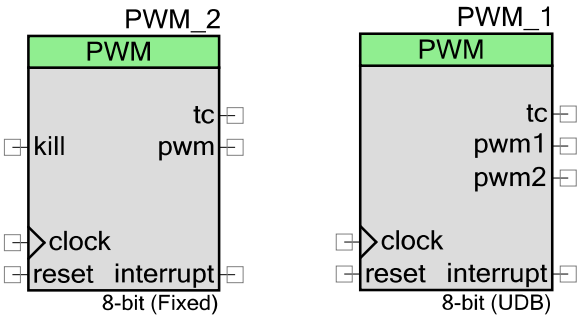
パルス幅変調器 (PWM)

2.10

特長



- 8 または 16 ビットの分解能
- 複数パルス幅出力モード
- 構成可能なトリガ
- 構成可能なキャプチャ
- 構成可能なハードウェア/ソフトウェア イネーブル
- 構成可能なデッドバンド
- 複数の構成可能なキル モード
- カスタマイズされた構成ツール

機能の概要

PWM コンポーネントは、コンペア出力を提供します。このコンペア出力はハードウェアで単一または連続したタイミング信号を生成したり、制御信号を生成します。この PWM は、最低限の CPU 介入によって、複雑なリアルタイムのイベントを精度よく発生させる簡単な方法を提供するよう設計されています。PWM 機能は他のアナログとデジタルコンポーネントと組み合わせることで、カスタムペリフェラルを作ることができます。

PWM は、最高 2 つのパルス左端位置、または右端位置設定 PWM 出力、あるいは 1 つのパルス中央位置設定、またはデュアルエッジの PWM 出力を生成します。実行中にデューティサイクルの変更により起こるグリッジを避けるために PWM 出力はダブルバッファされています。左端設定された PWM は、汎用 PWM として最も頻繁に使用されます。右端設定された PWM は主に、左端設定の PMW の反対側のアライメントが必要な特殊なケースに使用されます。中央位置設定の PWM は、フェーズアライメントを保つための AC モータ制御に最も頻繁に使用されます。デュアルエッジの PWM は、フェーズアライメントの調整が必要な場合に、パワー変換用に最適化されます。

オプションのデッドバンドは、移行と移行の間の出力がともにローとなる調整可能なデッドタイムを伴う補助出力を提供します。コンプリメンタリ出力とデッドタイムは、ハーフ ブリッジ構成のパワードライブデバイスにおいて、貫通電流の回避と、その結果による損傷を避けるためによく使用されます。キル インプットが有効なとき、デッドバンド出力を即座にディスエーブルにするキル出力も利用可能です。複数の使い方シナリオをサポートするために、3 つのキル モードが利用できます。

PWM の柔軟性を増加するために、2 つのハードウェア ディザー モードがあります。最初のディザー モードは、リソースやクロック周波数が PWM カウンタに標準的実装を行なっていなければ、実効的分解能を 2 ビット分高めます。第 2 のディザー モードは、デジタル入力をサイクルベースで 2 つの PWM 出力の選択に使用します。このモードは、とくにパワーコンバータでの高速応答の提供で使用されます。

トリガとリセット入力により、PWM は内部あるいは外部のハードウェアと同期できます。オプションのトリガ入力は、立ち上がりエッジが PWM で開始されるように設定することができます。リセット入力の立ち上がりエッジでは、最終カウントに達したような状態となり、PWM カウンタのカウントをリセットします。イネーブル入力は、ハードウェア信号に基づいて PWM 動作をゲートするハードウェア イネーブルを提供します。

PWM がターミナル カウントに達した場合、またはコンペア出力が HIGH になった場合、またはその組み合わせにより、割り込みを生成するようにプログラムできます。

PWM 使用の実際

PWM の最も一般的な使用法は、調整可能なデューティサイクルを持つ定期的な波形を生成することです。また PWM は、パワー制御、モーター制御、スイッチングレギュレータ、照明制御に対して最適な機能を提供します。PWM は、クロック出力として最終カウントまたは PWM 出力を分割して使用することにより、クロック分周器として使用されます。その出力でクロックを駆動してクロック入力に送信されます。

PWM、タイマ、カウンタは、多くの機能を共有していますが、それぞれに特有の機能もあります。カウンタ コンポーネントは、イベント数のカウントが必要な場合に使用することが推奨されますが、立ち上がりエッジのキャプチャ入力とコンペア出力も提供します。タイマ コンポーネントは、イベント長の時間測定に焦点を当てた状況で使用されることが推奨されています。複数の立ち上がりエッジや立ち下がりエッジの間隔、または複数のキャプチャイベントを測定します。

入出力の接続

このセクションでは、カウンタの様々な入力および出力接続について説明します。一部の I/O は、設定条件が与えられていない場合などには、シンボルの下に隠されて表示されない場合があります。

注 特記がない限り、すべての信号はアクティブ HIGH です。

入力	通常非表示	機能
クロック	N	クロック入力はカウントする信号を定義します。カウンタはクロックの各立ち上がりエッジで加算または減算されます。

入力	通常非表示	機能
リセット	N	ピリオド カウンタを **Period** にリセットし、通常動作を続けます。 **注:** リセット中、pwm、pwm1、pwm2 出力はディスエーブルです(0 駆動)。 固定機能を実装するために、pwm 出力はリセット中「1」駆動となります。 この回路図の修正は 機能説明 セクションの固定機能ブロックでのリセットを参照してください。 PSoC 3 ES2 シリコンでは、固定機能 PWM の最終カウント ピンは、リセットで HIGH に保たれます。pwm 出力用に提供されている回路図修正も、最終カウント用に使用できます。PSoC 3 以降のシリコンでは、固定機能PWMの最終カウント ピンは、リセットで LOW に保たれます。
enable (イネーブル)	Y	イネーブル入力は、ソフトウェアイネーブルおよびトリガ入力(トリガ入力がイネーブルな場合)と連動して動作し、ピリオド カウンタをイネーブルにします。**Enable Mode** パラメーターが **Software Only** に設定されている場合、イネーブル入力は表示されません。「Fixed Function(固定機能)」PWM実装が選択されている場合、この入力は使用できません。
Kill (キル)	Y	キル入力は PWM 出力をディスエーブルにします。キル モードには幾つかありますが、そのすべての出力信号を最終的にキルするかどうかは、この入力に依存します。デッドバンドが実装されている場合、デッドバンド出力(ph1とph2)のみがディスエーブルとなり、pwm、pwm1、pwm2 出力はディスエーブルとなりません。**Kill Mode** パラメータが **Disabled** に設定されている場合、キル入力は表示されません。固定機能 PWM 実装とデッドバンドをイネーブルに選択している場合、キルはデッドバンド出力だけをキルします。PSoC 5 では、デッドバンドがディスエーブルになっている場合、コンパレータ出力をキルしません。PSoC 3 では、デッドバンドがディスエーブルになっている場合、コンパレータ出力をキルします。
cmp_sel	Y	cmp_sel 入力は、pwm 端子への最終入力として、pwm1 または pwm2 を選択します。設定ツール波形ビューワに示すように、入力が「0」(low)の場合、pwm 出力は pwm 1で、入力が「1」(high)の場合、pwm 出力は pwm 2 です。**PWM Mode** パラメーターが **Hardware Select** に設定されている場合、cmp_sel 入力は表示されています。
capture (キャプチャ)	Y	キャプチャ入力は、ピリオド カウンタをリード FIFO に強制的に入れます。**Capture Mode** パラメーターでは、この入力ように幾つかのモードが定義されています。**Capture Mode** パラメーターが**None** に設定されている場合、キャプチャ入力は表示されません。「Fixed Function(固定機能)」PWM 実装が選択されている場合、キャプチャ入力は必ず立ち下がりエッジセンシティブです。
trigger (トリガ)	Y	トリガ入力は、PWM の動作をイネーブルにします。この入力の機能は **Trigger Mode** パラメータおよび **Run Mode** パラメータによって定義されています。「PWM_開始」API コマンド後、PWM はイネーブルになりますが、カウンタはトリガ条件が発生するまで減算されません。PWM の UDB への実装では、トリガ入力は入力クロックとともに登録されるため、カウンタはトリガ入力がアサートされた1クロック後に開始します。トリガ条件は **Trigger Mode** パラメータで設定します。パラメーターが **None** に設定されている場合、トリガ入力は表示されません。

出力	通常非表示	機能
tc	N	ピリオド カウンタがゼロと等しくなった場合、最終カウント出力は「1」になります。通常動作では、この出力はカウンタが周期でリロードされているシングルサイクルで「1」となります。ピリオド カウンタがゼロと等しくなって PWM が停止した場合、この信号はピリオド カウンタがゼロ以外になるまで HIGH の状態を続けます。出力は、コンポーネントのブロッククロック入力と同期されます。

出力	通常非表示	機能
割り込み	Y	割り込み出力は、割り込みソースとなれる信号グループの論理ORです。この信号は、イネーブルな割り込みソースが真である間は、HIGH に変わります。割り込み入力は、ソフトウェアが状況レジスタを読みだすまで、アサートの状態を続けます。次の割り込みを受け取るためには、PWM_ReadStatusRegister() APIを使用してステータス レジスタを読むことにより、割り込みがクリアされなければなりません。**Use Interrupt** パラメーターが設定されている場合、割り込み出力が表示されます。これにより必要に応じて、状態レジスタを削除してリソースを最適化することができます。
pwm/pwm1	Y	pwm または pwm1 出力は、最初の、または唯一の PWM 出力です。この信号は**Configure** ダイアログの波形で示されるように、**PWM Mode**、コンペア モード、比較値で定義されます。インスタンスが **One Output**、**Dual Edge**、 **Hardware Select**、 **Center Align**または **Dither** PWM モードのいずれかで設定されている場合、出力「 pwm」は表示されています。これ以外の場合は、出力「pwm1」がもう一つのパルス幅「pwm2」とともに表示されています。出力は、コンポーネントのブロッククロック入力と同期されます。
pwm2	Y	pwm2 出力は第 2 の PWM 出力です。pwm2 出力は **PWM Mode** が **Two Outputs** に設定されている場合に表示されています。出力は、コンポーネントのブロッククロック入力と同期されます。
ph1/ph2	Y	ph1 と ph2 出力は、PWM のデッドバンドフェーズ出力です。pwm のみが表示されるすべてのモードでは、フェーズ済みの pwm 出力があり、これも表示されています。**Two Outputs** モードでは、信号は pwm1 信号のみのフェーズ出力です。デッドバンドが 2～4 または 2～256 モードでイネーブルな場合、これらの出力は両方とも表示されており、デッドバンドがディスエーブルの場合は表示されません。出力は、コンポーネントのブロッククロック入力と同期されます。

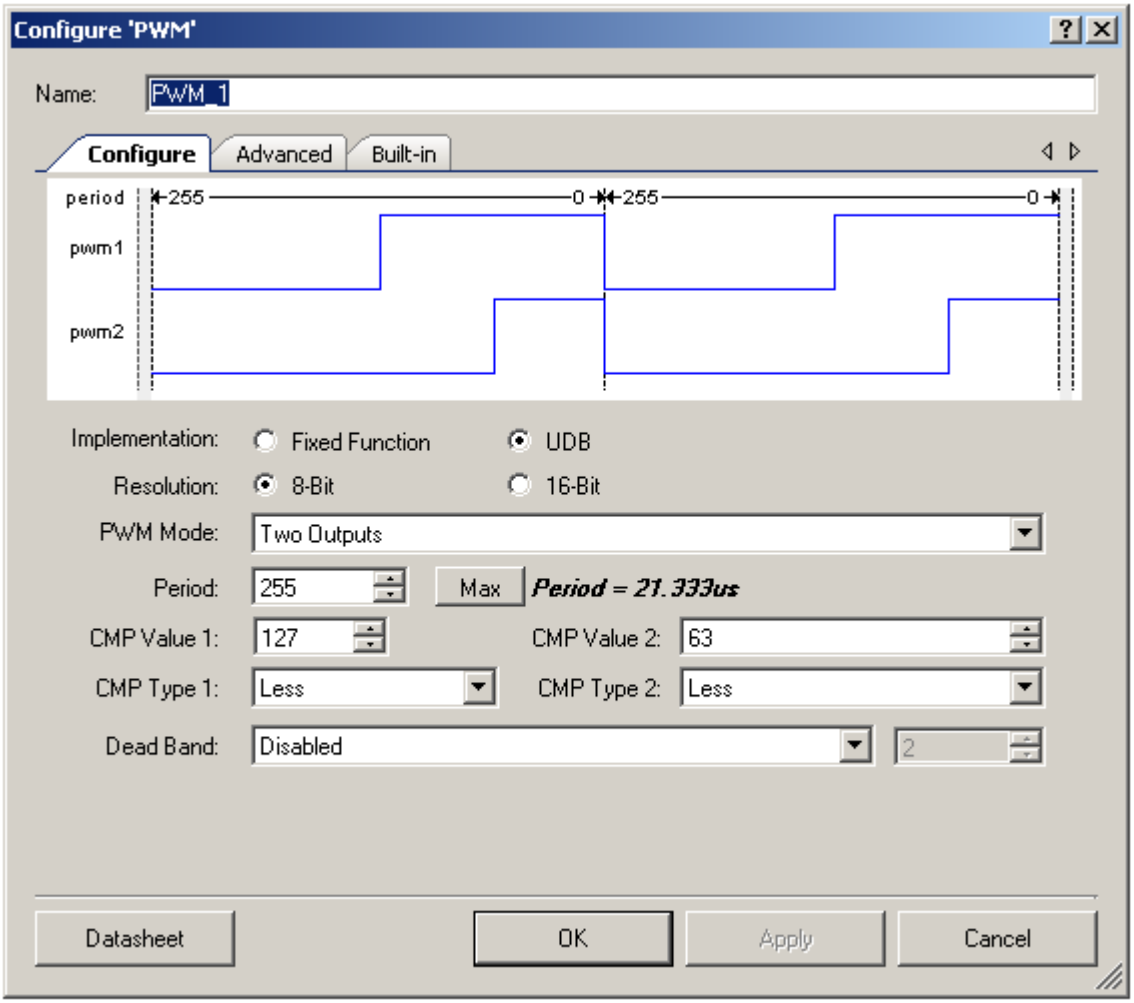
コンポーネント パラメータ

PMW コンポーネントをデザイン上にドラッグし、ダブルクリックして **Configure** ダイアログを開きます。**Configure PWM** ダイアログには、**[Configure (設定)]** と **Advanced** の 2 つのタブがあります。

ハードウェア vs. ソフトウェア構成オプション

ハードウェア構成オプションは、ハードウェア中でコンポーネントが合成、配置される方法を変えます。これらのオプションに変更する場合、ハードウェアをリビルドしなければなりません。ソフトウェア構成オプションは、合成や配置に影響を及ぼしません。ビルド以前にこれらのパラメータを設定すると、初期値を設定することになり、初期値は提供されている API を用いていつでも修正できます。次のセクションで説明するパラメータのほとんどは、ハードウェア オプションです。ソフトウェア オプションについても同様に説明されます。

[Configure (設定)] タブ



Implementation (実装)

このパラメーターを利用すると、**Fixed Function** および PWM の **UDB** 実装のいずれかを選択できます。このパラメーターが **Fixed Function** に設定された場合、PWM は固定機能ブロックに関連するリミットに基づいて実装されます。

分解能

Resolution パラメーターは、ピリオド カウンタのビット幅の分解能を定義します。

分解能	最大周期カウント値
8 (デフォルト)	255
16	65,535

注: **PWM Mode** が **Center Align** に設定されている場合、ピリオド値までカウントアップし、次にゼロまでカウントダウンします。これにより PWM の周期が倍になります。このモードでは、8 ビット PWM の制限は 254 サイクル(x2 = 508 サイクル)で、16 ビット PWM の制限は 65,534 サイクル(x2 = 131,068 サイクル)です。

PWM モード

PWM Mode パラメータは、PWM 全体の機能を定義します。**Implementation** が **Fixed Function** に設定されている場合、PWM 機能はディスエーブルになります。

コンフィギュレーション ツールの波形に示すように、このパラメータは、pwm、pwm1、pwm2 のシンボルおよび機能性の目に見えるピンに大きな影響を与えます。オプション:

- **One Output** – 単一の PWM 出力。このモードでは、「pwm」出力は表示されています。
- **Two Output** – 個別設定が可能な 2 つの PWM 出力。このモードでは、「pwm1」と「pwm2」が表示されます。
- **Dual Edge** – pwm1 信号と pwm2 信号を AND して生成される単一のデュアルエッジ 出力です。このモードでは、「pwm」出力は表示されています。
- **Center Align** – 比較値に基づいた中央位置設定されたパルス幅を作成しながら、カウンタをピリオド値までカウントアップし、次にゼロまでカウントダウンすることにより作成する単一中央位置設定出力。このモードでは、「pwm」出力は表示されています。
- **Dither** – pwm ハードウェア実装に含まれるハードウェア状態マシンによる 2 つの内部 pwm 信号 (pwm1 および pwm2)から選択する単一出力。ユーザは、出力パルス幅を 0.00、0.25、0.50 、0.75 ビットのいずれかでインクリメントできます。ハードウェアはこれを実行するために 2 つの pwm 信号間の選択を制御します。この場合、比較値は、比較と比較 +1 に設定されます。このモードでは、「pwm」出力は表示されています。
- **Hardware Select** – ハードウェアの入力ピン cmp_sel による 2 つの内部 pwm 信号から選択する単一の出力。cmp_sel が LOW の場合、pwm1 信号が pwm 出力ピンに出力され、cmp_sel が HIGH の場合、pwm2 信号が pwm 出力ピンに出力されます。このモードでは、「pwm」出力は表示されています。

Period (ソフトウェア)

Period パラメータはカウンタの開始初期値を定義し、最終カウントに達したとき、PWM モードがピリオド カウンタのリロードを可能にします。

PWM は、**Period** 値からゼロまでカウントダウンする際のカウンタとして実施されます。比較値は 1 より大きい数でなければならず、PWM の分解能では高い領域に限定されます。8 ビットの PWM では、最大ピリオド値は 255 です。それ以外では、最大ピリオド値は 65535 です。PWM モードが「Center Aligned (中央位置設定)」に設定されている場合、PWM がゼロからピリオド値までカウントアップし、その後カウントダウンしてゼロに戻り

