16-Bit PWM（パルス幅変調器）のデータシート PWM16 V 2.5

Copyright © 2000-2011 Cypress Semiconductor Corporation. All Rights Reserved.

リソース	PSoC® ブロック数			API に必要な容量（バイト）		外部 I/O に必要なピン数
	デジタル・ブロック	アナログ・連続時間・ブロック	アナログ・スイッチト・キャパシタ・ブロック	フラッシュ ROM	RAM	
CY8C29/27/24/22/21xxx、CY8C23x33、CY7C64215/603xx、CYWUSB6953、CY8CLED02/04/08/16、CY8CLED0xD、CY8CLED0xG、CY8CNP102、CY8CTST110、CY8CTMG110、CY8CTST120、CY8CTMG120、CY8CTMA120、CY8C21x45、CY8C22x45、CY8CTMA30xx、CY8C28x45、CY8CPLC20、CY8CLED16P01、CY8C28xxx						
16-bit	2	0	0	89	0	1
CY8C26/25xxx						
16-bit	2	0	0	138	0	1

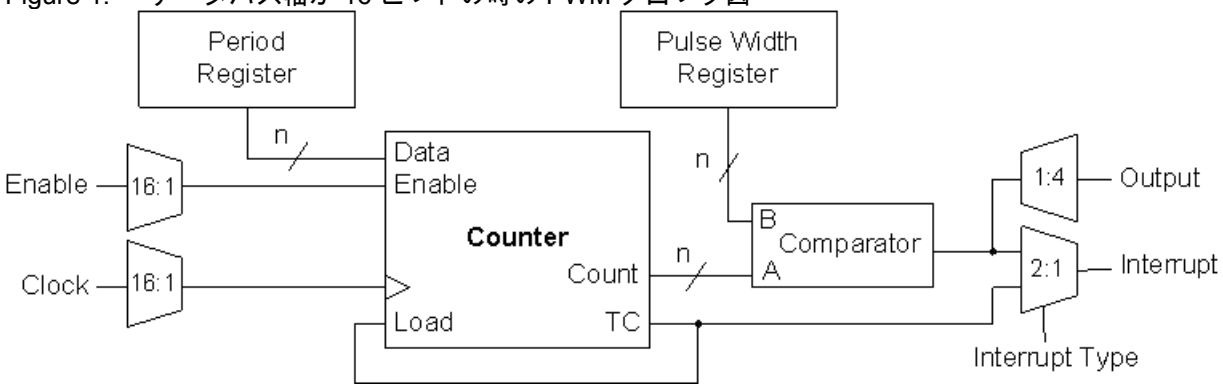
このユーザ モジュールを使用する機能例として完全に構成されたプロジェクトについては、以下を参照してください。www.cypress.com/psocexampleprojects

特徴と概要

- 16-bit 汎用 PWM（パルス幅変調器）は、2 つの PSoC ブロックを使用します。
- 48 MHz 以内のクロック源を使用できます。
- 各パルス周期で周期が自動的にリロードされます。
- パルス幅は、プログラム可能です。
- 連続カウント動作を有効 / 無効にする入力があります。
- 割り込みオプションでは、比較出力の立ち上がりエッジまたは最終カウント状態を選べます。

16-bit PWM ユーザ モジュールは、周期とパルス幅がプログラム可能なパルス幅変調器です。クロック信号とイネーブル信号は、複数の信号源から選択できます。出力信号は、一つのピン、または、グローバルな出力バスの一つに配線され、他のユーザ モジュールにより内部的に使用されます。割り込み条件はプログラム可能で、比較出力信号の立ち上がりエッジ、または、カウンタが最終カウントに到達した場合に発行されます。

Figure 1.　データパス幅が 16 ビットの時の PWM ブロック図



機能説明

PWM ユーザ モジュールは、2 つのデジタル PSoC ブロックを使用し、それぞれが 総分解能の 8 ビットずつを受け持ちます。16-bit パルス幅変調器を形成するため、2 つの連続したブロックが割り当てられるので、内部キャリー、最終カウント、コンペアの各信号が同期して接続されます。この接続により、個々の Count、Period、Compare のレジスタ (それぞれデータ レジスタ DR0、DR1、DR2) を連結し、必要とされる 16-bit の分解能を提供します。

PWM API は、C およびアセンブリから呼び出される関数を提供し、カウンタの動作を停止および開始し、さまざまなデータ レジスタの読み書きを行います。データ レジスタ値は、デバイス エディタを使用して設定する事もできます。カウント動作が開始されると、クロックの立ち上がりエッジで Hi- アクティブなイネーブル入力信号がアサートされていれば、Count レジスタがデクリメントされます。Count レジスタがゼロに達して、最終カウント状態になると、続くクロックの立ち上がりエッジで、Count レジスタに Period レジスタの値がリロードされます。

