改良型 16-Bit デッドバンド PWM データシート PWMDB16L V 1.0

Copyright © 2009-2011 Cypress Semiconductor Corporation. All Rights Reserved.

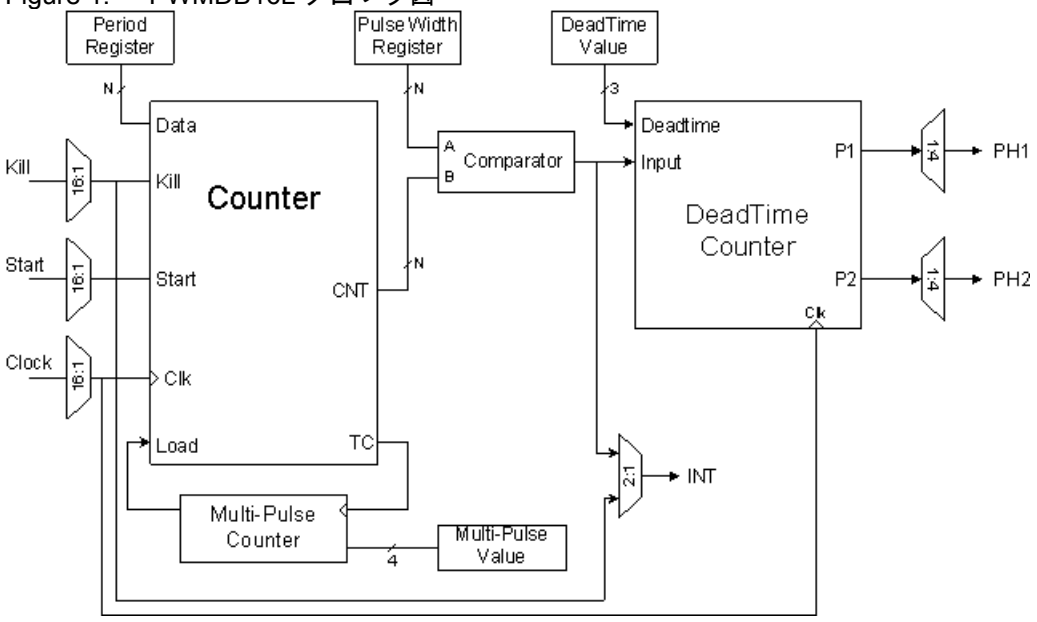
リソース	PSoC® ブロック数			API に必要な容量 (バイト数)		外部 I/O に必要なピン数
	デジタル・ブロック	アナログ・連続時間・ブロック	アナログ・スイッチド・キャパシタ・ブロック	フラッシュ ROM	RAM	
CY8C21/22x45, CY8C28x45, CY8C28xxx, CY8CTMA30xx						
16-bit	2	0	0	116	0	2

特性および概要

- 16-bit デッドバンド・ジェネレータ付き 16-bit 汎用パルス幅変調器 (PWM) は、2 つの PSoC ブロックを使用します。
- カウンタとデッドバンド機能を 1 つのデジタルブロックにまとめました。デッドバンド幅の選択肢に制限があります。
- 重ならない出力 Phase1 (PH1) と Phase2 (PH2) 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 (PH1) と Phase2 (PH2) を Low にします。
- ワンショット / マルチショット機能を搭載しています。
- 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 の Phase1 信号または Kill 入力の立ち上がりエッジから割り込み条件を選べます。

改良型 16-Bit デッドバンド PWM ユーザ モジュール (PWMDB16L) は、PWM16 ユーザ モジュールを改良したものです。PWMDB16L は、すべてのデジタルブロックでデッドバンドをサポートします。このユーザモジュールは、ワンショットやマルチショットなどの改良された機能をもっています。パルス幅変調器は、プログラム可能な周期とパルス幅をもった入力信号をデッドバンド・ジェネレータに提供します。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、パルス幅変調器の Phase1 出力または Kill 入力の立ち上がりで、効果的に発行されます。

Figure 1. PWMDB16L ブロック図



機能説明

PWMDB16L ユーザ モジュールは、2 つの PSoC ブロックを使用し、周期レジスタ、同期ダウンカウンタ、パルス幅レジスタ、デッドタイムカウンタで構成されます。