ます。この特殊機能のために、Center Aligned (中央位置設定)モードのピリオド値は、他のすべてのモードの 2 倍です。このピリオド値は、いつでも PWM_WritePeriod(period) API の呼び出しによって変更できます。このパラメータは、設定中に書かれた初期値のみを保持します。

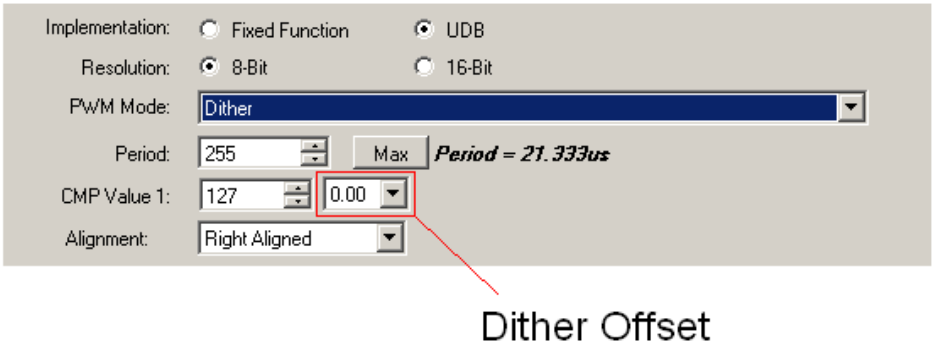
CMP 値 1 / CMP 値 2 (ソフトウェア)

比較値は、ハードウェアの **Compare Type** オプションとともに、比較出力機能を定義します。

比較値は 1 より大きい数でなければならず、PWM の分解能では高い領域に限定されます。8 ビットの PWM では、比較値は最大 255 です。それ以外では、比較値は最大 65535 です。比較値はピリオドによっても限定されます。ピリオドが減少すると、最大比較値は非使用コンペア出力を防止するために、**Period** – 1 に設定されます。比較値はいつでも、PWM_WriteCompare1() と PWM_WriteCompare2() API の呼び出しによって変更できます。これらのパラメータは、設定中に書かれた初期値のみを保持します。

Dither Offset (ディザーオフセット)

PWM が **Dither** PWM モードに設定されている場合、 **Dither Offset** パラメーターは pwm 出力の機能を設定します。



ディザーは、実装内部ステートマシンにより、最終「pwm」出力として、pwm1 と pwm2 出力のどちらか 1 つを選択します。pwm1 と pwm2 出力は、お互いにワン オフ ピリオド値に設定されます。つまり比較値のときは pwm1 が真で、比較値＋1 では pwm2 が真です。オプション:

- **DO00 –** ディザーなし。出力はいつでも pwm1 です。
- **DO25 –** 0.25 ディザー。ピリオド カウンタが 3～4 の場合、出力は pwm1 です。ピリオド カウンタが 1 の場合、出力は pwm2 です。
- **DO50 –** 0.50 ディザー。4 つあるピリオド カウンタのうち 2 つの場合、出力は pwm1 です。ピリオド カウンタが 4 つの場合、出力は pwm2 です。
- **DO75 –** 0.75 ディザー。4 つあるピリオド カウンタのうち 1 つの場合、出力は pwm1 です。4 つあるピリオド カウンタのうち 3 つの場合、出力は pwm2 です。

Alignment(アライメント)

Alignment パラメータは **PWM Mode** が **Dither** に設定されている場合、利用可能です。オプション:

- **Right Aligned(右端設定)**
- **Left Aligned(左端設定)**

CMP タイプ 1 / CMP タイプ 2 (ソフトウェア)

比較値パラメータは、PWM 出力を構成する 2 つのピリオド カウンタ比較を定義します。これらの実装は PWM モードによって異なるため、通常は設定ツールによって制御されます。2 つのコンペア モード パラメータのそれぞれは、次の計数タイプのうち 1 つに独立して設定できます。オプション:

- **Less** ― ピリオド カウンタが対応する比較値より小さい場合、コンペア出力は真です。
- **Less or Equal** ― ピリオド カウンタが対応する比較値以下の場合、コンペア出力は真です。
- **Greater** – ピリオド カウンタが対応する比較値より大きい場合、コンペア出力は真です。
- **Greater or Equal** – ピリオド カウンタが対応する比較値を超える場合、コンペア出力は真です。
- **Equal** – ピリオド カウンタが対応する比較値と同じ場合、コンペア出力は真です。
- **Firmware Control** – **Firmware Control** での実装により、動作時にコンペア モードが設定でき、より多くのリソース利用モデルを提供をします。コンペア モードはいつでも、PWM_WriteCompare1() と PWM_WriteCompare2() API の呼び出しによって変更できます。これらのパラメータは、設定中に書かれた初期値のみを保持します。**Firmware Control** 以外の実装が選択された場合、ハードウェアが事前設定され、ビルド時にその構成で固定されます。この場合、「WriteCompare APIs」はコンパイルから削除されるので、利用不可能になります。

デッドバンド

Dead Band パラメータは、PWM のデットバンド機能をイネーブル/ディスエーブルにします。デットバンドモードは、固定機能実装とわずかに異なります。デットバンド モードが 2 つのイネーブルなオプションのうちいずれかである場合、「ph1」と「ph2」出力が表示されています。オプション:

- **Disabled** – デッドバンドなし
- **0-3 Counts** ― デットバンドが最大 3 カウントまで「pwm」または「pwm1」出力に実装されます。これは PLD ロジックに実装され、カウンタ用のデータパスを利用しません。
- **2-4 Clock Cycles** – デットバンドが最大 4 クロックサイクルまで「pwm」または「pwm1」出力に実装されます。

- **2-256 Clock Cycles** – デットバンドが最大 256 クロックサイクルまで「pwm」または「pwm1」出力に実装されます。これはカウンタ用のデータパスに実装されます。

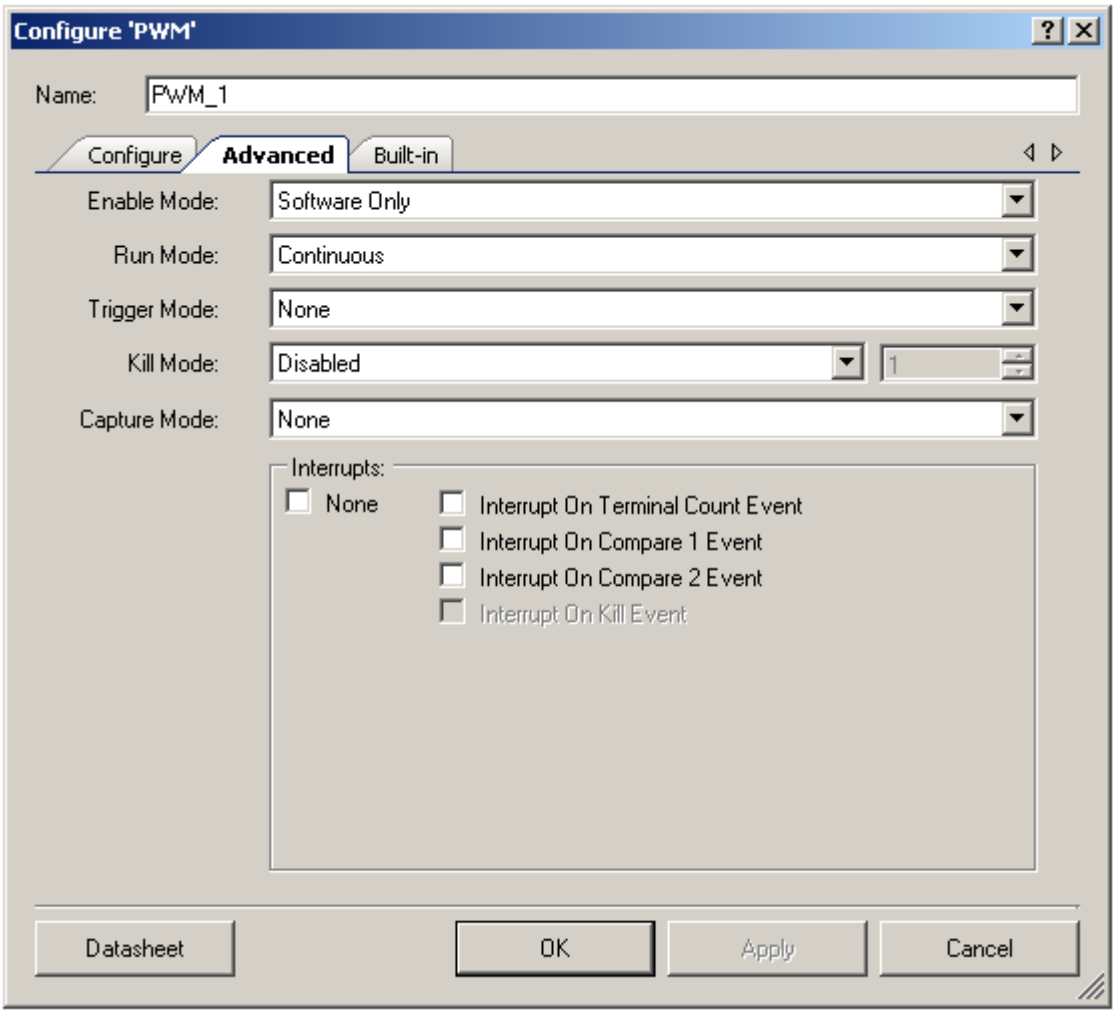
Dead Time (ソフトウェア)

デッドタイム値は、デッドバンド出力信号である ph1 と ph2 に実装されたデッドタイムの量を定義します。**Dead Band** がイネーブルで、**Dead Band** パラメーターで定義されているハードウェア設定オプションに基づいて制限されているときに限って、このパラメーターは有効です。



デッドタイムは、デッドバンドが 2～256 の範囲でイネーブルな場合のみソフトウェアで設定可能です。このデータは、PWM_WriteDeadTime() および PWM_ReadDeadTime() API の呼び出しによって制御されます。デッドバンドが 2～4 の範囲でイネーブルな場合、構成に設定されている値がハードウェアに組み込まれ、API 設定は不可となります。

Advanced Tab (詳細設定タブ)



Enable Mode (イネーブル モード)

Enable Mode パラメーターは、PWM の機種全体をイネーブルにするために、どのハードウェアとソフトウェアの組み合わせが必要かを指定します。オプション:

- **Software Only** – PWM は、コントロール レジスタ中のイネーブル ビットがソフトウェアによって制御されている場合のみイネーブルです。イネーブル モードが **Software Only** に設定されている場合、イネーブル入力は表示されません。
- **Hardware Only** – PWM は、ハードウェア イネーブル入力がアクティブ (HIGH) の場合のみイネーブルです。このモードでは、予期しない行動を防止するために、コンポーネントが正しく初期化するように、PWM_Start() API を呼び出さなければなりません。

- **Hardware And Software** ー PWM は、コントロール レジスタとハードウェア入力の両方のビットがアクティブ(HIGH)の場合にイネーブルとなります。

Run Mode (動作モード)

Run Mode パラメーターは、PWM のトリガー、開始、そして動作継続の方法を定義します。PWM は、以下に列挙した値が示すように、イネーブル入力に応じて稼働します。

- **Continuous**ーPWM はトリガ イベントで永続的に動作を続けます。
- **One-Shot Single Trigger** ー PWM はトリガ イベントで 1 回動作します。
- **One-Shot Multi Trigger** ー PWM はトリガ イベントで 1 回動作します。各周期が完了すると、PWM は次のトリガ イベントが起こるまで停止します。

Trigger Mode (トリガ モード)

トリガ モード パラメータは有効なトリガ イベントの構成要素を指定します。トリガ モードが **None** に設定されている場合、トリガ入力は表示されません。オプション:

- **None**ーイネーブルなトリガはなし (トリガはいつも真として扱われます)
- **Rising Edge (立ち上がりエッジ)**ートリガ イベントの信号はトリガ入力の立ち上がりエッジで発せられます。
- **Falling Edge (立ち下がりエッジ)** ートリガ イベントの信号はトリガ入力の立ち下がりエッジで発せられます。
- **Either Edge**ートリガ イベントの信号はトリガ入力の立ち上がりエッジまたは立ち下がりエッジで発せられます。

Kill Mode(キル モード)

Kill Mode パラメータは、ハードウェアのキル入力がアクティブの場合にハードウェアが pwm 入力を扱う方法を指定します。キル モードが **Disabled** に設定されている場合、「キル」入力は表示されません。オプション:

- **Disabled** ー イネーブルなキルはなし
- **Asynchronous** ー キルがアクティブの間、pwm 出力はディスエーブルとなります。
- **Single Cycle** ー キルがアクティブの間 pwm 出力はディスエーブルとなり、週 j 期の終わりに到達する(tc)まで再びイネーブルになりません。
- **Latched** – キルがアクティブの間 pwm 出力はディスエーブルとなり、PWM が再設定されるまで再び有効になりません。

- **Minimum Time** — pwm 出力は、キルがアクティブの間ディスエーブルとなり、最小時間が経過するまで再びイネーブルにはなりません。

Minimum Kill Time (最小キルタイム、ソフトウェア)

最小キルタイムパラメーターは、有効なキル信号となる最小長さを指定します。これは **Kill Mode** パラメーターが **Minimum Time** に設定されている場合に必要なキル信号のパラメーターです。



最小キルタイム値は、PWM_WriteKillTime() と PWM_ReadKillTime() API の呼び出しによって制御され、1～255 の間の値に制限されます。

Capture Mode (キャプチャモード)

Capture Mode パラメータは、どのハードウェアイベントによってリード FIFO にピリオド カウンタ値がキャプチャされるかを指定します。現在のカウンタ値(ソフトウェアキャプチャ)は、PWM_ReadCounter() API を呼び出すことによって、いつでも読み取ることができます。キャプチャモードが **None** に設定されている場合、キャプチャ入力は表示されません。オプション:

- **None** — イネーブルになっているキャプチャはなし
- **Rising Edge (立ち上がりエッジ)** — キャプチャ イベントの信号はトリガ入力の立ち上がりエッジで発せられます。
- **Falling Edge (立ち下がりエッジ)** — キャプチャ イベントの信号はトリガ入力の立ち下がりエッジで発せられます。
- **Either Edge** — キャプチャ イベントの信号はトリガ入力の立ち上がりエッジまたは立ち下がりエッジで発せられます。

割り込み

Interrupts パラメータを使うと、初期割り込みソースを設定できます。これらの値は、他の割り込みパラメータとの理論和で、割り込みをトリガする最終のイベントグループが得られます。**Interrupts** が **None** に設定されていない限り、ソフトウェアはいつでもこのモードを再構成できます。このパラメータはデフォルトを定義します。

- **None** — 設定されている割り込みはなし
- **Interrupt On Terminal Count Event** — このオプションはいつでも利用可能で、デフォルトでは非選択に設定されています。

- **Interrupt On Compare 1 Event** ー このオプションはデフォルトでは非選択に設定されています。これはいつも表示されています。
- **Interrupt On Compare 2 Event** ー このオプションはデフォルトでは非選択に設定されています。これは、**UDB** が **Implementation** に選択されていて、**PWM Mode** が適切に設定されている場合にのみ利用できます。
- **Interrupt On Kill Event** – このオプションはデフォルトでは非選択に設定されています。これは、**UDB** が **Implementation** に選択されていて、**PWM Mode** が適切に設定されている場合にのみ利用できます。

ローカルパラメータ (API での使用)

これらのパラメータは、API によって使用され、GUI には表示されません。

- **FixedFunctionUsed**ー固定機能ブロックを使用した PWM の実装を選択する場合、「1」(真)に指定します。
- **KillModeMinTime** – **Kill Mode** を **Minimum Time** として設定している場合は、「1」(真)に指定します。これにより、関数 PWM_WriteKillTIme() と PWM_ReadKillTime() を必要に応じて含むことができます。
- **PWMModeCenterAligned** – **PWM Mode** を **Center Aligned** として設定している場合は、「1」(真)に指定します。 PWM_ReadCompare() と PWM_WriteCompare() 関数は、このモードでは他のモードと異なって指定されます。このパラメーターは、正しい機能を追加して、不要な機能を削除します。
- **DeadBandUsed** – デッドバンドの 2～256 イネーブル モードでの実装を選択する場合、「1」(真)に指定します。これは関数 PWM_WriteDeadTime()と PWM_ReadDeadTime() API を条件付きで含めるために使用します。
- **DeadBand2_4** – デッドバンドの 2-4 カウント範囲での実装を選択する場合、「1」(真)に指定します。これはデッドタイムを処理するために必要な動作に対して、関数 PWM_WriteDeadTime()と PWM_ReadDeadTime() 内で使用されます。
- **UseStatus** – 構成が状態レジスタの使用を保証する場合に、「1」に指定します。これにより、状態レジスタのリソースが設計上で不要な場合、削除されます。
- **UseControl** – 構成がコントロール レジスタの使用を保証する場合に、「1」に指定します。これにより、状態レジスタのリソースが設計上で不要な場合、削除されます。
- **UseOneCompareMode** – 構成がシングル コンペア モード API のみの可用性を保証する場合に、「1」に指定します。これにより、選択されたアーキテクチャによって指定された API が削除されます。

Clock Select (クロック選択)

このコンポーネントには、内部クロックはありません。クロックソースを必ず取りつけてください。

警告 デバイスで固定機能ブロックを使用するように設定されている場合、PWM コンポーネントは次のように制限されます。

- クロック入力は、バスクロックと同期しているローカルクロック、またはバスクロックから直接とってきます（クロックタイプを「Existing」、ソースを「BUS_CLK」として設定）。
- クロックの周波数がバスクロックに適合すると、クロックはバスクロックから直接とっていることになります（クロックタイプを「Existing」、ソースを「BUS_CLK」として設定）。バスクロックに適合する周波数のローカルクロックでは、ビルドプロセス中にエラーが生じます。

UDB ベースのコンポーネント

コンポーネントが同期クロックをサポートする場合、デバイスの周波数範囲内であれば任意のクロック入力周波数が使用できます。

コンポーネントがバスクロックへの同期を必要とする場合、コンポーネントにクロックを供給するためにルーティングされたクロックを使用すると、[1] ルーティングされたクロックの周波数は、そのソースクロック周波数の 1/2 を超えることはできません。

