16-Bit タイマ データシート Timer16 V 2.6

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

リソース	PSoC® ブロック			API メモリ（バイト）		ピン（外部 I/O あたり）
	デジタル	アナログ CT	アナログ SC	Flash	RAM	
CY8C29/27/24/22/21xxx, CY8C23x33, CYWUSB6953, CY7C64215, CY8CLED02/04/08/16, CY8CLED0xD, CY8CLED0xG, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8CTMA30xx, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx						
	2	0	0	93	0	1
CY8C26/25xxx	2	0	0	142	0	1

このユーザ モジュールを使用する単一あるいはすべてを設定する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 汎用タイマは 2 つの PSoC ブロックを使用
- 48 MHz までのソース クロック レート
- カウント終了でカウント期間の自動リロード
- 24 MHz までのキャプチャクロック
- ターミナルカウント出力パルスを、他のアナログやデジタル機能用の入力クロックとして使用可能
- ターミナルカウントやキャプチャ（一部のデバイス）、またはカウンタがプリセット値に到達した場合の割り込みオプション

16-bit タイマ ユーザ モジュールは、ダウンカウンタで構成され、その周期とパルス幅はプログラマブルです。クロックとイネーブル信号は、システムのタイムベースまたは外部ソースから選択できます。開始すると、タイマは連続して動作し、カウント終了に到達した際に、ピリオドレジスタからその値をリロードします。ターミナルカウントのすぐ後の、クロックサイクルで出力パルスはハイになります。エッジ感知性キャプチャ入力信号をアサートすることにより、イベントはタイマの現在カウント値をキャプチャできます。各クロック周期で、タイマは、「未満」または「以下」のいずれかで、比較レジスタの値に対して、カウンタの値をテストします。割り込みは、最終カウントと比較信号に基づいて生成されます。一部のデバイスファミリは、さらに 2 つの追加機能を提供します。割り込みオプションには、「キャプチャによる割り込み」が含まれ、さらに、比較信号はバスにルーティングされることもあります。これらのオプションが利用可能なデバイスでは、デバイス エディタにそれが表示されます。

Figure 1.　タイマ ブロック ダイアグラム（ほとんどの PSoC デバイス）、データパス幅 n = 16

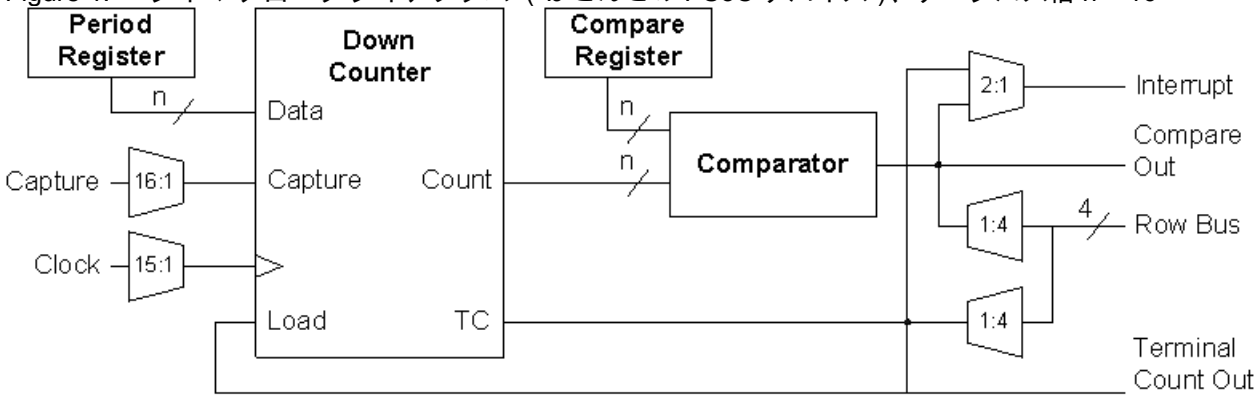
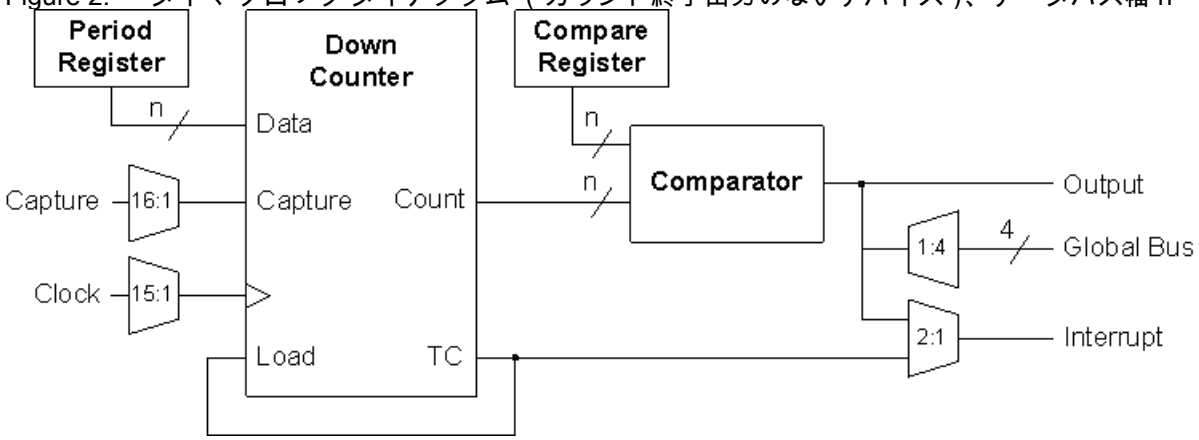


