How to Use Timer, Counter, and PWM (TCPWM) in Traveo II Family

About this document

Scope and purpose

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

Associated Part Family

Traveo™ II Family CYT2/CYT3/CYT4 Series

Intended audience

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

Table of contents

Table of contents

1 Introduction

This application note describes how to use TCPWM in Traveo II family MCUs.

The CYT2 series has one Arm® Cortex®-M4F-based CPU (CM4) and one Cortex-M0+-based CPU (CM0+). The CYT4 series has two Arm Cortex-M7-based CPUs (CM7) and one CM0+, and The CYT3 series has one Arm CM7 and one CM0+.

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 series, initial configuration, and several functional modes with use cases.

To understand the functionality described and 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. Different counters can operate 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.

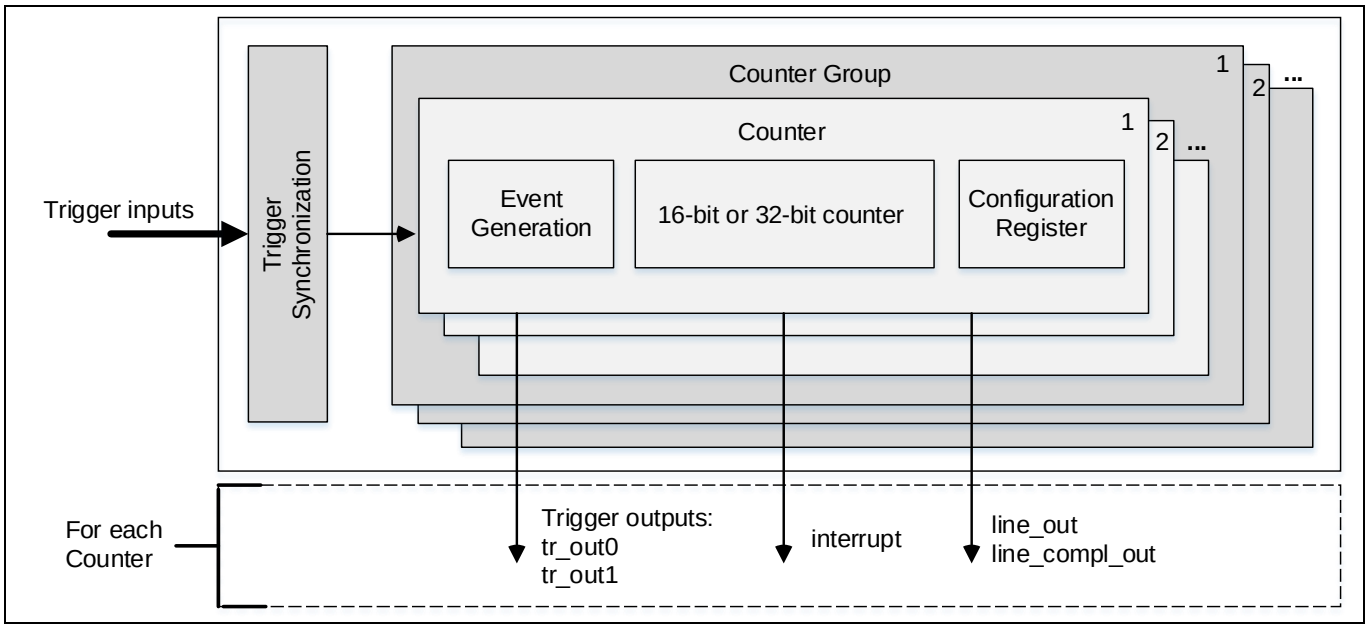

Figure 1 TCPWM Block Diagram

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

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

16-bit counter has an additional option for Motor Control. This counter has functions which are optimized for motor control operations.

Event Generation generates counter events for 16-bit or 32-bit counter as Reload, Start, Stop, Count, and Capture event. 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 Trigger Multiplexer.

See the "Trigger Multiplexer" chapter of the **Architecture TRM** for more details.

Figure 2 shows a TCPWM and Clock supplied block diagram.

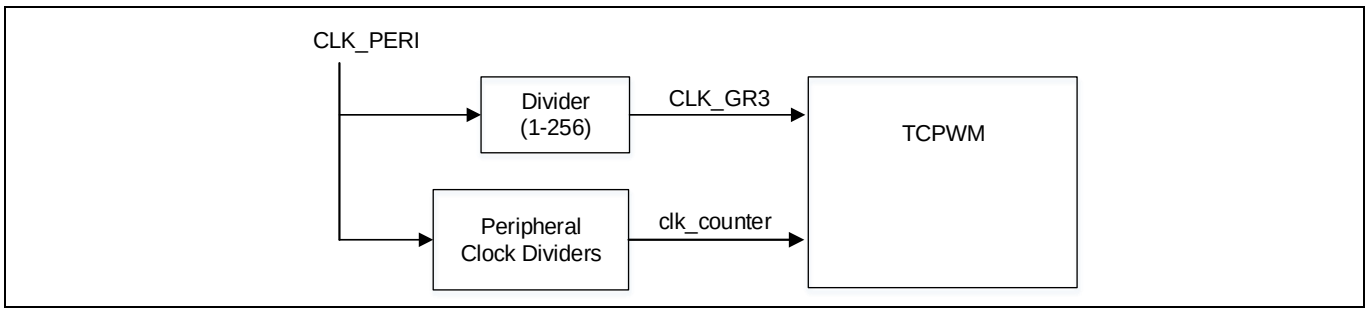

Figure 2 CYT2B7 Series: TCPWM and Clock

Introduction

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 a counter. This clock is generated by Peripheral Clock Dividers.

Figure 3 shows Peripheral Clock Dividers block diagram.

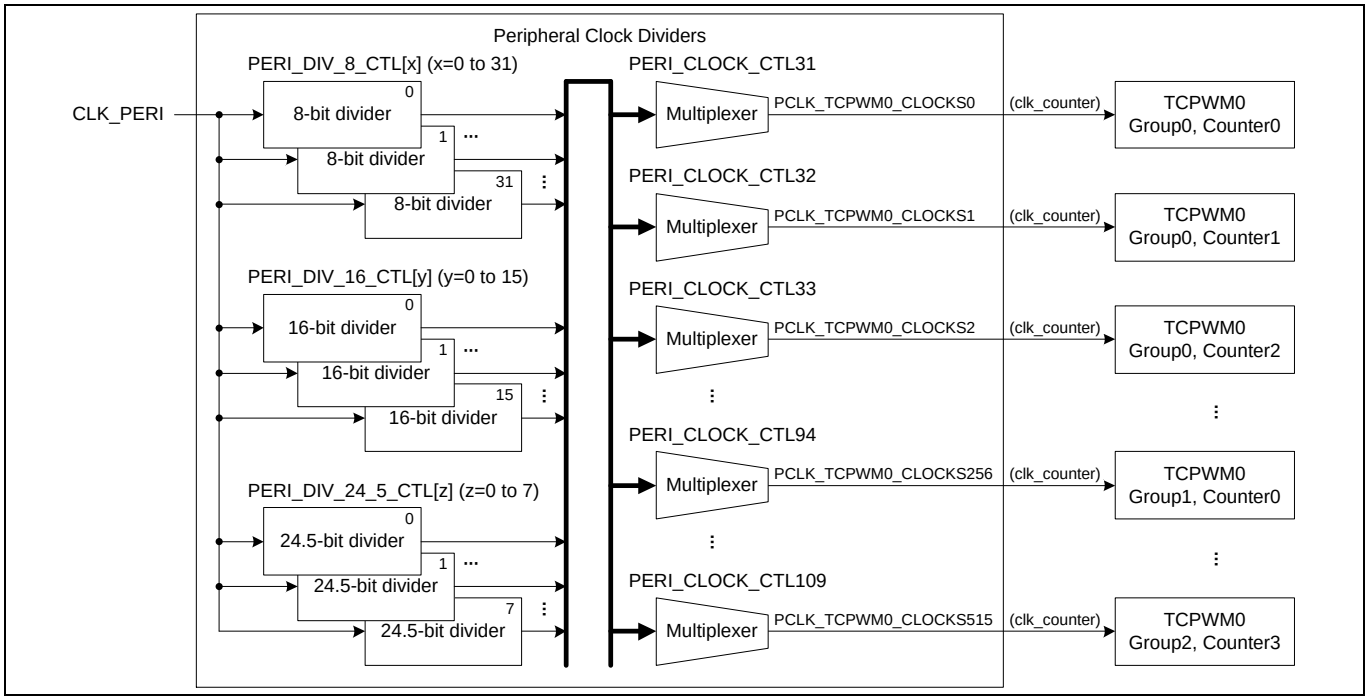

Figure 3 CYT2B7 Series: Peripheral Clock Dividers

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 channels available for each device.

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

The output of Peripheral Clock Divider, clk_counter, is included in 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 TCPWM of CYT2B7 series. For other series, see the "Peripheral Clocks" section in the **device datasheet**.

Introduction

Table 2 CYT2B7 Series: Peripheral Clock Number in TCPWM

Peripheral Clock Number	Description
PERI_CLOCK_CTL 31 to 93	TCPWM group #0, 16-bit counter #0 to #62 (63 ch)
PERI_CLOCK_CTL 94 to 105	TCPWM group #1, 16-bit counter for motor #0 to #11 (12 ch)
PERI_CLOCK_CTL 106 to 109	TCPWM group #2, 32-bit counter #0 to #3 (4 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 Sample Driver Library (SDL). The code snippets in this application note are part of SDL. See **Other References** for the SDL.

SDL has a configuration part and a driver part. The configuration part configures the parameter values for the desired operation. The driver part configures each register based on the parameter values in the configuration part. You can configure the configuration part according to your system.

2.1 Timer Mode

This section describes how to set up Timer Mode.

Timer Mode is for a basic counter application. This is ordinary counter usage to count the clock for 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 mode in the upward and downward directions

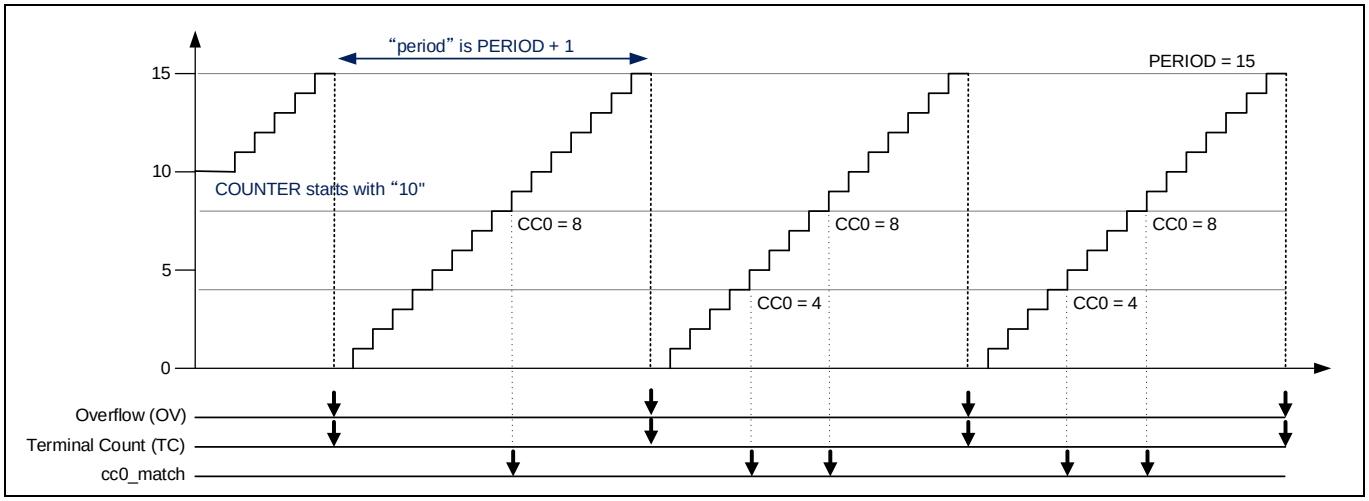

Figure 4 Timer Mode in Upward Counting Mode

Counter starts from initial value. Configured counter register (e.g., TCPWM0_GRP0_CNT0_CONTER) as COUNTER = 10, then counter starts with 10.

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 (e.g., 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.

TCPWM Operation Examples

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

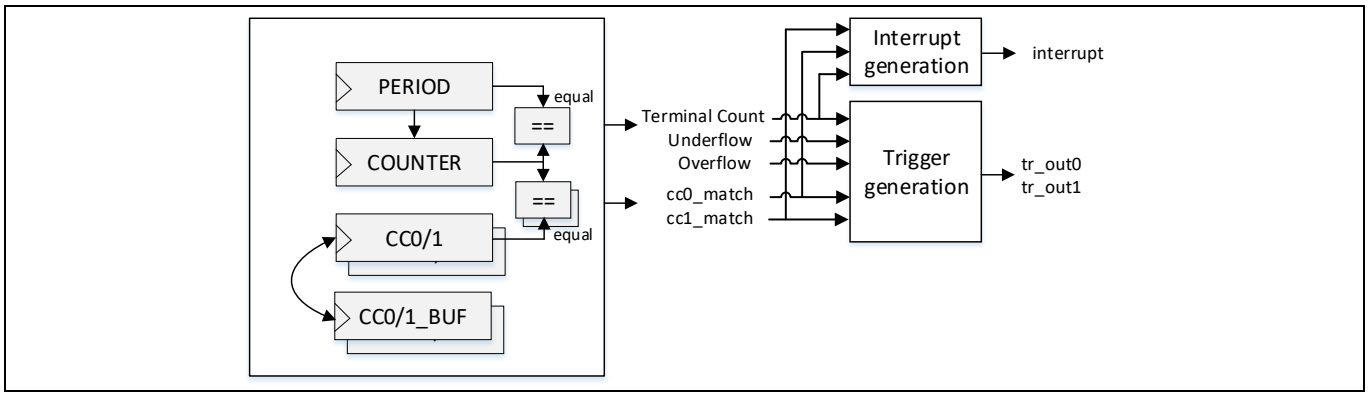

Figure 5 Timer Functionality

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 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 15,625-Hz counter clock. The following is an example of configuring the TCPWM using SDL.

- TCPWM Operation Mode : Timer Mode
- Using Counter : TCPWM0/Group0/Counter0
- Counter Start operation : Start by Software
- Input Clock : clk_counter = 2 MHz
  : Divide Value = Divided by 128 (2 MHz / 128 = 15,625 Hz)
- Interrupt Period : 1 sec (15,625*(1/15,625 Hz) = 1 sec)
- System Interrupt source : TCPWM0/Group0/Counter0 (IDX: 274)
- Mapped to CPU Interrupt : IRQ3
- CPU Interrupt Priority : 3

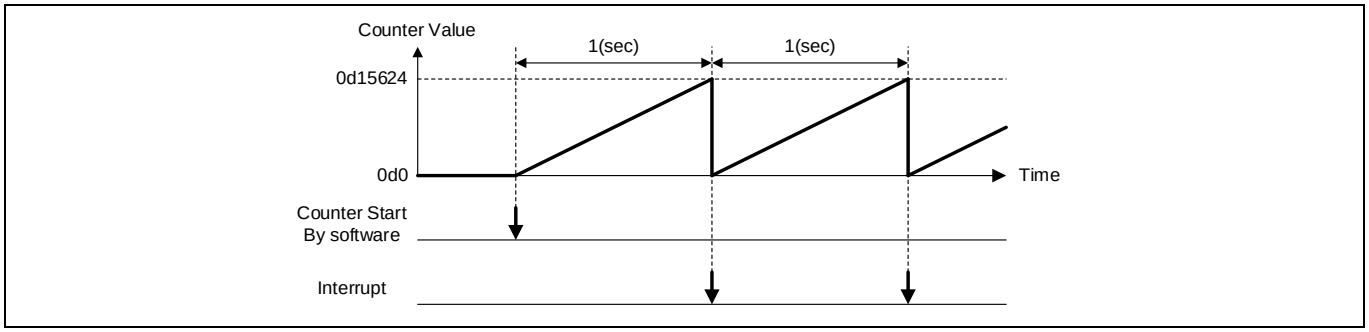

Figure 6 Timing chart of the timer mode

Figure 7 shows the operation flow of this use case.

TCPWM Operation Examples

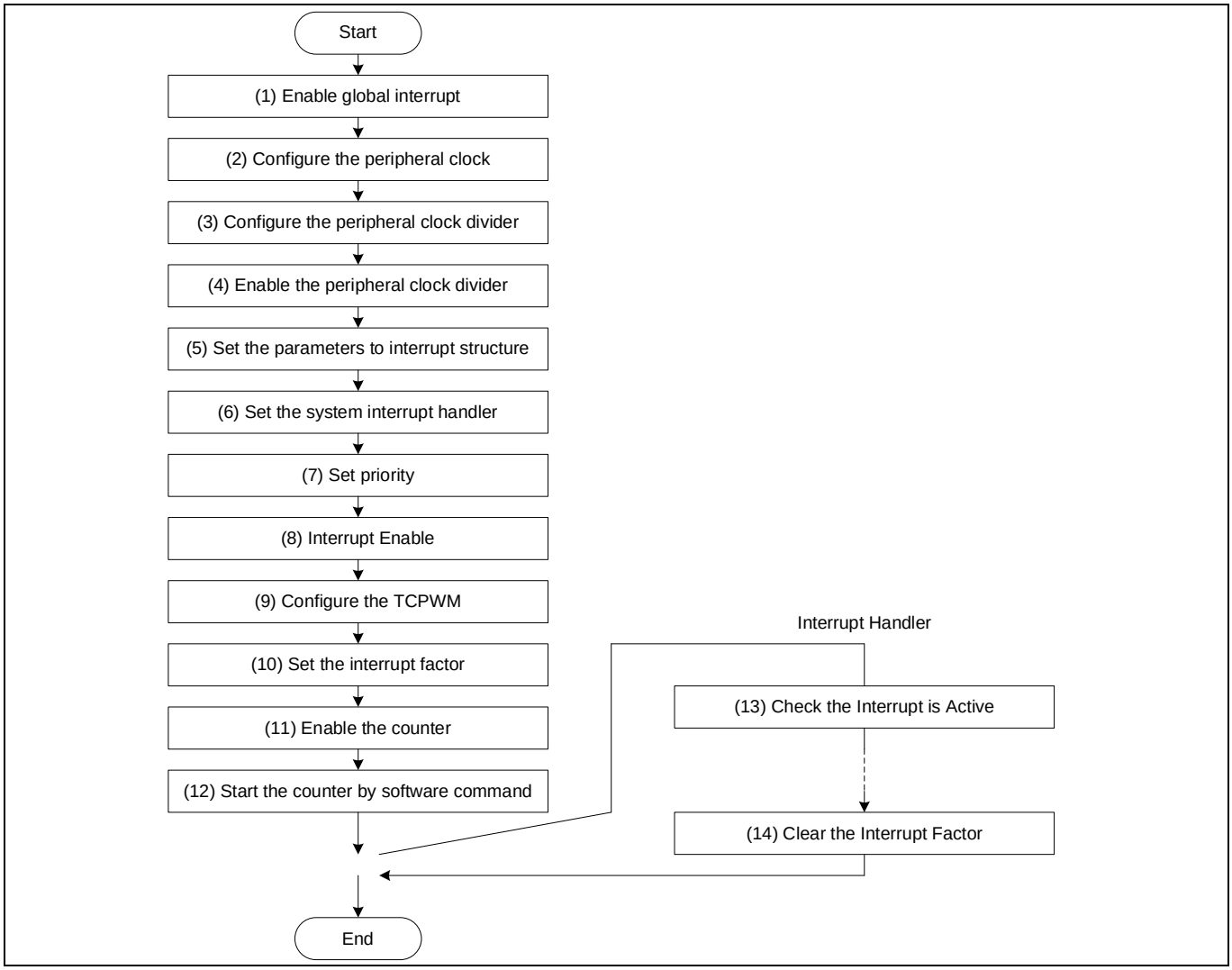

Figure 7 Operation Flow Example

(1) Enable Global Interrupt (CPU Interrupt Enable). For details, see the CPU interrupt handing sections in the **Architecture TRM**.

