Timer

I. Learning Objectives

1. Learn the basic usage of the MSPM0G3507 development board timer.
2. Implement timed LED status toggling.

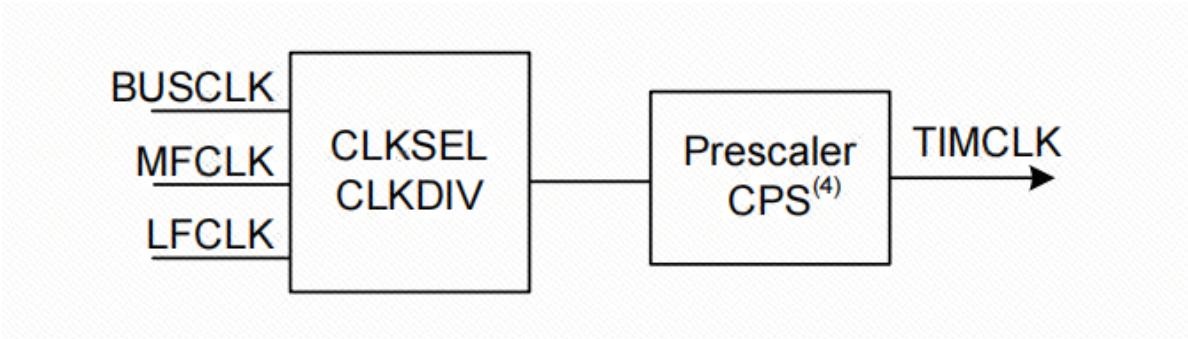
II. Timer Introduction

Timer is a very important peripheral in microcontrollers, used for timing and generating periodic interrupts. The core principle of a timer is to record time through an internal counter. The counter increments by one every machine cycle. When the counter reaches a preset value, it triggers an interrupt. The program can execute certain operations in the interrupt service routine, such as toggling LED, reading sensor data, controlling motors, etc.

Timer Basic Parameters

1. Prescaler

The prescaler can divide the timer clock (TIMCLK) frequency by any value between 1 and 256 (1 to 256 is based on the timer divider being 8-bit). TIMG can select BUSCLK, MFCLK, LFCLK as clock sources, can divide the clock by up to 8, and then passes through an 8-bit prescaler, finally becoming the timer's counting clock.



2. Up or Down Counting Mode

Up counting means the counter starts from 0 and continuously counts up to the auto-reload value. Once the counter counts to the auto-reload value, it restarts from 0 and counts up, generating an overflow event. Down counting means the counter starts from the auto-reload value and continuously counts down to 0. Once the counter counts to 0, it restarts from the auto-reload value and counts down, generating an underflow event.

3. Update Event

An update event is triggered when the counter overflows or underflows and starts a new counting cycle. The update event can trigger DMA requests to update timer operating parameters in time when the next counting cycle starts, which is especially suitable for real-time control.

III. Hardware Setup

The MSPM0G series has a total of 7 timers, which can be divided into 2 types: General Purpose Timers (TIMG) and Advanced Control Timers (TIMA). Different types of timers have different numbers of features. Generally, advanced timers have the most features, followed by general-purpose timers.

