Timer 使用指南

概述

本手册适用范围：

适用范围	
SPC1125 系列	SPC1125，SPC1128
SPC1168 系列	SPC1155，SPC1156，SPC1158，SPC1168，SPD1148，SPD1178，SPD1188，SPD1163，SPM1173
SPC2168 系列	SPC2168，SPC2165，SPC2166，SPC1198
SPC1169 系列	SPC1169，SPD1179，SPD1176
SPC2188 系列	SPC1185，SPC2188

目录

图片列表

表格列表

版本历史

版本	日期	作者	状态	变更
C/0	2024-06-07	HangSu	Outdated	1. 首次发布
C/1	2024-08-20	HangSu	Released	1. 增加章节 2。

1 SPC1169/SPC2188 系列

1.1 特性

具有以下特性：

- 32 位向下计数器，具有专用时钟；
- 在计数器达到零时生成中断；
- 在计数器达到零时生成 ADCSOC 事件；
- 在计数器达到零时生成 PWMSYNCI 事件。

支持 4 种模式：

- 通用定时器：周期性向下计数器；
- 门控定时器：由外部输入电平启用的向下计数器；
- 事件计数器：在外部输入边沿触发的向下计数器；
- 事件捕获：在外部输入边沿触发时，记录时间戳的向下计数器。

1.2 功能描述

如图 1-1 所示，每一个通用定时器都可采用以下方式向下计数：

- 通用定时器：周期性向下计数器；
- 门控定时器：由外部输入电平启用的向下计数器；
- 事件计数器：在外部输入边沿触发的向下计数器；
- 事件捕获：在外部输入边沿触发时，记录时间戳的向下计数器。

当计数到 0 后，计数器加载 TMRLOAD 寄存器的值并重新开始向下计数，且有如下动作发生：

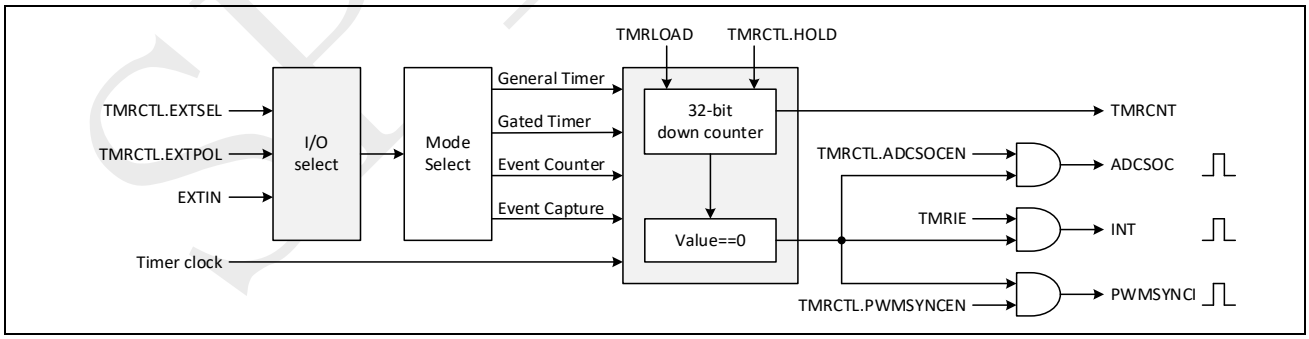
- 如果使能了定时器中断，将会有中断产生，并锁存中断标志直到被手动清除；
- 如果 TMRCTL.ADCSOCEN 使能，将会产生一个 ADCSOC 事件；
- 如果 TMRCTL.PWMSYNCEN 使能，将会产生一个 PWMSYNCI 事件。

用于外部信号输入的 I/O 引脚及其有效的电平或者边沿可以通过 TMRCTL.EXTSEL 和 TMRCTL.EXTPOL 寄存器位段控制。

表 1-1：TMRCTL.EXTPOL 位段描述

	0	1
TMRCTL.EXTPOL	门控定时模式：输入为低时使能计数器； 事件计数/捕获模式：输入下降沿触发事件	门控定时模式：输入为高时使能计数器； 事件计数/捕获模式：输入上升沿触发事件

图 1-1：通用定时器框图



1.3 功能实例

1.3.1 产生 ADCSOC 事件

1.3.1.1 功能需求

通过定时器触发 ADC 采样。

1.3.1.2 功能实现

用 General Timer 计数信号作为 ADCSOC 信号的源，触发 ADC 采样。

图 1-2：ADCSOC 触发框图

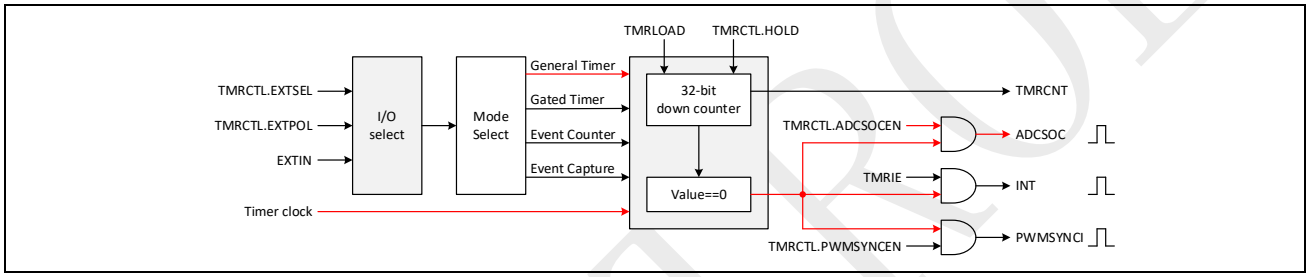


表 1-2：代码路径

MCU 产品类型	代码路径
SPC1169 系列	0_Examples\10_6_ADC_Triggered_by_Timer
SPC2188 系列	0_Examples\13_ADC_Triggered_by_Timer