(2) Configure the peripheral clocks for the TCPWM.

(3) Configure the peripheral clock dividers for the TCPWM.

(4) Enable the peripheral clock divider for the TCPWM.

(5) Set Interrupt Structure. For details, see the CPU interrupt handing sections in the **Architecture TRM**.

(6) Set the system interrupt handler. For details, see the CPU interrupt handing sections in the **Architecture TRM**.

(7) Set priority by the NVIC Priority Register. For details, see the CPU interrupt handing sections in the **Architecture TRM**.

(8) Enable the Interrupt by the NVIC Interrupt Controller. For details, see the CPU interrupt handing sections in the **Architecture TRM**.

(9) Configure the TCPWM.

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

(10) Set the interrupt factor for the TCPWM.

(11) Enable the TCPWM counter.
(12) Start the TCPWM counter by software command.
(13) When an interrupt occurs, check the interrupt is Active.
(14) After executing the interrupt process, the interrupt factor is cleared.

2.1.2 Configuration and Example

Table 3 lists the parameters of the configuration part in SDL for Timer Mode.

Table 3 CYT2 Series: List of Timer Mode configuration Parameters

Parameters	Description	Setting Value
For CLK		
TCPWMx_GRPx_CNTx_COUNTER	Using Counter Number	TCPWM0_GRP0_CNT0 (TCPWM0/Group0/Counter0)
PCLK_TCPWMx_CLOCKSx_COUNTER	Peripheral Clock Number	PCLK_TCPWM0_CLOCKS0 (PERI_CLOCK_CTL31)
TCPWM_PERI_CLK_DIVIDER_NO_COUNTER	Using Divider Number	0x0 (Divider 0)
periFreq	Frequency of peripheral clock	80000000ul (80MHz)
targetFreq	Frequency of clk_counter	2000000ul (2MHz)
For TCPWM		
.period	Value of the counter (This field should be set to “n-1”)	0d15624
.clockPrescaler	Pre-scaling of the selected counter clock	CY_TCPWM_COUNTER_PRESCALER_DIVBY_128 (0x7)
.runMode	Counter run mode	CY_TCPWM_PWM_CONTINUOUS (0x0)
.countDirection	Counter direction	CY_TCPWM_COUNTER_COUNT_UP (0x0)
.debug_pause	Counter behavior in debug mode	false (0x0)
.CompareOrCapture	Counter mode	CY_TCPWM_COUNTER_MODE_COMPARE (0x0)
.compare0	Compared to counter value for CC0	0x0000
.compare0_buff	Additional compared to counter value for CC0	0x0000
.compare1	Compared to counter value for CC1	0x0000
.compare1_buff	Additional compared to counter value for CC1	0x0000
.enableCompare0Swap	Exchange the CC0 and buffered CC0 values	false (0x0)

TCPWM Operation Examples

Parameters	Description	Setting Value
.enableCompare1Swap	Exchange the CC1 and buffered CC1 values	false (0x0)
.interruptSources	Interrupt Mask bit	0x0 (zero clear)
.capture0InputMode	Capture0 edge mode	0x3 (NO_EDGE_DET)
.capture0Input	Input triggers for capture0	0x0 (Constant 0)
.reloadInputMode	Reload edge mode	0x3 (NO_EDGE_DET)
.reloadInput	Input triggers for reload	0x0 (Constant 0)
.startInputMode	Start edge mode	0x3 (NO_EDGE_DET)
.startInput	Input triggers for start	0x0 (Constant 0)
.stopInputMode	Stop edge mode	0x3 (NO_EDGE_DET)
.stopInput	Input triggers for stop	0x0 (Constant 0)
.capture1InputMode	Capture1 edge mode	0x3 (NO_EDGE_DET)
.capture1Input	Input triggers for capture1	0x0 (Constant 0)
.countInputMode	Count edge mode	0x3 (NO_EDGE_DET)
.countInput	Input triggers for count	0x1 (Constant 1)
.trigger1	Internal events to generate the output trigger 0	CY_TCPWM_COUNTER_OVERFLOW (0x0)
For Interrupt		
irq_cfg.sysIntSrc	System interrupt index number	tcpwm_0_interrupts_0_IRQn
irq_cfg.intIdx	CPU interrupt number	CPUIntIdx3_IRQn
.isEnabled	CPU interrupt enable	true (0x1)

Code Listing 1 demonstrates an example program to configure timer mode in the configuration part.

Code Listing 1 CYT2 Series: Example to configure Timer Mode in Configuration Part

```
#define TCPWMx_GRPx_CNTx_COUNTER          TCPWM0_GRP0_CNT0
#define PCLK_TCPWMx_CLOCKSx_COUNTER       PCLK_TCPWM0_CLOCKS0
#define TCPWM_PERI_CLK_DIVIDER_NO_COUNTER 0ul


cy_stc_tcpwm_counter_config_t const MyCounter_config =
{
    .period             = 15625ul - 1ul,                  // 15,625 / 15625 = 1s
    .clockPrescaler    = CY_TCPWM_COUNTER_PRESCALER_DIVBY_128,  // 2,000,000Hz / 128 = 15,625Hz
    .runMode            = CY_TCPWM_PWM_CONTINUOUS,
    .countDirection     = CY_TCPWM_COUNTER_COUNT_UP,
    .debug_pause        = 0ul,
    .CompareOrCapture   = CY_TCPWM_COUNTER_MODE_COMPARE,
    .compare0           = 0ul,
    .compare0_buff      = 0ul,
    .compare1           = 0ul,
    .compare1_buff      = 0ul,
    .enableCompare0Swap = false,
    .enableCompare1Swap = false,
    .interruptSources   = 0ul,
    .capture0InputMode  = 3ul,
    .capture0Input      = 0ul,
    .reloadInputMode    = 3ul,
    .reloadInput        = 0ul,
    .startInputMode     = 3ul,
    .startInput         = 0ul,
    .stopInputMode      = 3ul,
    .stopInput          = 0ul,
```

Define Using Counter
Define Peripheral Clock
Define Peripheral Clock Divider
Configure the counter parameters

TCPWM Operation Examples

Code Listing 1 CYT2 Series: Example to configure Timer Mode in Configuration Part

```
    .capture1InputMode  = 3ul,
    .capture1Input      = 0ul,
    .countInputMode     = 3ul,
    .countInput         = 1ul,
    .trigger1           = CY_TCPWM_COUNTER_OVERFLOW,
};

cy_stc_sysint_irq_t irq_cfg =
{
    .sysIntSrc  = tcpwm_0_interrupts_0_IRQn,
    .intIdx     = CPUIntIdx3_IRQn,
    .isEnabled  = true,
};

int main(void)
{
    :
    __enable_irq(); /* Enable global interrupts. */

    /* Assign a programmable divider for TCPWM0_GRP0_CNT0 */
    uint32_t periFreq   = 80000000ul;
    uint32_t targetFreq = 2000000ul;
    uint32_t divNum = (periFreq / targetFreq);
    Cy_SysClk_PeriphAssignDivider(PCLK_TCPWMx_CLOCKSx_COUNTER, CY_SYSCLK_DIV_16_BIT,
TCPWM_PERI_CLK_DIVIDER_NO_COUNTER);

    /* Sets the 16-bit divider */
    Cy_SysClk_PeriphSetDivider(CY_SYSCLK_DIV_16_BIT, TCPWM_PERI_CLK_DIVIDER_NO_COUNTER,(divNum-1ul));
    Cy_SysClk_PeriphEnableDivider((cy_en_divider_types_t)CY_SYSCLK_DIV_16_BIT, TCPWM_PERI_CLK_DIVIDER_NO_COUNTER);

    /* Configure Interrupt for TCPWMs */
    Cy_SysInt_InitIRQ(&irq_cfg);
    Cy_SysInt_SetSystemIrqVector(irq_cfg.sysIntSrc, Timer_Handler);

    /* Set the Interrupt Priority & Enable the Interrupt */
    NVIC_SetPriority(irq_cfg.intIdx, 3ul);
    NVIC_EnableIRQ(irq_cfg.intIdx);

    /* Initialize TCPWM0_GPR0_CNT0 as Timer/Counter & Enable */
    Cy_Tcpwm_Counter_Init(TCPWMx_GRPx_CNTx_COUNTER, &MyCounter_config);
    Cy_Tcpwm_Counter_SetTC_IntrMask(TCPWMx_GRPx_CNTx_COUNTER);
    Cy_Tcpwm_Counter_Enable(TCPWMx_GRPx_CNTx_COUNTER);
    Cy_Tcpwm_TriggerStart(TCPWMx_GRPx_CNTx_COUNTER);

    :
    for(;;);
}
```

Configure interrupt structure parameters*1

(1)Enable global interrupt*1

Calculation of division ratio

(2)Configure the Peripheral Clock (See)

(3)Configure the integer division of the 16-bit divider (See **Code Listing 4**)

(4)Enable the 16-bit divider (See **Code Listing 5**)

(5)Set the parameters to interrupt structure*1

(6)Set the system interrupt handler*1

(7)Set priority*1

(8)Interrupt Enable*1

(9)Configure the counter based on above structure (See **Code Listing 6**)

(10)Set the interrupt factor*2 (See **Code Listing 7**)

(11)Enable the counter(See **Code Listing 8**)

(12)Start the counter (See **Code Listing 9**)

*1: For details, refer to the CPU interrupt handing sections in the **Architecture TRM**.

*2: TC (Terminal Count): A TC event is the logical OR of the counter underflow and overflow events

Code Listing 2 demonstrates an example of Interrupt Handler.

Code Listing 2 Example of Interrupt Handler

```
void Timer_Handler(void)
{
    if(Cy_Tcpwm_Counter_GetTC_IntrMasked(TCPWMx_GRPx_CNTx_COUNTER) == 1ul)
    {
    :
        Cy_Tcpwm_Counter_ClearTC_Intr(TCPWMx_GRPx_CNTx_COUNTER);
    }
}
```

Interrupt handler

(13)Check if Interrupt is Active (See **Code Listing 10**)

(14)Clear TC interrupt (See **Code Listing 11**)

Code Listing 3 to **Code Listing 5** demonstrate an example program to configure the CLK in the driver part.

The following description will help you understand the register notation of the driver part of SDL:

- PERI->unCLOCK_CTL and unDIV is the PERI_CLOCK_CTLx register mentioned in the Registers TRM. Other registers are also described in the same manner. “x” signifies the system interrupt index number.

TCPWM Operation Examples

- Performance improvement measures:
  To improve the performance of setting a register, the SDL writes a complete 32-bit data to the register. Each bit field is generated in advance in a bit-writable buffer and written to the register as the final 32-bit data.
- See "cyip_srss_v2.h" and “cyip_tcpwm_v2.h” under hdr/rev_x/ip for more information on the union and structure representation of registers.

Code Listing 3 CYT2 Series: Example to Configure the CLK in Driver Part (Cy_SysClk_PeriphAssignDivider)

```
/*******************************************************************************
* Function Name: Cy_SysClk_PeriphAssignDivider
*******************************************************************************/
__STATIC_INLINE cy_en_sysclk_status_t Cy_SysClk_PeriphAssignDivider(en_clk_dst_t ipBlock, cy_en_divider_types_t
dividerType, uint32_t dividerNum)
{
    if(Cy_SysClk_CheckDividerExisting(dividerType, dividerNum) == CY_DIVIDER_NOT_EXISTING)
    {
        return CY_SYSCLK_BAD_PARAM;
    }
    un_PERI_CLOCK_CTL_t tempCLOCK_CTL_RegValue;
    tempCLOCK_CTL_RegValue.u32Register = PERI->unCLOCK_CTL[ipBlock].u32Register;
    tempCLOCK_CTL_RegValue.stcField.u2TYPE_SEL = dividerType;
    tempCLOCK_CTL_RegValue.stcField.u8DIV_SEL = dividerNum;
    PERI->unCLOCK_CTL[ipBlock].u32Register = tempCLOCK_CTL_RegValue.u32Register;

    return CY_SYSCLK_SUCCESS;
}
```

Configure the Peripheral Clock

Check if configuration parameter values are valid.

Code Listing 4 CYT2 Series: Example to Configure the CLK in Driver Part (Cy_SysClk_PeriphSetDivider)

```
/*******************************************************************************
* Function Name: Cy_SysClk_PeriphSetDivider
*******************************************************************************/
__STATIC_INLINE cy_en_sysclk_status_t
Cy_SysClk_PeriphSetDivider(cy_en_divider_types_t dividerType, uint32_t dividerNum, uint32_t dividerValue)
{
    if (Cy_SysClk_CheckDividerExisting(dividerType, dividerNum) == CY_DIVIDER_NOT_EXISTING)
    {
        return CY_SYSCLK_BAD_PARAM;
    }
    if (dividerType == CY_SYSCLK_DIV_8_BIT)
    {
        if (dividerValue <= (PERI_DIV_8_CTL_INT8_DIV_Msk >> PERI_DIV_8_CTL_INT8_DIV_Pos))
        {
            PERI->unDIV_8_CTL[dividerNum].stcField.u8INT8_DIV = dividerValue;
        }
        else
        {
            return CY_SYSCLK_BAD_PARAM;
        }
    }
    else if (dividerType == CY_SYSCLK_DIV_16_BIT)
    {
        if (dividerValue <= (PERI_DIV_16_CTL_INT16_DIV_Msk >> PERI_DIV_16_CTL_INT16_DIV_Pos))
        {
            PERI->unDIV_16_CTL[dividerNum].stcField.u16INT16_DIV =  dividerValue;
        }
        else
        {
            return CY_SYSCLK_BAD_PARAM;
        }
    }
    else
    {
    /* return bad parameter */
        return CY_SYSCLK_BAD_PARAM;
    }
    return CY_SYSCLK_SUCCESS;
}
```

Configure the Peripheral Clock Divider

Check if configuration parameter values are valid.

Check the dividerType

Select INT8_DIV bits

Select INT16_DIB bits

TCPWM Operation Examples

Code Listing 5 CYT2 Series: Example to Configure the CLK in Driver Part (Cy_SysClk_PeriphEnableDivider)

```
/*******************************************************************************
* Function Name: Cy_SysClk_PeriphEnableDivider
*******************************************************************************/
__STATIC_INLINE cy_en_sysclk_status_t
Cy_SysClk_PeriphEnableDivider(cy_en_divider_types_t dividerType, uint32_t dividerNum)
{
    if(Cy_SysClk_CheckDividerExisting(dividerType, dividerNum) == CY_DIVIDER_NOT_EXISTING)
    {
        return CY_SYSCLK_BAD_PARAM;
    }
    /*specify the divider, make the reference = clk_peri, and enable the divider*/
    un_PERI_DIV_CMD_t tempDIV_CMD_RegValue;
    tempDIV_CMD_RegValue.u32Register = PERI->unDIV_CMD.u32Register;
    tempDIV_CMD_RegValue.stcField.u1ENABLE = 1ul;
    tempDIV_CMD_RegValue.stcField.u2PA_TYPE_SEL = 3ul;
    tempDIV_CMD_RegValue.stcField.u8PA_DIV_SEL = 0xFFul;
    tempDIV_CMD_RegValue.stcField.u2TYPE_SEL = dividerType;
    tempDIV_CMD_RegValue.stcField.u8DIV_SEL = dividerNum;
    PERI->unDIV_CMD.u32Register = tempDIV_CMD_RegValue.u32Register;
    (void)PERI->unDIV_CMD; /* dummy read to handle buffered writes */
    return CY_SYSCLK_SUCCESS;
}
```

Enable the Peripheral Clock Divider

Check if configuration parameter values are valid.

Code Listing 6 to **Code Listing 11** demonstrates an example program to configure the TCPWM in the driver part.

Code Listing 6 CYT2 Series: Example to Configure the TCPWM in Driver Part (Cy_Tcpwm_Counter_Init)

```
/*******************************************************************************
* Function Name: Cy_Tcpwm_Counter_Init
*******************************************************************************/
uint32_t Cy_Tcpwm_Counter_Init(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM, cy_stc_tcpwm_counter_config_t const *config)
{
    uint32_t status = CY_RET_BAD_PARAM;
    if (config->trigger1 > 0x04ul || config->trigger2 > 0x04ul)
    {
        return status;
    }
    if ((NULL != ptscTCPWM) && (NULL != config))
    {
        ptscTCPWM->unCTRL.stcField.u1ONE_SHOT = config->runMode;
        ptscTCPWM->unCTRL.stcField.u3MODE = config->CompareOrCapture;
        ptscTCPWM->unCTRL.stcField.u2UP_DOWN_MODE = config-> countDirection;
        ptscTCPWM->unCTRL.stcField.u1DBG_FREEZE_EN = config->debug_pause;
        ptscTCPWM->unCTRL.stcField.u1AUTO_RELOAD_CC0 = config->enableCompare0Swap;
        ptscTCPWM->unDT.stcField.u8DT_LINE_OUT_L = config->clockPrescaler;
        if (CY_TCPWM_COUNTER_COUNT_UP == config->runMode)
        {
            ptscTCPWM->unCOUNTER.u32Register = CY_TCPWM_CNT_UP_INIT_VAL;
        }
        else if (CY_TCPWM_COUNTER_COUNT_DOWN == config->runMode)

        {
            ptscTCPWM->unCOUNTER.u32Register = config->period;
        }
        else
        {
            ptscTCPWM->unCOUNTER.u32Register = CY_TCPWM_CNT_UP_DOWN_INIT_VAL;
        }
        ptscTCPWM->unCC0.u32Register = config->compare0;
        ptscTCPWM->unCC0_BUFF.u32Register = config->compare0_buff;
        ptscTCPWM->unPERIOD.u32Register = config->period;
        ptscTCPWM->unTR_IN_SEL0.stcField.u8CAPTURE0_SEL = config->capture0Input;
        ptscTCPWM->unTR_IN_SEL0.stcField.u8RELOAD_SEL = config->reloadInput;
        ptscTCPWM->unTR_IN_SEL0.stcField.u8STOP_SEL = config->stopInput;
        ptscTCPWM->unTR_IN_SEL0.stcField.u8COUNT_SEL = config->countInput;
        ptscTCPWM->unTR_IN_SEL1.stcField.u8START_SEL = config->startInput;
        ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2CAPTURE0_EDGE = config->capture0InputMode;
        ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2RELOAD_EDGE = config->reloadInputMode;
        ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2START_EDGE = config->startInputMode;
        ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2STOP_EDGE = config->stopInputMode;
        ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2COUNT_EDGE = config->countInputMode;
        ptscTCPWM->unTR_OUT_SEL.stcField.u3OUT0 = config->trigger1;
        ptscTCPWM->unTR_OUT_SEL.stcField.u3OUT1 = config->trigger2;
        ptscTCPWM->unINTR_MASK.u32Register = config->interruptSources;
        ptscTCPWM->unCTRL.stcField.u1AUTO_RELOAD_CC1 = config->enableCompare1Swap;
        ptscTCPWM->unCC1.u32Register = config->compare1;
```

Configure (Initialize) the counter

Check if configuration parameter values are valid.

