8-Bit PWM デッドバンド・ジェネレータのデータシート PWMDB8 V 2.5

Copyright © 2002-2011 Cypress Semiconductor Corporation. All Rights Reserved.

リソース	PSoC® ブロック数			API に必要な容量 （バイト数）		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・スイッチド・キャパシタ・ブロック	フラッシュ ROM	RAM	
CY8C29/27/24/22/21xxx、CY8C23x33、CYWUSB6953、CY8CLED02/04/08/16、CY8CLED0xD、CY8CLED0xG、CY8CNP102、CY8CTST110、CY8CTMG110、CY8CTST120、CY8CTMG120、CY8CTMA120、CY8C28x45、CY8CTMA30xx、CY8C28x45、CY8CPLC20、CY8CLED16P01、CY8C28xxx						
8-bit	2	0	0	35	0	1
16-bit	3	0	0	44	0	1
CY8C26/25xxx						
8-bit	3	0	0	74	0	1
16-bit	3	0	0	97	0	1

このユーザ モジュールを使用した機能するプロジェクト例は、完全に構成された状態でにあります。
www.cypress.com/psocexampleprojects

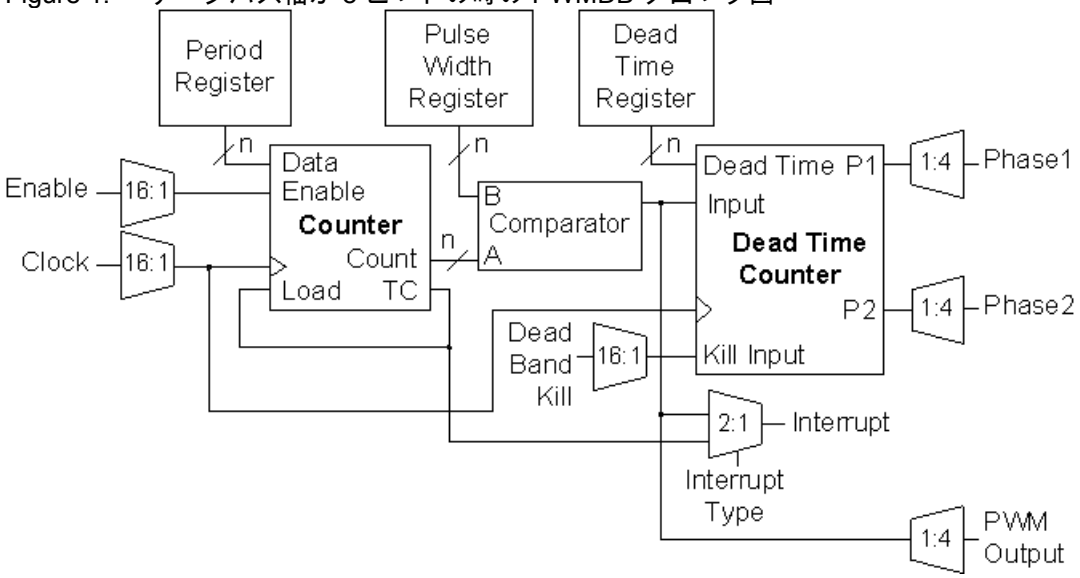
特徴と概要

- 8-bit デッドバンド・ジェネレータ付き 8-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。
- 重ならない出力 Phase1 と Phase2 が、生成された PWM 信号の周波数に追随します。
- デューティサイクルは、プログラム可能です。
- デッドタイムは、プログラム可能です。
- Dead Band Kill 入力が、出力 Phase1 と Phase2 を Low にします。
- 最大 48 MHz のカウンタ・クロックを使用できます。
- 割り込みオプションでは、PWM が生成した信号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。

8-bit PWMDB ユーザ モジュールは、8-bit デッドバンド・ジェネレータを備えたパルス幅変調器です。パルス幅変調器が、プログラム可能な周期とプログラム可能なパルス幅を持った入力信号をデッドバンド・ジェネレータに与えます。デッドバンド・ジェネレータは、二つの重ならない信号を出力し、これらの信号は入力信号と同じ周波数で、デッドタイムはプログラム可能です。Dead Band Kill 入力信号がアサートされると、出力 Phase1 と Phase2 は、共に Low になります。Clock と Enable 入力信号は、いくつかの信号源から選択することができます。出力 Phase1 と Phase2 は、外部ピン、または、グロー

