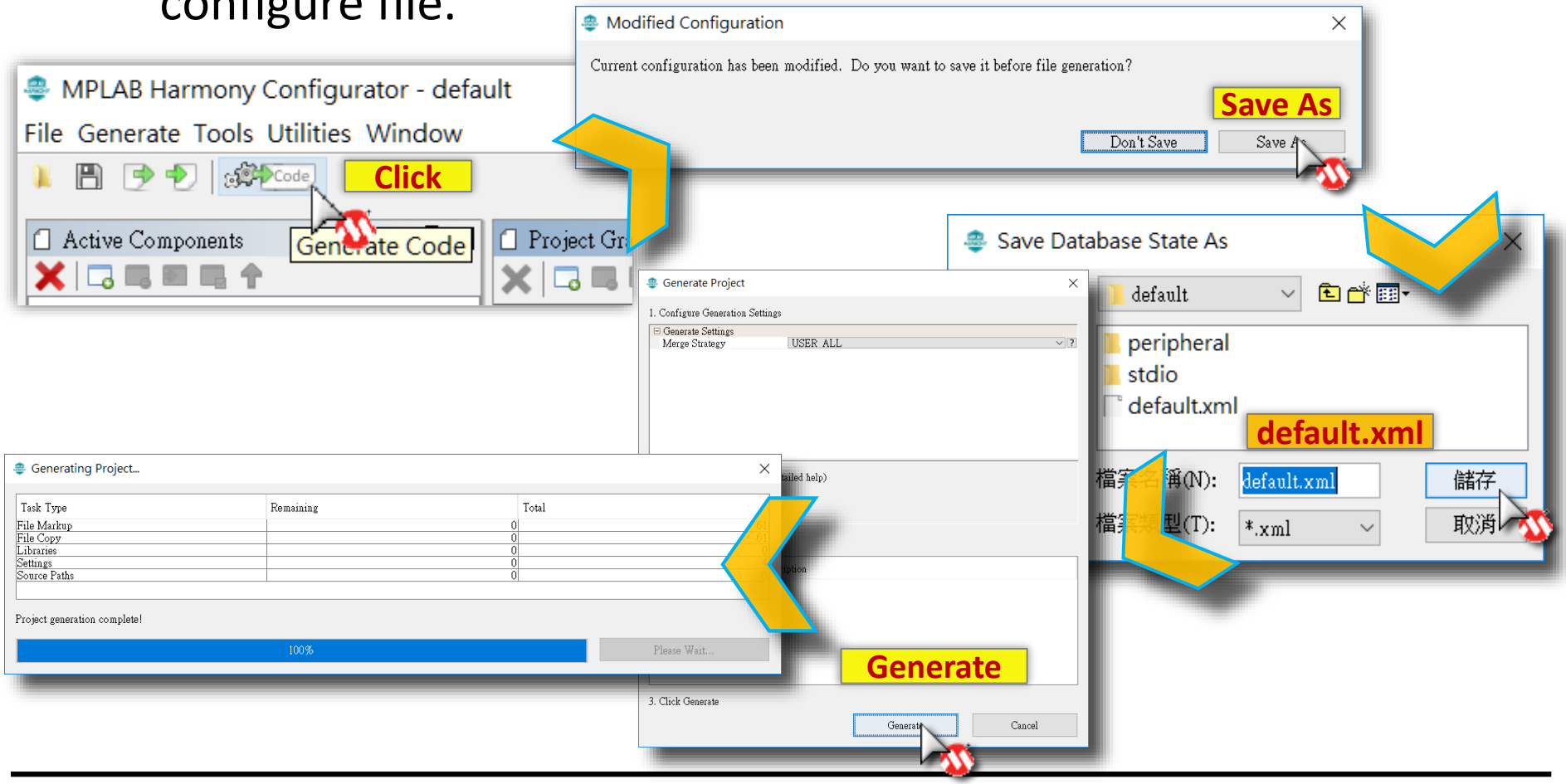


Lab5 TC MFRQ Interrupt Callback

Step 2

- Click 

to Generate Code and save changes to MHC configure file.



Lab5 TC MFRQ Interrupt Callback

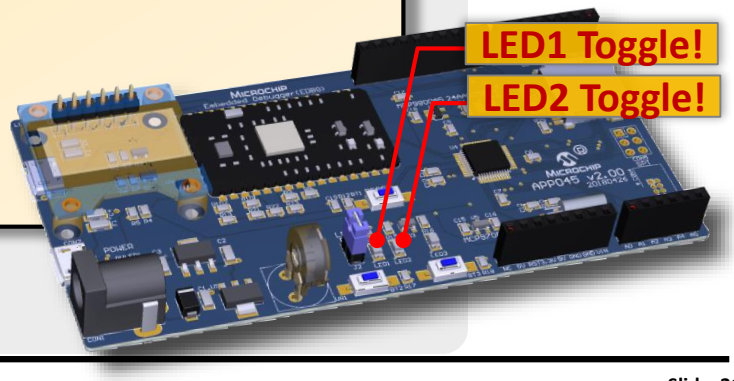
Step 3

a Add code segment to your main loop

```
// TODO 5.01
void TC3_Overflow(TC_COMPARE_STATUS status, uintptr_t context)
{
  if( status & TC_INTFLAG_OVF_Msk )
    LED1_Toggle();
}

int main (void)
{
   SYS_Initialize ( NULL );
   …
   // TODO 5.02
   TC3_CompareCallbackRegister( TC3_Overflow, (uintptr_t )NULL );
   TC3_CompareStart();

   while(1)
   { …
   }
}
```



b Program firmware to target board then observe result.