1.3.2 产生 PWMSYNCI 事件

1.3.2.1 功能需求

通过定时器触发 PWM 同步。

1.3.2.2 功能实现

用 General Timer 计数信号作为 PWMSYNCI 信号的源，触发 PWM 同步。

图 1-3：PWMSYNCI 触发框图

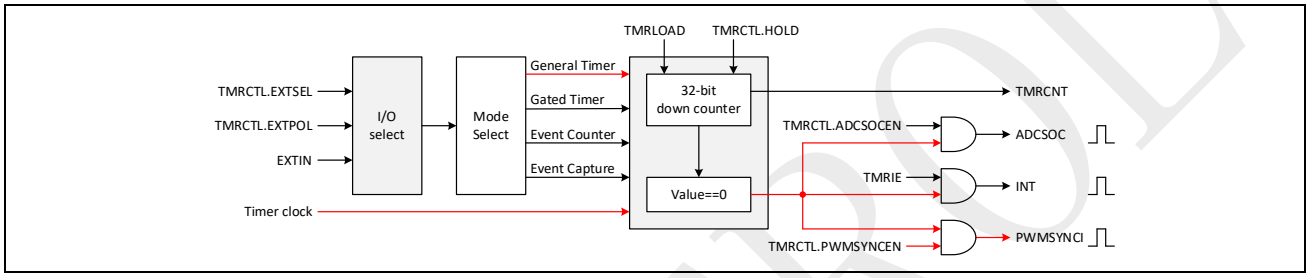


表 1-3：代码路径

MCU 产品类型	代码路径
SPC1169 系列	0_Examples\8_9_PWM_Timer_SYNC
SPC2188 系列	0_Examples\17_PWM_Timer_SYNC

1.3.3 通用计时模式

1.3.3.1 功能需求

通过定时器触发中断。

1.3.3.2 功能实现

如图 1-4 所示，该模式下 32 位计时器在模块时钟上升沿递减计数，计数到 0 时重新加载初值并继续计数。寄存器 TMRCNT 的回读值始终是实时的计数值。

图 1-4：通用定时器在通用定时模式下的波形（TMRLOAD=100）

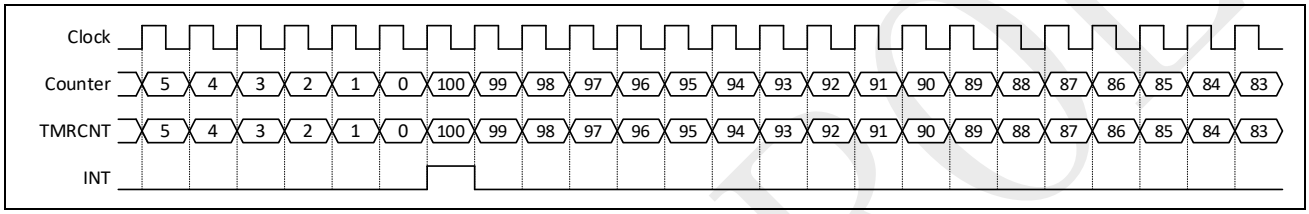


图 1-5：General Timer 框图

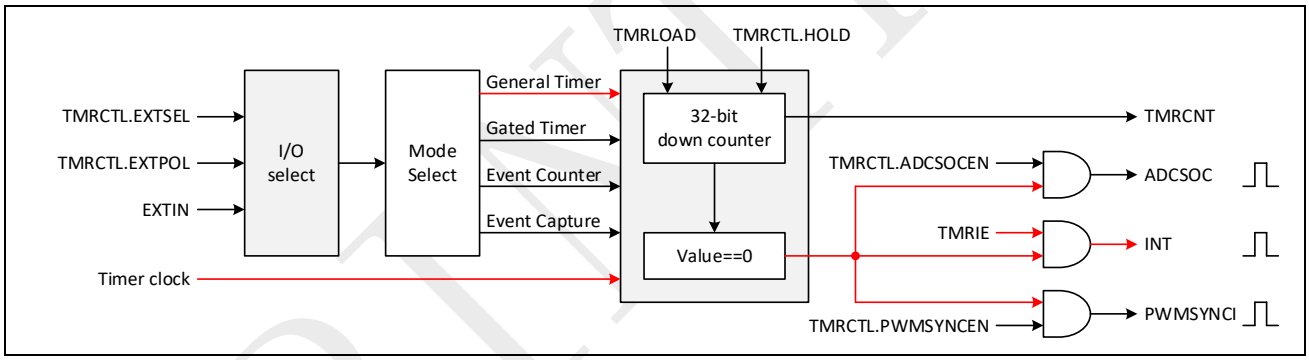


表 1-4：代码路径

MCU 产品类型	代码路径
SPC1169 系列	0_Examples\6_1_GTimer_Int
SPC2188 系列	0_Examples\6_GTimer_Int

1.3.4 门控定时模式

1.3.4.1 功能需求

通过外部引脚控制定时器使能，并在计数到 0 后触发中断。

1.3.4.2 功能实现

如图 1-6 所示，该模式与通用定时模式类似，只是计数器仅在外部引脚输入有效电平时才计数。用于外部输入的 I/O 引脚及有效电平软件可配。寄存器 TMRCNT 的回读值始终是实时的计数值。