Period レジスタは、いつでも新しい値に変更することができます。PWM が停止している時、Period レジスタにある値を書き込むと、同時に Count レジスタにも同じ値が書き込まれます。PWM がカウント動作を行っている時、Period レジスタへ値を書き込んでも、Count レジスタは影響を受けず、最終カウントに続いて次のリロードが発生するときに、新しい Period 値が Count レジスタに書き込まれます。最終カウント条件は、カウンタ値がゼロになった時に満たされるため、カウント動作と出力信号の周期は、Period レジスタに保存されている値より 1 だけ 大きくなります。以下の式は、PWM の出力と入力クロックおよび Period レジスタの値の関係を表しています。

TOUT = (PeriodValue+1)/FCLOCK

$$FOUT = FCLOCK/(PeriodValue+1) \qquad \textbf{Equation 1}$$

ここで、*F*OUT は PWM の出力周波数、*T*OUT は PWM の出力周期、*F*CLOCK は入力クロックの周波数、*PeriodValue* は Period レジスタに書き込まれた値です。

PWM が停止すると、出力を Low にします。PWM の動作中は、コンパレータが、出力信号のデューティサイクルを制御します。クロック周期ごとに、このコンパレータは、PulseWidth レジスタと Count レジスタの値を比較します。比較条件は、「未満」または 「以下」で、これらはデバイス エディタを使って選択するこができます。比較が行われた周期に続くクロックの立ち上がりエッジで、PWM は比較結果に従った正論理の信号をアサートします。PulseWidth の値と周期の値の比が、出力波形のデューティサイクルを設定します。デューティ比は、以下の式で計算できます。

PulseWidthValue < PeriodValue の時

Equation 2

$$DutyCycle = \begin{cases} \frac{PulseWidthValue}{PeriodValue + 1}, \text{For } Less\ Than \text{ comparison} \\ \frac{PulseWidthValue + 1}{PeriodValue + 1}, \text{For } Less\ Than\ Or\ Equal\ To \text{ comparison} \end{cases}$$

PulseWidthValue >= PeriodValue の時

DutyCycle = 100%

次の表では、Period、PulseWidth、比較条件の設定を特定の値にしたときの出力信号状態をあらわしています。

Table 1.　特定の出力信号状態

Period レジスタの値	比較条件	PulseWidth レジスタの値	パルスが HIGH の時間と周期の比
0	影響しない	> 0	1.0
0	≤	0	1.0
0	<	0	0.0
> 0	≤	0	1/(Peiord + 1)
> 0	<	0	0.0
Period = PulseWidth	≤	Period = PulseWidth	1.0
Period = PulseWidth	<	Period = PulseWidth	Period/(Period + 1)
PulseWidthValue > Period	影響しない	PulseWidthValue > Period	1.0

PulseWidth レジスタの値を設定するには、デバイス エディタを使うか、あるいは 動作時に API を使います。Period レジスタは Count レジスタが最終カウントに達するまでバッファリングされますが、PulseWidth レジスタは、バッファリングされません。このため、PulseWidth レジスタを変更すると、最終カウント状態にならなくても、次のクロックで比較出力に影響します。この動作により、出力波形の一周期内に複数のパルスが発生する可能性があります。

CY8C29/27/24/22/21xxx, CY8C23x33, CY7C64215/603xx, CYWUSB6953, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8C22x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx デバイス ファミリでは、PWM ユーザ モジュールは、補助出力として、最終カウント信号を出力します。この正論理信号は、最終カウントに続くクロックの立ち上がりエッジ、すなわち Count レジスタに Period レジスタの値がリロードされるタイミングでアサートされます。

割り込み条件は、プログラム可能で、最終カウント条件、または、コンパレータの出力がアサートされる時から選べます。コンパレータの出力は、出力信号の立ち上がりエッジで割り込みを発生させ、最終カウント条件は、出力信号の立ち下がりエッジの半クロック前に割り込みを発生します。このオプションは、デバイス エディタを使って設定されます。割り込みの許可および禁止は、動作時に Counter API を使って設定されます。全体の割り込みは、カウンタの割り込みが発生する前に許可する必要があります。

[+] Feedback

PulseWidth レジスタの変更には注意が必要です。これは、現在のカウント値と PulseWidth レジスタの値との関係により PWM の出力状態が決定されるためです。出力信号が早めに Low になったり、グリッチが発生するのを防ぐため、PulseWidth レジスタは、割り込みによって最終カウント状態が検出された後で変更してください。