- ルーティングされたクロックがバスクロックに同期している場合、それはバスクロックの 1/2 となります。
- ルーティングされたクロックがクロック分周器と同期している場合、最大値はそのクロックレートの 1/2 です。

配置

PWM コンポーネントの配置は、コンポーネントのインスタンス用の **Implementation** パラメータセレクションに基づいて行われます。**Fixed Function** が設定されている場合、PWM は固定機能ブロックのうちの 1 つに配置されます。**UDB** に設定されている場合、PWM は UDB アレイ全体に配置され、すべての配置情報は *cyfitter.h* ファイルを通じて API に提供されます。

[1] ルーティングされたクロックは、クロック入力に直接接続されたクロックシンボル以外のクロックを指します。

リソース

分解能	デジタル ブロック					API メモリ(バイト)		ピン (外部入出力ごと)
	データパス	マクロセル	ステータス レジスタ	コントロール レジスタ	Counter7	フラッシュ	RAM	
8 ビット One Output Mode[2]	1	6	1	1	0	226	5	-
8 ビット Two Outputs Mode[2]	1	9	1	1	0	237	5	-
8 ビット Dual Edged Mode[2]	1	8	1	1	0	237	5	-
8 ビット Center Align Mode[2]	1	9	1	1	0	226	5	-
8 ビット HW Select Mode[2]	1	9	1	1	0	237	5	-
8 ビット Dither Mode[2]	1	9	1	1	0	231	5	
16 ビット One Output Mode[2]	2	6	1	1	0	275	5	1/2
デッドバンド付き PWM8 2-4[3]	1	12	1	2	0	272	6	+2
デッドバンド付き PWM16 2-4[3]	2	12	1	2	0	321	7	
デッドバンド付き PWM8 2-256[3]	2	11	1	1	0	272	6	+2

[2] 構成 1: 対応する PWM モードの PWM と分解能は、「Software Only Enable」モード、「Continuous Run」モード、「None」に設定された「Trigger」モード 、「Kill」モードおよびノーデッドバンドと TC 上の割り込みでディスエーブルにされた「Capture」モードで設定できます。

[3] 構成 2: 2～4 のデッドバンド範囲と 2～256 のデッドバンド範囲は相互に排他的です。PWM は、「Software Only Enable」モードで、相当する分解能と「One Output PWM」モードに構成されます。「Trigger」 モードは、「None」、「Kill」モードと「Capture」モードは、TC 上の割り込みでディスエーブルに設定されます。

分解能	デジタル ブロック					API メモリ(バイト)		ピン (外部入出力ごと)
	データパス	マクロセル	ステータス レジスタ	コントロール レジスタ	Counter7	フラッシュ	RAM	
デッドバンド付き PWM16 2-256 [3]	3	11	1	1	0	308	7	

アプリケーション・プログラミング・インターフェース

アプリケーション・プログラミング・インターフェース (API) ルーチンにより、ソフトウェアを使用してコンポーネントを設定できます。次の表は、各関数へのインターフェースとその説明を示しています。その次のセクションでは、各関数について詳しく説明します。

デフォルトでは、PSoC Creator はインスタンス名「PWM_1」を設計上のコンポーネントの最初のインスタンスに割り当てます。インスタンス名は、識別子の構文ルールに従った固有の値に変更できます。インスタンス名は、すべてのグローバル関数名、変数名、定数名のプリフィックスになります。読みやすいように、下表では「PWM」というインスタンス名を使用しています。

関数	機能
PWM_Start()	PWM をデフォルトのカスタマイザ値で初期化します。
PWM_Stop()	PWM の動作をディスエーブルにします。ソフトウェア制御のイネーブル モードで、コントロール レジスタのイネーブル ビットをクリアします。
PWM_SetInterruptMode()	割り込みソース状態レジスタの割り込みマスク制御を設定します。
PWM_ReadStatusRegister()	ステータス レジスタの現在の状態を返します。
PWM_ReadControlRegister()	コントロール レジスタの現在の状態を返します。
Counter_WriteControlRegister()	コントロール レジスタのビット フィールドを設定します。
PWM_SetCompareMode()	ディザーモード、中央位置設定モード、1 出力モードに設定されている場合、コンペア 出力用にコンペア モードを書き込みます。
PWM_SetCompareMode()	コントロール レジスタにコンペア1出力用のコンペア モードを書き込みます。
PWM_SetCompareMode2()	コントロール レジスタにコンペア2出力用のコンペア モードを書き込みます。
PWM_ReadCounter()	現在のカウンタ値を読み取ります（ソフトウェアキャプチャ）。
PWM_ReadCapture()	キャプチャ FIFO からキャプチャ値を読み取ります。
PWM_WriteCounter()	カウンタレジスタに新しいカウンタ値を直接書き込みます。これは現在実行中の周期用としてのみ実装されます。
PWM_WritePeriod()	PWM ハードウェアが使用するピリオド値を書き込みます。

関数	機能
PWM_ReadPeriod()	PWM ハードウェアが使用するピリオド値を読み取ります。
PWM_WriteCompare()	インスタンスが ディザーモード、中央位置設定 モード、1出力 モードとして指定される場合に、比較値を書き込みます。
PWM_ReadCompare()	インスタンスが ディザーモード、中央位置設定 モード、1出力 モードとして指定される場合に、比較値を読み取ります。
PWM_WriteCompare1()	コンペア1出力用に比較値を書き込みます。
PWM_ReadCompare1()	コンペア 1出力用に比較値を読み取ります。
PWM_WriteCompare2()	コンペア2出力用に比較値を書き込みます。
PWM_ReadCompare2()	コンペア2出力用に比較値を読み取ります。
PWM_WriteDeadTime()	デッドバンド実装でハードウェアが使用するデッドタイム値を書き込みます。
PWM_ReadDeadTime()	デッドバンド実装でハードウェアが使用するデッドタイム値を読み取ります。
PWM_WriteKillTime()	キル モードが「Minimum Time(最小時間)」に設定されているときにハードウェアが使用するキルタイム値を書き込みます。
PWM_ReadDeadTime()	キル モードが「Minimum Time(最小時間)」に設定されているときにハードウェアが使用するキルタイム値を読み取ります。
PWM_ClearFIFO()	キャプチャ FIFO から、キャプチャデータをすべてクリアします。
PWM_Sleep()	動作を停止し、ユーザ設定を保存します。
PWM_Wakeup()	ユーザ設定を復元し、イネーブルにします。
PWM_Init()	コンポーネントのパラメータを回路図に配置されたカスタマイザでの設定に合わせて初期化します。
PWM_Enable()	PWM ブロック動作をイネーブルにします。
PWM_SaveConfig()	コンポーネントの現在のユーザ構成を保存します。
PWM_RestoreConfig()	コンポーネントの現在のユーザ構成を復元します。

グローバル変数

変数	機能
PWM_initVar	PWM が初期化されたかどうかを示します。変数は、0に初期化され、PWM_Start() が初めて呼び出されたときに1にセットされます。これにより、コンポーネントは PWM_Start() ルーチンへの最初の呼び出し後、再初期化なしに再起動できます。 コンポーネントの再初期化が必要な場合、関数 PWM_Start() や PWM_Enable() の前に、関数 PWM_Init() が呼び出されます。

void PWM_Start(void)

機能: PWM をデフォルトのカスタマイザ値で初期化します。いずれかのソフトウェア制御有効モードで、コントロール レジスタのイネーブル ビットを設定することにより、PWM 動作をイネーブルにします。

パラメータ: なし

戻り値: なし

副作用: PWM のコントロール レジスタのイネーブル ビットを設定します。**Enable Mode** が **Hardware Only**に設定されている場合、PWM に影響を与えません。**Enable Mode** が **Hardware and Software**に設定されている場合、このモードのソフトウェア部分をイネーブルにできるのはこれだけで、PWM をイネーブルにするには、ハードウェア入力もイネーブルにならなければなりません。

void PWM_Stop(void)

機能: いずれかのソフトウェア コントロール イネーブル モードで、コントロール レジスタの第 7 ビットをリセットすることにより、PWM 動作をディスエーブルにします。選択された固定機能ブロックをディスエーブルにします。

パラメータ: なし

戻り値: なし

副作用: PWM のコントロール レジスタのイネーブル ビットをクリアします。**Enable Mode** が **Hardware Only** に設定されていると、この関数は PWM に影響を与えません。**Enable Mode** が**Hardware and Software**に設定されている場合、これがこのモードのソフトウェア部分をディスエーブルにでき、ハードウェア入力はこれ以上 PWM のイネーブルに何の影響も与えません。

void PWM_SetInterruptMode(uint8 interruptMode)

機能: 割り込みソース状態レジスタの割り込みマスク制御を設定します。

パラメータ: uint8 interruptMode: イネーブルな割り込みソースを含むビット フィールド

戻り値: なし

副作用: なし

uint8 PWM_ReadStatusRegister(void)

機能: ステータス レジスタの現在の状態を返します。

パラメータ: なし

戻り値: uint8: 現在の状態レジスタ値 状態レジスタのビットは以下のとおりです。
[7:6] : 未使用 (0)
[5] : キルイベント出力
[4] : FIFO が空ではありません
[3] : FIFO がフルです
[2] : ターミナル カウント
[1] : コンペア出力2
[0] : コンペア出力1

副作用: 読み取りで、ステータス レジスタのビットがクリアされる可能性があります。

uint8 PWM_ReadControlRegister(void)

機能: コントロール レジスタの現在の状態を返します。コントロール レジスタが削除されていない場合のみ、APIは利用可能です。

パラメータ: なし

戻り値: uint8: 現在のコントロール レジスタ値

副作用: なし

void PWM_WriteControlRegister(uint8 control)

機能: コントロール レジスタのビット フィールドを設定します。コントロール レジスタが削除されていない場合のみ、APIは利用可能です。

パラメータ: uint8 control: コントロール レジスタのビット フィールド。状態レジスタのビットは次のとおりです。
[7] : PWM Enable
[6] : Reset
[5:3] : Compare Mode 2
[2:0] : Compare Mode 2

戻り値: なし

副作用: なし

void PWM_SetCompareMode(enum comparemode)

機能: ディザーモード、中央位置設定モード、1出力モードに設定されている場合、コンペア出力用にコンペア モードを書き込みます。

パラメータ: enum comparemode: Compare mode enumerated type

戻り値: なし

副作用: なし

void PWM_SetCompareMode1(enum comparemode)

機能: コントロール レジスタにコンペア1出力用にコンペアモードを書き込みます。

パラメータ: enum comparemode: Compare mode enumerated type

戻り値: なし

副作用: なし

void PWM_SetCompareMode2(enum comparemode)

機能: コントロール レジスタにコンペア2出力用にコンペアモードを書き込みます。

パラメータ: enum comparemode: Compare mode enumerated type

戻り値: なし

副作用: なし

uint8/16 PWM_ReadCounter(void)

機能: 現在のカウンタ値を読み取ります(ソフトウェアキャプチャ)。

パラメータ: なし

戻り値: uint8/uint16: 現在のピリオドカウンタ値

副作用: なし

uint8/16 PWM_ReadCapture(void)

機能: キャプチャ FIFO からキャプチャ値を読み取ります。

パラメータ: なし

戻り値: uint8/uint16: 現在のキャプチャ値

副作用: なし

注 FIFOs は低電力モードになった後で、消去されます。必要な場合は、低電源モードになる前に、キャプチャ FIFO からデータを読み出してください。

void PWM_WriteCounter(uint8/16 period)

機能: カウンタレジスタに新しいカウンタ値を直接書き込みます。これは現在実行中の周期用としてのみ実装されます。

パラメータ: uint8/uint16 period: ピリオド カウンタ値

戻り値: なし

副作用: なし

void PWM_WritePeriod(uint8/16 period)

機能: PWM ハードウェアが使用するピリオド値を書き込みます。

パラメータ: period: 分解能に応じてuint8または16。新しいピリオド値。

戻り値: なし

副作用: なし

uint8/16 PWM_ReadPeriod(void)

機能: PWM ハードウェアが使用するピリオド値を読み取ります。

パラメータ: なし

戻り値: uint8/16: ピリオド値

副作用: なし

void PWM_WriteCompare(uint8/16 compare)

機能: **PWM Mode** パラメータが**ディザー** モード、**中央位置設定** モード、または **1出力** モードに設定されている場合、コンペア出力用にコンペアモードを書き込みます。

パラメータ: uint8/16: 比較値

戻り値: なし

副作用: PWM が実行され機能している場合に、PWM_WriteCompare() APIを使用すると、指定されたコンペア モードおよびコンペア タイプによっては、PWM出力をただちに生じることがあります。コンペア出力の変化は、新しい比較値がコンペア レジスタにプログラムされるとすぐに発生します。デッドバンドがイネーブルな場合は、PWM 出力の変化はデッドバンド論理も引き起こします。

uint8/16 PWM_ReadCompare(void)

機能: **PWM Mode** パラメーターが**ディザー** モード、**中央位置設定** モード、または **1出力**モードに設定されている場合、コンペア出力用に比較値を読みだします。

パラメータ: なし

戻り値: uint8/uint16: 電流比較値

副作用: この関数は、**PWM Mode** パラメーターが上記に説明されているモードのうち 1 つに設定されている場合にのみ、利用できます。それ以外の場合は、ReadCompare1/2 関数を呼び出さなければなりません。

void PWM_WriteCompare1(uint8/16 compare)

機能: 比較値をコンペア1出力用に書き込みます。

パラメータ: uint8/uint16: pwm1 の新しい比較値

戻り値: なし

副作用: なし

uint8/16 PWM_ReadCompare1(void)

機能: 比較値をコンペア1用に読み取ります。

パラメータ: なし

戻り値: uint8/uint16: 電流比較値1

副作用: なし

void PWM_WriteCompare2(uint8/16 compare)

機能: 比較値をコンペア2出力用に書き込みます。

パラメータ: uint8/uint16: pwm2 の新しい比較値

戻り値: なし

副作用: なし

uint8/16 PWM_ReadCompare2(void)

機能: 比較値をコンペア2用に読み取ります。

パラメータ: なし

戻り値: uint8/uint16: 現在の比較値

副作用: なし

void PWM_WriteDeadTime(uint8 deadband)

機能: デッドバンド実装でハードウェアが使用するデッドタイム値を書き込みます。

パラメータ: uint8: デッドバンドカウント

戻り値: なし

副作用: なし

uint8 PWM_ReadDeadTime(void)

機能: デッドバンド実装でハードウェアが使用するデッドタイム値を読み取ります。

パラメータ: なし

戻り値: uint8: デッドバンドカウントの現在の設定

副作用: なし

void PWM_WriteKillTime(uint8 killtime)

機能: **Kill Mode** が **Minimum Time**に設定されているときにハードウェアが使用するキルタイム値を書き込みます。

パラメータ: uint8: 最小時間キルカウント

戻り値: なし

副作用: なし

uint8 PWM_ReadkillTime(void)

機能: **Kill Mode**が **Minimum Time**に設定されているときにハードウェアが使用するキルタイム値を読み出します。

パラメータ: なし

戻り値: uint8: 現在の最小時間キルカウント

副作用: なし

void PWM_ClearFIFO(void)

機能: キャプチャ FIFO から以前キャプチャしたデータをクリアします。ここで PWM_ReadCapture() は FIFO が空になるまで呼び出されます。

パラメータ: なし

戻り値: なし

副作用: なし

void PWM_Sleep(void)

機能: 動作を停止し、ユーザ設定を保存します。

パラメータ: なし

戻り値: なし

副作用: なし

void PWM_Wakeup(void)

機能: ユーザ設定を復元し、イネーブルにします。

パラメータ: なし

戻り値: なし

副作用: なし

void PWM_Init(void)

機能: コンポーネントのパラメーターを回路図に配置されたカスタマイザでの設定に合わせて初期化します。コンペアノードは、コントロール レジスタの各ビットを設定することにより、設定されます。ステータス レジスタからの出力として、割り込みが選択されます。固定機能モードを使用している場合、選択された固定機能ブロックがイネーブルになります。FIFO をクリアすると、FIFO フルビットがイネーブルになり、ステータス レジスタで設定されます。通常は PWM_Start() で呼び出されます。

パラメータ: なし

戻り値: なし

副作用: すべてのレジスタがリセットされ、初期値になります。これによって、コンポーネントが再初期化されます。

void PWM_Enable(void)

機能: PWMブロック動作を有効にし、コントロール レジスタの第7ビットを設定します。

パラメータ: なし

戻り値: なし

副作用: なし

void PWM_SaveConfig(void)

機能: コンポーネントの現在のユーザ構成を保存します。周期、デッドバンド、カウンタ、制御の各レジスタ値が保存されます。

パラメータ: なし

戻り値: なし

副作用: なし

void PWM_RestoreConfig(void)

機能: コンポーネントの現在のユーザ構成をリストアします。

パラメータ: なし

戻り値: なし

副作用: なし

ファームウェアソースコードのサンプル

PSoC Creator は、[Find Example Project (プロジェクト例を検索)] ダイアログに数多くのプロジェクト例を提供しており、そこには回路図およびコード例が含まれています。コンポーネントを使用した設計例を見るには、[Component Catalog (コンポーネントカタログ)] または回路図に置いたコンポーネントインスタンスからダイアログを開きます。一般的なサンプルについては、[Start Page (スタートページ)] または **File** メニューからダイアログを開きます。必要に応じてダイアログにある **Filter Options** を使用し、選択できるプロジェクトのリストを絞り込みます。

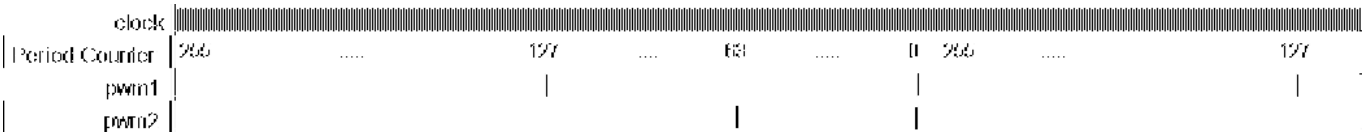
詳しくは、PSoC Creator ヘルプの「Find Example Project (プロジェクト例を検索)」トピックを参照してください。

機能説明

デフォルト構成

PWM 用のデフォルトは、2 出力の 8 ビット PWM です。これは 12MHz 未満のクロックを使用して、127 未満の比較 (255 のピリオド) で 1 つの出力を 63 未満で、第 2 の出力を生成します。図 1 デフォルトによる PWM の入力と出力を示しています。

