TRAVEO™ T2G ファミリのタイマ, カウンタ, および PWM (TCPWM) の設定方法

本書について

適用範囲と目的

このアプリケーションノートは TRAVEO™ T2G ファミリに搭載されたタイマ, カウンタ, PWM の機能を持つ TCPWM の使い方について説明します。TCPWM はいくつかの機能モードに対応する多機能タイマコンポーネントです。

対象者

本書は、TRAVEO™ T2G ファミリの TCPWM 機能を使用するすべての人を対象とします。

関連製品ファミリ

TRAVEO™ T2G ファミリ CYT2/CYT3/CYT4/CYT6 シリーズ

目次

1 はじめに

このアプリケーションノートは TRAVEO™ T2G ファミリのマイコンに搭載された TCPWM の使い方について説明します。

- CYT2 シリーズは 1 つの Arm® Cortex®-M4F ベースの CPU (CM4) と 1 つの Cortex®-M0+ベースの CPU (CM0+) を搭載します。
- CYT4 シリーズは 2 つの Arm® Cortex®-M7 ベースの CPU (CM7) と 1 つの CM0+を搭載します。
- CYT3 シリーズは 1 つの Arm® CM7 と 1 つの CM0+を、CYT6 シリーズは 4 つの Arm® Cortex®-M7 ベースの CPU (CM7) と 1 つの CM0+を搭載します。

TCPWM は、いくつかの機能モードに対応する多機能カウンタで構成されます。

TCPWM は 16 ビット幅と 32 ビット幅のカウンタを搭載します。また、16 ビットカウンタにはモータコントロールに特化した機能に対応します。

各デバイスの TCPWM の搭載チャネル数については、デバイスデータシートを参照してください。

このアプリケーションノートではユースケースとともに、いくつかの機能モードの初期設定方法を説明します。

このアプリケーションノートに記載されている機能説明や用語について理解いただくためには、Architecture reference manual の Timer, Counter, and PWM 章を参照してください。

1.1 機能

表 1 に TCPWM の機能モードを示します。

表 1 TCPWM 機能モード

モード	説明
タイマ	カウンタはクロックサイクルのカウントイベント検知により、カウント値が増えたり減ったりします。
キャプチャ	カウンタはクロックサイクルのカウントイベント検知により、カウント値が増えたり減ったりします。キャプチャイベント検知時、カウント値はキャプチャレジスタにコピーされます。
QUAD	クアッドレイチャデコーダのカウンタは 2 つの入力によりカウント値が増えたり減ったりします。2 つの入力は X1, X2, X4 またはアップ/ダウンロータリーエンコードです。クアッドレイチャモードは 4 つのサブモードを持ち、カウンタは 0 から PERIOD または 0x8000 から 0x0000/0xffff の間を比較機能かキャプチャ機能のコンビネーションで動きます。
PWM	パルス幅変調クロックプリスケーラ有り
PWM_DT	パルス幅変調デッドタイム有り
PWM_PR	疑似ランダム PWM。16 ビット幅または 32 ビット幅のリニアフィードバックシフトレジスタ方式で疑似ランダムノイズ作成幅の可変機能有り
SR	シフトレジスタ機能は右方向にカウンタ値をシフトします。capture0 入力が次のカウンタ値の最上位ビットに使われます。また、シフトレジスタ (カウンタ) の可変タブからライン出力信号が駆動されます。

各カウンタは上記の多機能モードに対応します。どんな時でも、1 つのカウンタで動作するのは 1 つの機能モードです。違うカウンタであれば違う機能モードが動作できます。

詳細については、Architecture reference manual の Timer, Counter, and PWM 章を参照してください。

1.2 ブロックダイヤグラム

図 1 に TCPWM のブロックダイヤグラムを示します。

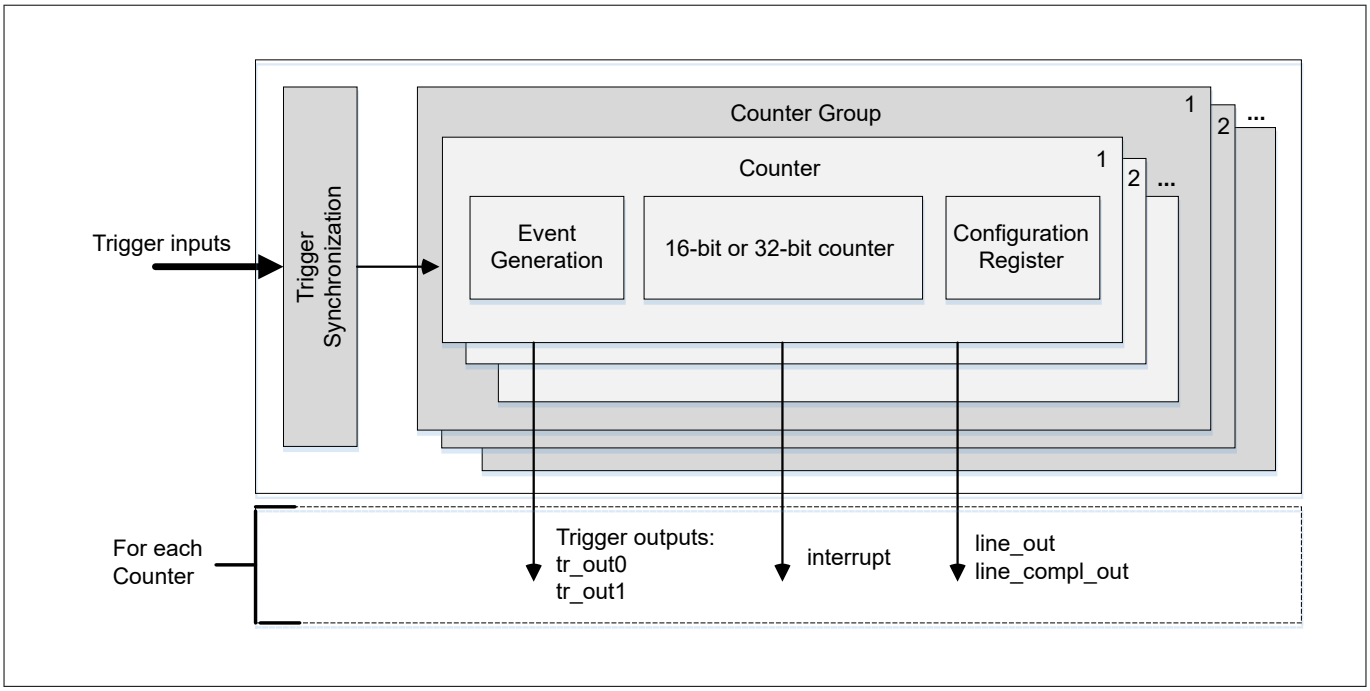

図 1　　TCPWM ブロックダイヤグラム

TCPWM はトリガ同期回路とカウンタ回路群で構成されます。各カウンタ回路群はカウンタ, 各カウンタのイベントジェネレータ, 16 ビット幅または 32 ビット幅のカウンタ, 設定レジスタで構成されます。

各カウンタは 2 つのトリガ出力 (tr_out0, tr_out1), 2 つのライン出力 (line_out, line_compl_out), 1 つの割込み出力(interrupt)を持ちます。

16 ビットカウンタはモータ制御の追加オプションを持っており、このカウンタはモータ制御に特化した機能を持ちます。

イベントジェネレータは 16 ビット幅または 32 ビット幅のカウンタ制御用のイベント (リロード, スタート, ストップ, カウント, キャプチャ) を発生します。そのイベントはリロード, スタート, ストップ, カウント, およびキャプチャイベントで、トリガ入力と連携します。

トリガ入力はトリガ同期回路により同期され、カウンタ回路へ入力されます。

TCPWM へはいくつかのトリガ入力が接続されます。GPIO 端子入力や SAR ADC のレンジ比較機能、固定の 0 または 1 入力、トリガ マルチプレクサによる汎用トリガ出力などがあります。

詳細については、Architecture reference manual の Trigger Multiplexer 章を参照してください。

図 2 に TCPWM とクロック供給のブロックダイヤグラムを示します。

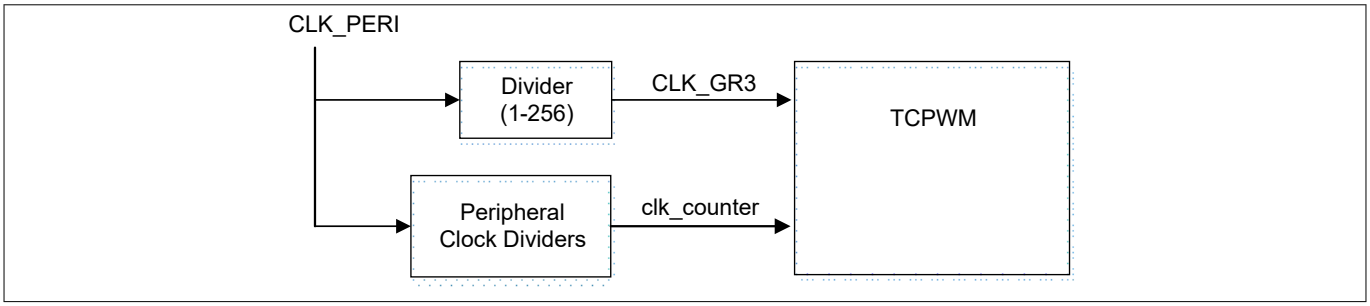

図 2　　CYT2B7 シリーズの TCPWM とクロック

TCPWM のシステムクロックはグループ 3 に属し、CLK_PERI を分周した CLK_GR3 が供給されます。このクロックはトリガ同期回路に使われます。

TCPWM の各カウンタクロックは CLK_PERI を周辺クロック分周器で分周した clk_counter が供給されます。

カウンタを有効化する前に、カウンタ用のクロックを選択してください。このクロックは周辺クロック分周器により生成されます。

図 3 に周辺クロック分周器のブロックダイヤグラムを示します。

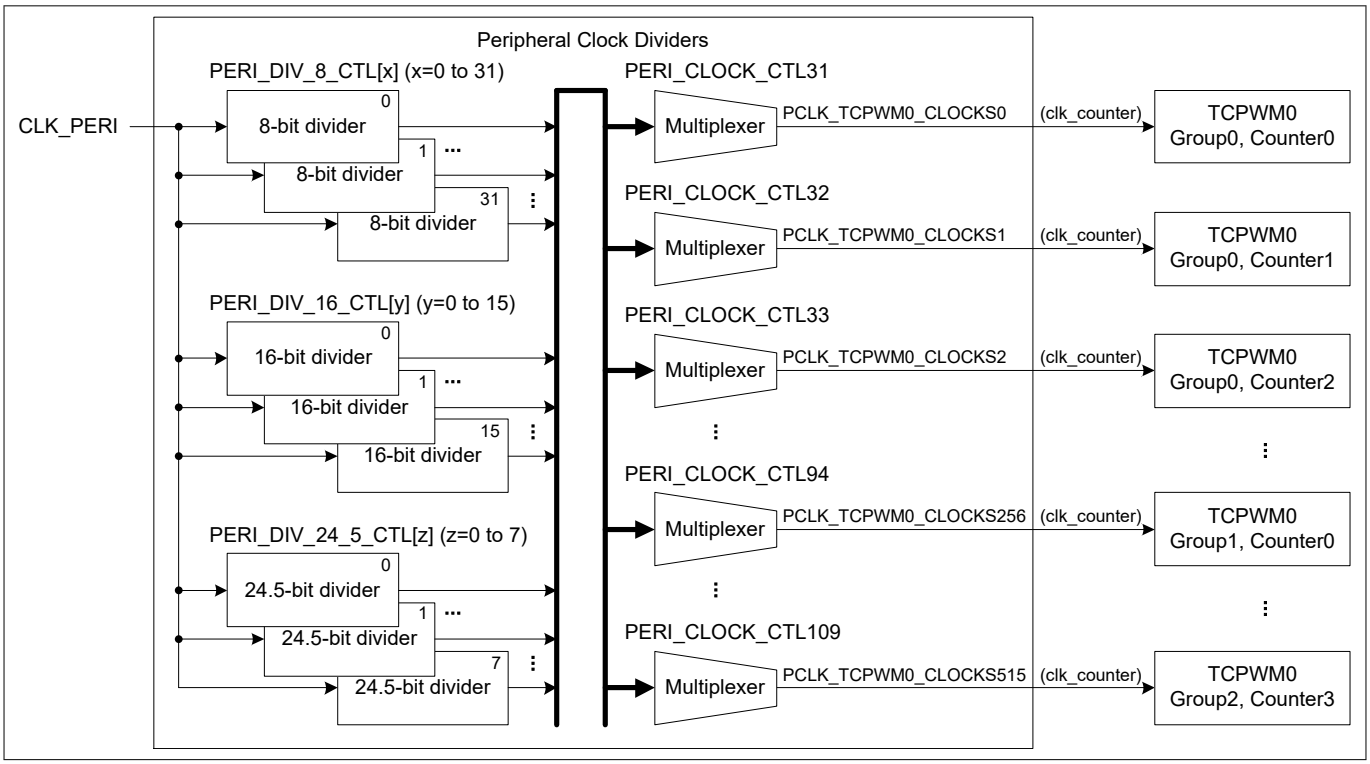

図 3　　CYT2B7 シリーズの周辺クロック分周器

周辺クロック分周器は 3 種類の分周機を持ちます。それらは 8 ビット分周器 (8.0 divider), 16 ビット分周器 (16.0 divider), 24.5 ビット分周器 (24.5 divider) です。各デバイス用の各分周機の搭載チャネル数についてはデバイスデータシートを参照してください。

各分周機は CLK_PERI クロックを分周しクロックを作ります。8 ビット分周器は CLK_PERI クロックを 1 から 2^8 分周まで、16 ビット分周器は CLK_PERI クロックを 1 から 2^16 分周まで分周します。また、24.5 ビット分周器は 24 ビットの整数分周機と 5 ビットの分数分周機を合わせ持ち、CLK_PERI クロックを 1 から 2^24 分周まで整数分周し 1 から 2^5 分周まで分数分周します。

周辺クロック分周器 clk_counter の出力は周辺クロックに含まれます。このアプリケーションノートとデバイスのデータシートでは、clk_counter は PCLK_TCPWM[m]_CLOCKS[n]と表記します (m = インプリメントされたモジュール番号、n = 周辺クロック番号)。ペリフェラルクロックは、各ペリフェラルモジュールに 1 対 1 で接続され、固有の番号を持ちます。表 2 に、CYT2B7 シリーズの TCPWM に接続されているペリフェラルクロック番号を示します。他のシリーズについては、デバイス データシートの“Peripheral clocks”を参照してください。

表 2　　CYT2B7 シリーズの TCPWM における周辺クロック数

周辺クロック番号	説明
PERI_CLOCK_CTL 31～93	TCPWM グループ#0, 16 ビットカウンタ#0 から#62 (63 ch)
PERI_CLOCK_CTL 94～105	TCPWM グループ#1, 16 ビットモータ用カウンタ#0 から#11 (12 ch)
PERI_CLOCK_CTL 106～109	TCPWM グループ#2, 32 ビットカウンタ#0 から#3 (4 ch)

詳細については、Architecture reference manual の Clocking System 章を参照してください。

2 TCPWM の動作例

ここではサンプルドライバライブラリ(SDL)を使用して、TCPWM を使用をする方法について説明します。このアプリケーションノートのプログラムコードは SDL の一部です。SDL については、その他の関連資料を参照してください。

SDL には設定部とドライバ部があります。設定部は、目的の操作のパラメータ値を設定します。ドライバ部は設定部のパラメータ値に基づいて各レジスタを設定します。ご使用のシステムに合わせて設定部を設定できます。

2.1 タイマモード

タイマモードの設定方法について説明します。

タイマモードは基本的なカウンタのアプリケーション向けの機能です。一般的なカウンタの使い方でクロックをカウントします。

カウンタの進み方として以下のモードがあります。

- COUNT_UP: カウントアップモード
- COUNT_DOWN: カウントダウンモード
- UPDOWN-COUNTER1 および UPDOWN-COUNTER2: カウントアップからカウントダウンするモード