バルな出力バスに配線され、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、パルス幅変調器出力の両方のエッジで、効果的に発行されます。

Figure 1. データパス幅が 8 ビットの時の PWMDB ブロック図



機能説明

PWMDB ユーザ モジュールは、2 つのデジタル PSoC ブロックを使用します。一つ目のブロック、PWM8 は、プログラム可能な周期とパルス幅を持つパルス幅変調器です。パルス幅変調器の出力信号は、2 つ目の PSoC ブロックである DB8 に供給されます。DB8 には、プログラム可能なデッドタイムを持つデッドバンド・ジェネレータが実装されます。2 つの出力信号、Phase1 と Phase2 が、PWMDB8 の出力を与えます。

Control レジスタは、PWMDB のコンポーネントである PWM と DB8 の両方を開始及び停止します。PWM が停止している時、Period レジスタに値を書き込むと、新しい Period レジスタの値が Counter レジスタにもコピーされます。PWM が停止している時、DeadTime レジスタに値を書き込むと、新しい DeadTime レジスタの値が、DeadTimeCounter レジスタにも書き込まれます。PWMDB が停止している時には、PWM 出力と DB8 の Phase1 及び Phase2 出力は、Low になります。

PWMDB は、正論理の Enable 信号によって制御されます。Enable 信号が Low にアサートされている間、PSoC ブロック PWM と DB8 の動作は無効になります。また、PWM の出力が現在の状態で一定に保たれ、DB8 の出力が変化するのを防ぎます。Enable 信号をアサートすると、現在のレジスタの内容を変更することなく PWM の動作を継続します。

同一の入力クロックが、PWMDB のコンポーネントである 8-bit PWM と DB8 の両方で使用されます。

パルス幅変調器

PWM が開始され有効になると、PWM は、クロックの各立ち上がりエッジで Counter レジスタをデクリメントします。Counter レジスタが最終カウントに達した後のクロック エッジで、Period レジスタの値が Counter レジスタにリロードされます。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。Period レジスタの値は、デバイス エディタから、または実行時に API を使用して割り当てることができるパラメータです。

PWM の出力周期は、Period レジスタで指定される周期値に 1 を足したものです。

Equation 1

$$OutputPeriod = PeriodValue + 1$$

[+] Feedback

生成される PWM 波形のデューティサイクルは、周期とパルス幅の値の関係によって定義されます。PulseWidth レジスタの値で、その周期中の、どのカウントで出力を High にするかを決定します。クロックごとに PWM は、Counter レジスタの値と PulseWidth レジスタの値を比較します。カウンタ値が周期値 「以下」である場合は、続くクロックで出力を High にします。周期が自動的にリロードされるとき、Counter レジスタと PulseWidth レジスタの値の比較結果が変化し、続くクロックで出力を Low にします。

デューティサイクルは、次の式で計算できます。

Equation 2

$$DutyCycle = \frac{PulseWidthValue + 1}{PeriodValue + 1}$$

周期及びパルス幅が同じ値に設定されると、永久に出力を High にします。パルス幅の値は、ゼロから周期レジスタに格納された周期値の間の値を取ります。PulseWidth レジスタの値は、デバイス エディタから、または実行時に API を使用して割り当てることができるパラメータです。

割り込み要因は、プログラム可能で、PWM 出力の立ち上がり、または、Counter レジスタが最終カウント条件に達した時から選べます。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周期前です。この割り込みオプションは、デバイス エディタを使って設定できます。実行時に API を使って、割り込みを有効または無効にすることができます。

デッドバンド・ジェネレータ

入力信号 (PWM8 の出力) の各エッジで、以下の処理が繰り返されます。

立ち上がりエッジ

- 次のクロックサイクルの立ち上がりエッジで、Phase2 信号が Low になります。
- DeadTimeCounter レジスタに DeadTime レジスタの値が格納されます。
- 入力クロックの各立ち上がりエッジで DeadTimeCounter レジスタが最終カウントに達するまでデクリメントされます。クロックの次の立ち下がりエッジで、Phase1 が High になります。

立ち下がりエッジ

- 次のクロックサイクルの立ち上がりエッジで、Phase1 信号が Low になります。
- DeadTimeCounter レジスタに DeadTime レジスタの値が格納されます。
- 入力クロックの各立ち上がりエッジで DeadTimeCounter レジスタが最終カウントに達するまでデクリメントされます。クロックの次の立ち下がりエッジで、Phase2 が High になります。