図 1. PWM 入力と出力



固定機能ブロックの限界

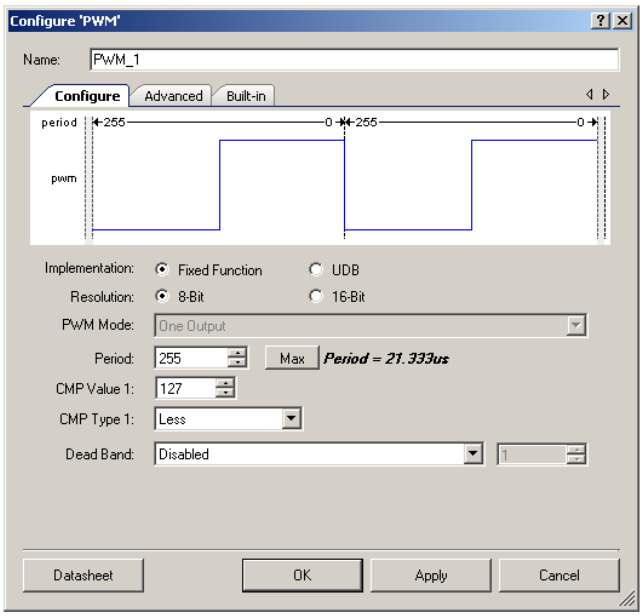
PWM の「Fixed Function (固定機能)」による実装は、構成可能なハードウェアブロックで機能を削減した PWM を実装することで、使用 UDB リソースを減らします。これらのブロックにおける PWM 機能には、次に挙げる制限があります。

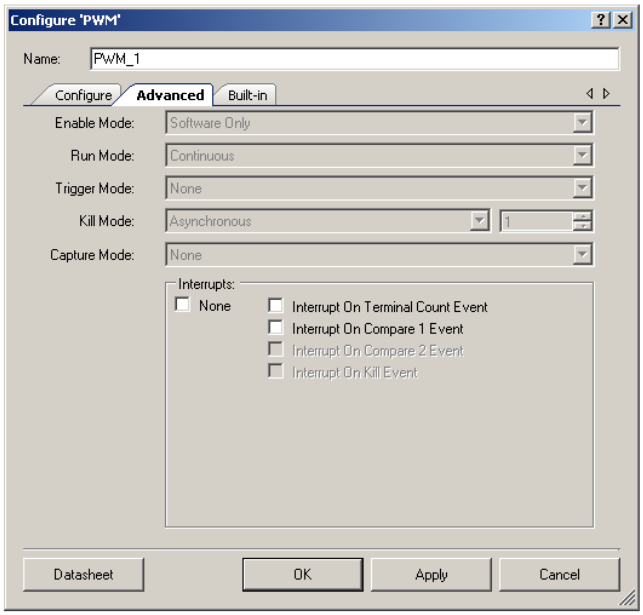
- カウンタ値アクセスなし – PWM_ReadCapture () および PWM_ReadCounter() は利用できません。

- 1 出力モードのみ – 中央位置設定、デュアルエッジ、調整、2 出力モードなし
- 非同期キル モードのみ
- トリガなし
- 連続動作モードのみ
- ソフトウェアイネーブルのみ – ハードウェア イネーブル モードなし
- デッドバンド機能縮小 – 0～3 のデッドバンド
- デッドバンドがイネーブルの場合の縮小 I/O – TC と CMP1 はそれぞれ PH1 と PH2 になります。

Fixed Function 実装を選択すると、**Configure** タブ ダイアログと **Advanced** タブ ダイアログは、、図 2 に示すように、パラメータを フィールドに設定し、オプションを無効にして、これらの制限を表示します。

図 2. 固定機能設定

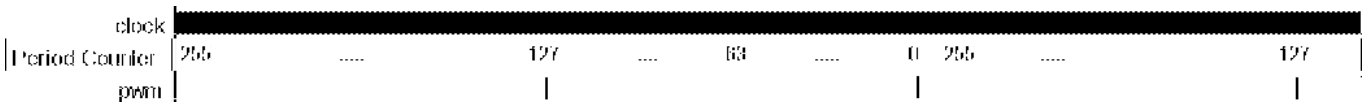




PWM モード

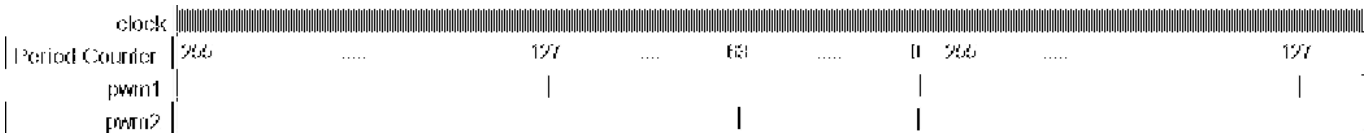
1 出力

1 出力の PWM には出力が 1 つしかなく、単一比較値と単一コンペア モードで制御されています。波形は **Greater** または **Greater or Equal** のコンペア モードと左端設定したままにできます。あるいは、**Less** または **Less or Equal** のコンペア モードと右端設定もできます。



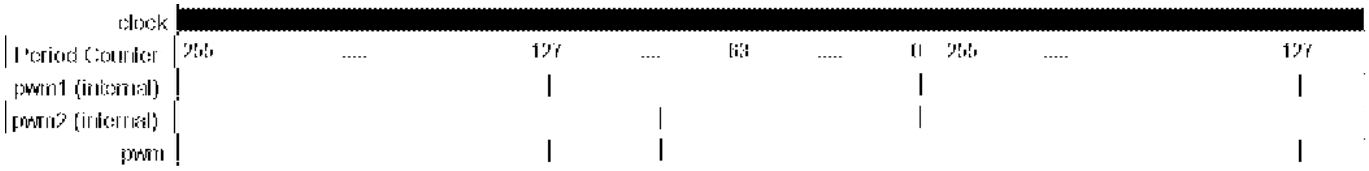
2 出力

2 出力の PWM は上述のとおりデフォルトです。2 つの PWM 出力は、2 つの比較値と 2 つのコンペア モードを使用して、それぞれ独立して定義します。この 2 つの出力はそれぞれ、上述の 1 出力 モードで説明しているように、左端または右端位置設定にしておくことができます。



デュアルエッジ

デュアルエッジ PWM は、2 つのコンペア出力と 2 つのコンペア モードを使用して単一の PWM 出力を生成します。最終出力は、2 つの比較値とコンペア モードによって定義される 2 つの異なる信号の AND です。このモードを使用するには、異なるモードで何が生成されるかを理解する必要があります。パラメーター編集カスタマイザの波形例は、最終波形の外観を提示します。しかし、比較値、コンペア モード、ピリオド値は全てランタイムで設定可能です。最終設定を理解せずにこのような値を変更すると、出力が 0 値になりかねません。

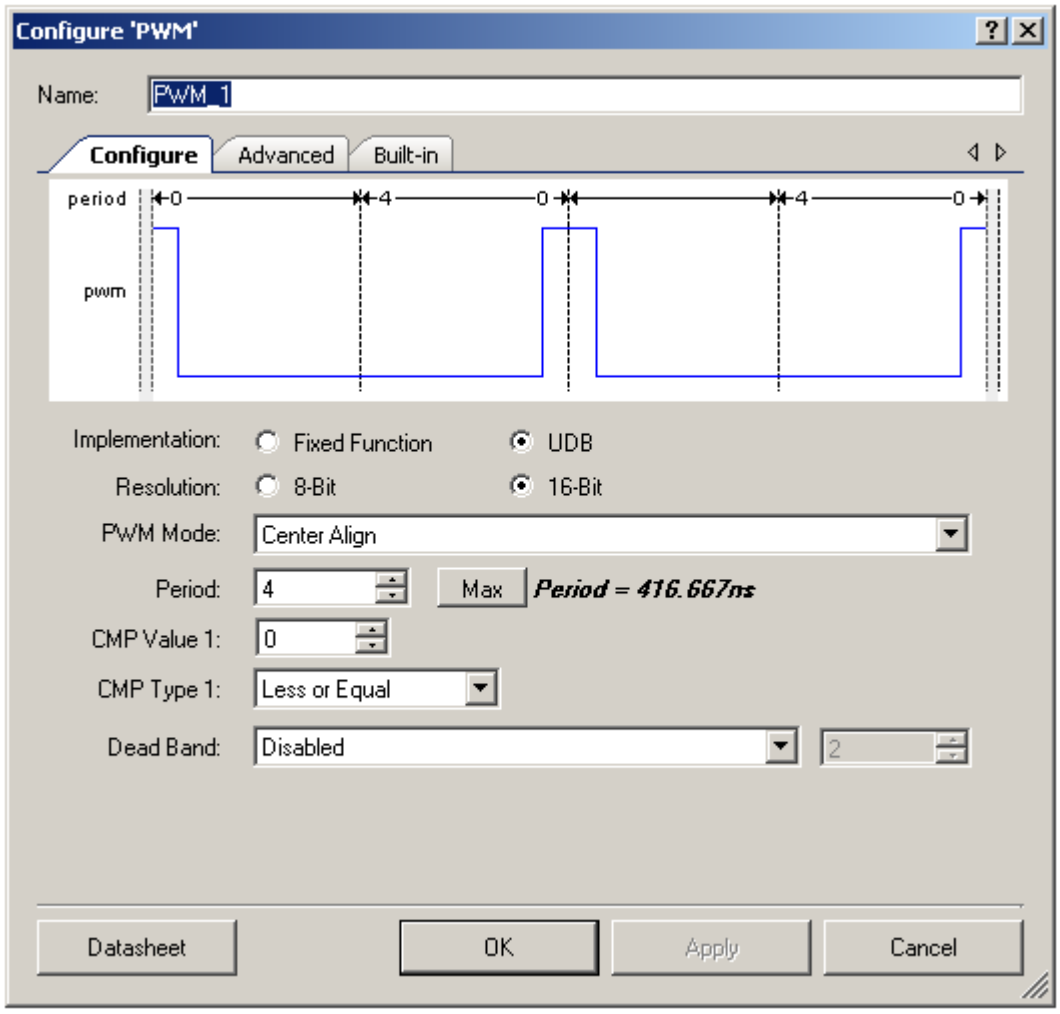


中央位置設定

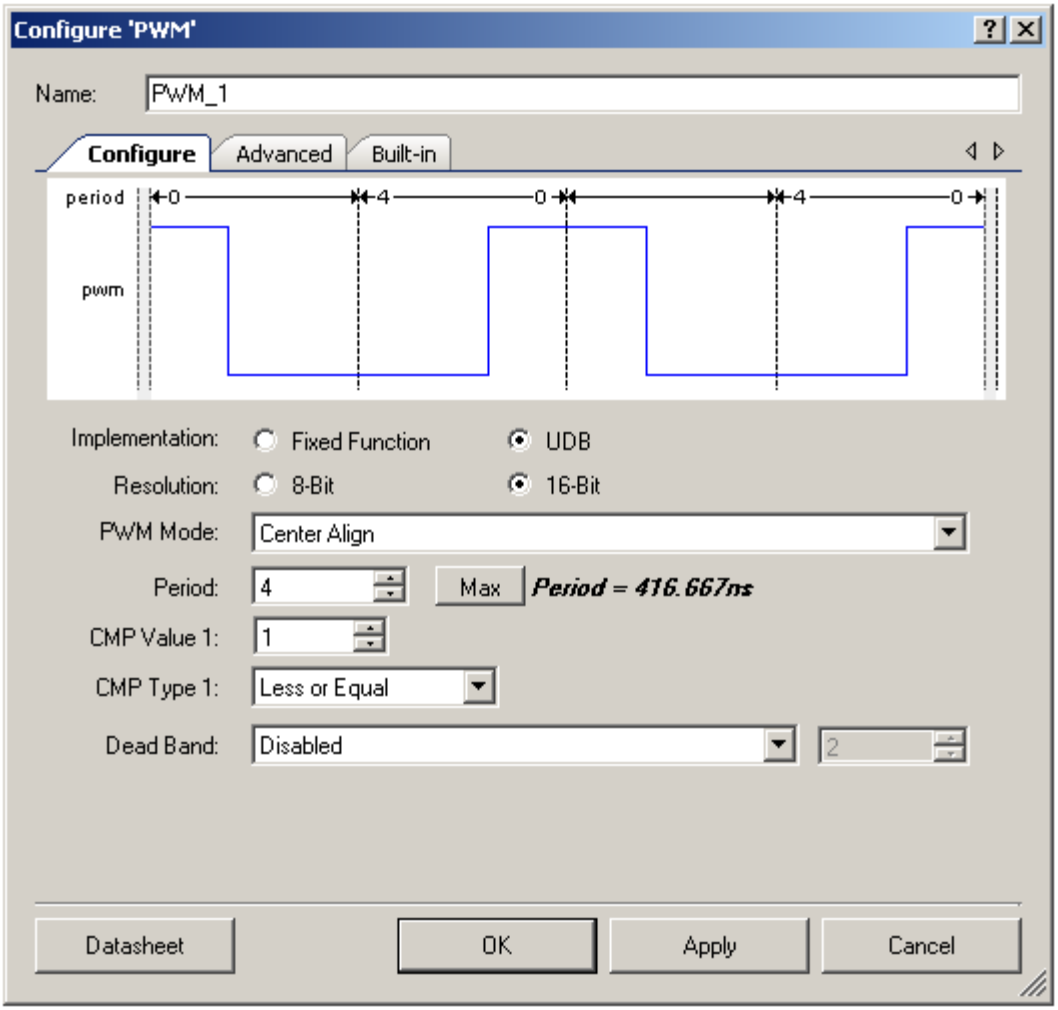
中央位置設定の PWM は、他のモード全てと違う形で PWM を実装します。望ましい出力は、ピリオド カウンタがゼロで始まってピリオド値までカウントアップし、ピリオド値に達したらカウントダウンしてゼロになります。このモードでは、ピリオド値は最終出力のピリオドの半分です。この機能では、単一の比較値とコンペア モードが利用できます。

PWM のその他のモードはいずれも、周期設定でピリオド カウンタが開始されて 0 までカウントダウンし、ピリオド値までリロードすることにより、実際のピリオド値に対して period+1 となります。これはカスタマイザに表示された計算した周期を表しています。中央位置設定モードでは、計算されたピリオドは NOT +1 です。これは、0 からピリオドまでのピリオド カウンタのカウントが即座にカウント バック ダウンを開始するためです。4 ピリオドの例では、カウンタは「0,1,2,3,4,3,2,1,0,1,2… 」とカウントし、4 クロックサイクルの周期を生成します。

カスタマイザは、表示される波形の最初と最後に半ビットの時間でこれを示します。



このカウンタの構成は単一クロックサイクルではゼロと等しいかゼロ以下です。これは最初のピリオドの始めと終わりの半ビット時間として示されます。次の図は、カウンタが 3 クロックサイクルで 1 以下である様子を示しています。これは波形の 2 つの周期のうちそれぞれで、始めと終わりで 1.5 クロックサイクルとして示されます。



ハードウェア選択

ハードウェア選択 PWM は、2 出力 PWM として実装されます。この実装には 2 つの独立した比較値とコンペア モードがあります。ハードウェア入力「cmp_sel」は、2 つの入力のうち、最後に PWM 出力となる方を選択します。これにより、必要に応じ、パラメーターを修正せずに、2 つの事前設定値の間で切り替えることが可能になります。

Dither

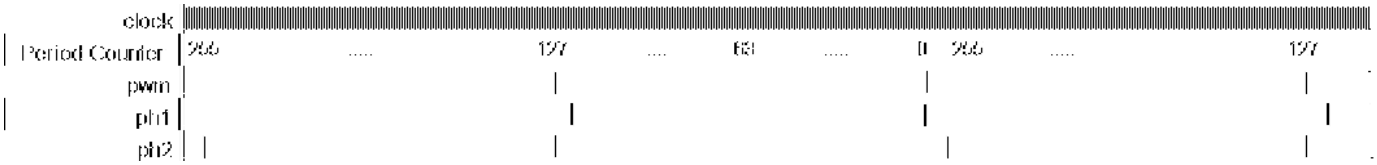
ディザーモード PWM は、最初の値と比較値は 1 の差があり、両方のコンペア モードが同一であるという条件化で、ハードウェア選択モード PWM として実装されます。また、ハードウェア選択を制御する内蔵のステートマシンもあります。このモードでは、「cmp_sel」入力は使用できません。このモードでは、表示されているパラメーターフィールドでオフセットを 0.00、0.25、0.50、0.75 から選択します。オフセットが 0.00 に設定されている場合、出力はいつでも「compare1」出力となります。0.25 に設定されている場合、出力は 3 サイクルでは「compare1」、単一サイクルでは「compare1 + 1」となります。

Dither Mode (ディザーモード)	サイクル0	サイクル1	サイクル2	サイクル3
0.00	Compare1	Compare1	Compare1	Compare1
0.25	Compare1 + 1	Compare1	Compare1	Compare1
0.50	Compare1	Compare1 + 1	Compare1	Compare1 + 1
0.75	Compare1 + 1	Compare1 + 1	Compare1 + 1	Compare 1

デッドバンド

デッドバンドは、上述の PWM モードのいずれにもアド オンできるオプションです。デッドバンドがイネーブルになると、2 つの新しい出力 ph1 と ph2 (フェーズ 1 とフェーズ 2) がシンボルで表示されます。デッドバンド出力は単一 PWM 出力に作用します。2 出力モード以外のすべてのモードで、デッドバンド出力は単一 PWM 出力に関連します。2 出力モードでは、デッドバンドは pwm1 出力にのみ実装されます。すべてのデッドバンド モードでは、オリジナルの出力も ph1 と ph2 出力も利用できます。

デッドバンドは、2～4 クロック サイクルまたは 2～256 クロック サイクル範囲のデッドバンド タイムとして構成できます。2～4 クロック サイクル範囲は、フル データパスを使用する代わりに、PLD のカウンタを実装することにより、リソースの使用を低減するよう設計されています。2～256 範囲のデッドバンドが選択された場合、フル データパスと必要なロジックが UDB アレイから使用されます。



Kill Mode(キル モード)