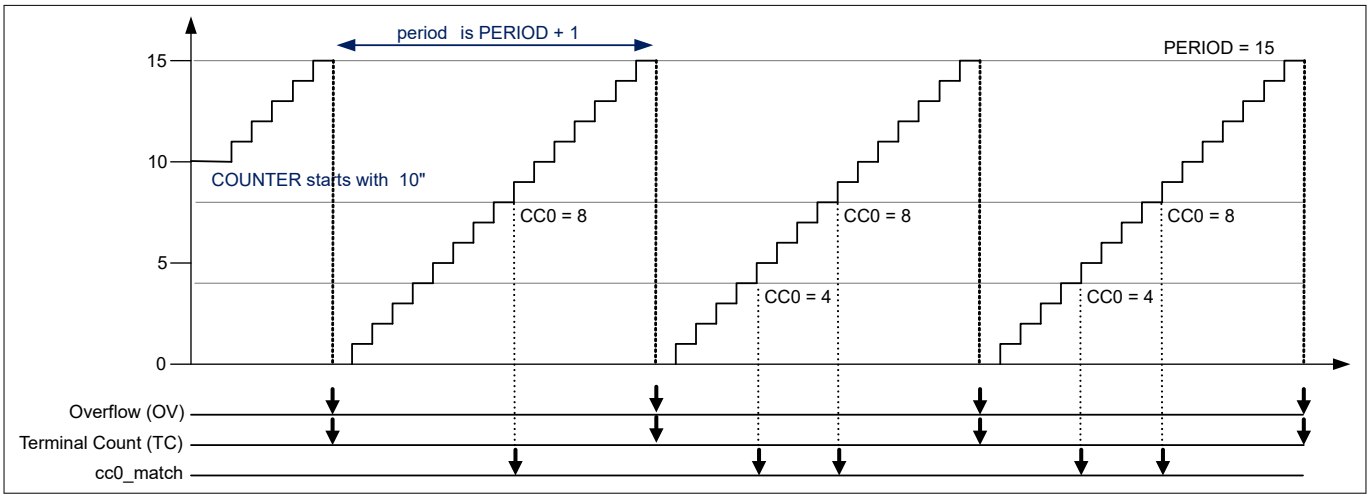

図 4 カウントアップモードのタイマモード

カウンタは初期値から開始します。カウンタレジスタ (例: TCPWM0_GRP0_CNT0_CONTER) を COUNTER = 10 と設定すると、カウンタは 10 から開始します。

カウンタはカウンタ値に合わせてイベントを発生します。このイベントは、アンダフロー (UV), オーバフロー (OV), ターミナルカウント (TC), cc0_match, および cc1_match の 5 つです。イベントの発生は動作モードと UP_DOWN_MODE の組合せによって変わります。アンダフローのイベントは COUNT_UP では発生しません。

オーバフローイベントはカウンタの値が PERIOD レジスタ (例: TCPWM0_GRP0_CNT0_PERIOD) の値と一致したときに発生します。

cc0_match イベントはカウンタの値が CC0 レジスタ値と一致したときに発生します。CC0 レジスタ (例: TCPWM0_GRP0_CNT0_CC0) の値は CC0_BUF レジスタ (例: TCPWM0_GRP0_CNT0_CC0_BUF) の値と cc0_match イベントが発生したときに切り替えられます。CC0 は CC0_BUF の値は関連するレジスタビットにより設定できます。図 4 に開始ポイントで CC0 レジスタ値が 8、CC0_BUF レジスタ値が 4 の時の波形を示します。cc0_match イベントのポイントで CC0 レジスタ値は 4 に変わり、CC0_BUF レジスタ値は 8 に変わります。

図 5 にイベントと割込み信号、tr_out0, tr_out1 信号の関係を含むタイマ機能を示します。

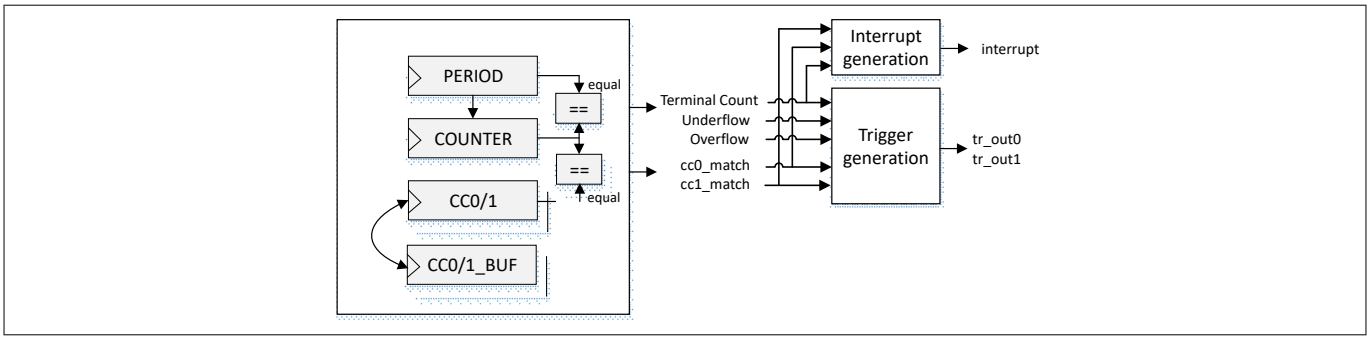

図 5　　タイマ機能

各イベントは tr_out0, tr_out1 よりトリガ信号として、または割込み信号として TCPWM から他のモジュールへ出力できます。

例えば、P-DMA を使った特定間隔データ転送のユースケースにおいて、特定間隔のトリガを cc0_match イベントで作成し、このトリガを P-DMA の起動信号に使えます。このトリガはトリガマルチプレクサモジュールにより P-DMA へ接続できます。

2.1.1　　ユースケース

このセクションでは、15,625Hz のカウンタ クロックで、1 秒カウンタ サイクル毎に割込みを発生させるタイマ モードの使用例について説明します。以下は、SDL を使用した TCPWM の設定例です。

- TCPWM 動作モード: タイマモード
- 使用カウンタ: TCPWM0/Group0/Counter0
- カウンタの開始操作: ソフトウェアから開始
- 入力クロック: clk_counter = 2 MHz: 分周値＝ 128 で割った値 (2 MHz/128 = 15,625 Hz)
- 割込み時間: 1 秒 (15,625 * (1/15,625 Hz) = 1 秒)
- システム割込みソース: TCPWM0/Group0/Counter0 (IDX: 274)
- CPU 割込みへ割当て: IRQ3
- CPU 割込み優先順位: 3

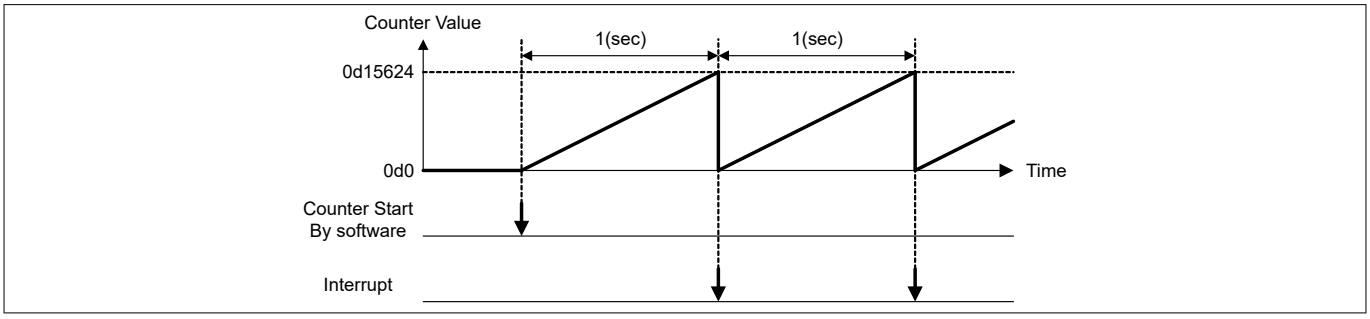

図 6　　タイマモードのタイミングチャート

図 7 に、このユースケースの動作フローを示します。

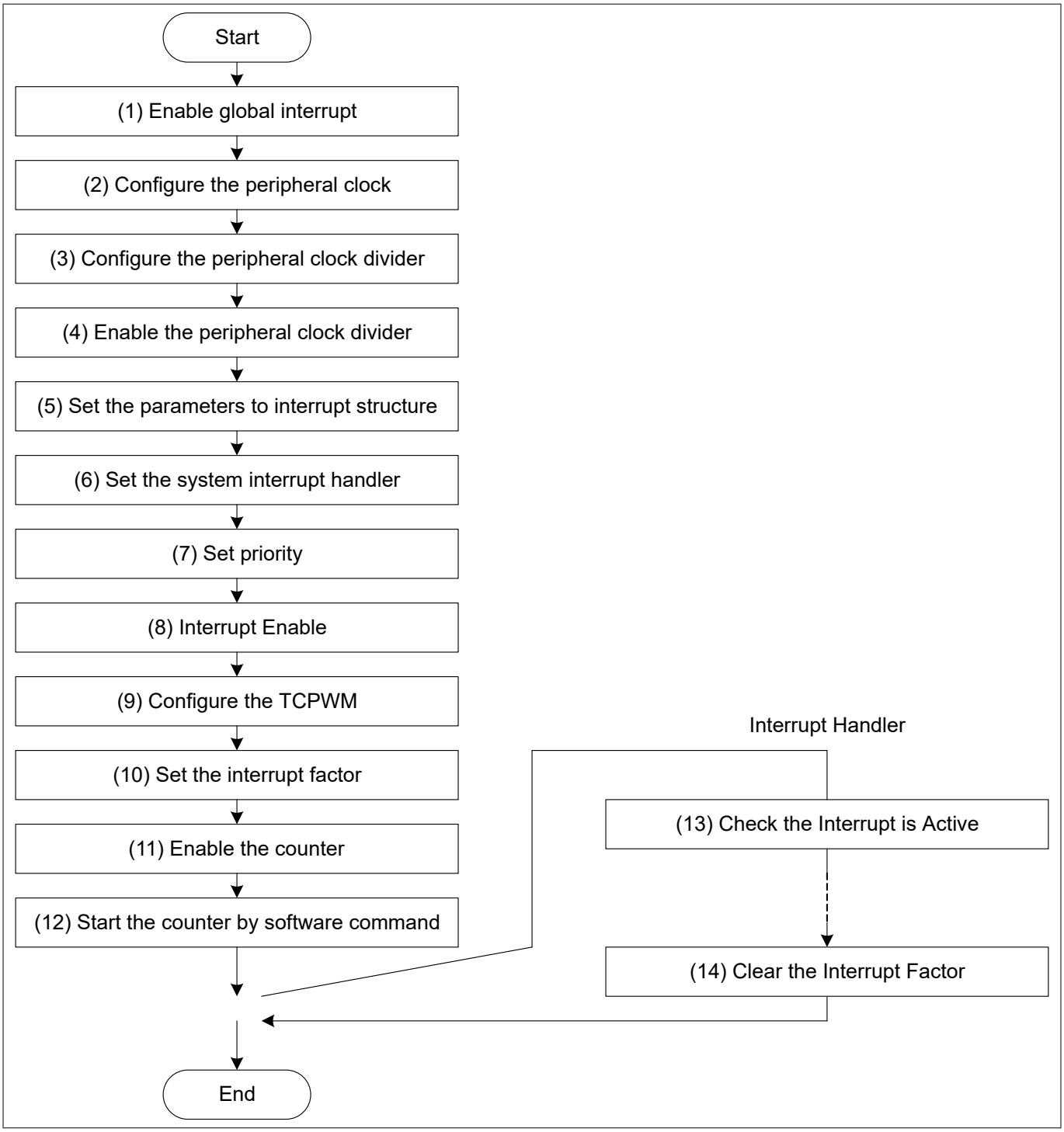

図 7　　動作フローの例

1. グローバル割込みを有効にしてください (CPU 割込みイネーブル)。詳細については、Architecture reference manual の CPU interrupt handing を参照してください。
2. TCPWM の周辺クロックを設定してください。
3. TCPWM 用の周辺クロック分周器を設定してください。
4. TCPWM の周辺クロックを有効にしてください。
5. 割込み構造を設定してください。詳細については、Architecture reference manual の CPU interrupt handing を参照してください。

6. システム割込みハンドラを設定してください。詳細については、Architecture reference manual の CPU interrupt handing を参照してください。
7. NVIC プライオリティレジスタでプライオリティを設定してください。詳細については、Architecture reference manual の CPU interrupt handing を参照してください。
8. NVIC 割込みコントローラで割込みを有効にしてください。詳細については、Architecture reference manual の CPU interrupt handing を参照してください。
9. TCPWM を設定してください。

   注: TCPWM カウンタが有効になっている場合は、誤動作を防ぐために無効にしてください。

10. TCPWM の割込み要素を設定してください。
11. TCPWM カウンタを有効にしてください。
12. ソフトウェア コマンドで TCPWM カウンタを開始してください。
13. 割込みが発生したら、割込みがアクティブであるかチェックしてください。
14. 割込み処理実行後、割込み要因をクリアしてください。

2.1.2 設定とサンプルコード

表 3 に、SDL タイマモードの設定部のパラメータを示します。

表 3 CYT2 シリーズのタイマモード設定パラメータの一覧

パラメータ	説明	設定値
クロック		
TCPWMx_GRPx_CNTx_COUNTER	使用する カウンタ番号	TCPWM0_GRP0_CNT0 (TCPWM0/ Group0/Counter0)
PCLK_TCPWMx_CLOCKSx_COUNTER	周辺クロック番号	PCLK_TCPWM0_CLOCKS0 (PERI_CLOCK_CTL31)
TCPWM_PERI_CLK_DIVIDER_NO_COUNTER	使用する分周器番号	0x0 (分周 0)
periFreq	周辺クロック周波数	80000000ul (80 MHz)
targetFreq	clk_counter の周波数	2000000ul (2 MHz)
TCPWM		
.period	カウンタ値 (このフィールドは "n-1 "に設定してください)	0d15624
.clockPrescaler	選択されたカウンタ クロックのプリスケーリング	CY_TCPWM_COUNTER_PRESCALER_DIVBY_128 (0x7)
.runMode	カウンタ動作モード	CY_TCPWM_PWM_CONTINUOUS (0x0)
.countDirection	カウンタ方向	CY_TCPWM_COUNTER_COUNT_UP (0x0)
.debug_pause	デバッグモードでのカウンタの動作	false (0x0)
.CompareOrCapture	カウンタ モード	CY_TCPWM_COUNTER_MODE_COMPARE (0x0)

(続く)

表 3 (続き) CYT2 シリーズのタイマモード設定パラメータの一覧

パラメータ	説明	設定値
.compare0	CC0 のカウンタ値との比較	0x0000
.compare0_buff	CC0 のカウンタ値との追加比較	0x0000
.compare1	CC1 のカウンタ値との比較	0x0000
.compare1_buff	CC1 のカウンタ値との追加比較	0x0000
.enableCompare0Swap	CC0 とバッファリングされた CC0 の値を入れ替えてください。	false (0x0)
.enableCompare1Swap	CC1 とバッファリングされた CC1 の値を入れ替えてください。	false (0x0)
.interruptSources	割込みマスクビット	0x0 (ゼロ消去)
.capture0InputMode	Capture0 エッジモード	0x3 (NO_EDGE_DET)
.capture0Input	capture0 の入力トリガ	0x0 (定数 0)
.reloadInputMode	リロードエッジモード	0x3 (NO_EDGE_DET)
.reloadInput	リロードの入力トリガ	0x0 (定数 0)
.startInputMode	開始エッジモード	0x3 (NO_EDGE_DET)
.startInput	開始の入力トリガ	0x0 (定数 0)
.stopInputMode	停止エッジモード	0x3 (NO_EDGE_DET)
.stopInput	停止の入力トリガ	0x0 (定数 0)
.capture1InputMode	Capture1 エッジモード	0x3 (NO_EDGE_DET)
.capture1Input	capture1 の入力トリガ	0x0 (定数 0)
.countInputMode	カウントエッジモード	0x3 (NO_EDGE_DET)
.countInput	カウントの入力トリガ	0x1 (定数 1)
.trigger1	出力トリガ 0 を生成する内部イベント	CY_TCPWM_COUNTER_OVERFLOW (0x0)
割込み		
irq_cfg.sysIntSrc	システム割込みインデックス番号	tcpwm_0_interrupts_0_IRQn
irq_cfg.intIdx	CPU 割込み番号	CPUIntIdx3_IRQn
.isEnabled	CPU 割込みイネーブル	true (0x1)

Code Listing 1 に、設定部にてタイマモードを設定するサンプルプログラムを示します。

Code listing 1 CYT2 シリーズの設定部でタイマ モードを設定する例

```
#define TCPWMx_GRPx_CNTx_COUNTER          TCPWM0_GRP0_CNT0    //Define Using Counter
#define PCLK_TCPWMx_CLOCKSx_COUNTER       PCLK_TCPWM0_CLOCKS0    //Define Peripheral Clock
#define TCPWM_PERI_CLK_DIVIDER_NO_COUNTER 0ul    //Define Peripheral Clock Divider



cy_stc_tcpwm_counter_config_t const MyCounter_config =    //Configure the counter parameters
{
  .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,
  .capture1InputMode  = 3ul,
  .capture1Input      = 0ul,
  .countInputMode     = 3ul,
  .countInput         = 1ul,
  .trigger1           = CY_TCPWM_COUNTER_OVERFLOW,
};

cy_stc_sysint_irq_t irq_cfg =    //Configure interrupt structure parameters*1
{
  .sysIntSrc  = tcpwm_0_interrupts_0_IRQn,
  .intIdx     = CPUIntIdx3_IRQn,
  .isEnabled  = true,
};

int main(void)
{
            :
  __enable_irq(); /* Enable global interrupts. */    //(1)Enable global interrupt*1

  /* Assign a programmable divider for TCPWM0_GRP0_CNT0 */
  //Calculation of division ratio
  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);    //(2)Configure the Peripheral Clock (See )

  /* Sets the 16-bit divider */
  Cy_SysClk_PeriphSetDivider(CY_SYSCLK_DIV_16_BIT, TCPWM_PERI_CLK_DIVIDER_NO_COUNTER,
(divNum-1ul));    //(3)Configure the integer division of the 16-bit divider (See Code listing 4)
  Cy_SysClk_PeriphEnableDivider((cy_en_divider_types_t)CY_SYSCLK_DIV_16_BIT,
TCPWM_PERI_CLK_DIVIDER_NO_COUNTER);    //(4)Enable the 16-bit divider (See Code listing 5)


  /* Configure Interrupt for TCPWMs */
  Cy_SysInt_InitIRQ(&irq_cfg);    //(5)Set the parameters to interrupt structure*1
  Cy_SysInt_SetSystemIrqVector(irq_cfg.sysIntSrc, Timer_Handler);    //(6)Set the system
interrupt handler*1


  /* Set the Interrupt Priority & Enable the Interrupt */
  //(7)Set priority*1
  NVIC_SetPriority(irq_cfg.intIdx, 3ul);    //(8)Interrupt Enable*1
  NVIC_EnableIRQ(irq_cfg.intIdx);

  /* Initialize TCPWM0_GPR0_CNT0 as Timer/Counter & Enable */
  Cy_Tcpwm_Counter_Init(TCPWMx_GRPx_CNTx_COUNTER, &MyCounter_config);    //(9)Configure the
counter based on above structure (See Code listing 6)
  Cy_Tcpwm_Counter_SetTC_IntrMask(TCPWMx_GRPx_CNTx_COUNTER);    //(10)Set the interrupt
factor*2 (See Code listing 7)
  Cy_Tcpwm_Counter_Enable(TCPWMx_GRPx_CNTx_COUNTER);    //(11)Enable the counter(See Code
listing 8)
  Cy_Tcpwm_TriggerStart(TCPWMx_GRPx_CNTx_COUNTER);    //(12)Start the counter (See Code listing
9)


          :
  for(;;);
}
```