The initial value of the counter is determined according to each countDirection.

TCPWM Operation Examples

Code Listing 6 CYT2 Series: Example to Configure the TCPWM in Driver Part (Cy_Tcpwm_Counter_Init)

```
        ptscTCPWM->unCC1_BUFF.u32Register = config->compare1_buff;
        ptscTCPWM->unTR_IN_SEL1.stcField.u8CAPTURE1_SEL = config->capture1Input;
        ptscTCPWM->unTR_IN_EDGE_SEL.stcField.u2CAPTURE1_EDGE = config->capture1InputMode;
        status = CY_RET_SUCCESS;
    }
    return(status);
}
```

Code Listing 7 CYT2 Series: Example to Configure the TCPWM in Driver Part (Cy_Tcpwm_Counter_SetTC_IntrMask)

```
/*******************************************************************************
* Function Name: Cy_Tcpwm_Counter_SetTC_IntrMask
*******************************************************************************/
void Cy_Tcpwm_Counter_SetTC_IntrMask(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
    ptscTCPWM->unINTR_MASK.stcField.u1TC = 1ul;
}
```

Configure the interrupt factor

Code Listing 8 CYT2 Series: Example to configure the TCPWM in Driver Part (Cy_Tcpwm_Counter_Enable)

```
/*******************************************************************************
* Function Name: Cy_Tcpwm_Counter_Enable
*******************************************************************************/
void Cy_Tcpwm_Counter_Enable(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
    ptscTCPWM->unCTRL.stcField.u1ENABLED = 0x01ul;
}
```

Enable the counter

Code Listing 9 CYT2 Series: Example to configure the TCPWM in Driver Part (Cy_Tcpwm_TriggerStart)

```
/*******************************************************************************
* Function Name: Cy_Tcpwm_TriggerStart
*******************************************************************************/
void Cy_Tcpwm_TriggerStart(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
    ptscTCPWM->unTR_CMD.stcField.u1START = 0x01ul;
}
```

Start the counter

Code Listing 10 CYT2 series: Example to configure TCPWM in Driver Part (Cy_Tcpwm_Counter_GetTC_IntrMasked)

```
/*******************************************************************************
* Function Name: Cy_Tcpwm_Counter_GetTC_IntrMasked
*******************************************************************************/
uint8_t Cy_Tcpwm_Counter_GetTC_IntrMasked(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
    return (uint8_t)(ptscTCPWM->unINTR_MASKED.stcField.u1TC);
}
```

Get TC interrupt masked

Code Listing 11 CYT2 series: Example to configure TCPWM in Driver Part (Cy_Tcpwm_Counter_ClearTC_Intr)

```
/*******************************************************************************
* Function Name: Cy_Tcpwm_Counter_ClearTC_Intr
********************************************************************************/
void Cy_Tcpwm_Counter_ClearTC_Intr(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
    ptscTCPWM->unINTR.stcField.u1TC = 1ul;
    ptscTCPWM->unINTR.u32Register;
}
```

Clear TC interrupt

2.2 Capture Mode

This section describes how to set up Capture Mode.

This Capture Mode is for an application to catch the counter value depending on the input trigger.

Figure 8 shows the Capture Mode in upward counting mode.

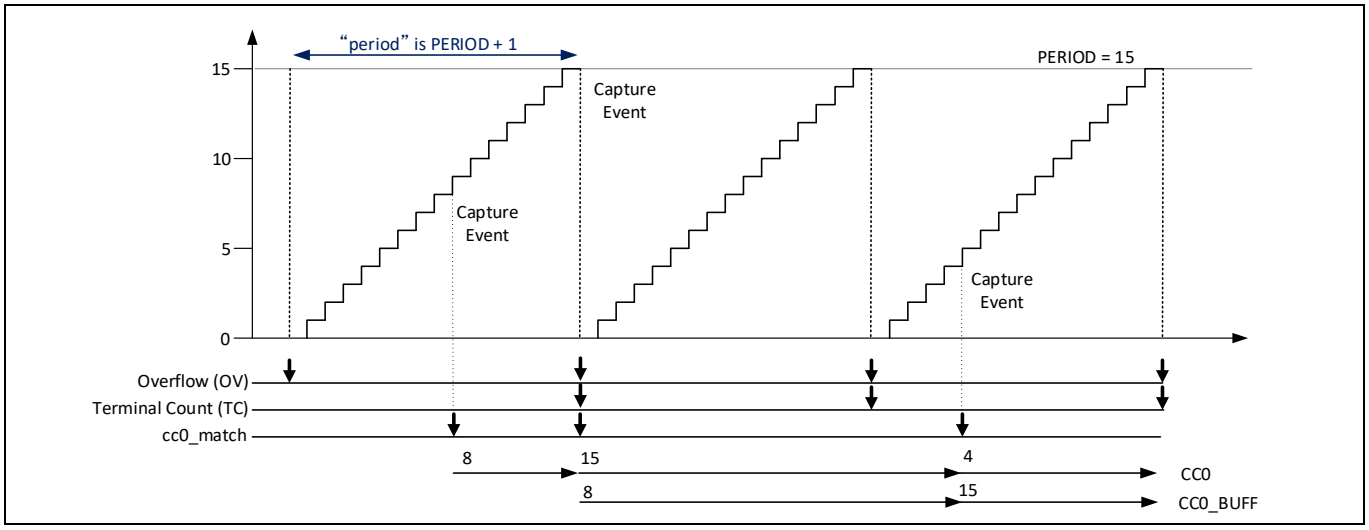

Figure 8 Capture Mode in Upward Counting Mode

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

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 channels available for each device.

Table 4 shows the input trigger sources of the 16-bit counter number 0 in the CYT2B7 series.

Table 4 CYT2B7 Series: Trigger Inputs List of 16-bit Counter Number 0

Trigger No.	Input Trigger	Input Trigger Sources
0	Constant 0	Always 0 input (Not counting)
1	Constant 1	Always 1 input (Counting with a clock)
2	HSIOM column ACT#2	TC_0_TR0 (External pin, P3.1 or P6.1)
3	HSIOM column ACT#3	TC_0_TR1 (External pin, P3.2 or P6.2)

TCPWM Operation Examples

Trigger No.	Input Trigger	Input Trigger Sources
:		
31	tr_all_cnt_in[26]	MAX Group 4 of One-to-one Trigger

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 SDL.

- TCPWM Operation Mode : Capture Mode
- Using Counter : TCPWM0/Group0/Counter0
- Counter Start operation : Start by Software
- Input Clock :
  - clk_counter = 2 MHz
  - Divide Value = Divided by 4 (2 MHz / 4 = 500 kHz)
- Used I/O port as capture0 event : TC_0_TR0/1 (External Pin)
- Interrupt : When the event of capture0/1 occurs.
- System Interrupt source : TCPWM0/Group0/Counter0 (IDX: 274)
- Mapped to CPU Interrupt : IRQ3
- CPU Interrupt Priority : 3

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

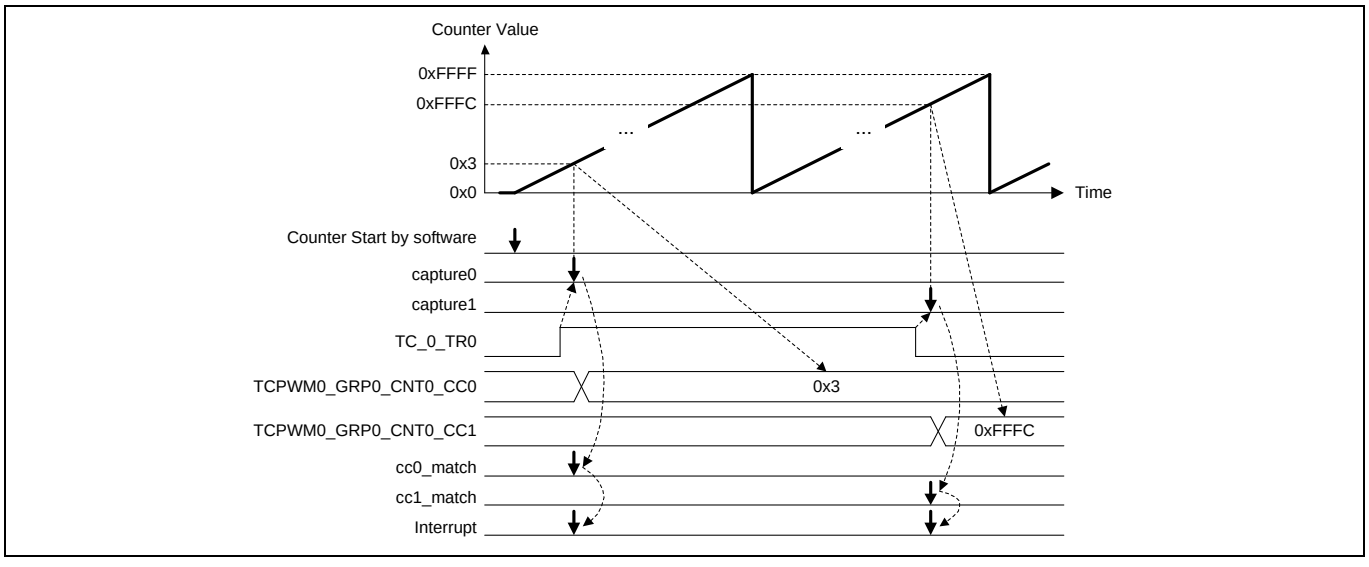

Figure 9 Timing Chart of the Capture Mode

TCPWM Operation Examples

Note: The capture0/1 signal is generated by the input of TC_0_TR0. Note that the input from TC0_0_TR0 is not always the counter value in the figure.

Figure 10 shows the operation flow of this use case.

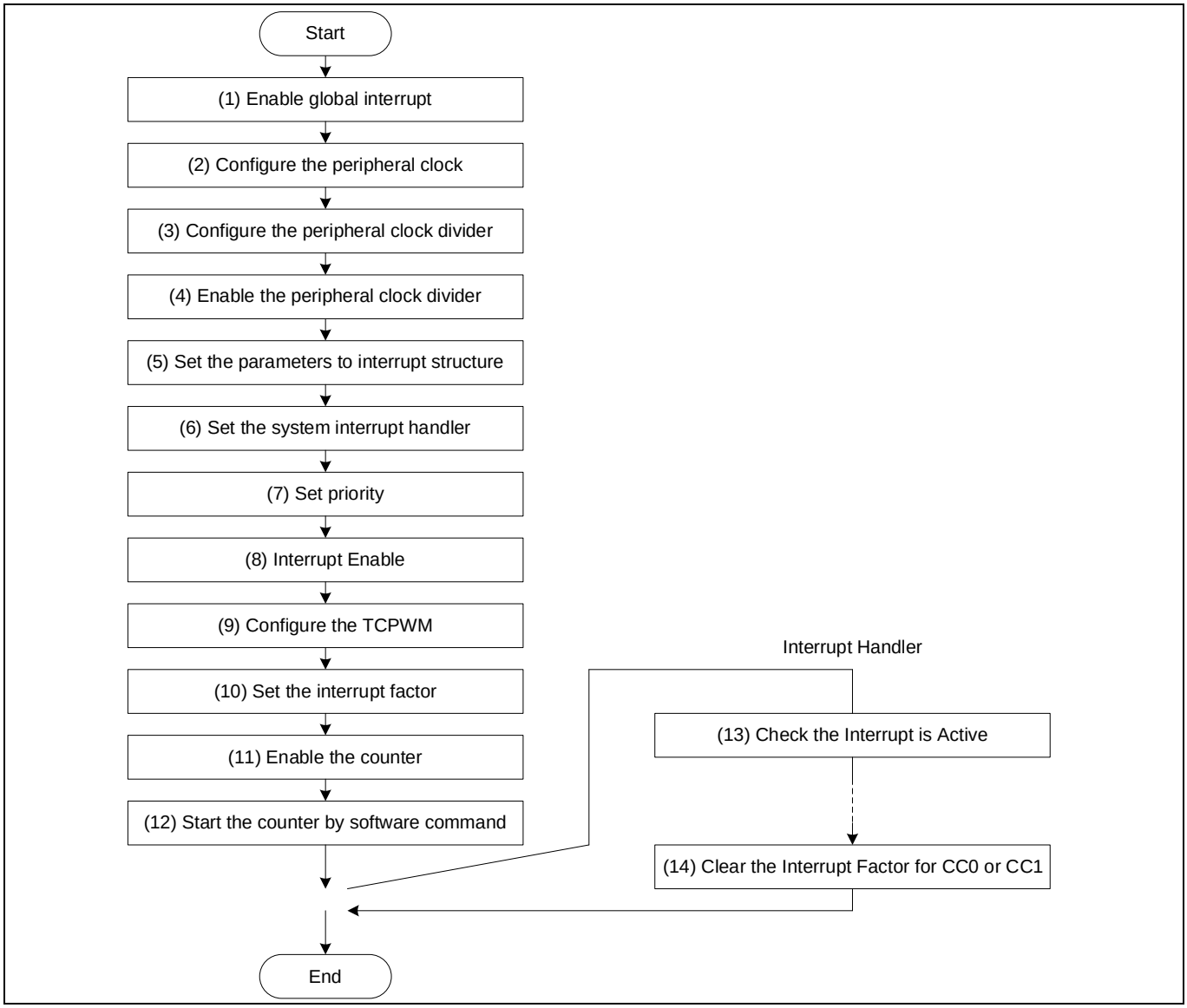

Figure 10 Operation Flow Example

(1) Enable Global Interrupt (CPU Interrupt Enable). For details, see the CPU interrupt handing sections in the **Architecture TRM**.
(2) Configure the peripheral clocks for the TCPWM.
(3) Configure the peripheral clock dividers for the TCPWM.
(4) Enable the peripheral clock divider for the TCPWM.
(5) Set Interrupt Structure. For details, see the CPU interrupt handing sections in the **Architecture TRM**.
(6) Set the system interrupt handler. For details, see the CPU interrupt handing sections in the **Architecture TRM**.
(7) Set priority by the NVIC Priority Register. For details, see the CPU interrupt handing sections in the **Architecture TRM**.

(8) Enable the Interrupt by the NVIC Interrupt Controller. For details, see the CPU interrupt handing sections in the **Architecture TRM**.

(9) Configure the TCPWM.

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

(10) Set the interrupt factor for the TCPWM.

(11) Enable the TCPWM counter.

(12) Start the TCPWM counter by software command.

(13) When an interrupt occurs, check the Interrupt is Active.

(14) After executing the interrupt process, the interrupt factor (CC0 or CC1) is cleared.

2.2.2 Configuration and Example

Table 5 lists the parameters of the configuration part in SDL for Capture Mode.

Table 5 CYT2 Series: List of Timer Mode Configuration Parameters

Parameters	Description	Setting Value
For CLK		
TCPWMx_GRPx_CNTx_CAPTURE	Using Counter Number	TCPWM0_GRP0_CNT0
PCLK_TCPWMx_CLOCKSx_CAPTURE	Peripheral Clock Number	PCLK_TCPWM0_CLOCKS0
TR_ONE_CNT_NR_USE	Input Trigger	0x0 (Constant 0)
TCPWMx_PERI_CLK_DIVIDER_NO	Using Divider Number	0x0
periFreq	Frequency of peripheral clock	80000000ul (80MHz)
targetFreq	Frequency of clk_counter	2000000ul (2MHz)
For TCPWM		
.period	Upper value of the counter (This field should be set to “n-1”)	0xffff
.clockPrescaler	Pre-scaling of the selected counter clock	CY_TCPWM_COUNTER_PRESCALER_DIVBY_4 (0x2)
.runMode	Counter run mode	CY_TCPWM_PWM_CONTINUOUS (0x0)
.countDirection	Counter direction	CY_TCPWM_COUNTER_COUNT_UP (0x0)
.debug_pause	Counter behavior in debug mode	false (0x0)
.CompareOrCapture	Counter mode	CY_TCPWM_COUNTER_MODE_CAPTURE (0x2)
.compare0	Compared to counter value for CC0	0x0000
.compare0_buff	Additional compared to counter value for CC0	0x0000
.compare1	Compared to counter value for CC1	0x0000
.compare1_buff	Additional compared to counter value for CC1	0x0000

TCPWM Operation Examples

Parameters	Description	Setting Value
.enableCompare0Swap	Exchange the CC0 and buffered CC0 values	false (0x0)
.enableCompare1Swap	Exchange the CC1 and buffered CC1 values	false (0x0)
.interruptSources	Interrupt Mask bit	0x0 (zero clear)
.capture0InputMode	Capture0 edge mode	0x0 (RISING_EDGE)
.capture0Input	Input triggers for capture0	0x2 (HSIOM column ACT#2)
.reloadInputMode	Reload edge mode	0x3 (NO_EDGE_DET)
.reloadInput	Input triggers for reload	0x0 (Constant 0)
.startInputMode	Start edge mode	0x3 (NO_EDGE_DET)
.startInput	Input triggers for start	0x0 (Constant 0)
.stopInputMode	Stop edge mode	0x3 (NO_EDGE_DET)
.stopInput	Input triggers for stop	0x0 (Constant 0)
.capture1InputMode	Capture1 edge mode	0x1 (FALLING_EDGE)
.capture1Input	Input triggers for capture1	0x2 (HSIOM column ACT#2)
.countInputMode	Count edge mode	0x3 (NO_EDGE_DET)
.countInput	Input triggers for count	0x1 (Constant 1)
.trigger1	Internal events to generate the output trigger 0	CY_TCPWM_COUNTER_CC0_MATCH (0x3)
For Interrupt		
irq_cfg.sysIntSrc	System interrupt index number	tcpwm_0_interrupts_0_IRQn
irq_cfg.intIdx	CPU interrupt number	CPUIntIdx3_IRQn
.isEnabled	CPU interrupt enable	true (0x1)

Code Listing 12 demonstrates an example program to configure the timer mode in the configuration part.