PWM から受信した入力信号の周波数に　Phase1 と Phase2 が追従します。入力信号の High 期間からデッドタイムを差し引いたデューティーサイクルに、Phase1 は追従します。入力信号の Low 期間からデッドタイムを差し引いたデューティーサイクルに、Phase2 は追従します。

入力信号の各位相における有効デッドタイムは、次のようになります。

Equation 3

$$DeadTime = ClockPeriod \times (DeadTime + 1)$$

DeadTime レジスタには、8-bit の値を格納しなくてはなりません。DeadTime レジスタの値は、ゼロから、PWM の Period レジスタの値から 2 を引いた値と PWM の PulseWidth レジスタの値から 2 を引いた値の内、小さいほうの値、または 255 までの値にしなくてはなりません。

DeadTime レジスタの値は、デバイス エディタから、または実行時に API を使用して割り当てることができるパラメータです。

CY8C26/25xxx ファミリを使用している場合は、アサートされた非同期な入力信号 Dead Band Kill が、出力 Phase1 と Phase2 を Low にします。Dead Band Kill 信号は、Phase1 及び Phase2 信号の出力抑制にのみ影響します。

Dead Band Kill 信号が、High にアサートされると、Dead Band Kill 入力が、出力 Phase1 と Phase2 を Low にします。この信号は、Phase1 及び Phase2 信号の出力抑制にのみ影響し、DeadTimeCounter レジスタには影響しません。Dead Band Kill 入力がリリース、または、Low にアサートされると、最初に該当する Phase 出力に、DeadTime クロック数未満かつ 1 以上のジッターを生じる可能性があります。この時、デッドバンド・ジェネレータは、PWM 信号入力に同期します。これは、最初の出力パルスが長くなることを意味します。

Dead Band Kill 入力がデアサートされ、さらに、Dead Band Kill 入力がアサートされたのを検出したとき、PWMDB の出力を常に PWM 信号入力に同期させたい時には、以下のように操作します。

1. API 関数 Stop() を呼び出して、PWMDB を停止します。
2. API 関数 WriteDeadTime() を呼び出して、デッドタイムの期間を再度書き込みます。
3. Dead Band Kill がデアサートされたのを検出したら、PWMDB を開始します。

CY8C29/27/24/22/21xxx, CY8C23x33, CYWUSB6953, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C28x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx ファミリは、3 つの KILL モードをサポートします。いずれの場合でも KILL 信号は、非同期的かつ強制的に出力をロジック 「0」にします。モード間の違いは、どのようにデッドバンド処理が再開されるかにあります。

1. **同期リスタート モード** : KILL 入力が High にアサートされると、内部状態がリセット状態に保持され、デッドバンド期間の初期値がカウンタにリロードされます。KILL 入力が High に保持されている間は、PWM から受信する基準 PWM 信号のエッジは無視されます。KILL 入力が Low にネゲートされると、次に受信する基準 PWM 信号のエッジで、デッドバンド処理が再開されます。デューティーサイクル 100% の PWM 信号が使用されていると、基準 PWM 信号のエッジが到着せず、KILL が Low にネゲートされてもデッドバンド処理は再開されません。5 ページの 「同期リスタート Kill モード」の図を参照してください。
2. **非同期リスタート モード** : KILL が High にアサートされている時、内部状態は影響を受けません。KILL が Low にネゲートされると、出力が復元されますが、最小ディセーブル時間は、クロックの 0.5 から 1.5 周期に制限されます。5 ページの 「非同期リスタート Kill モード」の図を参照してください。

3. **ディセーブルモード**：ディセーブルモードに関連するタイミングに規定はありません。デッドバンド処理ブロックが無効になり、処理を続けるためには、ユーザがファームウェアで再度ブロックを有効にする必要があります。