*1: 詳細については、Architecture reference manual の CPU interrupt handing を参照してください。
*2: TC (ターミナル カウント): Tc イベントは、カウンタのアンダーフローとオーバーフロー イベントの論理和です。
Code Listing 2 に、割込みハンドラのサンプルコードを示します。

Code listing 2: 割込みハンドラ例

```
void Timer_Handler(void)    //Interrupt handler
{
  if(Cy_Tcpwm_Counter_GetTC_IntrMasked(TCPWMx_GRPx_CNTx_COUNTER) == 1ul)    //(13)Check if
Interrupt is Active (See Code listing 10)
  {
  :
    Cy_Tcpwm_Counter_ClearTC_Intr(TCPWMx_GRPx_CNTx_COUNTER);    //(14)Clear TC interrupt (See
Code listing 11)
  }
}
```

Code Listing 3〜Code Listing 5 に、ドライバ部で CLK を設定するサンプルプログラムを示します。
以下の説明は、SDL のドライバ部のレジスタ表記を理解するのに役立ちます。

- PERI->unCLOCK_CTL および PERI->unDIV は、Register reference manual に記載されている PERI_CLOCK_CTLx レジスタです。他のレジスタも同様です。“x”は、システム割込みインデックス番号を示します。
- パフォーマンス改善策: レジスタの設定パフォーマンスを向上させるために、SDL は完全な 32 ビットデータをレジスタに書き込みます。各ビットフィールドは、事前にビット書込み可能-なバッファに生成され、最終的な 32 ビットデータとしてレジスタに書き込まれます。
- レジスタの共用体と構造体については hdr/rev_x/ip の cyip_srss_v2.h および cyip_tcpwm_v2.h を参照してください。

Code listing 3 CYT2 シリーズのドライバ部で CLK を設定する例 (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;
}
```

Code listing 4 CYT2 シリーズのドライバ部で CLK を設定する例 (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)    //Configure the Peripheral Clock
{
if (Cy_SysClk_CheckDividerExisting(dividerType, dividerNum) == CY_DIVIDER_NOT_EXISTING)    //
Check if configuration parameter values are valid.
  {
    greturn CY_SYSCLK_BAD_PARAM;
  }
  if (dividerType == CY_SYSCLK_DIV_8_BIT)    //Check the dividerType
  {
    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;    //Select INT8_DIV bits
    }
    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;    //Select INT16_DIB
bits
    }
    else
    {
      return CY_SYSCLK_BAD_PARAM;
    }
  }
  else
  {
    /* return bad parameter */
    return CY_SYSCLK_BAD_PARAM;
  }
  return CY_SYSCLK_SUCCESS;
}
```

Code listing 5 CYT2 シリーズのドライバ部で CLK を設定する例 (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)    //
Enable the Peripheral Clock Divider
{
if(Cy_SysClk_CheckDividerExisting(dividerType, dividerNum) == CY_DIVIDER_NOT_EXISTING)    //
Check if configuration parameter values are valid.
  {
    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;
}
```

Code Listing 6〜Code Listing 11 に、ドライバ部で TCPWM を設定するサンプルプログラムを示します。

Code listing 6 CYT2 シリーズのドライバ部で TCPWM を設定する例 (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)    //Configure (Initialize) the counter
{
  uint32_t status = CY_RET_BAD_PARAM;
  if (config->trigger1 > 0x04ul || config->trigger2 > 0x04ul)    //Check if configuration
parameter values are valid.
  {
    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)    //The initial value of the counter is
determined according to each countDirection.
    {
      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;
    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 シリーズのドライバ部で TCPWM を設定する例 (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)    //The initial
value of the counter is determined according to each countDirection.
{
  ptscTCPWM->unINTR_MASK.stcField.u1TC = 1ul;
}
```

Code listing 8 CYT2 シリーズのドライバ部で TCPWM を設定する例 (Cy_Tcpwm_Counter_Enable)

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

Code listing 9 CYT2 シリーズのドライバ部で TCPWM を設定する例 (Cy_Tcpwm_TriggerStart)

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

Code listing 10 CYT2 シリーズのドライバ部で TCPWM を設定する例 (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)    //Get TC
interrupt masked
{
  return (uint8_t)(ptscTCPWM->unINTR_MASKED.stcField.u1TC);
}
```

Code listing 11 CYT2 シリーズのドライバ部で TCPWM を設定する例 (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)     //Clear TC
interrupt
{
  ptscTCPWM->unINTR.stcField.u1TC = 1ul;
  ptscTCPWM->unINTR.u32Register;
}
```

2.2 キャプチャモード

キャプチャモードの設定方法について説明します。

キャプチャモードは入力トリガに合わせてカウンタの値を取り込むアプリケーション向けの機能です。

図 8 にカウントアップモードのキャプチャモードを示します。

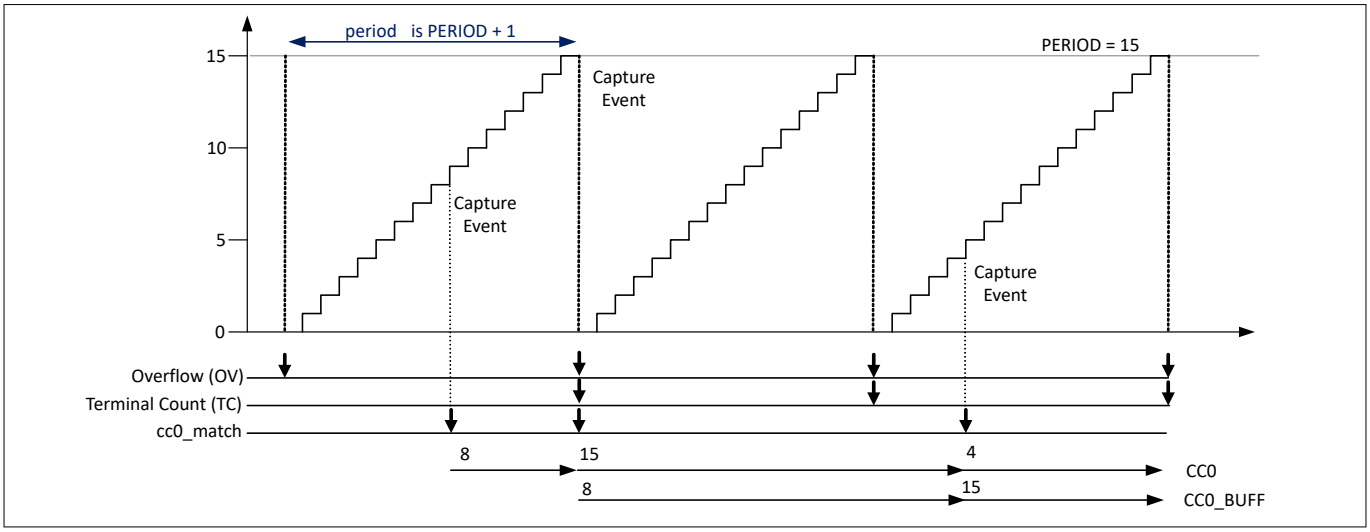

図 8 カウントアップモードのキャプチャモード

トリガ入力が認識されると、キャプチャイベントが発生しカウンタの値が CC0 レジスタに格納されます。それと同時に cc0_match イベントも発生します。

次の cc0_match イベントが発生すると、CC0 のレジスタ値は CC0_BUFF レジスタにコピーされ、カウンタの値は CC0 レジスタに格納されます。

TCPWM のカウンタは入力トリガを入力トリガソースから選択できます。各デバイスの各カウンタの搭載チャネル数についてはデバイスデータシートを参照してください。

表 4 に CYT2B7 シリーズの 16 ビットカウンタ 0 の入力トリガソースを示します。

表 4　　CYT2B7 シリーズでの 16 ビットカウンタ 0 の入力トリガ

トリガ番号	入力トリガ	入力トリガソース
0	Constant 0	0 入力 (カウントなし)
1	Constant 1	1 入力 (クロックでのカウント)
2	HSIOM 列 ACT#2	TC_0_TR0 (外部ピン, P3.1 または P6.1)
3	HSIOM 列 ACT#3	TC_0_TR1 (外部ピン, P3.2 または P6.2)
:	-	-
31	tr_all_cnt_in[26]	1 対 1 トリガの MAX グループ 4

注: これらは、Reference manual からの抜粋です。詳細については、Architecture reference manual の Timer, Counter, and PWM 章を参照してください。

TCPWM は入力トリガをいくつかのイベントに割り当てられます。キャプチャモードは次の 6 つのイベント、リロード, スタート, ストップ, カウント, capture0, capture1 が使えます。

2.2.1　　ユースケース

ここでは、I/O ポートからの入力トリガを capture0/1 イベントとして使用する場合の、キャプチャモードの使用例について説明します。割込みは、外部ピンの立ち上がりエッジと立ち下がりエッジで発生します。以下は、SDL を使用した TCPWM の設定例です。

- TCPWM 動作モード: キャプチャモード
- 使用カウンタ: TCPWM0/Group0/Counter0
- カウンタの開始操作: ソフトウェアから開始
- 入力クロック:
  - clk_counter = 2 MHz
  - 分周値＝ 4 で割った値 (2 MHz/4 = 500 kHz)
- capture0 イベントとして使用した I/O ポート: TC_0_TR0/1 (外部ピン)
- 割込み: capture0/1 のイベントが発生したとき。
- システム割込みソース: TCPWM0/Group0/Counter0 (IDX: 274)
- CPU 割込みへ割当て: IRQ3
- CPU 割込み優先順位: 3

ここでは外部ピンの詳細を説明しません。詳細については、Architecture reference manual の I/O System および Trigger Multiplexer の章を参照してください。

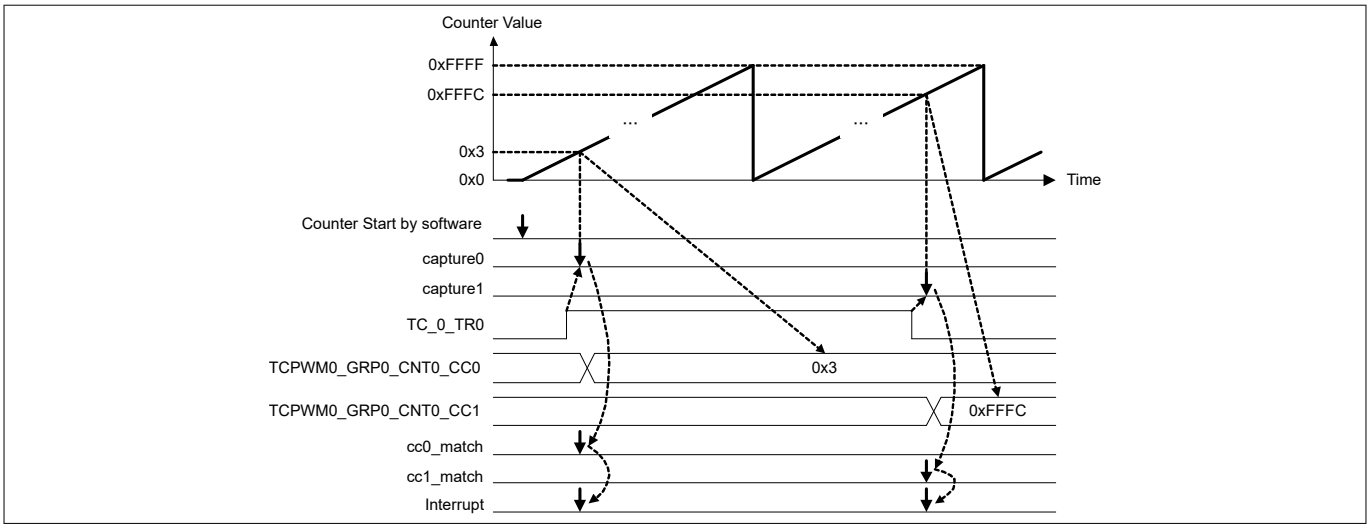

図 9　　キャプチャモードのタイミングチャート

注: capture0/1 信号は TC_0_TR0 の入力によって生成されます。TC0_0_TR0 からの入力は、常に図中のカウンタ値であるとは限らないことに注意してください。

図 10 に、このユースケースの動作フローを示します。

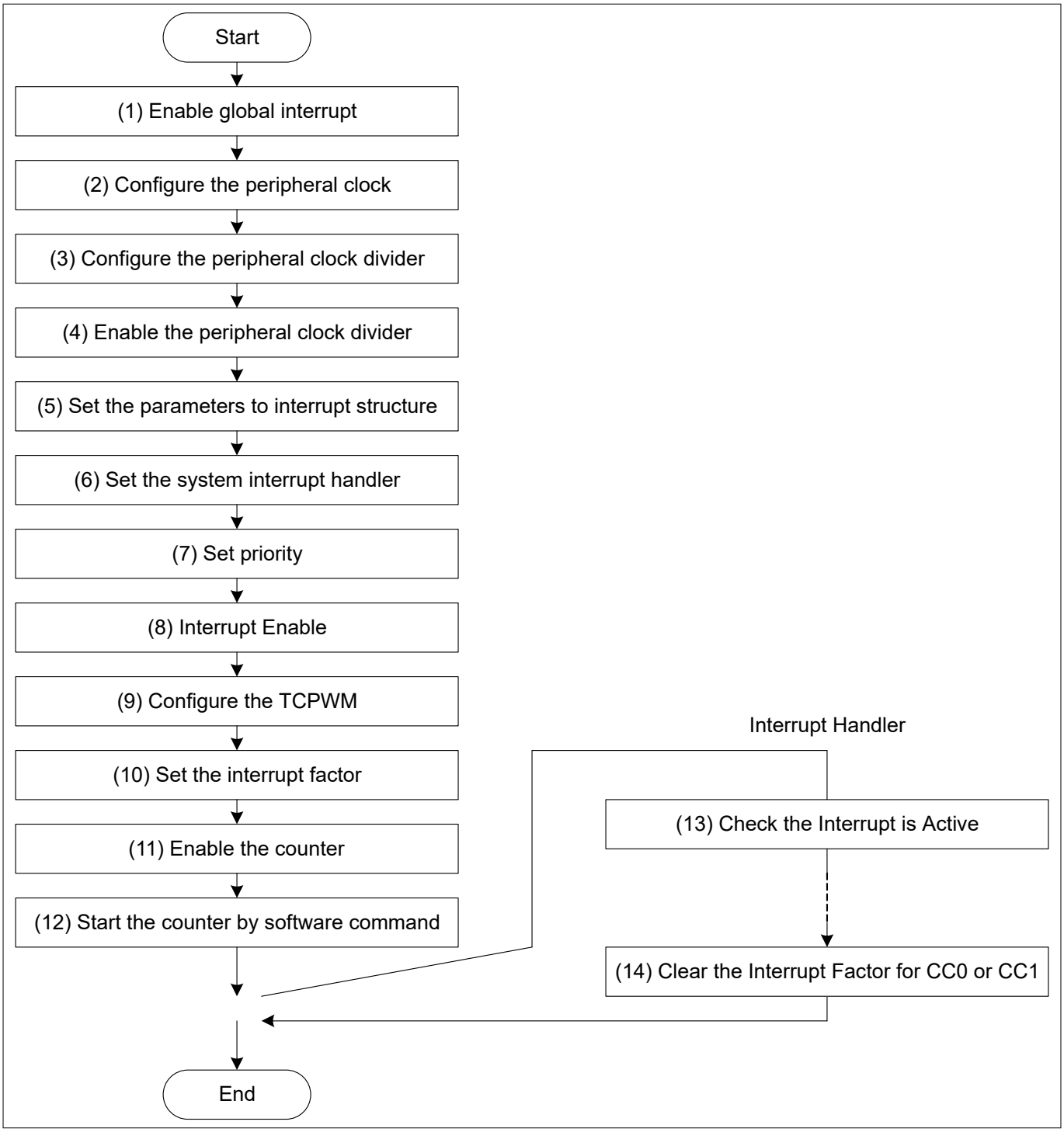

図 10　　動作フローの例

1. グローバル割込みを有効にしてください (CPU 割込みイネーブル)。詳細については、Architecture reference manual の CPU interrupt handing を参照してください。
2. TCPWM の周辺クロックを設定してください。
3. TCPWM 用の周辺クロック分周器を設定してください。
4. TCPWM の周辺クロックを有効にしてください。
5. 割込み構造を設定してください。詳細については、Architecture reference manual の CPU interrupt handing を参照してください。