デューティサイクルをより短い間隔で更新する必要があるアプリケーションでは、PWM の出力をある端子に迂回させ、その状態をポーリングすることができます。PWM 出力が High から Low へ移行したのが検出されたら、PulseWidth レジスタを更新することができます。PulseWidth レジスタの更新によりにより、コンパレータ出力がアサートされたら、PWM 出力は次のクロックで High にアサートされます。

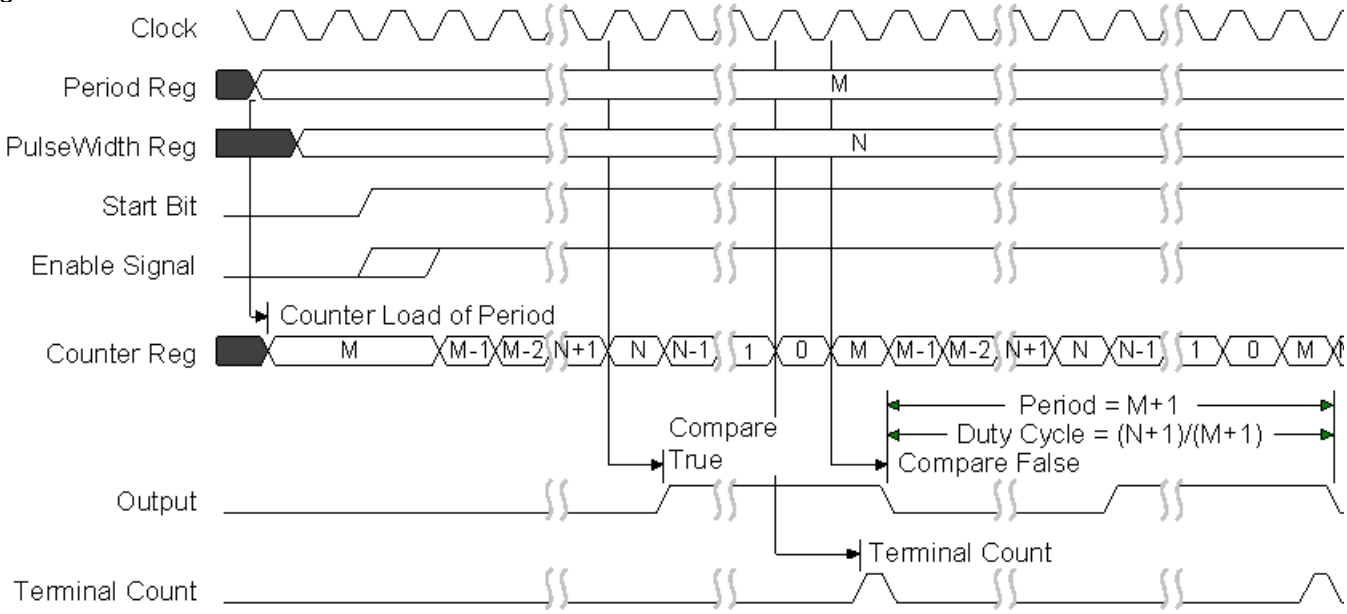
Count レジスタの値を読み出す時は、注意する必要があります。Count レジスタを読み出すと、その内容が PulseWidth レジスタにラッチされます。これにより、出力デューティサイクルが変化します。

カウンタ動作時に Count レジスタを読み取る必要がある場合には、ReadCounter() API 関数を呼び出すことができます。この関数は、クロックを一時的に無効にし、PulseWidth レジスタの内容を保存し、Count レジスタを読み出し、PulseWidth レジスタを読み出し、PulseWidth レジスタを復元し、クロックを復元します。この動作の結果、考えられる副作用については、「アプリケーション プログラミング インターフェイス」に記載されている ReadCounter() 関数の説明を参照してください。

タイミング

PWM の動作は、オンとオフを切り替えたり、デバイスのグローバル バス機能により PWM と結線される外部ピンからクロックを導入したりできます。

Figure 2. PWM タイミング図



DC および AC の電気的特徴

Table 2. PWM の DC および AC の電気的特徴

パラメータ	標準値	上限	単位	条件および注意
最大 PWM 出力周波数 $FOutput_{max}$	--	24[1]	MHz	電源電圧 5.0V 入力クロック 48 MHz
	--	12[2]	MHz	電源電圧 3.3V 入力クロック 24 MHz

電気的特性に関する注意

1. 出力が、グローバル バスを介して配線される場合は、周波数は、最大 12 MHz に制限されます。
2. PSoC ブロックへ供給可能なクロックの最高周波数は、電源電圧 3.3V の時に 24 MHz です。