Code Listing 12 CYT2 Series: Example to configure Capture Mode in Configuration Part

```
#define TCPWMx_GRPx_CNTx_CAPTURE                TCPWM0_GRP0_CNT0
#define PCLK_TCPWMx_CLOCKSx_CAPTURE             PCLK_TCPWM0_CLOCKS0
#define TR_ONE_CNT_NR_USE                  0ul // from 0 to 2
#define TCPWMx_PERI_CLK_DIVIDER_NO         0ul

    :
cy_stc_tcpwm_counter_config_t const MyCounter_config =
    .period             = 0xFFFFul,     // TCPWM in GRP0 has 16 bit counter
    .clockPrescaler     = CY_TCPWM_COUNTER_PRESCALER_DIVBY_4,     // 2 MHz/4 = 500kHz
    .runMode            = CY_TCPWM_PWM_CONTINUOUS,
    .countDirection     = CY_TCPWM_COUNTER_COUNT_UP,
    .debug_pause        = false,
    .CompareOrCapture   = CY_TCPWM_COUNTER_MODE_CAPTURE,
    .compare0           = 0uL,
    .compare0_buff      = 0uL,
    .compare1           = 0uL,
    .compare1_buff      = 0uL,
    .enableCompare0Swap = false,
    .enableCompare1Swap = false,
    .interruptSources   = 0uL,
    .capture0InputMode  = 0uL,                  // detect rising edge
    .capture0Input      = TR_ONE_CNT_NR_USE+2uL, // 0: always "0". 1: always "1". x (above 2): HSIOM column
ACT#2[offset+x]
    .reloadInputMode    = 3uL,
    .reloadInput        = 0uL,
    .startInputMode     = 3uL,
```

Define Using Counter
Define Peripheral Clock
Define Input Trigger
Define Peripheral Clock Divider
Configure the counter parameters

TCPWM Operation Examples

Code Listing 12 CYT2 Series: Example to configure Capture Mode in Configuration Part

```
    .startInput         = 0uL,
    .stopInputMode      = 3uL,
    .stopInput          = 0uL,
    .capture1InputMode  = 1uL,                    // detect falling edge
    .capture1Input      = TR_ONE_CNT_NR_USE+2uL, // 0: always "0". 1: always "1". x (above 2): HSIOM column
ACT#3[offset+x]
    .countInputMode     = 3uL,
    .countInput         = 1uL,
    .trigger1           = CY_TCPWM_COUNTER_CC0_MATCH,
}

cy_stc_sysint_irq_t irq_cfg =
{
    .sysIntSrc  = tcpwm_0_interrupts_0_IRQn,
    .intIdx     = CPUIntIdx3_IRQn,
    .isEnabled  = true,
};

int main(void)
{
    __enable_irq(); /* Enable global interrupts. */
    :
    uint32_t periFreq   = 80000000ul;
    uint32_t targetFreq = 2000000ul;
    uint32_t divNum = (periFreq / targetFreq);
    :
    /* Assign a programmable divider  for TCPWM0_GRPx_CNTx_COUNTER */
    Cy_SysClk_PeriphAssignDivider(PCLK_TCPWMx_CLOCKSx_CAPTURE, CY_SYSCLK_DIV_16_BIT, TCPWMx_PERI_CLK_DIVIDER_NO);


    /* Sets the 16-bit divider */
    Cy_SysClk_PeriphSetDivider(CY_SYSCLK_DIV_16_BIT, TCPWMx_PERI_CLK_DIVIDER_NO, (divNum - 1ul));

    /* Enable the divider */
    Cy_SysClk_PeriphEnableDivider(CY_SYSCLK_DIV_16_BIT, TCPWMx_PERI_CLK_DIVIDER_NO);

    /* Interrupt setting for Capture  */
    Cy_SysInt_InitIRQ(&irq_cfg);
    Cy_SysInt_SetSystemIrqVector(irq_cfg.sysIntSrc, capture_isr_handler);

    /* Set the Interrupt Priority & Enable the Interrupt */
    NVIC_SetPriority(irq_cfg.intIdx, 3ul);
    NVIC_EnableIRQ(irq_cfg.intIdx);

    :
    /* Initialize PCLK_TCPWM0_CLOCKSx_CAPTURE as Capture Mode & Enable */
    Cy_Tcpwm_Counter_Init(TCPWMx_GRPx_CNTx_CAPTURE, &MyCounter_config);
    Cy_Tcpwm_Counter_SetCC0_IntrMask(TCPWMx_GRPx_CNTx_CAPTURE);
    Cy_Tcpwm_Counter_SetCC1_IntrMask(TCPWMx_GRPx_CNTx_CAPTURE);
    Cy_Tcpwm_Counter_Enable(TCPWMx_GRPx_CNTx_CAPTURE);
    Cy_Tcpwm_TriggerStart(TCPWMx_GRPx_CNTx_CAPTURE);


    :
    for(;;);
}
```

Configure interrupt structure parameters*1

(1)Enable global interrupt*1

Calculation of division ratio

(2)Configure the Peripheral Clock (See **Code Listing 3**)

(3)Configure the integer division of the 16-bit divider (See **Code Listing 4**)

(4)Enable the 16-bit divider (See **Code Listing 5**)

(5)Set the parameters to interrupt structure*1

(6)Set the system interrupt handler*1

(7)Set priority*1

(8)Interrupt Enable*1

(9)Initialize the counter based on above structure (See **Code Listing 6**)

(10)Set the interrupt factor (See **Code Listing 14**)

(10)Set the interrupt factor (See **Code Listing 15**)

(11)Enable the counter (See **Code Listing 8**)

(12)Start the counter (See **Code Listing 9**)

*1: For details, refer to the CPU interrupt handing sections in the **Architecture TRM**.

Code Listing 13 demonstrates an example of the Interrupt Handler.

Code Listing 13 Example of Interrupt Handler

```
void capture_isr_handler(void)
{
    :
    if(Cy_Tcpwm_Counter_GetCC0_IntrMasked(TCPWMx_GRPx_CNTx_CAPTURE))
    {
        // CC0 would capture rising edge of input pulse
        Cy_Tcpwm_Counter_ClearCC0_Intr(TCPWMx_GRPx_CNTx_CAPTURE);
    }
    :
    if(Cy_Tcpwm_Counter_GetCC1_IntrMasked(TCPWMx_GRPx_CNTx_CAPTURE))
    {
        // CC1 would capture falling edge of input pulse
        Cy_Tcpwm_Counter_ClearCC1_Intr(TCPWMx_GRPx_CNTx_CAPTURE);
    }
}
```

Interrupt handler*1

(13)Check if Interrupt is Active for CC0 (See **Code Listing 16**)

(14)Clear TC interrupt for CC0 (See **Code Listing 17**)

(13)Check if Interrupt is Active for CC1 (See **Code Listing 18**)

(14)Clear TC interrupt for CC1 (See **Code Listing 19**)

TCPWM Operation Examples

Code Listing 14 to **Code Listing 19** demonstrate an example program to configure the TCPWM in the driver part.

Code Listing 14 CYT2 series: Example to configure TCPWM in Driver Part (Cy_Tcpwm_Counter_SetCC0_IntrMask)

```
/*******************************************************************************
* Function Name: Cy_Tcpwm_Counter_SetCC0_IntrMask
*******************************************************************************/
void Cy_Tcpwm_Counter_SetCC0_IntrMask(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM);
{
    ptscTCPWM->unINTR_MASK.stcField.u1CC0_MATCH = 0x1ul;
}
```

Configure the interrupt factor for CC0

Code Listing 15 CYT2 Series: Example to Configure the TCPWM in Driver Part (Cy_Tcpwm_Counter_SetCC1_IntrMask)

```
/*******************************************************************************
* Function Name: Cy_Tcpwm_Counter_SetCC1_IntrMask
*******************************************************************************/
void Cy_Tcpwm_Counter_SetCC1_IntrMask(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
    ptscTCPWM->unINTR_MASK.stcField.u1CC1_MATCH = 0x1ul;
}
```

Configure the interrupt factor for CC1

Code Listing 16 CYT2 Series: Example to Configure the TCPWM in Driver Part (Cy_Tcpwm_Counter_GetCC0_IntrMasked)

```
/*******************************************************************************
* Function Name: Cy_Tcpwm_Counter_GetCC0_IntrMasked
*******************************************************************************/
uint8_t Cy_Tcpwm_Counter_GetCC0_IntrMasked(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
    return (uint8_t)(ptscTCPWM->unINTR_MASKED.stcField.u1CC0_MATCH);
}
```

Get CC0 interrupt masked

Code Listing 17 CYT2 Series: Example to Configure the TCPWM in Driver Part (Cy_Tcpwm_Counter_ClearCC0_Intr)

```
/*******************************************************************************
* Function Name: Cy_Tcpwm_Counter_ClearCC0_Intr
*******************************************************************************/
void Cy_Tcpwm_Counter_ClearCC0_Intr(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
    ptscTCPWM->unINTR.stcField.u1CC0_MATCH = 1ul;
    ptscTCPWM->unINTR.u32Register;
}
```

Clear CC0 interrupt

Code Listing 18 CYT2 Series: Example to Configure the TCPWM in Driver Part (Cy_Tcpwm_Counter_GetCC1_IntrMasked)

```
/*******************************************************************************
* Function Name: Cy_Tcpwm_Counter_GetCC1_IntrMasked
*******************************************************************************/
uint8_t Cy_Tcpwm_Counter_GetCC1_IntrMasked(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
    return (uint8_t)(ptscTCPWM->unINTR_MASKED.stcField.u1CC1_MATCH);
}
```

Get CC1 interrupt masked

Code Listing 19 CYT2 Series: Example to Configure the TCPWM in Driver Part (Cy_Tcpwm_Counter_ClearCC1_Intr)

```
/*******************************************************************************
* Function Name: Cy_Tcpwm_Counter_ClearCC1_Intr
*******************************************************************************/
void Cy_Tcpwm_Counter_ClearCC1_Intr(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
    ptscTCPWM->unINTR.stcField.u1CC1_MATCH = 1ul;
    ptscTCPWM->unINTR.u32Register;
}
```

Clear CC1 interrupt

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 11 shows PWM Mode in upward counting mode.

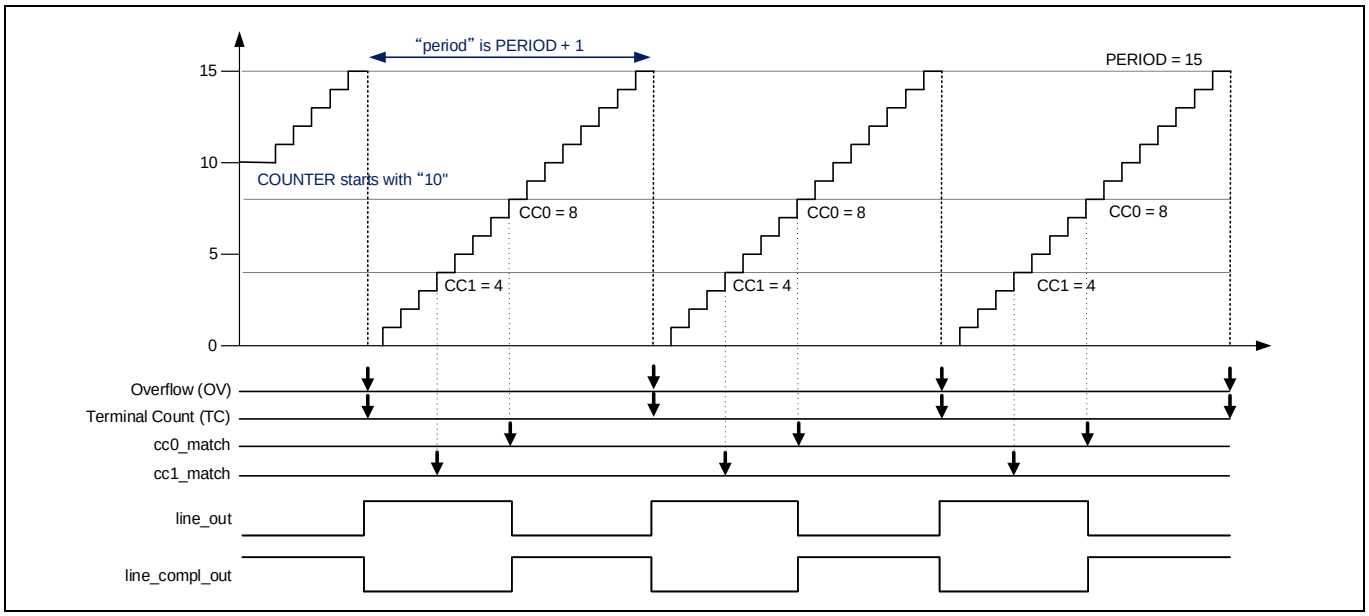

Figure 11 PWM Mode in Up Counting Mode

PWM signal frequency is configured by the PERIOD register. This PWM signal period is the value of 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 12 shows line generation logic. TR_PWM_CTL (e.g., TCPWM0_GRP0_CNT0_TR_PWM_CTRL) register controls the line state change as per four events: Underflow, Overflow, cc0_match, and cc1_match.

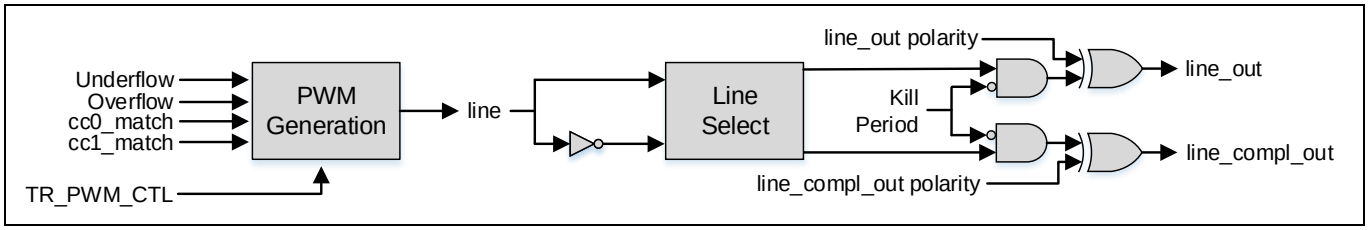

Figure 12 Line Generation Logic

There are two output lines: a PWM signal is output from line_out, and a complementary PMW signal is output from line_compl_out. Relevant I/O port is configured as PWM output resource, line_out, and line_compl_out are output by PWM and PWM_N port. PWM and PWM_N port are assigned to I/O port P0.0 and P0.1 in CYT2B7 series. See the device datasheet for details.

The polarity of both 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; 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.

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.

Counter point is configured by the COUNTER register. In **Figure 10**, beginning from "10" counter point, configured by the COUNTER register, to "15" first Overflow event period is set as waiting time.

Four internal events, Underflow, Overflow, cc0_match, and cc1_mach, can be used to output the trigger. In **Figure 10**, cc1_match event can be configured with CC1 register in the flexible point within a period. This cc1_match event is used to activate trigger for other modules such as SAR ADC.

2.3.1 Use Case

This section describes a use case of PWM mode. PWM signal is generated with Overflow and cc0_match event. The following is an example of configuring TCPWM using SDL.

- TCPWM Operation Mode : PWM Mode
- Using Counter : TCPWM0/Group0/Counter0
- Counter Start operation : Start by Software
- PWM Duty : 50%

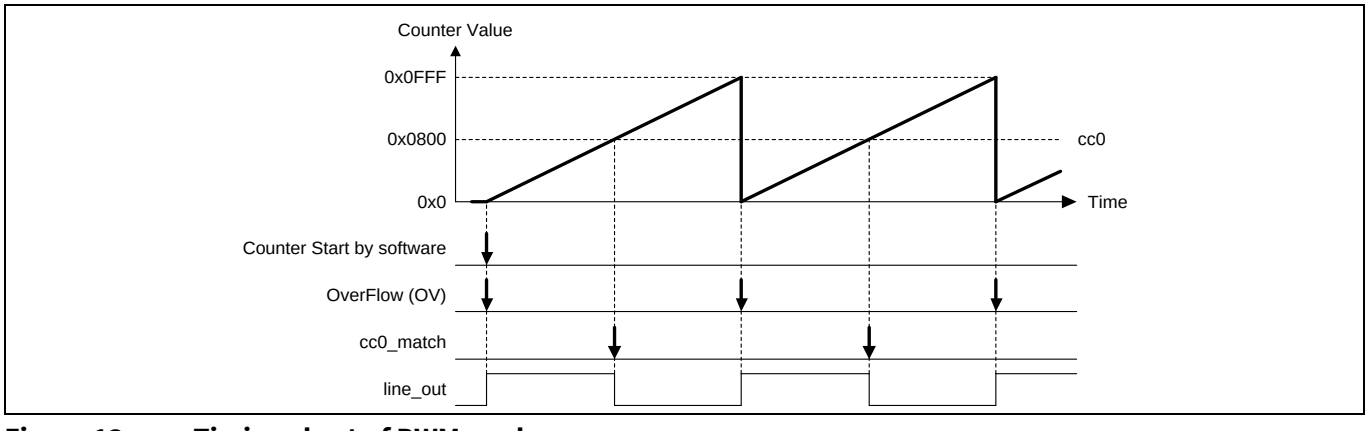

Figure 13 Timing chart of PWM mode

TCPWM Operation Examples

Figure 14 shows the operation flow of this use case.

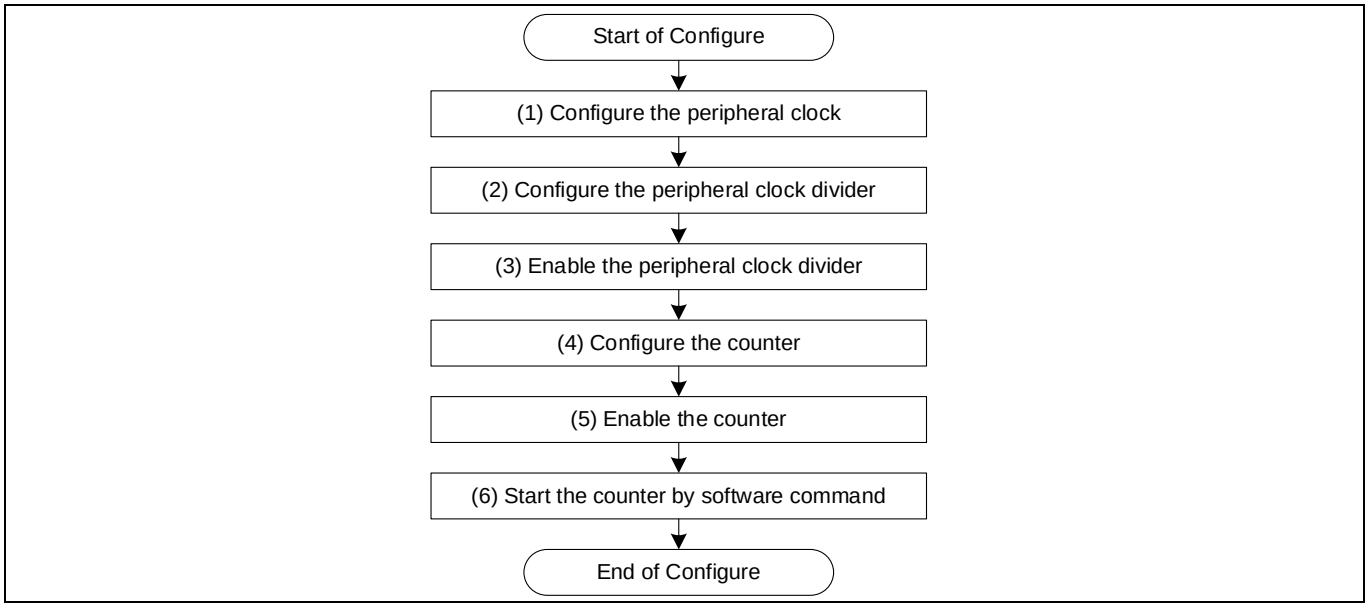

Figure 14 Operation Flow Example

(1) Configure the peripheral clocks for the TCPWM.
(2) Configure the peripheral clock dividers for the TCPWM.
(3) Enable the peripheral clock divider for the TCPWM.
(4) Configure the TCPWM.

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

(5) Enable the TCPWM counter.
(6) Start the TCPWM counter by software command.

2.3.2 Configuration and Example

Table 6 lists the parameters of the configuration part in SDL for PWM Mode.

