Timer, Counter, and PWM (TCPWM) usage in XMC7000 family

About this document

Scope and purpose

This application note describes how to use Timer, Counter, and Pulse Width Modulator (TCPWM) in XMC7000 family MCUs. The TCPWM is a multifunctional timer component that supports several functional modes. The application note explains how to configure TCPWM.

Intended audience

This document is intended for anyone who uses the TCPWM function of the XMC7000 family.

Associated part family

XMC7000 family of XMC™ industrial microcontrollers

Table of contents

About this document **1**
Table of contents **2**
1 Introduction **3**
1.1 Features 3
1.2 Block diagram 4
2 TCPWM operation examples **7**
2.1 Timer mode 7
2.1.1 Use case 8
2.1.2 Configuration and example 10
2.2 Capture mode 12
2.2.1 Use case 14
2.2.2 Configuration and example 16
2.3 PWM mode 18
2.3.1 Use case 19
2.3.2 Configuration and example 21
2.4 PWM Dead Time (PWM_DT) mode 22
2.4.1 Use case 23
2.4.2 Configuration and example 25
3 Relation of trigger multiplexer **27**
3.1 Starting three TCPWMs simultaneously 27
3.1.1 Use case 27
3.1.2 Configuration and example 29
3.2 Starting AD conversion with TCPWM output 32
3.2.1 Use case 32
3.2.2 Configuration and example 33
References **37**
Glossary **38**
Revision history **39**
Disclaimer **40**

1 Introduction

This application note describes how to use TCPWM of XMC7000 family MCUs in ModusToolbox™. TCPWM is a multifunctional counter component, which supports several functional modes. TCPWM counter width is 16-bit or 32-bit. In addition, 16-bit counters support special functions optimized for motor control.

See the Device datasheet for the number of TCPWM channels available for each device.

This application note explains the functioning of TCPWM in the XMC7000 series, initial configuration, and use cases in which the functional modes are based on the PDL and the device configurator tool.

To understand the functionality described and the terminology used in this application note, see the “Timer, Counter, and PWM” chapter of the Architecture technical reference manual (TRM).

1.1 Features

Table 1 shows the TCPWM function modes.

Table 1 TCPWM function modes

Mode	Description
Timer	Counter increments or decrements by every counter clock cycle in which a count event is detected
Capture	Counter increments or decrements by every counter clock cycle in which a count event is detected. A capture event copies the counter value into the capture register.
QUAD	Quadrature decoding. Counter is decremented or incremented based on two phases according to X1, X2, X4, or up/down rotary encoding scheme. Quadrature mode will have four sub-modes to move the counter between 0 and PERIOD or between 0x8000 and 0x0000/0xffff in combination with compare or capture functionality.
PWM	Pulse width modulation with clock pre-scaling
PWM_DT	Pulse width modulation with dead time
PWM_PR	Pseudo-random PWM using 16- or 32-bit Linear Feedback Shift Register (LFSR) with programmable length to generate pseudo-random noise
SR	Shift Register functionality shifts the counter value in the right direction. The capture0 input is used to generate the MSB of the next counter value. The line output signal is driven from a programmable tab of the shift register (counter).

Each counter supports multiple function modes. At any time, a single counter is operating in one mode and different counters are operating in different modes.

See the “Timer, Counter, and PWM” chapter of the Architecture TRM for more details.

1.2 Block diagram

Figure 1 shows a simplified TCPWM block diagram.

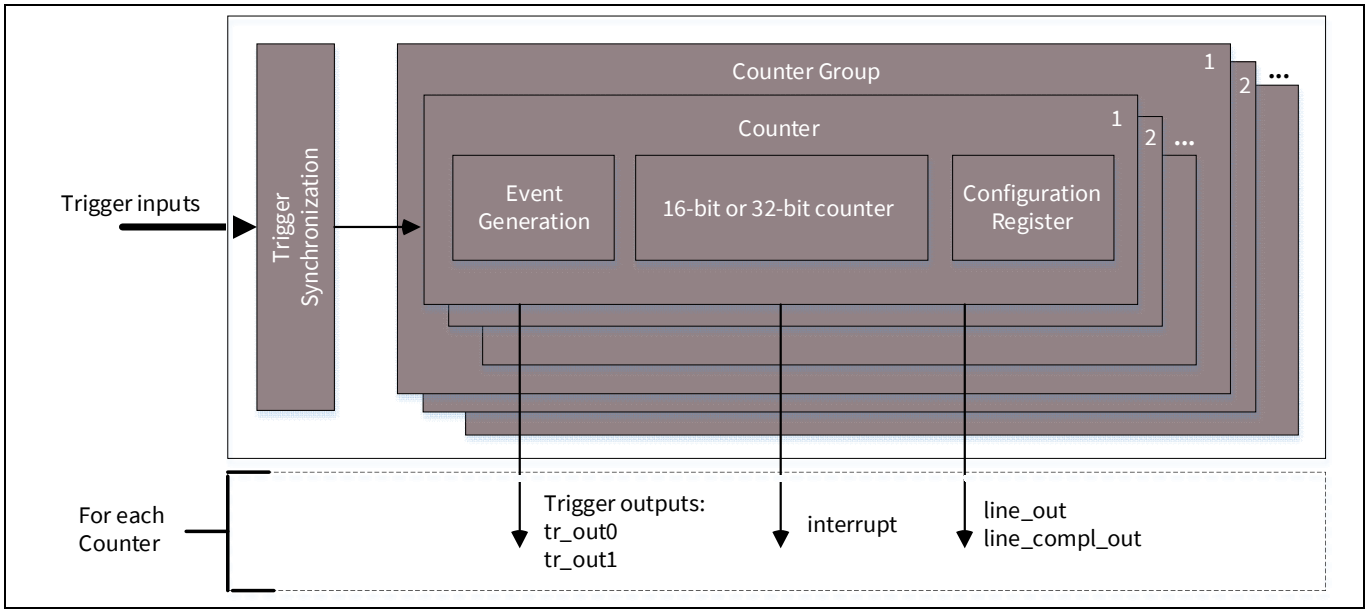


Figure 1 TCPWM block diagram

TCPWM consists of Trigger Synchronization and Counter Group. Each Counter Group consists of counters, and each counter consists of Event Generation, a 16-bit or 32-bit counter, and a Configuration Register.

Each counter has two trigger outputs (tr_out0, tr_out1), two lines output (line_out, line_compl_out), and one interrupt.

The 16-bit counter has an additional option for motor control. This counter has functions that are optimized for motor control operations.

Event Generation generates counter events for 16-bit or 32-bit counters as Reload, Start, Stop, Count, and Capture events. Those events can relate to Trigger inputs.

The trigger input is synchronized by the Trigger Synchronization block and input to the Counter block. Several trigger inputs are connected to TCPWM. Those trigger inputs are GPIO ports, SAR ADC Range violation detected, constant 0 and 1, and general triggers output from the trigger multiplexer. See the "Trigger Multiplexer" chapter of the Architecture TRM for more details.

Introduction

Figure 2 shows a TCPWM and clock supplied block diagram.

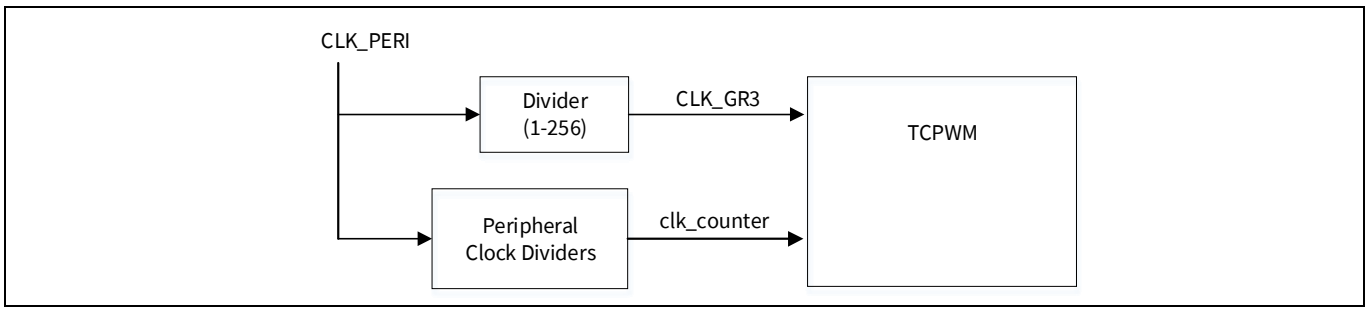


Figure 2 XMC7200 series: TCPWM and clock

The system clock for TCPWM is in group 3, which is supplied from CLK_PERI through the Divider to CLK_GR3. This clock is used in Trigger Synchronization.

Counter clock for each counter in TCPWM is supplied from CLK_PERI through the Peripheral Clock Dividers to clk_counter.

Before enabling the counter, a clock should be selected for the counter. This clock is generated by the peripheral clock dividers. Figure 3 shows the peripheral clock dividers block diagram.

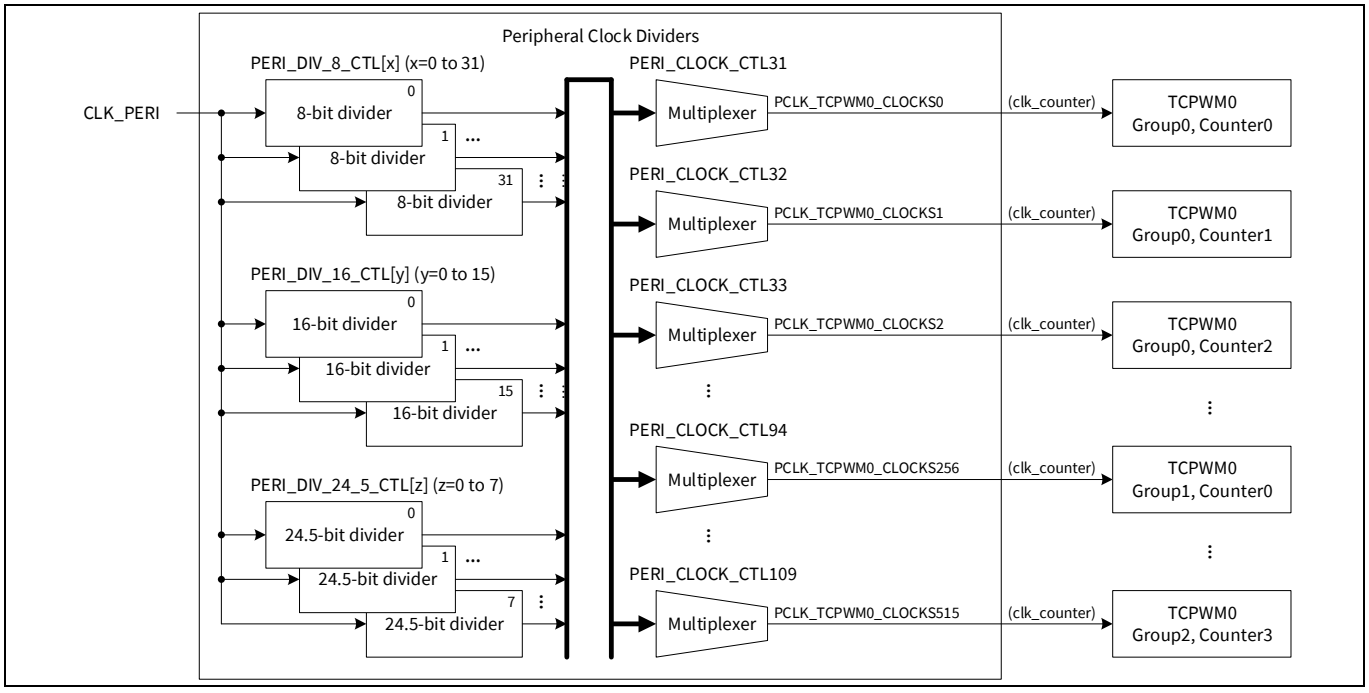


Figure 3 XMC7200: Peripheral clock dividers

Peripheral clock dividers include three types of dividers: 8-bit divider (8.0 divider), 16-bit divider (16.0 divider), and 24.5-bit divider (24.5 divider). See the Device datasheet for the number of each divider channel available for each device.

Each divider makes a clock to divide clock CLK_PERI. The 8-bit divider can divide CLK_PERI by 1 to 2^8 and the 16-bit divider can divide CLK_PERI by 1 to 2^{16}. The 24.5-bit divider is a divider that has 24 integer bits and 5 fractional bits. This divider can divide CLK_PERI by 1 to 2^{24} for integer and 1 to 2^5 for fractional.

Introduction

The output of the peripheral clock divider, clk_counter, is included in the peripheral clocks. The clk_counter is represented as PCLK_TCPWM[m]_CLOCKS[n] in this application note and the device datasheet (m = implemented module number, n = peripheral clock number). Peripheral clocks are connected to each peripheral module on one-to-one connections and have unique numbers. Table 2 shows the peripheral clock number connected to the TCPWM of XMC7200 MCUs. For other series, see the “Peripheral clocks” section in the Device datasheet.

Table 2 XMC7200 series: Peripheral clock number in TCPWM

Peripheral clock number	Description
`PCLK_TCPWM0_CLOCKS0 to S2`	TCPWM0 group #0, counter #0 to #2 (3 ch)
`PCLK_TCPWM0_CLOCKS256 to S258`	TCPWM0 group #1, counter #0 to #2 (3 ch)
`PCLK_TCPWM0_CLOCKS512 to S514`	TCPWM0 group #2, counter #0 to #2 (3 ch)
`PCLK_TCPWM1_CLOCKS0 to S83`	TCPWM1 group #0, counter #0 to #83 (84 ch)
`PCLK_TCPWM0_CLOCKS256 to S267`	TCPWM1 group #1, counter #0 to #11 (12 ch)
`PCLK_TCPWM0_CLOCKS512 to S524`	TCPWM1 group #2, counter #0 to #12 (13 ch)

See the “Clocking system” chapter of the Architecture TRM for more details.

2 TCPWM operation examples

This section describes how to use TCPWM using the Peripheral Driver Library (PDL) combined with the device configurator tool. The code snippets in this application note are part of PDL. Contact Infineon support for the PDL.

The PDL provides the low-level drivers for Infineon devices. The PDL integrates device header files, startup code, and peripheral drivers into a single package. The drivers abstract the hardware functions into a set of easy-to-use APIs.

The device configurator tool is part of a collection of tools included with the ModusToolbox™ software. Use this tool to enable and configure device peripherals, such as clocks and pins, as well as standard MCU peripherals that do not require their own tool.

2.1 Timer mode

This section describes how to set up the Timer mode. Timer mode is used for a basic counter application. This is an ordinary counter used to count the clock for the timer.

The following are the different modes of counters based on the direction:

- COUNT_UP: Counting mode in the upward direction
- COUNT_DOWN: Counting mode in the downward direction
- UPDOWN-COUNTER1 and UPDOWN-COUNTER2: Counting modes in the upward and downward directions

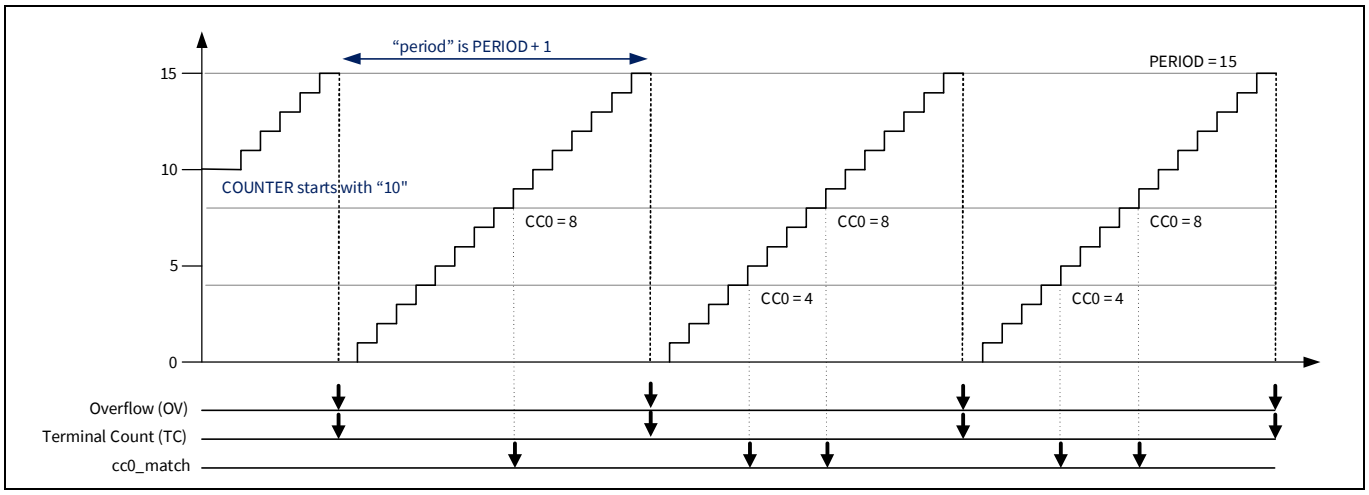


Figure 4 Timer mode in upward counting mode

The counter starts from an initial value. If configured counter register (e.g., `TCPWM0_GRP0_CNT0_CONTER`) as COUNTER = 10, the counter starts with 10.

The counter generates events depending on the counter value. There are five events: Underflow (UV), Overflow (OV), Terminal Count (TC), cc0_match, and cc1_match. The event generation depends on the combination of operation mode and `UP_DOWN_MODE`. Underflow event is not generated in COUNT_UP.