Figure 2.　タイマ ブロック ダイアグラム（カウント終了出力のないデバイス）、データパス幅 n = 16



機能説明

Timer16 ユーザ モジュールは、2 つのデジタル PSoC ブロックを用います。それぞれ総分解能中で 8 ビットずつ分担します。連続した複数のブロックがリンクされ、その結果として内部キャリー、カウント終了あるいは比較信号が同期してつながれます。これは、ブロックからブロックへの 8-bit カウント、ピリオドおよび比較レジスタ (それぞれデータ レジスタ DR0、DR1、DR2) を連結し、必要な分解能を提供します。この方法を使用すると、16-bit タイマが単一のモノリシック同期タイマとして動作します。

タイマ API は、C およびアセンブリから呼び出できる関数を提供し、タイマの動作を停止および開始したり、さまざまなデータ レジスタとの読み書きを実行します。コントロールレジスタがタイマ ユーザ モジュールを開始したり、停止したりします。タイマの停止中に期間レジスタを書き込むことにより、期間レジスタの値が、カウンタ レジスタにコピーされます。タイマの停止中は、出力はローにアサートされます。

タイマが開始すると、カウントレジスタはクロックの各立ち上がりエッジで 1 減ります。ゼロカウントに達した後の立ち上がりクロック エッジで、期間レジスタからカウントレジスタがリロードされます。次の立ち下がりエッジで、ターミナルカウントイベントがトリガされ、出力が 0.5 クロックサイクルの間、ハイにアサートされます。CY8C29/27/24/22/21xxx, CY8C23x33, CYWUSB6953, CY7C64215, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CNP102, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx デバイスファミリの場合、1 クロックサイクルの間、ハイにアサートされます。このように、タイマはクロック分周

[+] Feedback

器として作動します。その期間と周波数は、タイマの期間レジスタ値 + 1 と同等の因数によって、ソースクロックの期間と周波数と関連しています。

Equation 1

$$OutputPeriod = SourceClockPeriod \times (PeriodRegisterValue + 1)$$

期間値がゼロの場合、半クロックサイクルによってシフトされた入力ソースクロックを出力することで、1 分周クロックを生成します。CY8C29/27/24/22/21xxx, CY8C23x33, CYWUSB6953, CY7C64215, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CNP102, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx デバイスファミリでは、ターミナルカウントパルス幅はかならず半サイクルに設定します。

ターミナルカウント出力のデューティサイクルは次の通りです。

Equation 2

$$DutyCycle = \frac{0.5}{PeriodValue + 1}$$

ターミナルカウントパルス幅が 1 サイクルに設定されているときは、デューティサイクルの長さは 2 倍になります。

Equation 3

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

ピリオドレジスタの値は、デバイス エディタを使用して割り当てることができるパラメータです。さらに、API を使用してランタイムで修正することもできます。カウントレジスタ値がゼロ（最終カウント）に達した後のサイクルで、期間レジスタの値は、カウントレジスタに自動的にコピーされます。そのため、期間が API によって変更された場合、新しい値はすぐに有効にはなりません。ランタイムにすぐに変更するための正しい手順はタイマーをとめることであり、新しい期間値を書き込み、次にタイマをリスタートすることです。

各入力クロックで、カウントレジスタのカウントは、比較レジスタに保存されている値と比較されます。比較では、デバイス エディタの CompareType パラメータに割り当てられているオプションに従って、「未満」または 「以下」のテストを実施します。比較条件が満たされた場合、次のクロックで比較イベントがトリガされます。

CY8C29/27/24/22/21xxx, CY8C23x33, CYWUSB6953, CY7C64215, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CNP102, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx デバイスファミリでは、タイマ ユーザ モジュールは比較出力信号を補助出力として提供します。このアクティブ HIGH 信号は、比較条件が満足されたサイクルの次のクロックサイクルの立ち上がりエッジで、アサートされます。補助出力は、隣接のデジタル PSoC ブロックに直接接続できませんが、その他のデジタル PSoC ブロック、またはローカルロー出力バスを介して GPIO ピンに接続できます。

キャプチャ入力が高にアサートされている場合、移行はシステムクロックに同期され、カウントレジスタの値は比較レジスタに転送されます。CY8C26/25xxx ファミリでは、これによって、比較タイプが「以下」に設定されている場合は次のタイマ入力クロックサイクルで比較イベントが発生し、比較タイプが 「未満」に設定されている場合は 2 クロックサイクル後に比較イベントが発生します。CY8C29/27/24/22/21xxx ファミリと CY8CLED04/08/16 ファミリでは、割り込みタイプが 「キャプチャ」に設定されている場合、キャプチャイベントに続いて割り込みが発生します。次に ReadTimer API 関数を使用して、カウント値を読むことができます。次の条件が満たされた場合、「比較 = 真」イベントで割り込みが発生します。