Table 6 CYT2 Series: List of PWM Mode Configuration Parameters

Parameters	Description	Setting Value
For CLK		
TCPWMx_GRPx_CNTx_PWM	Using Counter Number	TCPWM0_GRP0_CNT0
PCLK_TCPWMx_CLOCKSx_PWM	Peripheral Clock Number	PCLK_TCPWM0_CLOCKS0
TCPWM_PERI_CLK_DIVIDER_NO_PWM	Using Divider Number	0x0
TCPWMx_PWM_PRESCALAR_DIV_x	Pre-scaling of the selected counter clock	CY_TCPWM_PWM_PRESCALER_DIVBY_128
sourceFreq	Frequency of peripheral clock	80000000ul (80MHz)
targetFreq	Frequency of clk_counter	2000000ul (2MHz)

TCPWM Operation Examples

Parameters	Description	Setting Value
For TCPWM		
.pwmMode	Counter mode	CY_TCPWM_PWM_MODE_PWM (0x4)
.clockPrescaler	Pre-scaling of the selected counter clock	TCPWMx_PWM_PRESCALAR_DIV_x
.debug_pause	Counter behavior in debug mode	false (0x0)
.Cc0MatchMode	Determines the effect of a compare match 0 event	CY_TCPWM_PWM_TR_CTRL2_CLEAR (0x1)
.OverflowMode	Determines the effect of a counter overflow event	CY_TCPWM_PWM_TR_CTRL2_SET (0x0)
.UnderflowMode	Determines the effect of a counter underflow event	CY_TCPWM_PWM_TR_CTRL2_NO_CHANGE (0x3)
.Cc1MatchMode	Determines the effect of a compare match 1 event	CY_TCPWM_PWM_TR_CTRL2_NO_CHANGE (0x3)
.deadTime	Determine the dead time	- (This parameter is only valid in PWM_DT mode.)
.deadTimeComp	Determine the dead time	- (This parameter is only valid in PWM_DT mode.)
.runMode	Counter run mode	CY_TCPWM_PWM_CONTINUOUS (0x0)
.period	Value of the counter (This field should be set to “n-1”)	0x0FFF
.period_buff		0x0
.enablePeriodSwap	Exchange the PERIOD and buffered PERIOD values	false (0x0)
.compare0	Compared to counter value for CC0	TCPWMx_COMPARE0 (0x800)
.compare1	Compared to counter value for CC1	0x0000
.enableCompare0Swap	Exchange the CC0 and buffered CC0 values	false (0x0)
.enableCompare1Swap	Exchange the CC1 and buffered CC1 values	false (0x0)
.interruptSources	Interrupt Mask bit	0x0 (zero clear)
.invertPWMOut	behavior of the PWM outputs "line_out" and "line_compl_out"	0x0
.invertPWMOutN		
.killMode	The counter stops on a kill events	CY_TCPWM_PWM_STOP_ON_KILL (0x2)
.switchInputMode	Capture0 edge mode	0x3 (NO_EDGE_DET)
.switchInput	Input triggers for capture0	0x0 (Constant 0)
.reloadInputMode	Reload edge mode	0x3 (NO_EDGE_DET)
.reloadInput	Input triggers for reload	0x0 (Constant 0)
.startInputMode	Start edge mode	0x3 (NO_EDGE_DET)
.startInput	Input triggers for start	0x0 (Constant 0)
.kill0InputMode	Stop edge mode	0x3 (NO_EDGE_DET)
.kill0Input	Input triggers for stop	0x0 (Constant 0)
.kill1InputMode	Capture1 edge mode	0x3 (NO_EDGE_DET)

TCPWM Operation Examples

Parameters	Description	Setting Value
.kill1Input	Input triggers for capture1	0x0 (Constant 0)
.countInputMode	Count edge mode	0x3 (NO_EDGE_DET)
.countInput	Input triggers for count	0x1 (Constant 1)

Code Listing 20 demonstrates an example program to configure PWM mode in the configuration part.

Code Listing 20 CYT2 Series: Example to Configure PWM Mode in Configuration Part

```
#define TCPWMx_GRPx_CNTx_PWM            TCPWM0_GRP0_CNT0
#define PCLK_TCPWMx_CLOCKSx_PWM         PCLK_TCPWM0_CLOCKS0
#define TCPWM_PERI_CLK_DIVIDER_NO_PWM   0ul

#define TCPWMx_PWM_PRESCALAR_DIV_x      CY_TCPWM_PWM_PRESCALER_DIVBY_128  // 2,000,000 / 128 = 15,625Hz

#define TCPWMx_PERIOD    0x1000ul  // 15,625Hz / 4096 (0x1000) = 3.815Hz (PWM frequency)
#define TCPWMx_COMPARE0  0x800ul   // 0x800 / 0x1000 = 0.5 (PWM duty)

    :
cy_stc_tcpwm_pwm_config_t const MyPWM_config =
{
    .pwmMode            = CY_TCPWM_PWM_MODE_PWM,
    .clockPrescaler     = TCPWMx_PWM_PRESCALAR_DIV_x,
    .debug_pause        = false,
    .Cc0MatchMode       = CY_TCPWM_PWM_TR_CTRL2_CLEAR,
    .OverflowMode       = CY_TCPWM_PWM_TR_CTRL2_SET,
    .UnderflowMode      = CY_TCPWM_PWM_TR_CTRL2_NO_CHANGE,
    .Cc1MatchMode       = CY_TCPWM_PWM_TR_CTRL2_NO_CHANGE,
    .deadTime           = 0ul,
    .deadTimeComp       = 0ul,
    .runMode            = CY_TCPWM_PWM_CONTINUOUS,
    .period             = TCPWMx_PERIOD - 1ul,
    .period_buff        = 0ul,
    .enablePeriodSwap   = false,
    .compare0           = TCPWMx_COMPARE0,
    .compare1           = 0ul,
    .enableCompare0Swap = false,
    .enableCompare1Swap = false,
    .interruptSources   = 0ul,
    .invertPWMOut       = 0ul,
    .invertPWMOutN      = 0ul,
    .killMode           = CY_TCPWM_PWM_STOP_ON_KILL,
    .switchInputMode    = 3ul,
    .switchInput        = 0ul,
    .reloadInputMode    = 3ul,
    .reloadInput        = 0ul,
    .startInputMode     = 3ul,
    .startInput         = 0ul,
    .kill0InputMode     = 3ul,
    .kill0Input         = 0ul,
    .kill1InputMode     = 3ul,
    .kill1Input         = 0ul,
    .countInputMode     = 3ul,
    .countInput         = 1ul,
};

int main(void)
{
    :
    uint32_t sourceFreq = 80000000ul;
    uint32_t targetFreq = 2000000ul;
    uint32_t divNum = (sourceFreq / targetFreq);
    :

    /* Assign a programmable divider  for TCPWM0_GRPx_CNTx_COUNTER */
    Cy_SysClk_PeriphAssignDivider(PCLK_TCPWMx_CLOCKSx_PWM, CY_SYSCLK_DIV_16_BIT, TCPWM_PERI_CLK_DIVIDER_NO_PWM);

    /* Sets the 16-bit divider */
    Cy_SysClk_PeriphSetDivider(CY_SYSCLK_DIV_16_BIT, TCPWM_PERI_CLK_DIVIDER_NO_PWM, (divNum-1ul));

    /* Enable the divider */
    Cy_SysClk_PeriphEnableDivider(CY_SYSCLK_DIV_16_BIT, TCPWM_PERI_CLK_DIVIDER_NO_PWM);

    /* Initialize TCPWM0_GRPx_CNTx_PWM_PR as PWM Mode & Enable */
    Cy_Tcpwm_Pwm_Init(TCPWMx_GRPx_CNTx_PWM, &MyPWM_config);
```

Define Using Counter
Define Peripheral Clock
Define Peripheral Clock Divider
Define Prescaler of counter Clock
Define PWM Period
Define Compare Period
Configure the PWM parameters
Calculation of division ratio
(1)Configure the Peripheral Clock (See **Code Listing 3**)
(2)Configure the integer division of the 16-bit divider (See **Code Listing 4**)
(3)Enable the 16-bit divider (See **Code Listing 5**)
(4)Initialize the counter based on above structure (See **Code Listing 21**)

TCPWM Operation Examples

Code Listing 20 CYT2 Series: Example to Configure PWM Mode in Configuration Part

```
    Cy_Tcpwm_Pwm_Enable(TCPWMx_GRPx_CNTx_PWM);
    Cy_Tcpwm_TriggerStart(TCPWMx_GRPx_CNTx_PWM);
    :
    for(;;);
}
```

(5)Enable the counter (See **Code Listing 22**)

(6)Start the counter (See **Code Listing 9**)

Code Listing 21 and **Code Listing 22** demonstrate an example program to configure TCPWM in the driver part.

Code Listing 21 CYT2 Series: Example to Configure TCPWM in Driver Part (Cy_Tcpwm_Pwm_Init)

```
/*******************************************************************************
* Function Name: Cy_Tcpwm_Pwm_Init
*******************************************************************************/
uint32_t Cy_Tcpwm_Pwm_Init(volatile stc_TCPWM_GRP_CNT_t* ptscTCPWM, cy_stc_tcpwm_pwm_config_t const* config)
{
    uint32_t status = CY_RET_BAD_PARAM;
    if((NULL != ptscTCPWM) && (NULL != config))
    {
        un_TCPWM_GRP_CNT_CTRL_t workCTRL = {.u32Register = 0ul};
        workCTRL.stcField.u1ONE_SHOT              = config->runMode;
        workCTRL.stcField.u2UP_DOWN_MODE          = config->countDirection;
        workCTRL.stcField.u3MODE                  = config->pwmMode;
        workCTRL.stcField.u1DBG_FREEZE_EN         = config->debug_pause;
        workCTRL.stcField.u1AUTO_RELOAD_CC0       = config->enableCompare0Swap;
        workCTRL.stcField.u1AUTO_RELOAD_CC1       = config->enableCompare1Swap;
        workCTRL.stcField.u1AUTO_RELOAD_PERIOD    = config->enablePeriodSwap;
        workCTRL.stcField.u1AUTO_RELOAD_LINE_SEL  = config->enableLineSelSwap;
        workCTRL.stcField.u1PWM_SYNC_KILL         = config->killMode;
        workCTRL.stcField.u1PWM_STOP_ON_KILL      = (config->killMode>>1ul);
        ptscTCPWM->unCTRL.u32Register             = workCTRL.u32Register;

        if(CY_TCPWM_COUNTER_COUNT_UP == config->runMode)
        {
            ptscTCPWM->unCOUNTER.u32Register = CY_TCPWM_CNT_UP_INIT_VAL;
        }
        else if(CY_TCPWM_COUNTER_COUNT_DOWN == config->runMode)
        {
            ptscTCPWM->unCOUNTER.u32Register = config->period;
        }
        else
        {
            ptscTCPWM->unCOUNTER.u32Register = CY_TCPWM_CNT_UP_DOWN_INIT_VAL;
        }
        ptscTCPWM->unCC0.u32Register = config->compare0;
        ptscTCPWM->unCC0_BUFF.u32Register = config->compare0_buff;
        ptscTCPWM->unCC1.u32Register = config->compare1;
        ptscTCPWM->unCC1_BUFF.u32Register = config->compare1_buff;
        ptscTCPWM->unPERIOD.u32Register = config->period;
        ptscTCPWM->unPERIOD_BUFF.u32Register = config->period_buff;
        un_TCPWM_GRP_CNT_TR_IN_SEL0_t workTR_IN_SEL0 = {.u32Register = 0ul};
        workTR_IN_SEL0.stcField.u8CAPTURE0_SEL = config->switchInput;
        workTR_IN_SEL0.stcField.u8RELOAD_SEL   = config->reloadInput;
        workTR_IN_SEL0.stcField.u8STOP_SEL     = config->kill0Input;
        workTR_IN_SEL0.stcField.u8COUNT_SEL    = config->countInput;
        ptscTCPWM->unTR_IN_SEL0.u32Register    = workTR_IN_SEL0.u32Register;
        un_TCPWM_GRP_CNT_TR_IN_SEL1_t workTR_IN_SEL1 = {.u32Register = 0ul};
        workTR_IN_SEL1.stcField.u8CAPTURE1_SEL = config->kill1Input;
        workTR_IN_SEL1.stcField.u8START_SEL    = config->startInput;
        ptscTCPWM->unTR_IN_SEL1.u32Register    = workTR_IN_SEL1.u32Register;
        un_TCPWM_GRP_CNT_TR_IN_EDGE_SEL_t workTR_IN_EDGE_SEL = {.u32Register = 0ul};
        workTR_IN_EDGE_SEL.stcField.u2CAPTURE0_EDGE = config->switchInputMode;
        workTR_IN_EDGE_SEL.stcField.u2CAPTURE1_EDGE = config->kill1InputMode;
        workTR_IN_EDGE_SEL.stcField.u2RELOAD_EDGE   = config->reloadInputMode;
        workTR_IN_EDGE_SEL.stcField.u2START_EDGE    = config->startInputMode;
        workTR_IN_EDGE_SEL.stcField.u2STOP_EDGE     = config->kill0InputMode;
        workTR_IN_EDGE_SEL.stcField.u2COUNT_EDGE    = config->countInputMode;
        ptscTCPWM->unTR_IN_EDGE_SEL.u32Register = workTR_IN_EDGE_SEL.u32Register;
        ptscTCPWM->unINTR_MASK.u32Register = config->interruptSources;
        un_TCPWM_GRP_CNT_TR_PWM_CTRL_t workTR_PWM_CTRL = {.u32Register = 0ul};
        workTR_PWM_CTRL.stcField.u2CC0_MATCH_MODE = config->Cc0MatchMode;
        workTR_PWM_CTRL.stcField.u2CC1_MATCH_MODE = config->Cc1MatchMode;
        workTR_PWM_CTRL.stcField.u2OVERFLOW_MODE  = config->OverflowMode;
        workTR_PWM_CTRL.stcField.u2UNDERFLOW_MODE = config->UnderflowMode;
        ptscTCPWM->unTR_PWM_CTRL.u32Register      = workTR_PWM_CTRL.u32Register;
        un_TCPWM_GRP_CNT_DT_t workDT = {.u32Register = 0ul};
        workDT.stcField.u16DT_LINE_COMPL_OUT = config->deadTimeComp;
        workDT.stcField.u8DT_LINE_OUT_H      = (config->deadTime>>8ul);
        if(config->pwmMode == CY_TCPWM_PWM_MODE_DEADTIME)
```

Initialize the PWM counter

Check if configuration parameter values are valid

The initial value of the counter is determined according to each countDirection

Code Listing 21 CYT2 Series: Example to Configure TCPWM in Driver Part (Cy_Tcpwm_Pwm_Init)

```
        {
            workDT.stcField.u8DT_LINE_OUT_L = config->deadTime;
        }
        else
        {
            workDT.stcField.u8DT_LINE_OUT_L = config->clockPrescaler;
        }
        ptscTCPWM->unDT.u32Register          = workDT.u32Register;
        status = CY_RET_SUCCESS;
}
```

Code Listing 22 CYT2 Series: Example to Configure the TCPWM in Driver Part (Cy_Tcpwm_Pwm_Enable)

```
/*******************************************************************************
* Function Name: Cy_Tcpwm_Pwm_Enable
*******************************************************************************/
void Cy_Tcpwm_Pwm_Enable(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
    ptscTCPWM->unCTRL.stcField.u1ENABLED = 0x1ul;
}
```

Enable the counter

2.4 PWM Dead Time (PWM_DT) Mode

This section describes how to set up 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 15 shows PWM_DT Mode in up counting mode.

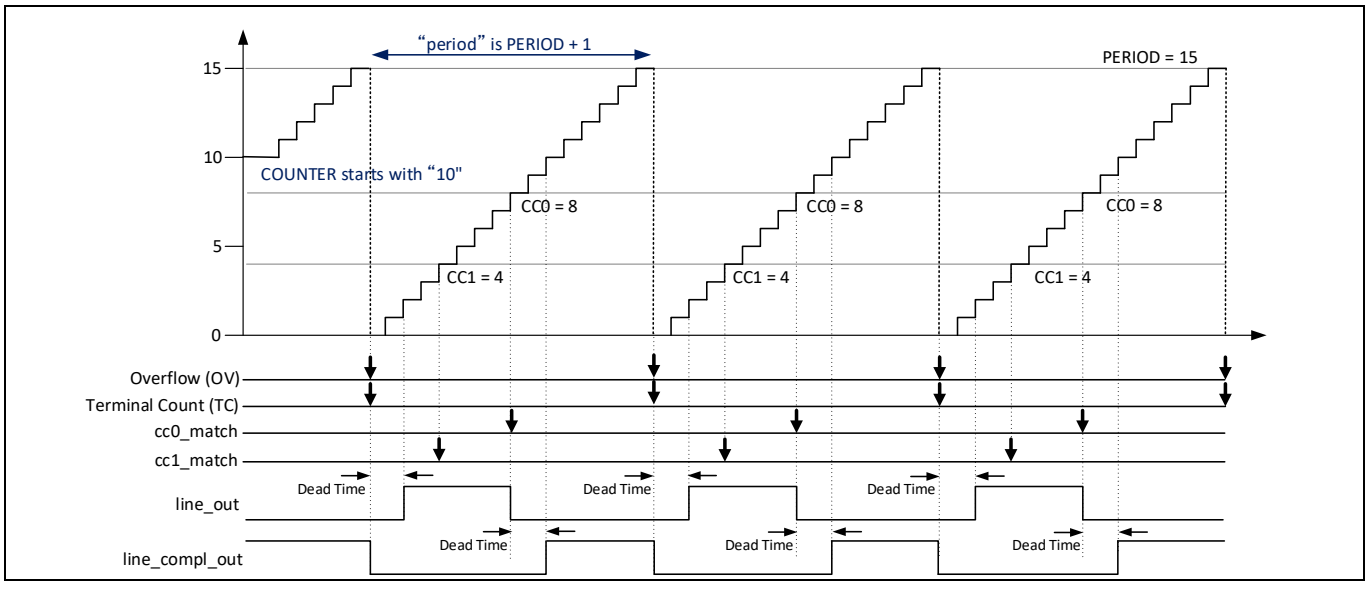

Figure 15 Counting Behavior for PWM_DT Mode

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.

TCPWM Operation Examples

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. 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 different values.

2.4.1 Use Case

This section describes a use case of PWM_DT mode. PWM_DT signal is generated with Overflow and cc0_match event. The following is an example of configuring TCPWM using SDL.

- TCPWM Operation Mode : PWM_DT Mode
- Using Counter : TCPWM0/Group1/Counter0
- Counter Start operation : Start by Software
- PWM Duty : 50%
- Amount of dead time cycles in the counter clock : Line_out = 0d500
  : Line_compl_out = 0d1000

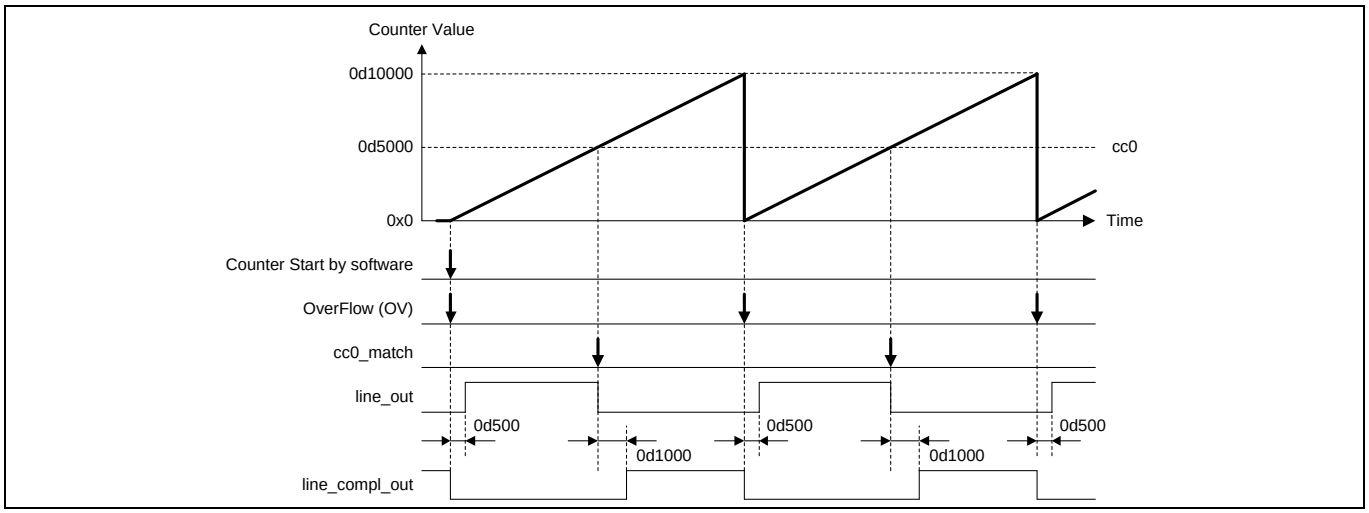

Figure 16 Timing Chart of PWM_DT mode

Figure 17 shows the operation flow of this use case.