Overflow event is generated in the counter in which the counter value equals the PERIOD (For example, `TCPWM0_GRP0_CNT0_PERIOD`) register value.

cc0_match event is generated in the counter in which the counter value equals the CC0 register value. CC0 (e.g., `TCPWM0_GRP0_CNT0_CC0`) value can be switched with CC0_BUF (e.g., `TCPWM0_GRP0_CNT0_CC0_BUF`) value at the cc0_match event point. CC0 and CC0_BUF values are configured by the related register bits. Figure 4 shows that the CC0 register value is 8 and the CC0_BUF register value is 4 at the start point, and then the CC0 register value is changed to 4 and the CC0_BUF register value is changed to 8 at the cc0_match point.

Figure 5 shows the timer functionality which includes the events, and interrupts tr_out0 and tr_out1 relationship.

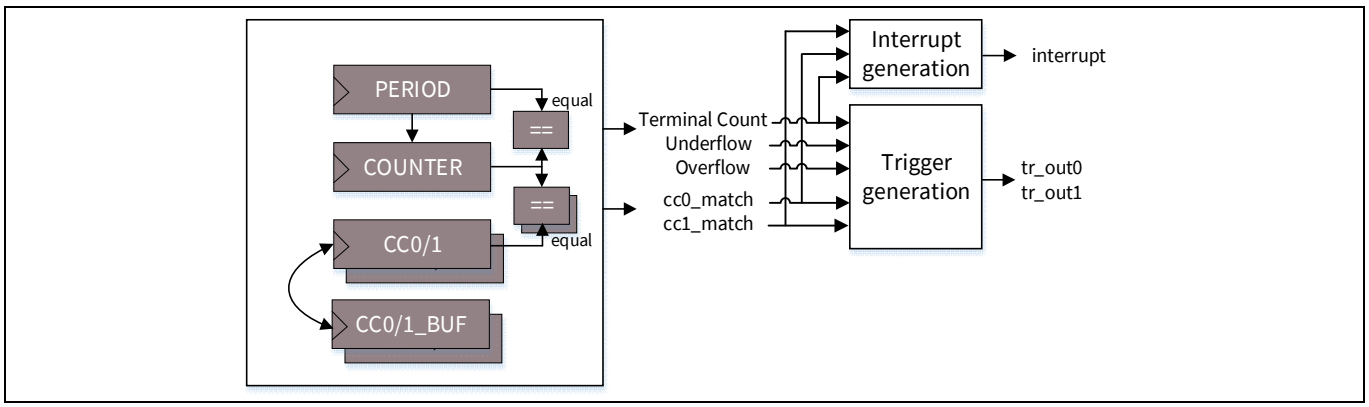


Figure 5 Timer functionality

Every event can be output as trigger tr_out0, tr_out1, or an interrupt from the counter in the TCPWM to other modules.

For example, in the use case of specific interval data translation by P-DMA, a periodic trigger is generated by cc0_match and this cc0_match is used as a trigger to activate P-DMA. This trigger and P-DMA connection are handled by the Triggers Multiplexer module.

2.1.1 Use case

This section describes a use case of Timer mode for generating an interrupt every 1 second of the counter cycle with a 500 kHz counter clock. The following is an example of configuring the TCPWM using PDL:

- TCPWM operation mode: Timer mode
- Using counter: TCPWM0/Group0/16-bit Counter0
- Counter start operation: Start by software
- Input clock
  - CLK_PERI: 50 MHz
  - Divide value: Divided by 100 (50 MHz/100 = 500 kHz)
- Interrupt period: 0.1 s [50000*(1/500 kHz)]
- System interrupt source: TCPWM0/Group0/Counter0 (tcpwm_0_interrupts_0_IRQn IDX: 519)
- Mapped to CPU interrupt: IRQ3 (NvicMux3_IRQn)
- CPU interrupt priority: 3

TCPWM operation examples

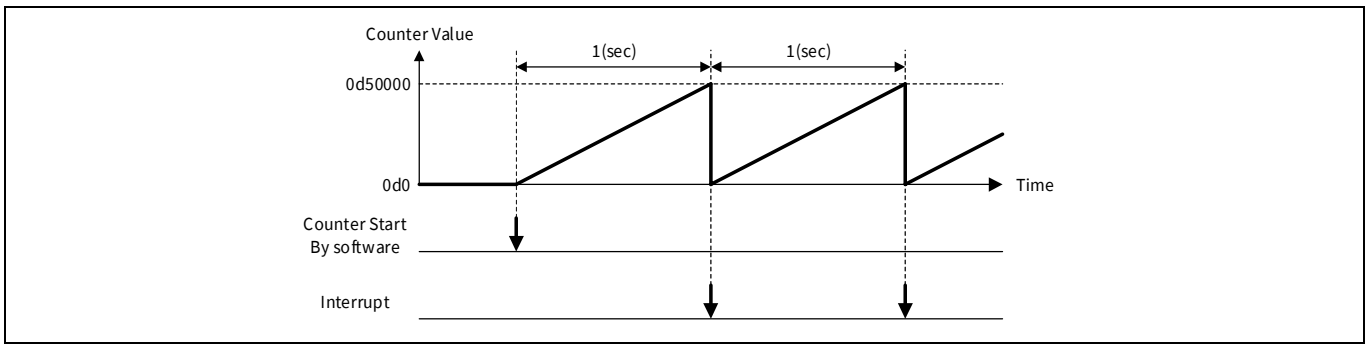


Figure 6 Timing chart of the Timer mode

Figure 7 shows the operation flow of this use case.

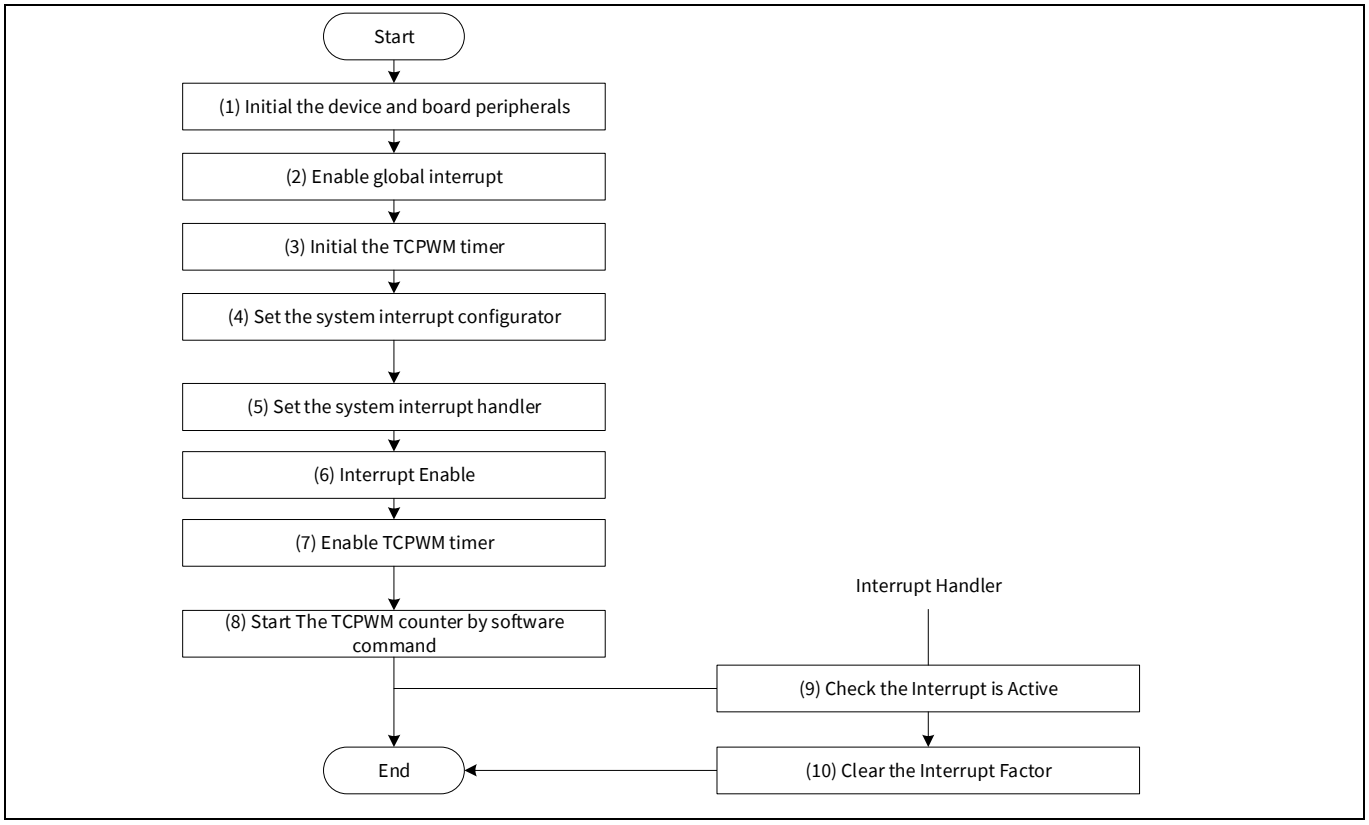


Figure 7 Operation flow example

1. Initialize the device and board peripherals
2. Enable global interrupt (CPU interrupt enable). For more details, see the CPU interrupt handling sections in the Architecture TRM
3. Initialize the peripheral clocks, mode, and parameters for the TCPWM timer
4. Set interrupt structure. For more details, see the CPU interrupt handling sections in the Architecture TRM
5. Set the system interrupt handler. For more details, see the CPU interrupt handling sections in the Architecture TRM
6. Enable the interrupt by the NVIC interrupt controller. For more details, see the CPU interrupt handling sections in the Architecture TRM
7. Enable the TCPWM timer

8. Start the TCPWM timer with a software command

Note: If the TCPWM counter is enabled, disable it to prevent a malfunction.

9. When an interrupt occurs, check the interrupt is active
10. After executing the interrupt process, the interrupt factor is cleared

2.1.2 Configuration and example

Figure 8 shows the parameters of the configuration part in device configurator tool for TCPWM Timer mode.

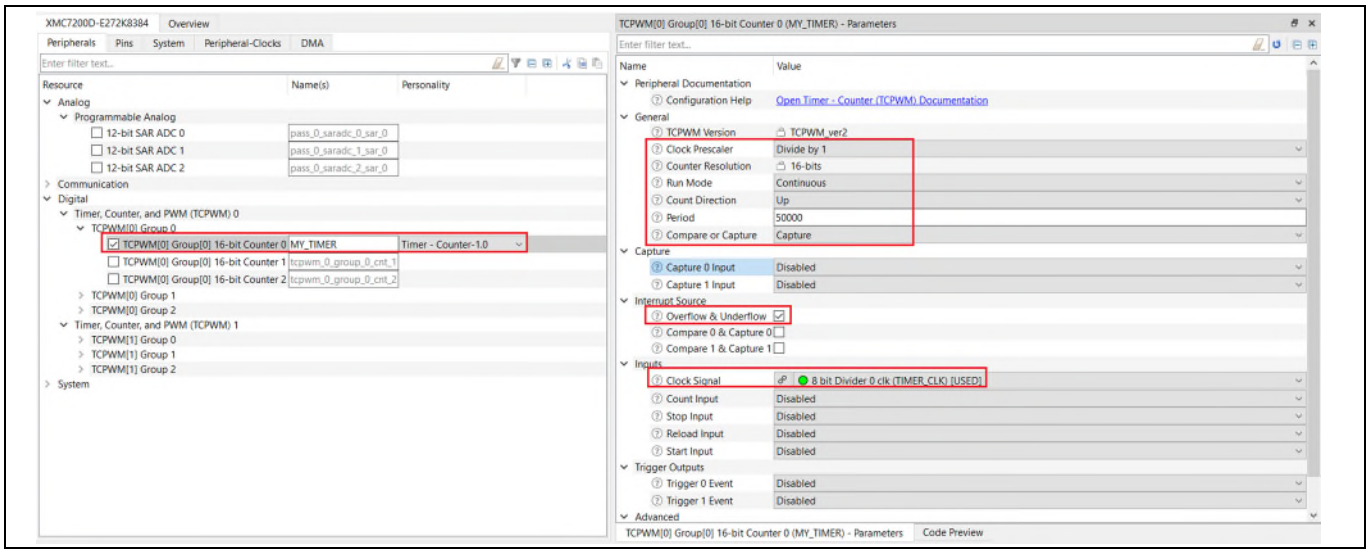


Figure 8 TCPWM timer configuration in device configurator tool

Figure 9 shows the clock parameters configuration in device configurator tool for TCPWM timer.

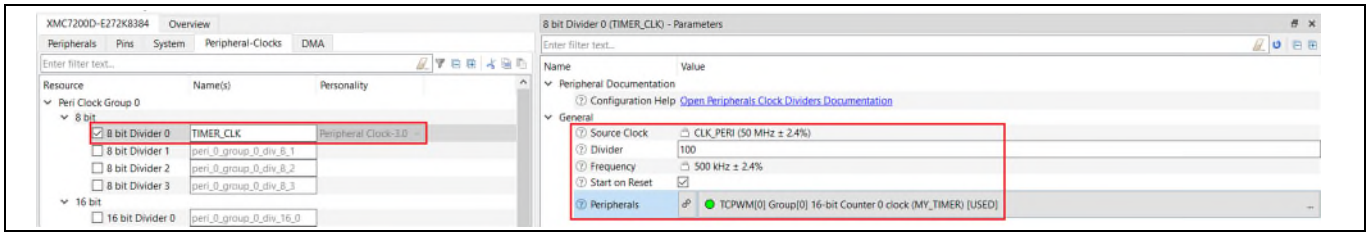


Figure 9 TCPWM timer clock configuration in device configurator tool

Code Listing 1 demonstrates the TCPWM timer in all configuration parameters with PDL codes.

Code Listing 1 XMC7200: Example to configure Timer mode in configuration part

```
const cy_stc_tcpwm_counter_config_t MY_TIMER_config =

{
    .period = 50000,
    .clockPrescaler = CY_TCPWM_COUNTER_PRESCALER_DIVBY_1,
    .runMode = CY_TCPWM_COUNTER_CONTINUOUS,
    .countDirection = CY_TCPWM_COUNTER_COUNT_UP,
    .compareOrCapture = CY_TCPWM_COUNTER_MODE_CAPTURE,
    .compare0 = 16384,
    .compare1 = 16384,
    .enableCompareSwap = false,
    .interruptSources = (CY_TCPWM_INT_ON_TC ) | (CY_TCPWM_INT_ON_CC0 & 0U) | (CY_TCPWM_INT_ON_CC1 &
```

TCPWM operation examples

Code Listing 1 XMC7200: Example to configure Timer mode in configuration part

```
0U),
    .captureInputMode = MY_TIMER_INPUT_DISABLED & 0x3U,
    .captureInput = CY_TCPWM_INPUT_0,
    .reloadInputMode = MY_TIMER_INPUT_DISABLED & 0x3U,
    .reloadInput = CY_TCPWM_INPUT_0,
    .startInputMode = MY_TIMER_INPUT_DISABLED & 0x3U,
    .startInput = CY_TCPWM_INPUT_0,
    .stopInputMode = MY_TIMER_INPUT_DISABLED & 0x3U,
    .stopInput = CY_TCPWM_INPUT_0,
    .countInputMode = MY_TIMER_INPUT_DISABLED & 0x3U,
    .countInput = CY_TCPWM_INPUT_1,
    .capture1InputMode = MY_TIMER_INPUT_DISABLED & 0x3U,
    .capture1Input = CY_TCPWM_INPUT_0,
    .compare2 = 16384,
    .compare3 = 16384,
    .enableCompare1Swap = false,
    .trigger0Event = CY_TCPWM_CNT_TRIGGER_ON_DISABLED,
    .trigger1Event = CY_TCPWM_CNT_TRIGGER_ON_DISABLED,

};
```

Code Listing 2 demonstrates an example of the timer interrupt setting and program TCPWM timer.

Code Listing 2 XMC7200: Example to program TCPWM timer with interrupt function

```
/*******************************************************************************
 * Macros for both examples
 ******************************************************************************/
#define TIMER_INT_PRIORITY     3U

/*******************************************************************************
* Function Prototypes
*******************************************************************************/
void Timer_Handler(void);

/*******************************************************************************
* Global Variables
*******************************************************************************/
cy_stc_sysint_t intrCfg =
{
    .intrSrc = ((NvicMux3_IRQn << 16) | MY_TIMER_IRQ),     /* Interrupt source is
tcpwm_0_interrupts_0_IRQn */
    .intrPriority = TIMER_INT_PRIORITY                      /* Interrupt priority is 3 */

};


int main(void)
{
    cy_rslt_t result;

    /* Initialize the device and board peripherals */
    result = cybsp_init() ;
    if (result != CY_RSLT_SUCCESS)
    {
        CY_ASSERT(0);
    }


     __enable_irq();


   /*TCPWM Timer initialization*/
    if(Cy_TCPWM_Counter_Init(MY_TIMER_HW, MY_TIMER_NUM, &MY_TIMER_config) != CY_TCPWM_SUCCESS)
    {
       CY_ASSERT(0);
    }

    /*Configure Timer interrupt and enable it*/
    Cy_SysInt_Init(&intrCfg, Timer_Handler);
    NVIC_ClearPendingIRQ(intrCfg.intrSrc);
    NVIC_EnableIRQ((IRQn_Type)NvicMux3_IRQn);
```

Code Listing 2 XMC7200: Example to program TCPWM timer with interrupt function

```
/*TCPWM Timer enable*/
Cy_TCPWM_Counter_Enable(MY_TIMER_HW, MY_TIMER_NUM);

/*Start the TCPWM Timer*/
Cy_TCPWM_TriggerStart_Single(MY_TIMER_HW,MY_TIMER_NUM);

for (;;)
{

}
```

Code Listing 3 demonstrates the timer TC interrupt handler.