图 1-6：通用定时器在门控定时模式下的波形（TMRLOAD=100）

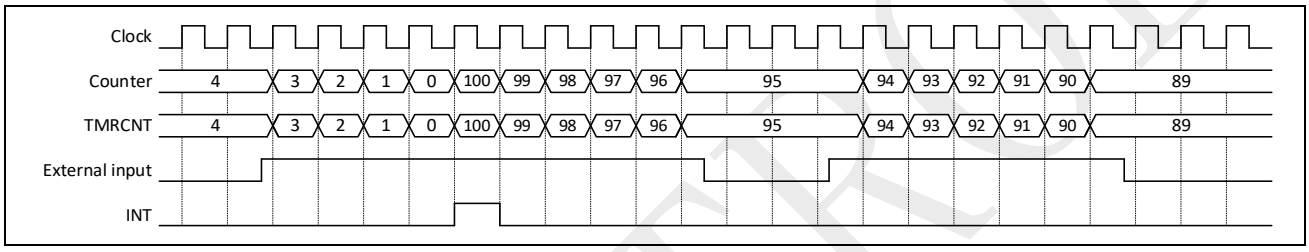
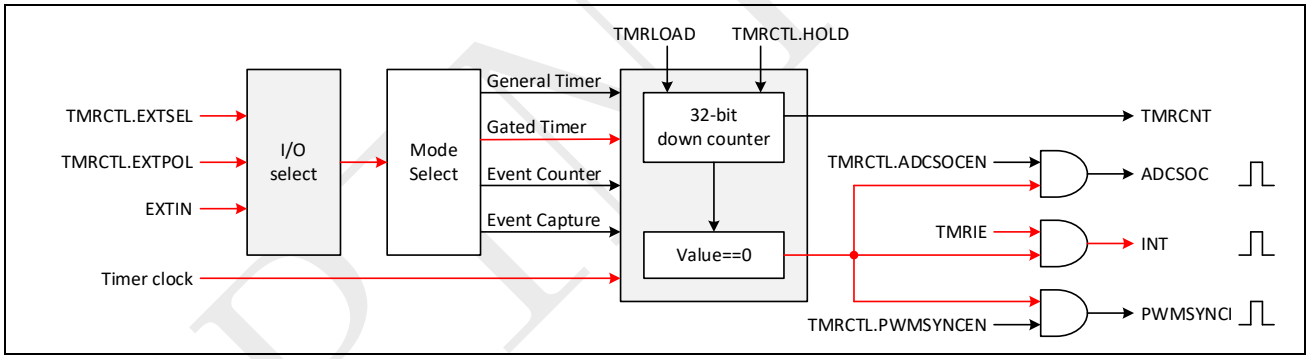


图 1-7：Gated Timer 框图



选中 GPIO24 高电平作为使能信号，以开启定时器；

其中 GPIO24 电平由 GPIO23 给定，使用前需要将其连接。

main.c

```
#include <stdio.h>

#include "spc1169.h"


#define TimerPeridCount         100


int main(void)
{
    CLOCK_InitWithRCO(100000000);

    Delay_Init();
```

main.c

```
    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

     printf("Enter the test\n");

    /* Init PINMUX */
    PIN_SetChannel(PIN_GPIO23, PIN_GPIO23_GPIO23);
    PIN_SetChannel(PIN_GPIO24, PIN_GPIO24_GPIO24);

    /* Set PIN_GPIO23 as GPIO FUNC */
    PIN_SetPinAsGPIO(PIN_GPIO23);

    /* Set PIN_GPIO23 direction */
    GPIO_SetPinDir(PIN_GPIO23, GPIO_OUTPUT);

    /* Set the GPIO as low level */
    GPIO_SetHigh(PIN_GPIO23);

    /* Init Timer */
    TIMER_Init(TIMER1, TimerPeridCount);

    /* Set External Input PIN and level */
    TIMER_SetExternalInput(TIMER1, PIN_GPIO24, GPIO_LEVEL_HIGH);

    /* Set Mode */
    TIMER_SetMode(TIMER1, TIMER_GATED_TIMER);

    /* Overwrite TMRLOAD */
    TIMER_SetReloadValue(TIMER1, TimerPeridCount);

    /* Open Global INT for Timer */
    NVIC_EnableIRQ(TIMER1_IRQn);

    /* Enable Timer */
    TIMER_Enable(TIMER1);

    while (1)
    {
    }
}


void TIMER1_IRQHandler()
{
    printf("G timer INT test OK\n");

    /* Clear the INT */
    TIMER_ClearInt(TIMER1);
}
```

[1] 示例代码适用于 SPC1169。其它系列产品的示例代码会根据实际需求进行补充。

1.3.5 事件计数模式

1.3.5.1 功能需求

捕获到外部引脚上对应数目的边沿信号后，触发中断。

1.3.5.2 功能实现

如图 1-8 所示，该模式下计数器仅当在外部输入引脚上检测到一个有效边沿的时候才计数。当计数到 0 时，将自动加载初始值，为下一轮计数做准备。因此，每当外部输入引脚上累计检测到的有效边沿数等于 TMRLOAD 时，将产生一个中断。该模式可以帮助用户在某个引脚上检测预定数量的信号沿并通过中断告知。用于外部输入的 I/O 引脚及有效电平软件可配。寄存器 TMRCNT 的回读值始终是实时的计数值。为了能正确检测信号沿，外部输入信号的高低电平保持时间必须大于定时器模块的时钟周期。