Table 25-1. TIMx Instance Configuration

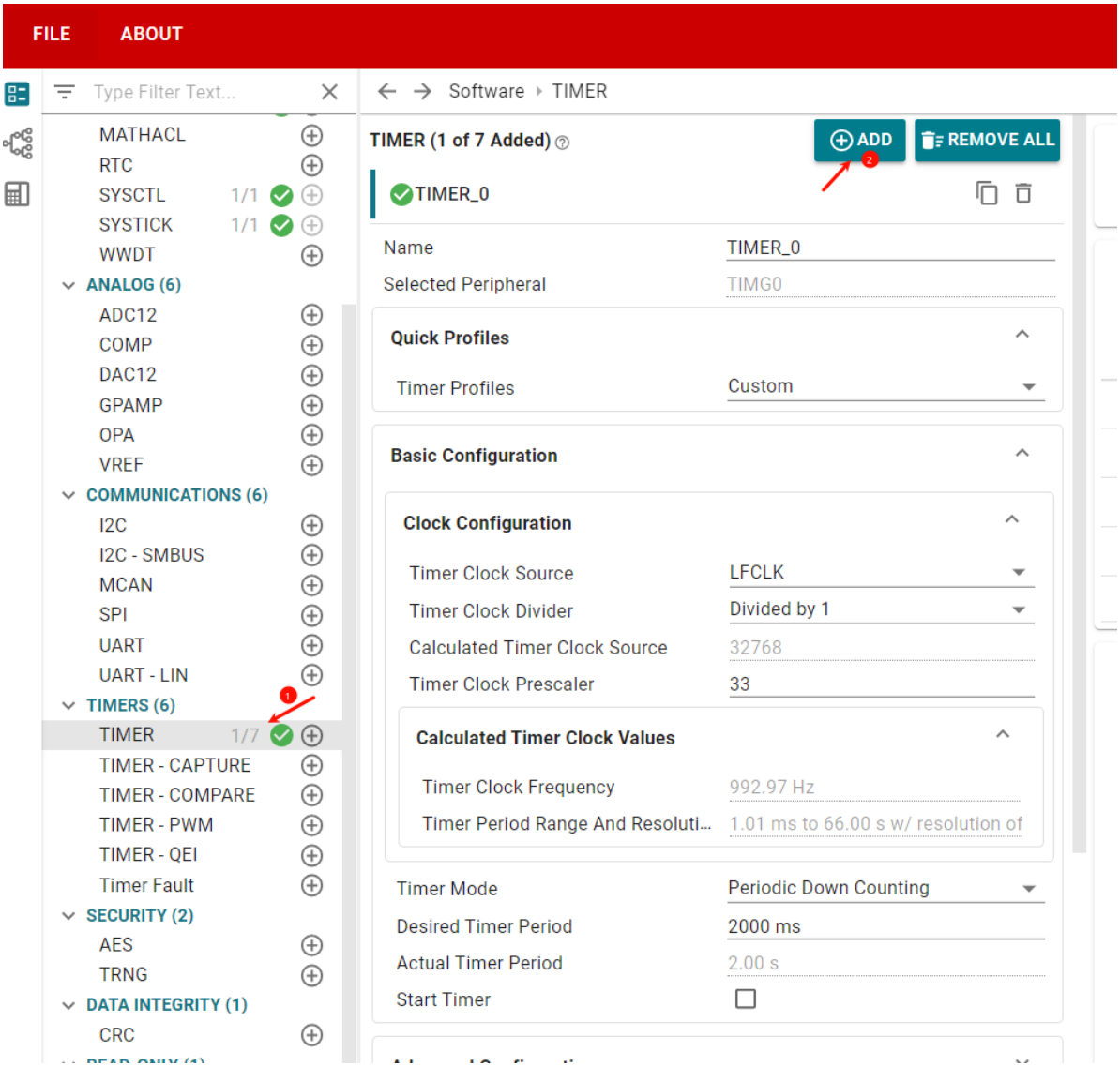
Instance	Power Domain	Counter Resolution	Prescaler	Repeat Counter	CCP Channels (External/ Internal)	External PWM Channels	Phase Load	Shadow Load	Shadow CCs	Deadband	Fault Handler	QEI / Hall Input Mode
TIMG0	PD0	16-bit	8-bit	-	2	2	-	-	-	-	-	-
TIMG6	PD1	16-bit	8-bit	-	2	2	-	Yes	Yes	-	-	-
TIMG7	PD1	16-bit	8-bit	-	2	2	-	Yes	Yes	-	-	-
TIMG8	PD0	16-bit	8-bit	-	2	2	-	-	-	-	-	Yes
TIMG12	PD1	32-bit	-	-	2	2	-	-	Yes	-	-	-
TIMA0	PD1	16-bit	8-bit	Yes	4/2	8	Yes	Yes	Yes	Yes	Yes	-
TIMA1	PD1	16-bit	8-bit	Yes	2/2	4	Yes	Yes	Yes	Yes	Yes	-

IV. Experiment Steps

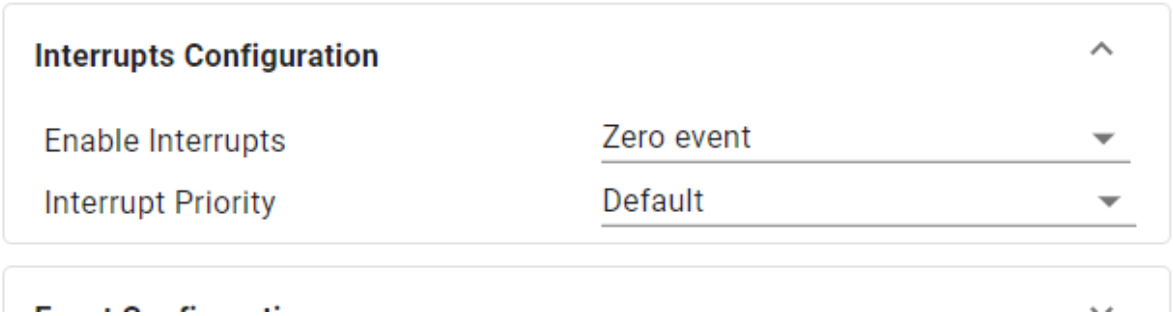
If the project needs to execute specific tasks periodically or repeatedly, timer configuration becomes particularly important. The following will introduce how to set up a timer to trigger an interrupt every 2 seconds and implement LED status toggling in the interrupt service function, thereby achieving the effect of LED on for 2 seconds and off for 2 seconds.