配置

PWM は、分解能 8 ビットごとに、1 つのデジタル PSoC ブロックを消費します。これらのブロックは、デバイス エディタにより連続したブロックに配置され、最下位バイト (LSB) から最上位バイト (MSB) までブロック番号が増加していきます。各ブロックには、配置中または配置後、デバイス エディタで表示されるシンボリック名が与えられます。API は、ユーザが割り当てたインスタンス名とブロック名に対して、すべてのレジスタ名を関連付け、API が提供する include ファイルを通して PWM レジスタへの直接アクセスを提供します。さまざまなビット幅によって使用されるブロック名を、以下の表に示します。

Table 3. PWM のシンボリック PSoC ブロック名

PSoC ブロック番号	16-Bit PWM
1	PWM16_LSB
2	PWM16_MSB

パラメータおよびリソース

Clock (クロック)

Clock パラメータは、16 のクロック源のひとつを選択します。これらのクロック源には、48 MHz オシレータ (5.0V 動作のみ)、24 MHz システム クロック、他の PSoC ブロック、グローバル入出力を通して配線される外部入力を分周して得られるより低い周波数 (V1、V2 および V3) が含まれます。外部デジタル クロックをブロックに使用する場合は、最高の精度およびスリープ動作を得るため、ROW の入力同期を切るべきです。

Enable (イネーブル)

Enable パラメータは、16 のソースのいずれかから選択されます。High 入力が連続カウントを有効にするのに対して、Low 入力はカウンタをリセットすることなくカウント動作を無効にします。

CompareOut (比較出力)

比較出力は、割り込み動作を妨害することなく無効にしたり、ROW 出力バスに接続することができます。比較出力は、このパラメータの設定にかかわらず、次に高いデジタル PSoC ブロックへの入力、およびアナログ コラムのクロック選択マルチプレクサの入力として常に利用できます。このパラメータは、PSoC デバイスファミリ CY8C29/27/24/22/21xxx, CY8C23x33,

CY7C64215/603xx, CYWUSB6953, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8C22x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx のみに現れます。

TerminalCountOut (最終カウント出力)

最終カウント出力は、補助 カウンタ出力です。このパラメータを使うと、最終カウント出力を無効にするか、いずれかの ROW 出力バスに接続することができます。このパラメータは、PSoC デバイスファミリ CY8C29/27/24/22/21xxx, CY8C23x33, CY7C64215/603xx, CYWUSB6953, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8C22x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx のみに現れます。

Period (周期)

このパラメータは、カウンタの周期を設定します。PWM8 では、ゼロと 255 の間の値を設定できます。PWM16 では、ゼロと 2^{16}-1 の間の値を設定できます。周期は、Period レジスタに設定されます。PWM16 の有効な出力波形周期は、周期カウント + 1 です。周期値は、API を使って変更することもできます。

PulseWidth (パルス幅)

PWM 出力のパルス幅を設定します。パルス幅に設定できるのは、ゼロと周期値の間の値です。パルス幅は、API を使って変更することもできます。

InterruptType (割り込みタイプ)

このパラメータは、割り込み要因の種類を設定します。PWM 出力の立ち上がりエッジ、または、Count レジスタの最終カウントで割り込みが発生するように、設定することができます。別のレジスタで、割り込みを個別に有効にします。

CompareType (比較タイプ)

このパラメータは、「未満」または 「以下」の比較タイプを設定します。

ClockSync (クロック同期)

PSoC デバイスでは、システム クロックに加えて、デジタル ブロックがクロック源を提供することができます。デジタル クロック源は、リップル形で生成することも可能です。こうすることで、デジタルクロック源に、システム クロックに対するスキューを取り入れます。これらのスキューは、PSoC デバイスファミリ CY8C29/27/24/22/21xxx, CY8C23x33, CY7C64215/603xx, CYWUSB6953, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8C22x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx でより重要です。なぜなら、さまざまなデータパス、特に、システム バスに適用される最適化に使われるからです。このパラメータは、クロック スキューを制御するために使用され、PSoC ブロック レジスタ値を読み書きする場合の、正しい動作を保証します。このパラメータの適切な値は、以下の表から決定してください。