图 1-8：通用定时器在事件计数模式下的波形（TMRLOAD=100）

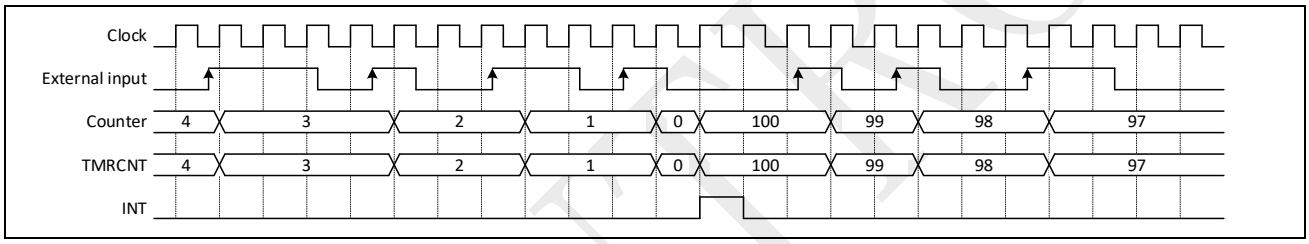
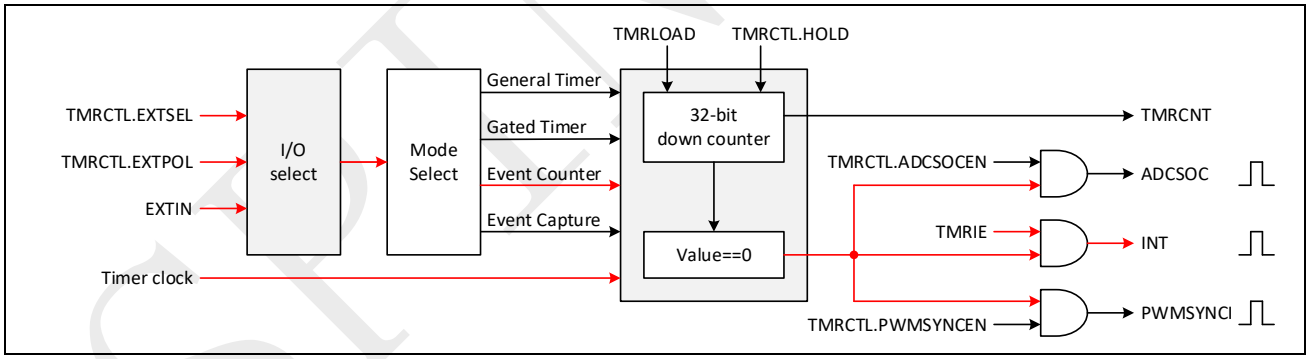


图 1-9：Event Counter 框图



选中 GPIO24 上升沿作为计数事件；

其中 GPIO24 电平由 GPIO23 给定，使用前需要将其连接。

main.c

```
#include <stdio.h>

#include "spc1169.h"


#define TimerPeridCount         100


int main(void)
{
```

main.c

```
    uint32_t i;
    CLOCK_InitWithRCO(100000000);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

     printf("Enter the test\n");

    /* Init PINMUX */
    PIN_SetChannel(PIN_GPIO23, PIN_GPIO23_GPIO23);
    PIN_SetChannel(PIN_GPIO24, PIN_GPIO24_GPIO24);

    /* Set PIN_GPIO23 as GPIO FUNC */
    PIN_SetPinAsGPIO(PIN_GPIO23);

    /* Set PIN_GPIO23 direction */
    GPIO_SetPinDir(PIN_GPIO23, GPIO_OUTPUT);

    /* Set the GPIO as low level */
    GPIO_SetLow(PIN_GPIO23);

    /* Init Timer */
    TIMER_Init(TIMER1, TimerPeridCount);

    /* Set External Input PIN and edge */
    TIMER_SetExternalInput(TIMER1, PIN_GPIO24, GPIO_LEVEL_HIGH);

    /* Set Mode */
    TIMER_SetMode(TIMER1, TIMER_EVENT_COUNTER);

    /* Overwrite TMRLOAD */
    TIMER_SetReloadValue(TIMER1, TimerPeridCount);

    /* Open Global INT for Timer */
    NVIC_EnableIRQ(TIMER1_IRQn);

    /* Enable Timer */
    TIMER_Enable(TIMER1);

    printf("%d\n", TIMER_GetCounterValue(TIMER1));

    for (i = 0; i < TimerPeridCount; i++)
    {
        GPIO_SetHigh(PIN_GPIO23);
        Delay_Ms(1);
        GPIO_SetLow(PIN_GPIO23);
        Delay_Ms(1);
    }

    while (1)
    {
    }
}
```

main.c

```
void TIMER1_IRQHandler()
{
    printf("G timer INT test OK\n");
    printf("%d\n", TIMER_GetCounterValue(TIMER1));

    /* Clear the INT */
    TIMER_ClearInt(TIMER1);
}
```

[1] 示例代码适用于 SPC1169。其它系列产品的示例代码会根据实际需求进行补充。

1.3.6 事件捕获模式

1.3.6.1 功能需求

捕获到外部引脚上对应边沿信号的触发时刻，并在计数到 0 后触发中断。

1.3.6.2 功能实现

如图 1-10 所示，该模式下计数器的行为与通用定时模式完全相同。但是，TMRCNT 寄存器的回读值并非实时的计数值，而是检测到外部输入有效边沿时锁存的计数值。用于外部输入的 I/O 引脚及有效电平软件可配。为了能正确检测信号沿，外部输入信号的高低电平保持时间必须大于定时器模块的时钟周期。