1. CY8C26/25xxx ファミリでは、割り込みタイプは比較イベントでトリガされるよう設定します。CY8C29/27/24/22/21xxx, CY8C23x33, CYWUSB6953, CY7C64215, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CNP102, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx ファミリでは、割り込みタイプは 「キャプチャ」でトリガされるよう設定します。
2. タイマ割り込みが有効
3. グローバル割り込みが有効

経過時間は次のように計算されます。

Equation 4

$$ElapsedTime = ClockPeriod \times (PeriodValue - CounterValue)$$

割り込みを有効にして、最終カウントまたは比較イベントでトリガすることができます。PSoC デバイスの CY8C29/27/24/22/21xxx, CY8C23x33, CYWUSB6953, CY7C64215, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CNP102, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx ファミリでは、キャプチャ信号自体でトリガされます。CY8C26/25xxx ファミリでは、外部キャプチャ信号を使用してイベントタイミングを実行する場合、比較イベントでトリガするよう割り込みを設定します。Arch27Name;/Arch27NameCont; では、外部キャプチャ信号を使用してイベントタイミングを実行する場合、キャプチャイベントでトリガするよう割り込みを設定します。経過タイミング測定を実行する場合、最終カウントでトリガするよう割り込みを設定します。

カウントレジスタや比較レジスタに影響を及ぼさずに、動作中にタイマのカウントダウン値の読むことが必要なアルゴリズムでは、ReadTimerSaveCV() API を呼び出すことができます。この関数は、比較レジスタのコンテンツを保持しながら、カウントレジスタ値を読みます。このユーザ モジュールの API セクションに記述されているように、この関数ではレイテンシの副作用が発生する可能性があります。

キャプチャメカニズムによって、動作が起こる前に、外部イベントのタイミングを最大時間で限定することが可能になります。この手順は次のとおりです。

1. 期間レジスタを最大値と同等の期間値に設定します。
2. 比較レジスタを、次の方式で計算される最大時間制限カウントに設定します。

Equation 5

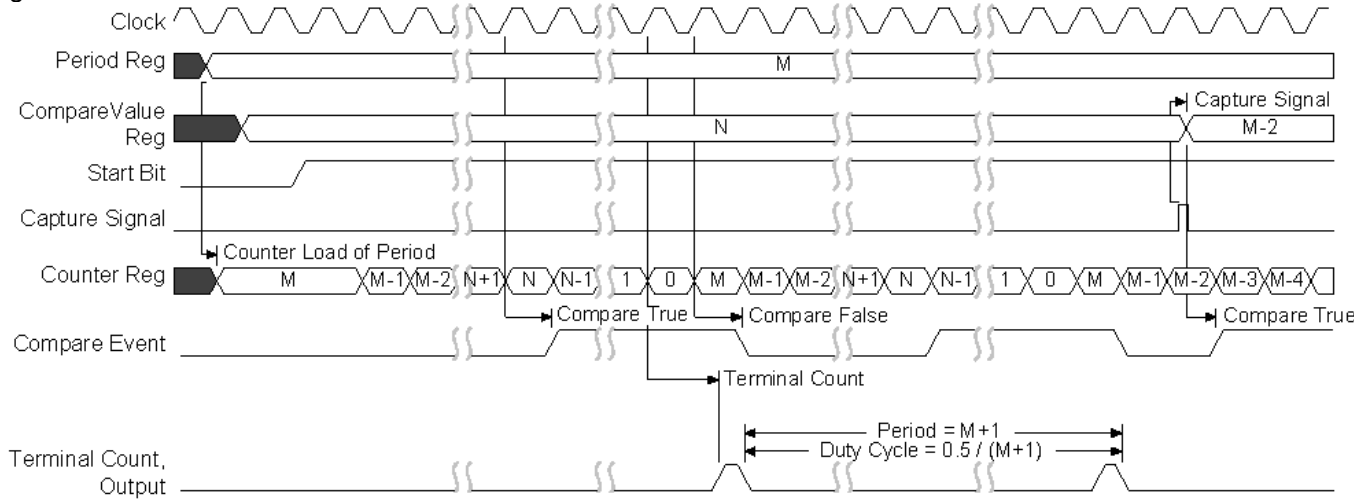
$$MaxTimeLimitCount = PeriodValue - \frac{MaxTimeLimit}{ClockPeriod}$$

1. CY8C26/25xxx ファミリでは、割り込みを、「以下」の比較関数で比較イベントがトリガされるように設定します。CY8C29/27/24/22/21xxx, CY8C23x33, CYWUSB6953, CY7C64215, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CNP102, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx ファミリでは、「キャプチャ」で割り込みがトリガされるよう設定します。
2. 適宜、タイマを開始します。
3. 割り込みがトリガされたとき、比較レジスタを読みます。
4. CY8C26/25xxx ファミリでは、比較値が最大時間制限カウントより大きい場合は、外部キャプチャイベントが発生したと想定でき、経過時間を計算できます。比較値が制限カウント以下の場合、イベントは発生せず、最大時間制限は期限切れとなったことを意味します。

タイミング

PSoC デバイスのグローバルバス機能によってカウンタにルーティングされている外部ピンは、タイマのクロックを提供します。以下の図は、タイマ ユーザ モジュールのタイミングを示しています。