Code Listing 3 XMC7200: Timer TC interrupt handler

```
/*******************************************************************************
* Function Name: Timer_Handler
********************************************************************************
* Summary:
* Return: void
*
*******************************************************************************/
void Timer_Handler(void)
{
  Cy_TCPWM_ClearInterrupt(MY_TIMER_HW,MY_TIMER_NUM, CY_TCPWM_INT_ON_TC);


}
```

2.2 Capture mode

This section describes how to set up the Capture mode. The Capture mode is for an application to catch the counter value depending on the input trigger. Figure 10 shows the Capture mode in upward counting mode.

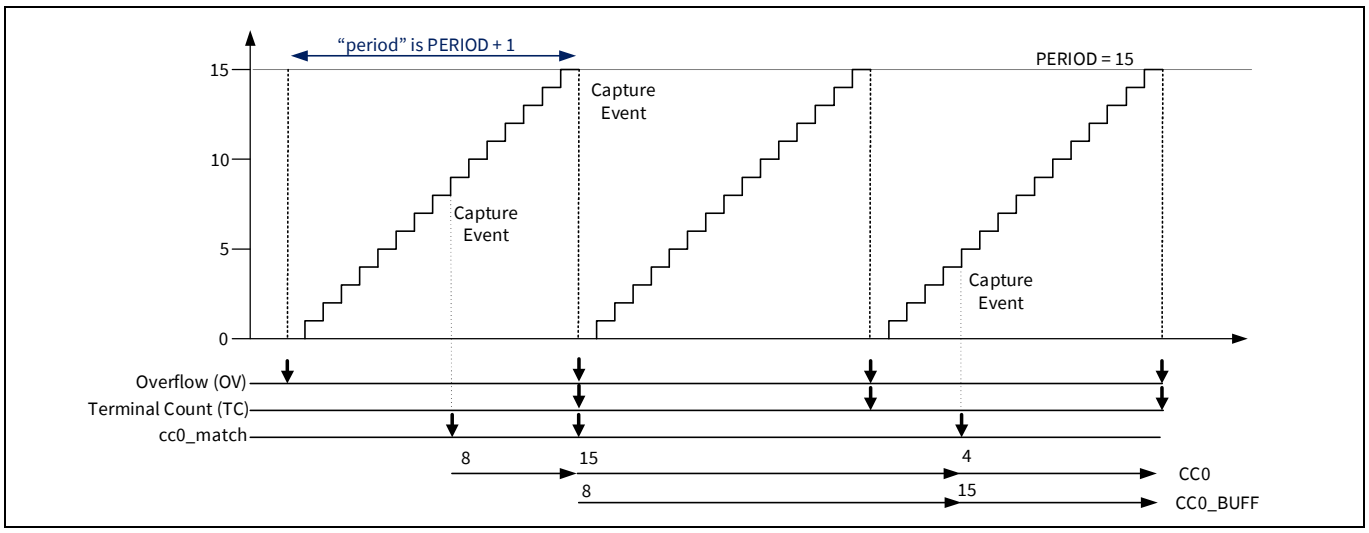


Figure 10 Capture mode in upward counting mode

When the trigger input is detected, the capture event occurs, and the counter value is captured in the CC0 register. Also, the cc0_match event is generated.

TCPWM operation examples

When the next cc0_match event occurs, the CC0 register value is copied to the CC0_BUFF register and the counter value is captured in the CC0 register.

The counter in TCPWM can select the input trigger from the input trigger sources. See the Device datasheet for the number of each counter channel available for each device.

Table 3 shows the input trigger sources of the 16-bit counter number 0 in the XMC7200 series.

Table 3 XMC7200: Trigger inputs list of 16-bit counter number 0

Trigger no.	Input trigger	Input trigger sources
0	Constant '0'	Constant '0'
1	Constant 1	Constant '1'
2	HSIOM column ACT#2	See the "Alternate pin function" section in the device datasheet
3	HSIOM column ACT#3	See the "Alternate pin function" section in the device datasheet
4	PASS (programmable analog subsystem), through 1:1 trigger mux #0	See the product sheet tab "triggersOnetoOne". Not all counters will have this input trigger
5	tr_all_cnt_in[0]	See the "Trigger Multiplexer" block in the Architecture TRM
:	:	
31	tr_all_cnt_in[26]	See the "Trigger Multiplexer" block in the Architecture TRM

Note: These are some excerpts from the TRM. See the "Timer, Counter, and PWM" chapter of the Architecture TRM for more details.

TCPWM can configure the input trigger as several events. Capture mode can use six events: reload, start, stop, count, capture0, and capture1.

2.2.1 Use case

This section describes a use case of Capture mode when an input trigger from an I/O port is used as the capture0/1 event. Interrupts are generated at rising and falling edges at external pins. The following is an example of configuring the TCPWM using PDL:

- TCPWM operation mode: Capture mode
- Using counter: TCPWM0/Group0/Counter0
- Counter start operation: Start by software
- Input clock
  - CLK_PERI: 50 MHz
  - Divide value: Divided by 100 (50 MHz/100 = 500 kHz)
- Used TCPWM TC signal as capture0 or capture1 event
- Interrupt: When the event of capture0 or capture1 occurs
- System interrupt source: TCPWM0/Group0/Counter1 (tcpwm_0_interrupts_1_IRQn IDX: 520)
- Mapped to CPU interrupt: IRQ3
- CPU interrupt priority: 3

The details of the external pins are not described here. For more information, see the “I/O System” and “Trigger Multiplexer” chapters in the Architecture TRM.

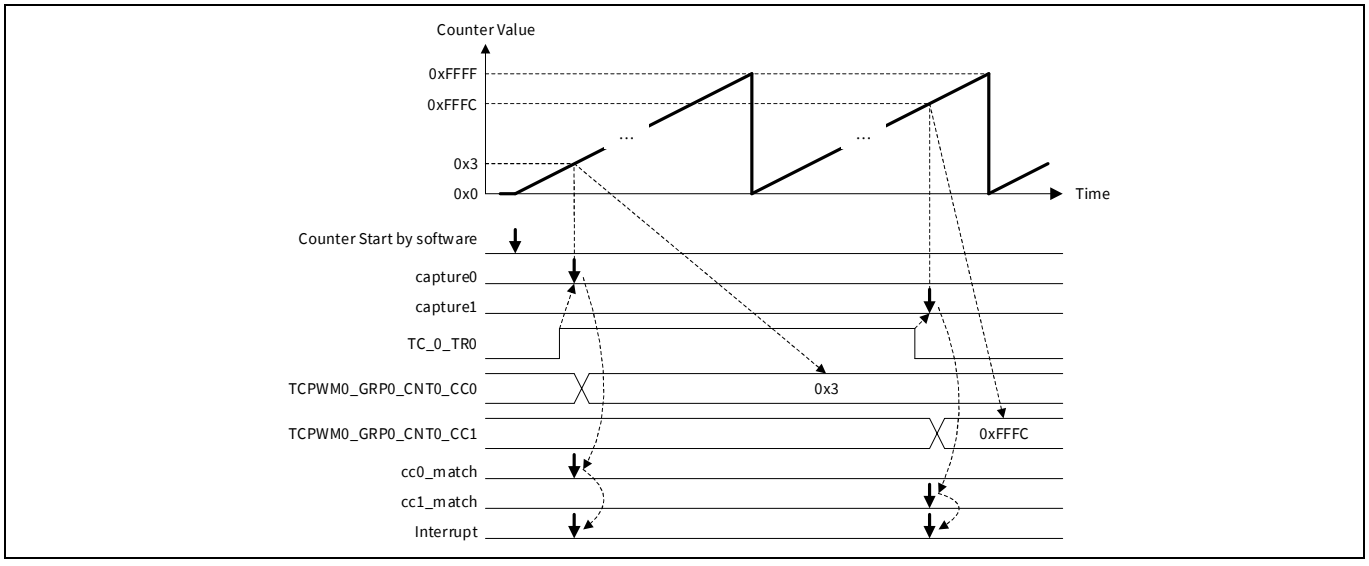


Figure 11 Timing chart of the Capture mode

Note: The capture0 or capture1 signal is generated by the input of TC_0_TR0. The input from TC0_0_TR0 is not always the counter value in the figure.

Figure 12 shows the operation flow of this use case.

TCPWM operation examples

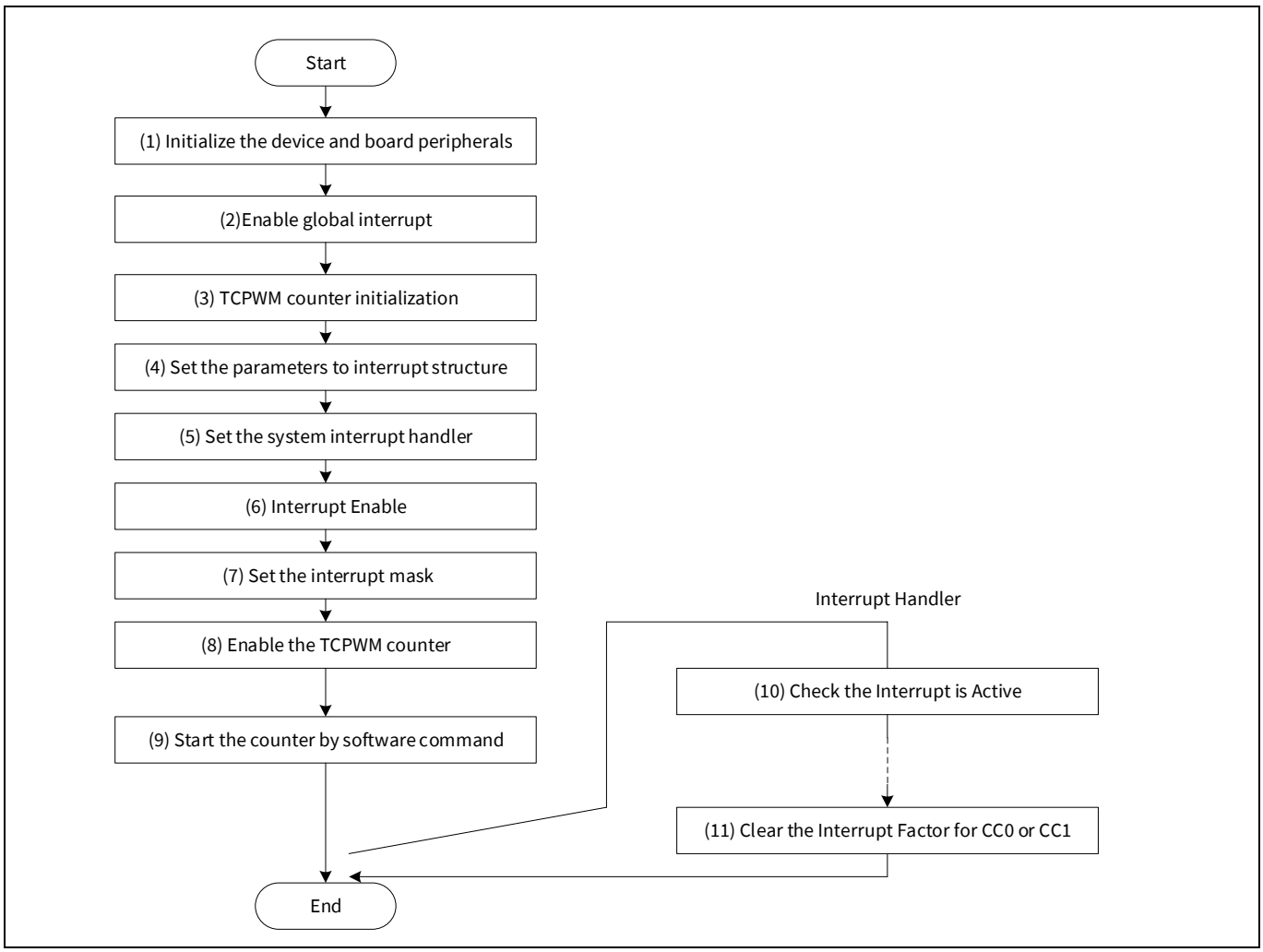


Figure 12 Operation flow example

1. Initialize the device and board peripherals
2. Enable global interrupt (CPU interrupt enable). For more details, see the CPU interrupt handling sections in the Architecture TRM.
3. Initializes the counter in the TCPWM block for the counter operation
4. Set interrupt structure. For more details, see the CPU interrupt handling sections in the Architecture TRM
5. Set the system interrupt handler. For more details, see the CPU interrupt handing sections in the Architecture TRM
6. Enable the interrupt by the NVIC interrupt controller. For more details, see the CPU interrupt handling sections in the Architecture TRM
7. Set the capture0 or capture1 interrupt mask value
8. Enable the TCPWM counter

Note: If the TCPWM counter is enabled, disable it to prevent a malfunction.

9. Start the TCPWM counter with a software command
10. When an interrupt occurs, check the Interrupt is active
11. After executing the interrupt process, the interrupt factor (CC0 or CC1) is cleared

2.2.2 Configuration and example

Figure 13 shows the parameters of the configuration part in device configurator tool for TCPWM Capture mode.

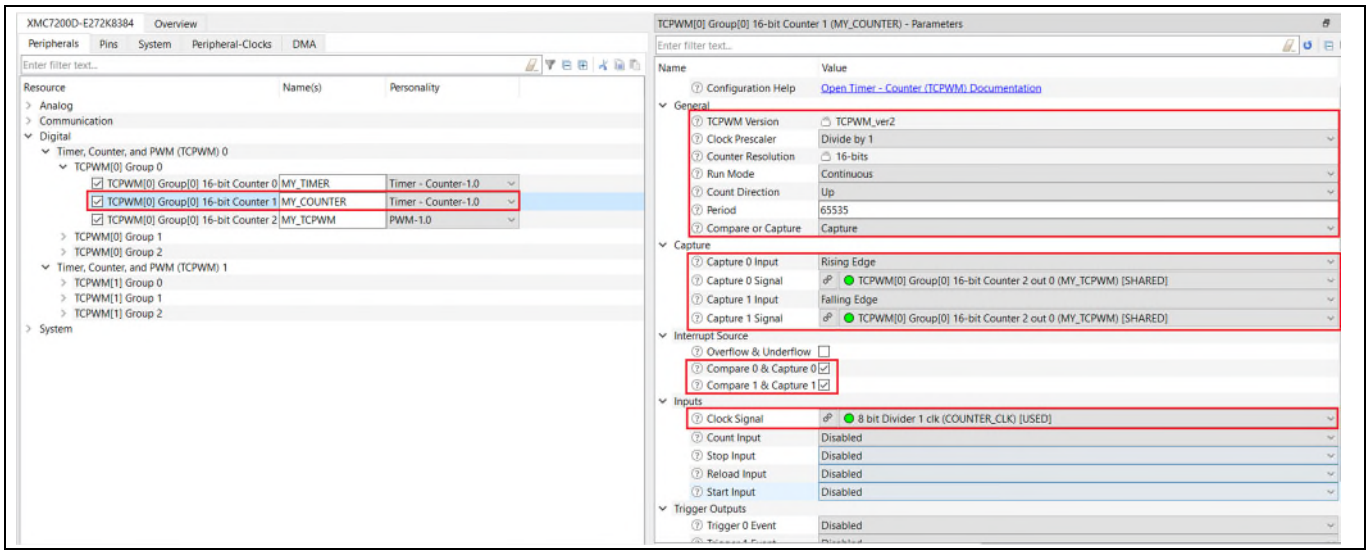


Figure 13 TCPWM counter configuration in device configurator tool

Figure 14 shows the clock parameters configuration in device configurator tool for TCPWM counter.

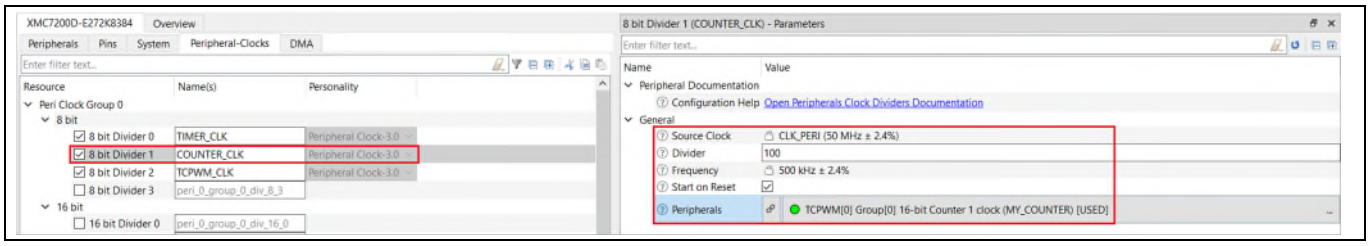


Figure 14 TCPWM counter clock configuration in device configurator tool

Code Listing 4 demonstrates the TCPWM counter in all configuration parameters with PDL codes.