2 つのブロック、PWM16_LSB と PWM16_MSB は、周期とパルス幅がプログラム可能な 16-bit パルス幅変調器を実装します。PWM 出力信号が、デッドタイムがプログラム可能なデッドバンド・ジェネレータに入力されます。2 つの出力信号、Phase1 と Phase2 が、PWMDB16 の出力を与えます。

Control レジスタは、PWMDB16L を開始および停止します。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、Control1 レジスタの DeadTime ビットに値を書き込むと、新しい DeadTime 値が、レジスタに書き込まれます。PWMDB16L が停止している時には、Phase1 及び Phase2 出力は、Low になります。

PWMDB16L は、正論理の Start 信号によって制御されます。Start 信号が Low にアサートされている間、PSoC ブロック PWMDB16L の動作は無効になります。Start 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。

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

$$\mathrm{OutputPeriod} = \mathrm{PeriodValue} + 1$$

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

[+] Feedback

PulseWidth レジスタの値 「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 の Phase1 出力の立ち上がり、または、PWM モジュールの Kill 入力の立ち上がりから選べます。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。

入力信号の各エッジで、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。
- Control1 レジスタの DeadTime ビットフィールドから DeadTime 値が取り出されます。
- 入力クロックの各立ち上がりエッジで DeadTime 値が最終カウントに達するまでデクリメントされます。次のクロック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。
- Control1 レジスタの DeadTime ビットフィールドから DeadTime 値が取り出されます。
- 入力クロックの各立ち上がりエッジで DeadTime 値が最終カウントに達するまでデクリメントされます。次のクロック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、3-bit の値を格納しなくてはなりません。従って、DeadTime 値は、0、1、2、4、8、16、32、64 ブロッククロックのいずれかの値をとります。

DeadTime 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。

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、DeadTime 値には影響しません。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最低 1 クロック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。

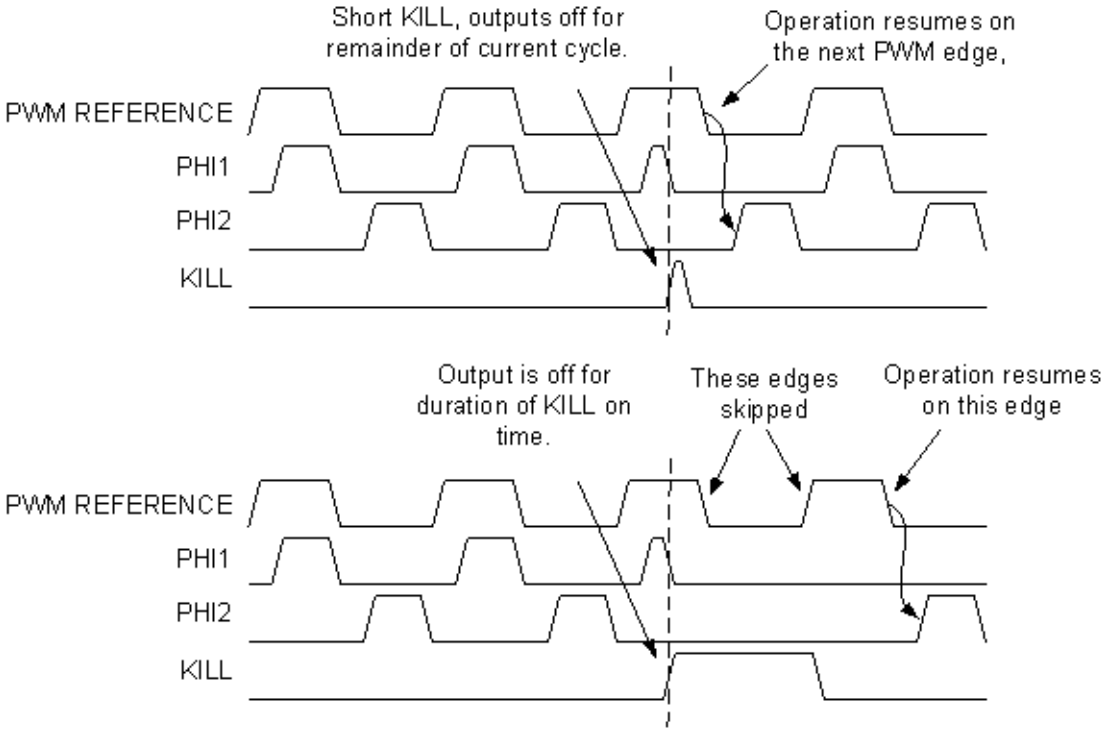
Dead Band Kill 入力がデアサートされ、さらに、Dead Band Kill 入力がアサートされたのを検出したとき、PWMDB の出力を常に PWM 信号入力に同期させたい時には、以下のように操作します。