ClockSync 値	用途
Sync to SysClk (SysClk への同期)	24 MHz (SysClk) から生成されたあらゆる 24 MHz 未満の入力クロック源に対しては、この設定を使用します。たとえば、VC1、VC2、VC3 (VC3 が SysClk によって駆動される場合)、32KHz、SysClk ベースのクロックで動作するデジタル PSoC ブロックがあります。外部で生成されたクロック源に対しても、この値を使用して適切に同期してください。
Sync to SysClk*2 (SysClk*2 への同期)	48 MHz (SysClk の二倍) から生成されたあらゆる 48 MHz 未満の入力クロック源に対しては、この設定を使用します。
Use SysClk Direct (SysClk を直接使用する)	24 MHz (SysClk/1) クロックが求められる場合、この設定を使用します。この設定では、実際にはクロックを同期させず、システム クロック自身へのスキューの少ないアクセスを提供します。この設定を選択すると、上述の Clock パラメータの設定を上書きします。すべての分周器を組み合わせた最終出力が 24MHz 出力を生成するような、VC1、VC2、VC3、またはデジタル ブロックの代わりに使用しなくてはなりません。
Unsynchronized (同期せず)	48 MHz (SysClk*2) 入力が選択される場合に使用します。 同期されていない入力が求められる場合、この設定を使用します。一般に、カウンタが割り込みを発生させる目的だけに使用される場合にのみ、この設定を使用することを推奨します。この設定は、スリープ中にアクティブ状態に維持されるブロックで必要です。

InvertEnable （有効入力の反転）

このパラメータは、イネーブル入力信号の極性を決定します。「Normal (標準)」を選択すると、イネーブル入力が正論理になります。「Invert (反転)」を選択すると、イネーブル入力が負論理として解釈されます。InvertEnable は、PSoC デバイスの CY8C29/27/24/22/21xxx, CY8C23x33, CY7C64215/603xx, CYWUSB6953, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8C22x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx ファミリにのみ適用されます。

割り込み生成の制御

PSoC Designer の "Enable interrupt generation control" （割り込み生成制御を有効にする）チェックボックスがチェックされている時に、さらに二つのパラメータが設定できます。これらのパラメータは、メニューの "Project>Settings>Chip Editor" から利用できます。「割り込み生成の制御」は、オーバーレイ全体で複数のユーザ モジュールにより共有される割り込みとともに、複数のオーバーレイが使用される場合に重要です。

- 割り込み API
- IntDispatchMode

InterruptAPI

InterruptAPI パラメータを使うと、ユーザ モジュールの割り込みハンドラと割り込みベクタ テーブル エントリの状況に応じた生成が可能になります。「Enable (有効)」を選択すると、割り込みハンドラと割り込みベクタ テーブル エントリが生成されます。「Disable (無効)」を選択すると、割り込みハンドラと割り込みベクタ テーブル エントリが生成されません。ある単一のブロックのリソースが異なるオーバレイによって使用されるような、複数のオーバレイをもつプロジェクトの場合には、割り込み API を生成するかしないかを適切に選択することが、特に求められます。必要に応じて、割り込み API の生成のみを選択すると、割り込みディスパッチ コードが生成されなくなり、オーバーヘッドを軽減できます。

IntDispatchMode

IntDispatchMode パラメータは、同一ブロック内の異なるオーバーレイに存在する複数のユーザ モジュールによって共有される割り込みで、割り込みリクエストをどのように取り扱うかを指定します。「ActiveStatus」を選択すると、共有される割り込みリクエストに応答する前に、どちらのオーバーレイがアクティブであるかをファームウェアにテストさせます。このテストは、共有割り込みがリクエストされるたびに実行されます。これはレイテンシーを生むほか、共有割り込みリクエストに対応する非決定性のプロシージャを生成しますが、RAM は必要としません。
「OffsetPreCalc」を選択すると、オーバーレイが最初にロードされるときだけ、共有割り込みリクエストのソースをファームウェアに計算させます。この計算は、割り込みレイテンシーを低減し、共有割り込みリクエストに応答する決定性のプロシージャを生成しますが、RAM のバイト消費が発生します。

アプリケーション プログラミング インタフェース (API)

設計者が高いレベルでモジュールを取り扱うことができるようにユーザ モジュールの一部としてアプリケーション プログラミング インタフェース (API) ルーチンを提供します。このセクションでは、各機能に対するインタフェースを 「include」ファイルによって提供される関連定数とともに示します。

Note この API では、すべてのユーザ モジュール API の場合と同じように、API 関数を呼び出すことで A と X レジスタの値が変更されることがあります。関数を呼び出した後で A および X レジスタの値が必要になるのであれば、関数を呼び出す側の責任において、これらのレジスタの値を関数呼び出し前に退避させてください。この 「registers are volatile」(レジスタは揮発性である) というポリシーは、効率上の理由から選択されて、PSoC Designer のバージョン 1.0 より採用されています。C コンパイラは、このポリシーを自動的に適用しています。アセンブラ言語のプログラマも、コードがこのポリシーを守っていることを保証する必要があります。一部のユーザ モジュール API 関数では A と X が変更されないかもしれませんが、将来も変更されないという保証はありません。

設計者が高いレベルでモジュールを取り扱うことができるようにユーザ モジュールの一部としてアプリケーション プログラミング インタフェース (API) ルーチンを提供します。以下に、PWM16 で提供されている API プログラミング ルーチンを示します。