Code Listing 4 XMC7200: Example to configure Capture mode in configuration part

```
const cy_stc_tcpwm_counter_config_t MY_COUNTER_config =


{
    .period = 65535,
    .clockPrescaler = CY_TCPWM_COUNTER_PRESCALER_DIVBY_1,
    .runMode = CY_TCPWM_COUNTER_CONTINUOUS,
    .countDirection = CY_TCPWM_COUNTER_COUNT_UP,
    .compareOrCapture = CY_TCPWM_COUNTER_MODE_CAPTURE,
    .compare0 = 16384,
    .compare1 = 16384,
    .enableCompareSwap = false,
    .interruptSources = (CY_TCPWM_INT_ON_TC & 0U) | (CY_TCPWM_INT_ON_CC0 ) | (CY_TCPWM_INT_ON_CC1 ),
    .captureInputMode = CY_TCPWM_INPUT_RISINGEDGE,
    .captureInput = TCPWM0_GRP0_CNT1_CAPTURE0_VALUE,
    .reloadInputMode = MY_COUNTER_INPUT_DISABLED & 0x3U,
    .reloadInput = CY_TCPWM_INPUT_0,
    .startInputMode = MY_COUNTER_INPUT_DISABLED & 0x3U,
    .startInput = CY_TCPWM_INPUT_0,
    .stopInputMode = MY_COUNTER_INPUT_DISABLED & 0x3U,
    .stopInput = CY_TCPWM_INPUT_0,
    .countInputMode = MY_COUNTER_INPUT_DISABLED & 0x3U,
    .countInput = CY_TCPWM_INPUT_1,
```

TCPWM operation examples

Code Listing 4 XMC7200: Example to configure Capture mode in configuration part

```
    .capture1InputMode = CY_TCPWM_INPUT_FALLINGEDGE,
    .capture1Input = TCPWM0_GRP0_CNT1_CAPTURE1_VALUE,
    .compare2 = 16384,
    .compare3 = 16384,
    .enableCompare1Swap = false,
    .trigger0Event = CY_TCPWM_CNT_TRIGGER_ON_DISABLED,
    .trigger1Event = CY_TCPWM_CNT_TRIGGER_ON_DISABLED,
};
```

Code Listing 5 demonstrates an example of the capture0 interrupt setting and program TCPWM counter.

Code Listing 5 Example of interrupt handler

```
#define CAPTURE_INT_PRIORITY     3U

void Capture_Handler(void);

cy_stc_sysint_t intrCfg =
{
    /* Interrupt source is tcpwm_0_interrupts_1_IRQn */
    .intrSrc = ((NvicMux3_IRQn << 16) | MY_COUNTER_IRQ),
    .intrPriority = CAPTURE_INT_PRIORITY                  /* Interrupt priority is 3 */
};

int main(void)
{
    cy_rslt_t result;

    /* Initialize the device and board peripherals */
    result = cybsp_init() ;
    if (result != CY_RSLT_SUCCESS)
    {
        CY_ASSERT(0);
    }


     __enable_irq();

     :


   /*TCPWM Counter Mode initial*/
   if (CY_TCPWM_SUCCESS != Cy_TCPWM_Counter_Init(MY_COUNTER_HW, MY_COUNTER_NUM,&MY_COUNTER_config))
         {
             CY_ASSERT(0);
         }

    /*Configure Capture interrupt and enable it*/
    Cy_SysInt_Init(&intrCfg, Capture_Handler);
    NVIC_ClearPendingIRQ(intrCfg.intrSrc);
    NVIC_EnableIRQ((IRQn_Type)NvicMux3_IRQn);

     /*Set the capture0 interrupt mask value*/
    Cy_TCPWM_SetInterruptMask(MY_COUNTER_HW,MY_COUNTER_NUM,CY_TCPWM_INT_ON_CC0);


   /* Enable the TCPWM counter */
    Cy_TCPWM_Counter_Enable(MY_COUNTER_HW, MY_COUNTER_NUM);

    /*Start the TCPWM counter by software command*/
    Cy_TCPWM_TriggerStart_Single(MY_COUNTER_HW,MY_COUNTER_NUM);

    :

    for(;;)
    {

    }
```

*1: For more details, see the CPU interrupt handling sections in the Architecture TRM.

TCPWM operation examples

Code Listing 6 demonstrates an example of the interrupt handler.

Code Listing 6 Example of interrupt handler

```
void Capture_Handler (void)
{
    :
    if(Cy_TCPWM_GetInterruptMask (MY_COUNTER_HW, MY_COUNTER_NUM) == CY_TCPWM_INT_ON_CC0))
    {
        /* CCO would capture rising edge of input pulse */
        Cy_TCPWM_ClearInterrupt(MY_COUNTER_HW, MY_COUNTER_NUM, CY_TCPWM_INT_ON_CC0);
    }
    :
    if((Cy_TCPWM_GetInterruptMask (MY_COUNTER_HW, MY_COUNTER_NUM) == CY_TCPWM_INT_ON_CC1))
    {
        /* CC1 would capture falling edge of input pulse */
        Cy_TCPWM_ClearInterrupt(MY_COUNTER_HW, MY_COUNTER_NUM, CY_TCPWM_INT_ON_CC1);

    }
}
```

2.3 PWM mode

This section describes how to set up the PWM mode. PWM mode is for an application to output the pulse width modulated signal on the line_out and line_compl_out. Figure 15 shows the PWM mode in upward counting mode.

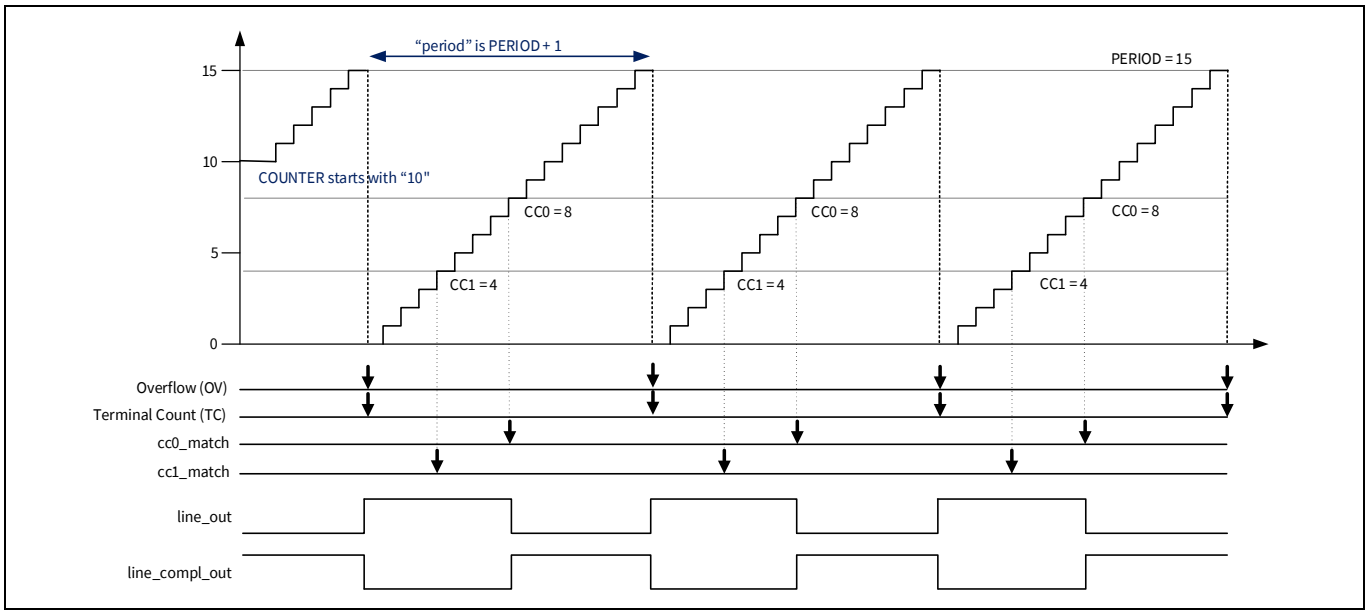


Figure 15 PWM mode in upward counting mode

PWM signal frequency is configured by the PERIOD register. This PWM signal period is the value of the PERIOD register plus 1. PWM duty is configured by CC0 or CC1 (e.g., `TCPWM0_GRP0_CNT0_CC1`) register. cc0_match and cc1_match events in PWM mode occur at the configured COUNTER register value. PWM signal is generated to use Overflow, Underflow, cc0_match, and cc1_match events.

Figure 16 shows the line generation logic. TR_PWM_CTL (e.g., `TCPWM0_GRP0_CNT0_TR_PWM_CTRL`) register controls the line state change based on four events: Underflow, Overflow, cc0_match, and cc1_match.

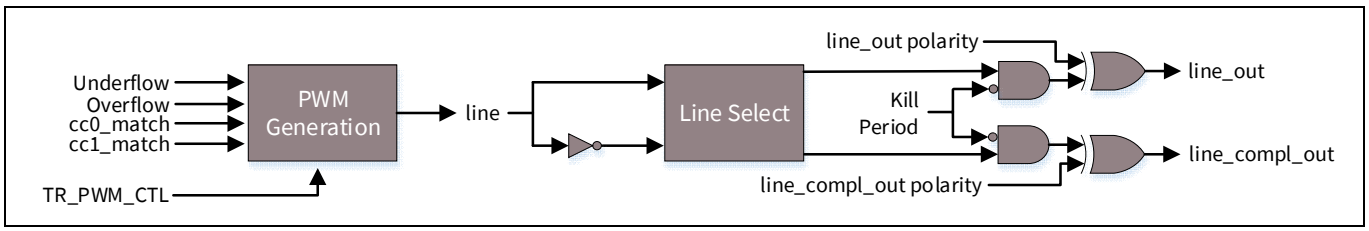


Figure 16 Line generation logic

There are two output lines: a PWM signal is output from line_out, and a complementary PWM signal is output from line_compl_out. Relevant I/O ports configured as PWM output resource, line_out, and line_compl_out are output by PWM and PWM_N ports. PWM and PWM_N ports are assigned to I/O ports P0.0 and P0.1 in the XMC7000 series. See the Device datasheet for more details.

The polarity of both the line_out signals can be configured in the CTRL (e.g., `TCPWM0_GRP0_CNT0_CTRL`) register. The QUAD_ENCODING_MODE[0] bit sets the polarity of line_out; and the QUAD_ENCODING_MODE[1] bit can be used to set the polarity of line_compl_out. The value '1' inverts the corresponding line_out signals.

The Kill period input will disable both line_out and line_compl_out. The Kill mode is specified by the PWM_IMM_KILL, PWM_STOP_ON_KILL, and PWM_SYNC_KILL registers.

Counterpoint is configured by the COUNTER register. In Figure 12, beginning from the "10" counterpoint, configured by the COUNTER register, to the "15" first overflow event period is set as the waiting time.

Four internal events, Underflow, Overflow, cc0_match, and cc1_mach, can be used to output the trigger. In Figure 12, the cc1_match event can be configured with the CC1 register in the flexible point within a time period. This cc1_match event is used to activate the trigger for other modules, such as SAR ADC.

2.3.1 Use case

This section describes a use case of the PWM mode. PWM signal is generated with overflow and cc0_match events. The following is an example of configuring TCPWM using PDL:

- TCPWM operation mode: PWM mode
- Using counter: TCPWM0/Group0/Counter2
- Counter start operation: Start by software
- Input clock
  - CLK_PERI: 50 MHz
  - Divide value: Divided by 100 (50 MHz/100 = 500 kHz)
- PWM duty: 50%

TCPWM operation examples

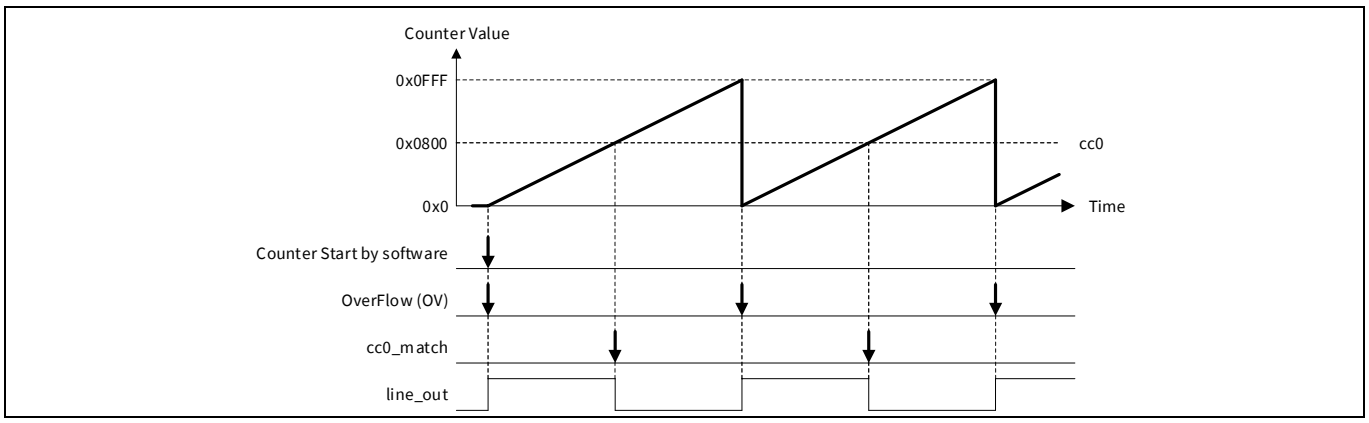


Figure 17 Timing chart of PWM mode

Figure 18 shows the operation flow of this use case.

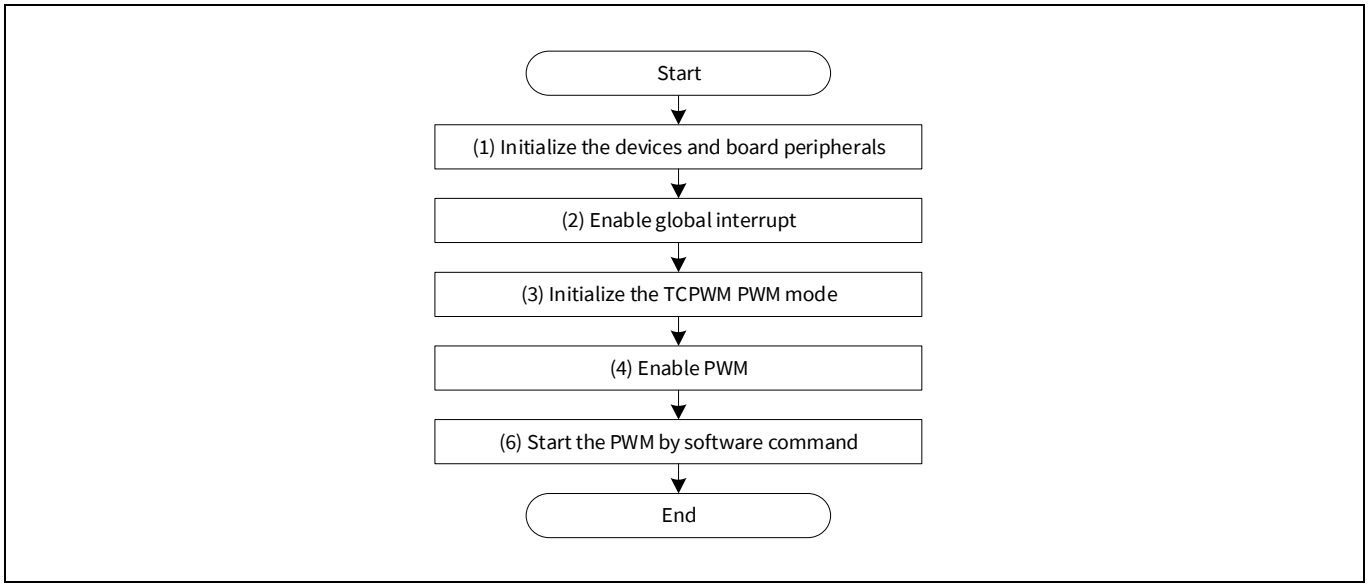


Figure 18 Operation flow example

1. Initialize the device and board peripherals
2. Enable global interrupt (CPU interrupt enable). For more details, see the CPU interrupt handling sections in the Architecture TRM
3. Initialize the TCPWM PWM mode
4. Enable the TCPWM PWM

Note: If the TCPWM counter is enabled, disable it to prevent a malfunction.

5. Start the TCPWM PWM by a software command

2.3.2 Configuration and example

Figure 19 shows the parameters of the configuration part in device configurator tool for TCPWM PWM mode.