1. API 関数 Stop() を呼び出して、PWMDB16L 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、PWMDB16L を開始します。

3 つの KILL モードがサポートされています。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口ジック「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。

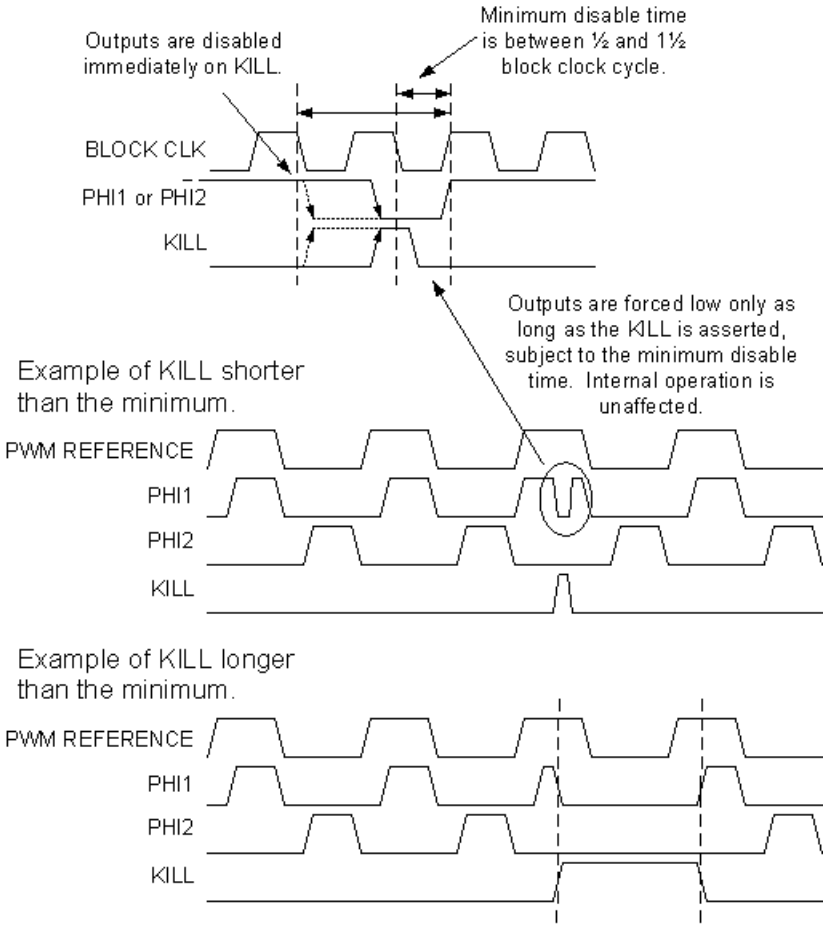
1. **同期リスタート モード**：KILL 入力が High にアサートされると、内部状態がリセット状態に保持され、デッドバンド期間の初期値がカウンタにリロードされます。KILL 入力が High に保持されている間は、PWM から受信する基準 PWM 信号のエッジは無視されます。KILL 入力が Low にネゲートされると、次に受信する基準 PWM 信号のエッジで、デッドバンド処理が再開されます。下の「同期リスタート Kill モードの図」を参照してください。
2. **非同期リスタート モード**：KILL が High にアサートされている時、内部状態は影響を受けません。KILL が 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。下の「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