Figure 3. タイミング図



DC および AC の電気的特性

Table 1. タイマ AC の電気的特性

パラメータ	標準値	制限	単位	条件および注意
最大出力周波数	--	48[ab]	MHz	16-bit 幅、Vdd=5.0V[2]
最大出力周波数	--	24[a]	MHz	Vdd=5.0V および 48 MHz の入力クロック
	--	12[c]	MHz	Vdd=3.3V および 24 MHz の入力クロック

a. 入力または出力が、グローバル バスを通して迂回される場合は、周波数は、最大 12 MHz に制限されます。

b. タイマがアクティブなキャプチャ機能で使用されている場合、入力クロックの周波数制限は 24MHz です。

c. PSoC ブロックへの可能な最速のクロックは、3.3V の動作で 24 MHz です。

配置

タイマは、2 つのデジタル PSoC ブロックを消費します。最下位のバイト (LSB) から最上位のバイト (MSB) までブロック番号が増えるよう、デバイス エディタで両方のブロックが順番に配置されます。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確認し、API include ファイルを通してタイマレジスタへの直接アクセスを提供します。ブロック名を以下の表に示します。

[+] Feedback

Table 2.　シンボリック PSoC ブロック名

PSoC ブロック	16-Bit タイマ
1	TIMER16_LSB
2	TIMER16_MSB

パラメータおよびリソース

タイマ ユーザ モジュールが、デバイス エディタを使って選択されて配置されると、以下のパラメータの値を選択したり、変更したりできます。

クロック

Clock パラメータは、利用可能なソースのいずれかから選択されます。これらのソースには、48 MHz オシレータ (5.0V 動作のみ)、24V1、24V2、その他の PSoC ブロック、グローバル入出力を通して迂回される外部入力が含まれます。ブロックで外部デジタル 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、入力の同期をオフにする必要があります。

キャプチャ

このパラメータは、利用可能なソースのいずれかから選択されます。この入力の立ち上がり工ッジでは、カウントレジスタが比較レジスタに転送されます。このパラメータが 1 の値に設定されているか、外部でハイに維持されている場合、ソフトウェアのキャプチャメカニズムは正しく作動しません。

出力

出力パラメータは、無効にすること、あるいは 4 つのグローバル出力信号のひとつに接続することができます。このパラメターは PSoC デバイスの CY8C26/25xxx ファミリのみに適用されます。

TerminalCountOut (カウント終了出力)

TerminalCountOut は、補助的なカウンタ出力です。このパラメータを使って、この信号を無効にするか、未処理の出力バスに接続できます。PSoC デバイスの CY8C29/27/24/22/21xxx ファミリおよび CY8CLED04/08/16 ファミリのみで適用されます。

CompareOut (比較出力)

比較出力は、無効にするか (割り込み動作に影響することなく)、未処理の出力バスに接続できます。このパラメータの設定にかかわらず、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として常に利用できます。PSoC デバイスの CY8C29/27/24/22/21xxx, CY8C23x33, CYWUSB6953, CY7C64215, CY8CLED02/04/08/16, CY8CLED03D/04D, CY8CNP102, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx ファミリのみに適用されます。

ピリオド (期間)

このパラメータは、タイマの期間を設定します。許可される値は、0 と 2^{32}-1 の間です。この値は、ピリオドレジスタにロードされます。ピリオドは、カウンタがゼロに達したとき、またはタイマが無効の状態から有効になったときに、自動的にリロードされます。この値は、API を使って変更できます。

CompareValue (比較値)

このパラメータは、比較イベントがトリガされたとき、タイマ期間にカウント点を設定します。この値は、比較レジスタにロードされます。許可される値は、ゼロと期間値の間です。この値は、API を使って変更できます。

CompareType (比較タイプ)

このパラメータは、上記の機能の説明に記載されているように、「未満」または 「以下」の比較関数タイプを設定します。

InterruptType (割り込みタイプ)

このパラメータは、割り込みをトリガするのが最終カウントイベントか比較イベントかを指定します。割り込みは API によって有効にされます。

ClockSync (クロック同期)

PSoC デバイスでは、デジタル ブロックは、システム クロックに加えて、クロック ソースを提供できます。デジタル クロック ソースは、リップル形式で連結することも可能です。これで、システム クロックに関するスキューが発生することになります。様々なデータ最適化作業、とりわけシステムバスに適用された最適化作業のため、このようなスキューは CY8C29/27/24/22/21xxx および CY8CLED04/08/16 PSoC デバイスの製品ファミリにとって特に重要です。このパラメータは、制御クロック スキューに使用され、PSoC ブロック レジスタ値を読み書きする場合に、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	使い方
SysClk への同期	2 以上で除算される 24 MHz (SysClk) からの派生クロック ソースでこの設定を使用します。例には、VC1、VC2、VC3 (VC3 が SysClk によって駆動される場合)、32KHz、SysClk ベースのソースを持つデジタル PSoC ブロックが含まれます。正しい同期が行われるよう、外部で生成されたクロック ソースもこの値を使用してください。
SysClk*2 への同期	結果の周波数が 48 MHz でない限り (言い換えると、すべての除算結果が 1 になる場合)、48 MHz (SysClk*2) ベースのクロックでは、この設定を使用します。
SysClk 指示の使用	24 MHz (SysClk/1) クロックが必要な場合に使用します。これは、実際には同期されませんが、システム クロック自体への低スキュー アクセスを提供します。選択すると、このオプションは、上の Clock パラメータの設定を上書きします。除算の最終結果が 24 MHz 出力を生成するすべての場合は、VC1、VC2、VC3、またはデジタル ブロックの代わりに常に使用してください。
Unsynchronized (非同期)	48 MHz (SysClk*2) 入力が選択される場合に使用します。 非同期入力が適切な場合に使用します。一般に、割り込み生成がカウンタとしての単独の用途である場合にのみ使用することを推奨します。この設定は、スリープ中でもアクティブに維持されるブロックで必要です。