6. システム割込みハンドラを設定してください。詳細については、Architecture reference manual の CPU interrupt handing を参照してください。
7. NVIC プライオリティレジスタでプライオリティを設定してください。詳細については、Architecture reference manual の CPU interrupt handing を参照してください。
8. NVIC 割込みコントローラで割込みを有効してください。詳細については、Architecture reference manual の CPU interrupt handing を参照してください。
9. TCPWM を設定してください。

   注: TCPWM カウンタが有効になっている場合は、誤動作を防ぐために無効にしてください。

10. TCPWM の割込み要素を設定してください。
11. TCPWM カウンタを有効にしてください。
12. ソフトウェア コマンドで TCPWM カウンタを開始してください。
13. 割込みが発生したら、割込みがアクティブであるかチェックしてください。
14. 割込み処理実行後、割込み要因 (CC0 または CC1) をクリアしてください。

2.2.2 設定とサンプルコード

表 5 に、キャプチャモードの SDL 設定部のパラメータを示します。

表 5 CYT2 シリーズのタイマモード設定パラメータの一覧

パラメータ	説明	設定値
クロック		
TCPWMx_GRPx_CNTx_CAPTURE	使用する カウンタ番号	TCPWM0_GRP0_CNT0
PCLK_TCPWMx_CLOCKSx_CAPTURE	周辺クロック番号	PCLK_TCPWM0_CLOCKS0
TR_ONE_CNT_NR_USE	入力トリガ	0x0 (定数 0)
TCPWMx_PERI_CLK_DIVIDER_NO	使用する分周器番号	0x0
periFreq	周辺クロック周波数	80000000ul (80MHz)
targetFreq	clk_counter の周波数	2000000ul (2MHz)
TCPWM		
.period	カウンタの上限値 (このフィールドは "n-1 "に設定してください)	0xffff
.clockPrescaler	選択されたカウンタ クロックのプリスケーリング	CY_TCPWM_COUNTER_PRESCALER_DIVBY_4 (0x2)
.runMode	カウンタ動作モード	CY_TCPWM_PWM_CONTINUOUS (0x0)
.countDirection	カウンタ方向	CY_TCPWM_COUNTER_COUNT_UP (0x0)
.debug_pause	デバッグモードでのカウンタの動作	false (0x0)
.CompareOrCapture	カウンタ モード	CY_TCPWM_COUNTER_MODE_CAPTURE (0x2)

(続く)

表 5 (続き) CYT2 シリーズのタイマモード設定パラメータの一覧

パラメータ	説明	設定値
.compare0	CC0 のカウンタ値との比較	0x0000
.compare0_buff	CC0 のカウンタ値との追加比較	0x0000
.compare1	CC1 のカウンタ値との比較	0x0000
.compare1_buff	CC1 のカウンタ値との追加比較	0x0000
.enableCompare0Swap	CC0 とバッファリングされた CC0 の値を入れ替えてください。	false (0x0)
.enableCompare1Swap	CC1 とバッファリングされた CC1 の値を入れ替えてください。	false (0x0)
.interruptSources	割込みマスクビット	0x0 (ゼロ消去)
.capture0InputMode	Capture0 エッジモード	0x0 (RISING_EDGE)
.capture0Input	capture0 の入力トリガ	0x2 (HSIOM 列 ACT#2)
.reloadInputMode	リロードエッジモード	0x3 (NO_EDGE_DET)
.reloadInput	リロードの入力トリガ	0x0 (定数 0)
.startInputMode	開始エッジモード	0x3 (NO_EDGE_DET)
.startInput	開始の入力トリガ	0x0 (定数 0)
.stopInputMode	停止エッジモード	0x3 (NO_EDGE_DET)
.stopInput	停止の入力トリガ	0x0 (定数 0)
.capture1InputMode	Capture1 エッジモード	0x1 (FALLING_EDGE)
.capture1Input	capture1 の入力トリガ	0x2 (HSIOM 列 ACT#2)
.countInputMode	カウントエッジモード	0x3 (NO_EDGE_DET)
.countInput	カウントの入力トリガ	0x1 (定数 1)
.trigger1	出力トリガ 0 を生成する内部イベント	CY_TCPWM_COUNTER_CC0_MATCH (0x3)
割込み		
irq_cfg.sysIntSrc	システム割込みインデックス番号	tcpwm_0_interrupts_0_IRQn
irq_cfg.intIdx	CPU 割込み番号	CPUIntIdx3_IRQn
.isEnabled	CPU 割込みイネーブル	true (0x1)

Code Listing 12 に、設定部でタイマモードを設定するサンプルプログラムを示します。

Code listing 12 CYT2 シリーズの設定部でキャプチャ モードを設定する例

```
#define TCPWMx_GRPx_CNTx_CAPTURE                TCPWM0_GRP0_CNT0    //Define Using Counter
#define PCLK_TCPWMx_CLOCKSx_CAPTURE             PCLK_TCPWM0_CLOCKS0    //Define Peripheral Clock
#define TR_ONE_CNT_NR_USE                  0ul // from 0 to 2    //Define Input Trigger
#define TCPWMx_PERI_CLK_DIVIDER_NO       0ul    //Define Peripheral Clock Divider




  :
cy_stc_tcpwm_counter_config_t const MyCounter_config =    Configure the counter parameters
  .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,
  .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 =    //Configure interrupt structure parameters*1
{
  .sysIntSrc  = tcpwm_0_interrupts_0_IRQn,
  .intIdx     = CPUIntIdx3_IRQn,
  .isEnabled  = true,
};

int main(void)
{
  __enable_irq(); /* Enable global interrupts. */    //(1)Enable global interrupt*1
  :
```

```
  uint32_t periFreq    = 80000000ul;
  uint32_t targetFreq = 2000000ul;
  uint32_t divNum = (periFreq / targetFreq);    //Calculation of division ratio
  :
  /* 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);    //(2)Configure the Peripheral Clock (See Code Listing 3)


  /* Sets the 16-bit divider */
  Cy_SysClk_PeriphSetDivider(CY_SYSCLK_DIV_16_BIT, TCPWMx_PERI_CLK_DIVIDER_NO, (divNum -
1ul));    //(3)Configure the integer division of the 16-bit divider (See Code listing 4)

  /* Enable the divider */
  Cy_SysClk_PeriphEnableDivider(CY_SYSCLK_DIV_16_BIT, TCPWMx_PERI_CLK_DIVIDER_NO);    //
(4)Enable the 16-bit divider (See Code listing 5)

  /* Interrupt setting for Capture  */
  Cy_SysInt_InitIRQ(&irq_cfg);    //(5)Set the parameters to interrupt structure*1
  Cy_SysInt_SetSystemIrqVector(irq_cfg.sysIntSrc, capture_isr_handler);    //(6)Set the system
interrupt handler*1

  /* Set the Interrupt Priority & Enable the Interrupt */
  NVIC_SetPriority(irq_cfg.intIdx, 3ul);    //(7)Set priority*1
  NVIC_EnableIRQ(irq_cfg.intIdx);     //(8)Interrupt Enable*1

            :
  /* Initialize PCLK_TCPWM0_CLOCKSx_CAPTURE as Capture Mode & Enable */
  Cy_Tcpwm_Counter_Init(TCPWMx_GRPx_CNTx_CAPTURE, &MyCounter_config);    //(9)Initialize the
counter based on above structure (See Code listing 6)
  Cy_Tcpwm_Counter_SetCC0_IntrMask(TCPWMx_GRPx_CNTx_CAPTURE);    //(10)Set the interrupt factor
(See Code listing 14)
  Cy_Tcpwm_Counter_SetCC1_IntrMask(TCPWMx_GRPx_CNTx_CAPTURE);    //(10)Set the interrupt factor
(See Code listing 15)
  Cy_Tcpwm_Counter_Enable(TCPWMx_GRPx_CNTx_CAPTURE);    //(11)Enable the counter (See Code
listing 8
  Cy_Tcpwm_TriggerStart(TCPWMx_GRPx_CNTx_CAPTURE);    //(12)Start the counter (See Code listing
9)


  :
  for(;;);
}
```

*1: 詳細については、Architecture reference manual の CPU interrupt handing を参照してください。
Code Listing 13 に、割込みハンドラのサンプルコードを示します。

Code listing 13: 割込みハンドラ例

```
void capture_isr_handler(void)    //Interrupt handler*1
{
  :
  if(Cy_Tcpwm_Counter_GetCC0_IntrMasked(TCPWMx_GRPx_CNTx_CAPTURE))    //(13)Check if Interrupt
is Active for CC0 (See Code listing 16)
  {
    // CCO would capture rising edge of input pulse
    Cy_Tcpwm_Counter_ClearCC0_Intr(TCPWMx_GRPx_CNTx_CAPTURE);    //(14)Clear TC interrupt for
CC0 (See Code listing 17)
  }
  :
  if(Cy_Tcpwm_Counter_GetCC1_IntrMasked(TCPWMx_GRPx_CNTx_CAPTURE))    //(13)Check if Interrupt
is Active for CC1 (See Code listing 18)
  {
    // CC1 would capture falling edge of input pulse
    Cy_Tcpwm_Counter_ClearCC1_Intr(TCPWMx_GRPx_CNTx_CAPTURE);    //(14)Clear TC interrupt for
CC1 (See Code listing 19)
  }
}
```

Code Listing 14～Code Listing 19 に、ドライバ部で TCPWM を設定するサンプルプログラムを示します。

Code listing 14 CYT2 シリーズのドライバ部で TCPWM を設定する例 (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);    //Configure
the interrupt factor for CC0
{
  ptscTCPWM->unINTR_MASK.stcField.u1CC0_MATCH = 0x1ul;
}
```

Code listing 15 CYT2 シリーズのドライバ部で TCPWM を設定する例 (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)    //Configure
the interrupt factor for CC1
{
  ptscTCPWM->unINTR_MASK.stcField.u1CC1_MATCH = 0x1ul;
}
```

Code listing 16 CYT2 シリーズのドライバ部で TCPWM を設定する例 (Cy_Tcpwm_Counter_GetCC0_IntrMasked)

```
/*******************************************************************************
* Function Name: Cy_Tcpwm_Counter_GetCC0_IntrMasked
*******************************************************************************/    //Get CC0
interrupt masked
uint8_t Cy_Tcpwm_Counter_GetCC0_IntrMasked(volatile stc_TCPWM_GRP_CNT_t *ptscTCPWM)
{
  return (uint8_t)(ptscTCPWM->unINTR_MASKED.stcField.u1CC0_MATCH);
}
```

Code listing 17 CYT2 シリーズのドライバ部で TCPWM を設定する例 (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)    //Clear CC0
interrupt
{
  ptscTCPWM->unINTR.stcField.u1CC0_MATCH = 1ul;
  ptscTCPWM->unINTR.u32Register;
}
```

Code listing 18 CYT2 シリーズのドライバ部で TCPWM を設定する例 (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)    //Get
CC1 interrupt masked
{
  return (uint8_t)(ptscTCPWM->unINTR_MASKED.stcField.u1CC1_MATCH);
}
```