[+] Feedback

Figure 3.　非同期リスタート KILL モード



PWMDB16L 機能は、カウンタ機能と同一ですが、次の点だけが異なっています。

- カウンタゲート入力がありません。カウントダウンは、別のサブモードで制御されます。
- PWMDB16L のマルチショットモードは PPG（Programmable Pulse Generator「プログラム可能なパルス信号発生器」の略）モードと呼ばれます。PPG モードで PWMDB16L を開始するには、マルチショットレジスタがゼロ以外の値に設定されていなければなりません。最後のショットの後、この機能はディスエーブルモードにはならず、単にカウントを止めます。ハードウェアで開始するか、または、ソフトウェアにより EN ビットに 1 を書いて開始することでカウントが再開されます。最後のショットで START が High にアサートされていれば、カウントは停止しません。また、START を High にアサートし続けても、カウントには影響しません。
- 比較条件は、DR0<=DR2 や DR0<DR2 ではなく、DR0>DR2 です。そのため、比較出力波形が逆相になっています。
- PWMDB16L ユーザ モジュールの動作中、DR2 への書き込みは常にバッファされます。
- カウンタ機能のようにレジスタを設定する必要はありません。
- TC を出力することはできません。比較出力は直接出力されませんが、デッドバンド機能を通じて出力されます。
- KILL モードはデッドバンド機能の設定に従います。
  - 同期リスタート KILL (SyncRestartKill)
  - ディセーブル KILL (DisableKill)

  – 非同期 KILL (AsyncKill)

- KILL は、DisableKill モードを除き、カウンタに影響を及ぼしません。DisableKill モードで KILL がアサートされていると、機能全体が無効になります。

PWMDB16L のデッドバンド機能は、DeadBand 機能と同一ですが、次の点だけが異なっています。

- 前段のブロックから基準クロック入力を取り入れるように設定する必要がありません。基準クロックは、現在のブロックのカウンタ機能の比較出力からとり入れます。
- デッドバンド幅の選択肢は、限られています。0、1、2、4、8、16、32、64 ブロッククロックサイクルからデッドバンド幅を選択できます。デッドバンドを 0 に設定すると、デッドバンド保護機能が無効になります。
- デッドバンド機能はブロッククロックを使用します。カウンタ機能も同じクロックで動作します。
- ユーザ モジュールは、KILL 信号を割り込み要因として取り込めます。

DC 電気的特性と AC 電気的特性

Table 1. PWMDB16L DC 電気的特性と AC 電気的特性

パラメータ	条件および注記	標準値	上限または下限	単位
最大 PWM 出力周波数 $FOutput_{max}$	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

PWMDB16L ユーザ モジュールは、2 つの改良型デジタルブロックを使用します。

パラメータおよびリソース

Clock (クロック)

PWMDB16L ユーザモジュールは、いくつかのソースのうちの 1 つからクロックの供給を受けます。クロック入力を外部ピンもしくは別の PSoC ブロックによって生成されたクロック機能に接続するために、グローバル I/O バスを使用することができます。48 MHz クロック、CPU_32 kHz クロック、分周されたクロックのうちの 1 つ (24V1 もしくは 24V2)、または別の PSoC ブロック出力を クロック入力として指定できます。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。

Start （ 開始 ）

Start パラメータは、いくつかのソースから Start 信号ソースを選択します。Start 信号を High にすると、デジタルブロックで連続カウントを有効にします。Start 信号を Low にすると、カウンタをリセットせずにカウントを無効にします。出力は Start 入力信号の状態によって影響を受けません。例えば、Start 信号が Low にデアサートされたときに出力が High であった場合、出力は High のままとどまります。

[+] Feedback

Start 信号選択パラメータは、デジタルブロックを占有するほかの多くのユーザモジュールの Enable 信号選択パラメータに相当します。Start 信号は、ブロックの Enable 信号の一つであるとも解釈できます。Start パラメータは、アプリケーションプログラム中でユーザ モジュール操作を開始するために使用される API 関数の Start と混同しないよう注意してください。

InvertStart （ Start 入力の反転)

このパラメータを使用すると、Start 入力に到着した信号を反転できます。

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。16 ビットの PWM では、0 ~ 2^{16} – 1 の間の値を選択できます。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、実行中に 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、実行中に API を使って変更することもできます。注意事項 : Phase2 出力はそのまま出力され、Phase1 出力は反転して出力されます。