TCPWM Operation Examples

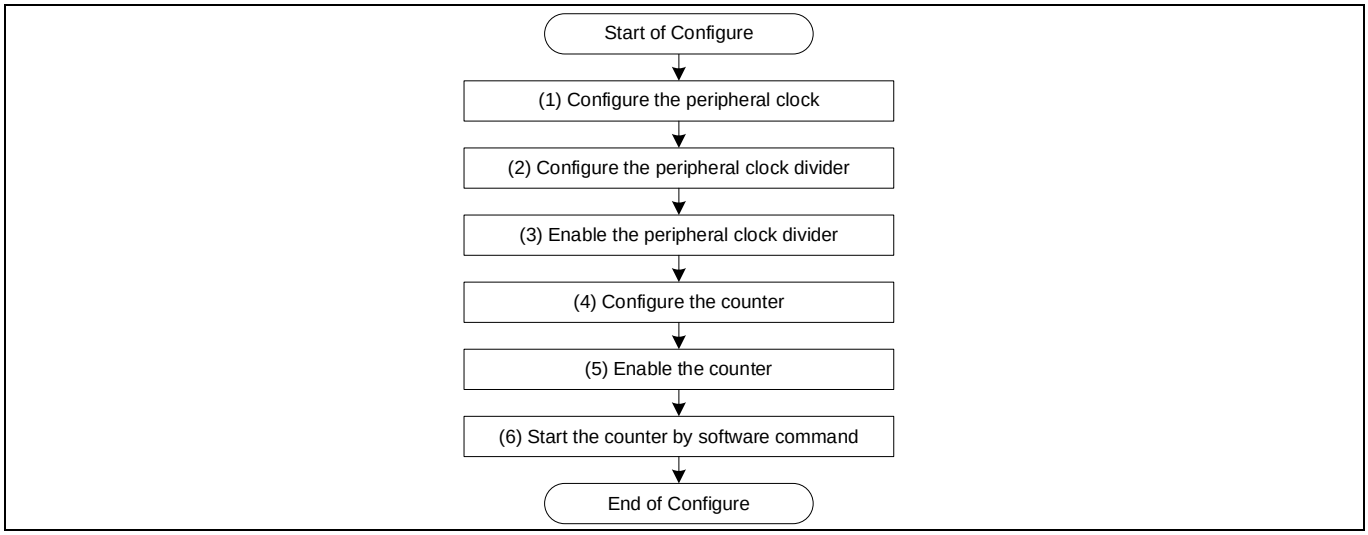

Figure 17 Operation Flow Example

(1) Configure the peripheral clocks for the TCPWM.
(2) Configure the peripheral clock dividers for the TCPWM.
(3) Enable the peripheral clock divider for the TCPWM.
(4) Configure the TCPWM.

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

(5) Enable the TCPWM counter.
(6) Start the TCPWM counter by software command.

2.4.2 Configuration and Example

Table 7 lists the parameters of the configuration part in SDL for PWM Mode.

Table 7 CYT2 Series: List of PWM_DT Mode Configuration Parameters

Parameters	Description	Setting Value
For CLK		
TCPWMx_GRPx_CNTx_PWM_DT	Using Counter Number	TCPWM0_GRP1_CNT0
PCLK_TCPWMx_CLOCKSx_PWM_DT	Peripheral Clock Number	PCLK_TCPWM0_CLOCKS256
TCPWM_PERI_CLK_DIVIDER_NO_PWM_DT	Using Divider Number	0x0
TCPWMx_PERIOD	PWM Period	0d10000
TCPWMx_COMPARE0	Compare Period	0d5000
TCPWMx_DEADTIME	Dead Time for line_out	0d500
TCPWMx_DEADTIME_COMPL	Dead Time for line_compl_out	0d1000
sourceFreq	Frequency of peripheral clock	80000000ul (80MHz)
targetFreq	Frequency of clk_counter	2000000ul (2MHz)
For TCPWM		
.pwmMode	Counter mode	CY_TCPWM_PWM_MODE_DEADTIME (0x5)
.clockPrescaler	Pre-scaling of the selected counter clock	CY_TCPWM_PWM_PRESCALER_DIVBY_1 (0x0)
.debug_pause	Counter behavior in debug mode	false (0x0)
.Cc0MatchMode	Determines the effect of a compare match 0 event	CY_TCPWM_PWM_TR_CTRL2_CLEAR (0x1)
.OverflowMode	Determines the effect of a counter overflow event	CY_TCPWM_PWM_TR_CTRL2_SET (0x0)
.UnderflowMode	Determines the effect of a counter underflow event	CY_TCPWM_PWM_TR_CTRL2_NO_CHANGE (0x3)
.Cc1MatchMode	Determines the effect of a compare match 1 event	CY_TCPWM_PWM_TR_CTRL2_NO_CHANGE (0x3)
.deadTime	Determine the dead time	TCPWMx_DEADTIME (0d500)
.deadTimeComp	Determine the dead time	TCPWMx_DEADTIME_COMPL (0d1000)
.runMode	Counter run mode	CY_TCPWM_PWM_CONTINUOUS (0x0)
.period	Value of the counter (This field should be set to “n-1”)	TCPWMx_PERIOD - 1ul (0d9999)
.period_buff		0d0
.enablePeriodSwap	Exchange the PERIOD and buffered PERIOD values	false (0x0)
.compare0	Compared to counter value for CC0	TCPWMx_COMPARE0 (0d5000)
.compare1	Compared to counter value for CC1	0d0

TCPWM Operation Examples

Parameters	Description	Setting Value
.enableCompare0Swap	Exchange the CC0 and buffered CC0 values	false (0x0)
.enableCompare1Swap	Exchange the CC1 and buffered CC1 values	false (0x0)
.interruptSources	Interrupt Mask bit	0d0 (zero clear)
.invertPWMOut	behavior of the PWM outputs "line_out" and "line_compl_out"	0d0
.invertPWMOutN		0d0
.killMode	The counter stops on a kill events	CY_TCPWM_PWM_STOP_ON_KILL (0x2)
.switchInputMode	Capture0 edge mode	0x3 (NO_EDGE_DET)
.switchInput	Input triggers for capture0	0x0 (Constant 0)
.reloadInputMode	Reload edge mode	0x3 (NO_EDGE_DET)
.reloadInput	Input triggers for reload	0x0 (Constant 0)
.startInputMode	Start edge mode	0x3 (NO_EDGE_DET)
.startInput	Input triggers for start	0x0 (Constant 0)
.kill0InputMode	Stop edge mode	0x3 (NO_EDGE_DET)
.kill0Input	Input triggers for stop	0x0 (Constant 0)
.kill1InputMode	Capture1 edge mode	0x3 (NO_EDGE_DET)
.kill1Input	Input triggers for capture1	0x0 (Constant 0)
.countInputMode	Count edge mode	0x3 (NO_EDGE_DET)
.countInput	Input triggers for count	0x1 (Constant 1)

Code Listing 23 demonstrates an example program to configure PWM_DT mode in the configuration part.

Code Listing 23 CYT2 Series: Example Program to Configure PWM_DT Mode in the Configuration Part

```
#define TCPWMx_GRPx_CNTx_PWM_DT           TCPWM0_GRP1_CNT0
#define PCLK_TCPWMx_CLOCKSx_PWM_DT        PCLK_TCPWM0_CLOCKS256
#define TCPWM_PERI_CLK_DIVIDER_NO_PWM_DT  0ul


#define TCPWMx_PERIOD          10000ul    // 2,000,000 / 10000 = 200Hz
#define TCPWMx_COMPARE0        5000ul     // 5000 / 10000 = 0.5 (duty)
#define TCPWMx_DEADTIME        500ul      // Right side: 500*(1/2,000,000) = 250 us
#define TCPWMx_DEADTIME_COMPL  1000ul     // Left side :1000*(1/2,000,000) = 500 us


cy_stc_tcpwm_pwm_config_t const MyPWM_config =
{
    .pwmMode            = CY_TCPWM_PWM_MODE_DEADTIME,
    .clockPrescaler     = CY_TCPWM_PWM_PRESCALER_DIVBY_1,
    .debug_pause        = false,
    .Cc0MatchMode       = CY_TCPWM_PWM_TR_CTRL2_CLEAR,
    .OverflowMode       = CY_TCPWM_PWM_TR_CTRL2_SET,
    .UnderflowMode      = CY_TCPWM_PWM_TR_CTRL2_NO_CHANGE,
    .Cc1MatchMode       = CY_TCPWM_PWM_TR_CTRL2_NO_CHANGE,
    .deadTime           = TCPWMx_DEADTIME,
    .deadTimeComp       = TCPWMx_DEADTIME_COMPL,
    .runMode            = CY_TCPWM_PWM_CONTINUOUS,
    .period             = TCPWMx_PERIOD - 1ul,
    .period_buff        = 0ul,
    .enablePeriodSwap   = false,
    .compare0           = TCPWMx_COMPARE0,
    .compare1           = 0ul,
    .enableCompare0Swap = false,
    .enableCompare1Swap = false,
    .interruptSources   = 0ul,
    .invertPWMOut       = 0ul,
```

Define Using Counter
Define Peripheral Clock
Define Peripheral Clock Divider
Define PWM Period
Define Compare Period
Define Dead Time for line_out
Define Dead Time for line_compl_out
Configure the PWM_DT parameters

TCPWM Operation Examples

Code Listing 23 CYT2 Series: Example Program to Configure PWM_DT Mode in the Configuration Part

```
    .invertPWMOutN      = 0ul,
    .killMode           = CY_TCPWM_PWM_STOP_ON_KILL,
    .switchInputMode    = 3ul,
    .switchInput        = 0ul,
    .reloadInputMode    = 3ul,
    .reloadInput        = 0ul,
    .startInputMode     = 3ul,
    .startInput         = 0ul,
    .kill0InputMode     = 3ul,
    .kill0Input         = 0ul,
    .kill1InputMode     = 3ul,
    .kill1Input         = 0ul,
    .countInputMode     = 3ul,
    .countInput         = 1ul,
};

int main(void)
{
    :
    uint32_t sourceFreq = 80000000ul;
    uint32_t targetFreq = 2000000ul;
    uint32_t divNum = (sourceFreq / targetFreq);
    :
    /* Assign a programmable divider  for TCPWM0_GRPx_CNTx_COUNTER */
    Cy_SysClk_PeriphAssignDivider(PCLK_TCPWMx_CLOCKSx_PWM_DT, CY_SYSCLK_DIV_16_BIT, TCPWM_PERI_CLK_DIVIDER_NO_PWM_DT);

    /* Sets the 16-bit divider */
    Cy_SysClk_PeriphSetDivider(CY_SYSCLK_DIV_16_BIT, TCPWM_PERI_CLK_DIVIDER_NO_PWM_DT, (divNum-1ul));

    /* Enable the divider */
    Cy_SysClk_PeriphEnableDivider(CY_SYSCLK_DIV_16_BIT, TCPWM_PERI_CLK_DIVIDER_NO_PWM_DT)

    /* Initialize TCPWMx_GRPx_CNTx_PWM_PR as PWM-DT Mode & Enable */
    Cy_Tcpwm_Pwm_Init(TCPWMx_GRPx_CNTx_PWM, &MyPWM_config);
    Cy_Tcpwm_Pwm_Enable(TCPWMx_GRPx_CNTx_PWM);
    Cy_Tcpwm_TriggerStart(TCPWMx_GRPx_CNTx_PWM);
    :
    for(;;);
}
```

Calculation of division ratio

(1)Configure the Peripheral Clock (See **Code Listing 3**)

(2)Configure the integer division of the 16-bit divider (See **Code Listing 4**)

(3)Enable the 16-bit divider (See **Code Listing 5**)

(4)Initialize the counter based on above structure (See **Code Listing 21**)

(5)Enable the counter (See **Code Listing 22**)

(6)Start the counter (See **Code Listing 9**)

There are no new drivers here. See the link above.

3 Relation of Trigger Multiplexer

Traveo II 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 18 shows the operation flow when using peripherals such as a Trigger Multiplexer, ADC and TCPWM. This flow is the same for Section **3.1** and Section **3.2**.

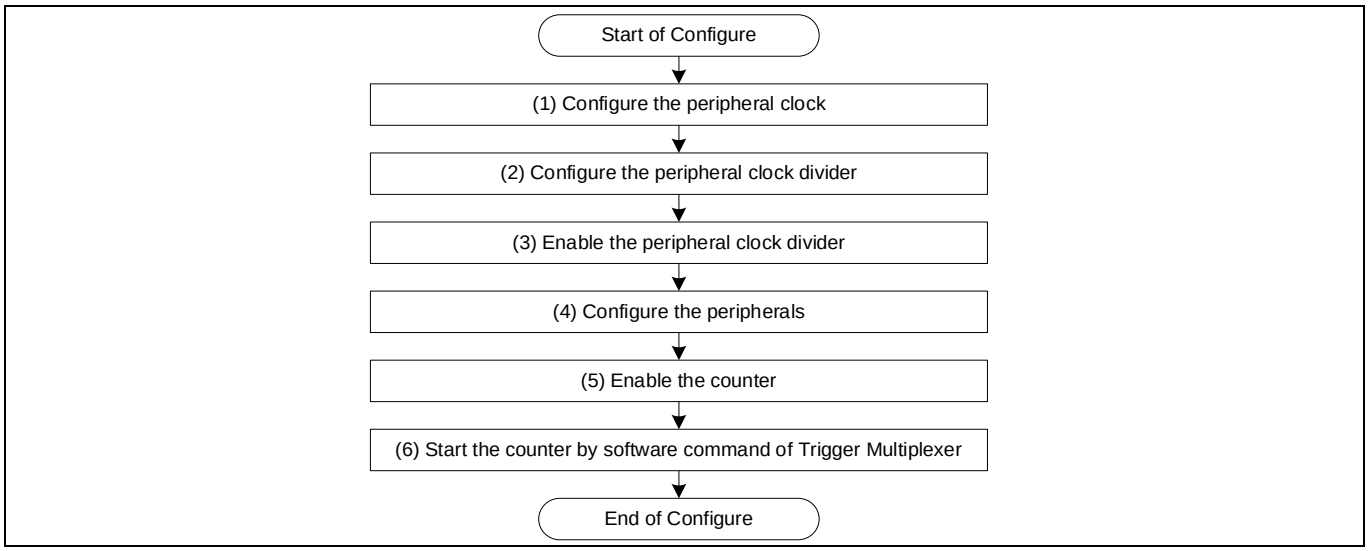

Figure 18 Operation Flow Example

(1) Configure the peripheral clocks for the TCPWM.
(2) Configure the peripheral clock dividers for the TCPWM.
(3) Enable the peripheral clock divider for the TCPWM.
(4) Configure the peripherals (Trigger Multiplexer, SAR ADC and TCPWM).

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

(5) Enable the TCPWM counter.
(6) Start the TCPWM counter by software command of Trigger Multiplexer.

3.1 Three TCPWM Simultaneous Start

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 SDL.

- TCPWM Operation Mode : PWM_DT Mode
- Using Counter : TCPWM0/Group1/Counter0, 1, 2
- Counter Start operation : Start by Software

Relation of Trigger Multiplexer

Figure 19 shows an example to use Triger 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.

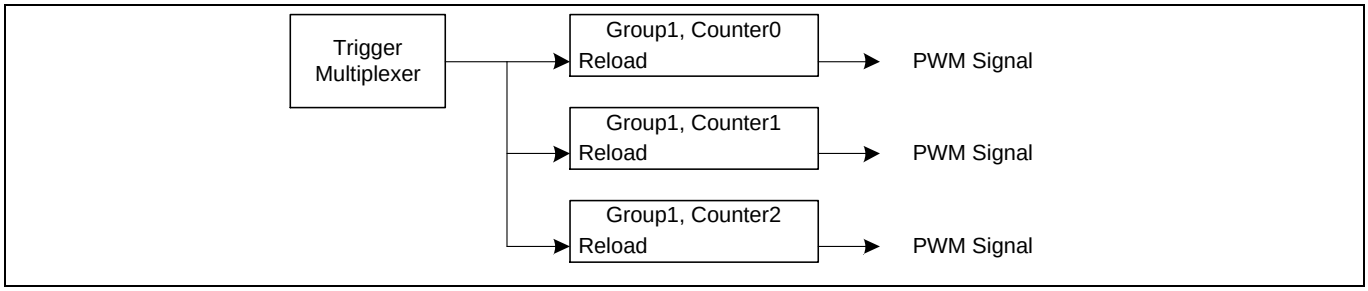

Figure 19 CYT2B7 series: Three TCPWM Simultaneous Start by Reload Signal

To explain how to set up this example, a detailed block diagram of the Trigger Multiplexer and TCPWM is shown **Figure 20**. As shown, the 27 outputs of the trigger multiplexer are connected to the 32-to-1 selector of each counter. Each counter can select the signal by TCPWM0_GRP1_CNT[a]_TR_IN_SEL0/1 register (a = counter number (0, 1, 2)).