Code listing 19 CYT2 シリーズのドライバ部で TCPWM を設定する例 (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)    //Clear CC1
interrupt
{
  ptscTCPWM->unINTR.stcField.u1CC1_MATCH = 1ul;
  ptscTCPWM->unINTR.u32Register;
}
```

2.3 PWM モード

PWM モードの設定方法について説明します。

PWM モードはパルス幅変調(PWM)信号を line_out と line_compl_out に乗せて出力するアプリケーション向けの機能です。

図 11 にカウントアップモードの PWM モードを示します。

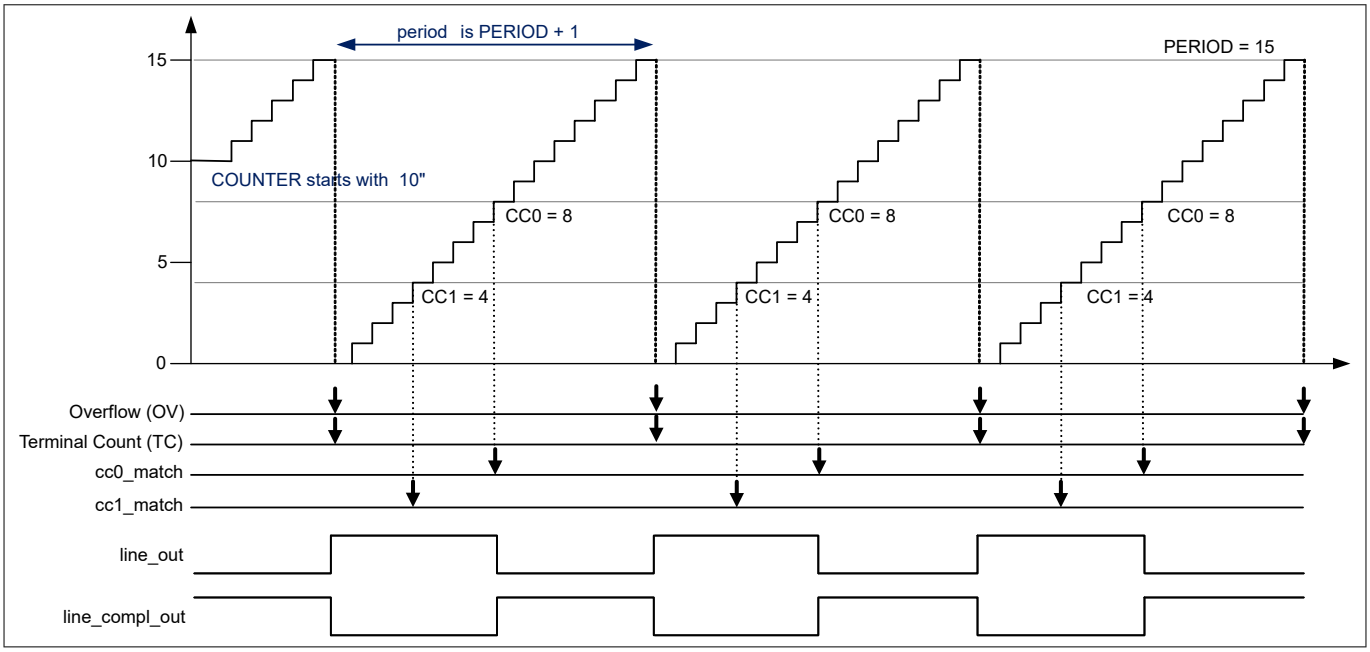

図 11 カウントアップモードの PWM モード

PWM 信号の周期は PERIOD レジスタで設定します。この PWM 信号の周期は PERIOD レジスタ値プラス 1 です。PWM 信号の Duty は CC0 または CC1 レジスタ (例: TCPWM0_GRP0_CNT0_CC1) で設定します。設定されたカウンタレジスタ値になると cc0_match または cc1_match イベントが発生します。

PWM 信号はオーバフロー, アンダフロー, cc0_match, および cc1_match のイベントを使って生成します。

図 12 に line 生成ロジックを示します。TR_PWM_CTL (例: TCPWM0_GRP0_CNT0_TR_PWM_CTRL) レジスタは、アンダーフロー、オーバーフロー、cc0_match、cc1_match の 4 つのイベントに従って、ラインの状態の変化を制御します。

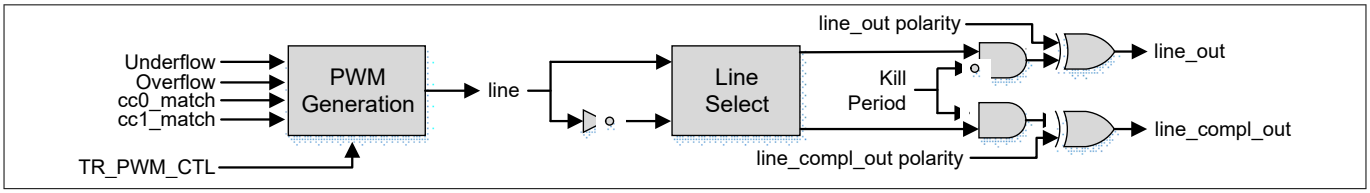

図 12 ライン生成ロジック

ライン出力は 2 つあります。PWM 信号は line_out から出力し、その相補の PWM 信号が line_compl_out から出力されます。関連する I/O 端子を PWM 信号出力として設定することで、line_out と line_compl_out は PWM と PWM_N として設定された I/O 端子から出力されます。PWM と PWM_N ポートは、CYT2B7 シリーズの I/O ポート P0.0 と P0.1 に割り当てられます。詳細についてはデバイスデータシートをご覧ください。

line_out、line_compl_out の極性は CTRL レジスタ (例: TCPWM0_GRP0_CNT0_CTRL) で設定できます。QUAD_ENCODING_MODE[0] ビットが line_out の極性を、QUAD_ENCODING_MODE[1] ビットが line_compl_out の極性を設定します。'1'設定で極性は反転します。

Kill period の入力により line_out と line_compl_out の両出力を停止できます。PWM_IMM_KILL, PWM_STOP_ON_KILL, PWM_SYNC_KILL のレジスタにより Kill モードの設定ができます。

カウンタ値は COUNTER レジスタにより設定できます。図 10 では、カウンタ値を設定することで“10”のカウンタ値からカウンタがスタートし、最初のオーバフローイベントの“15”までが待ち時間として設定できます。

アンダフロー, オーバフロー, cc0_match, および cc1_mach の 4 つのイベントはトリガとして出力できます。図 10 に示した cc1_match イベントは CC1 レジスタで任意のカウンタ値に設定できます。この cc1_match イベントは他の周辺回路、たとえば SAR ADC の起動トリガとしても使えます。

2.3.1 ユースケース

PWM モードのユースケースについて説明します。PWM 信号はオーバフローおよび cc0_match イベントで生成されます。以下は、SDL を使用した TCPWM の設定例です。

- TCPWM 動作モード: PWM モード
- 使用カウンタ: TCPWM0/Group0/Counter0
- カウンタの開始操作: ソフトウェアから開始
- PWM デューティ: 50%

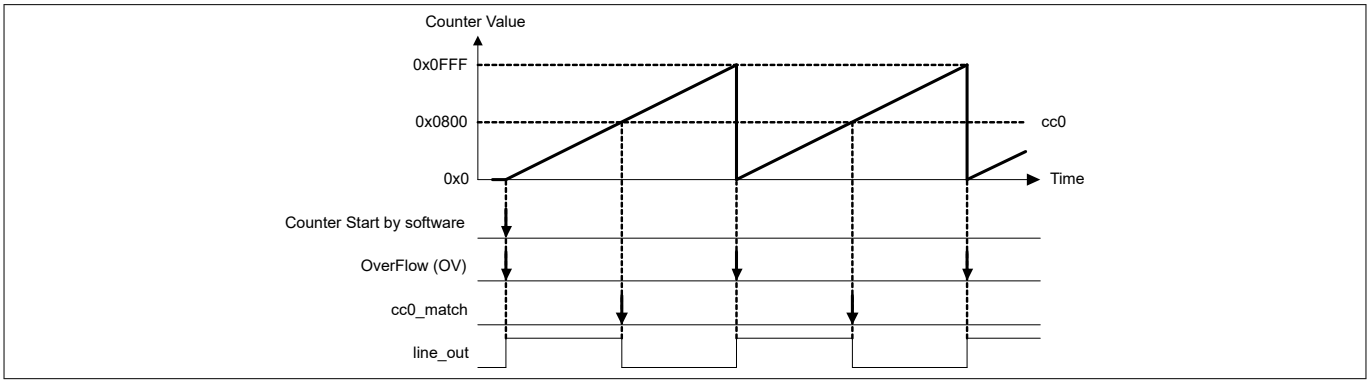

図 13 PWM モードのタイミングチャート

図 14 に、このユースケースの動作フローを示します。

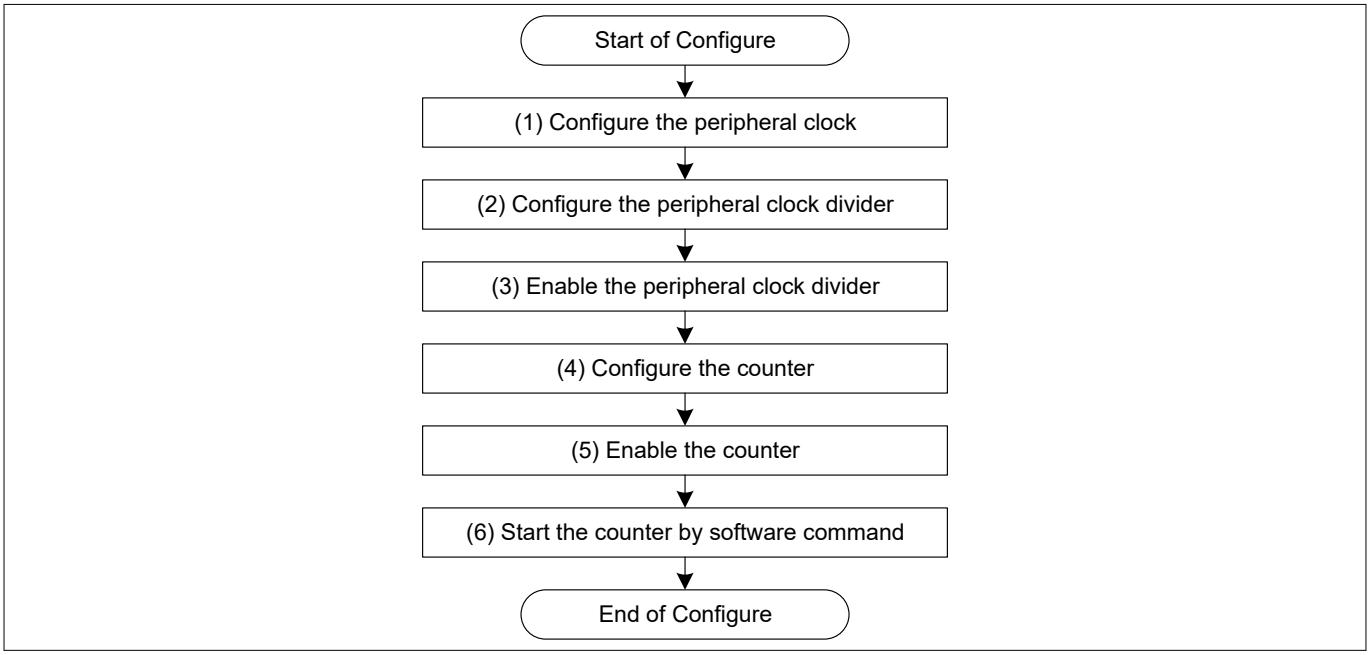

図 14 動作フローの例

1. TCPWM の周辺クロックを設定してください。
2. TCPWM 用の周辺クロック分周器を設定してください。

3. TCPWM の周辺クロックを有効にしてください。
4. TCPWM を設定してください。

   注: TCPWM カウンタが有効になっている場合は、誤動作を防ぐために無効にしてください。

5. TCPWM カウンタを有効にしてください。
6. ソフトウェア コマンドで TCPWM カウンタを開始してください。

2.3.2 設定とサンプルコード

表 6 に、PWM モードの SDL 設定部のパラメータを示します。

表 6 CYT2 シリーズの PWM モード設定パラメータの一覧

パラメータ	説明	設定値
クロック		
TCPWMx_GRPx_CNTx_PWM	使用する カウンタ番号	TCPWM0_GRP0_CNT0
PCLK_TCPWMx_CLOCKSx_PWM	周辺クロック番号	PCLK_TCPWM0_CLOCKS0
TCPWM_PERI_CLK_DIVIDER_NO_PWM	使用する分周器番号	0x0
TCPWMx_PWM_PRESCALAR_DIV_x	選択されたカウンタ クロックのプリスケーリング	CY_TCPWM_PWM_PRESCALER_DIVBY_128
sourceFreq	周辺クロック周波数	80000000ul (80 MHz)
targetFreq	clk_counter の周波数	2000000ul (2M Hz)
TCPWM		
.pwmMode	カウンタ モード	CY_TCPWM_PWM_MODE_PWM (0x4)
.clockPrescaler	選択されたカウンタ クロックのプリスケーリング	TCPWMx_PWM_PRESCALAR_DIV_x
.debug_pause	デバッグモードでのカウンタの動作	false (0x0)
.Cc0MatchMode	コンペアマッチ 0 イベントの効果を決定	CY_TCPWM_PWM_TR_CTRL2_CLEAR (0x1)
.OverflowMode	カウンタのオーバーフローイベントの影響を決定	CY_TCPWM_PWM_TR_CTRL2_SET (0x0)
.UnderflowMode	カウンタのアンダーフローイベントの影響を決定	CY_TCPWM_PWM_TR_CTRL2_NO_CHANGE (0x3)
.Cc1MatchMode	コンペアマッチ 1 イベントの効果を決定	CY_TCPWM_PWM_TR_CTRL2_NO_CHANGE (0x3)
.deadTime	デッドタイムの決定	- (このパラメータは PWM_DT モードでのみ有効です。)
.deadTimeComp	デッドタイムの決定	- (このパラメータは PWM_DT モードでのみ有効です。)
.runMode	カウンタ動作モード	CY_TCPWM_PWM_CONTINUOUS (0x0)

(続く)

表 6 (続き) CYT2 シリーズの PWM モード設定パラメータの一覧

パラメータ	説明	設定値
.period	カウンタ値 (このフィールドは "n-1 "に設定してください)	0x0FFF
.period_buff		0x0
.enablePeriodSwap	PERIOD とバッファリングされた PERIOD の値を入れ替えてください。	false (0x0)
.compare0	CC0 のカウンタ値との比較	TCPWMx_COMPARE0 (0x800)
.compare1	CC1 のカウンタ値との比較	0x0000
.enableCompare0Swap	CC0 とバッファリングされた CC0 の値を入れ替えてください。	false (0x0)
.enableCompare1Swap	CC1 とバッファリングされた CC1 の値を入れ替えてください。	false (0x0)
.interruptSources	割込みマスクビット	0x0 (ゼロ消去)
.invertPWMOut	PWM 出力 "line_out"と "line_compl_out"の動作	0x0
.invertPWMOutN		
.killMode	Kill イベントでカウンタが停止	CY_TCPWM_PWM_STOP_ON_KILL (0x2)
.switchInputMode	Capture0 エッジモード	0x3 (NO_EDGE_DET)
.switchInput	capture0 の入力トリガ	0x0 (定数 0)
.reloadInputMode	リロードエッジモード	0x3 (NO_EDGE_DET)
.reloadInput	リロードの入力トリガ	0x0 (定数 0)
.startInputMode	開始エッジモード	0x3 (NO_EDGE_DET)
.startInput	開始の入力トリガ	0x0 (定数 0)
.kill0InputMode	停止エッジモード	0x3 (NO_EDGE_DET)
.kill0Input	停止の入力トリガ	0x0 (定数 0)
.kill1InputMode	Capture1 エッジモード	0x3 (NO_EDGE_DET)
.kill1Input	capture1 の入力トリガ	0x0 (定数 0)
.countInputMode	カウントエッジモード	0x3 (NO_EDGE_DET)
.countInput	カウントの入力トリガ	0x1 (定数 1)

Code Listing 20 に、設定部で PWM モードを設定するサンプルプログラムを示します。

Code listing 20 CYT2 シリーズの設定部で PWM モードを設定する例

```
#define TCPWMx_GRPx_CNTx_PWM            TCPWM0_GRP0_CNT0    //Define Using Counter
#define PCLK_TCPWMx_CLOCKSx_PWM         PCLK_TCPWM0_CLOCKS0    //Define Peripheral Clock
#define TCPWM_PERI_CLK_DIVIDER_NO_PWM   0ul    //Define Peripheral Clock Divider

#define TCPWMx_PWM_PRESCALAR_DIV_x      CY_TCPWM_PWM_PRESCALER_DIVBY_128  // 2,000,000 / 128 =
15,625Hz    //Define Prescaler of counter Clock

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

  :
cy_stc_tcpwm_pwm_config_t const MyPWM_config =    //Configure the PWM parameters
{
  .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);    //Calculation of division ratio
  :

  /* 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);    //(1)Configure the Peripheral Clock (See Code listing 3)


  /* Sets the 16-bit divider */
  Cy_SysClk_PeriphSetDivider(CY_SYSCLK_DIV_16_BIT, TCPWM_PERI_CLK_DIVIDER_NO_PWM,
(divNum-1ul));    //(2)Configure the integer division of the 16-bit divider (See Code listing 4)



  /* Enable the divider */
  Cy_SysClk_PeriphEnableDivider(CY_SYSCLK_DIV_16_BIT, TCPWM_PERI_CLK_DIVIDER_NO_PWM);    //
(3)Enable the 16-bit divider (See Code listing 5)