Figure 2. 同期リスタート KILL モード

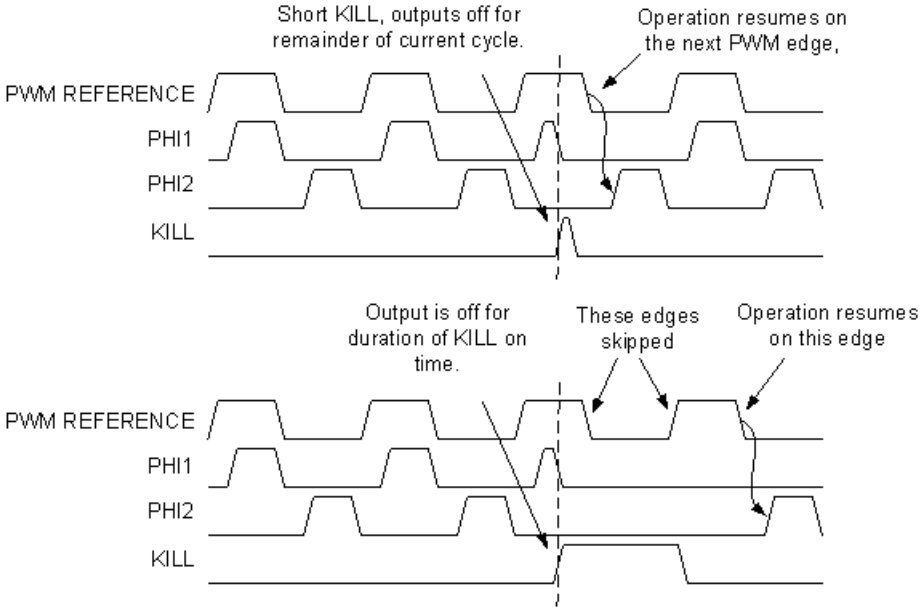
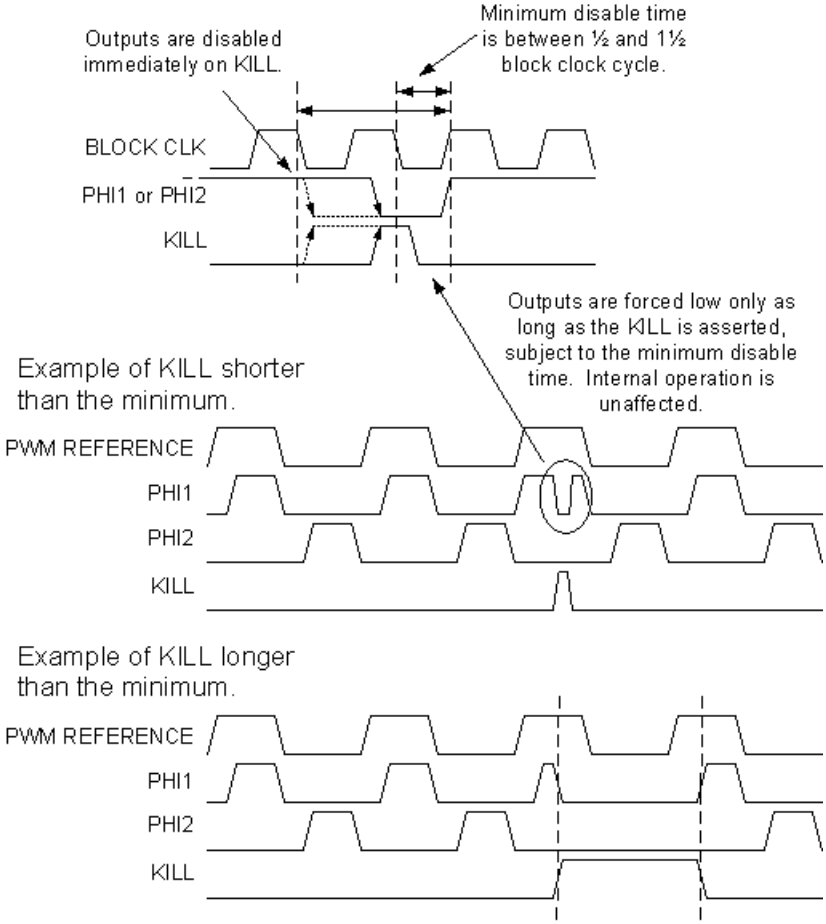