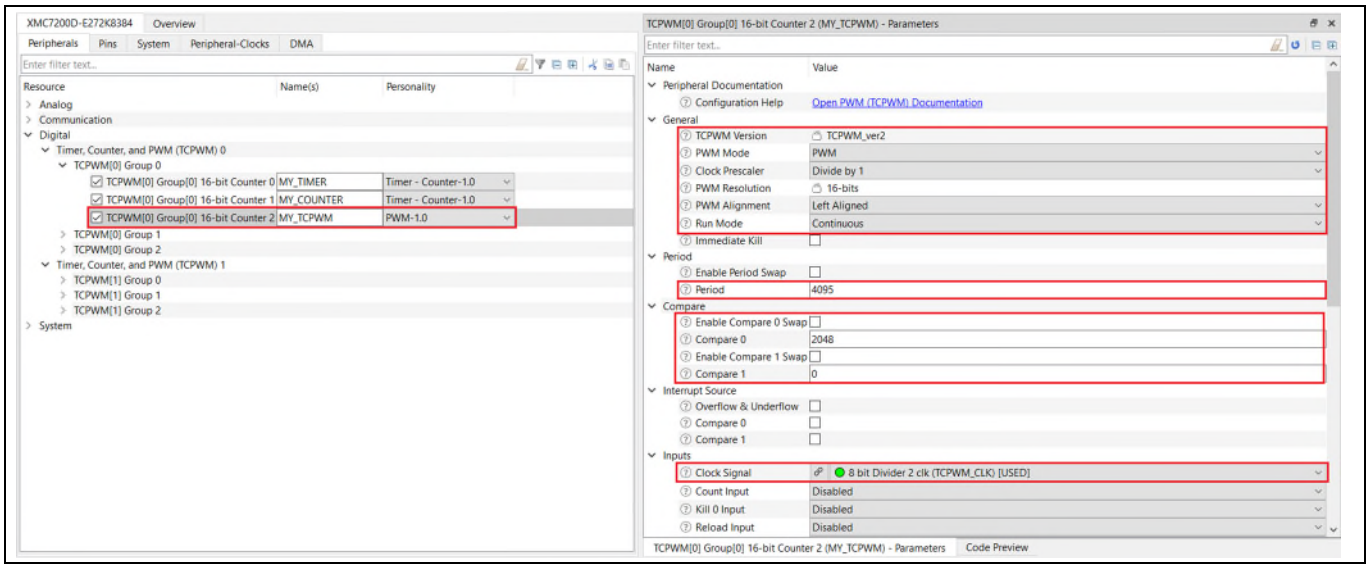


Figure 19 TCPWM PWM configuration in device configurator tool

Figure 20 shows the clock parameters configuration in device configurator tool for TCPWM PWM.

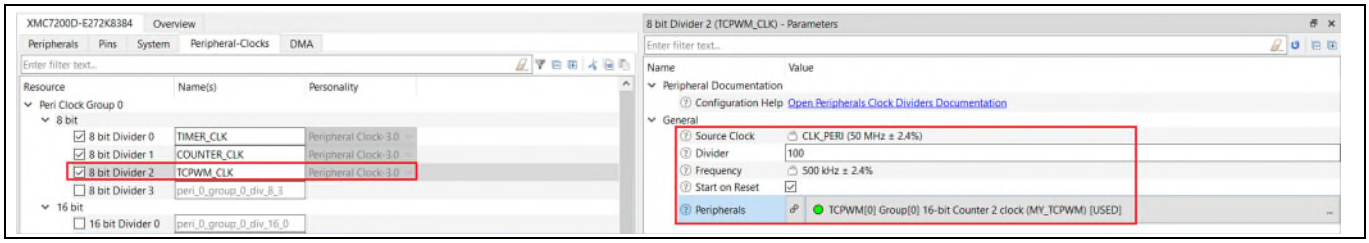


Figure 20 TCPWM PWM clock configuration in device configurator tool

Code Listing 7 demonstrates an example to configure PWM mode in the configuration part.

Code Listing 7 XMC7200: Example to configure PWM mode in configuration part

```
const cy_stc_tcpwm_pwm_config_t MY_TCPWM_config =

{
    .pwmMode = CY_TCPWM_PWM_MODE_PWM,
    .clockPrescaler = CY_TCPWM_PWM_PRESCALER_DIVBY_1,
    .pwmAlignment = CY_TCPWM_PWM_LEFT_ALIGN,
    .deadTimeClocks = 0,
    .runMode = CY_TCPWM_PWM_CONTINUOUS,
    .period0 = 4095,
    .period1 = 32768,
    .enablePeriodSwap = false,
    .compare0 = 2048,
    .compare1 = 16384,
    .enableCompareSwap = false,
    .interruptSources = (CY_TCPWM_INT_ON_TC & 0U) | (CY_TCPWM_INT_ON_CC0 & 0U) | (CY_TCPWM_INT_ON_CC1 &
0U),
    .invertPWMOut = CY_TCPWM_PWM_INVERT_DISABLE,
    .invertPWMOutN = CY_TCPWM_PWM_INVERT_DISABLE,
    .killMode = CY_TCPWM_PWM_STOP_ON_KILL,
    .swapInputMode = MY_TCPWM_INPUT_DISABLED & 0x3U,
    .swapInput = CY_TCPWM_INPUT_0,
    .reloadInputMode = MY_TCPWM_INPUT_DISABLED & 0x3U,
```

Code Listing 7 XMC7200: Example to configure PWM mode in configuration part

```
    .reloadInput = CY_TCPWM_INPUT_0,
    .startInputMode = MY_TCPWM_INPUT_DISABLED & 0x3U,
    .startInput = CY_TCPWM_INPUT_0,
    .killInputMode = MY_TCPWM_INPUT_DISABLED & 0x3U,
    .killInput = CY_TCPWM_INPUT_0,
    .countInputMode = MY_TCPWM_INPUT_DISABLED & 0x3U,
    .countInput = CY_TCPWM_INPUT_1,
    .swapOverflowUnderflow = false,
    .immediateKill = false,
    .tapsEnabled = 45,
   .compare2 = 0,
   .compare3 = 16384,
    .enableCompare1Swap = false,
    .compare0MatchUp = true,
    .compare0MatchDown = false,
    .compare1MatchUp = true,
    .compare1MatchDown = false,
    .kill1InputMode = MY_TCPWM_INPUT_DISABLED & 0x3U,
    .kill1Input = CY_TCPWM_INPUT_0,
    .pwmOnDisable = CY_TCPWM_PWM_OUTPUT_HIGHZ,
    .trigger0Event = CY_TCPWM_CNT_TRIGGER_ON_TC,
    .trigger1Event = CY_TCPWM_CNT_TRIGGER_ON_DISABLED,
};
```

Code Listing 8 demonstrates an example program to TCPWM in the driver part.

Code Listing 8 XMC7200: Example to program PWM mode in driver part

```
int main(void)
{
    cy_rslt_t result;

    /* Initialize the device and board peripherals */
    result = cybsp_init() ;
    if (result != CY_RSLT_SUCCESS)
    {
        CY_ASSERT(0);
    }

    /*Enable global interrupt*/
    __enable_irq();

   /*TCPWM PWM Mode initial*/
   if (CY_TCPWM_SUCCESS != Cy_TCPWM_PWM_Init(MY_TCPWM_HW, MY_TCPWM_NUM, &MY_TCPWM_config))
           {
             CY_ASSERT(0);
           }

   /* Enable the TCPWM PWM */
   Cy_TCPWM_PWM_Enable(MY_TCPWM_HW, MY_TCPWM_NUM);

   /* Start the TCPWM PWM by software command*/
   Cy_TCPWM_TriggerReloadOrIndex_Single(MY_TCPWM_HW, MY_TCPWM_NUM);

       for(;;)
       {

       }
}
```

2.4 PWM Dead Time (PWM_DT) mode

This section describes how to set up the PWM_DT mode. PWM_DT mode is for an application to output the PWM signal with dead time on the line_out and line_compl_out. Figure 21 shows the PWM_DT mode in upward counting mode.

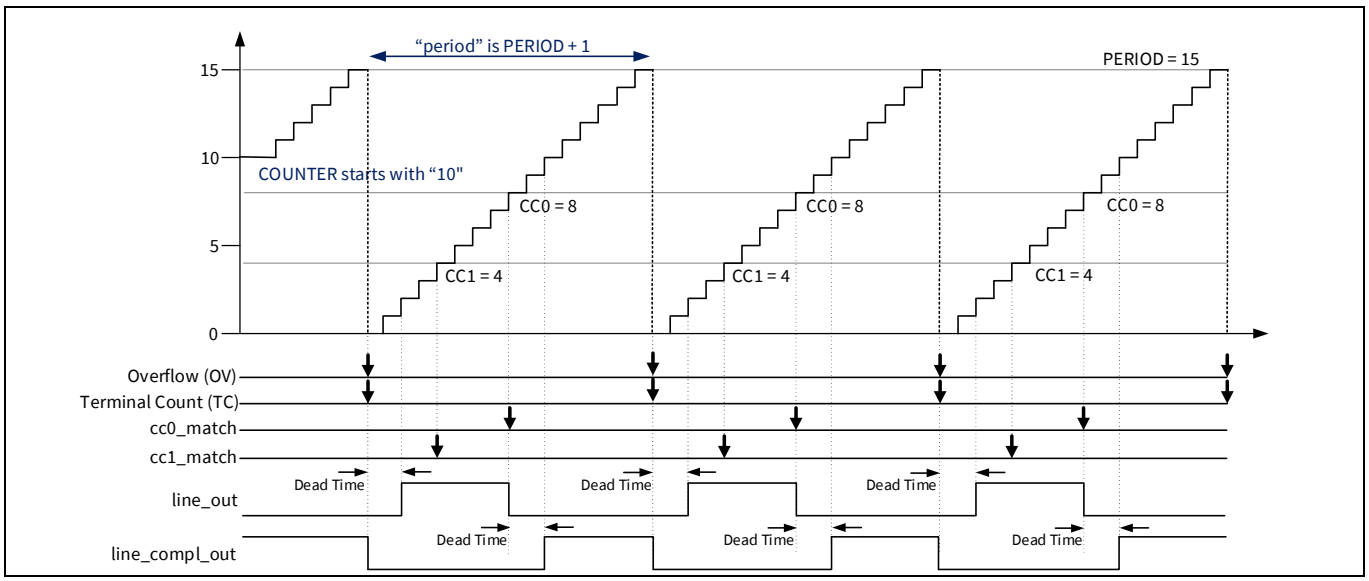


Figure 21 Counting behavior for PWM_DT mode

PWM signal with dead time is configured like the PWM mode. PWM_DT mode is similar to the PWM mode. PWM signal in PWM_DT mode has a dead time.

The definition of dead time is configured by the DT_LINE_OUT_L bits in the DT (e.g., `TCPWM0_GRP0_CNT0_DT`) register. Dead time is added to each PWM rising edge of line_out and line_compl_out. The dead time width of both line_out signals is the same.

The 16-bit counter for motors has advanced motor control features. The dead time for line_out can be configured by DT_LINE_OUT_L and DT_LINE_OUT_H bits in the DT register, and the dead time for line_compl_out can be configured by DT_LINE_COMPL_OUT bits in the DT register. Dead time width of Line_out and line_compl_out can be set in different values.

2.4.1 Use case

This section describes a use case of the PWM_DT mode. PWM_DT signal is generated with overflow and cc0_match events. The following is an example of configuring TCPWM using PDL:

- TCPWM operation mode: PWM_DT mode
- Using counter: TCPWM0/Group1/Counter0
- Counter start operation: Start by software
- PWM duty: 50%
- Input clock
  - CLK_PERI: 50 MHz
  - Divide value: Divided by 100 (50 MHz/100 = 500 kHz)
- Amount of dead time cycles in the counter clock
  - Line_out: 1000 clocks
  - Line_compl_out: 1000 clocks

TCPWM operation examples

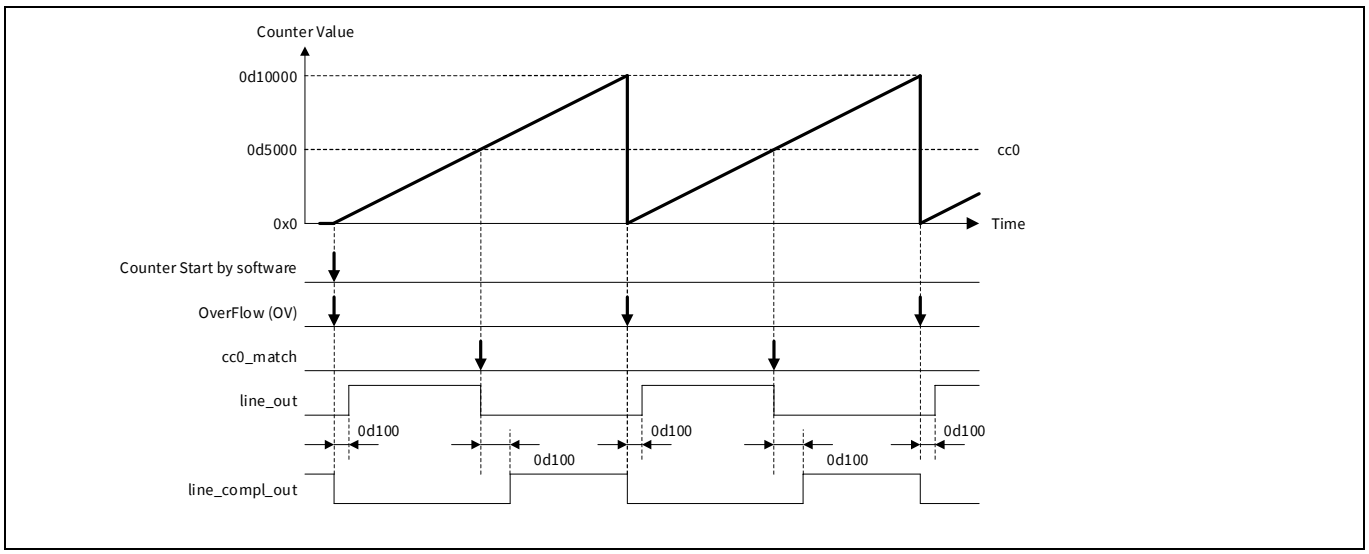


Figure 22 Timing chart of PWM_DT mode

Figure 23 shows the operation flow of this use case.

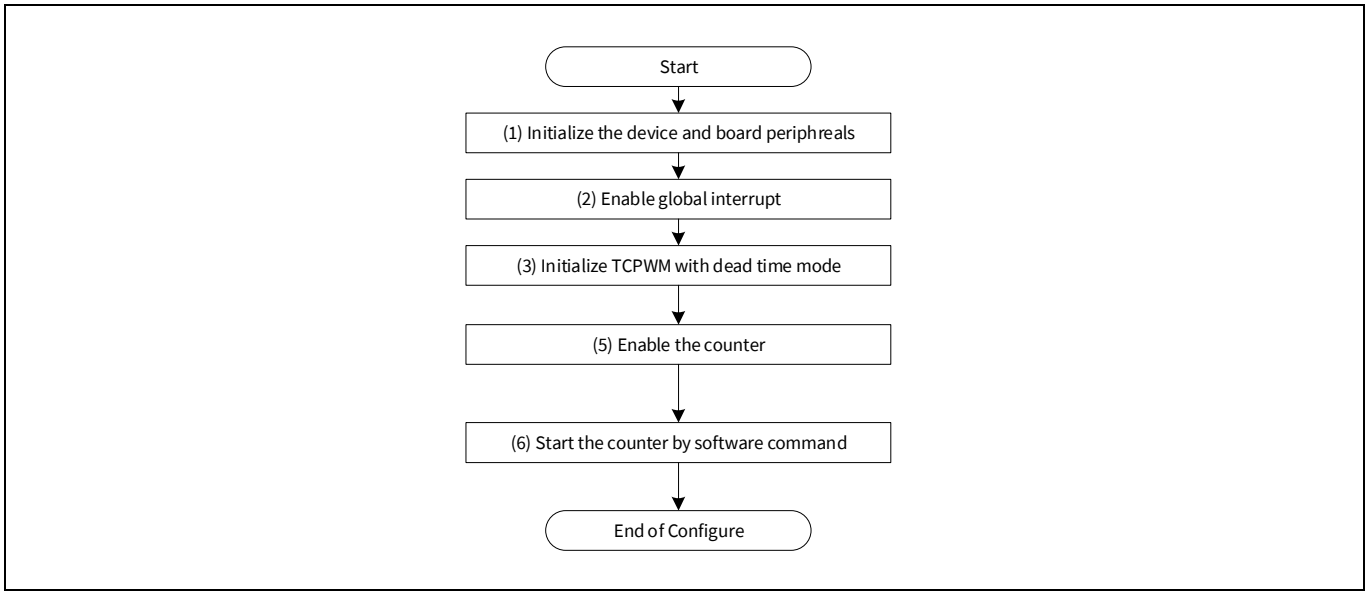


Figure 23 Operation flow example

1. Initialize the device and board peripherals
2. Enable global interrupt (CPU interrupt enable). For more details, see the CPU interrupt handling sections in the Architecture TRM
3. Initialize the TCPWM with dead time mode
4. Enable the TCPWM counter

Note: If the TCPWM counter is enabled, disable it to prevent malfunction.

5. Start the TCPWM counter with a software command

2.4.2 Configuration and example

Figure 24 shows the parameters of the configuration part in device configurator tool for PWM_DT mode.