PWM16_PERIOD

説明：

デバイス エディタにおいて、PWM16 の Period フィールドで選択された値を示します。この値の範囲は、0 ～ 65535 の間です。

PWM16_PULSE_WIDTH

説明：

デバイス エディタにおいて、PWM16 の PulseWidth フィールドで選択された値を示します。この値の範囲は、0 ～ 65535 の間です。

PWM16_EnableInt

説明：

割り込みモード動作を有効にします。

C プロトタイプ：

```
void  PWM16_EnableInt(void);
```

アセンブリ：

```
lcall  PWM16_EnableInt
```

パラメータ：

なし

戻り値：

なし

副作用：

A および X レジスタがこの関数により変更される場合があります。

PWM16_DisableInt

説明：

割り込みモード動作を無効にします。

C プロトタイプ：

```
void  PWM16_DisableInt(void);
```

アセンブリ：

```
lcall  PWM16_DisableInt
```

パラメータ：

なし

戻り値：

なし

副作用：

A および X レジスタがこの関数により変更される場合があります。

PWM16_Start

説明：

PWM16 ユーザ モジュールを開始します。イネーブル入力が High の場合は、カウンタがダウンカウントを開始します。

C プロトタイプ：

```
void  PWM16_Start(void);
```

アセンブリ：

```
lcall  PWM16_Start
```

パラメータ：

なし

戻り値：

なし

副作用：

A および X レジスタがこの関数により変更される場合があります。

PWM16_Stop

説明：

カウンタの動作を停止します。

C プロトタイプ：

```
void  PWM16_Stop(void);
```

アセンブリ：

```
lcall  PWM16_Stop
```

パラメータ：

なし

戻り値：

なし

副作用：

PWM 出力が Low にリセットされ、Period レジスタへ値を書き込むと、Count レジスタがその新しい周期値で更新されるようになります。A および X レジスタがこの関数により変更される場合があります。

PWM16_WritePeriod

説明：

Period レジスタに周期値を書き込みます。PWM16 が停止される、またはカウンタがゼロに達すると、周期値は、Period レジスタから Count レジスタに直ちに転送されます。

C プロトタイプ：

```
void  PWM16_WritePeriod(WORD wPeriod);
```

アセンブリ：

```
mov   X, [wPeriod]
mov   A, [wPeriod+1]
lcall  PWM16_WritePeriod
```

パラメータ：

wPeriod：wPeriod の値は、0 ～ 2^{16}-1 の値です。MSB が X レジスタで、LSB がアキュムレータで渡されます。

戻り値：

なし

副作用：

A および X レジスタがこの関数により変更される場合があります。

PWM16_WritePulseWidth

説明：

PulseWidth レジスタにパルス幅値 を書き込みます。

C プロトタイプ：

```
void  PWM16_WritePulseWidth(WORD wPulseWidth);
```

アセンブリ：

```
mov   X, [wPulseWidth]
mov   A, [wPulseWidth+1]
lcall  PWM16_WritePulseWidth
```

パラメータ：

wPulseWidth: wPulseWidth の値は、0 から周期値までの値です。MSB が X レジスタで、LSB がアキュムレータで渡されます。

戻り値：

なし

副作用：

カウンタ動作中に PulseWidth レジスタに書き込むと、出力のデューティサイクルが変更されます。この関数により、出力にグリッチが発生したり、予期せず出力が変更されたりする可能性があります。A および X レジスタがこの関数により変更される場合があります。

PWM16_wReadPulseWidth

説明：

PulseWidth レジスタを読み出します。

C プロトタイプ：

```
WORD  PWM16_wReadPulseWidth();
```

アセンブリ：

```
lcall  PWM16_wReadPulseWidth
mov   [wPulseWidth], X
mov   [wPulseWidth+1], A
```

パラメータ：

なし

戻り値：

パルス幅値を PulseWidth レジスタに保存します。MSB が X レジスタで、LSB がアキュムレータで渡されます。

副作用：

A および X レジスタがこの関数により変更される場合があります。

PWM16_wReadCounter

説明：

Count レジスタを読み出します。

この関数は、カウント動作中の Count レジスタを読み出す必要があるアプリケーション用で、副作用を生じることに注意してください。

C プロトタイプ：

```
BYTE  PWM16_wReadCounter();
```

アセンブリ：

```
lcall  PWM16_wReadCounter
```

```
mov   [wCounter], X
mov   [wCounter+1], A
```