图 1-10：通用定时器在事件捕获模式下的波形（TMRLOAD=100）

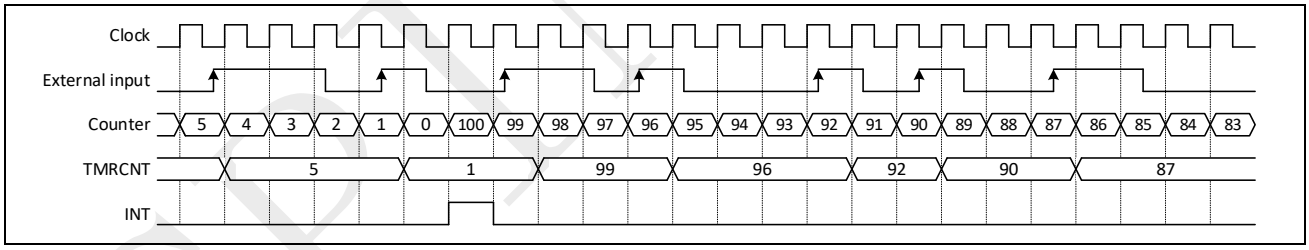
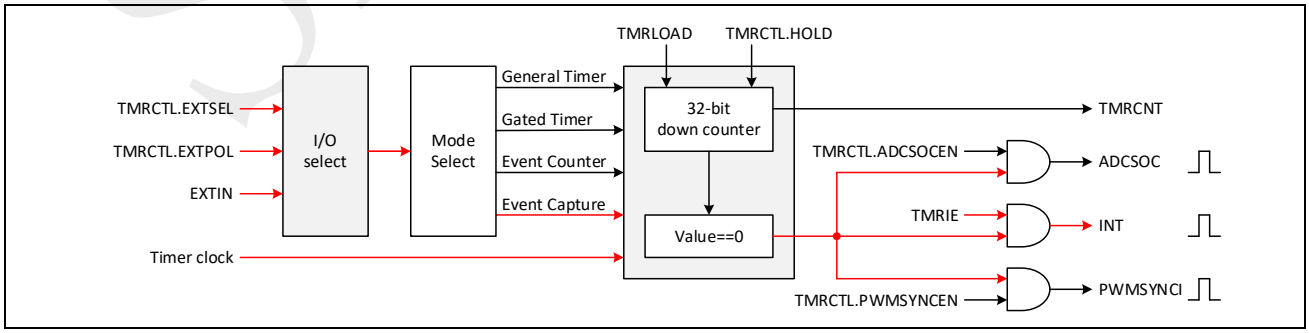


图 1-11：Event Capture 框图



选中 GPIO24 上升沿作为计数值捕获时机；

其中 GPIO24 电平由 GPIO23 给定，使用前需要将其连接。

main.c

```
#include <stdio.h>

#include "spc1169.h"


#define TimerPeridCount         100
uint32_t u32Cnt1, u32Cnt2;


int main(void)
{
    uint32_t i;
    CLOCK_InitWithRCO(100000000);

    Delay_Init();

    /*
     * Init the UART
     */
    PIN_SetChannel(PIN_GPIO10, PIN_GPIO10_UART0_TXD);
    PIN_SetChannel(PIN_GPIO11, PIN_GPIO11_UART0_RXD);
    UART_Init(UART0, 38400);

     printf("Enter the test\n");

    /* Init PINMUX */
    PIN_SetChannel(PIN_GPIO23, PIN_GPIO23_GPIO23);
    PIN_SetChannel(PIN_GPIO24, PIN_GPIO24_GPIO24);


    /* Set PIN_GPIO23 as GPIO FUNC */
    PIN_SetPinAsGPIO(PIN_GPIO23);

    /* Set PIN_GPIO23 direction */
    GPIO_SetPinDir(PIN_GPIO23, GPIO_OUTPUT);

    /* Set the GPIO as low level */
    GPIO_SetLow(PIN_GPIO23);

    /* Init Timer */
    TIMER_Init(TIMER1, TimerPeridCount);

    /* Set External Input PIN and edge */
    TIMER_SetExternalInput(TIMER1, PIN_GPIO24, GPIO_LEVEL_HIGH);

    /* Set Mode */
    TIMER_SetMode(TIMER1, TIMER_EVENT_CAPTURE);

    /* Overwrite TMRLOAD */
    TIMER_SetReloadValue(TIMER1, TimerPeridCount);

    /* Open Global INT for Timer */
    NVIC_EnableIRQ(TIMER1_IRQn);

    /* Enable Timer */
    TIMER_Enable(TIMER1);

    GPIO_SetHigh(PIN_GPIO23);
    Delay_Ms(1);
    u32Cnt1 =  TIMER_GetCounterValue(TIMER1);
    GPIO_SetLow(PIN_GPIO23);
```

main.c

```
    Delay_Ms(1);
    u32Cnt2 =  TIMER_GetCounterValue(TIMER1);

    while (1)
    {
    }
}


void TIMER1_IRQHandler()
{
    if (u32Cnt1 == u32Cnt2)
    {
        printf("G timer INT test OK\n");
        printf("%d\n", TIMER_GetCounterValue(TIMER1));
    }

    /* Clear the INT */
    TIMER_ClearInt(TIMER1);
}
```

[1] 示例代码适用于 SPC1169。其它系列产品的示例代码会根据实际需求进行补充。

2 SPC1168/SPC2168/SPC1125 系列

2.1 特性

具有以下特性：

- 32 位向下计数器，具有专用时钟；
- 在计数器达到零时生成中断；
- 在计数器达到零时生成 ADCSOC 事件；
- 在计数器达到零时生成 PWMSYNCI 事件。