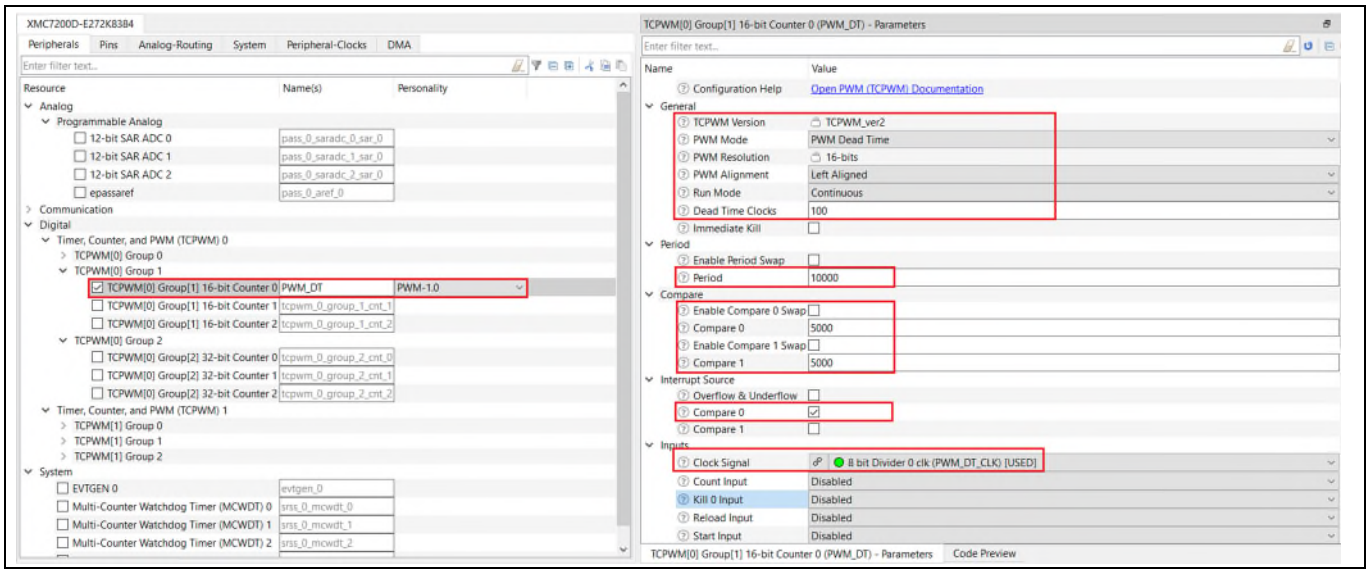


Figure 24 TCPWM dead time configuration in device configurator tool

Figure 25 shows the clock parameters configuration in device configurator tool for PWM_DT mode.

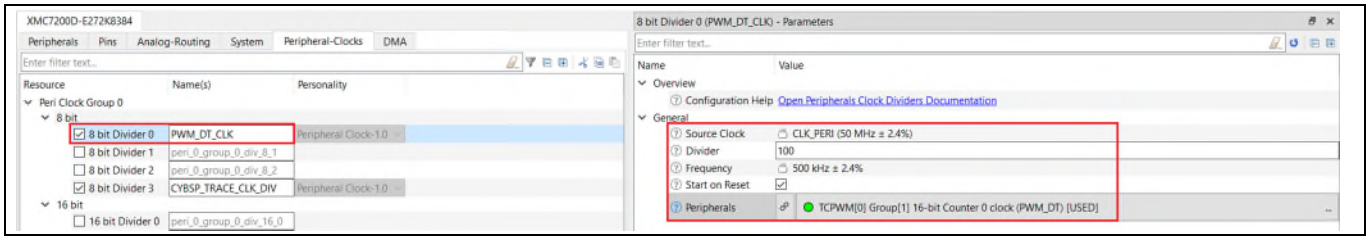


Figure 25 TCPWM dead time clock configuration in device configurator tool

Code Listing 9 demonstrates an example to configure PWM_DT mode in the configuration part.

Code Listing 9 XMC7200: Example to configure PWM_DT mode in the configuration part

```
const cy_stc_tcpwm_pwm_config_t PWM_DT_config =
{
.pwmMode = CY_TCPWM_PWM_MODE_DEADTIME,
.clockPrescaler = CY_TCPWM_PWM_PRESCALER_DIVBY_1,
.pwmAlignment = CY_TCPWM_PWM_LEFT_ALIGN,
.deadTimeClocks = 100,
.runMode = CY_TCPWM_PWM_CONTINUOUS,
.period0 = 10000,
.period1 = 32768,
.enablePeriodSwap = false,
.compare0 = 5000,
.compare1 = 16384,
.enableCompareSwap = false,
.interruptSources = (CY_TCPWM_INT_ON_TC & 0U) | (CY_TCPWM_INT_ON_CC0 ) | (CY_TCPWM_INT_ON_CC1 & 0U),
.invertPWMOut = CY_TCPWM_PWM_INVERT_DISABLE,
.invertPWMOutN = CY_TCPWM_PWM_INVERT_DISABLE,
.killMode = CY_TCPWM_PWM_STOP_ON_KILL,
.swapInputMode = PWM_DT_INPUT_DISABLED & 0x3U,
.swapInput = CY_TCPWM_INPUT_0,
.reloadInputMode = PWM_DT_INPUT_DISABLED & 0x3U,
.reloadInput = CY_TCPWM_INPUT_0,
```

TCPWM operation examples

Code Listing 9 XMC7200: Example to configure PWM_DT mode in the configuration part

```
.startInputMode = PWM_DT_INPUT_DISABLED & 0x3U,
.startInput = CY_TCPWM_INPUT_0,
.killInputMode = PWM_DT_INPUT_DISABLED & 0x3U,
.killInput = CY_TCPWM_INPUT_0,
.countInputMode = PWM_DT_INPUT_DISABLED & 0x3U,
.countInput = CY_TCPWM_INPUT_1,
.swapOverflowUnderflow = false,
.immediateKill = false,
.tapsEnabled = 45,
.compare2 = 5000,
.compare3 = 16384,
.enableCompare1Swap = false,
.compare0MatchUp = true,
.compare0MatchDown = false,
.compare1MatchUp = true,
.compare1MatchDown = false,
.kill1InputMode = PWM_DT_INPUT_DISABLED & 0x3U,
.kill1Input = CY_TCPWM_INPUT_0,
.pwmOnDisable = CY_TCPWM_PWM_OUTPUT_HIGHZ,
.trigger0Event = CY_TCPWM_CNT_TRIGGER_ON_DISABLED,
.trigger1Event = CY_TCPWM_CNT_TRIGGER_ON_DISABLED,
};
```

There are no new drivers here. Refer to the link above.

Code Listing 10 demonstrates an example to program PWM_DT in the driver part.

Code Listing 10 XMC7200: Example to program PWM_DT mode in the driver part

```
int main(void)
{
    cy_rslt_t result;

    /* Initialize the device and board peripherals */
    result = cybsp_init() ;
    if (result != CY_RSLT_SUCCESS)
    {
        CY_ASSERT(0);
    }

   /*Enable global interrupt*/
    __enable_irq();

   /*TCPWM Dead Time Mode initial*/
   if (CY_TCPWM_SUCCESS != Cy_TCPWM_PWM_Init(PWM_DT_HW, PWM_DT_NUM, &PWM_DT_config))
           {
             CY_ASSERT(0);
           }

   /* Enable the PWM_DT */
   Cy_TCPWM_PWM_Enable(PWM_DT_HW, PWM_DT_NUM);

   /* Then start the PWM_DT*/
   Cy_TCPWM_TriggerReloadOrIndex_Single(PWM_DT_HW, PWM_DT_NUM);

       For(;;)
       {

       }
}
```

3 Relation of trigger multiplexer

XMC7000 family has the trigger multiplexer module. TCPWM uses the trigger multiplexer to connect modules, such as SAR ADC, P-DMA, and TCPWM itself. See the Device datasheet for the trigger multiplexer connection for each device.

Figure 26 shows the operation flow when using peripherals, such as a trigger multiplexer, ADC, and TCPWM. This flow is the same for sections 3.1 and 3.2.

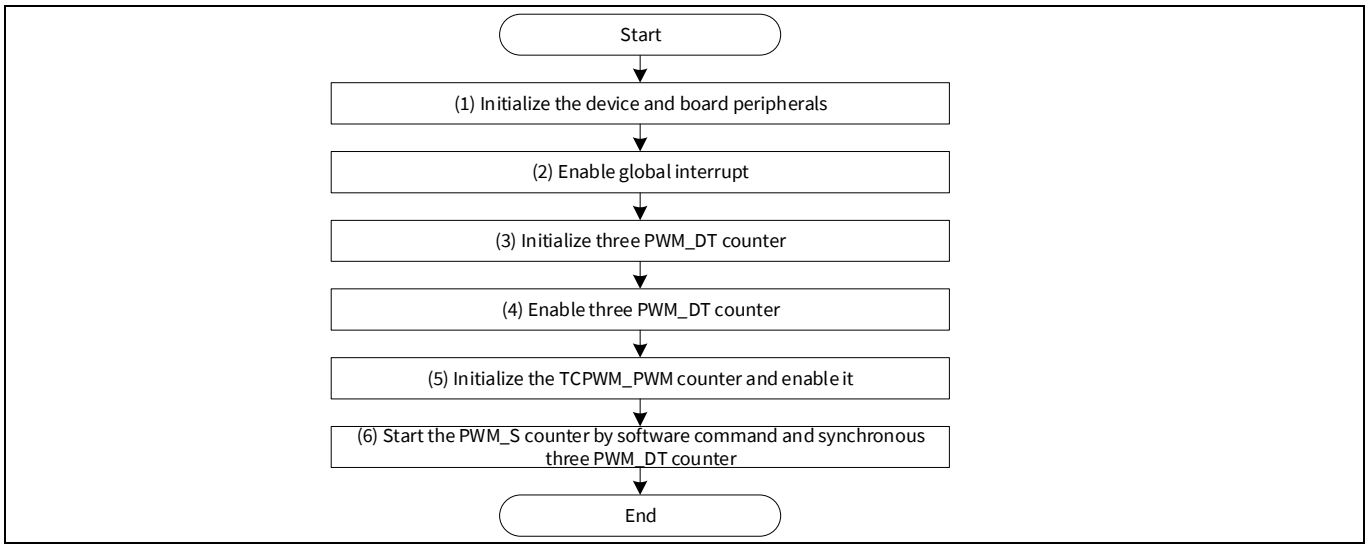


Figure 26 Operation flow example

1. Initialize the device and board peripherals
2. Enable global interrupt (CPU interrupt enable). For more details, see the CPU interrupt handling sections in the Architecture TRM
3. Initialize three PWM_DT counter (PWM_U, PWM_V, PWM_W) with rising signal as reload input through TCPWM1_GRP0_CNT0_RELOAD_VALUE
4. Enable three PWM_DT (PWM_U, PWM_V, PWM_W) counters

Note: If the TCPWM counter is enabled, disable it to prevent malfunction.

5. Initialize the TCPWM_S counter with PWM mode and enable it
6. Start the TCPWM_S by a software command to synchronous three TCPWM_DT counters

Note: TCPWM_S: The TCPWM start counter is used to trigger the other three PWM_DT counters at the same time.

3.1 Starting three TCPWMs simultaneously

3.1.1 Use case

This section describes a use case for starting three TCPWMs at the same time by software. The following is an example of configuring TCPWM using PDL:

- TCPWM operation mode: PWM_DT mode

Relation of trigger multiplexer

- Using counter: TCPWM1/Group0/Counter0, 1, 2
- Counter start operation: Start by software
- Input clock
  - CLK_PERI: 196 MHz
  - Divide value: Divided by 196 (196 MHz/196 = 1 MHz)

Figure 27 shows an example to use a trigger multiplexer for starting three counters of TCPWM simultaneously to output PMW signals. These counters use the same input trigger for starting the event. This input trigger is configured by the trigger multiplexer.

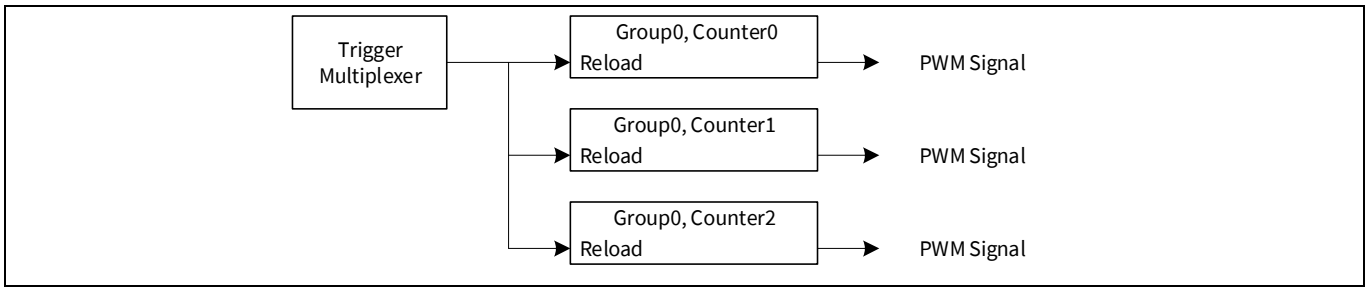


Figure 27 XMC7200: Starting three TCPWMs simultaneously by reloading signal

To explain how to set up this example, a detailed block diagram of the trigger multiplexer and TCPWM is shown in Figure 28. As shown, the 40 outputs of the trigger multiplexer are connected to the 32-to-1 selector of each counter. Each counter can select the signal by TCPWM1_GRP0_CNT[a]_TR_IN_SEL0/1 register [a = counter number (0, 1, 2)].

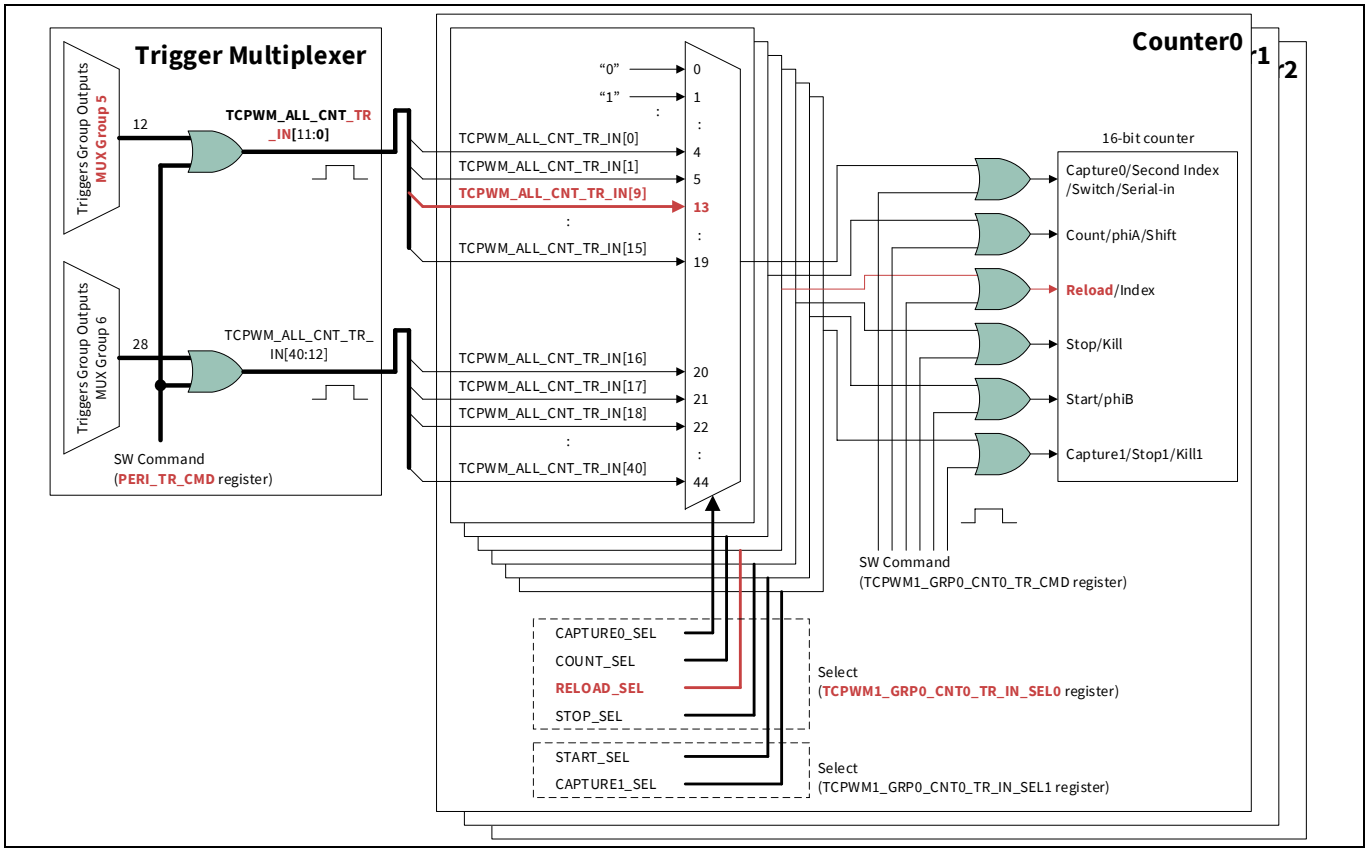


Figure 28 XMC7200: Detailed block diagram of the trigger multiplexer and TCPWM for connection

Relation of trigger multiplexer

Note: The function of the counter control signal varies depending on the mode.

For example, if the RELOAD_SEL bits in the TCPWM1_GRP0_CNT0_TR_IN_SEL0 register is set to “13”, TCPWM_ALL_CNT_TR_IN[9] of the trigger multiplexer will be selected as the counter reload. TCPWM_ALL_CNT_TR_IN[9] belongs to MUX Group5 of “Group Trigger”. If the PERI_TRM_CMD register is used, a high/low/pulse signal can be output to TCPWM_ALL_CNT_TR_IN[9] (here, this function is called a software command). If this output is supplied to reload all counters, all counters start at the same time.

For more information about MUX GROUP, see the "Triggers group outputs" chapter in the Device datasheet.