Open the SYSCONFIG configuration tool, open the empty project in the SDK in KEIL or migrate the LED example source code. Here we use the LED source code as an example

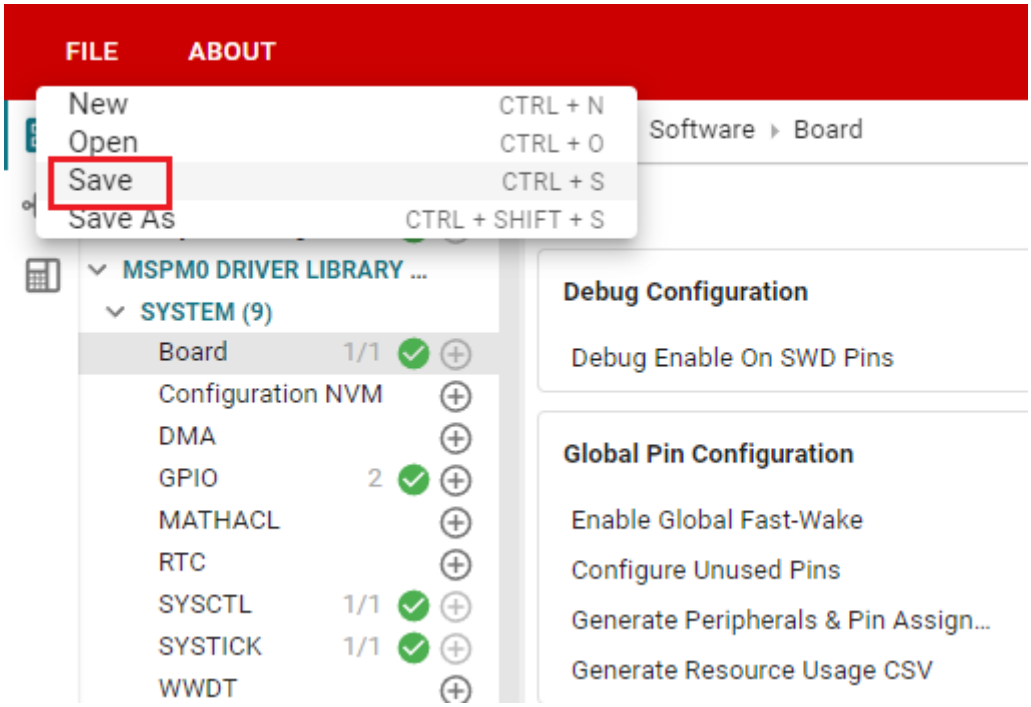
First, we open the graphical configuration tool through the sysconfig tool to add timer configuration



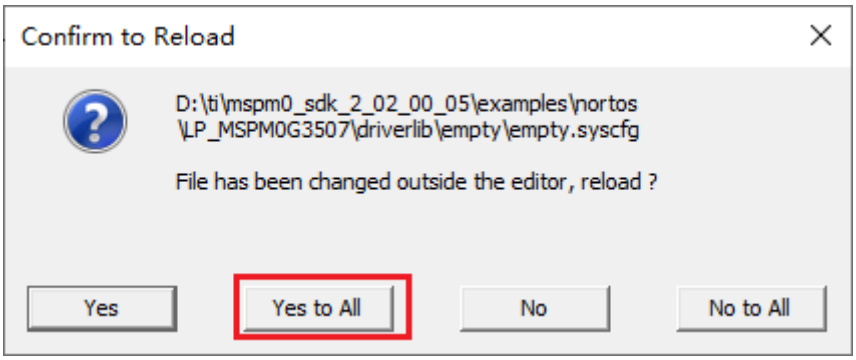
Enable timer interrupt



Click SAVE to save the configuration in SYSCONFIG, then close SYSCONFIG and return to keil.



Select `Yes to All` in the pop-up window



Similarly, we also need to confirm whether the ti_msp_dl_config.c and ti_msp_dl_config.h files are updated. Compile directly, and the compilation will automatically update to keil. If not updated, we can also copy the file contents from SYSCONFIG.

After saving the generated files, add the following code

```
#include "ti_msp_dl_config.h"

int main(void)
{
    // System configuration initialization
    SYSCFG_DL_init();

    /* Enable device TimerG interrupt */
    NVIC_EnableIRQ(TIMER_0_INST_INT_IRQN);

    /* Enable sleep on exit */
    DL_SYSCTL_enableSleepOnExit();

    /* Set LED initial state to indicate timer counter is enabled */
    DL_GPIO_clearPins(LED_PORT, LED_MCU_PIN);

    /* Start TimerG counter */
    DL_TimerG_startCounter(TIMER_0_INST);
```

```
    // Main loop - wait for interrupt
    while (1) {
        __WFI(); //  Wait for interrupt
    }
}

/**
 * @brief Timer interrupt service function
 */
void TIMER_0_INST_IRQHandler(void)
{
    // Check and process timer interrupt
    switch (DL_TimerG_getPendingInterrupt(TIMER_0_INST)) {
        // Zero count interrupt
        case DL_TIMER_IIDX_ZERO:
            // Toggle LED status every 2 seconds
            DL_GPIO_togglePins(LED_PORT, LED_MCU_PIN);
            break;

        // Other interrupt cases
        default:
            break;
    }
}
```

V. Experiment Phenomenon

After flashing the program, the MCU indicator LED (red) is on for two seconds and off for two seconds