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。割り込みは、PWM Phase1 信号の立ち上がりエッジ、または PWM モジュールの Kill 入力で、トリガするよう設定できます。別のレジスタで、割り込みを個別に有効にします。

DeadTime (デッドタイム)

このパラメータは、デッドバンド幅の制御に使用します。デッドバンドは 0、1、2、4、8、16、32、64 ブロッククロックサイクルから選択できます。デッドバンドを 0 に設定すると、デッドバンド保護機能が無効になります。

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。

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。

DeadBandKill Mode (デッドバンド Kill モード)

このパラメータでは、SyncRestartKill、DisableKill、AsyncKill という 3 つの Kill モードから 1 つを選択します。詳しくは、上述の 「デッドバンドジェネレータ」のセクションを参照してください。

ClockSync (クロック同期)

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	説明
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。
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場合、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。

PWMEdgeAlign （PWM エッジ合わせ）

このパラメータが設定されると、比較出力が半クロックサイクル分遅れます。この設定を使うと、48MHz クロックがブロッククロックに使われている場合、もっと高分解能な結果が得られます。

SWTrigger （ソフトウェアトリガ）

このパラメータは、ソフトウェアでトリガを有効にする制御ビットです。このパラメータがセットされていると、PWM は PWMDB16L_CONTROL0_REG レジスタの Bit[0] に 1 を書き込むことによってのみ、トリガされます。MultiShot[3:0] がゼロ以外の場合、PWMDB16L はマルチショットモードで稼動します。このパラメータがリセットされていると、PWM はハードウェアによってのみトリガされます。

アプリケーション プログラミング インタフェース

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。

ユーザ モジュールを配置するたびに、インスタンス名が割り当てられます。デフォルトでは、PSoC Designer は与えられたプロジェクト内のこのユーザモジュールの最初のインスタンス名として、PWMDB16L_1 を割り当てます。インスタンス名は、識別子の構文ルールに従っていれば、任意の一意な名前に変更することができます。割り当てられたインスタンス名は、すべてのグローバル関数名、変数名、そして定数記号のプリフィックスになります。次の説明では、簡単にするために、インスタンス名は省略されて単に 「PWMDB16L」となっています。

Note ** すべてのユーザモジュール API と同様にこのユーザモジュールでも、API 関数を呼び出すと A および X レジスタの値は、破壊される可能性があります。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。

PWMDB16L_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、PWM16 のクロックが開始されます。Start 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 PWMDB16L_Start(void);
```

アセンブリ：

```
lcall PWMDB16L_Start
```

パラメータ：

なし

戻り値：

なし

副作用：

API セクションの冒頭にある注意事項 ** を参照してください。

PWMDB16L_Stop

説明

PWMDB16LPSoC ブロックを無効にします。

C プロトタイプ：

```
void PWMDB16L_Stop(void);
```

アセンブラ：

```
lcall PWMDB16L_Stop
```

パラメータ：

なし

戻り値：

なし

副作用：

API セクションの冒頭にある注意事項 ** を参照してください。

PWMDB16L_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 PWMDB16L_EnableInt(void);
```

アセンブラ :

```
lcall PWMDB16L_EnableInt
```

パラメータ :

なし

戻り値 :

なし

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

PWMDB16L_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 PWMDB16L_DisableInt(void);
```

アセンブラ :

```
lcall PWMDB16L_DisableInt
```

パラメータ :

なし

戻り値 :

なし

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

PWMDB16L_WritePeriod

説明

周期値を PWMDB16L の Period レジスタに書き込みます。

C プロトタイプ :

```
void PWMDB16L_WritePeriod(WORD wPeriod);
```

アセンブラ :

```
mov   A, [wPeriod+1]
mov   X, [wPeriod]
lcall PWMDB16L_WritePeriod
```

パラメータ：

wPeriod 値は、0 ～ 2^{16}-1 の間の値です。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：

API セクションの冒頭にある注意事項 ** を参照してください。

PWMDB16L_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 PWMDB16L_WritePulseWidth(WORD wPulseWidth);
```

アセンブラ：

```
mov   A, [wPulseWidth+1]
mov   X, [wPulseWidth]
lcall PWMDB16L_WritePulseWidth
```