Figure 3. 非同期リスタート KILL モード

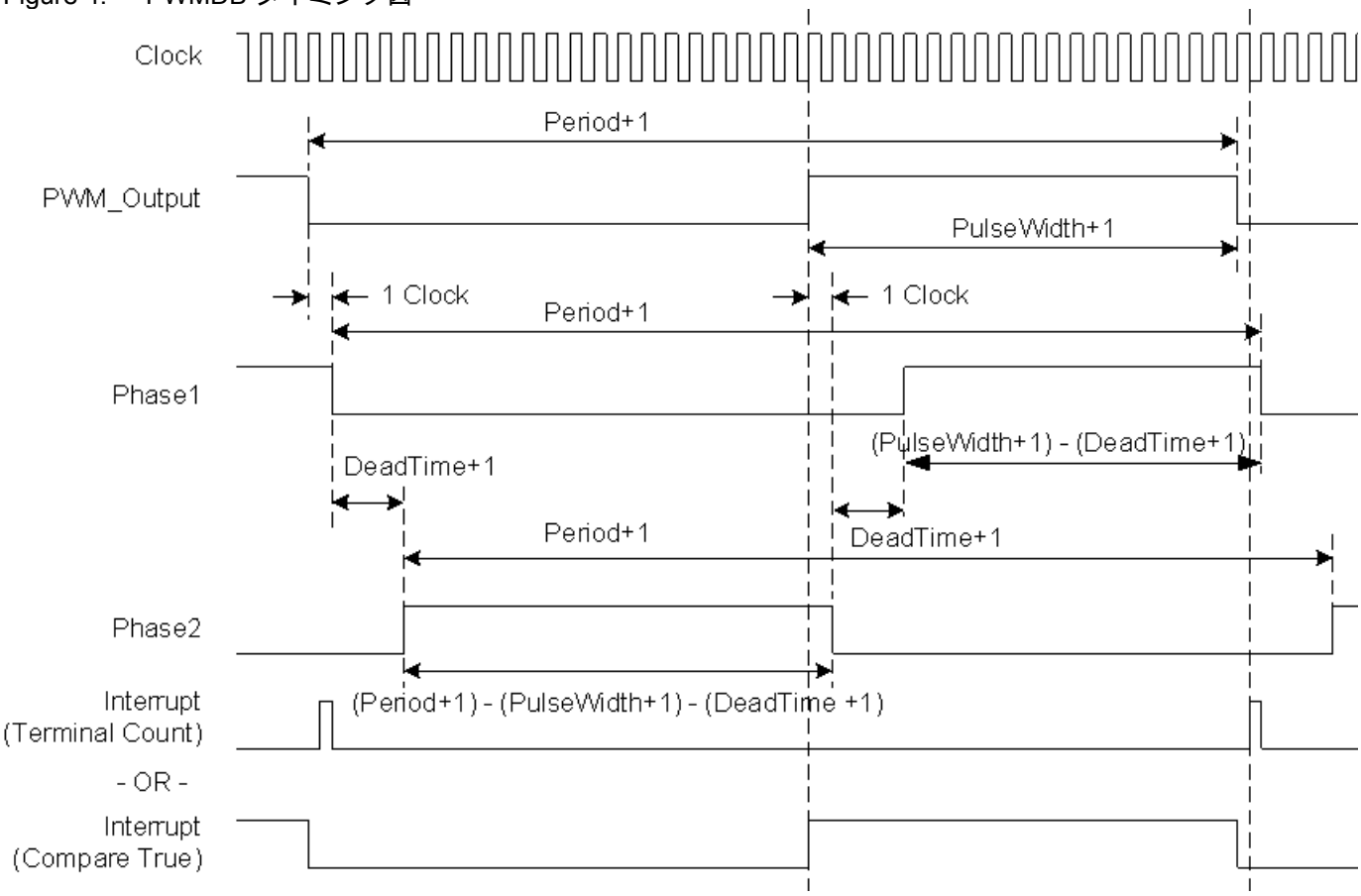


[+] Feedback

タイミング

外部ピンをデバイスのグローバルバス機能を使って PWMDB に配線することで、PWMDB の動作を ON/OFF したり、クロックを供給したりすることができます。グローバル バスの周波数上限は、12 MHz です。

Figure 4. PWMDB タイミング図



DC 電気的特性と AC 電気的特性

Table 1. PWMDB DC および AC 電気的特性

パラメータ	条件および注記	標準値	上下限	単位
$FOutput_{max}$	電源電圧 5.0V 、入力クロック 48 MHz	--	24[1]	MHz
	電源電圧 3.3V、入力クロック 24 MHz	--	12[2]	MHz