デッドバンドと同様に、キル モードもアド オン機能で、内部の pwm 実装を阻害しません。このアド オンは PWM の出力に配置され、最終出力信号のみを操作します。デッドバンドが実装されていない場合、キル操作は PWM 出力を LOW に引き下げることにより、ディスエーブルにします。デッドバンドが実装されている場合、キル操作は ph1 を LOW に ph2 を HIGH にすることにより、ph1 と ph2 出力をディスエーブルにします。

Asynchronous (非同期)

非同期キル モードでは、キル入力がアクティブ (HIGH) の間、出力はディスエーブルで、キル出力がインアクティブになると即、出力は再びイネーブルになります。

Single Cycle (シングルサイクル)

シングルサイクル キル モードでは、キル入力がアクティブ (HIGH) の間は出力はディスエーブルとなり、次の周期の開始時に出力は再びイネーブルになります。

Latched (ラッチ済み)

ラッチ済みキル モードでは、キル入力がアクティブになるときに出力はディスエーブルになります。PWM のリセット後もキル入力がアクティブにならない場合は、PWM 出力は再びイネーブルになります。それ以外の場合、インアクティブのキル入力を伴う PWM の次のリセットまでキルの状態にとどまります。

最小時間

最小時間キル モードでは、キル入力がアクティブ (high) である間、出力はディスエーブルです。キル入力が無効になり、最小時間が経過した後、出力は再びイネーブルになります。このモードでは、最小キルタイムにクロックカウント 1～255 の間の値を設定します。最小キルタイムカウントを制御するために必要な API は、このキル モードが選択されている場合にのみ利用できます。

Run Mode (動作モード)

連続

連続動作モードは PWM のデフォルトで、このモードでは、PWM はイネーブルの間、永久に動作し続けられます。PWM がイネーブルな限り、出力は周期を次々と巡回し、指定されたパルス幅出力を実行します。

ワンショットシングル

ワンショットシングル動作モードは、有効なトリガ イベントで単一ピリオドに PWM を実行します。ピリオドが満了すると、PWM は停止します。ピリオド レジスタからの値をカウンタにリロードした後で、PWM は停止します。ハードウェアリセット パルスが PWM を再度準備させ、次のトリガ イベントによる、次の PWM ピリオド実行を可能にします。

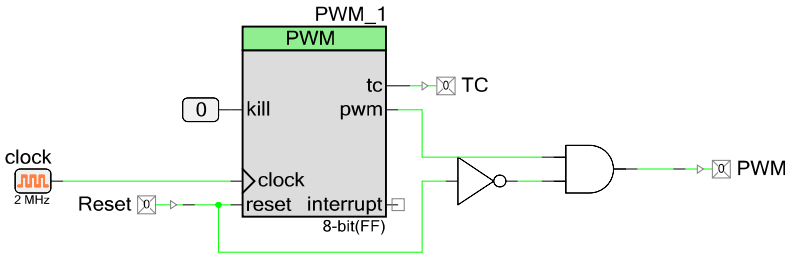
ワンショットマルチ

ワンショットマルチ動作モードは、有効なトリガ イベントで単一ピリオドに PWM を実行します。ピリオドが満了すると、PWM は停止し次のトリガ イベントに再度準備します。ピリオド レジスタからの値をカウンタにリロードした後で、PWM は停止します。ワンショットシングル動作モードとワンショットマルチ動作モードの違いは、ワンショットマルチはリセットせずに再準備するということです。

固定機能ブロックでのリセット

PWM の固定機能実装は、リセットの間 PWM 出力が HIGH になるという点で UDB 実装とは異なります。UDB 実装では PWM 出力が LOW になります。図 3 に示すように、2 つの実装のいずれかの機能を変更することは簡単にできます。図 3 は、リセット入力がアクティブの間 PWM 出力が LOW となる固定機能タイマの実装を示しています。これにより、UDB 実装と同じ機能が提供されます。

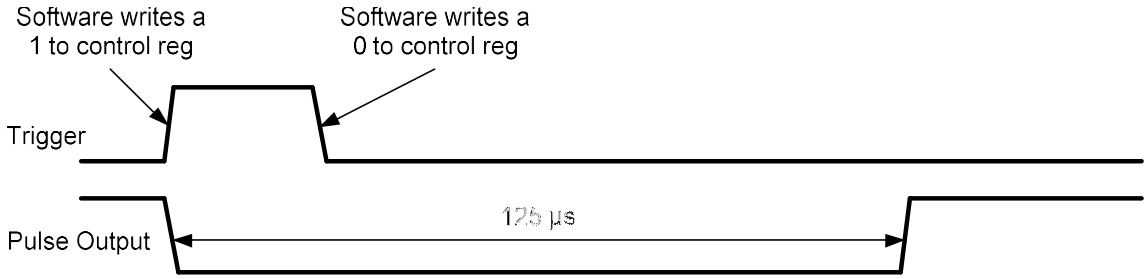
図 3. 固定機能 PWM 実装回路図



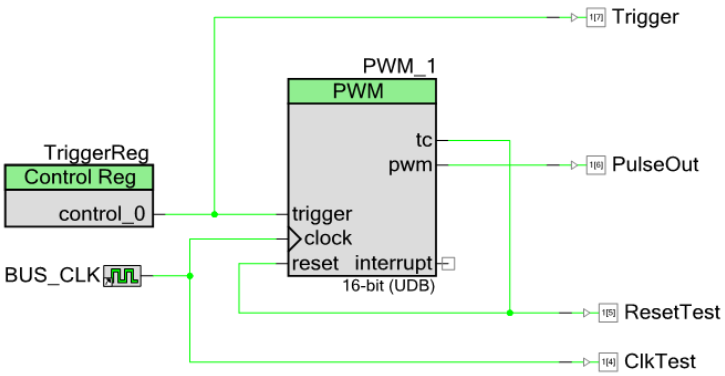
この変更を固定機能に実装することが推奨されますが、この固定機能実装では処理する出力は 1 つだからです。それ対し、UDB 実装では 1 モードで 2 つあります。また、ほとんどの状況では、リセットの間、出力を LOW にしておくことが推奨されます。

「Pulse Generator (パルス発生器)」としての PWM コンポーネント

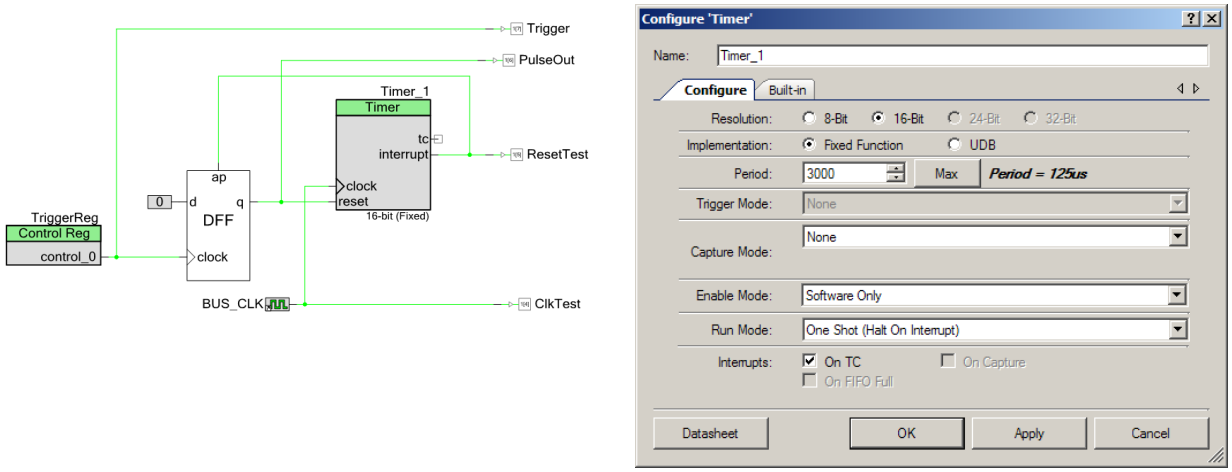
PWM コンポーネントは、ソフトウェアがトリガするパルス発生器回路の設計に用いられ、既知の周期の時間パルスを生成します。以下のタイミング ダイヤグラムは、ソフトウェアトリガ上で 125 µs のタイミング パルスを生成する、パルス生成アプリケーションの一例を示しています。



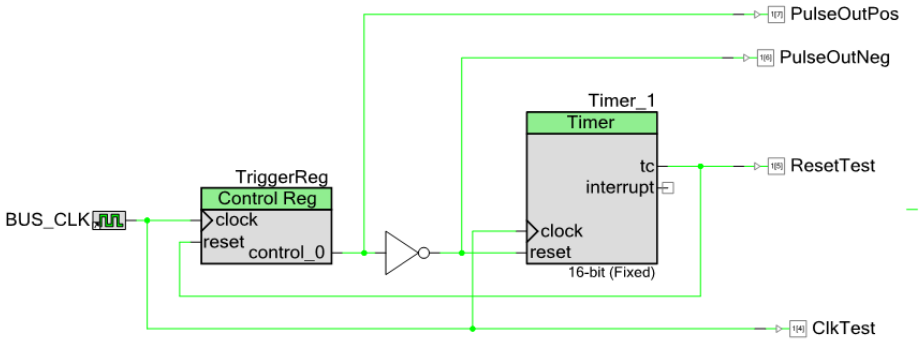
ワンショット シングル動作モードで設定された PWM がコントロール レジスタ コンポーネントと一緒に使用され、この設計を可能にします。ワンショット シングル モードでは、そのピリオド値に達した後 PWM をリセットし、正確に機能していることを確認します。TC 出力をそのリセット入力に接続した後で上記が可能です。以下の回路図は、このような回路を作る PWM の UDB 実装を示します。



以前の設計は UDB リソース 重視です。以下に示す回路図のように、固定機能実装を選択することができます。この設計は追加の DFF コンポーネントを使用しており、以前の実装とは違って、データパス要素を使用しません。



プロダクション PSoC 3 では、コントロール レジスタコン ポーネントを異なった設定にできるので、DFF を必要としません。この設定では、コントロール レジスタ書き込み関数は、1 を書き込みのに一度だけ呼び出される必要があるです。プロダクション PSoC3 へのこのタイミング パルス生成回路の実装は、以下の回路図で示されています。

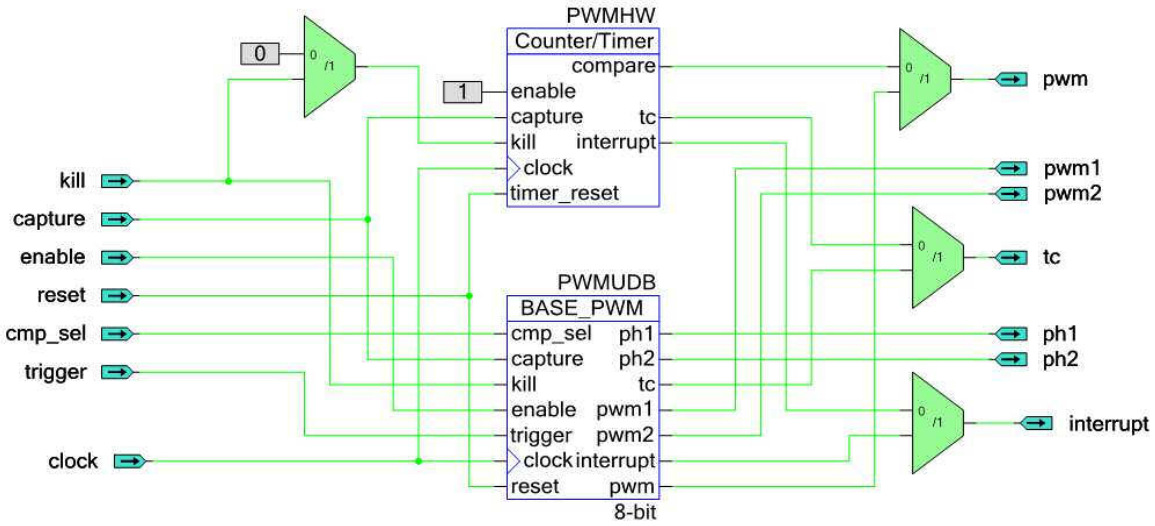


ブロック ダイアグラムと設定

PWM は、固定機能ブロックまたは UDB コンポーネントを使用して実装することができます。詳細設定パラメーター**Implementation** を使用すると、このコンポーネントを配置するブロックをユーザが指定できます。固定機能実装は、タイマ/カウンタ/PWM ブロックのうち 1 つを消費します。固定機能または UDB 構成のどちらでも、すべてのレジスタと API は同一で、単一のルック アンド フィールを実現します。API は前セクションに説明されており、レジスタの説明では PWM の全体的な実装を定義しています。

図 4 に示すように、ユーザが選択した 2 つのハードウェア実装が最上位の回路図から選択されます。

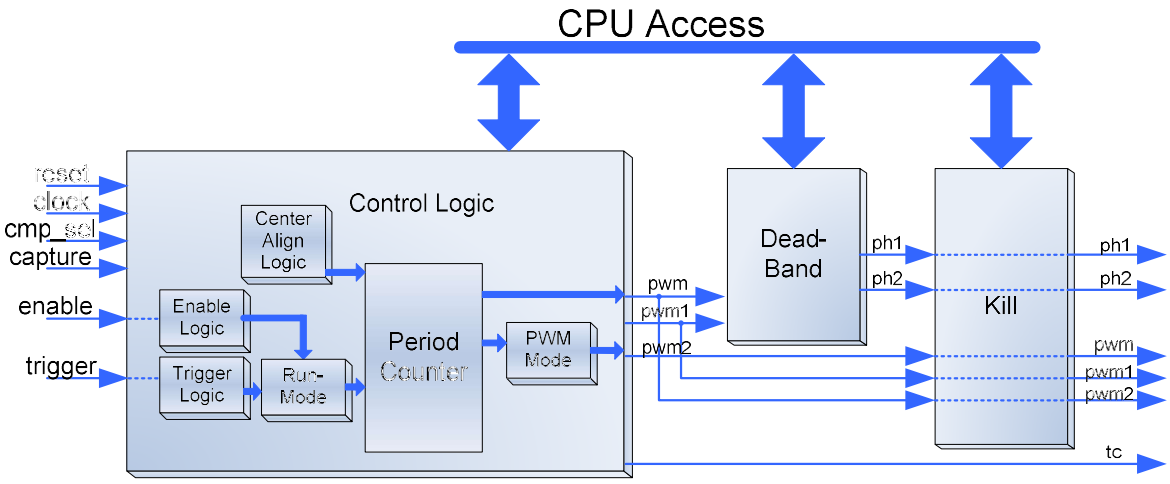
図 4. 最上位の回路図



このコンフィギュレーションを使うと、固定機能ブロックまたは UDB 実装のどちらかを選択することができます。I/O のルーチンはバックグランドでハンドルされており、この単一コンポーネントのルック アンド フィールを実現します。

UDB 実装は 図 5 で説明しています。

図 5. UDB 実装



レジスタ

ステータス

ステータス レジスタは、PWM に設定された様々なステータスを示す定義ビットを含み、読み取り専用で使うレジスタです。このレジスタ値は、関数 PWM_ReadStatusRegister() の呼び出しによって利用できます。割り込み出力信号(割り込み)は、レジスタ内のマスクされたビット フィールドの 論理和で生成されます。PWM_SetInterruptMode() の関数呼び出しを使用してマスクを設定でき、割り込みを受け取ると、PWM_ReadStatusRegister() 関数呼び出しを用いてステータス レジスタを読み取ることにより、割り込みソースを取得できます。ステータス レジスタは「クリア-オン-リード 」レジスタで、関数 PWM_ReadStatusRegister() が呼び出されるまで割り込みソースは保持されます。PWM_ReadStatusRegister() API は、どの割り込みがイネーブルになり、割り込みの実際のソースが何であるかの正確なレポートを提供できるかの処理を行います。これらのビット フィールドは配置配線の間にステータス レジスタ内で移動している可能性があるため、すべての操作は、次のビット フィールド用に定義された名前を使用しなければなりません。

コンフィギュレーション エディタの **Interrupts** セクションで **None** のオプションを選択することで、ハードウェアからステータス レジスタを完全に削除することができます。このオプションが設定されている場合、API はステータス レジスタのアクセスをサポートしません。ステータス レジスタの API アクセスを持つデザインをビルドする場合、PWM_1_PWMUDB_sSTSReg_stsreg__STATUS_REG が未定義の識別子であるというエラーが発生することがあります。これは、API を削除して、コンフィギュレーション エディタの割り込みで **None** オプションのチェックを外すことにより、修正できます。

ステータス データはカウンタの入力クロックのエッジでレジスタに入れられ、「Mode = 1」として設定された全 ビットにカウンタのタイミング分解能を与えます。これらのビットはスティッキーで、ステータス レジスタの読み取り時に消去されます。「Mode = 0」と構成されている全ビットは透過的で、入力からステータス レジスタへ直接読み取られます。これらはスティッキーではなく、読み取り時にクリアされません。「Mode = 1」と構成されている全ビットは、以下のリストの定義内で(*)付きで示されます。

これらは、ステータス レジスタで定義されているいくつかのビット フィールドマスクです。どのビット フィールドでも、割り込みソースとして含めることができます。下記のように、#defines は作成されたヘッダファイル(.h)の中で利用可能です。

PWM_STATUS_TC *

最終カウント出力のステータス。最終カウント出力が HIGH のとき、このビットは HIGH になります。

PWM_STATUS_CMP1 *

ピリオド カウンタとの関係における pwm1 比較値のステータス。固定機能実装では、コンペア 出力が HIGH のとき、このビットは HIGH になります。UDB 実装では、このビットはコンペア出力の記録されたバージョンとともにアサートされます。つまり、コンペア出力が HIGH になった 2 クロック後にこのビットはアサートされます。

PWM_STATUS_CMP2 *

ピリオド カウンタとの関係における pwm2 比較値のステータス。固定機能実装では、コンペア 出力が HIGH のとき、このビットは HIGH になります。UDB 実装では、このビットはコンペア 出力の記録されたバージョンとともにアサートされます。つまり、コンペア出力が HIGH になった 2 クロック後にこのビットはアサートされます。

PWM_STATUS_KILL

出力キルのステータス。これが現在アクティブの場合、出力は HIGH になります。

PWM_STATUS_FIFOFULL

キャプチャ FIFO レベルのステータス。このビットは、FIFO が現在フルであることを示す FIFO レベルのリアルタイム ステータスです。ステータス レジスタのこのビットに「0」があるときは、FIFO はフルではないことを示しますが、FIFO に空であると示しているわけではありません。