パラメータ：

なし

戻り値：

Count レジスタの値を返します。MSB が X レジスタで、LSB がアキュムレータで渡されます。

副作用：

PWM16 Count レジスタを読み取るために、PulseWidth レジスタは一時的に変更されなくてはなりません。これにより、PWM16 Count レジスタの動作が 1 から数カウント延期される場合があります。さらに、この関数は、不用意に割り込み条件を発生することがあります。A および X レジスタがこの関数により変更される場合があります。

ファームウェア ソースコードの見本

以下の例では、C およびアセンブリの両コード間の対応は単純で直接的です。周期および比較値として示される値は、各レジスタがゼロをベースにしており、また、ゼロがダウンカウントの最終カウントになるため、基数値から 1 だけずれる値になります。ユーザ モジュール API に対して、スタックではなく単純に A レジスタで 1 バイトのパラメータを渡しているのは、性能を最適化するためで、アセンブラおよび C コンパイラの両方で使われています。C コンパイラが、PWM16.h ファイルに #pragma fastcall 宣言を見つけたら、スタックに引数をプッシュする代わりに、レジスタで 「INT」タイプのパラメータを渡すメカニズムを導入します。

以下は、API の使用法を示すアセンブリ言語ソースです。

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   Function:  GenerateOneThirdDutyCycle
;   Description:
;      This sample shows how to create a 33% duty cycle output
;      pulse. The clock selected should be 1000 times the required period.
;      The comparator operation is specified to be "Less than or Equal".
;
;   Parameters:  none
;   Returns:     none
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

include "PWM16.inc" ; include the PWM16 API include file

GenerateOneThirdDutyCycle:
   mov   A, E7h                 ; set period to be 1000 counts of the clock
   mov   X, 03h                 ; Period is set to 1000 - 1 = 999 (0x3E7).
   call  PWM16_WritePeriod
   mov   A, 4ch                 ; set PulseWidth to generate a 33% duty cycle
   mov   X, 01h                 ; Pulse Width = 1000/3 - 1 = 332 (0x14C).
   call  PWM16_WritePulseWidth
   call  PWM16_DisableInt       ; ensure that interrupts are disabled
   call  PWM16_Start            ; start the PWM16 - counter will start to
   ret                          ; count when the enable input is asserted
                                ; high
```

同じコードを C で書くと、以下のようになります。

```
/* include the Counter16 API header file    */
#include "PWM16.h"

/* function prototype */
void GenerateOneThirdDutyCycle(void);