3.1.2 Configuration and example

Figure 29 shows the parameters of the configuration part in the device configurator tool for three PWM_DT simultaneous start-by-reload signals.

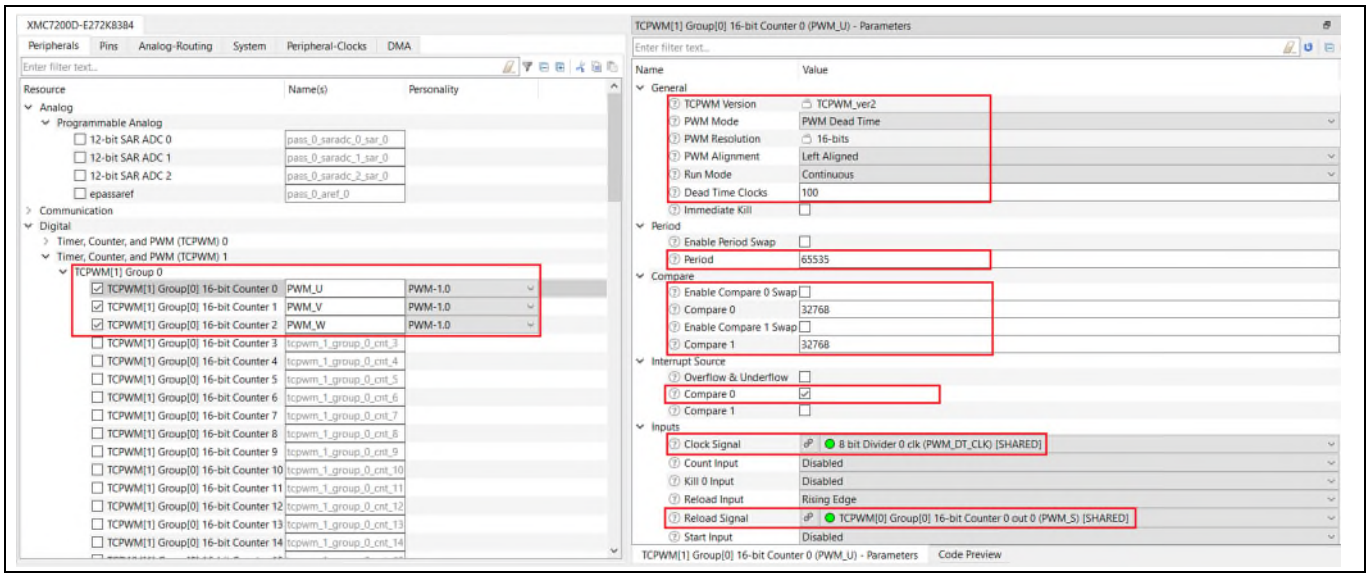


Figure 29 XMC7200: Detailed block diagram of the trigger multiplexer and TCPWM for connection

Figure 30 shows the clock parameters in device configurator tool for three PWM_DT.

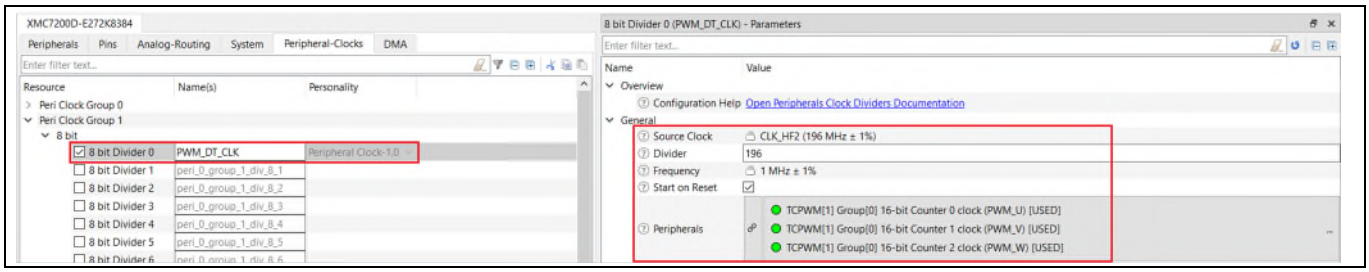


Figure 30 XMC7200: Detailed block diagram of the trigger multiplexer and TCPWM for connection

Code Listing 11 demonstrates an example to start three PWM_DT simultaneously in the configuration part for PWM_U. PWM_V and PWM_W configurations are same.

Code Listing 11 XMC7200: Example to start three TCPWMs simultaneously in configuration part

```
const cy_stc_tcpwm_pwm_config_t PWM_U_config =
{
.pwmMode = CY_TCPWM_PWM_MODE_DEADTIME,
```

Relation of trigger multiplexer

Code Listing 11 XMC7200: Example to start three TCPWMs simultaneously in configuration part

```
.clockPrescaler = CY_TCPWM_PWM_PRESCALER_DIVBY_1,
.pwmAlignment = CY_TCPWM_PWM_LEFT_ALIGN,
.deadTimeClocks = 100,
.runMode = CY_TCPWM_PWM_CONTINUOUS,
.period0 = 65535,
.period1 = 32768,
.enablePeriodSwap = false,
.compare0 = 32768,
.compare1 = 16384,
.enableCompareSwap = false,
.interruptSources = (CY_TCPWM_INT_ON_TC & 0U) | (CY_TCPWM_INT_ON_CC0 ) | (CY_TCPWM_INT_ON_CC1 & 0U),
.invertPWMOut = CY_TCPWM_PWM_INVERT_DISABLE,
.invertPWMOutN = CY_TCPWM_PWM_INVERT_DISABLE,
.killMode = CY_TCPWM_PWM_STOP_ON_KILL,
.swapInputMode = PWM_U_INPUT_DISABLED & 0x3U,
.swapInput = CY_TCPWM_INPUT_0,
.reloadInputMode = CY_TCPWM_INPUT_RISINGEDGE,
.reloadInput = TCPWM1_GRP0_CNT0_RELOAD_VALUE,
.startInputMode = PWM_U_INPUT_DISABLED & 0x3U,
.startInput = CY_TCPWM_INPUT_0,
.killInputMode = PWM_U_INPUT_DISABLED & 0x3U,
.killInput = CY_TCPWM_INPUT_0,
.countInputMode = PWM_U_INPUT_DISABLED & 0x3U,
.countInput = CY_TCPWM_INPUT_1,
.swapOverflowUnderflow = false,
.immediateKill = false,
.tapsEnabled = 45,
.compare2 = 32768,
.compare3 = 16384,
.enableCompare1Swap = false,
.compare0MatchUp = true,
.compare0MatchDown = false,
.compare1MatchUp = true,
.compare1MatchDown = false,
.kill1InputMode = PWM_U_INPUT_DISABLED & 0x3U,
.kill1Input = CY_TCPWM_INPUT_0,
.pwmOnDisable = CY_TCPWM_PWM_OUTPUT_HIGHZ,
.trigger0Event = CY_TCPWM_CNT_TRIGGER_ON_DISABLED,
.trigger1Event = CY_TCPWM_CNT_TRIGGER_ON_DISABLED,
};
```

Code Listing 12 demonstrates an example to program three PWM_DT (PWM_U, PWM_V, and PWM_W) simultaneously in the driver part.

Code Listing 12 XMC7200: Example to program three PWM_DT counters in driver part

```
/* PWM_U, PWM_V, PWM_W CC0 interrups configuration structure */
const cy_stc_sysint_t PWM_U_cfg = {
    .intrSrc = ((NvicMux3_IRQn << 16) | tcpwm_1_interrupts_0_IRQn),
    .intrPriority = 7u
};
const cy_stc_sysint_t PWM_V_cfg = {
    .intrSrc = ((NvicMux3_IRQn << 16) | tcpwm_1_interrupts_1_IRQn),
    .intrPriority = 7u
};
const cy_stc_sysint_t PWM_W_cfg = {
    .intrSrc = ((NvicMux3_IRQn << 16) | tcpwm_1_interrupts_2_IRQn),
    .intrPriority = 7u
};


int main(void)
{
     cy_rslt_t result;

     /* Initialize the device and board peripherals */
      result = cybsp_init() ;
       if (result != CY_RSLT_SUCCESS)
           {
               CY_ASSERT(0);
           }
```

Relation of trigger multiplexer

Code Listing 12 XMC7200: Example to program three PWM_DT counters in driver part

```
     /*enable global interrupt*/
       __enable_irq();

     /*Initial the user LED1, LED2 and LED3*/
            :
     /*PWM_U Dead Time Mode initial*/
     if (CY_TCPWM_SUCCESS != Cy_TCPWM_PWM_Init(PWM_U_HW, PWM_U_NUM, &PWM_U_config))
         {
            CY_ASSERT(0);
         }
     /* Enable the PWM_U */
     Cy_TCPWM_PWM_Enable(PWM_U_HW, PWM_U_NUM);

     /*PWM_W Dead Time Mode initial*/
    if (CY_TCPWM_SUCCESS != Cy_TCPWM_PWM_Init(PWM_W_HW, PWM_W_NUM, &PWM_W_config))
        {
            CY_ASSERT(0);
         }
   /* Enable the PWM_W */
    Cy_TCPWM_PWM_Enable(PWM_W_HW, PWM_W_NUM);

    /*PWM_V Dead Time Mode initial*/
    if (CY_TCPWM_SUCCESS != Cy_TCPWM_PWM_Init(PWM_V_HW, PWM_V_NUM, &PWM_V_config))
          {
                CY_ASSERT(0);
          }
    /* Enable the PWM_W */
     Cy_TCPWM_PWM_Enable(PWM_V_HW, PWM_V_NUM);

   /*Register and enable PWM_U, PWM_V,PWM_W interrupts*/
     pwms_interrupt_initial();

    /*PWM_S PWM Mode initial*/
    if (CY_TCPWM_SUCCESS != Cy_TCPWM_PWM_Init(PWM_S_HW, PWM_S_NUM, &PWM_S_config))
         {
             CY_ASSERT(0);
         }
    /* Enable the TCPWM_S */
    Cy_TCPWM_PWM_Enable(PWM_S_HW, PWM_S_NUM);

    /* Then start the TCPWM_S to synchronous PWM_U, PWM_V and PWM_W */
    Cy_TCPWM_TriggerReloadOrIndex_Single(PWM_S_HW, PWM_S_NUM);

    for(;;)
    {

    }
}

static void pwms_interrupt_initial(void)
{
        /*Configure and register pwm_u interrupt*/
        Cy_SysInt_Init(&PWM_U_cfg, pwm_u_interrupt_handler);
        NVIC_ClearPendingIRQ((IRQn_Type)PWM_U_cfg.intrSrc);

        /*Configure and register pwm_v interrupt*/
      Cy_SysInt_Init(&PWM_V_cfg, pwm_v_interrupt_handler);
        NVIC_ClearPendingIRQ((IRQn_Type)PWM_V_cfg.intrSrc);

        /*Configure and register pwm_w interrupt*/
        Cy_SysInt_Init(&PWM_W_cfg, pwm_w_interrupt_handler);
        NVIC_ClearPendingIRQ((IRQn_Type)PWM_W_cfg.intrSrc);
        NVIC_EnableIRQ((IRQn_Type) NvicMux3_IRQn);
}

/*PWM_U, PWM_V, PWM_W CC0 interrupt handlers to toggle user LED1, LED2 and LED3*/
:
```

3.2 Starting AD conversion with TCPWM output

3.2.1 Use case

This section describes an example of using the TCPWM trigger output as the start signal for AD conversion. Here, ch0 of ADC0 is converted by Group1_counter0 of TCPWM1, respectively. The following is an example of configuring TCPWM and trigger one-to-one using PDL and ModusToolbox™ device configurator tool:

- TCPWM operation mode: PWM mode
- Using counter: TCPWM1/Group1/Counter0
- Counter start operation: Start by software
- Using ADC0 channel: Ch0
- Using triggers: Triggers one-to-one, MUX Group 7
  - For ADC0 Ch0: TCPWM1_16M_TR_OUT1[0]

Figure 31 shows an example of starting AD conversion with TCPWM outputs (tr_out1).

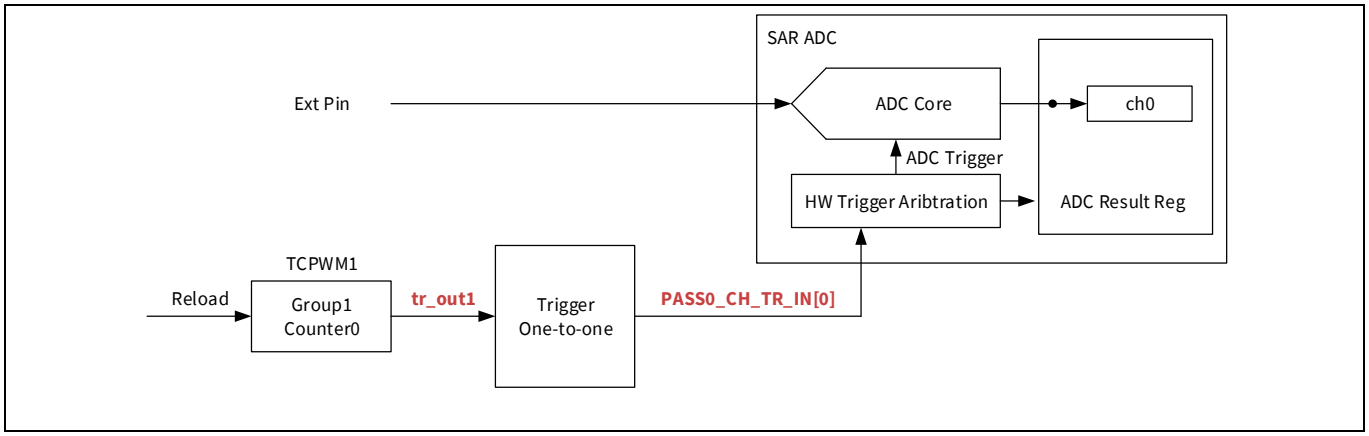


Figure 31 XMC7200: Starting AD conversion with TCPWM output

In addition, Figure 32 shows a timing chart of AD conversion by a cc0 match event of each counter. The results of each AD conversion executed by the start trigger is stored in registers for the channel.

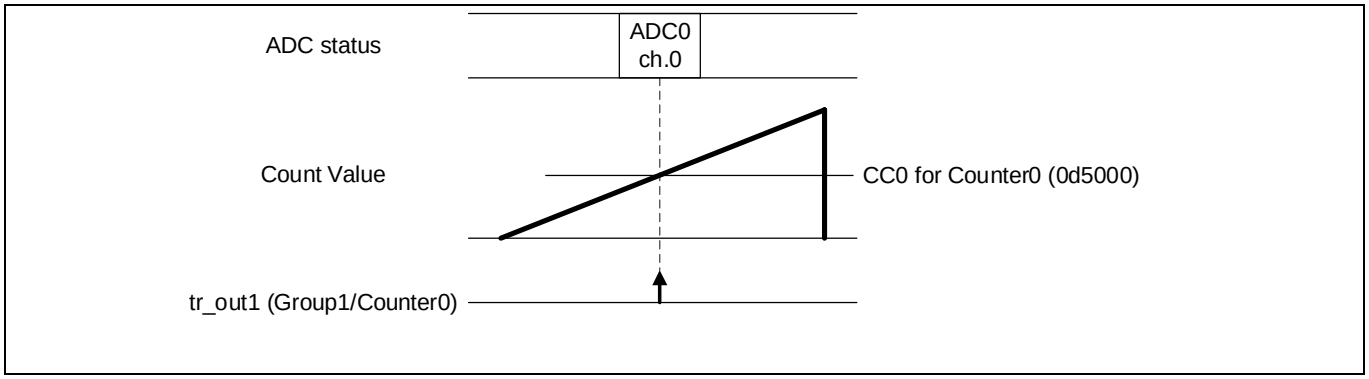


Figure 32 Timing chart for AD conversion with TCPWM output

For more details on the cc0 match event, see PWM mode and PWM Dead Time (PWM_DT) mode.

Figure 33 is a detailed listing of the trigger one-to-one for connecting tr_out1 to the trigger of the SAR ADC.