パラメータ：

wPulseWidth はパルス幅の値です。有効な値はゼロから周期値までです。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：

API セクションの冒頭にある注意事項 ** を参照してください。

PWMDB16L_WriteDeadTime

説明

Control1 レジスタの DeadTime ビットにデッドタイム・カウント値（DeadTime[2:0]）を書き込みます。

C プロトタイプ：

```
void PWMDB16L_WriteDeadTime(BYTE bDeadTime);
```

アセンブラ：

```
mov A,[bDeadTime]
lcall PWMDB16L_WriteDeadTime
```

パラメータ：

bDeadTime はデッドタイムを設定する値です。有効な値は 0 ~ 7 です。値は、アキュムレータを介して渡されます。

値	説明
0	デッドタイムなし
1	デッドタイムは、1 ブロッククロック
2	デッドタイムは、2 ブロッククロック
3	デッドタイムは、4 ブロッククロック
4	デッドタイムは、8 ブロッククロック
5	デッドタイムは、16 ブロッククロック
6	デッドタイムは、32 ブロッククロック
7	デッドタイムは、64 ブロッククロック

戻り値：

なし

副作用：

API セクションの冒頭にある注意事項 ** を参照してください。

PWMDB16L_TriggerMultiShot

説明

PWMDB16L の Enable ビット（PWMDB16L_CONTROL0_REG. レジスタの bit[0]）をセットします。

Note マルチショットレジスタがゼロ以外に設定されている場合、PWMDB16L ユーザ モジュールは PPG（Programmable Pulse Generator「プログラム可能なパルス信号発生器」）モードで動作します。また最後のショット以降の出力パルスを停止します。停止されたパルス出力は、ハードウェアの Start とソフトウェアの Start （再度 EN ビットに 1 を書きこむ）によってのみ再開できます。

C プロトタイプ：

```
void PWMDB16L_TriggerMultiShot(void);
```

アセンブラ：

```
lcall PWMDB16L_TriggerMultiShot
```

パラメータ：

なし

戻り値：

なし

副作用：

API セクションの冒頭にある注意事項 ** を参照してください。

PWMDB16L_SetTrigMode

説明

PWM のソフトウェアトリガモード、SWT（PWMDB16L_CONTROL0_REG. レジスタの Bit[1]）を設定します。

C プロトタイプ：

```
void PWMDB16L_SetTrigMode(BYTE bSWTMode);
```

アセンブラ：

```
mov A,[bSWTMode]
lcall PWMDB16L_SetTrigMode
```

パラメータ：

定数名	値	説明
TRIGMODE_SOFTWARE_DISABLE	0	ソフトウェアトリガモードを無効にします
TRIGMODE_SOFTWARE_ENABLE	1	ソフトウェアトリガモードを有効にします

戻り値：

なし

副作用：

API セクションの冒頭にある注意事項 ** を参照してください。

PWMDB16L_SetEdgeAlign

説明

PWM のエッジ合わせモード、NPS（PWMDB16L_CONTROL0_REG. レジスタの Bit[3]）を設定します。

C プロトタイプ：

```
void PWMDB16L_SetEdgeAlign(BYTE bEdgeAlign);
```

アセンブラ：

```
mov A,[bEdgeAlign]
lcall PWMDB16L_SetEdgeAlign
```

パラメータ：

定数名	値	説明
EDGEALIGN_DISABLE	0	通常モード
EDGEALIGN_ENABLE	1	比較出力は半クロックサイクル遅れて終了します。このモードは、48MHz のクロックがブロッククロックとして使用されているときに、さらに高い分解能を得るために使用されます。

戻り値：

なし

副作用：

API セクションの冒頭にある注意事項 ** を参照してください。

PWMDB16L_WriteMultiShotCounter

説明

マルチショットカウンタレジスタ、MULTI_SHOT[3:0] （PWMDB16L_CONTROL1_REG. レジスタの Bit[7:4] ）にマルチショットカウント値を書き込みます。