  /* Initialize TCPWM0_GRPx_CNTx_PWM_PR as PWM Mode & Enable */
  Cy_Tcpwm_Pwm_Init(TCPWMx_GRPx_CNTx_PWM, &MyPWM_config);    //(4)Initialize the counter based
on above structure (See Code listing 21)
  Cy_Tcpwm_Pwm_Enable(TCPWMx_GRPx_CNTx_PWM);    //(5)Enable the counter (See Code listing 22)
  Cy_Tcpwm_TriggerStart(TCPWMx_GRPx_CNTx_PWM);    //(6)Start the counter (See Code listing 9)
  :
  for(;;);
}
```

Code Listing 21～Code Listing 22 に、ドライバ部で TCPWM を設定するサンプルプログラムを示します。

Code listing 21 CYT2 シリーズのドライバ部で TCPWM を設定する例 (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)    //Initialize the PWM counter
{
  uint32_t status = CY_RET_BAD_PARAM;    //Check if configuration parameter values are valid
  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.u 1 DBG__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)    //The initial value of the counter is
determined according to each countDirection
    {
      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)
    {
      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 シリーズのドライバ部で TCPWM を設定する例 (Cy_Tcpwm_Pwm_Enable)

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

2.4 PWM デッドタイム (PWM_DT) モード

PWM_DT モードの設定方法について説明します。

PWM_DT モードは、line_out と line_compl_out にてデッドタイム付き PWM 信号を出力するアプリケーション向けの機能です。

図 15 にカウントアップモードの PWM_DT モードを示します。

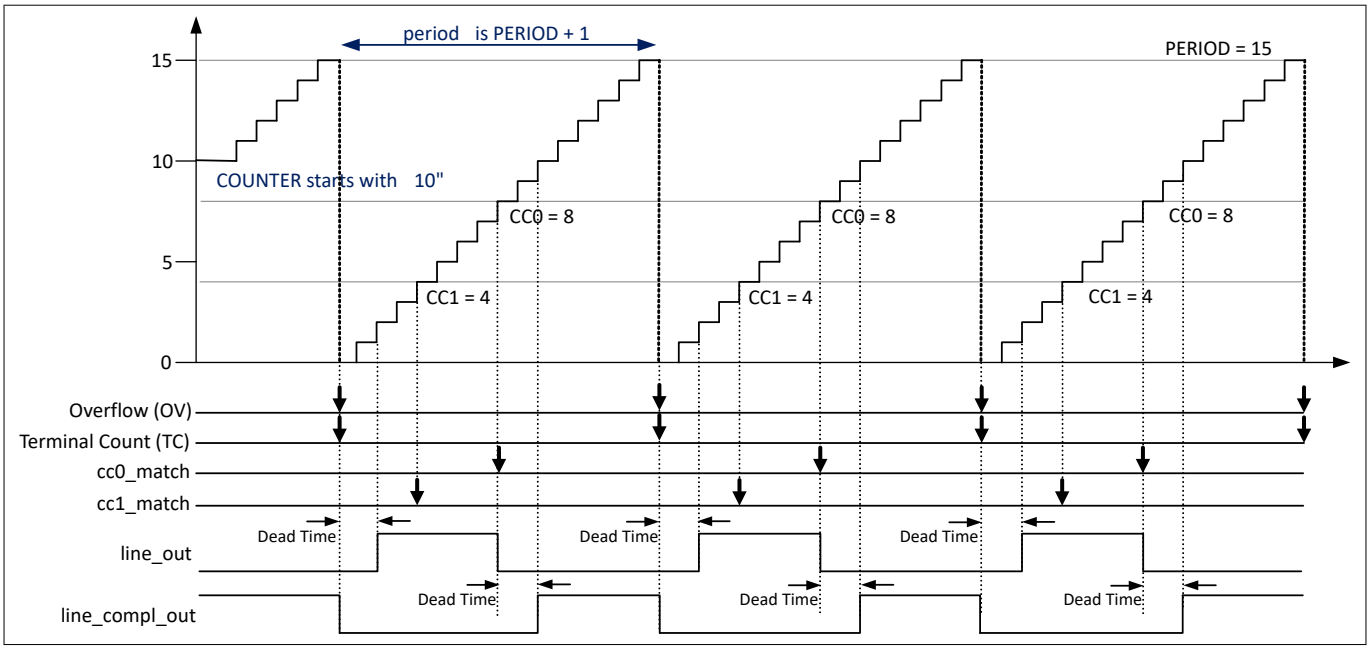

図 15 カウントアップモードの PWM_DT モード

デッドタイム付き PWM 信号は PWM モードと同じように設定します。PWM_DT モードは PWM モードと同様なモードで、PWM 信号にデッドタイムが付きます。

デッドタイムの設定は DT レジスタ (例: TCPWM0_GRP0_CNT0_DT) の DT_LINE_OUT_L ビットにより行います。デッドタイムは line_out, line_compl_out の各 PWM 立上りエッジに付加されます。両 line_out, line_compl_out ともデッドタイムの幅は同じです。

いくつかの 16 ビットカウンタはモータ制御に特化した機能を持ちます。この場合、line_out のデッドタイムは DT レジスタの DT_LINE_OUT_L と DT_LINE_OUT_H ビットにより設定でき、line_compl_out のデッドタイムは DT レジスタの DT_LINE_COMPL_OUT ビットにより設定できます。このため、line_out と line_compl_out のデッドタイム幅は別々の値を設定できます。

2.4.1 ユースケース

PWM_DT モードのユースケースについて説明します。PWM_DT 信号はオーバフローおよび cc0_match イベントで生成されます。以下は、SDL を使用した TCPWM の設定例です。

- TCPWM 動作モード: PWM_DT モード
- 使用カウンタ: TCPWM0/Group1/Counter0
- カウンタの開始操作: ソフトウェアから開始
- PWM デューティ: 50%
- カウンタ クロックのデッドタイム サイクル量: Line_out = 0d500 : Line_compl_out = 0d1000

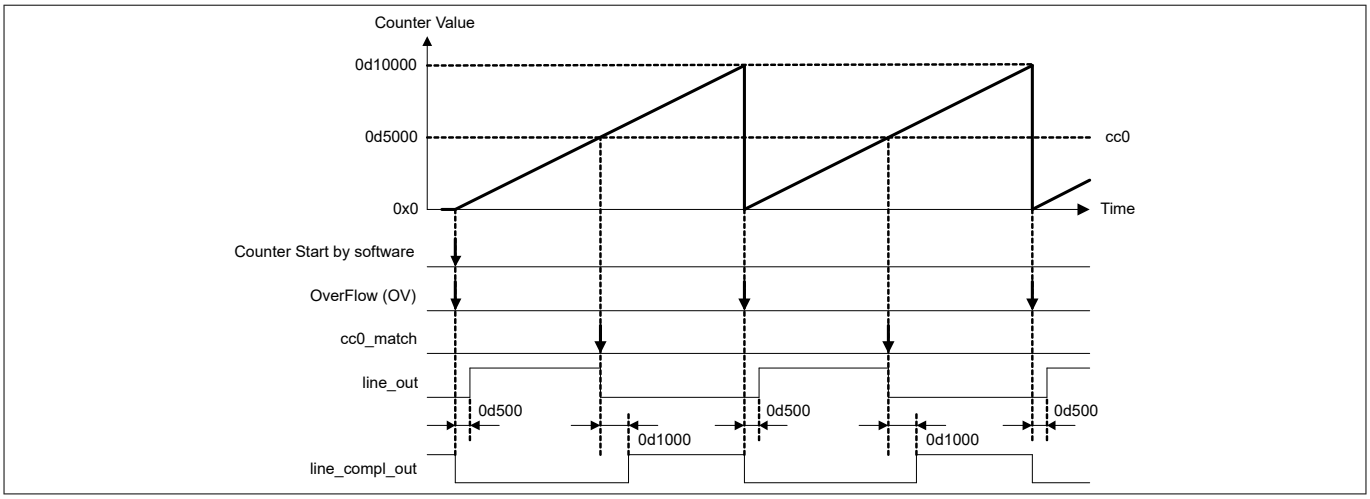

図 16　　PWM_DT モードのタイミングチャート

図 17 に、このユースケースの動作フローを示します。

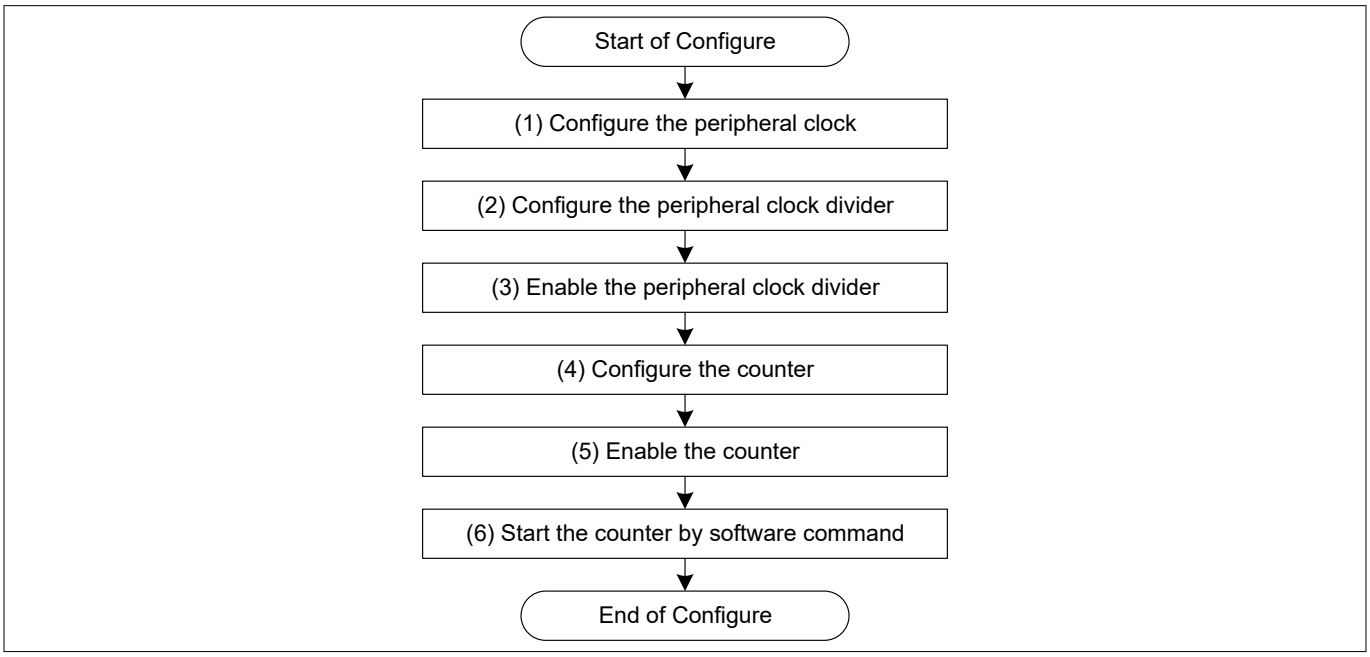

図 17　　動作フローの例

1. TCPWM の周辺クロックを設定してください。
2. TCPWM 用の周辺クロック分周器を設定してください。
3. TCPWM の周辺クロックを有効にしてください。
4. TCPWM を設定してください。

   ***注**: TCPWM カウンタが有効になっている場合は、誤動作を防ぐために無効にしてください。*

5. TCPWM カウンタを有効にしてください。
6. ソフトウェア コマンドで TCPWM カウンタを開始してください。

2.4.2　　設定とサンプルコード

表 7 に、PWM モードの SDL 設定部のパラメータを示します。

表 7 CYT2 シリーズの PWM_DT モード設定パラメータの一覧

パラメータ	説明	設定値
クロック		
TCPWMx_GRPx_CNTx_PWM_DT	使用する カウンタ番号	TCPWM0_GRP1_CNT0
PCLK_TCPWMx_CLOCKSx_PWM_DT	周辺クロック番号	PCLK_TCPWM0_CLOCKS256
TCPWM_PERI_CLK_DIVIDER_NO_PWM_DT	使用する分周器番号	0x0
TCPWMx_PERIOD	PWM 時間	0d10000
TCPWMx_COMPARE0	比較時間	0d5000
TCPWMx_DEADTIME	line_out のデッドタイム	0d500
TCPWMx_DEADTIME_COMPL	line_compl_out のデッドタイム	0d1000
sourceFreq	周辺クロック周波数	80000000ul (80 MHz)
targetFreq	clk_counter の周波数	2000000ul (2 MHz)
TCPWM		
.pwmMode	カウンタ モード	CY_TCPWM_PWM_MODE_DEADTIME (0x5)
.clockPrescaler	選択されたカウンタ クロックのプリスケーリング	CY_TCPWM_PWM_PRESCALER_DIVBY_1 (0x0)
.debug_pause	デバッグモードでのカウンタの動作	false (0x0)
.Cc0MatchMode	コンペアマッチ 0 イベントの効果を決定	CY_TCPWM_PWM_TR_CTRL2_CLEAR (0x1)
.OverflowMode	カウンタのオーバーフローイベントの影響を決定	CY_TCPWM_PWM_TR_CTRL2_SET (0x0)
.UnderflowMode	カウンタのアンダーフローイベントの影響を決定	CY_TCPWM_PWM_TR_CTRL2_NO_CHANGE (0x3)
.Cc1MatchMode	コンペアマッチ 1 イベントの効果を決定	CY_TCPWM_PWM_TR_CTRL2_NO_CHANGE (0x3)
.deadTime	デッドタイムの決定	TCPWMx_DEADTIME (0d500)
.deadTimeComp	デッドタイムの決定	TCPWMx_DEADTIME_COMPL (0d1000)
.runMode	カウンタ動作モード	CY_TCPWM_PWM_CONTINUOUS (0x0)
.period	カウンタ値 (このフィールドは "n-1 "に設定してください)	TCPWMx_PERIOD - 1ul (0d9999)
.period_buff		0d0
.enablePeriodSwap	PERIOD とバッファリングされた PERIOD の値を入れ替えてください。	false (0x0)
.compare0	CC0 のカウンタ値との比較	TCPWMx_COMPARE0 (0d5000)
.compare1	CC1 のカウンタ値との比較	0d0

(続く)

表 7 (続き) CYT2 シリーズの PWM_DT モード設定パラメータの一覧

パラメータ	説明	設定値
.enableCompare0Swap	CC0 とバッファリングされた CC0 の値を入れ替えてください。	false (0x0)
.enableCompare1Swap	CC1 とバッファリングされた CC1 の値を入れ替えてください。	false (0x0)
.interruptSources	割込みマスクビット	0d0 (ゼロ消去)
.invertPWMOut	PWM 出力 "line_out"と "line_compl_out"の動作	0d0
.invertPWMOutN		0d0
.killMode	Kill イベントでカウンタが停止	CY_TCPWM_PWM_STOP_ON_KILL (0x2)
.switchInputMode	Capture0 エッジモード	0x3 (NO_EDGE_DET)
.switchInput	capture0 の入力トリガ	0x0 (定数 0)
.reloadInputMode	リロードエッジモード	0x3 (NO_EDGE_DET)
.reloadInput	リロードの入力トリガ	0x0 (定数 0)
.startInputMode	開始エッジモード	0x3 (NO_EDGE_DET)
.startInput	開始の入力トリガ	0x0 (定数 0)
.kill0InputMode	停止エッジモード	0x3 (NO_EDGE_DET)
.kill0Input	停止の入力トリガ	0x0 (定数 0)
.kill1InputMode	Capture1 エッジモード	0x3 (NO_EDGE_DET)
.kill1Input	capture1 の入力トリガ	0x0 (定数 0)
.countInputMode	カウントエッジモード	0x3 (NO_EDGE_DET)
.countInput	カウントの入力トリガ	0x1 (定数 1)

Code Listing 23 に、設定部で PWM_DT モードを設定するサンプルプログラムを示します。

Code listing 23 CYT2 シリーズの設定部で PWM_DT モードを設定する例

```
#define TCPWMx_GRPx_CNTx_PWM_DT           TCPWM0_GRP1_CNT0    //Define Using Counter
#define PCLK_TCPWMx_CLOCKSx_PWM_DT        PCLK_TCPWM0_CLOCKS256    //Define Peripheral Clock
#define TCPWM_PERI_CLK_DIVIDER_NO_PWM_DT  0ul    //Define Peripheral Clock Divider



#define TCPWMx_PERIOD         10000ul    // 2,000,000 / 10000 = 200Hz    //Define PWM Period
#define TCPWMx_COMPARE0       5000ul     // 5000 / 10000 = 0.5 (duty)    //Define Compare Period
#define TCPWMx_DEADTIME       500ul      // Right side: 500*(1/2,000,000) = 250 us    //Define
Dead Time for line_out
#define TCPWMx_DEADTIME_COMPL 1000ul     // Left side :1000*(1/2,000,000) = 500 us    //Define
Dead Time for line_compl_out


cy_stc_tcpwm_pwm_config_t const MyPWM_config =    //Configure the PWM_DT parameters
{
  .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,
  .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);    //Calculation of division ratio
  :
  /* 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);    //(1)Configure the Peripheral Clock (See Code listing 3)

  /* Sets the 16-bit divider */
  Cy_SysClk_PeriphSetDivider(CY_SYSCLK_DIV_16_BIT, TCPWM_PERI_CLK_DIVIDER_NO_PWM_DT,
(divNum-1ul));    //(2)Configure the integer division of the 16-bit divider (See Code listing 4)


  /* Enable the divider */
  Cy_SysClk_PeriphEnableDivider(CY_SYSCLK_DIV_16_BIT, TCPWM_PERI_CLK_DIVIDER_NO_PWM_DT)    //
(3)Enable the 16-bit divider (See Code listing 5)