支持 3 种模式：

- 周期性向下计数器；
- 捕获外部输入作为定时器的使能信号；
- 捕获外部输入作为定时器的时钟源。

2.2 功能描述

如图 2-1 所示，每一个通用定时器都可采用以下方式向下计数：

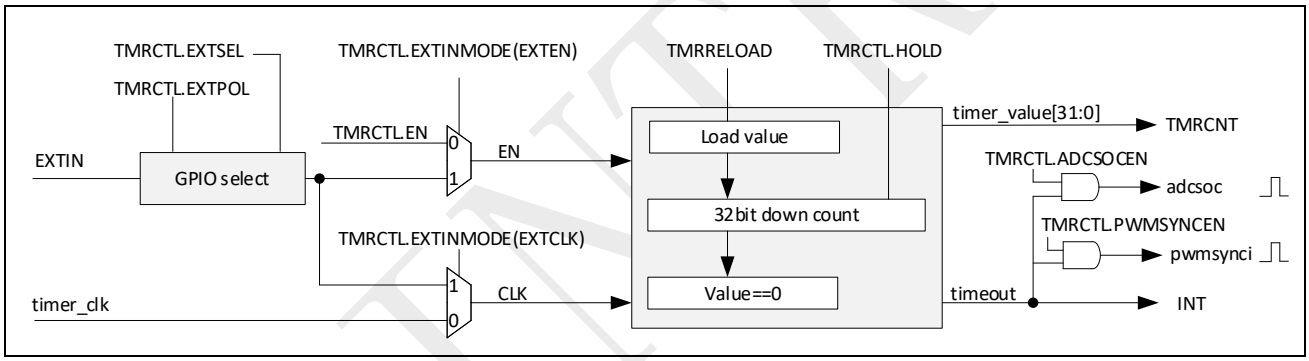
- 周期性向下计数器；
- 捕获外部输入作为定时器的使能信号；
- 捕获外部输入作为定时器的时钟源。

当计数到 0 后，计数器加载 TMRRELOAD 寄存器的值并重新开始向下计数，且有如下动作发生：

- 如果使能了定时器中断，将会有中断产生，这个中断标志将会一直存在直到被手动清除；
- 如果 TMRCTL.ADCSOCEN 使能，将会产生一个 ADCSOC 事件；
- 如果 TMRCTL.PWMSYNCEN 使能，将会产生一个 PWMSYNCI 事件。

用于外部信号输入的 I/O 引脚及其有效的电平或者边沿可以通过 TMRCTL.EXTSEL 和 TMRCTL.EXTPOL 寄存器位段控制。

图 2-1：通用定时器框图



2.3 功能实例

2.3.1 产生 ADCSOC 事件

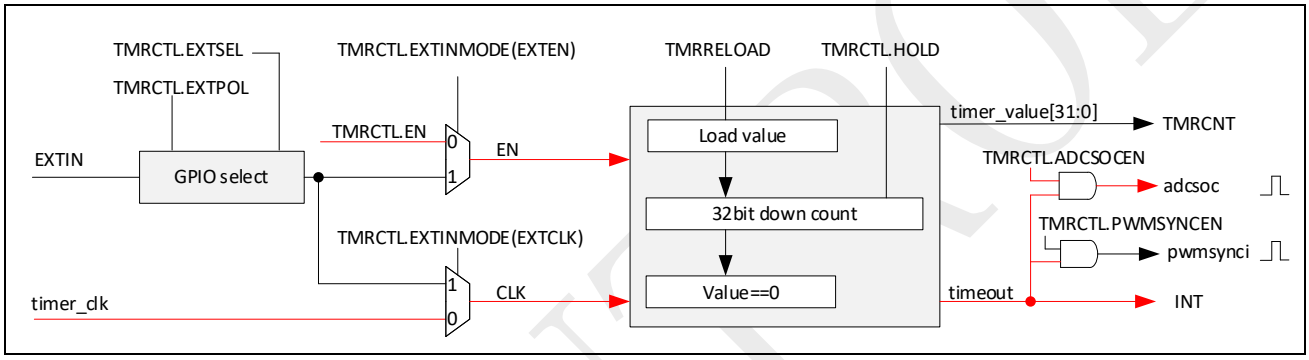
2.3.1.1 功能需求

通过定时器触发 ADC 采样。

2.3.1.2 功能实现

用 General Timer 计数信号作为 ADCSOC 信号的源，触发 ADC 采样。

图 2-2：ADCSOC 触发框图



main.c

```
#include<stdio.h>

#if defined(SPC1158)
#include "spc1158.h"
#elif defined(SPD1148)
#include "spd1148.h"
#elif defined(SPD1178)
#include "spd1178.h"
#elif defined(SPD1188)
#include "spd1188.h"
#elif defined(SPD1163)
#include "spd1163.h"
#else
#include "spc1168.h"
#endif

/* As description below, ADC result convert to voltage can calculate as the
follow */
#define          ValueToVoltage(x)        ((x * 3657) / 8192) /1000

int32_t          i32VSP;


/*****************************************************************************
*****************************************
*
* @brief     In this case, we use one GPIO as the input, then getting the GPIO
voltage digital code from the result register.
*
```