0	TCPWM1_16M_TR_OUT1[0]	PASS0_CH_TR_IN[0]	TCPWM1 Group #1 Counter #00 (PWM1_M_0) to SAR0 ch#0
1	TCPWM1_16M_TR_OUT1[3]	PASS0_CH_TR_IN[1]	TCPWM1 Group #1 Counter #03 (PWM1_M_3) to SAR0 ch#1
2	TCPWM1_16M_TR_OUT1[6]	PASS0_CH_TR_IN[2]	TCPWM1 Group #1 Counter #06 (PWM1_M_6) to SAR0 ch#2
3	TCPWM1_16M_TR_OUT1[9]	PASS0_CH_TR_IN[3]	TCPWM1 Group #1 Counter #09 (PWM1_M_9) to SAR0 ch#3
4:31	TCPWM1_16_TR_OUT1[0:27]	PASS0_CH_TR_IN[4:31]	TCPWM1 Group #0 Counter #00 through 27 (PWM1_0 to PWM1_27) to SAR0 ch#4 through SAR0 ch#31
32	TCPWM1_16M_TR_OUT1[1]	PASS0_CH_TR_IN[32]	TCPWM1 Group #1 Counter #01 (PWM1_M_1) to SAR1 ch#0
33	TCPWM1_16M_TR_OUT1[4]	PASS0_CH_TR_IN[33]	TCPWM1 Group #1 Counter #04 (PWM1_M_4) to SAR1 ch#1
34	TCPWM1_16M_TR_OUT1[7]	PASS0_CH_TR_IN[34]	TCPWM1 Group #1 Counter #07 (PWM1_M_7) to SAR1 ch#2
35	TCPWM1_16M_TR_OUT1[10]	PASS0_CH_TR_IN[35]	TCPWM1 Group #1 Counter #10 (PWM1_M_10) to SAR1 ch#3
36:63	TCPWM1_16_TR_OUT1[28:55]	PASS0_CH_TR_IN[36:63]	TCPWM1 Group #0 Counter #28 through 55 (PWM1_28 to PWM1_55) to SAR1 ch#4 through SAR1 ch#31
64	TCPWM1_16M_TR_OUT1[2]	PASS0_CH_TR_IN[64]	TCPWM1 Group #1 Counter #02 (PWM1_M_2) to SAR2 ch#0
65	TCPWM1_16M_TR_OUT1[5]	PASS0_CH_TR_IN[65]	TCPWM1 Group #1 Counter #05 (PWM1_M_5) to SAR2 ch#1
66	TCPWM1_16M_TR_OUT1[8]	PASS0_CH_TR_IN[66]	TCPWM1 Group #1 Counter #08 (PWM1_M_8) to SAR2 ch#2
67	TCPWM1_16M_TR_OUT1[11]	PASS0_CH_TR_IN[67]	TCPWM1 Group #1 Counter #11 (PWM1_M_11) to SAR2 ch#3
68:95	TCPWM1_16_TR_OUT1[56:83]	PASS0_CH_TR_IN[68:95]	TCPWM1 Group #1 Counter #56 through 83 (PWM1_56 to PWM1_83) to SAR2 ch#4 through SAR2 ch#31

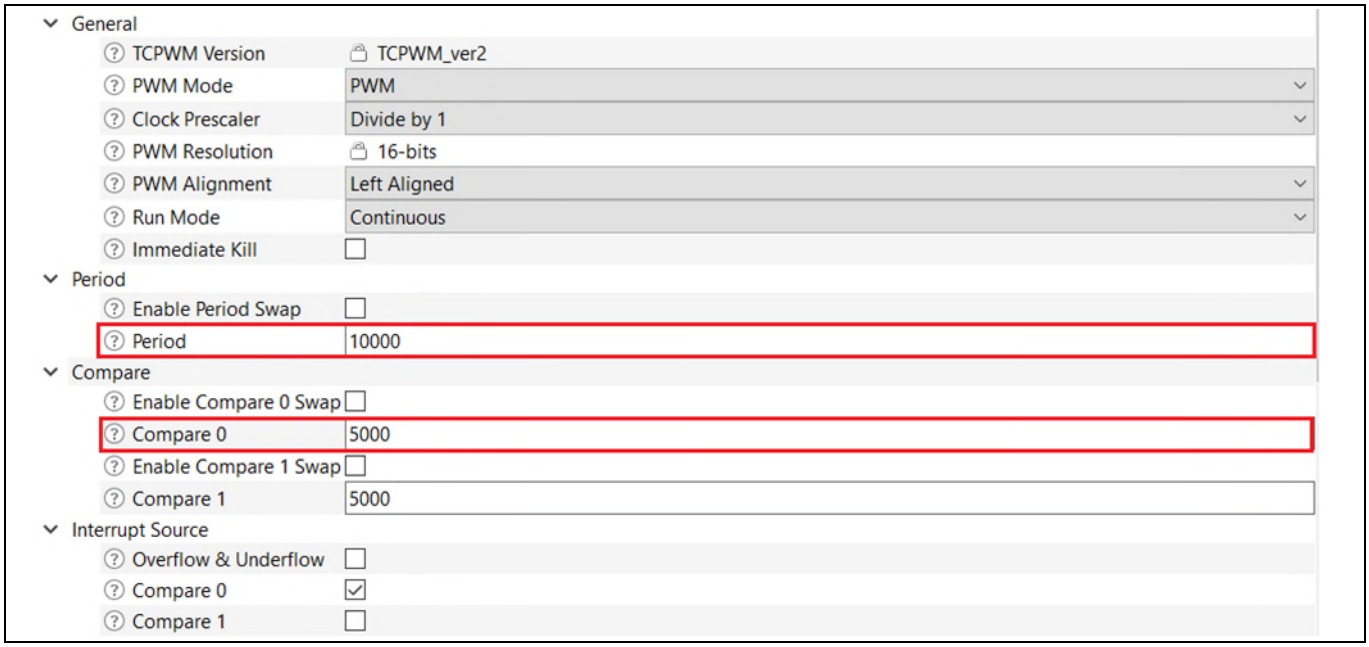
Figure 33 XMC7200: Detailed listing of the trigger one-to-one

As shown above, tr_out1 of Group1/Counter0 is connected to the ch0 trigger of SAR ADC0 via “one-to-one trigger groups” of the trigger multiplexer. PASS0_CH_TR_IN[0] can be activated by the PERI_TR_1TO1_GR7_TR_CTL0 registers.

For more information about MUX GROUP, see the "Triggers One-to-One" chapter in the Device datasheet.

3.2.2 Configuration and example

Figure 34 shows the parameters of the configuration part in the ModusToolbox™ device configurator tool for TCPWM1 Group1 Counter0 PWM mode. Figure 35 shows the TCPWM1 Group1 Counter0 PWM mode parameters with the PDL codes.



Relation of trigger multiplexer

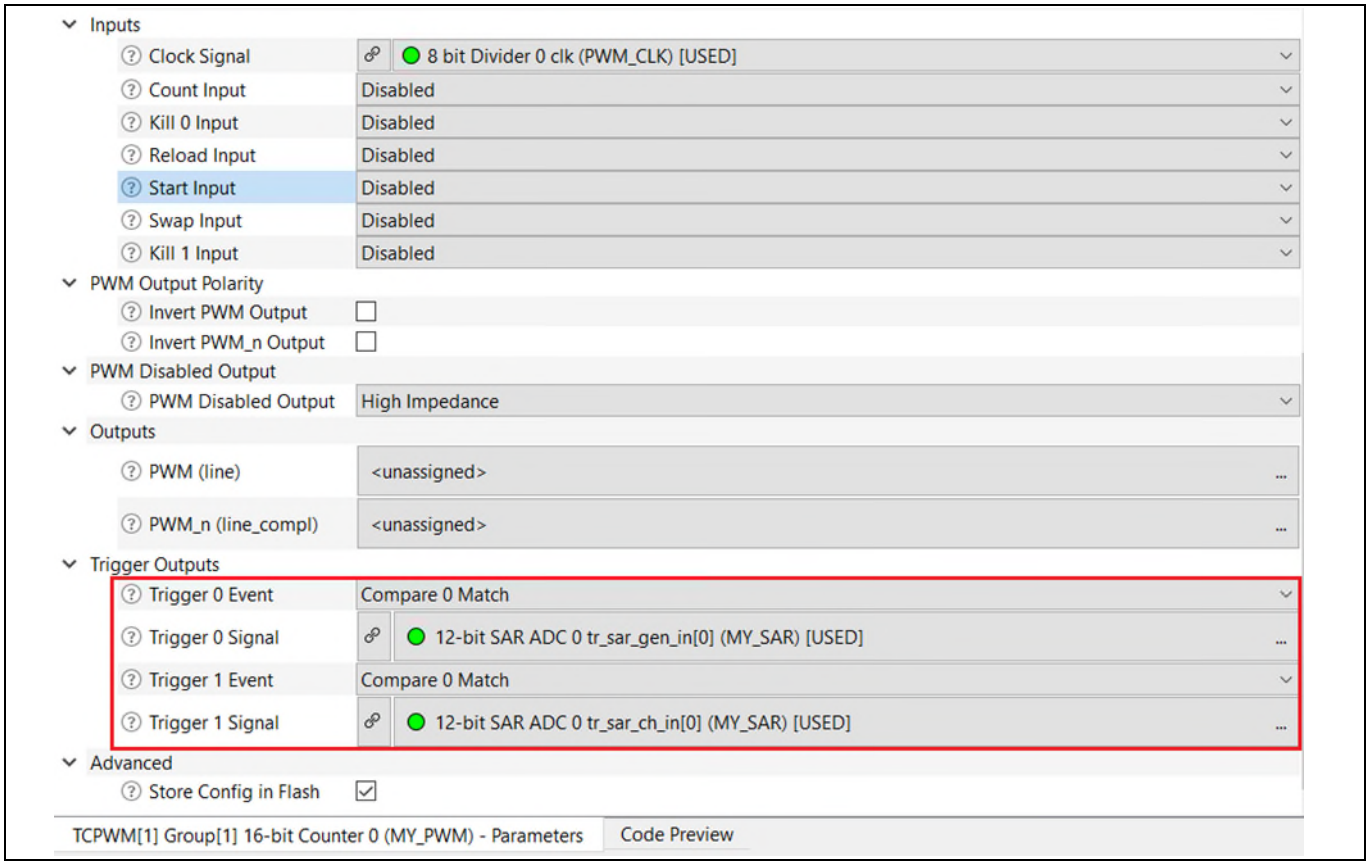


Figure 34 XMC7200: TCPWM counter output in configuration part

```
{
    .pwmMode = CY_TCPWM_PWM_MODE_PWM,
    .clockPrescaler = CY_TCPWM_PWM_PRESCALER_DIVBY_1,
    .pwmAlignment = CY_TCPWM_PWM_LEFT_ALIGN,
    .deadTimeClocks = 0,
    .runMode = CY_TCPWM_PWM_CONTINUOUS,
    .period0 = 10000,
    .period1 = 32768,
    .enablePeriodSwap = false,
    .compare0 = 5000,
    .compare1 = 16384,
    .enableCompareSwap = false,
    .interruptSources = (CY_TCPWM_INT_ON_TC & 0U) | (CY_TCPWM_INT_ON_CC0 ) | (CY_TCPWM_INT_ON_CC1 & 0U),
    .invertPWMOut = CY_TCPWM_PWM_INVERT_DISABLE,
    .invertPWMOutN = CY_TCPWM_PWM_INVERT_DISABLE,
    .killMode = CY_TCPWM_PWM_STOP_ON_KILL,
    .swapInputMode = MY_PWM_INPUT_DISABLED & 0x3U,
    .swapInput = CY_TCPWM_INPUT_0,
    .reloadInputMode = MY_PWM_INPUT_DISABLED & 0x3U,
    .reloadInput = CY_TCPWM_INPUT_0,
    .startInputMode = MY_PWM_INPUT_DISABLED & 0x3U,
    .startInput = CY_TCPWM_INPUT_0,
    .killInputMode = MY_PWM_INPUT_DISABLED & 0x3U,
    .killInput = CY_TCPWM_INPUT_0,
    .countInputMode = MY_PWM_INPUT_DISABLED & 0x3U,
    .countInput = CY_TCPWM_INPUT_1,
    .swapOverflowUnderflow = false,
    .immediateKill = false,
    .tapsEnabled = 45,
    .compare2 = 5000,
    .compare3 = 16384,
    .enableCompare1Swap = false,
    .compare0MatchUp = true,
    .compare0MatchDown = false,
    .compare1MatchUp = true,
    .compare1MatchDown = false,
    .kill1InputMode = MY_PWM_INPUT_DISABLED & 0x3U,
    .kill1Input = CY_TCPWM_INPUT_0,
    .pwmOnDisable = CY_TCPWM_PWM_OUTPUT_HIGHZ,
    .trigger0Event = CY_TCPWM_CNT_TRIGGER_ON_CC0_MATCH,
    .trigger1Event = CY_TCPWM_CNT_TRIGGER_ON_CC0_MATCH,
```

Figure 35 XMC7200: TCPWM counter parameters with PDL code

Relation of trigger multiplexer

Code Listing 13 demonstrates an example program to start AD conversion with TCPWM counter output in the configuration part.

Code Listing 13 XMC7200: Example to start AD conversion with TCPWM output in configuration part

```
/* SAR0 interrupt configuration structure */
const cy_stc_sysint_t SAR0_IRQ_cfg = {
    .intrSrc = ((NvicMux3_IRQn << 16) | MY_SAR_CH0_IRQ),
    .intrPriority = 7u
};

static bool sar0_isr_set = false;

static void sar0_interrupt_handler(void);
static void init_analog_resources(void);

int main(void)
{
    cy_rslt_t result;
    uint16_t sar0Result0 = 0u;

    /* Initialize the device and board peripherals */
    result = cybsp_init();

    /* Board init failed. Stop program execution */
    if (result != CY_RSLT_SUCCESS)
    {
        CY_ASSERT(0);
    }

    /* Enable global interrupts */
    __enable_irq();

    /* Initialize retarget-io to use the debug UART port */
   result = cy_retarget_io_init(CYBSP_DEBUG_UART_TX, CYBSP_DEBUG_UART_RX, CY_RETARGET_IO_BAUDRATE);
       if (result != CY_RSLT_SUCCESS)
       {
              CY_ASSERT(0);
       }

   printf("XMC7000 MCU: TCPWM trigger ADC conversion\r\n");

    /*Analog resource initialize*/
    init_analog_resources();

    /* Initialize PWM using the config structure generated using device configurator*/
    if (CY_TCPWM_SUCCESS != Cy_TCPWM_PWM_Init(MY_PWM_HW, MY_PWM_NUM, &MY_PWM_config))
       {
           CY_ASSERT(0);
       }

    /* Enable the initialized PWM */
    Cy_TCPWM_PWM_Enable(MY_PWM_HW, MY_PWM_NUM);

    /* Then start the PWM */
    Cy_TCPWM_TriggerReloadOrIndex_Single(MY_PWM_HW, MY_PWM_NUM);

    for (;;)
    {
       if(sar0_isr_set)
          {
             sar0Result0 = Cy_SAR2_Channel_GetResult(MY_SAR_HW, 0u, NULL);
             sar0_isr_set = false;
             printf("sar0Result0 = %d \r\n",sar0Result0 );
          }
    }
}

static void sar0_interrupt_handler(void)
```

Relation of trigger multiplexer

Code Listing 13 XMC7200: Example to start AD conversion with TCPWM output in configuration part

```
{
    /* Check if End-Of-Scan trigger has occurred. If yes, set sar0_isr_set flag to true  */
    if (Cy_SAR2_Channel_GetInterruptStatus(MY_SAR_HW,0u) == CY_SAR2_INT_GRP_DONE)
    {
        sar0_isr_set = true;
    }

    /* Clear the interrupts */
    Cy_SAR2_Channel_ClearInterrupt(MY_SAR_HW, 0u ,CY_SAR2_INT_GRP_DONE);
}

static void init_analog_resources(void)
{
    /* Variable to capture return value of functions */
    cy_en_sar2_status_t status;

    /* Initialize SAR0 */
    status = Cy_SAR2_Init(MY_SAR_HW, &MY_SAR_config);
    if (CY_SAR2_SUCCESS != status)
    {
        CY_ASSERT(0);
    }

    /*Enable SAR0*/
    Cy_SAR2_Enable(MY_SAR_HW);

    /*Configure and register SAR0 channel interrupts*/
    Cy_SysInt_Init(&SAR0_IRQ_cfg, sar0_interrupt_handler);
    NVIC_ClearPendingIRQ((IRQn_Type)SAR0_IRQ_cfg.intrSrc);
    NVIC_EnableIRQ((IRQn_Type) NvicMux3_IRQn);

    /*Initialize SAR0 channel 0*/
    status = Cy_SAR2_Channel_Init(MY_SAR_HW, 0u, &MY_SAR_channel_0_config);
    if (CY_SAR2_SUCCESS != status)
    {
        CY_ASSERT(0);
    }

    /*Set SAR0 channel 0 interrupt mask*/
    Cy_SAR2_Channel_SetInterruptMask(MY_SAR_HW, 0u,CY_SAR2_INT_GRP_DONE);

    /*Enable SAR0 channel 0*/
    Cy_SAR2_Channel_Enable(MY_SAR_HW, 0u);
}
```

References

Contact Technical support to obtain XMC7000 family references documents.

[1] Device datasheet

- XMC7100 series 32-bit Arm® Cortex®-M7 microcontroller datasheet
- XMC7200 series 32-bit Arm® Cortex®-M7 microcontroller datasheet

[2] Devices architecture TRM

- XMC7000 MCU architecture technical reference manual

[3] Application notes

- AN234282 Using a SAR ADC in XMC7000 family
- AN234197 Trigger Multiplexer usage in XMC7000 family

Glossary

SAR ADC

Analog-to-digital converter. See the “SAR ADC” chapter of the Architecture TRM for more details

Peripheral clock divider

Peripheral clock divider derives a clock to use of each peripheral function, such as counters in TCPWM

Trigger multiplexer

A trigger multiplexer routes triggers from a source peripheral to a destination. See the “Trigger Multiplexer” chapter of the Architecture TRM for more details

Revision history

Document revision	Date	Description of changes
**	2021-12-03	Initial release
*A	2022-05-23	Updated use case code snippets and related description contents
*B	2023-03-22	Updated Starting AD conversion with TCPWM output use case code snippets Updated PWM_DT use case code snippets Updated starting three TCPWMs simultaneously code snippets Updated broken reference links Deleted other reference section Changed all TCPWM_DT to PWM_DT Updated some description content incorrect words

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2023-03-22

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2023 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

002-34119 Rev. *B

Important notice

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Warnings

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.