TC_PulseWidth

このパラメータで、最終カウント出力パルスの幅が 1 クロックサイクルか、半クロックサイクルかを指定することができます。

InvertCapture (反転キャプチャ)

このパラメータによって、イネーブル入力信号のセンスを決定します。 「Normal (標準)」を選択すると、入力としてアクティブハイが有効になります。 「Invert (反転)」を選択すると、センスがアクティブローが認識されるべきとして、解釈されます。InvertCapture は、PSoC デバイスの CY8C29/27/24/22/21xxx, CY8C23x33, CYWUSB6953, CY7C64215, CY8CLED02/04/08/16,

CY8CLED03D/04D, CY8CNP102, CY8CTST110, CY8CTMG110, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C21x45, CY8CTMG300, CY8CTST300, CY8CTMA300, CY8CTMA301, CY8CTMA301D, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx ファミリにのみ適用されます。

割り込み生成制御

PSoC Designer で、**[Enable interrupt generation control**（割り込み生成の制御を有効にする)] チェックボックスが選択されている場合、さらに二つのパラメターが有効になります。これは **[プロジェクト] > > [設定] > > [チップ エディタ] >** でアクセスできます。「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トル 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トル 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トル テーブル エントリがバイパスされます。1 つのブロック リソースが異なるオーバーレイで使用されるような、複数のオーバーレイを持つプロジェクトでは特に、割り込み API が生成されるかどうかを正しく選択してください。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 コードを生成する必要がなくなり、オーバーヘッドを軽減できます。

IntDispatchMode

IntDispatchMode パラメータを使用して、同一ブロック内の異なるオーバレイ内に存在する複数のユーザ モジュールで共用している割り込みについて、割り込みをどのように処理するかを指定します。「ActiveStatus」を選択すると、ファームウェアは共用されている割り込み要求を処理する前に、どのオーバレイがアクティブかをテストします。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。このためにレイテンシが付加され、共用割り込み要求を処理する非決定性のプロセスも生じますが、RAM は不要です。「OffsetPreCalc」を選択すると、ファームウェアはオーバレイが最初にロードされたときだけ共用割り込みの要求のソースを計算します。この計算によって割り込みレイテンシは減少し、共用割り込み要求を処理する決定性のプロセスが生じますが、これは RAM のバイトを消費します。

アプリケーション プログラミング インタフェース (API)

アプリケーション プログラミング インタフェース (API) ルーチンは、設計者が高級言語でモジュールを操作できるようにユーザ モジュールをコンポーネントとして提供します。このセクションでは、「include」ファイルによって提供される各関数に対するインタフェースおよび定数を示します。

Note 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。A と X の値が呼び出し後に必要な場合は、呼び出し元関数で A と X の値を保存してください。PSoc Designer のバージョン 1.0 以降、効率性の観点から、この「registers are volatile（レジスタの揮発性）」ポリシーが採用されています。C コンパイラは、自動的にこの条件を取り込んで処理されています。アセンブリ言語のプログラマは、コードがこのポリシーを遵守していることも確認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 は変更されないこともありますが、将来も変更されないという保証はありません。

アプリケーション プログラミング インタフェース (API) ルーチンは、設計者が高級言語でモジュールを操作できるようにユーザ モジュールをコンポーネントとして提供します。以下に、Timer16 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。

Timer16_PERIOD

説明：

デバイス エディタにおいて、Timer16 の Period フィールドで選択された値を示します。値は、0 ～ 65535 の範囲を持ちます。

Timer16_COMPARE_VALUE

説明：

デバイス エディタにおいて、Timer16 の PulseWidth フィールドで選択された値を示します。値は、0 ～ 65535 の範囲を持ちます。

Timer16_EnableInt

説明：

割り込みモード動作を有効にします。注：ただし、グローバル割り込みも、割り込みが実際に実行される前に有効にされなければなりません。

C プロトタイプ：

```
void  Timer16_EnableInt(void);
```

アセンブラ：

```
lcall  Timer16_EnableInt
```

パラメータ：

なし

戻り値：

なし

特殊作用：

このルーチンは、IO 空間の適切な割り込み有効化レジスタを調整します。A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ

モデル（c）のすべての RAM ページ ポインタ レジスタにも同じことが言えます。必要な場合は、すべての fastcall16 関数呼び出しにわたって、これらの値を保存してください。

Timer16_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  Timer16_DisableInt(void);
```

アセンブラ：

```
lcall  Timer16_DisableInt
```

パラメータ：

なし

戻り値：

なし

副作用：

このルーチンは、IO 空間の適切な割り込み有効化レジスタを調整します。A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル（c）のすべての RAM ページ ポインタ レジスタにも同じことが言えます。必要な場合は、すべての fastcall16 関数呼び出しにわたって、これらの値を保存してください。

Timer16_Start

説明：

Timer16 操作を開始します。カウントレジスタは次のクロックサイクルで減少します。

C プロトタイプ：