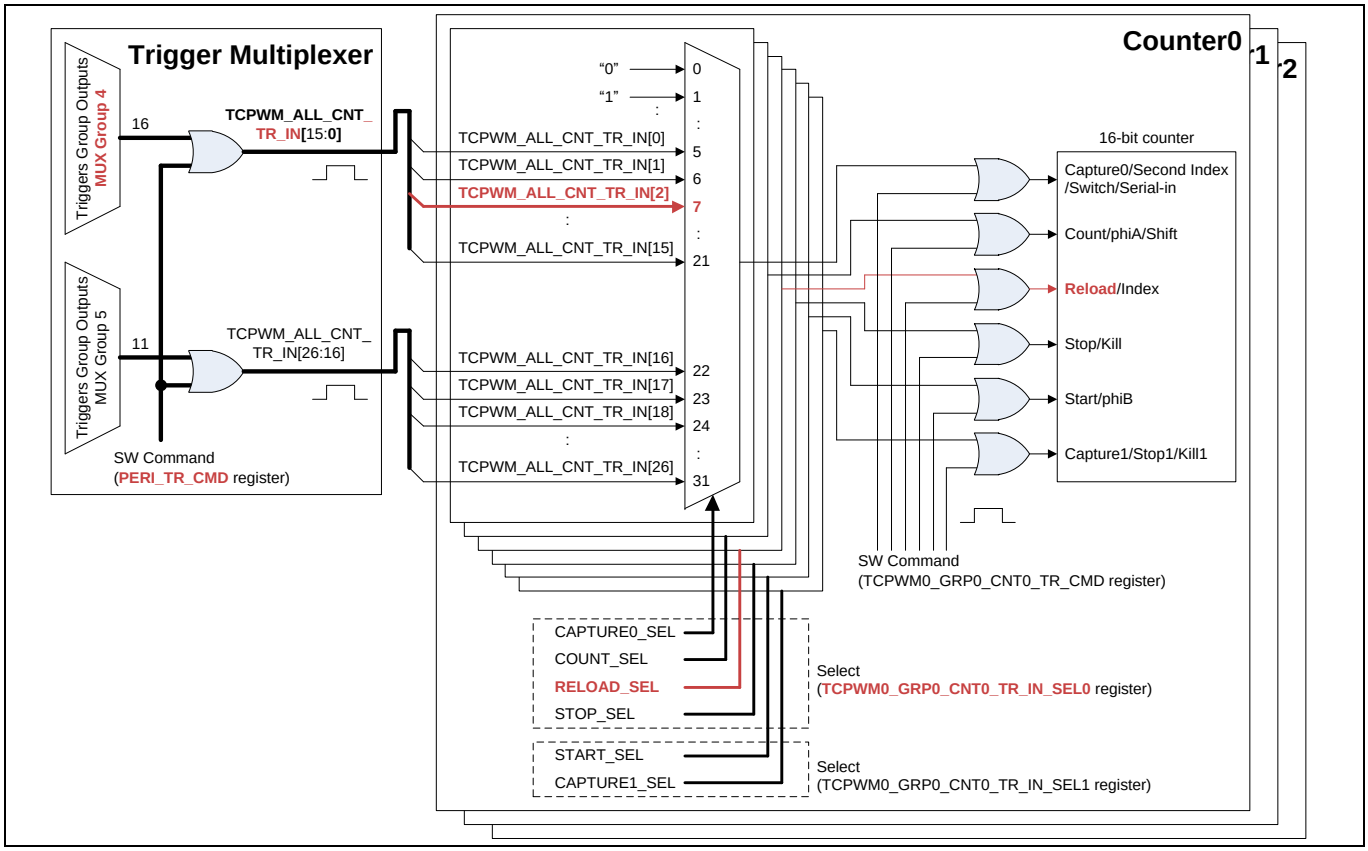

Figure 20 CYT2B7 Series: Detailed Block Diagram of the Trigger Multiplexer and TCPWM for Connection

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

For example, if the RELOAD_SEL bits in the TCPWM0_GRP1_CNTx_TR_IN_SEL0 (x = 0, 1, 2) register is set to "7", TCPWM_ALL_CNT_TR_IN[2] of the Trigger Multiplexer will be selected as the counter Reload. TCPWM_ALL_CNT_TR_IN[2] belongs to MUX Group4 of "Group Trigger". If the PERI_TRM_CMD register is used, a

Relation of Trigger Multiplexer

HIGH/LOW/pulse signal can be output to TCPWM_ALL_CNT_TR_IN[2] (here, this function is called Software Command). If this output is supplied to Reload of 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

Table 8 lists the parameters of the configuration part in SDL for Three TCPWM Simultaneous Start by Reload Signal.

Table 8 CYT2 Series: List of Three TCPWM Simultaneous Start by Reload Signal Configuration Parameters

Parameters	Description	Setting Value
For TCPWM		
.pwmMode	Counter mode	CY_TCPWM_PWM_MODE_DEADTIME (0x5)
.clockPrescaler	Pre-scaling of the selected counter clock	CY_TCPWM_PWM_PRESCALER_DIVBY_1 (0x0)
.debug_pause	Counter behavior in debug mode	false (0x0)
.countDirection	Counter Direction	COUNT_UPDN2 (0x3)
.Cc0MatchMode	Determines the effect of a compare match 0 event	CY_TCPWM_PWM_TR_CTRL2_SET (0x0)
.OverflowMode	Determines the effect of a counter overflow event	CY_TCPWM_PWM_TR_CTRL2_SET (0x0)
.UnderflowMode	Determines the effect of a counter underflow event	CY_TCPWM_PWM_TR_CTRL2_CLEAR (0x1)
.Cc1MatchMode	Determines the effect of a compare match 1 event	CY_TCPWM_PWM_TR_CTRL2_CLEAR (0x1)
.deadTime	Determine the dead time	100
.deadTimeComp	Determine the dead time	100
.runMode	Counter run mode	CY_TCPWM_PWM_CONTINUOUS (0x0)
.period	Value of the counter (This field should be set to “n-1”)	2000
.period_buff		2000
.enablePeriodSwap	Exchange the PERIOD and buffered PERIOD values	false (0x0)
.enableCompare0Swap	Exchange the CC0 and buffered CC0 values	true (0x1)
.enableCompare1Swap	Exchange the CC1 and buffered CC1 values	true (0x1)
.interruptSources	Interrupt Mask bit	0d0 (zero clear)
.invertPWMOut	behavior of the PWM outputs "line_out" and "line_compl_out"	0d0
.invertPWMOutN		0d0
.killMode	The counter stops on kill events	CY_TCPWM_PWM_STOP_ON_KILL (0x2)
.switchInputMode	Capture0 edge mode	0x3 (NO_EDGE_DET)
.switchInput	Input triggers for capture0	0x0 (Constant 0)
.reloadInputMode	Reload edge mode	0x3 (NO_EDGE_DET)

Relation of Trigger Multiplexer

Parameters	Description	Setting Value
.reloadInput	Input triggers for reload	0x7 (TCPWM_ALL_CNT_TR_IN[2])
.startInputMode	Start edge mode	0x3 (NO_EDGE_DET)
.startInput	Input triggers for start	0x0 (Constant 0)
.kill0InputMode	Stop edge mode	0x3 (NO_EDGE_DET)
.kill0Input	Input triggers for stop	0x0 (Constant 0)
.kill1InputMode	Capture1 edge mode	0x3 (NO_EDGE_DET)
.kill1Input	Input triggers for capture1	0x0 (Constant 0)
.countInputMode	Count edge mode	0x3 (NO_EDGE_DET)
.countInput	Input triggers for count	0x1 (Constant 1)
For Trigger Multiplexer		
trigLine	Triggers Group Outputs	TRIG_OUT_MUX_4_TCPWM_ALL_CNT_TR_IN2
trigType	Level Sensitive or Edge Sensitive for output trigger	TRIGGER_TYPE_EDGE
outSel	Specifies input trigger	0x1

Code Listing 24 demonstrates an example program to start Three TCPWM Simultaneous in the configuration part.

Code Listing 24 CYT2 Series: Example to start Three TCPWM Simultaneous in Configuration Part

```
/* Configuration for U/V/W-phase Timer */
cy_stc_tcpwm_pwm_config_t MyPWM_config = ──────────────── Configure the PWM_DT parameters
{
    .pwmMode            = CY_TCPWM_PWM_MODE_DEADTIME,
    .clockPrescaler     = CY_TCPWM_PWM_PRESCALER_DIVBY_1,
    .debug_pause        = false,
    .countDirection     = 3ul,    /* Set UPDN2 modes */
    .Cc0MatchMode       = CY_TCPWM_PWM_TR_CTRL2_SET,
    .OverflowMode       = CY_TCPWM_PWM_TR_CTRL2_SET,
    .UnderflowMode      = CY_TCPWM_PWM_TR_CTRL2_CLEAR,
    .Cc1MatchMode       = CY_TCPWM_PWM_TR_CTRL2_CLEAR,
    .deadTime           = 100ul,  /* Right side dead time: 100*(1/40,000,000) = 2.5us */
    .deadTimeComp       = 100ul,  /* Left  side dead time: 100*(1/40,000,000) = 2.5us */
    .runMode            = CY_TCPWM_PWM_CONTINUOUS,
    .period             = 2000ul, /* 40,000,000 / (2,000*2) = 10,000Hz (10kHz) */
    .period_buff        = 2000ul,
    .enablePeriodSwap   = false,  /* Auto Reload Period = OFF */
    .enableCompare0Swap = true,   /* Auto Reload CC0 = ON */
    .enableCompare1Swap = true,   /* Auto Reload CC1 = ON */
    .interruptSources   = 0ul,    /* Interrupt Mask for TC, CC0/CC1_MATCH (0:OFF, 1:TC, 2:CC0 MATCH, 4:CC1 MATCH,
7:all) */
    .invertPWMOut       = 0ul,
    .invertPWMOutN      = 0ul,
    .killMode           = CY_TCPWM_PWM_STOP_ON_KILL,
    .switchInputMode    = 3ul,  /* NO_EDGE_DET: No edge detection, use trigger as is */
    .switchInput        = 0ul,  /* Select the constant 0 */
    .reloadInputMode    = 3ul,  /* NO_EDGE_DET: No edge detection, use trigger as is */
    .reloadInput        = 7ul,  /* Select the TCPWM_ALL_CNT_TR_IN[2] */
    .startInputMode     = 3ul,  /* NO_EDGE_DET: No edge detection, use trigger as is */
    .startInput         = 0ul,  /* Select the constant 0 */
    .kill0InputMode     = 3ul,  /* NO_EDGE_DET: No edge detection, use trigger as is */
    .kill0Input         = 0ul,  /* Select the constant 0 */
    .kill1InputMode     = 3ul,  /* NO_EDGE_DET: No edge detection, use trigger as is */
    .kill1Input         = 0ul,  /* Select the constant 0 */
    .countInputMode     = 3ul,  /* NO_EDGE_DET: No edge detection, use trigger as is */
    .countInput         = 1ul,  /* Select the constant 1 */
};


int main(void)
{
    :
```

Relation of Trigger Multiplexer

Code Listing 24 CYT2 Series: Example to start Three TCPWM Simultaneous in Configuration Part

```
    /* Clock Configuration for TCPWMs */
    Cy_SysClk_PeriphAssignDivider(PCLK_TCPWM0_CLOCKS256,
 (cy_en_divider_types_t)CY_SYSCLK_DIV_16_BIT, 0ul);        /* U-phase Counter */
    Cy_SysClk_PeriphAssignDivider(PCLK_TCPWM0_CLOCKS257,
 (cy_en_divider_types_t)CY_SYSCLK_DIV_16_BIT, 0ul);        /* V-phase Counter */
    Cy_SysClk_PeriphAssignDivider(PCLK_TCPWM0_CLOCKS258,
 (cy_en_divider_types_t)CY_SYSCLK_DIV_16_BIT, 0ul);        /* W-phase Counter */

    Cy_SysClk_PeriphSetDivider((cy_en_divider_types_t)CY_SYSCLK_DIV_16_BIT, 0ul, 1ul;
        /* Divider 1 --> 80MHz / (1+1) = 40MHz */
    Cy_SysClk_PeriphEnableDivider((cy_en_divider_types_t)CY_SYSCLK_DIV_16_BIT, 0ul);

    /* Initialize and Enable PWM_ Counter */
    Cy_Tcpwm_Pwm_Init(TCPWM0_GRP1_CNT0, &MyPWM_config);  /* U-phase */
    :
    Cy_Tcpwm_Pwm_Enable(TCPWM0_GRP1_CNT0);

    Cy_Tcpwm_Pwm_Init(TCPWM0_GRP1_CNT1, &MyPWM_config);  /* V-phase */
    :
    Cy_Tcpwm_Pwm_Enable(TCPWM0_GRP1_CNT1);

    Cy_Tcpwm_Pwm_Init(TCPWM0_GRP1_CNT2, &MyPWM_config);  /* W-phase */
    :
    Cy_Tcpwm_Pwm_Enable(TCPWM0_GRP1_CNT2);

    /* Synchronize all counters */
    Cy_TrigMux_SwTrigger(TRIG_OUT_MUX_4_TCPWM_ALL_CNT_TR_IN2, TRIGGER_TYPE_EDGE, 1ul /*output*/);  /*Output the Reload
signal to TCPWM_ALL_CNT_TR_IN[2] */
    for(;;)
    {
      :
    }
}
```

(1 to 3)Configure and select the clock for the counters (See **Code Listing 3** to **Code Listing 5**)

(4)Configure the counter for U-phase (See **Code Listing 21**)

(5)Enable the counter for U-phase (See **Code Listing 22**)

(4)Configure the counter for V-phase (See **Code Listing 21**)

(5)Enable the counter for V-phase (See **Code Listing 22**)

(4)Configure the counter for W-phase (See **Code Listing 21**)

(5)Enable the counter for W-phase (See **Code Listing 22**)

(6)Start the all counters by software command (See **Code Listing 25**)

Code Listing 25 to **Code Listing 27** demonstrates an example program to configure Trigger Multiplexer in the driver part.

Code Listing 25 CYT2 Series: Example to Configure Trigger Multiplexer in Driver Part (Cy_TrigMux_SwTrigger)

```
/*******************************************************************************
* Function Name: Cy_TrigMux_SwTrigger
*******************************************************************************/
cy_en_trigmux_status_t Cy_TrigMux_SwTrigger(uint32_t trigLine, en_trig_type_t trigType, uint32_t outSel)
{
    cy_en_trigmux_status_t retVal = CY_TRIGMUX_INVALID_STATE;
    if (PERI->unTR_CMD.stcField.u1ACTIVATE == 0ul)
    {
        PERI->unTR_CMD.stcField.u8TR_SEL     = Cy_TrigMux_GetNo(trigLine);
        PERI->unTR_CMD.stcField.u5GROUP_SEL  = Cy_TrigMux_GetGroup(trigLine);
        PERI->unTR_CMD.stcField.u1TR_EDGE    = trigType;
        PERI->unTR_CMD.stcField.u1OUT_SEL    = outSel;
        PERI->unTR_CMD.stcField.u1ACTIVATE   = 1ul;
        retVal = CY_TRIGMUX_SUCCESS;
    }
    return retVal;
}
```

Generate the software trigger for trigLine.

See **Code Listing 26**

See **Code Listing 27**

Code Listing 26 CYT2 Series: Example to Configure Trigger Multiplexer in Driver Part (Cy_TrigMux_GetNo)

```
/*******************************************************************************
* Function Name: Cy_TrigMux_GetNo
*******************************************************************************/
static uint8_t Cy_TrigMux_GetNo(uint32_t trig)
{
    // Distill trigger selection field of input parameter
    return ((trig & CY_TR_MASK) >> CY_TR_SHIFT);
}
```

Select input trigger

Code Listing 27 CYT2 Series: Example to Configure Trigger Multiplexer in Driver Part (Cy_TrigMux_GetGroup)

```
/*******************************************************************************
* Function Name: Cy_TrigMux_GetGroup
*******************************************************************************/
static uint8_t Cy_TrigMux_GetGroup(uint32_t trig)
{
    // Distill group field of input parameter
    return ((trig & CY_TR_GROUP_MASK) >> CY_TR_GROUP_SHIFT);
}
```

Select trigger group

3.2 Starting AD Conversion with TCPWM Output

3.2.1 Use Case

This section describes an example for using the TCPWM trigger output as the start signal for AD conversion. Here, ch4 and ch5 of ADC are converted by Group0_counter0 and Group0_counter1 of TCPWM, respectively. The following is an example of configuring TCPWM and Trigger Multiplexer using SDL.

- TCPWM Operation Mode : PWM Mode
- Using Counter : TCPWM0/Group1/Counter0, 1
- Counter Start operation : Start by Software
- Using ADC Channel : Ch4, Ch5
- Using Triggers : Triggers One-to-One, MUX Group 1
  - For ADC Ch4: TCPWM0_16_TR_OUT1[4]
  - For ADC Ch5: TCPWM0_16_TR_OUT1[5]

Figure 21 shows an example of starting AD conversion with TCPWM outputs (tr_out1). Two “tr_out1” can be connected as a start trigger for the SAR ADC via a trigger multiplexer.

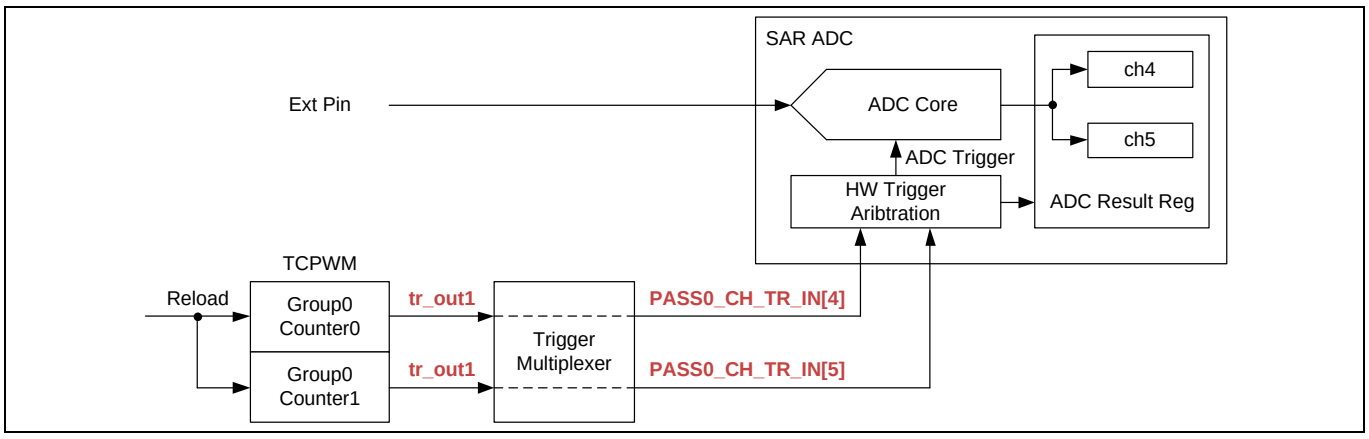

Figure 21 CYT2B7 Series: Starting AD conversion with TCPWM output

In addition, **Figure 22** shows a timing chart of AD conversion by a cc0 match event of each counter. The results of each AD conversion executed by the two start triggers, are stored in registers for each channel.

Relation of Trigger Multiplexer

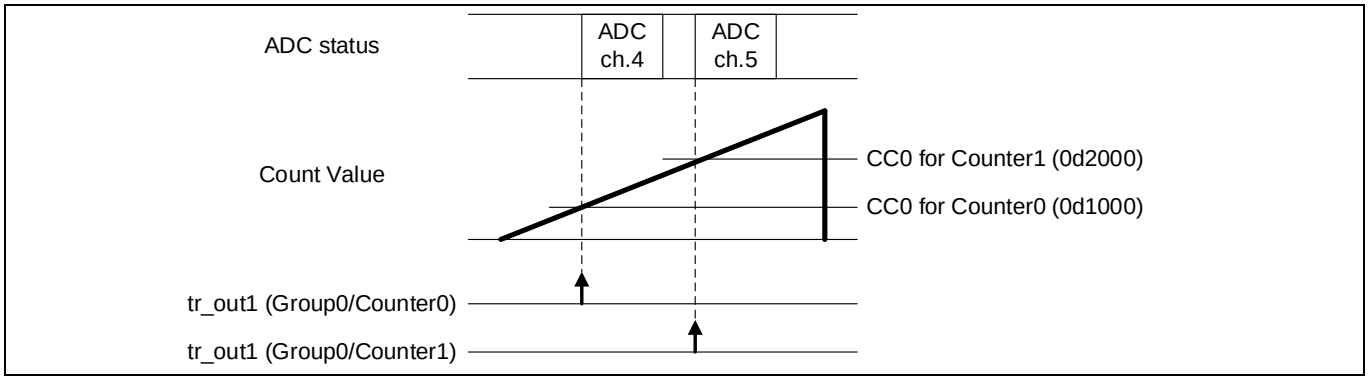

Figure 22 Timing Chart for AD conversion with TCPWM output

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

Figure 23 is a detailed block diagram of the trigger multiplexer for connecting tr_out1 to the trigger of the SAR ADC.