main.c

```
*******************************************************************************
*****************************************/


int main()
{
    FLASH_WALLOW();

#if defined(SPC1158)
    FLASH_SetTiming(100000000);
    /* Disable flash write access after flash operation had done */
    FLASH_WDIS();

    CLOCK_InitWithRCO(100000000UL);
#else
    FLASH_SetTiming(200000000);
    /* Disable flash write access after flash operation had done */
    FLASH_WDIS();

    CLOCK_InitWithRCO(200000000UL);
#endif

    Delay_Init();

    /*
     * Init the UART
     *
     * 1.Set the GPIO34/35 as UART FUNC
     *
     * 2.Enable the UART CLK
     *
     * 3.Set the rest para
     */
    GPIO_SetPinChannel(GPIO_34, GPIO34_UART_TXD);
    GPIO_SetPinChannel(GPIO_35, GPIO35_UART_RXD);
    CLOCK_EnableModule(UART_MODULE);
    UART_Init(UART, 38400);

    printf("Enter the test\n");

    /*ADC Init*/
    GPIO_SetPinChannel(GPIO_8, GPIO8_ADC8);
    ADC_Init(ADC_SOC_0, ADC_SHC_P_ANA_IN8, ADC_SHC_N_GND, SHC, ADCTRIG_Timer0);

    /* Enable G-INT for SOC0 */
    NVIC_EnableIRQ(ADC0_IRQn);

    /* Init TIMER0 to count for 1000ms */
    TIMER_Init(TIMER0, 1000);

    /* Enable TIMER0 to generate a signal to trigger ADC SOC to work */
    TIMER_EnableADCSOC(TIMER0);

    while (1)
    {

    }
}

void ADC0_IRQHandler(void)
{
```

main.c

```
    /* Result gotten from 'ADC_GetResult()' can be a negative value or a
positive value */
    i32VSP = ADC_GetResult(ADC_SOC_0);
    printf("ADC SOC0 Voltage = %2fV(Raw data is %d)\n",
(double)ValueToVoltage(i32VSP), i32VSP);

    /* Result gotten from 'ADC_GetTrimResult1()' is a positive value */
    i32VSP = ADC_GetTrimResult1(ADC_SOC_0);
    printf("ADC SOC0 Voltage = %2fV(Trim data is %d)\n",
(double)ValueToVoltage(i32VSP), i32VSP);

    /* Clear ADC SOC0 INT flag */
    ADC_ClearInt(ADC_SOC_0);
}
```

[1] 示例代码适用于 SPC1169。其它系列产品的示例代码会根据实际需求进行补充。

2.3.2 产生 PWMSYNCI 事件

2.3.2.1 功能需求

通过定时器触发 PWM 同步。

2.3.2.2 功能实现

用 General Timer 计数信号作为 PWMSYNCI 信号的源，触发 PWM 同步。

图 2-3：PWMSYNCI 触发框图

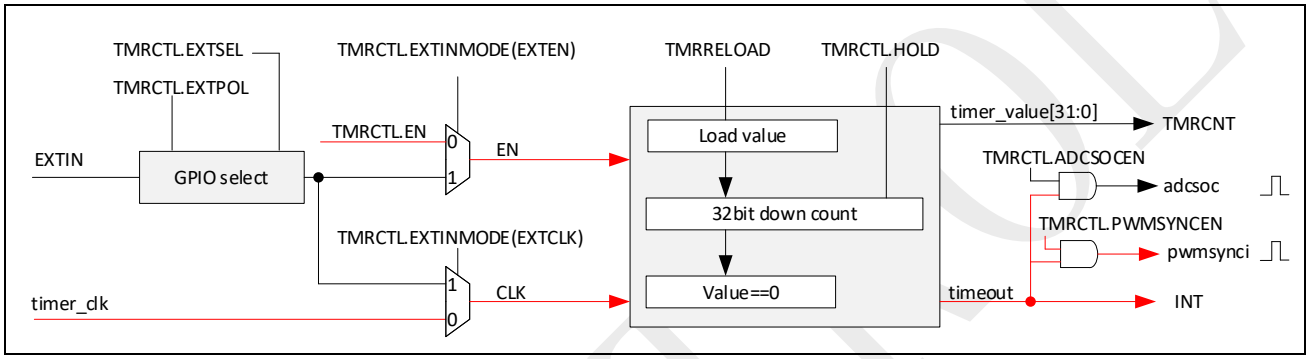


表 2-1：代码路径

MCU 产品类型	代码路径
SPC168 系列	0_Examples\PWM_Global_Timer_SYNC
SPC2168 系列	0_Examples\PWM_Global_Timer_SYNC
SPC1125 系列	0_Examples\PWM_Timer_SYNC

2.3.3 通用计时模式

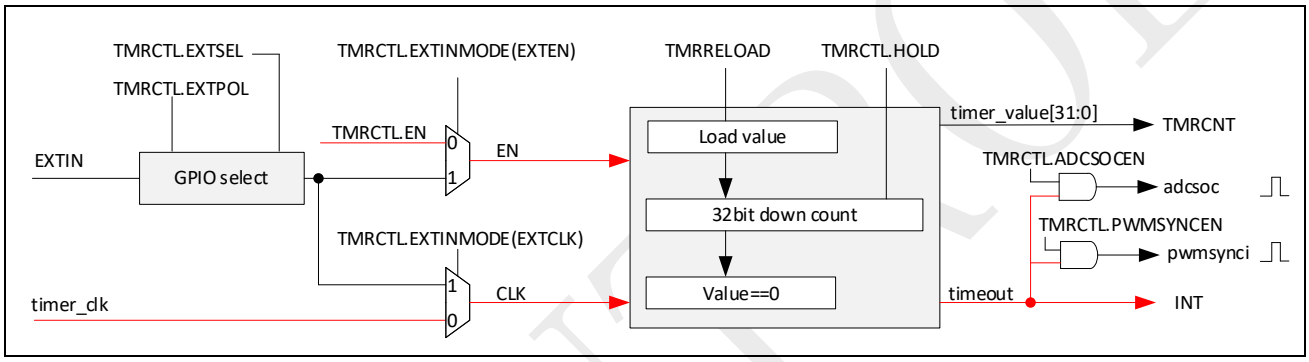
2.3.3.1 功能需求

通过定时器触发中断。

2.3.3.2 功能实现

TMRCTL.EXTINMODE 设为 0 以关闭外部输入，该模式下 32 位计时器在模块时钟上升沿递减计数，计数到 0 时重新加载初值并继续计数。寄存器 TMRCNT 的回读值始终是实时的计数值。