C プロトタイプ：

```
void PWMDB16L_WriteMultiShotCounter(BYTE bMultiShotCounter);
```

アセンブラ：

```
mov A,[bMultiShotCounter]
lcall PWMDB16L_WriteMultiShotCounter
```

パラメータ：

マルチショットカウンタ値は、0 ~ 15 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：

API セクションの冒頭にある注意事項 ** を参照してください。

PWMDB16L_wReadPulseWidth

説明

PWM の PulseWidth レジスタを読み取ります。

C プロトタイプ：

```
BYTE PWMDB16L_wReadPulseWidth(void);
```

アセンブラ：

```
lcall PWMDB16L_wReadPulseWidth
mov   [wPulseWidth], X
mov   [wPulseWidth+1], A
```

パラメータ：

なし

戻り値：

PulseWidth レジスタに保存されている PulseWidth　値をアキュムレータと X レジスタを介して戻します。

副作用：

API セクションの冒頭にある注意事項 ** を参照してください。

PWMDB16L_ClearInt

説明

処理待ちになっているユーザモジュールの割り込みをクリアします。

C プロトタイプ :

```
void PWMDB16L_ClearInt(void);
```

アセンブリ :

```
lcall PWMDB16L_ClearInt
```

パラメータ :

なし

戻り値 :

なし

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

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、PWMDBL16.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 "PWMDB16L.inc"          ; include the PWMDB16L API include file

GenerateFetDrive:
    mov   A, 29               ; set the period to be 30 counts of the clock
    mov   X, 0
    lcall  PWMDB16L_WritePeriod
     mov   A, 14              ; set the pulse width to create 50% duty cycle
    mov   X, 0
    lcall  PWMDB16L_WritePulseWidth
    mov   A, 2                ; set the dead time to 20% ->  (15*0.2)-1
    lcall  PWMDB16L_WriteDeadTime
    lcall  PWMDB16L_DisableInt  ; ensure that interrupts are disabled
     lcall  PWMDB16L_Start       ; start the PWMDB16L - counter will start to
    ret
```

同じコードを C で書くと、以下のようになります。