コントロール

コントロール レジスタは、PWM の一般操作のコントロールを可能にします。このレジスタは、PWM_WriteControlRegister() 関数呼び出しで書き込み、PWM_ReadControlRegister() で読み取ります。コントロール レジスタの読み取りまたは書き込みを行うとき、ヘッダ (.h) ファイルに定義されているビット フィールド定義を使用しなければなりません。コントロール レジスタの #defines は次のとおりです。

PWM_CTRL_ENABLE

イネーブル ビットは PWM 動作のソフトウェアイネーブルをコントロールします。PWM は、ビルド時に定義された構成可能なイネーブル モードを持っています。**Enable Mode** パラメーターが **Hardware Only** に設定されている場合は、このビットには関数がありません。しかし、他のモードのいずれでも、PWM はこのビットが「high」に設定されるまで、減算されません。通常動作には、PWM の全操作でこのビットの「high」に設定とその維持が必要です。

PWM_CTRL_CMPMODE1_MASK

コンペア モード制御は、pwm1 出力用に期待されるコンペア出力動作を指定するために使用される 3 ビットのフィールドです。ビット フィールドはコントロール レジスタ内の 3 つの連続したビットです。このビット フィールド上の全ての動作は、利用可能なコンペア モードに伴う「 #defines」を使用します。これらを以下に示します。

- PWM_1_B_PWM_CM_LESSTHAN
- PWM_1_B_PWM_CM_LESSTHANOREQUAL
- PWM_1_B_PWM_CM_EQUAL
- PWM_1_B_PWM_CM_GREATERTHAN

- PWM_1_B_PWM_CM_GREATERTHANOREQUAL

このビット フィールドは、CompareMode1 パラメータで定義されたコンペア モードを使用した初期化で設定され、PWM_SetCompareMode() または PWM_SetCompareMode1() API の呼び出しによって修正できます。

PWM_CTRL_CMPMODE2_MASK

コンペア モード制御は、pwm2 出力用に期待されるコンペア出力動作を指定するために使用される 3 ビットのフィールドです。ビット フィールドはコントロール レジスタ内の 3 つの連続したビットです。このビット フィールド上の全ての動作は、利用可能なコンペア モードに伴う「 #defines」を使用します。これらを以下に示します。

- PWM_1_B_PWM_CM_LESSTHAN
- PWM_1_B_PWM_CM_LESSTHANOREQUAL
- PWM_1_B_PWM_CM_EQUAL
- PWM_1_B_PWM_CM_GREATERTHAN
- PWM_1_B_PWM_CM_GREATERTHANOREQUAL

このビット フィールドは、CompareMode2 パラメータで定義されたコンペア モードを使用した初期化で設定され、PWM_SetCompareMode2() API の呼び出しによって修正できます。

ピリオド(分解能に基づき、8 または 16 ビット)

このピリオド レジスタは、ユーザによって関数 PWM_WritePeriod() の呼び出しによって設定され、初期化時に **Period** パラメーターによって定義されるピリオド値を含みます。関数 PWM_ReadPeriod() は、現在のレジスタ値を見つけるために使用できます。ピリオド レジスタは、最終カウントに到達するまで PWM に何の影響も及ぼしません。最終カウントに到達すると、この値がピリオド カウンタにリロードされます。

Compare1/Compare2(分解能に基づき、8 または 16 ビット)

コンペア レジスタは、pwm または pwm1、および pwm2 出力の状態の特定に使用する比較値を含みます (**PWM モード**パラメータの設定による)。pwm/pwm1 と pwm2 出力は、コントロール レジスタで規定されるコンペア モードで、これらのレジスタがどのようにピリオドカウンタ値と比較されるかに基づきます。

ピリオド カウンタ(分解能に基づき、8 または 16 ビット)

ピリオド カウンタレジスタには、PWM の動作中ずっと、ピリオド カウンタ値が含まれています。基本動作の間、PWM がイネーブルのとき、このレジスタはクロック入力の各立ち上がりエッジで 1 ずつ減算されます。PWM_ReadCounter() 関数を呼び出して、このレジスタの内容をいつでも読みだすことができます。最終カウントに達すると、PWM_WritePeriod() の関数呼び出しを通じて、または **Period** パラメータの初期化時に、このレジスタはピリオド レジスタでユーザが定義するピリオド 値でリロードされます。

pwm、pwm1、pwm2 出力は、このレジスタが保持する値と、PWM_WriteCompare() の関数呼び出しを介して、または CompareValue パラメータを用いた初期化時に、コンペア レジスタで定義される値の間の関係に基づいています。

条件付きコンパイル情報

PWM API は、サポートしなければならない複数の構成を処理するために、いくつかの条件付きコンパイル定義が必要です。API は、選択された分解能と、固定機能ブロックまたは UDB ブロック、デッドバンド モード、キル モード、PWM モードから選択された実装を条件としてコンパイルしなければなりません。定義付けられた条件は、FixedFunction、Resolution、DeadBand、KillMode、PWMMode のパラメータに基づきます。API は、これらのパラメータを直接は使いませんが、下記の定義は使用します。

PWM_Resolution

分解能の定義は、ビルド時の「Resolution」値に割り当てられています。これは、情報に基づき、API 関数について正しいデータ幅タイプでコンパイルするために API 全体で使用されます。

PWM_UsingFixedFunction

固定機能定義の使用は、適切なレジスタ割り当てを実行するために多くはヘッダーファイルで使用されます。これが必要な理由は、PWM が UDB に実装されている場合は、固定機能ブロックで提供されるレジスタはこの場合に使用されているものと異なるためです。

PWM_DeadBandMode

デッドバンド モードの定義は関数 PWM_WriteDeadTime() と PWM_ReadDeadTime() API を条件付きでコンパイルするために使用します。

PWM_KillModeMinTime

キル モードの最低時間の定義は、PWM_WriteKillTime() と PWM_ReadKillTime() API で条件付きでコンパイルするために使用します。

PWM_KillMode

キル モードの定義は、**Kill Mode** が **Minimum Time** に設定されている場合、「Kill Mode Min Time(キル モード最小レジスタ)」用のレジスタ アクセスポイントを定義するために使用します。

PWM_PWMMode

PWM モードの定義は、モードでの使用に必要な正しい PWM_WriteCompare()と PWM_ReadCompare() API 関数を含めるために使用します。

PWM_PWMModeIsCenterAligned

PWM モードでは、中央位置設定定義はピリオド レジスタ アドレスを再定義するために使用されます。中央位置設定は、その他のモードで実装した場合とは異なり、ヘッダ ファイルでハンドルされた、異なるレジスタの使い方が必要となります。

PWM_DeadBandUsed

デッドバンドを使用する定義は、PWM_WriteDeadTime()と PWM_ReadDeadTime() API の条件付きコンパイルを制御します。

PWM_DeadBand2_4

デッドバンド 2～4 定義は、PWM_WriteDeadTime() と PWM_ReadDeadTime() API 内での実装の条件付きコンパイルを制御します。

PWM_UseStatus

もし設計が Verilog コード内や、ヘッダと C ファイルでステータス レジスタ定義のある API を条件付きコンパイルする必要のある場合などでは、ステータス定義を使って、ステータス レジスタを削除します。

PWM_UseControl

もし設計が Verilog コード内や、ヘッダと C ファイルでステータス レジスタ定義のある API を条件付きコンパイルする必要のある場合などでは、ステータス定義を使って、ステータス レジスタを削除します。

PWM_UseOneCompareMode

使用 1 比較モードは、1 つまたは 2 つの比較モード PWM モード機能に必要な API 呼び出しの内外で条件付きのコンパイルに使用されます。

PWM_MinimumKillTime

Kill Mode が **Minimum Kill Time** に設定されている場合、最小時間データパスにプログラムされる最小キルタイムを提供します。

PWM_EnableMode

特定のイネーブル モード用に提供される API を削除する条件付きコンパイル機能を提供します。

定数

幾つかの定数が、ステータス レジスタやコントロール レジスタとして定義されます。また一部の計数タイプも同様です。コントロールおよびステータス レジスタ用のほとんどの定数については、このデータシートですでに説明済です。但し、すべての操作を可能にするためには、ヘッダファイル中により多くの定数が必要です。各レジスタ定義には、

レジスタ データへのポインタまたはレジスタアドレスが必要です。コンパイラには複数のエンディアンがあるため、8 ビットより大きなレジスタ アクセスには CY_GET_REGX と CY_SET_REGX マクロの使用が必要です。これらのマクロでは、各レジスタについて _PTR 定義の使用が必要です。

コントロール レジスタとステータス レジスタのビットがフィッタエンジンによって配置とルーティングされる必要があり、これらのビットの配置を定義する定数が必要となります。ステータス レジスタとコントロール レジスタの各ビットには、関連している _SHIFT 値があり、これはレジスタ内でビットのオフセットを定義します。これらは、最終ビットマスクを _MASK 定義として定義するヘッダファイルで使用します。(拡張子_MASK は 1 ビットより大きなビット フィールドにのみつけられます。1 ビットの値では拡張子_MASK extension は付けられません。)

UDB 実装と比較すると、固定機能ブロックには制限があります。これは固定機能ブロックの設計で構成可能性が限定されているためです。

PSoC 3 用 DC 電気的特性と AC 電気的特性 (FF 実装)

以下の値は、期待される性能を示しており、初期特性データを基にしています。

PWM の DC 仕様

パラメータ	機能	条件	Min	Typ	Max	Units
	ブロックの消費電流	16 ビット PWM、各入力クロック周波数時	–	–	–	µA
	3 MHz		–	15	–	µA
	12 MHz		–	30	–	µA
	48 MHz		–	260	–	µA
	67 MHz		–	350	–	µA

PWM の AC 仕様

パラメータ	機能	条件	Min	Typ	Max	Units
	動作周波数		DC	–	67	MHz
	キャプチャ パルス幅(内部)		15	–	–	ns
	キャプチャ パルス幅(外部)		30	–	–	ns
	タイマ分解能		15	–	–	ns
	イネーブル パルス幅		15	–	–	ns
	イネーブル パルス幅(外部)		30	–	–	ns

パラメータ	機能	条件	Min	Typ	Max	Units
	リセット パルス幅		15	–	–	ns
	リセット パルス幅(外部)		30	–	–	ns

PSoC 5 用 DC 電気的特性と AC 電気的特性 (FF 実装)

以下の値は、期待される性能を示しており、初期特性データを基にしています。

PWM の DC 仕様

パラメータ	機能	条件	Min	Typ	Max	Units
	ブロックの消費電流	16 ビット PWM、各入力クロック周波数時	–	–	–	µA
	3 MHz		–	65	–	µA
	12 MHz		–	170	–	µA
	48 MHz		–	650	–	µA
	67 MHz		–	900	–	µA

PWM の AC 仕様

パラメータ	機能	条件	Min	Typ	Max	Units
	動作周波数		DC	–	67.01	MHz
	パルス幅		13	–	–	ns
	パルス幅(外部)		30	–	–	ns
	キル パルス幅		13	–	–	ns
	キル パルス幅 (外部)		30	–	–	ns
	イネーブル パルス幅		13	–	–	ns
	イネーブル パルス幅(外部)		30	–	–	ns
	リセット パルス幅		13	–	–	ns
	リセット パルス幅(外部)		30	–	–	ns

DC 電気的特性と AC 電気的特性(UDB 実装)

以下の値は、期待される性能を示しており、初期特性データを基にしています。

「公称ルーティングでの最大」タイミング特性

パラメータ	機能	構成 [4]	Min	Typ	Max	Units
f_{CLOCK}	コンポーネント クロック周波数	構成 1	–	–	50	MHz
		構成 2	–	–	45	MHz
		構成 3	–	–	40	MHz
		構成 4	–	–	50	MHz
		構成 5	–	–	50	MHz
		構成 6	–	–	40	MHz
		構成 7	–	–	35	MHz
		構成 8	–	–	30	MHz
		構成 9	–	–	35	MHz
		構成 10	–	–	35	MHz
t_{CLOCKH}	入力クロック HIGH 時 [5]	該当なし	–	0.5	–	$1/f_{CLOCK}$
t_{CLOCKL}	入力クロック LOW 時 [5]	該当なし	–	0.5	–	$1/f_{CLOCK}$

[4] PWM コンポーネントの構成

構成 1:
- 分解能:8 ビット
- PWM タイプ : 連続動作モードの 1 出力

構成 2:
- Resolution (分解能): 8 ビット
- PWM タイプ : 調整付き連続動作モードの 1 出力

構成 3:
- 分解能:8 ビット
- PWM タイプ : 中央位置設定出力

構成 4:
- Resolution (分解能): 8 ビット
- PWM タイプ : デッドバンドを伴う 1 出力

構成 5:
- 分解能:8 ビット
- PWM タイプ : キル モードを伴う連続動作モード。

構成 6 :
- 分解能:16 ビット
- PWM タイプ : 連続動作モードの 1 出力

構成 7 :
- Resolution (分解能): 16 ビット
- PWM タイプ : ディザー使用 連続動作モードの 1 出力

構成 8 :
- 分解能:16 ビット
- PWM タイプ : 中央位置設定出力

構成 9:
- Resolution (分解能): 16 ビット
- PWM タイプ : デッドバンドを伴う 1 出力

構成 10:
- 分解能:16 ビット
- PWM タイプ : キル モードを伴う連続動作モード。

[5] $t_{CY_clock} = 1/f_{CLOCK}$. これは 1 クロック周期のサイクルタイムです。

パラメータ	機能	構成 [4]	Min	Typ	Max	Units
Inputs (入力)						
t_{PD_ps}	入力パス遅延、同期ピン [6]	1	–	–	STA [7]	ns
t_{PD_ps}	入力パス遅延、ピンからsync [8]	2	–	–	8.5	ns
t_{PD_si}	同期出力の入力パスへの遅延（ルート）	1,2,3,4	–	–	STA [7]	ns
t_{I_clk}	clockX とクclockのアライメント	1,2,3,4	0	–	1	t_{CY_clock}
t_{PD_IE}	コンポーネント クロックへの入力パス遅延（エッジセンシティブ入力）	1,2	t_{PD_ps} + t_{SYNC} + t_{PD_si}	–	t_{PD_ps} + t_{SYNC} + t_{PD_si} + t_{I_clk}	ns
t_{PD_IE}	コンポーネント クロックへの入力パス遅延（エッジセンシティブ入力）	3,4	t_{SYNC} + t_{PD_si}	–	t_{SYNC} + t_{PD_si} + t_{I_clk}	ns
t_{IH}	入力 HIGH 時	1,2,3,4	t_{CY_clock}	–	–	ns
t_{IL}	入力 LOW 時	1,2,3,4	t_{CY_clock}	–	–	Ns

[6] t_{PD_ps} は、後述される通り静的タイミング結果内にあります。ここに挙げた数字は、多くの入力の STA 分析に基づく公称値です。

[7] t_{PD_ps} と t_{PD_si} はルートパスの遅延。ルーティングは動的なためこれらの値は変化することがあり、最大コンポーネント クロックと同期クロック周波数に直接の影響を及ぼします。これらの値は、静的タイミング分析 結果に記載されています。

[8] t_{PD_ps} 構成 2 で、デバイスのピン毎に定義された固定値。ここにリストした数字は、デバイスで利用可能なすべてのピンの公称値。

「すべてのルーティングでの最大」タイミング特性

パラメータ	機能	構成 [9]	Min	Typ	Max [10]	Units
f_{CLOCK}	コンポーネント クロック周波数	構成 1			25	MHz
		構成 2			23	MHz
		構成 3			20	MHz
		構成 4			25	MHz
		構成 5			25	MHz
		構成 6			20	MHz
		構成 7			18	MHz
		構成 8			15	MHz
		構成 9			18	MHz
		構成 10			18	MHz
t_{CLOCKH}	入力クロック HIGH 時 [11]	該当なし		0.5		$1/f_{clock}$
t_{CLOCKL}	入力クロック LOW 時[11]	該当なし		0.5		$1/f_{clock}$

[9] PWM コンポーネントの構成
- 構成 1:
  - 分解能:8 ビット
  - PWM タイプ : 連続動作モードの 1 出力
- 構成 2:
  - Resolution (分解能): 8 ビット
  - PWM タイプ : 調整付き連続動作モードの 1 出力
- 構成 3:
  - 分解能:8 ビット
  - PWM タイプ : 中央位置設定出力
- 構成 4:
  - Resolution (分解能): 8 ビット
  - PWM タイプ : デッドバンドを伴う 1 出力
- 構成 5:
  - 分解能:8 ビット
  - PWM タイプ : キル モードを伴う連続動作モード。
- 構成 6 :
  - 分解能:16 ビット
  - PWM タイプ : 連続動作モードの 1 出力
- 構成 7 :
  - Resolution (分解能): 16 ビット
  - PWM タイプ : 調整付き連続動作モードの 1 出力
- 構成 8 :
  - 分解能:16 ビット
  - PWM タイプ : 中央位置設定出力
- 構成 9:
  - Resolution (分解能): 16 ビット
  - PWM タイプ : デッドバンドを伴う 1 出力
- 構成 10:
  - 分解能:16 ビット
  - PWM タイプ : キル モードを伴う連続動作モード。

[10] ルーティング タイミング数の最大値は、公称ルーティング タイミング数を 2 の倍数で除算して得られます。お使いのコンポーネント インスタンスがこの速度と同じ、または遅い速度で動作している場合は、ミーティング タイミングはこのコンポーネントでは問題となりません。

[11] $t_{CY_clock} = 1/f_{CLOCK}$. これは 1 クロック周期のサイクルタイムです。

パラメータ	機能	構成 [9]	Min	Typ	Max [10]	Units
Inputs (入力)						
t_{PD_ps}	入力パス遅延、同期ピン [12]	1			STA [13]	ns
t_{PD_ps}	入力パス遅延、同期ピン [14]	2			8.5	ns
t_{PD_si}	同期出力からの入力パス遅延（ルート）	1,2,3,4			STA [13]	ns
t_{I_clk}	clockX とclockのアライメント	1,2,3,4	0		1	t_{CY_clock}
t_{PD_IE}	コンポーネント クロックへの入力パス遅延（エッジセンシティブ入力）	1,2	t_{PD_ps} + t_{SYNC} + t_{PD_si}		t_{PD_ps} + t_{SYNC} + t_{PD_si} + t_{I_clk}	ns
t_{PD_IE}	コンポーネント クロックへの入力パス遅延（エッジセンシティブ入力）	3,4	t_{SYNC} + t_{PD_si}		t_{SYNC} + t_{PD_si} + t_{I_clk}	ns
t_{IH}	入力 HIGH 時	1,2,3,4	t_{CY_clock}			ns
t_{IL}	入力 LOW 時	1,2,3,4	t_{CY_clock}			ns