Lab6 TCs MFRQ Interrupt Callback

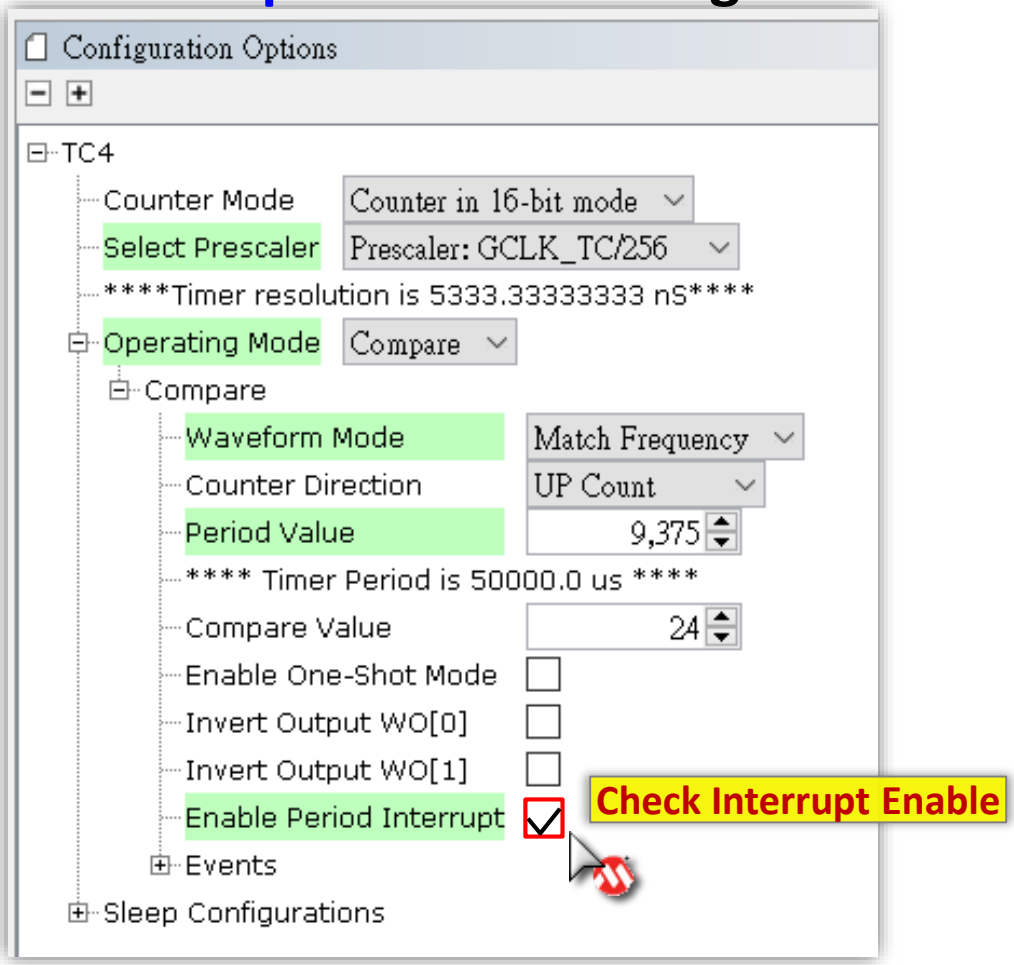
- Try to change TC4, LED2 toggle function to interrupt callback style, also.

- Let's go!

Lab6 TCs MFRQ Interrupt Callback

Step 1

- **Enable Period Interrupt** of TC4 in configuration windows.