```
/* include the PWMDB16L API header file  */
#include "PWMDB16L.h"
/* function prototype */
void GenerateFetDrive(void);
/* Generate Fet drive function*/
void GenerateFetDrive(void)
{
    /* set period to 30 clocks */
    PWMDB16L_WritePeriod(29);
    /* set pulse width to generate a 50% duty cycle */
    PWMDB16L_WritePulseWidth(14);
    /* set dead time to 20% -> (15*0.2)-1 */
    PWMDB16L_WriteDeadTime(2);
    /* start the PWM16L! */
    PWMDB16L_Start();
```

設定 レジスタ

PWMDB16L ユーザ モジュールは、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、レジスタに設定された定数とパラメータの役割をする名前付きビットフィールドを短い説明とともに示します。これらのレジスタの名前は、ユーザ モジュール の C およびアセンブリ言語インターフェイス ファイル (「.h」および 「.inc」ファイル) で定義されます。

Table 2. PWMDB16L_FUNC_REG

ブロック / ビット	7	6	5	4	3	2	1	0
MSB	0	0	1	DeadBandKillMode (デッドバンド Kill モード)		0	1	1
LSB	InvertDeadBandKill (デッドバンド Kill の反転)	0	0	0	0	0	1	1

InvertDeadBandKill フラグを使用すると、入力される DeadBand Kill 信号の極性を反転できます。DeadBandKillMode は、3 つの KILL モード (SyncRestartKill、DisableKill、AsyncKill.) のうちの 1 つを選択します。パラメータはデバイス エディタで設定します。

Table 3. PWMDB16L_INPUT_REG

ブロック / ビット	7	6	5	4	3	2	1	0
MSB	0	0	1	1	Clock[3:0]			
LSB	DeadBandKill[3:0]				Clock[3:0]			

DeadBandKill は 16 のソースのいずれかから、同じ名前の入力信号を選択します。デバイス エディタのユーザ モジュールの 「DeadBandKill」パラメータの設定がその値を決定します。同様に、ユーザ モジュールの 「Clock」パラメータの設定が、Clock の値を決定します。

Table 4. PWMDB16L_OUTPUT_REG

ブロック / ビット	7	6	5	4	3	2	1	0
MSB	ClockSync (クロック同期)		1	Phase2[1:0]		1	Phase1[1:0]	
LSB	ClockSync (クロック同期)		0	0	0	0	0	0

デバイス エディタの 「ClockSync」ユーザ モジュールパラメータが、ClockSync ビットの値を決定します。デバイス エディタのパラメータ 「Phase1」と 「Phase2」は、4 つのグローバル出力バスのうちの 1 つに配線することができます。

Table 5. PWMDB16L_COUNT_REG

ブロック / ビット	7	6	5	4	3	2	1	0
MSB	Count(MSB)							
LSB	Count(LSB)							

Count は、PWMDB16L の MSB および LSB ダウンカウンタです。この値は、PWMDB16L の API を使用して読み取ることができます。

Table 6. PWMDB16L_PERIOD_REG

ブロック / ビット	7	6	5	4	3	2	1	0
MSB	Period (周期：MSB)							
LSB	Period (周期：LSB)							

Period は、周期値の MSB および LSB 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 PWMDB16L　の API で設定できます。

Table 7. PWMDB16L_PULSE_WIDTH_REG

ブロック / ビット	7	6	5	4	3	2	1	0
MSB	Pulse Width(MSB)							
LSB	Pulse Width(LSB)							

PulseWidth は、パルス幅値の 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。これはデバイス エディタおよび PWMDB16L API で設定されます。

Table 8. PWMDB16L_CONTROL0_REG

ブロック / ビット	7	6	5	4	3	2	1	0
MSB	0	0	0	0	PWMEdgeAlign （エッジ合わせ）	InterruptType (割り込みタイプ)	0	0
LSB	Start [3:0]				0	InterruptType (割り込みタイプ)	SWTrigger (ソフトウェアトリガ)	Enable (イネーブル)

Start [3:0] は 16 のデジタル入力ソースから 「開始」トリガソース入力を選択します。InterruptType ビットは割り込みトリガタイプを設定します。これらのパラメータの設定は、デバイス エディタだけでできます。PWMEdgeAlign ビットは、PWM 出力エッジ合わせ選択ビットです。SWTrigger ビットは、ソフトウェアトリガイネーブル ビットです。これらのパラメータは、デバイス エディタと PWMDB16LAPI で設定されます。Enable が、セットされていれば PWMDB16L が有効であることを示し、クリアされていれば無効であることを示します。Enable は、PWMDB16LAPI を使って変更できます。

Table 9. PWMDB16L_CONTROL1_REG

ブロック / ビット	7	6	5	4	3	2	1	0
MSB	MULTI_SHOT[3:0]				0	DeadTime[2:0]		
LSB	0	0	0	0	InvertStart (Start の反転)	0	0	0

MULTI_SHOT[3:0] の値はマルチショットカウンタレジスタです。このパラメータは、PWMDB16LAPI を使って変更できます。InvertStart ビットは、受信する Start 信号の極性を決定します。これはデバイス エディタで設定されます。DeadTime はデッドバンド時間制御です。これは、0、1、2、4、8、16、32、64 ブロッククロックサイクルから選べます。DeadTime をゼロに設定すると、デッドバンド保護が無くなります。これはデバイス エディタおよび PWMDB16L API で設定されます。

[+] Feedback

更新履歴

バージョン	著者	説明
1.0	DHA	初期バージョン

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更新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 © 2009-2011 © 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 (サイプレス セミコンダクタ社) 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。特許またはその他の権限下で、ライセンスを譲渡また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、または安全の用途のために使用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ト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® は Cypress Semiconductor Corp の登録商標であり、PSoC Creator ™ および Programmable System-on-Chip ™ は Cypress Semiconductor Corp. の商標です。本文書で言及するその他すべての商標または登録商標は、各社の所有物です。

全てのソース コード (ソフトウェアおよび / またはファームウェア) はサイプレス セミコンダクタ社 (以下「サイプレス」) が所有し、全世界の特許権保護 (米国およびその他の国)、米国の著作権法なら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り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 ソフトウェアおよび / また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コピー、使用、変更そして作成するためのライセンス、ならびにサイプレスのソース コードおよ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、また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、明示的または黙示的を問わず、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、商品性または特定目的への適合性の黙示的な保証か゛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または回路を適用また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ト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。