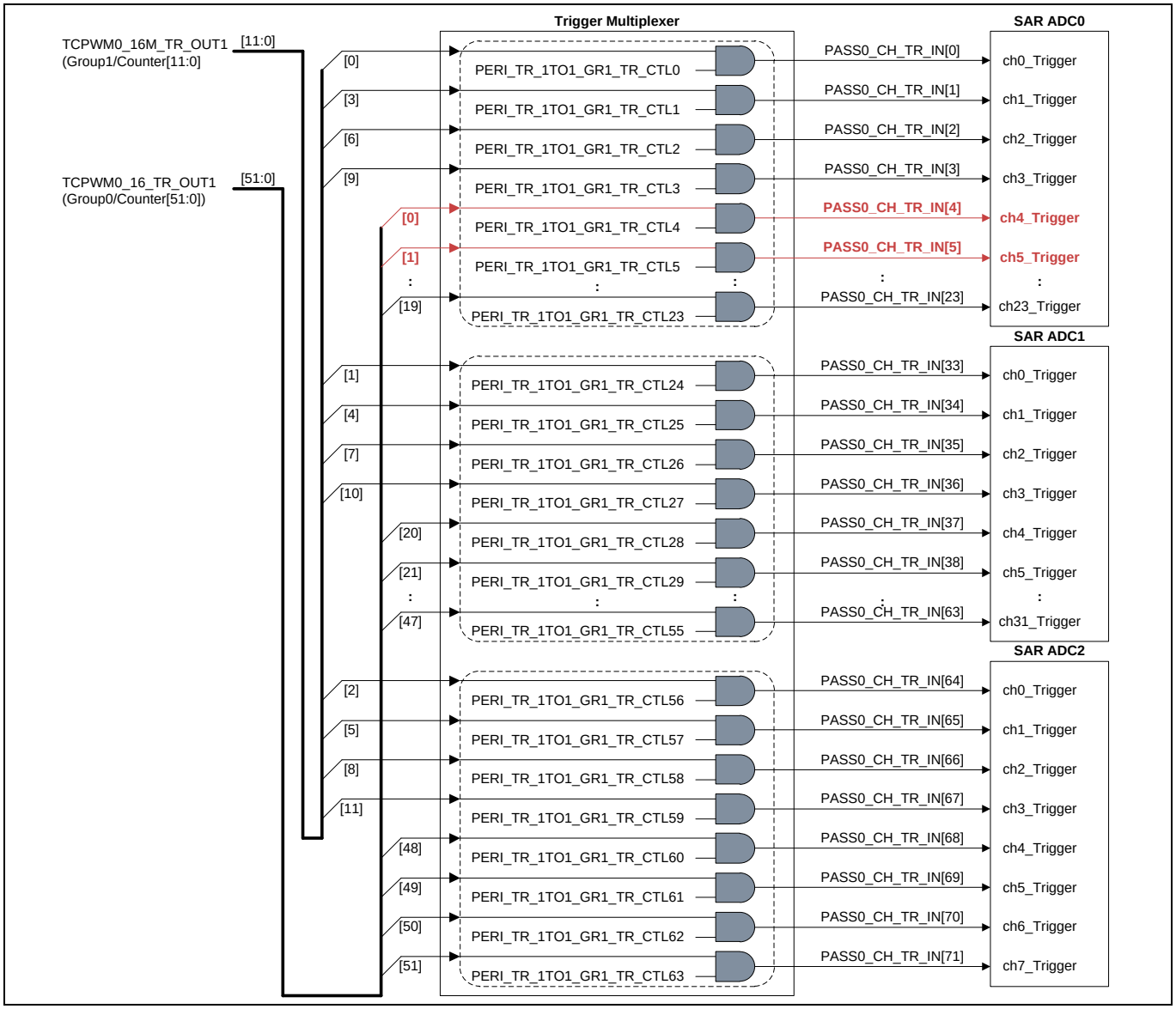

Figure 23 CYT2B7 Series: Detailed Block Diagram of the Trigger Multiplexer

Relation of Trigger Multiplexer

As shown, tr_out1 of Group0/Counter0 is connected to ch4 Trigger of SAR ADC0 via “one-to-one trigger groups” of the Trigger Multiplexer. Similarly, Group0/Counter1 is connected to ch5_Trigger of SAR ADC0. PASS0_CH_TR_IN4, 5 belongs to MUX Group1 of "one-to-one trigger groups". PASS0_CH_TR_IN4, 5 can be activated by the PERI_TR_1TO1_GR1_TR_CTL4 ,5 registers, respectively.

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

3.2.2 Configuration and Example

Table 9 lists the parameters of the configuration part in SDL for PWM Mode.

Table 9 CYT2 Series: Example to Start AD Conversion with TCPWM Output in Configuration Part

Parameters	Description	Setting Value
For TCPWM		
.period	Value of the counter (This field should be set to “n-1”)	0d8000
.clockPrescaler	Pre-scaling of the selected counter clock	CY_TCPWM_COUNTER_PRESCALER_DIVBY_1 (0x0)
.runMode	Counter run mode	CY_TCPWM_PWM_CONTINUOUS (0x0)
.countDirection	Counter direction	CY_TCPWM_COUNTER_COUNT_UP (0x0)
.debug_pause	Counter behavior in debug mode	false (0x0)
.CompareOrCapture	Counter mode	CY_TCPWM_COUNTER_MODE_COMPARE (0x0)
.enableCompare0Swap	Exchange the CC0 and buffered CC0 values	false (0x0)
.enableCompare1Swap	Exchange the CC1 and buffered CC1 values	false (0x0)
.interruptSources	Interrupt Mask bit	0x0 (zero clear)
.capture0InputMode	Capture0 edge mode	0x3 (NO_EDGE_DET)
.capture0Input	Input triggers for capture0	0x0 (Constant 0)
.reloadInputMode	Reload edge mode	0x3 (NO_EDGE_DET)
.reloadInput	Input triggers for reload	0x7 (TCPWM_ALL_CNT_TR_IN[2])
.startInputMode	Start edge mode	0x3 (NO_EDGE_DET)
.startInput	Input triggers for start	0x0 (Constant 0)
.stopInputMode	Stop edge mode	0x3 (NO_EDGE_DET)
.stopInput	Input triggers for stop	0x0 (Constant 0)
.capture1InputMode	Capture1 edge mode	0x3 (NO_EDGE_DET)
.capture1Input	Input triggers for capture1	0x0 (Constant 0)
.countInputMode	Count edge mode	0x3 (NO_EDGE_DET)
.countInput	Input triggers for count	0x1 (Constant 1)
.trigger1	Internal events to generate the output trigger 0	CY_TCPWM_COUNTER_OVERFLOW (0x0)
For Trigger Multiplexer (Cy_TrigMux_Connect1To1)		
trig	Number of Triggers One-to-One	TRIG_IN_1TO1_1_TCPWM_TO_PASS_CH_TR4 , 5

Relation of Trigger Multiplexer

Parameters	Description	Setting Value
trigType	Level Sensitive or Edge Sensitive for output trigger	TRIGGER_TYPE_PASS_TR_SAR_CH_IN__EDGE
For Trigger Multiplexer (Cy_TrigMux_SwTrigger)		
trigLine	Number of Triggers Group Outputs	TRIG_OUT_MUX_4_TCPWM_ALL_CNT_TR_IN2
trigType	Level Sensitive or Edge Sensitive for output trigger	TRIGGER_TYPE_EDGE
outSel	Specifies input trigger	0x1

Code Listing 28 demonstrates an example program to start AD Conversion with TCPWM output in the configuration part.

Code Listing 28 CYT2 Series: Example to Start AD Conversion with TCPWM Output in Configuration Part

```
Void Cy_Tcpwm_Counter_SetTROUT(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM);
/* Configuration for Timer for ADC */
cy_stc_tcpwm_counter_config_t MyCounter_config =
{
    .period             = 8000ul - 1ul,  /* 40,000,000 / 8000 = 5,000Hz (5kHz) */
    .clockPrescaler     = CY_TCPWM_COUNTER_PRESCALER_DIVBY_1,
    .runMode            = CY_TCPWM_PWM_CONTINUOUS,
    .countDirection     = CY_TCPWM_COUNTER_COUNT_UP,
    .debug_pause        = false,
    .CompareOrCapture   = CY_TCPWM_COUNTER_MODE_COMPARE,
    .enableCompare0Swap = false,
    .enableCompare1Swap = false,
    .interruptSources   = 0ul,
    .capture0InputMode  = 3ul,
    .capture0Input      = 0ul,
    .reloadInputMode    = 3ul,  /* NO_EDGE_DET: No edge detection, use trigger as is */
    .reloadInput        = 7ul,  /* Select the TCPWM_ALL_CNT_TR_IN[2] */
    .startInputMode     = 3ul,  /* NO_EDGE_DET: No edge detection, use trigger as is */
    .startInput         = 0ul,
    .stopInputMode      = 3ul,  /* NO_EDGE_DET: No edge detection, use trigger as is */
    .stopInput          = 0ul,
    .capture1InputMode  = 3ul,
    .capture1Input      = 0ul,
    .countInputMode     = 3ul,
    .countInput         = 1ul,
    .trigger1           = CY_TCPWM_COUNTER_OVERFLOW,
};

    :
int main(void)
{
    :
    /* Clock Configuration for TCPWMs */
    Cy_SysClk_PeriphAssignDivider(PCLK_TCPWM0_CLOCKS0, (cy_en_divider_types_t)CY_SYSCLK_DIV_16_BIT, 0ul);
    /* TCPWM_CNT0 for ADC */
    Cy_SysClk_PeriphAssignDivider(PCLK_TCPWM0_CLOCKS1, (cy_en_divider_types_t)CY_SYSCLK_DIV_16_BIT, 0ul);
    /* TCPWM_CNT1 for ADC */
    Cy_SysClk_PeriphSetDivider((cy_en_divider_types_t)CY_SYSCLK_DIV_16_BIT, 0ul, 1ul);
    /* Divider 1 --> 80MHz / (1+1) = 40MHz */
    Cy_SysClk_PeriphEnableDivider((cy_en_divider_types_t)CY_SYSCLK_DIV_16_BIT, 0ul);

    /* Trigger Multiplexer Setting (pass.tr_sar_ch_in[4]) */
    Cy_TrigMux_Connect1To1(TRIG_IN_1TO1_1_TCPWM_TO_PASS_CH_TR4,
                           0ul,
                           TRIGGER_TYPE_PASS_TR_SAR_CH_IN__EDGE,
                           0ul);
    /* Trigger Multiplexer Setting (pass.tr_sar_ch_in[5]) */
    Cy_TrigMux_Connect1To1(TRIG_IN_1TO1_1_TCPWM_TO_PASS_CH_TR5,
                           0ul,
                           TRIGGER_TYPE_PASS_TR_SAR_CH_IN__EDGE,
                           0ul);


    /* Initialize and Enable TCPWM Timer for ADC */
    Cy_Tcpwm_Counter_Init(TCPWM0_GRP0_CNT0, &MyCounter_config);    // TCPWM_CNT0 ADC
      :
    Cy_Tcpwm_Counter_SetCompare0(TCPWM0_GRP0_CNT0, 1000ul);
    Cy_Tcpwm_Counter_Enable(TCPWM0_GRP0_CNT0);
```

Configure the counter parameters

(1 to 3)Configure and select the clock for the counter (See **Code Listing 3** to **Code Listing 5**)

(4)Configure the Trigger Multiplexer (See **Code Listing 30**)

(4)Configure the counter for ADC Ch4 (See **Code Listing 6**)

(4)Set the compare value for ADC Ch4 (See **Code Listing 29**)

(5)Enable the counter for ADC Ch4 (See **Code Listing 8**)

Relation of Trigger Multiplexer

Code Listing 28 CYT2 Series: Example to Start AD Conversion with TCPWM Output in Configuration Part

```
    Cy_Tcpwm_Counter_Init(TCPWM0_GRP0_CNT1, &MyCounter_config);    // TCPWM_CNT1 ADC
      :
    Cy_Tcpwm_Counter_SetCompare0(TCPWM0_GRP0_CNT1, 2000ul);
    Cy_Tcpwm_Counter_Enable(TCPWM0_GRP0_CNT1);


    /* Synchronize all counters */
    Cy_TrigMux_SwTrigger(TRIG_OUT_MUX_4_TCPWM_ALL_CNT_TR_IN2, TRIGGER_TYPE_EDGE, 1ul /*output*/);  /* Output the
Reload signal to TCPWM_ALL_CNT_TR_IN[2] */

    for(;;)
    {
      :
    }
}
```

(4)Configure the counter for ADC Ch5 (See **Code Listing 6**)

(4)Set the compare value for ADC Ch5 (See **Code Listing 29**)

(5)Enable the counter for ADC Ch5 (See **Code Listing 8**)

(6)Start the all counters by software command (See **Code Listing 25**)

Code Listing 29 demonstrates an example program to configure the Trigger Multiplexer in the driver part.

Code Listing 29 CYT2 Series: Example to Start AD Conversion with TCPWM Output in Driver Part (Cy_Tcpwm_Counter_SetCompare0)

```
/*******************************************************************************
* Function Name: Cy_Tcpwm_Counter_SetCompare0
*******************************************************************************/
void Cy_Tcpwm_Counter_SetCompare0(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM,  uint32_t compare0)
{
    ptscTCPWM->unCC0.u32Register = compare0;
}
```

Sets the compare0 value for counter

Code Listing 30 and **Code Listing 31** demonstrates an example program to configure the Trigger Multiplexer in the driver part.

Code Listing 30 CYT2 Series: Example to Start AD Conversion with TCPWM Output in the Driver Part (Cy_TrigMux_Connect1To1)

```
/*******************************************************************************
* Function Name: Cy_TrigMux_Connect1To1
*******************************************************************************/
cy_en_trigmux_status_t Cy_TrigMux_Connect1To1(uint32_t trig, uint32_t invert,
en_trig_type_t trigType, uint32_t dbg_frz_en)                //connects an input trigger source and output trigger
{
    cy_en_trigmux_status_t retVal = CY_TRIGMUX_BAD_PARAM;

    /* Validate output trigger */
    if(Cy_TrigMux_IsOneToOne(trig) == false)
    {
        /* input trigger parameter is not One-To-One type */
        return retVal;
    }

    /* Distill group and trigger No value from input trigger parameter */
    uint8_t trigGroup = Cy_TrigMux_GetGroup(trig);
    uint8_t trigNo    = Cy_TrigMux_GetNo(trig);

    /* Get a pointer to target trigger setting register */
    volatile stc_PERI_TR_1TO1_GR_TR_CTL_field_t* pTR_CTL;
    pTR_CTL = &(PERI->TR_1TO1_GR[trigGroup].unTR_CTL[trigNo].stcField);

    /* Set input parameters to the register */
    pTR_CTL->u1TR_SEL        = true;
    pTR_CTL->u1TR_INV        = invert;
    pTR_CTL->u1TR_EDGE       = trigType;
    pTR_CTL->u1DBG_FREEZE_EN = dbg_frz_en;
```

Sets the compare0 value for counter

See **Code Listing 31**

See **Code Listing 27**

See **Code Listing 26**

Relation of Trigger Multiplexer

Code Listing 30 CYT2 Series: Example to Start AD Conversion with TCPWM Output in the Driver Part (Cy_TrigMux_Connect1To1)

```
    /* Return success status */
    retVal = CY_TRIGMUX_SUCCESS;
    return retVal;
```

Code Listing 31 CYT2 Series: Example to Start AD Conversion with TCPWM Output in the Driver Part (Cy_TrigMux_IsOneToOne)

```
/*******************************************************************************
* Function Name: Cy_TrigMux_IsOneToOne
*******************************************************************************/
static bool Cy_TrigMux_IsOneToOne(uint32_t trig)
{
    // Check trigger type bit field
    if((trig & CY_TR_TYPE_MASK) == 0ul)
    {
        // Trigger type normal (not one to one)
        return false;
    }
    else
    {
        // Trigger type one to one
        return true;
    }
}
```

checks whether trigger parameter is for One-To-One trigger or not

4 Glossary

Terms	Description
SAR ADC	Analog-to-digital converter. See the SAR ADC chapter of **Architecture TRM** for details.
P-DMA	Peripheral DMA
Peripheral Clock Divider	Peripheral Clock Divider derive 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 **Architecture TRM** for details.

5 Related Documents

The following are the Traveo II family series datasheets and Technical Reference Manuals. Contact **Technical Support** to obtain these documents.

- Device datasheet
  - CYT2B7 Datasheet 32-Bit Arm® Cortex®-M4F Microcontroller Traveo™ II Family
  - CYT2B9 Datasheet 32-Bit Arm® Cortex®-M4F Microcontroller Traveo™ II Family
  - CYT4BF Datasheet 32-Bit Arm® Cortex®-M7 Microcontroller Traveo™ II Family
  - CYT4DN Datasheet 32-Bit Arm® Cortex®-M7 Microcontroller Traveo™ II Family
  - CYT3BB/4BB Datasheet 32-Bit Arm® Cortex®-M7 Microcontroller Traveo™ II Family
- Body Controller Entry Family
  - Traveo™ II Automotive Body Controller Entry Family Architecture Technical Reference Manual (TRM)
  - Traveo™ II Automotive Body Controller Entry Registers Technical Reference Manual (TRM) for CYT2B7
  - Traveo™ II Automotive Body Controller Entry Registers Technical Reference Manual (TRM) for CYT2B9
- Body Controller High Family
  - Traveo™ II Automotive Body Controller High Family Architecture Technical Reference Manual (TRM)
  - Traveo™ II Automotive Body Controller High Registers Technical Reference Manual (TRM) for CYT4BF
  - Traveo™ II Automotive Body Controller High Registers Technical Reference Manual (TRM) for CYT3BB/4BB
- Cluster 2D Family
  - Traveo™ II Automotive Cluster 2D Family Architecture Technical Reference Manual (TRM)
  - Traveo™ II Automotive Cluster 2D Registers Technical Reference Manual (TRM)

6 Other References

Infineon provides the Sample Driver Library (SDL) including startup code as sample software to access various peripherals. SDL also serves as a reference to customers for drivers that are not covered by the official AUTOSAR products. The SDL cannot be used for production purposes because it does not qualify to any automotive standards. The code snippets in this application note are part of the SDL. Contact **Technical Support** to obtain the SDL.

Revision history

Revision	ECN No.	Submit Date	Description of Change
**	6103855	06/26/2019	New Application Note.
*A	6719536	10/31/2019	Added CYT4D Series
*B	6810474	03/11/2020	Changed target parts number (CYT2/CYT4 series). Added target parts number (CYT3 series).
*C	7035468	2020-12-02	Updated code examples using SDL MOVED TO INFINEON TEMPLATE.

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

Edition 2020-12-02

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Go to www.cypress.com/support

Document reference

002-20224 Rev.*C

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.

For further information on the product, technology delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

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.