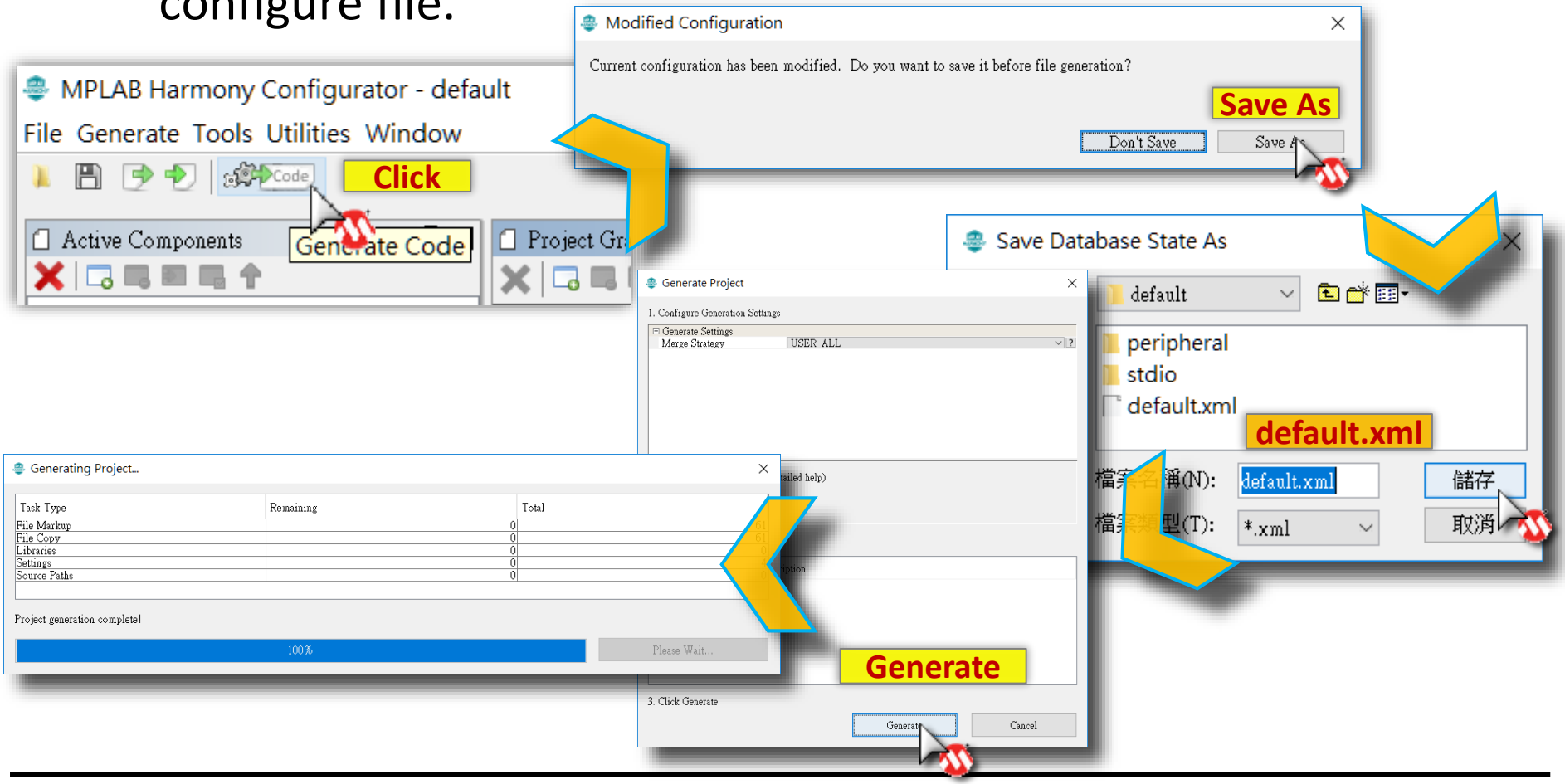


Lab6 TCs MFRQ Interrupt Callback

Step 2

- Click 

to Generate Code and save changes to MHC configure file.



Lab6 TCs MFRQ Interrupt Callback

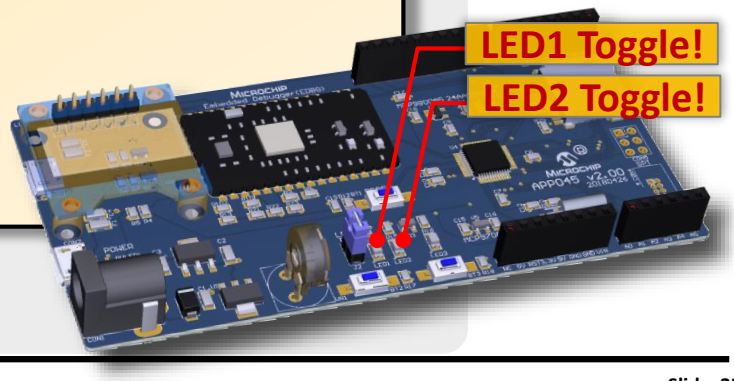
Step 3

a Add code segment to your main loop

```
// TODO 6.01
void TC4_Overflow(TC_COMPARE_STATUS status, uintptr_t context)
{
  if( status & TC_INTFLAG_OVF_Msk )
    LED2_Toggle();
}

int main (void)
{
   SYS_Initialize ( NULL );
   …
   // TODO 6.02
   TC4_CompareCallbackRegister( TC4_Overflow, (uintptr_t )NULL );
   TC4_CompareStart();

   while(1)
   { …
   }
}
```



b Program firmware to target board then observe result.

Bonus Lab

- Try use three LEDs to implement the Binary Counting.
- The MSB LED is LED1 and toggle in one second period

- **Let's go!**

	LED1	LED2	LED3
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

1 sec

1 sec