特性データ用の STA 結果の使用方法

公称ルーティング最大値は、静的タイミング分析 (STA) を使って、複数のテスパスから収集されます。STA 結果を用いた場合、次の手法で設計の最大値を計算できます。

f_{clock} Maximum Component Clock Frequency (最大コンポーネント クロック周波数) が、外部クロックという名前のクロック サマリにタイミング結果として表示されます。下図は、*_timing.html* によるクロック制限の例を示しています。

-Clock Summary

Clock	Actual Freq	Max Freq	Violation
BUS_CLK	24.000 MHz	118.683 MHz	
clock	24.000 MHz	56.967 MHz	

[12] t_{PD_ps} は、後述される通り静的タイミング結果内にあります。ここに挙げた数字は、多くの入力の STA 分析に基づく公称値です。

[13] t_{PD_ps} と t_{PD_si} はルートパスの遅延。ルーティングは動的なためこれらの値は変化することがあり、最大コンポーネント クロックと同期クロック周波数に直接の影響を及ぼします。これらの値は、静的タイミング分析 結果に記載されています。

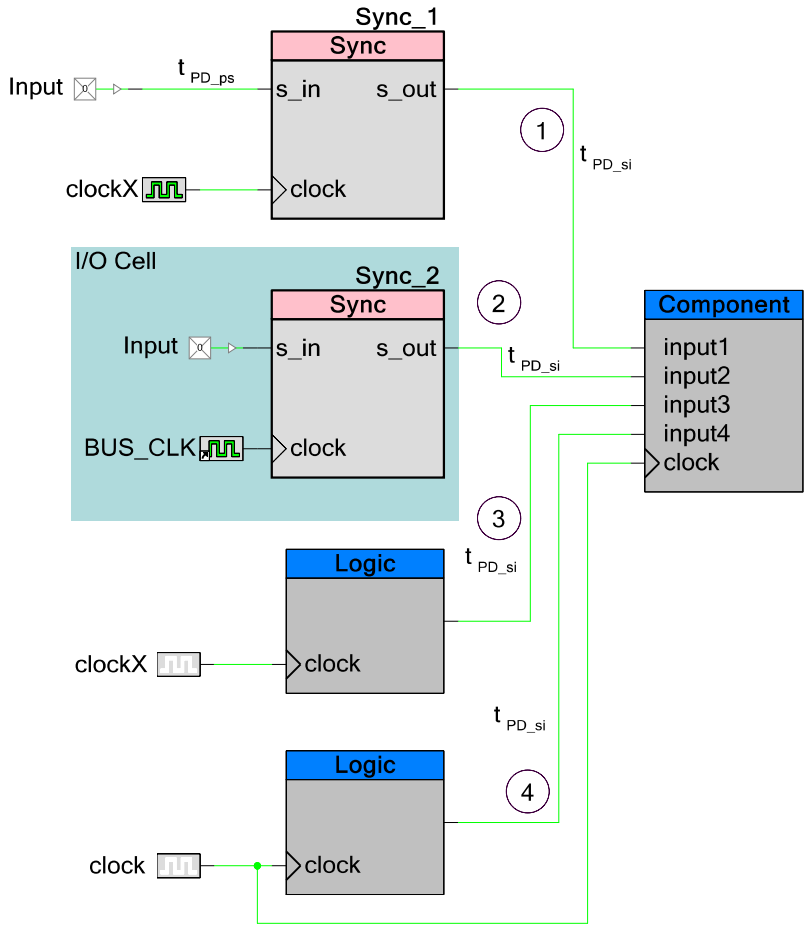
[14] t_{PD_ps} 構成 2 で、デバイスのピン毎に定義された固定値。ここに挙げた数字は、デバイスで利用可能なすべてのピンの公称値。

入力パス遅延とパルス幅

入力の機能分析を行う場合、すべての入力は、どのような構成になっているかに関わらず、図 6 に示す 4 つの構成の 1 つに該当します。

すべての入力は同期されていなければなりません。同期のメカニズムは、コンポーネントへの入力ソースによって異なります。システムの動作を完全に解釈するには、各入力でどの入力構成を設定したか、またシステムのクロック構成を理解する必要があります。このセクションでは、Static Timing Analysis (静的タイミング分析、STA) の結果を使用して、システムの特性分析を行う方法について説明します。

図 6.コンポーネントタイミング仕様のための入力構成



構成	コンポーネント クロック	同期クロック(周波数)	図
1	master_clock	master_clock	図 11
1	クロック	master_clock	図 9
1	クロック	clockX = clock [15]	図 7
1	クロック	clockX > clock	図 8
1	クロック	clockX < clock	図 10
2	master_clock	master_clock	図 11
2	クロック	master_clock	図 9
3	master_clock	master_clock	図 16
3	クロック	master_clock	図 14
3	クロック	clockX = clock [15]	図 12
3	クロック	clockX > clock	図 13
3	クロック	clockX < clock	図 15
4	master_clock	master_clock	図 16
4	クロック	クロック	図 12

1. 入力は、デバイスピンによって駆動され、内部で「sync」コンポーネントと同期します。このコンポーネントは、そのコンポーネントが使用するクロックと異なる内部クロックを使用してクロックを供給されます（すべての内部クロックは master_clock から派生）。

   このような方法で構成された入力の特性を分析する際は、clockX はコンポーネントのクロックより速い、同じ、遅い場合があります。またこれは、図 7、図 8、図 10、および 図 11 に示す特性分析パラメータを生成する master_clock と等しい場合もあります。

2. この入力は、デバイスピンによって駆動され、master_clock を使用してそのピンと同期されます。

   このような方法で構成された入力の特性を分析する際は、master_clock はコンポーネントのクロックより速いか同じ場合があります(遅いことはありません)。これにより、図 8 と図 11 に示す特性分析パラメータが生成されます。

[15] クロック周波数は同じですが、立ち上がりエッジのアライメントは保証されていません。

図 7.入力設定 1 および 2。Synchronizer Clock Frequency (シンクロナイザ クロック周波数) = Component Clock Frequency (コンポーネント クロック周波数) (クロックとクロック X のエッジアライメントは保証されません)

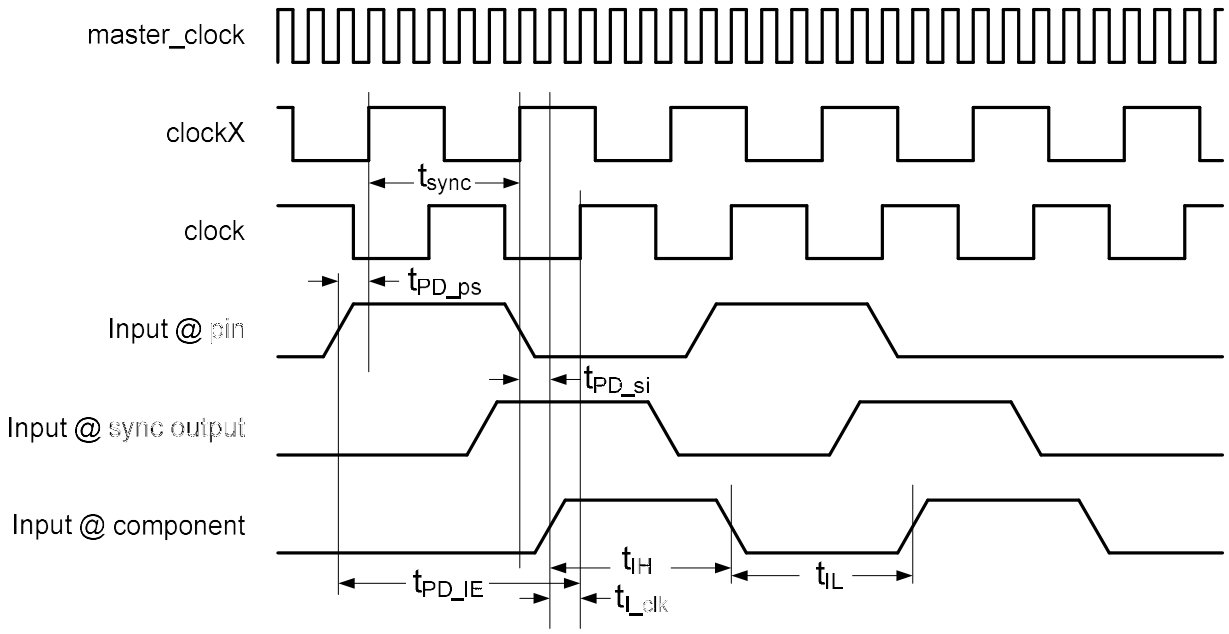


図 8.入力構成 1 および 2；シンクロナイザ クロック周波数 > コンポーネント クロック周波数

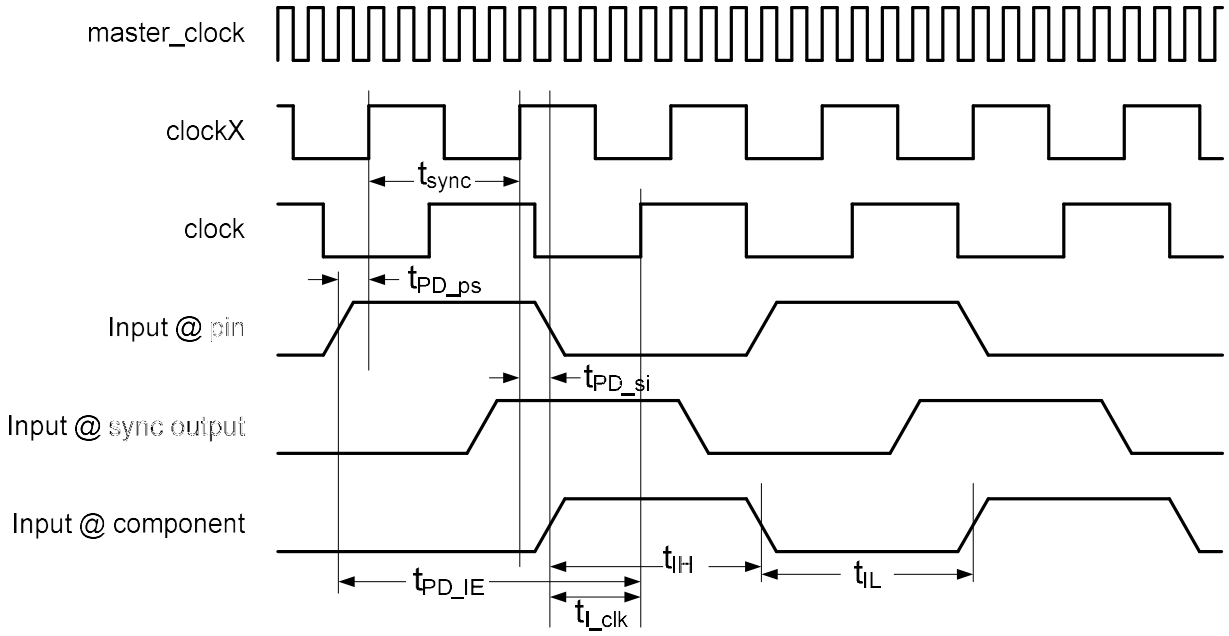


図 9.入力構成 1 と 2。[シンクロナイザ クロック周波数 =マスタ クロック] > コンポーネント クロック周波数。

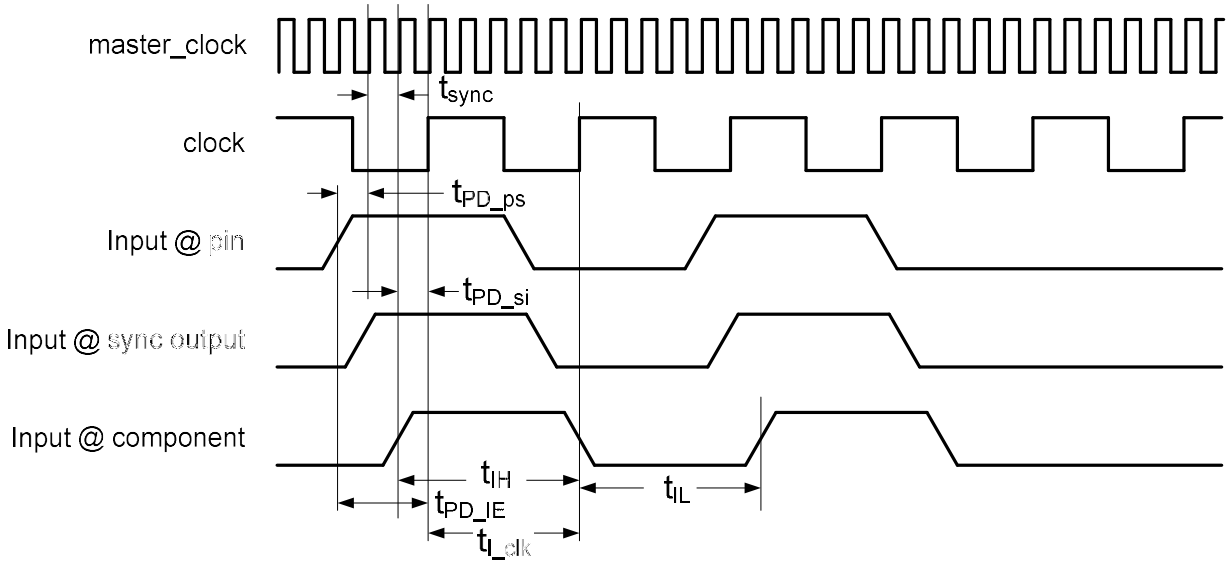


図 10.入力構成 1。シンクロナイザ クロック周波数 < コンポーネント クロック周波数

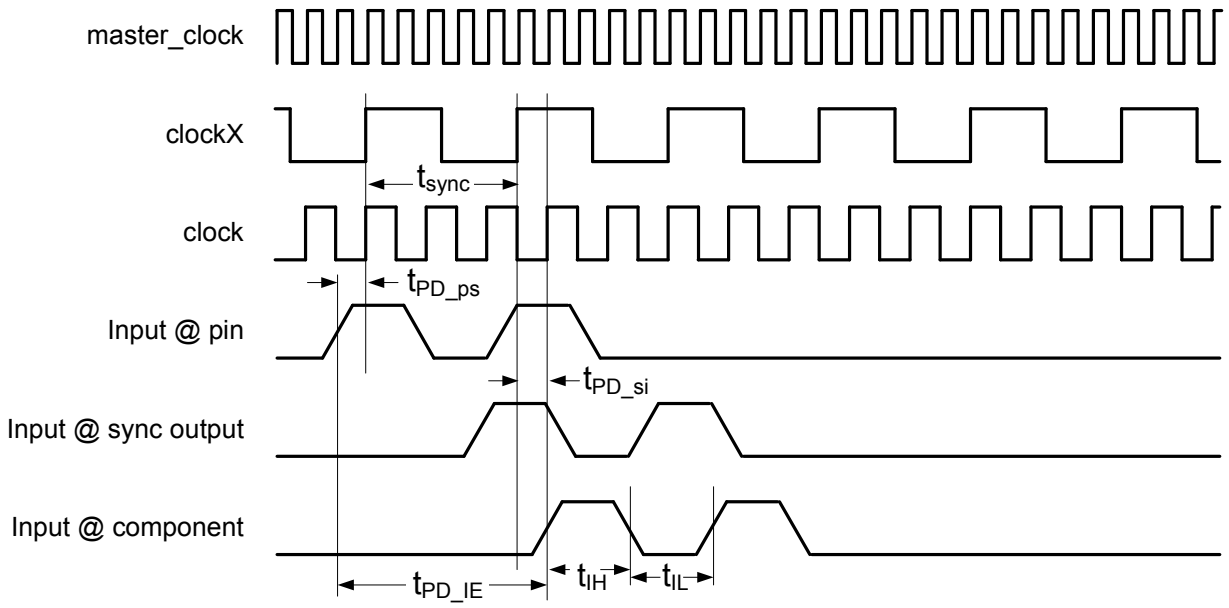
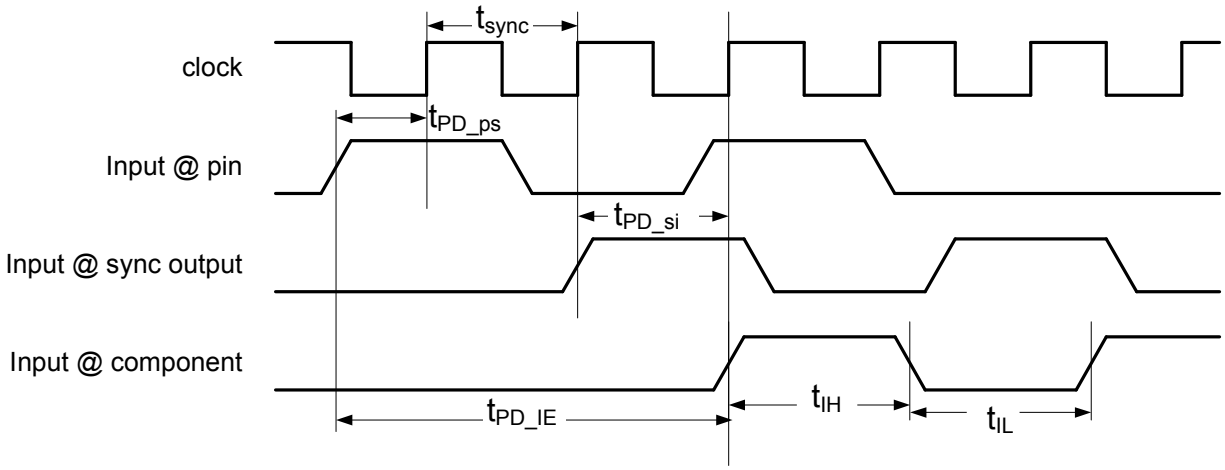