/* Divide by eight function */
void GenerateOneThirdDutyCycle(void)
{
    /* set period to eight clocks */
    PWM16_WritePeriod(999);
    /* set pulse width to generate a 33% duty cycle */
    PWM16_WritePulseWidth(332);
    /* ensure interrupt is disabled */
    PWM16_DisableInt();
    /* start the PWM16! */
    PWM16_Start();
}
```

設定レジスタ

明記していない限り、ここに記載されているレジスタの仕様は、すべての PSoC デバイス ファミリに適用されます。

16-bit PWM は、2 つのデジタル PSoC ブロックを使用します。これらのブロックは、左から右への配置順に PWM16_LSB および PWM16_MSB という名前が付けられています。それぞれのブロックは、7 つのレジスタによって独自にパラメータ化されます。以下の表に、定数としての 「パーソナリティ」値と名前付きビットフィールドとしてのパラメータおよび短い説明を示します。これらのレジスタのシンボリック名は、ユーザ モジュール インスタンスの C およびアセンブリ言語インターフェース ファイル (「.h」および 「.inc」ファイル) で定義されます。

Table 4.　機能レジスタ、バンク 1 CY8C26/25xxx

ブロック名 \ ビット	7	6	5	4	3	2	1	0
MSB	0	0	1	Compare Type （比較条件）	Interrupt Type （割り込みタイプ）	0	0	1
LSB	0	0	1	Compare Type （比較条件）	0	0	0	1

[+] Feedback

Table 5. 機能レジスタ、バンク 1、CY8C29/27/24/22/21xxx および CY8CLED04/08/16

ブロック名 \ ビット	7	6	5	4	3	2	1	0
MSB	Data Invert (データ反転)	0	1	Compare Type （比較条件）	Interrupt Type （割り込みタイプ）	0	0	1
LSB	0	BCEN	1	Compare Type （比較条件）	0	0	0	1

BCEN は、ROW のブロードキャスト バス ラインへ比較出力が出てくるのを制御します。このビット フィールドは、デバイス エディタでブロードキャスト ラインを直接設定することでセットされます。デバイス エディタで表示されるユーザ モジュール パラメータを通して設定される Data Invert （データの反転）フラグは、イネーブル入力信号の極性を決定します。CompareType （比較条件）フラグは、比較機能が 「以下」または 「未満」のどちらに設定されているかを示します。InterruptType （割り込みタイプ）フラグは、比較イベントまたは最終カウントのどちらで割り込みを発生させるかを決定します。CompareType と InterruptType は、この文書の前の章に記述されているように、デバイス エディタでユーザ モジュール パラメータにより、直接設定されます。

Table 6. 入力レジスタ、バンク 1

ブロック名 \ ビット	7	6	5	4	3	2	1	0
MSB	0	0	1	1	Clock (クロック)			
LSB	Enable (イネーブル)				Clock (クロック)			

Enable と Clock は、16 種類の信号源のいずれかから、 それぞれ、Enable と Clock 入力信号を選択します。デバイス エディタでユーザ モジュール 「Enable」パラメータを設定すると、Enable の値が決定されます。同様に、ユーザ モジュール「Clock」パラメータを設定すると、Clock の値が決定されます。

Table 7. 出力レジスタ、バンク 1 CY8C26/25xxx

ブロック名 \ ビット	7	6	5	4	3	2	1	0
MSB	0	0	0	0	0	Out Enable (出力許可)	OutputSelect （出力選択）	
LSB	0	0	0	0	0	0	0	0

Table 8. 出力レジスタ、バンク 1、CY8C29/27/24/22/21xxx および CY8CLED04/08/16

ブロック名 \ ビット	7	6	5	4	3	2	1	0
MSB	AuxClk （補助クロック）		AuxEnable (補助イネーブル)	AuxSelect （補助セレクト）		OutEnable (出力許可)	OutputSelect （出力選択）	
LSB	AuxClk （補助クロック）		0	0	0	0	0	0

[+] Feedback

デバイス エディタでユーザ モジュール 「ClockSync」パラメータを設定すると、AuxClk ビットの値が決定されます。たとえ類似した名前が付いていようとも、AuxEnable および AuxSelect は、むしろ、OutEnable および OutSelect ビットフィールドに関連しています。AuxEnable および AuxSelect は、最終カウント出力信号で ROW 出力バスのひとつを駆動することを許可します。これは、デバイス エディタの配置ビューでグラフィカルに ROW バスを操作することで制御されます。OutEnable は、比較出力で ROW またはグローバル出力バスのひとつを駆動する場合にセットされます。OutputSelect は、比較出力で、どのバスを駆動するかを制御します。

Table 9. カウント レジスタ (DR0)、バンク 0

ブロック名 \ ビット	7	6	5	4	3	2	1	0
MSB	Count (MSB：上位)							
LSB	Count (LSB：下位)							

Count （ カウント) は、PWM が持つ PWM16 カウンタの MSB および LSB です。両方とも、PWM16 API を使用して読み出すことができます。

Table 10. 周期レジスタ (DR1)、バンク 0

ブロック名 \ ビット	7	6	5	4	3	2	1	0
MSB	Period (MSB：上位)							
LSB	Period(LSB：下位)							

Period （ 周期) は、周期値の MSB および LSB を保持しており、イネーブルまたは最終カウント条件によって Count レジスタにロードされます。両方とも、デバイス エディタおよび PWM16 API で設定できます。

Table 11. パルス幅レジスタ (DR2)、バンク 0

ブロック名 \ ビット	7	6	5	4	3	2	1	0
MSB	Pulse Width (MSB：パルス幅上位)							
LSB	Pulse Width(LSB：パルス幅下位)							

PulseWidth は、パルス幅の値の MSB および LSB を保持しており、比較イベントの生成に使用されます。両方とも、デバイス エディタおよび PWM16 API で設定されます。

Table 12. 制御レジスタ (CR0)、バンク 0

ブロック名 \ ビット	7	6	5	4	3	2	1	0
MSB	0	0	0	0	0	0	0	0[1]
LSB	0	0	0	0	0	0	0	Start/Stop (開始 / 停止)

Start/Stop がセットされている時には、PWM16 がイネーブルされていることを示します。このビットフィールドは、 PWM16API を使って変更できます。

Start/Stop は、チェーン化された PSoC ブロックの LSB 制御レジスタで制御され、ゼロに設定されます。

更新履歴

バージョン	発起人	説明
2.5	TDU	クロックの説明を以下の通り更新した ブロックに外部デジタル クロックを使用している場合は、最高の精度およびスリープ動作を得るため、ROW の入力同期をオフにすべきです。

Note PSoC Designer 5.1 は、すべてのユーザ モジュール データシートにおいて変更履歴を導入しています。このセクションでは、ユーザー モジュールの現在と以前のバージョンとの差異の概要を掲載しています。

Copyright © 2000-2011 Cypress Semiconductor Corporation. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC Designer ™ and Programmable System-on-Chip ™ are trademarks and PSoC® is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.