  /* Initialize TCPWMx_GRPx_CNTx_PWM_PR as PWM-DT Mode & Enable */
  Cy_Tcpwm_Pwm_Init(TCPWMx_GRPx_CNTx_PWM, &MyPWM_config);    //(4)Initialize the counter based
on above structure (See Code listing 21)
  Cy_Tcpwm_Pwm_Enable(TCPWMx_GRPx_CNTx_PWM);    //(5)Enable the counter (See Code listing 22)
  Cy_Tcpwm_TriggerStart(TCPWMx_GRPx_CNTx_PWM);    //(6)Start the counter (See Code listing 9)
  :
  for(;;);
}
```

3 トリガ マルチプレクサとの連携

TRAVEO™ T2G ファミリはトリガ マルチプレクサ機能を搭載します。TCPWM はトリガ マルチプレクサを使って SAR ADC や P-DMA などのモジュールや、TCPWM 自身に接続できます。各デバイスのトリガ マルチプレクサ接続についてはデバイスデータシートを参照してください。

図 18 は、トリガ マルチプレクサ、ADC、TCPWM などのペリフェラルを使用する場合の動作フローを示します。このフローチャートは、3 つの TCPWM 同時開始セクションと TCPWM 出力による AD 変換開始セクションで同じです。

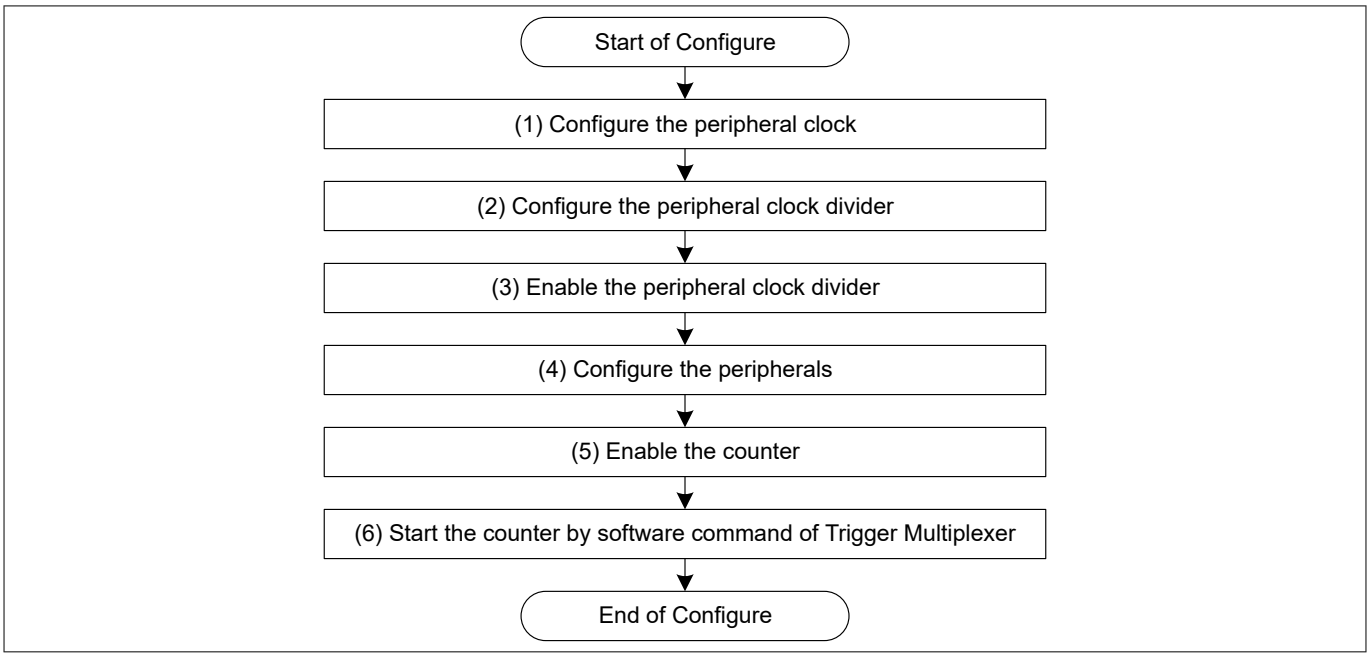

図 18 動作フローの例

1. TCPWM の周辺クロックを設定してください。
2. TCPWM 用の周辺クロック分周器を設定してください。
3. TCPWM の周辺クロックを有効にしてください。
4. ペリフェラル (トリガ マルチプレクサ、SAR ADC、TCPWM) を設定してください。

   注: TCPWM カウンタが有効になっている場合は、誤動作を防ぐために無効にしてください。

5. TCPWM カウンタを有効にしてください。
6. トリガ マルチプレクサのソフトウェア コマンドで TCPWM カウンタを開始してください。

3.1 3 つの TCPWM 同時開始

3.1.1 ユースケース

ここでは、ソフトウェアによる 3 つの TCPWM 同時開始の使用例について説明します。以下は、SDL を使用した TCPWM の設定例です。

- TCPWM 動作モード: PWM_DT モード
- 使用カウンタ: TCPWM0/Group1/Counter0, 1, 2
- カウンタの開始操作: ソフトウェアから開始

図 19 にトリガ マルチプレクサを使って 3 つの TCPWM のカウンタを同時に開始させ、PMW 信号を出力する例を示します。これらのカウンタはイベントの開始に同じ入力トリガを使っています。この入力トリガはトリガ マルチプレクサで設定します。

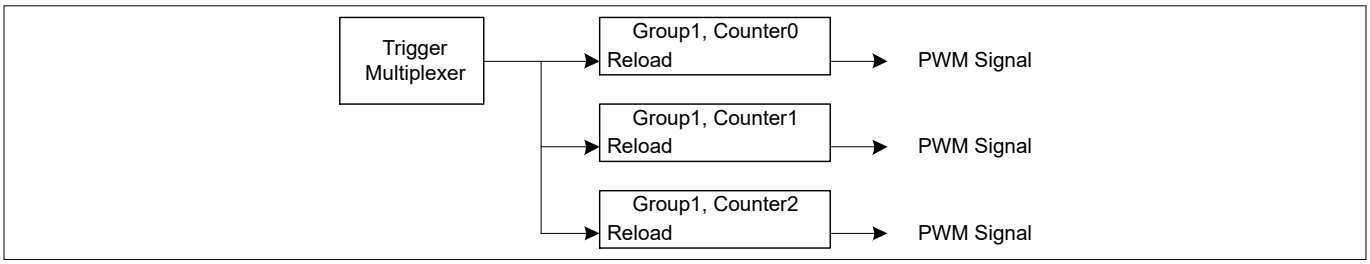

図 19　　CYT2B7 シリーズのリロード信号による 3 つの TCPWM 同時開始

この例の設定方法について、図 20 に、トリガ マルチプレクサと TCPWM の詳細なブロック図を示します。図のように、トリガ マルチプレクサの 27 出力は、各カウンタの 32 対 1 セレクタに接続されています。各カウンタは、TCPWM0_GRP1_CNT[a]_TR_IN_SEL0/1 レジスタ (a = カウンタ番号(0, 1, 2))によって信号を選択できます。

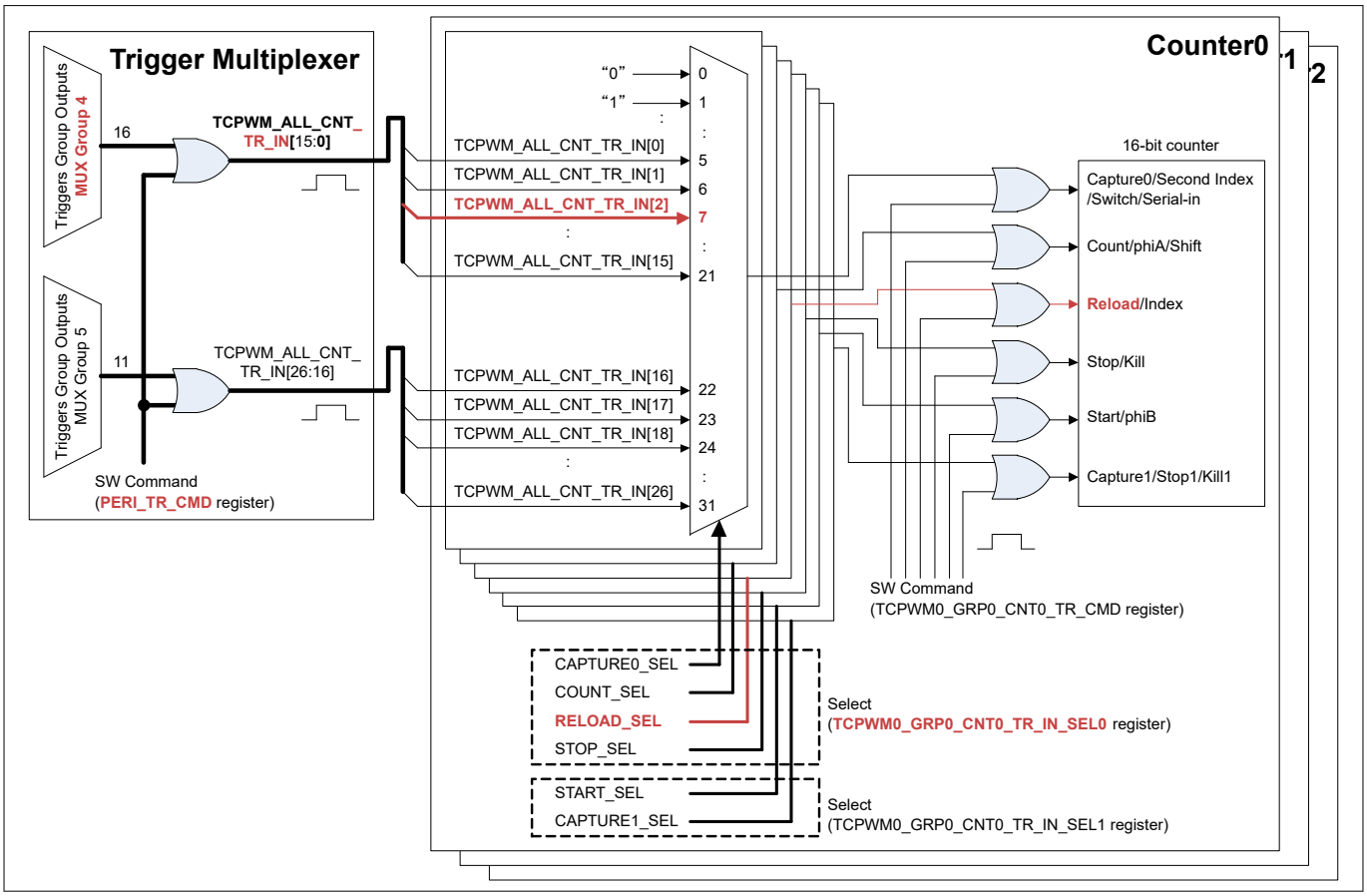

図 20　　CYT2B7 シリーズのトリガ マルチプレクサと TCPWM の接続詳細ブロック図

***注**:　　カウンタ制御信号の機能は、モードによって異なります。*

例えば、TCPWM0_GRP1_CNTx_TR_IN_SEL0 (x = 0, 1, 2) レジスタの RELOAD_SEL ビットが "7"に設定されている場合、トリガ マルチプレクサの TCPWM_ALL_CNT_TR_IN[2]がカウンタのリロードとして選択されます。TCPWM_ALL_CNT_TR_IN[2]は「グループトリガ」の MUX グループ 4 に属します。PERI_TRM_CMD レジスタを使用する場合、TCPWM_ALL_CNT_TR_IN[2]に HIGH/LOW/パルス信号を出力できます (この機能をソフトウェアコマンドと呼びます)。この出力が全カウンタのリロードに供給された場合、全カウンターが同時に開始します。

MUX GROUP の詳細については、デバイス データシートの"Triggers group outputs"の章を参照してください。

3.1.2 設定とサンプルコード

表 8 に、リロード信号による 3 つの TCPWM 同時開始用の SDL におけるパラメータ構成を示します。

表 8 CYT2 シリーズのリロード信号による 3 つの TCPWM 同時開始設定パラメーター覧

パラメータ	説明	設定値
TCPWM		
.pwmMode	カウンタ モード	CY_TCPWM_PWM_MODE_DEADTIME (0x5)
.clockPrescaler	選択されたカウンタ クロックのプリスケーリング	CY_TCPWM_PWM_PRESCALER_DIVBY_1 (0x0)
.debug_pause	デバッグモードでのカウンタの動作	false (0x0)
.countDirection	カウンタ方向	COUNT_UPDN2 (0x3)
.Cc0MatchMode	コンペアマッチ 0 イベントの効果を決定	CY_TCPWM_PWM_TR_CTRL2_SET (0x0)
.OverflowMode	カウンタのオーバーフローイベントの影響を決定	CY_TCPWM_PWM_TR_CTRL2_SET (0x0)
.UnderflowMode	カウンタのアンダーフローイベントの影響を決定	CY_TCPWM_PWM_TR_CTRL2_CLEAR (0x1)
.Cc1MatchMode	コンペアマッチ 1 イベントの効果を決定	CY_TCPWM_PWM_TR_CTRL2_CLEAR (0x1)
.deadTime	デッドタイムの決定	100
.deadTimeComp	デッドタイムの決定	100
.runMode	カウンタ動作モード	CY_TCPWM_PWM_CONTINUOUS (0x0)
.period	カウンタ値 (このフィールドは "n-1 "に設定してください)	2000
.period_buff		2000
.enablePeriodSwap	PERIOD とバッファリングされた PERIOD の値を入れ替えてください。	false (0x0)
.enableCompare0Swap	CC0 とバッファリングされた CC0 の値を入れ替えてください。	true (0x1)
.enableCompare1Swap	CC1 とバッファリングされた CC1 の値を入れ替えてください。	true (0x1)
.interruptSources	割込みマスクビット	0d0 (ゼロ消去)
.invertPWMOut	PWM 出力 "line_out"と "line_compl_out"の動作	0d0
.invertPWMOutN		0d0
.killMode	Kill イベントでカウンタが停止	CY_TCPWM_PWM_STOP_ON_KILL (0x2)
.switchInputMode	Capture0 エッジモード	0x3 (NO_EDGE_DET)
.switchInput	capture0 の入力トリガ	0x0 (定数 0)
.reloadInputMode	リロードエッジモード	0x3 (NO_EDGE_DET)
.reloadInput	リロードの入力トリガ	0x7 (TCPWM_ALL_CNT_TR_IN[2])
.startInputMode	開始エッジモード	0x3 (NO_EDGE_DET)

(続く)

表 8 (続き) CYT2 シリーズのリロード信号による 3 つの TCPWM 同時開始設定パラメーター覧

パラメータ	説明	設定値
.startInput	開始の入力トリガ	0x0 (定数 0)
.kill0InputMode	停止エッジモード	0x3 (NO_EDGE_DET)
.kill0Input	停止の入力トリガ	0x0 (定数 0)
.kill1InputMode	Capture1 エッジモード	0x3 (NO_EDGE_DET)
.kill1Input	capture1 の入力トリガ	0x0 (定数 0)
.countInputMode	カウントエッジモード	0x3 (NO_EDGE_DET)
.countInput	カウントの入力トリガ	0x1 (定数 1)
トリガ マルチプレクサ		
trigLine	トリガ グループ出力	TRIG_OUT_MUX_4_TCPWM_ALL_CNT_TR_IN2
trigType	出力トリガはレベルセンシティブまたはエッジセンシティブ	TRIGGER_TYPE_EDGE
outSel	入力トリガを指定	0x1

Code Listing 24 に、設定部で 3 つの TCPWM を同時に開始するサンプルプログラムを示します。

Code listing 24 CYT2 シリーズの設定部で 3 つの TCPWM 同時に開始する例

```
/* 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)
{
  :
  /* Clock Configuration for TCPWMs */
  //(1 to 3)Configure and select the clock for the counters (See Code Listing 3 to Code listing
5)
  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 */ //(4)Configure the counter
for U-phase (See Code listing 21)
  :
  Cy_Tcpwm_Pwm_Enable(TCPWM0_GRP1_CNT0);    //(5)Enable the counter for U-phase (See Code
listing 22)

  Cy_Tcpwm_Pwm_Init(TCPWM0_GRP1_CNT1, &MyPWM_config);  /* V-phase */    //(4)Configure the
counter for V-phase (See Code listing 21)
  :
  Cy_Tcpwm_Pwm_Enable(TCPWM0_GRP1_CNT1);    //(5)Enable the counter for V-phase (See Code
listing 22)

  Cy_Tcpwm_Pwm_Init(TCPWM0_GRP1_CNT2, &MyPWM_config);  /* W-phase */    //(4)Configure the
counter for W-phase (See Code listing 21)
  :
  Cy_Tcpwm_Pwm_Enable(TCPWM0_GRP1_CNT2);    //(5)Enable the counter for W-phase (See Code
listing 22)

  /* 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] */    //(6)Start the all
counters by software command (See Code listing 25)
  for(;;)
  {
    :
  }
}
```

Code Listing 25～Code Listing 27 に、ドライバ部でトリガ マルチプレクサを設定するサンプルプログラムを示します。

Code listing 25 CYT2 シリーズのドライバ部でトリガ マルチプレクサを設定する例 (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)    //Generate the software trigger for trigLine.
{
  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);    //See Code listing 26
    PERI->unTR_CMD.stcField.u5GROUP_SEL  = Cy_TrigMux_GetGroup(trigLine);    //See Code listing
27
    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;
}
```

Code listing 26 CYT2 シリーズのドライバ部でトリガ マルチプレクサを設定する例 (Cy_TrigMux_GetNo)

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

Code listing 27 CYT2 シリーズのドライバ部でトリガ マルチプレクサを設定する例 (Cy_TrigMux_GetGroup)

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

3.2 TCPWM 出力による AD 変換開始

3.2.1 ユースケース

ここでは、TCPWM トリガ出力を AD 変換の開始信号として使用する例を説明します。ADC の ch4 と ch5 はそれぞれ TCPWM の Group0_counter0 と Group0_counter1 で変換されます。以下は、SDL を使用した TCPWM とトリガ マルチプレクサの設定例です。

- TCPWM 動作モード: PWM モード
- 使用カウンタ: TCPWM0/Group1/Counter0, 1
- カウンタの開始操作: ソフトウェアから開始
- 使用する ADC チャネル: ch4, ch5
- 使用するトリガ: トリガは 1 対 1、MUX グループ 1
  - ADC Ch4: TCPWM0_16_TR_OUT1[4]
  - ADC Ch5: TCPWM0_16_TR_OUT1[5]

図 21 は、TCPWM 出力 (tr_out1) で AD 変換を開始する例を示します。2 つの"tr_out1"はトリガ マルチプレクサを介して SAR ADC の開始トリガとして接続できます。

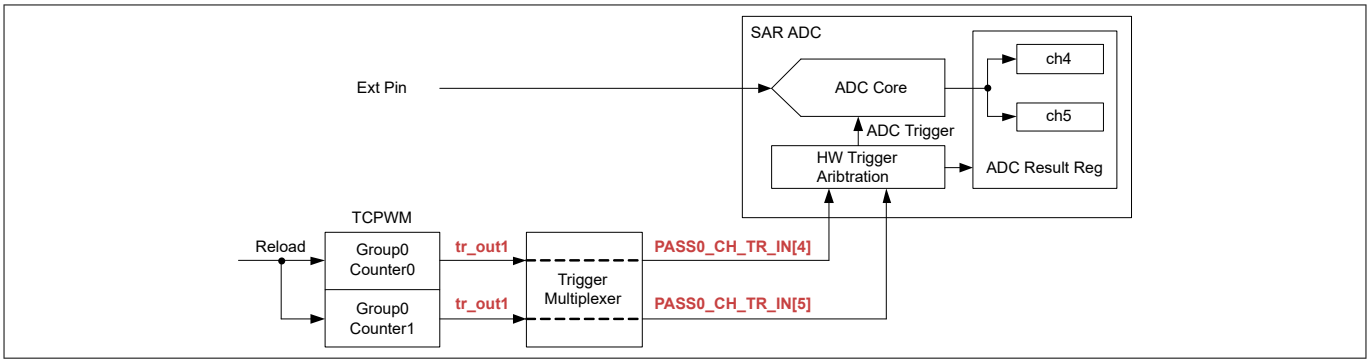

図 21 CYT2B7 シリーズの TCPWM 出力による AD 変換開始

また図 22 は、各カウンタの cc0 一致イベントによる AD 変換のタイミングチャートを示します。2 つの開始トリガによって実行された各 AD 変換の結果は各チャネルのレジスタに保存されます。