図 11.入力構成 1 と 2; シンクロナイザ クロック = コンポーネント クロック = master_clock



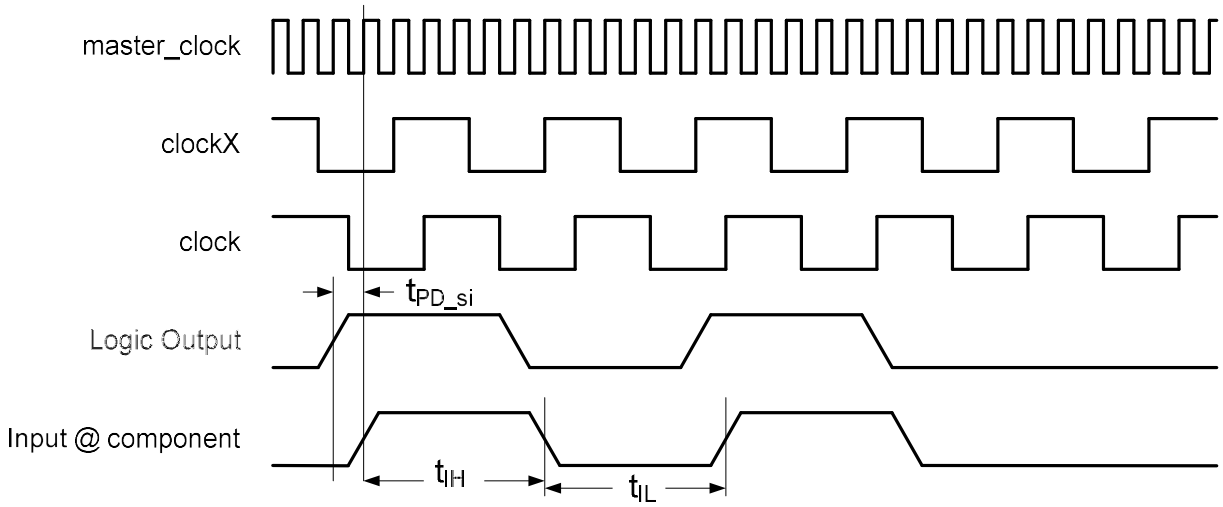
3. 入力は PSoC 内部のロジックに駆動されます。これはコンポーネントが使用するクロックとは異なるクロックをベースにして同期しています(すべての内蔵クロックは master_clock によって駆動しています)。

   このような方法で構成された入力の特性を分析する際は、シンクロナイザ クロックは、コンポーネントのクロックより速い、遅い、または同じ場合があります。これにより、図 12、図 13、および 図 15 に示す特性分析パラメーターが生成されます。

4. 入力は PSoC 内部のロジックに駆動されます。これはコンポーネントが使用するクロックと同じクロックをベースにして同期しています。

   このような方法で構成された入力の特性を分析する際は、シンクロナイザ クロックは、コンポーネントのクロックと同じです。これにより、図 16 に示す特性分析パラメータが生成されます。

図 12.入力設定 3 のみ;「Synchronizer Clock Frequency(シンクロナイザ クロック周波数)」= 「Component Clock Frequency(コンポーネント クロック周波数)」(クロックとクロック X のエッジアライメントは保証されません)



この数字は、静的タイミング分析がクロックに対して持つ理解を示します。デジタル クロック領域のすべてのクロックは master_clock と同期します。但し、同じ周波数を持つ 2 つのクロックは、立ち上がりエッジでアライメントされないことがあります。そのため、静的タイミング分析ツールは、クロックが同期しているエッジがどちらか判別できず、最低 1 master_clock サイクルを想定します。つまり、t_{PD_si} がシステムの master_clock に与える影響には制限があります。このパス遅延が長すぎると、master_clock 設定時間違反が表示されます。この場合、システムの同期クロックを変更するか、master_clock を遅い周波数で実行しなければなりません。

図 13.入力構成 3。シンクロナイザ クロック周波数 > コンポーネント クロック周波数

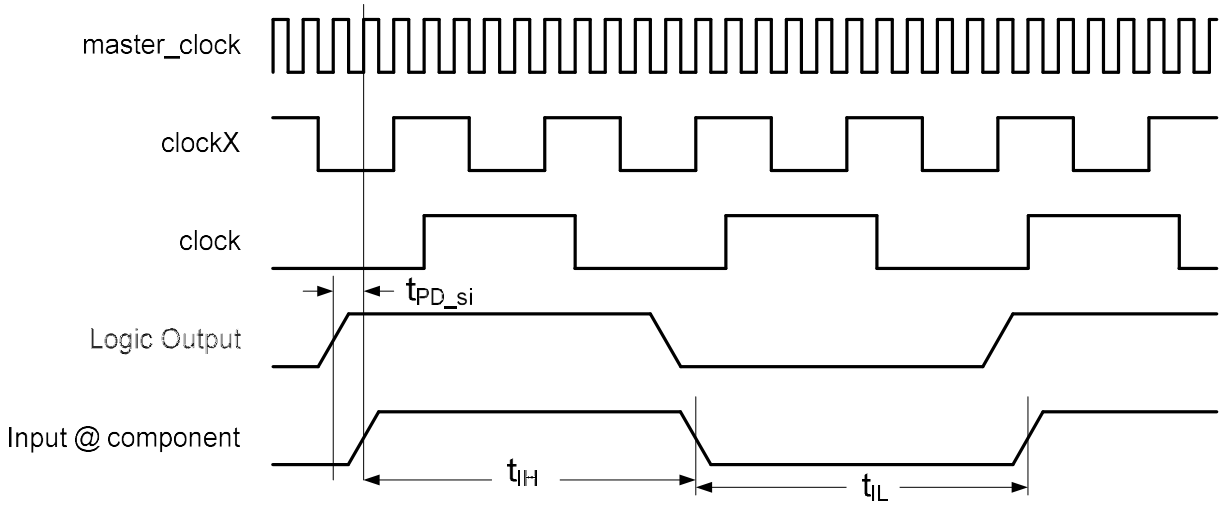


図 12 に示すのと類似した方法で、すべてのクロックは master_clock から派生します。STA は、この構成で 1master_clock サイクル分、master_clock における t_{PD_si} の制限を示します。このパスの遅延が長すぎると、master_clock セットアップ時間の違反が発生します。この場合、システムの同期クロックを変更するか、master_clock を遅い周波数で実行しなければなりません。

図 14.入力構成 3。シンクロナイザ クロック周波数 = master_clock > コンポーネント クロック周波数

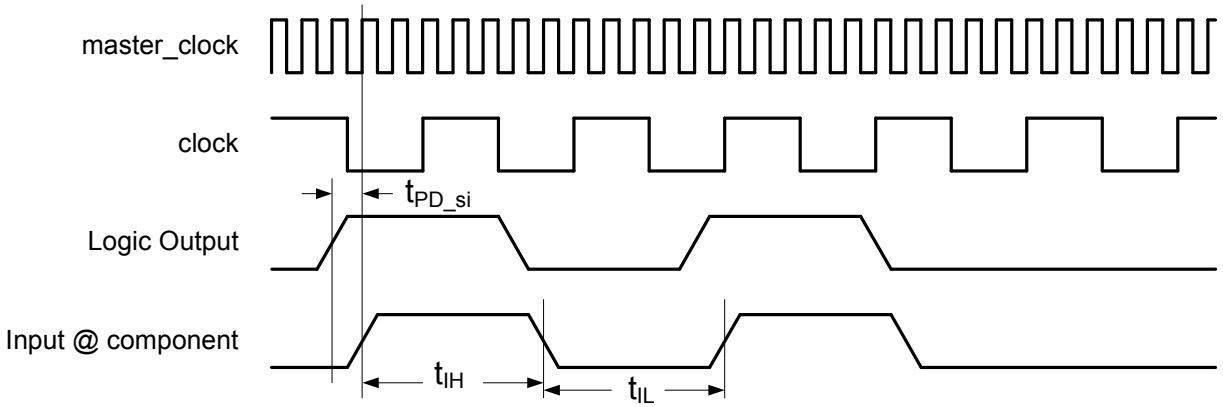


図 15.入力構成 3。シンクロナイザ クロック周波数 < コンポーネント クロック周波数

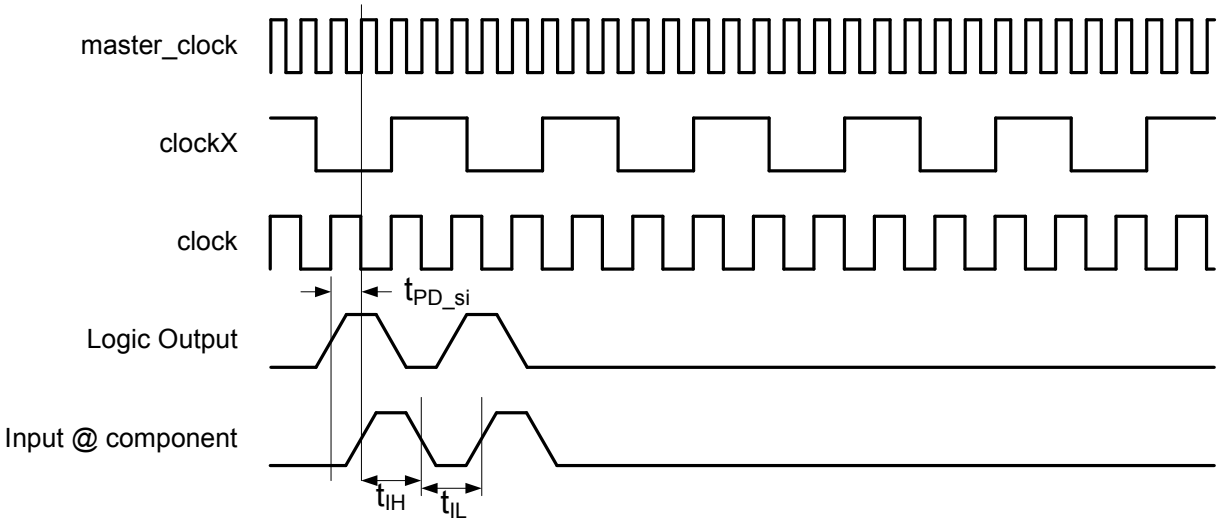
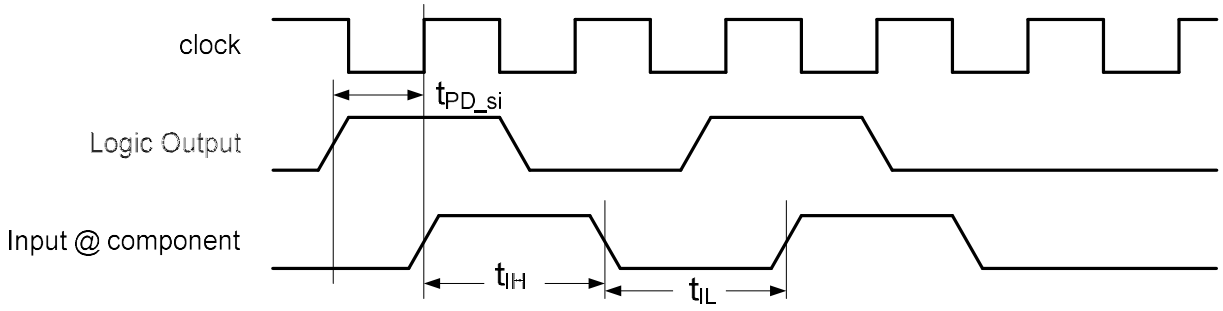


図 12 に示すのと類似した方法で、すべてのクロックは master_clock から派生します。STA は、この構成で 1master_clock サイクル分、master_clock における t_{PD_si} の制限を示します。このパスの遅延が長すぎると、master_clock セットアップ タイムの違反が発生します。この場合、システムの同期クロックを変更するか、master_clock を遅い周波数で実行しなければなりません。

図 16. 入力構成 4 のみ。Synchronizer Clock = Component Clock



このセクションでこれまで示した図の中で、実装を理解するために最も重要なパラメータは、f_{CLOCK} と t_{PD_IE} です。t_{PD_IE} は t_{PD_ps} と t_{SYNC} (構成 1 と 2 のみ)、t_{PD_si}、、t_{I_Clk} で定義されます。非常に重要な事は、t_{PD_si} が最大コンポーネント クロック周波数を定義するということです。t_{I_Clk} は STA の結果によるものではありませんが、t_{PD_IE} が登録された場合は重要となります。これはシンクロナイザとコンポーネント クロックの間のルートにあるマージンです。

t_{PD_ps} と t_{PD_si} は STA の結果に含まれています。

t_{PD_ps} は、*_timing.html* ファイルで定義されている入力設定時間を参照してください。この入力の Fan-out は複数ある可能性があり、これらのパスの最大値を評価する必要があります。

-Setup times

-Setup times to clock BUS_CLK

Start	Register	Clock	Delay (ns)
input1(0):iocell.pad_in	input1(0):iocell.ind	BUS_CLK	16.500

t_{PD_si} は、レジスタ～レジスタ時間に定義されています。*_timing.html* を使用するには、ネット名を知っていなければなりません。この入力の Fan-out は複数ある可能性があり、これらのパスの最大値を評価する必要があります。

-Register-to-register times

-Destination clock clock

Destination clock clock (Actual freq: 24.000 MHz)

+Source clock clock

-Source clock clock_1

Source clock clock_1 (Actual freq: 24.000 MHz)
Affected clock: BUS_CLK (Actual freq: 24.000 MHz)

Start	End	Period (ns)	Max Freq	Frequency	Violation
\Sync_1:genblk1[0]:INST\:synccell.syncq	\PWM_1:PWMUDB:runmode_enable\:macrocell.mc_d	7.843	127.508 MHz	24.000 MHz	

出力パス遅延

出力のパス遅延の特性分析を行う場合、STA 結果のどこでデータを見つけることができるかを知るために、出力の送信先を考えなければなりません。このコンポーネントでは、すべての出力がコンポーネント クロックに同期されています。出力は 2 つのカテゴリのうち、いずれかに該当します。出力は、デバイス内の別のコンポーネントへ送られるか、デバイス外のピンに進むかのどちらかです。前者の場合、上述のロジック～入力の説明に記載されているレジスタ～レジスタ時間を見ます(ソースクロックはコンポーネント クロックです)。後者の場合、*_timing.html* STA 結果のクロック～出力時間を見ます。

コンポーネントの更新履歴

ここでは、過去のバージョからコンポーネントに加えられた主な変更を示します。

バージョン	変更内容	変更理由 / 影響
2.10	カスタマイザ関連の更新	GUI 関連の小規模な問題の修正。
	PWM_RestoreConfig() APIを更新	低電源モードからウェイクアップした後で、問題を割り込みトリガで修正する方法。
	PWM_Stop() and PWM_Enable() APIを更新	FF PWM の他のアクティブパワーモードをイネーブルにする方法。
	PWM_SaveConfig() and PWM_RestoreConfig()updates を更新	ウェイクアップした後の PWM ピリオド レジスタの回復方法。
	データシートに PSoC 5 FF DC および AC の特性を追加	
2.0.a	データシート更新	
2.0	入力の同期	固定機能実装で、ブロック入力の地点ですべての入力を同期。
	関数PWM_GetInterruptSource() をマクロに変更	関数 PWM_GetInterruptSource() は関数 PWM_ReadStatusRegister() と全く同じ実装。コード スペースを節約するため、これを関数 PWM_ReadStatusRegister() のマクロ代替に変換。
	出力がコンポーネント クロックに登録されるように変更	コンポーネントの出力上のグリッチを避けるために、すべての出力が同期されることが必要でした。これは、リソース使用量を抑えるためにできる限りデータパスの内部で行われます。
	補助コントロール レジスタに書き込む際の、実装された重要領域。	補助コントロール レジスタに書き込む際は、関数 CyEnterCriticalSection と CyExitCriticalSections を使用。他のプロセススレッドによる変更を避けるため。
	データシートに特性データを追加	
	データシートのマイナーな編集と更新	

Copyright © 2005-2012 Cypress Semiconductor Corporation 本文書に記載される情報は、予告なく変更される場合があります。Cypress Semiconductor Corporationは、サイプレス製品に組み込まれた回路以外のいかなる回路を使用することに対しても一切の責任を負いません。特許又はその他の権限下で、ライセンスを譲渡又は暗示することもありません。サイプレス製品は、サイプレスとの書面による合意に基づくものでない限り、医療、生命維持、救命、重要な管理、又は安全の用途のために仕様することを保証するものではなく、また使用することを意図したものでもありません。さらにサイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことを合理的に予想される、生命維持システムの重要なコンポーネンツとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。

PSoC Designer™及びProgrammable System-on-Chip™は、Cypress Semiconductor Corp.の商標、PSoC®は同社の登録商標です。本文書で言及するその他全ての商標又は登録商標は各社の所有物です。

全てのソースコード(ソフトウェア及び/又はファームウェア)はCypress Semiconductor Corporation (以下「サイプレス」)が所有し、全世界(米国及びその他の国)の特許権保護、米国の著作権法並びに国際協定の条項により保護され、かつそれらに従います。サイプレスが本書面によるライセンシーに付与するライセンスは、個人的、非独占的かつ譲渡不能のライセンスであって、適用される契約で指定されたサイプレスの集積回路と併用されるライセンシーの製品のみをサポートするカスタムソフトウェア及び/又はカスタムファームウェアを作成する目的に限って、サイプレスのソースコードの派生著作物を複製、使用、変更、そして作成するためのライセンス、並びにサイプレスのソースコード及び派生著作物をコンパイルするためのライセンスです。上記で指定された場合を除き、サイプレスの書面による明示的な許可なくして本ソースコードを複製、変更、変換、コンパイル、又は表示することは全て禁止されます。

免責条項:サイプレスは、明示的又は黙示的を問わず、本資料に関するいかなる種類の保証も行いません。これには、商品性又は特定目的への適合性の黙示的な保証が含まれますが、これに限定されません。サイプレスは、本文書に記載される資料に対して今後予告なく変更を加える権利を留保します。サイプレスは、本文書に記載されるいかなる製品又は回路を適用又は使用したことによって生ずるいかなる責任も負いません。サイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことが合理的に予想される生命維持システムの重要なコンポーネンツとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。

ソフトウェアの使用は、適用されるサイプレスソフトウェアライセンス契約によって制限され、かつ制約される場合があります。