```
void  Timer16_Start(void);
```

アセンブラ：

```
lcall  Timer16_Start
```

パラメータ：

なし

戻り値：

なし

副作用：

A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル（c）のすべての RAM ページ ポインタ レジスタにも同じことが言えます。必要な場合は、すべての fastcall16 関数呼び出しにわたって、これらの値を保存してください。

Timer16_Stop

説明：

Timer16 操作を停止します。

C プロトタイプ：

```
void  Timer16_Stop(void);
```

アセンブラ：

```
lcall  Timer16_Stop
```

パラメータ：

なし

戻り値：

なし

特殊作用：

出力がローにセットされ、それに続く期間レジスタへの書き込みにより、カウント レジスタが新しい期間値で更新されます。A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル（c）のすべての RAM ページ ポインタ レジスタにも同じことが言えます。必要な場合は、すべての fastcall16 関数呼び出しにわたって、これらの値を保存してください。

Timer16_WritePeriod

説明：

期間値を期間レジスタに書き込みます。この期間は、ゼロカウント状態になったとき、または Timer16 が停止した直後に、カウントレジスタにロードされます。

C プロトタイプ：

```
void  Timer16_WritePeriod(WORD wPeriod);
```

アセンブラ：

```
mov   X, [wPeriod]           ; place MSB in X
mov   A, [wPeriod+1]         ; place LSB in A
lcall  Timer16_WritePeriod
```

パラメータ：

wPeriod: wPeriod は 0 と 2^{16}-1 の間の値で、Timer16 期間を設定します。X レジスタでは MSB、累算器では LSB が渡されます。

戻り値：

なし

副作用：

A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル（c）のすべての RAM ページ ポインタ レジスタにも同じことが言えます。必要な場合は、すべての fastcall16 関数呼び出しにわたって、これらの値を保存してください。

Timer16_WriteCompareValue

説明：

タイマの比較レジスタ値を変更します。予期しない副作用を避けるために、タイマは（開始 API 関数を介して有効になっていない、または事前に停止 API 関数を呼び出すことによって）無効になっている必要があります。

C プロトタイプ：

```
void  Timer16_WriteCompareValue(WORD wCompareValue);
```

アセンブラ：

```
mov   X, [wCompareValue]           ; place MSB in X
mov   A, [wCompareValue+1]         ; place LSB in A
lcall  Timer16_WriteCompareValue
```

パラメータ：

wCompareValue: wCompareValue は 0 と期間レジスタ値の間の値で、Timer16 比較値を設定します。X レジスタでは MSB、累算器では LSB が渡されます。

戻り値：

なし

副作用：

この関数は、タイマが実行中で比較値がカウントレジスタの現在値以上であるときに呼び出され、比較イベントが発生します。比較レジスタは複数の PSoC ブロックに分配され、一度に 1 バイトずつ書き込まれるため、比較レジスタ値は、多少予期しない値に変わることがあります。バイトが書かれる順番は指定されておらず、変更される可能性があります。これによって、割り込みタイプが比較イベントをトリガするように設定されていてタイマ割り込みが有効な場合、割り込みが発生することがあります。A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル（c）のすべての RAM ページ ポインタ レジスタにも同じことが言えます。必要な場合は、すべての fastcall16 関数呼び出しにわたって、これらの値を保存してください。

Timer16_wReadCompareValue

説明：

Timer16 比較レジスタを読みます。

C プロトタイプ：

```
WORD  Timer16_wReadCompareValue(void);
```

アセンブラ：

```
lcall  Timer16_wReadCompareValue
mov   [wCompareValue], X           ; MSB returned in X
mov   [wCompareValue+1], A         ; LSB returned in A
```

パラメータ：

なし

戻り値：

wCompareValue: 比較レジスタの内容です。X レジスタでは MSB、累算器では LSB が渡されます。

副作用：

A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル（c）のすべての RAM ページ ポインタ レジスタにも同じことが言えます。必要な場合は、すべての fastcall16 関数呼び出しにわたって、これらの値を保存してください。

Timer16_wReadTimerSaveCV

説明：

比較レジスタを保持しながら、Timer16 カウント レジスタの現在値を読み取ります。これは、ソフトウェアによるハードウェア同期のカウンタキャプチャ操作を実行します。この関数は、比較レジスタの内容を保存する必要があるときにのみ使用します。比較レジスタの内容を保存する必要がない場合は、wReadTimer() 関数を使用することを推奨します。注：以前この API ルーチンは、wReadCounter という名称でした。

C プロトタイプ：