图 2-4：General Timer 框图



用 timer_clk 作为 CLK，TMCTL.EN 作为使能信号。

表 2-2：代码路径

MCU 产品类型	代码路径
SPC168 系列	0_Examples\GTimer_INT
SPC2168 系列	0_Examples\GTimer_INT
SPC1125 系列	0_Examples\Gtimer_Int

2.3.4 门控定时模式

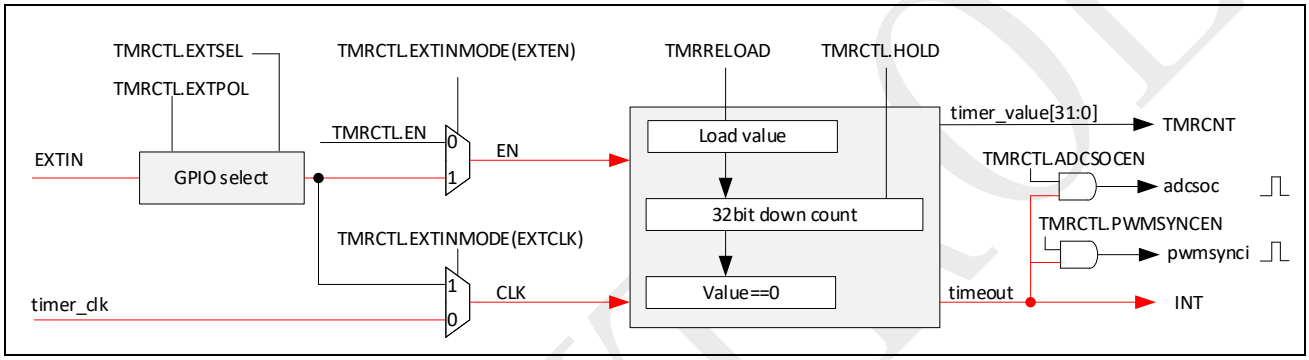
2.3.4.1 功能需求

通过外部引脚控制定时器使能，并在计数到 0 后触发中断。

2.3.4.2 功能实现

TMRCTL.EXTINMODE 设为 EXTEN，该模式与通用定时模式类似，只是计数器仅在外部引脚输入有效电平时才计数。

图 2-5：Gated Timer 框图



用 timer_clk 作为 CLK，EXTIN 作为使能信号。

表 2-3：代码路径

MCU 产品类型	代码路径
SPC168 系列	0_Examples\GTimer_Extren_Gpio_Input_Enable
SPC2168 系列	0_Examples\GTimer_Extren_Gpio_Input_Enable
SPC1125 系列	0_Examples\GTimer_Extren_Gpio_Input_Enable

2.3.5 外部时钟模式

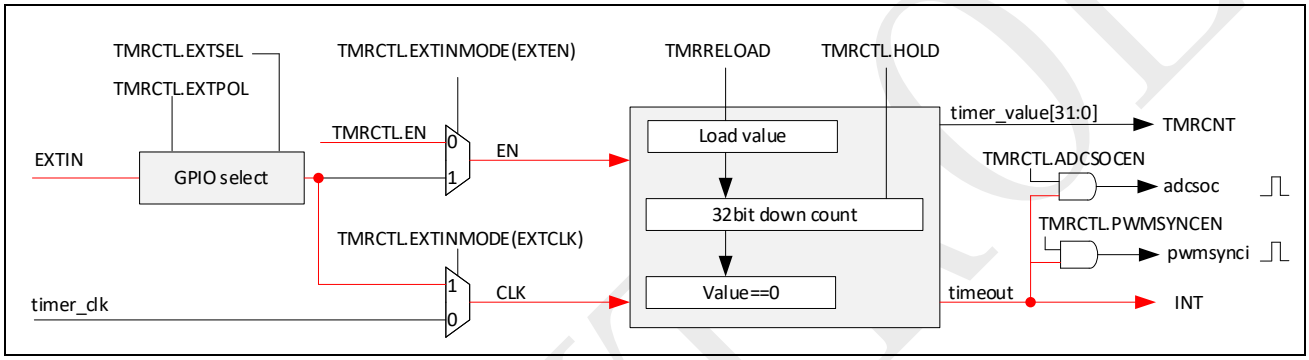
2.3.5.1 功能需求

通过外部引脚提供定时器时钟，并在计数到 0 后触发中断。

2.3.5.2 功能实现

TMRCTL.EXTINMODE 设为 EXTCLK，该模式与通用定时模式类似，只是计数时钟改为外部 EXTIN。

图 2-6：EXTCLK 框图



用 EXTIN 作为 CLK，TMCTL.EN 作为使能信号。

表 2-4：代码路径

MCU 产品类型	代码路径
SPC168 系列	0_Examples\GTimer_Extren_Gpio_Input_Clk
SPC2168 系列	0_Examples\GTimer_Extren_Gpio_Input_Clk
SPC1125 系列	0_Examples\GTimer_Extren_Gpio_Input_Clk