電気的特性に関する注意事項

1. 出力がグローバル バスを介して配線される場合、周波数は最大 12 MHz に制限されます。
2. PSoC ブロックへ与えることができる最大クロック周波数は、電源電圧 3.3V の時、24 MHz です。

配置

8-bit PWMDB は、2 つのデジタル PSoC ブロックを消費します。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ル名が与えられます。API は、すべてのレジスタ名にユーザが割り当てたインスタンス名とブロック名を付けて、API のインクルードファイルを介して PWMDB のレジスタに直接アクセスできるようにします 複数種類のビット幅で使用されるブロック名を、以下の表に示します。

Table 2. PSoC ブロックに割り当てられるのシンボル名

PSoC ブロック番号	8-bit PWMDB
1	PWM8
2	DB8

パラメータおよびリソース

Clock (クロック)

Clock パラメータは、多く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 (24V1 および 24V2) が含まれます。パルス幅変調器およびデッドバンド・ジェネレータは両方とも、同じクロック源を使用します。外部デジタル クロックをブロックで使用する場合は、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、多くのソースのいずれかから選択されます。グローバル入出力を介して到着する外部入力は、デバイスが持つ 24 MHz の内部発振器に自動的に同期されます。

Period (周期)

このパラメータは、PWM カウンタの周期を設定します。8-bit PWM では 0 ～ 255 の間、16 ビット PWM で 0 ～ 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。PWM 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)

このパラメータは、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、Counter 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。

PWMOutput (PWM 出力)

この出力パラメータにより、PWM 出力を 4 つのグローバル出力バスのいずれか 1 つに接続できます。PWM 出力が必要ない場合は、その他の出力のグローバル リソースを節約するために、この信号を接続しないようにしてください。

DeadTime (デッドタイム)

このパラメータは、DB8 出力のデッドタイムカウントを設定します。ゼロから、次に示す値の最小値の間の 8-bit 値が設定可能です。PWM の Period パラメータから 2 を引いた値、PWM の Pulse-Width パラメータの値から 2 を引いた値、255

Phase1

この出力パラメータにより、4 つのグローバル出力バスのいずれか 1 つに配線できます。

Phase2

この出力パラメータにより、4 つのグローバル出力バスのいずれか 1 つに配線できます。

DeadBandKill (デッドバンド Kill)

このパラメータは、多くのソースのいずれかから選択されます。DeadBandKill が High にアサートされると、Phase1 及び Phase2 出力は Low になります。

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 ブロックからクロック源を供給することができます。デジタル クロック源は、リップル形式で生成することも可能です。すると、クロック源が、システム クロックに対してスキューを持つようになります。これらのスキューは、PSoC デバイスファミリ CY8C29/27/24/22/21xxx および CY8CLED04/08/16 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 2 以上の分周比で分周された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、この設定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 （ 言い換えると、すべての分周比の積が 1 になるとき ）以外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この設定を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。

DeadBandKill Mode (デッドバンド Kill モード)

このパラメータは、SyncRestartKill、DisableKill、AsyncKill という 3 つの Kill モードから 1 つ選択されます。詳しくは、「デッドバンド・ ジェネレータ」の節を参照してください。

Invert DeadBandKill (デッドバンド Kill の反転)

このパラメータを使って、到着する DeadBand Kill 信号の極性を決定します。

InvertEnable （ 有効入力の反転)

このパラメータは、受信するイネーブル信号の極性を決定します。

割り込み生成制御

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、共用割り込みが要求されるたびに行われます。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 を消費します。

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。

PWMDB8_PERIOD

説明：

デバイス エディタにおいて、PWMDB8 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 ～ 255 の間です。

PWMDB8_PULSE_WIDTH

説明：

デバイス エディタにおいて、PWMDB8 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 ～ 255 の間です。

PWMDB8_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  PWMDB8_EnableInt(void);
```

アセンブリ

```
lcall  PWMDB8_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。

PWMDB8_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  PWMDB8_DisableInt(void);
```

アセンブリ

```
lcall  PWMDB8_DisableInt
```

パラメータ

なし

戻り値

なし

副作用

この関数によって、A および X レジスタが変更される場合があります。

PWMDB8_Start

説明

PSoC ブロック、パルス幅変調器およびデッドバンド・ジェネレータの両方の動作を開始します。PWM の Period レジスタの値が Counter レジスタにロードされ、PWM8 のクロックが開始されます。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  PWMDB8_Start(void);
```