```
WORD  Timer16_wReadTimerSaveCV(void);
```

アセンブラ：

```
lcall  Timer16_wReadTimerSaveCV
mov    [wCount], X          ; MSB returned in X
mov    [wCount+1], A        ; LSB returned in A
```

パラメータ：

なし

戻り値：

wCount: カウントレジスタの内容です。X レジスタでは MSB、累算器では LSB が渡されます。

副作用：

カウントレジスタ値を読むために、その値は返される前に、比較レジスタに一時的に転送されなければなりません。これにより、すぐに、または次のタイマ入力クロックサイクルで比較条件が真となりますが、そのタイミングは、CompareType パラメータが 「以下」か 「未満」のどちらに設定されているかによって決まります。ユーザ モジュールとグローバル割り込みが有効になっている場合、割り込みが実行されます。そのタイミングは、この API 関数が呼び出し元へ戻る前、または比較レジスタが以前の状態に復元される前である場合が多くあります。割り込みは一時的に無効にされます。最後に、比較レジスタを復元するために、ユーザ モジュール自体が一時的に無効にされます。これにより、カウントレジスタが 1 つ以上のカウントをミスします。A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル（c）のすべての RAM ページ ポインタ レジスタにも同じことが言えます。必要な場合は、すべての fastcall16 関数呼び出しにわたって、これらの値を保存してください。

Timer16_wReadTimer

説明：

Timer16 カウントレジスタの現在値を読みます。これは、ソフトウェアによるハードウェア同期のカウンタキャプチャ操作を実行します。これは、比較レジスタを保存する必要がない場合のカウントレジスタを読み出す推奨手法です。注：この API ルーチンは、以前は wCaptureCounter という名称でした。

C プロトタイプ：

```
WORD  Timer16_wReadTimer(void);
```

アセンブラ :

```
lcall  Timer16_wReadTimer

mov   [wCount], X           ; MSB returned in X
mov   [wCount+1], A         ; LSB returned in A
```

パラメータ :

なし

戻り値 :

wCount: カウントレジスタの内容です。X レジスタでは MSB、累算器では LSB が渡されます。

副作用 :

比較レジスタの内容は消去されました。すぐに、または次のタイマ入力クロックサイクルで比較条件が真となりますが、そのタイミングは、CompareType パラメータが 「以下」か 「未満」のどちらに設定されているかによって決まります。ユーザ モジュールとグローバル割り込みが有効になっている場合、割り込みが実行されます。そのタイミングは、この API 関数が呼び出し元へ制御を返す前になることがあります。A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル （c) のすべての RAM ページ ポインタ レジスタにも同じことが言えます。必要な場合は、すべての fastcall16 関数呼び出しにわたって、これらの値を保存してください。

ファームウェア ソースコードの例

以下の例は、C とアセンブリ コード間の対比が単純で直接的な例です。期間および比較値として表示される値は、レジスタがゼロをベースにしており、ダウンカウント周期でゼロが最終カウントになるため、基本値から 1 だけずれる値になります。アセンブラおよび C コンパイラは、ユーザ モジュール API に対して、スタックではなく A レジスタで簡単な 1 バイトのパラメータを渡すことによってパフォーマンスを最適化します。C コンパイラは、Timer16.h ファイルに #pragma fastcall 宣言を見つけると、スタックで引数をプッシュする代わりに、「INT」タイプでこのメカニズムを導入します。

以下は、API の使用を示すアセンブリ言語ソースです。

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;  Description:
;    This sample shows how to capture an event with a bounded time limit.
;    The count resolution is 30.5 us, with a bounded time limit of 1.99
;    seconds.
;
;    The interrupt should be set to interrupt on the Compare - Less than
;    equal. The capture input should be connected to the event that is
;    being measured. The clock should be connected to the 32.768K internal
;    clock.
;
;    Computed time lapse is:  (65,535 - wCount ) / 32,768.
;
;    The foreground routine sets and starts the timer. The interrupt
;    level routine captures the value.
;
;  Parameters: none
;  Returns:    none
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

include "m8c.inc"       ; part specific constants and macros
```

```
include "memory.inc"    ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules

export _main

area bss (RAM,REL)
_wElapsedTime::
 wElapsedTime::    BLK   2

area text(ROM, REL, CON)
_main:

CapturePulse::
    mov   [wElapsedTime], 0
    mov   [wElapsedTime+1], 0
    mov   A, FFh                ; set the period to the maximum
    mov   X, FFh
    lcall  Timer16_WritePeriod
    mov   X, 0                  ; set the compare to trigger at 1
    mov   A, 0
    lcall  Timer16_WriteCompareValue
    lcall  Timer16_EnableInt     ; enable the timer interrupt mask
    M8C_EnableGInt              ; enable global interrupts
    lcall  Timer16_Start         ; start the timer - timer will start to
.WaitForCapture:
    mov   A, [wElapsedTime]
    or    A, [wElapsedTime+1]
    jz    .WaitForCapture

;Evaluate captured value here!
;If wElapsedTime is not > 1 then compute elapsed time.
;else if wElapsedTime is 1 or 0 then event did not occur within
;time limit.

.terminate:
    jmp .terminate
```

ファイル *Timer16int.asm* に位置する割り込みレベルのルーチンは次の通りです。

```
_Timer16_ISR:

   ;@PSoC_UserCode_BODY@ (Do not change this line.)
   ;---------------------------------------------------
   ; Insert your custom code below this banner
   ;---------------------------------------------------
   ;   NOTE: interrupt service routines must preserve
   ;   the values of the A and X CPU registers.
    push  X
    push  A
    call  Timer16_wReadTimer
    mov   [_wElapsedTime+1], A
    mov   [_wElapsedTime], X
    call  Timer16_Stop
    pop   A
    pop   X
```

```
   ;-----------------------------------------------
   ; Insert your custom code above this banner
   ;-----------------------------------------------
   ;@PSoC_UserCode_END@ (Do not change this line.)

   reti
```