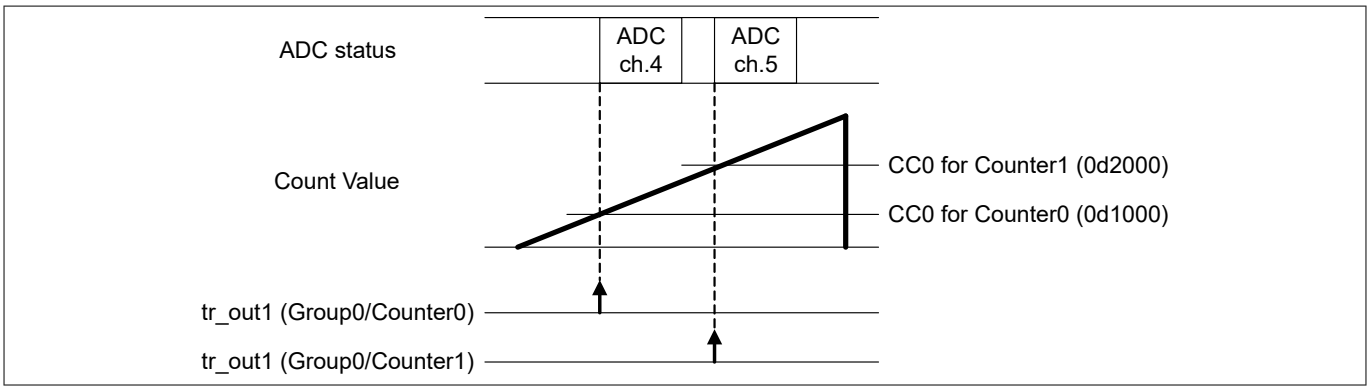

図 22 TCPWM 出力による AD 変換のスタートのタイミング チャート

cc0 一致イベントの詳細については、PWM モードと PWM デッドタイム（PWM_DT）モードを参照してください。

図 23 は、tr_out1 を SAR ADC のトリガに接続するトリガ マルチプレクサの詳細ブロック図です。

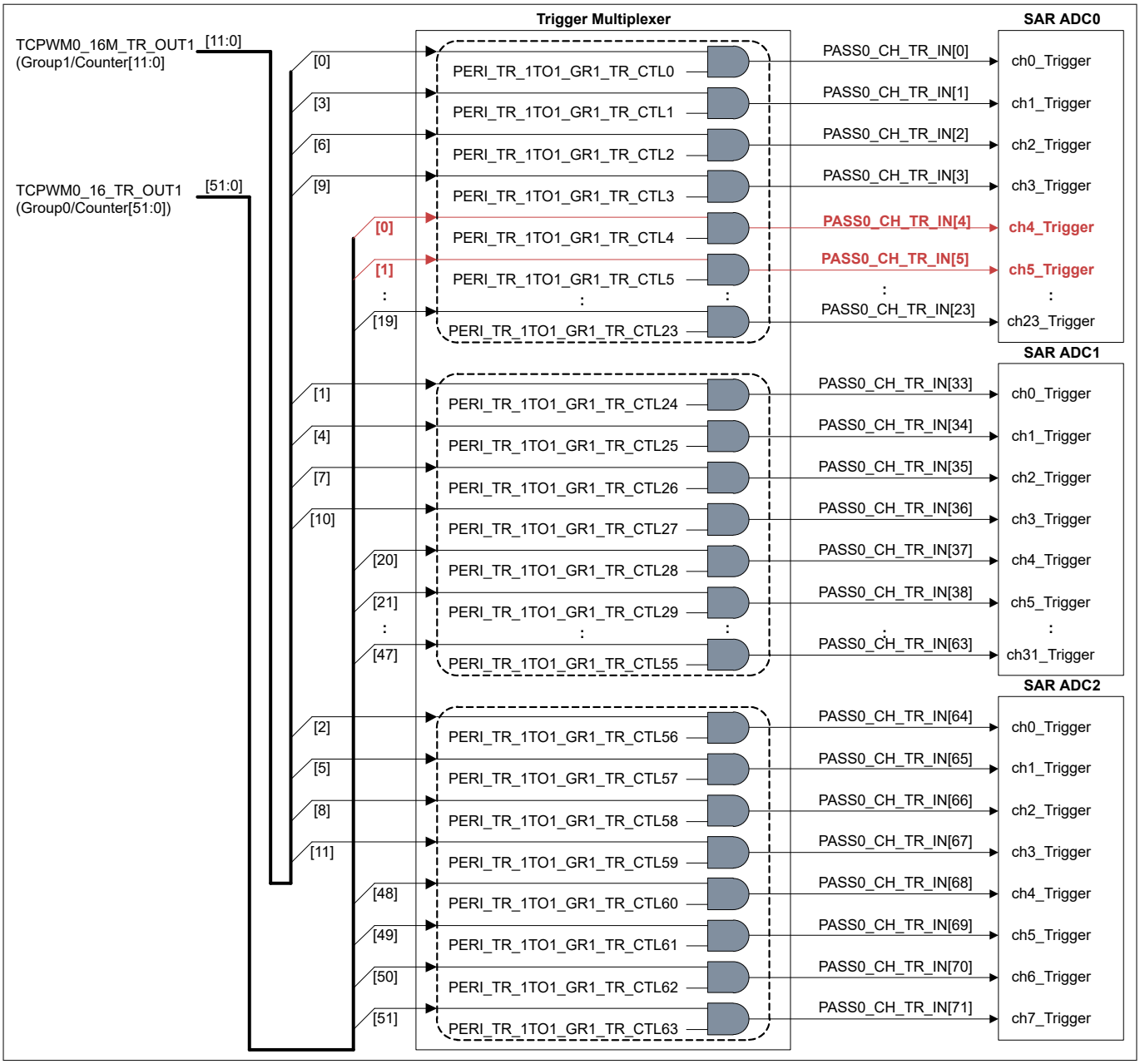

図 23　　CYT2B7 シリーズのトリガ マルチプレクサの詳細ブロック図

このように、Group0/Counter0 の tr_out1 は、トリガ マルチプレクサの"1 対 1 トリガ グループ"を介して SAR ADC0 の ch4 トリガに接続されます。同様に、Group0/Counter1 は SAR ADC0 の ch5_Trigger に接続されます。PASS0_CH_TR_IN4, 5 は「1 対 1 トリガ グループ」の MUX Group1 に属します。PASS0_CH_TR_IN4、5 は、それぞれ PERI_TR_1TO1_GR1_TR_CTL4、5 レジスタでアクティブにできます。

MUX GROUP の詳細については、デバイス・データシートの"Triggers one-to-one"の章を参照してください。

3.2.2　　設定とサンプルコード

表 9 に、PWM モードの SDL 設定部のパラメータを示します。

表 9 CYT2 シリーズの設定部で TCPWM 出力による AD 変換を開始する例

パラメータ	説明	設定値
TCPWM		
.period	カウンタ値 (このフィールドは "n-1 "に設定してください)	0d8000
.clockPrescaler	選択されたカウンタ クロックのプリスケーリング	CY_TCPWM_COUNTER_PRESCALER_DIVBY_1 (0x0)
.runMode	カウンタ動作モード	CY_TCPWM_PWM_CONTINUOUS (0x0)
.countDirection	カウンタ方向	CY_TCPWM_COUNTER_COUNT_UP (0x0)
.debug_pause	デバッグモードでのカウンタの動作	false (0x0)
.CompareOrCapture	カウンタ モード	CY_TCPWM_COUNTER_MODE_COMPARE (0x0)
.enableCompare0Swap	CC0 とバッファリングされた CC0 の値を入れ替えてください。	false (0x0)
.enableCompare1Swap	CC1 とバッファリングされた CC1 の値を入れ替えてください。	false (0x0)
.interruptSources	割込みマスク ビット	0x0 (ゼロ消去)
.capture0InputMode	Capture0 エッジモード	0x3 (NO_EDGE_DET)
.capture0Input	capture0 の入力トリガ	0x0 (定数 0)
.reloadInputMode	リロードエッジモード	0x3 (NO_EDGE_DET)
.reloadInput	リロードの入力トリガ	0x7 (TCPWM_ALL_CNT_TR_IN[2])
.startInputMode	開始エッジモード	0x3 (NO_EDGE_DET)
.startInput	開始の入力トリガ	0x0 (定数 0)
.stopInputMode	停止エッジモード	0x3 (NO_EDGE_DET)
.stopInput	停止の入力トリガ	0x0 (定数 0)
.capture1InputMode	Capture1 エッジモード	0x3 (NO_EDGE_DET)
.capture1Input	capture1 の入力トリガ	0x0 (定数 0)
.countInputMode	カウントエッジモード	0x3 (NO_EDGE_DET)
.countInput	カウントの入力トリガ	0x1 (定数 1)
.trigger1	出力トリガ 0 を生成する内部イベント	CY_TCPWM_COUNTER_OVERFLOW (0x0)
トリガ マルチプレクサ (Cy_TrigMux_Connect1To1)		
trig	1 対 1 トリガの数	TRIG_IN_1TO1_1_TCPWM_TO_PASS_CH_TR4, 5
trigType	出力トリガはレベルセンシティブまたはエッジセンシティブ	TRIGGER_TYPE_PASS_TR_SAR_CH_IN__EDGE
トリガ マルチプレクサ (Cy_TrigMux_SwTrigger)		
trigLine	トリガ グループ出力の数	TRIG_OUT_MUX_4_TCPWM_ALL_CNT_TR_IN2

(続く)

表 9 (続き) CYT2 シリーズの設定部で TCPWM 出力による AD 変換を開始する例

パラメータ	説明	設定値
trigType	出力トリガはレベルセンシティブまたはエッジセンシティブ	TRIGGER_TYPE_EDGE
outSel	入力トリガを指定	0x1

Code Listing 28 に、設定部で TCPWM 出力による AD 変換を開始するサンプルプログラムを示します。

Code listing 28 CYT2 シリーズの設定部で TCPWM 出力による AD 変換を開始する例

```
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 =    //Configure the counter parameters
{
  .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 */
  //(1 to 3)Configure and select the clock for the counter (See Code listing 3 to Code listing
5)
  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]) */
  //(4)Configure the Trigger Multiplexer (See Code listing 30)
  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);    //(4)Configure the counter for ADC
Ch4 (See Code listing 6)    //(4)Set the compare value for ADC Ch4 (See Code listing 29)
  Cy_Tcpwm_Counter_Enable(TCPWM0_GRP0_CNT0);    //(5)Enable the counter for ADC Ch4 (See Code
listing 8)



  Cy_Tcpwm_Counter_Init(TCPWM0_GRP0_CNT1, &MyCounter_config);    // TCPWM_CNT1 ADC    //
(4)Configure the counter for ADC Ch5 (See Code listing 6)
  :
  Cy_Tcpwm_Counter_SetCompare0(TCPWM0_GRP0_CNT1, 2000ul);    //(4)Set the compare value for ADC
Ch5 (See Code listing 29)
  Cy_Tcpwm_Counter_Enable(TCPWM0_GRP0_CNT1);    //(5)Enable the counter for ADC Ch5 (See Code
listing 8)



  /* 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] */    //(6)Start the all
counters by software command (See Code listing 25)

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

Code Listing 29 に、ドライバ部でトリガ マルチプレクサを設定するサンプルプログラムを示します。

Code listing 29 CYT2 シリーズのドライバ部で出力による AD 変換を開始する例 (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)    //Sets the compare0 value for counter
{
  ptscTCPWM->unCC0.u32Register = compare0;
}
```

Code Listing 30～Code Listing 31 に、ドライバ部でトリガ マルチプレクサを設定するサンプルプログラムを示します。

Code listing 30 CYT2 シリーズのドライバ部で出力による AD 変換を開始する例 (Cy_TrigMux_Connect1To1)

```
/*******************************************************************************
* Function Name: Cy_TrigMux_Connect1To1
*******************************************************************************/
cy_en_trigmux_status_t Cy_TrigMux_Connect1To1(uint32_t trig, uint32_t invert,    //Sets the
compare0 value for counter
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)    //See Code listing 31
  {
    /* 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);    //See Code listing 27
  uint8_t trigNo    = Cy_TrigMux_GetNo(trig);    //See Code listing 26

  /* 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;

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

Code listing 31 CYT2 シリーズのドライバ部で出力による AD 変換を開始する例 (Cy_TrigMux_IsOneToOne)

```
/*******************************************************************************
* Function Name: Cy_TrigMux_IsOneToOne
*******************************************************************************/
static bool Cy_TrigMux_IsOneToOne(uint32_t trig)    //checks whether trigger parameter is for
One-To-One trigger or not
{
  // 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;
  }
}
```

用語集

用語	説明
SAR ADC	アナログからデジタルへの変換器。詳細については、Architecture reference manual の SAR ADC 章を参照してください。
P-DMA	ペリフェラル ダイレクトメモリアクセス
周辺クロック分周器	ペリフェラルクロックディバイダは TCPWM のカウンタのような各ペリフェラル機能へクロックを分配します。
トリガ マルチプレクサ	トリガ マルチプレクサはソースペリフェラルから使用先へトリガを接続します。詳細については、Architecture reference manual の Trigger Multiplexer 章を参照してください。

関連ドキュメント

以下は TRAVEO™ T2G ファミリのデータシートおよびテクニカルリファレンスマニュアルです。これらドキュメントの入手については Technical Support に連絡してください。

[1] デバイスデータシート

- CYT2B6 datasheet 32-bit Arm® Cortex®-M4F microcontroller TRAVEO™ T2G family
- CYT2B7 datasheet 32-bit Arm® Cortex®-M4F microcontroller TRAVEO™ T2G family
- CYT2B9 datasheet 32-bit Arm® Cortex®-M4F microcontroller TRAVEO™ T2G family
- CYT2BL datasheet 32-bit Arm® Cortex®-M4F microcontroller TRAVEO™ T2G family
- CYT3BB/4BB datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family
- CYT4BF datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family
- CYT6BJ datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family (Doc No. 002-33466)
- CYT3DL datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family
- CYT4DN datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family
- CYT4EN datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family (Doc No. 002-30842)
- CYT2CL datasheet 32-bit Arm® Cortex®-M4F microcontroller TRAVEO™ T2G family

[2] Body Controller Entry ファミリ

- TRAVEO™ T2G automotive body controller entry family architecture technical reference manual (TRM)
- TRAVEO™ T2G automotive body controller entry registers technical reference manual (TRM) for CYT2B7
- TRAVEO™ T2G automotive body controller entry registers technical reference manual (TRM) for CYT2B9
- TRAVEO™ T2G automotive body controller entry registers technical reference manual (TRM) for CYT2BL (Doc No. 002-29852)

[3] Body Controller high ファミリ

- TRAVEO™ T2G automotive body controller high family architecture technical reference manual (TRM)
- TRAVEO™ T2G automotive body controller high registers technical reference manual (TRM) for CYT3BB/4BB
- TRAVEO™ T2G automotive body controller high registers technical reference manual (TRM) for CYT4BF
- TRAVEO™ T2G automotive body controller high registers technical reference manual (TRM) for CYT6BJ (Doc No. 002-36068)

[4] Cluster 2D ファミリ

- TRAVEO™ T2G automotive cluster 2D architecture technical reference manual (TRM)
- TRAVEO™ T2G automotive cluster 2D registers technical reference manual (TRM) for CYT3DL
- TRAVEO™ T2G automotive cluster 2D registers technical reference manual (TRM) for CYT4DN
- TRAVEO™ T2G automotive cluster 2D registers technical reference manual (TRM) for CYT4EN (Doc No. 002-35181)

[5] Cluster Entry ファミリ

- TRAVEO™ T2G automotive cluster entry family architecture technical reference manual (TRM)
- TRAVEO™ T2G automotive cluster entry registers technical reference manual (TRM) for CYT2CL

その他の関連資料

インフィニオンは、さまざまな周辺機器にアクセスするためのサンプルソフトウェアとして、スタートアップコードを含む Sample Driver Library (SDL) を提供しています。SDL は、公式の AUTOSAR 製品でカバーされないドライバの顧客へのリファレンスとしても機能します。SDL は自動車規格に適合していないため、製造目的では使用できません。このアプリケーションノートのプログラムコードは SDL の一部です。SDL を入手するには、テクニカルサポートに連絡してください。

改訂履歴

版数	日付	変更内容
**	2019-07-12	これは英語版 002-20224 Rev. **を翻訳した日本語版 Rev. **です。英語版の改訂内容: New Application Note.
*A	2019-12-06	これは英語版 002-20224 Rev. *A を翻訳した日本語版 Rev. *A です。英語版の改訂内容: Added CYT4D Series
*B	2020-06-22	これは英語版 002-20224 Rev. *B を翻訳した日本語版 Rev. *B です。英語版の改訂内容: Changed target parts number (CYT2/CYT4 series). Added target parts number (CYT3 series).
英語版(*C)	-	この版は英語版のみです。英語版の改訂内容: Updated code examples using SDL MOVED TO INFINEON TEMPLATE.
英語版(*D)	-	この版は英語版のみです。英語版の改訂内容: Template update; no content changes.
*C	2024-12-09	これは英語版 002-20224 Rev. *E を翻訳した日本語版 Rev. *C です。英語版の改訂内容: Updated to add CYT6 series.

Trademarks

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

Edition 2024-12-09

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-hku1681442821171

重要事項

本手引書に記載された情報は、本製品の使用に関する手引きとして提供されるものであり、いかなる場合も、本製品における特定の機能性能や品質について保証するものではありません。本製品の使用の前に、当該手引書の受領者は実際の使用環境の下であらゆる本製品の機能及びその他本手引書に記された一切の技術的情報について確認する義務が有ります。インフィニオンテクノロジーズはここに当該手引書内で記される情報につき、第三者の知的所有権の不侵害の保証を含むがこれに限らず、あらゆる種類の一切の保証および責任を否定いたします。

本文書に含まれるデータは、技術的訓練を受けた従業員のみを対象としています。本製品の対象用途への適合性、およびこれら用途に関連して本文書に記載された製品情報の完全性についての評価は、お客様の技術部門の責任にて実施してください。

警告事項

技術的要件に伴い、製品には危険物質が含まれる可能性があります。当該種別の詳細については、インフィニオンの最寄りの営業所までお問い合わせください。

インフィニオンの正式代表者が署名した書面を通じ、インフィニオンによる明示の承認が存在する場合を除き、インフィニオンの製品は、当該製品の障害またはその使用に関する一切の結果が、合理的に人的傷害を招く恐れのある一切の用途に使用することはできないこと予めご了承ください。