アセンブリ

```
lcall  PWMDB8_Start
```

パラメータ

なし

戻り値

なし

副作用

この関数によって、A および X レジスタが変更される場合があります。

PWMDB8_Stop

説明

PSoC ブロック、PWM8 と DB8 を無効にします。

C プロトタイプ

```
void  PWMDB8_Stop(void);
```

アセンブリ

```
lcall  PWMDB8_Stop
```

パラメータ

なし

戻り値

なし

副作用

この関数によって、A および X レジスタが変更される場合があります。

PWMDB8_WritePeriod

説明

周期値を PWM の Period レジスタに書き込みます。

C プロトタイプ

```
void  PWMDB8_WritePeriod(BYTE bPeriod);
```

アセンブリ

```
mov   A, [bPeriod]
lcall  PWMDB8_WritePeriod
```

パラメータ

周期の値は 0 ~ 255 の間の値で、アキュムレータを介して渡されます。

戻り値

なし

副作用

この関数によって、A および X レジスタが変更される場合があります。

PWMDB8_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  PWMDB8_WritePulseWidth(BYTE bPulseWidth);
```

アセンブリ

```
mov   A, [bPulseWidth]
lcall  PWMDB8_WritePulseWidth
```

パラメータ

パルス幅値は 0 から周期値までの値で、アキュムレータを介して渡されます。

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。この関数によって、A および X レジスタが変更される場合があります。

PWMDB8_WriteDeadTime

説明

DB8 の DeadTime レジスタにデッドタイム・カウント値を書き込みます。

C プロトタイプ

```
void  PWMDB8_WriteDeadTime(BYTE bDeadTime);
```

アセンブリ

```
mov   A, [bDeadTime]
lcall  PWMDB8_WriteDeadTime
```

パラメータ

デッドタイム値は 0 から周期値までの値で、アキュムレータを介して渡されます。

戻り値

なし

副作用

この関数によって、A および X レジスタが変更される場合があります。

PWMDB8_bReadPulseWidth

説明

PWM の PulseWidth レジスタを読み出します。

C プロトタイプ

```
BYTE  PWMDB8_bReadPulseWidth();
```

アセンブリ

```
lcall  PWMDB8_bReadPulseWidth
mov   [bPulseWidth], A
```

パラメータ

なし

戻り値

PulseWidth レジスタに保存されたパルス幅値をアキュムレータを介して戻します。

副作用

この関数によって、A および X レジスタが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、PWMDB8.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: GenerateFetDrive
;  Description:
;      This sample shows how to generate 20% under-lapped output signals.
;      The clock selected should be 30 times the required period.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

include "PWMDB8.inc"          ; include the PWMDB8 API include file

GenerateFetDrive:
    mov   A, 29               ; set the period to be 30 counts of the clock
    call  PWMDB8_WritePeriod
    mov   A, 14               ; set the pulse width to create 50% duty cycle
    call  PWMDB8_WritePulseWidth
    mov   A, 2                ; set the dead time to 20% ->  (15*0.2)-1
    call  PWMDB8_WriteDeadTime
    call  PWMDB8_DisableInt   ; ensure that interrupts are disabled
    call  PWMDB8_Start        ; start the PWMDB8 - 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 Counter8 API header file  */
#include "PWMDB8.h"

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

/* Generate Fet drive function*/
void GenerateFetDrive(void)
{
    /* set period to 30 clocks */
    PWMDB8_WritePeriod(29);
    /* set pulse width to generate a 50% duty cycle */
    PWMDB8_WritePulseWidth(14);
    /* set dead time to 20% -> (15*0.2)-1 */
    PWMDB8_WriteDeadTime(2);
    /* ensure interrupt is disabled */
    PWMDB8_DisableInt();

    /* start the PWM8! */
    PWMDB8_Start();
}
```

設定レジスタ

8-bit PWMDB は、PWM8 および PWMDB8 という名前を持つ 2 つのデジタル PSoC ブロックを使用します。各ブロックは、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ル名は、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フェース ファイル (「.h」および 「.inc」ファイル) で定義されます。

PWM8 設定レジスタ

Table 3.　PWM8 ブロック：レジスタ機能

ビット	7	6	5	4	3	2	1	0
値	0	0	1	0	Interrupt Type （割り込みタイプ）	0	0	1

Interrupt Type (割り込みタイプ) は、出力信号の立ち上がりエッジと最終カウント条件のどちらで割り込みを発生させるかを決定するフラグです。このパラメータはデバイス エディタで設定します。