C での同じコードは以下のようになります。注 : 割り込みルーチンはアセンブリ言語で書かなければなりません。

```
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules

WORD wElapsedTime;

void main(void)
{
    Timer16_WritePeriod(0xffff);
    Timer16_WriteCompareValue(0x0001);
    Timer16_EnableInt();
    M8C_EnableGInt;
    Timer16_Start();
    while( wElapsedTime == 0 );
}
```

設定レジスタ

16-bit タイマは、2 つのデジタル PSoC ブロックを使用します。左から右への配置順では、Timer16_LSB および Timer16_MSB. という名前が付けられ、7 つのレジスタを通して各ブロックがパーソナライズされ、パラメータ化されます。以下の表では、定数としての「パーソナリティ」値あるいは名前が付けられたビットフィールドとしてのパラメータおよびその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イス ファイル (「.h」および 「.inc」ファイル) で定義されます。

Table 3.　関数レジスタ、バンク 1

ブロック / ビット	7	6	5	4	3	2	1	0
MSB	0	0	1	比較タイプ	割り込みタイプ	0	0	0
LSB	0	0	1	比較タイプ	0	0	0	0

Table 4.　関数レジスタ、バンク 1

ブロック / ビット	7	6	5	4	3	2	1	0
MSB	0	0	1	比較タイプ	割り込みタイプ	0	0	0
LSB	データ反転	BCEN	1	比較タイプ	0	0	0	0

BCEN は、最終カウント出力を未処理のブロードキャスト バス ラインに出すかどうかゲートします。このビット フィールドは、ブロードキャスト ラインを直接設定して、デバイス エディタで設定されます。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データの反転フラグは、キャプチャ入力信号のセンスを制御します。CompareType フラグは、比較関数が 「以下」または「未満」のどちらに設定されるかを示します。InterruptType フラグは、比較イベントまたはカウント終了のどちらで割り込みをトリガするかを決定します (カウントレジスタの 「CaptureInt」も参照してください)。CompareType と InterruptType は、このトピックで前述されているように、ユーザ モジュール パラメータを通して、デバイス エディタで直接設定されます。

Table 5. 入力レジスタ、バンク 1

ブロック / ビット	7	6	5	4	3	2	1	0
MSB	0	0	1	1	クロック			
LSB	キャプチャ				クロック			

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

Table 6. 出力レジスタ、バンク 1

ブロック / ビット	7	6	5	4	3	2	1	0
MSB	0	0	0	0	0	Out Enable	OutputSelect	
LSB	0	0	0	0	0	0	0	0

Table 7. 出力レジスタ、バンク 1

ブロック / ビット	7	6	5	4	3	2	1	0
MSB	AuxClk		AuxEnable	AuxSelect		OutEnable	OutputSelect	
LSB	AuxClk		0	0	0	0	0	0

デバイス エディタのユーザ モジュール 「ClockSync」パラメータは、AuxClk ビットの値を決定します。同じような名前が付いていますが、AuxEnable および AuxSelect は、OutEnable および OutSelect ビットフィールドにそれぞれ関係しています。AuxEnable および AuxSelect は、行出力バスのいずれかで比較出力信号の駆動を許可し、デバイス エディタの相互接続ビューでグラフによる行バス操作を使って制御されます。OutEnable は、最終カウント出力が未処理のバスまたはグローバル出力バスのいずれかに駆動される場合に設定されます。OutputSelect は、比較出力からどちらのバスが駆動されるかを制御します。

Table 8. カウント レジスタ (DR0)、バンク 0

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

[+] Feedback

カウント レジスタは、イネーブル入力がアクティブであるクロック周期ごとに、1 減少する 16-bit ダウン カウント値です。その値は、カウント終了に続くクロック周期で、期間レジスタの内容からロードされます (ゼロ値)。この値は、Timer16 API を使用して読み取れます。

Table 9. 期間レジスタ (DR1)、バンク 0

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

期間レジスタは、デバイス エディタまたは Timer16 API によって設定できる書き込み専用レジスタです。書き込まれると、ユーザ モジュールが API を通して無効になっている場合は、値がカウント レジスタに転送されます。その値は、カウント終了に続くクロック周期で、カウント レジスタに自動的にコピーされます。

Table 10. 比較レジスタ (DR2)、バンク 0

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

比較レジスタは、比較出力を生成するために、どのカウント レジスタがテストされるかに対する値を保持します。この値は、デバイス エディタおよび Timer16 API で設定できます。

Table 11. 制御レジスタ (CR0)、バンク 0

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

Start/Stop は、設定された場合に Timer16 を有効にし、クリアされた場合に無効にすることを示します。Timer16API を使って変更できます。

改訂履歴

バージョン	作成者	説明
2.6	TDU	クロックの説明に関する更新：ブロックで外部デジタル 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、入力の同期がオフになります。

Note PSoC Designer 5.1 より、全ユーザ モジュールの データシートのバージョン履歴が追加されます このセクションは、現在および以前のユーザ モジュール バージョン間の差分の概要を説明するものです。

Copyright © 2000-2011 © 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。