Table 4.　PWM8 ブロック：レジスタ入力

ビット	7	6	5	4	3	2	1	0
値	Enable (イネーブル)				Clock (クロック)			

Enable は、多くの信号源のいずれかから、入力信号を選択します。Clock は、多くの信号源のいずれかから、入力信号を選択します。どちらのパラメータも、デバイス エディタで設定します。

Table 5. PWM8 ブロック：レジスタ出力

ビット	7	6	5	4	3	2	1	0
値	0	0	0	0	0	Out Enable (出力許可)	OutputSelect （出力選択）	

Output Enable は、出力が有効なことを示すフラグです。Output Select は、PWM の出力が配線される場所を示すフラグです。どちらのパラメータも、デバイス エディタで設定します。

Table 6. PWM8 ブロック：カウンタ レジスタ DR0

ビット	7	6	5	4	3	2	1	0
値	Count (カウント)							

Count は PWM8 のダウン・カウンタです。PWM8 API を使用すると、Count の値を読み取ることができます。

Table 7. PWM8 ブロック：周期レジスタ DR1

ビット	7	6	5	4	3	2	1	0
値	Period (周期)							

Period （周期）は、周期値を保持しており、開始または最終カウント条件によって Counter レジスタにロードされます。この値は、デバイス エディタ及び PWM8 API で設定できます。

Table 8. PWM8 ブロック：パルス幅レジスタ DR2

ビット	7	6	5	4	3	2	1	0
値	PulseWidth (パルス幅)							

PulseWidth は、パルス幅の値を保持しており、出力の生成に使用されます。この値は、デバイス エディタ及び PWM8 API で設定できます。

Table 9. PWM8 ブロック：制御レジスタ CR0

ビット	7	6	5	4	3	2	1	0
値	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、PWM8 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、PWM8 API を使って変更できます。

DB8 設定レジスタ

Table 10. DB8 ブロック：レジスタ機能

ビット	7	6	5	4	3	2	1	0
値	0	0	1	0	0	1	0	1

Table 11. DB8 ブロック：レジスタ入力

ビット	7	6	5	4	3	2	1	0
値	Dead Bank Kill (デッドバンド Kill)				Clock (クロック)			

Dead Bank Kill は、多くの信号源のいずれかから、データ入力を選択します。Clock は、多くの信号源のいずれかから、入力クロックを選択します。どちらのパラメータも、デバイス エディタで設定します。

Table 12. DB8 ブロック：レジスタ出力

ビット	7	6	5	4	3	2	1	0
値	0	0	Phase2 Output Enable (Phase2 出力イネーブル)	Phase2 Output Select (Phase2 出力選択)		Phase1 Output Enable (Phase1 出力イネーブル)	Phase1 Output Select (Phase1 出力選択)	

Phase1 Output Enable は、Phase1 出力が有効であることを示すフラグです。Phase1 Output Select は、DB8 の Phase1 出力が配線される場所を示します。Phase2 Output Enable は、Phase2 出力が有効であることを示すフラグです。Phase2 Output Select は、DB8 の Phase2 出力が配線される場所を示します。すべてのパラメータは、デバイス エディタで設定します。

Table 13. DB8 ブロック：デッドタイム・カウンタ レジスタ DR0

ビット	7	6	5	4	3	2	1	0
値	Dead Time Counter (デッドタイム・カウンタ)							

Dead Time Counter は DB8 ブロックのダウン カウンタです。

Table 14. DB8 ブロック：デッドタイム・レジスタ DR1

ビット	7	6	5	4	3	2	1	0
値	Dead Time (デッドタイム)							

Dead Time は、デッドタイム・カウント値を保持します。PWMDB8 の API を使って変更できます。

Table 15. DB8 ブロック：レジスタ DR2

ビット	7	6	5	4	3	2	1	0
値	0	0	0	0	0	0	0	0

このレジスタは使用されません。

Table 16. DB8 ブロック：制御レジスタ CR0

ビット	7	6	5	4	3	2	1	0
値	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、DB8 がイネーブルされていることを示します。PWMDB8 API を使って変更します。

更新履歴

バージョン	考案者	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 © 2002-2011 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.