AN72382

PSoC® 3 および PSoC 5LP GPIO ピンの使用

著者: Greg Reynolds
関連プロジェクト: なし
関連製品ファミリ: すべての PSoC® 3 および PSoC 5LP デバイス
ソフトウェア バージョン: PSoC Creator™ 2.1 SP1 以降
関連アプリケーション ノートの完全なリストについては、ここをクリックしてください。

AN72382 は PSoC® 3 および PSoC 5LP で GPIO ピンを効果的に使用する方法について説明します。主な内容は GPIO の概要、設定、混合信号の用途、レジスタ、割り込みや低電力時の動作を含みます。

目次

1 はじめに 1
2 GPIO ピンの基本情報 2
2.1 GPIO ピンの物理構造 2
2.2 デジタル システム相互接続の概要 2
2.3 アナログ ルーティングの概要 3
2.4 GPIO 電源の構造および制限 3
2.5 V_{DDA}、V_{DDD} および V_{DDIO} の相対電圧 4
2.6 スタートアップと低電力での挙動 5
2.7 GPIO ピンへの DMA アクセス 5
2.8 ポート割り込み制御ユニット 5
3 PSoC Creator の GPIO ピン 6
3.1 PSoC Creator の API 6
3.2 ピン コンポーネント シンボルおよびマクロ 6
3.3 ピン コンポーネント割り込み 7
3.4 外部端子 8
3.5 手動のピン割り当て 8
4 GPIO の例、ヒントと要領 9
4.1 GPIO「Hello World」プロジェクト 9
4.2 入力の読み出しと出力への書き込み 9
4.3 複数の GPIO ピンを論理ポートとして追加 10
4.4 GPIO 出力イネーブル ロジックを設定 12
4.5 設定可能な XRES 機能を有効化 14
4.6 GPIO ピンのデバッグ ロジックを無効化 14
4.7 データ レジスタで GPIO をより速くトグル 15
4.8 8051 特殊機能レジスタを使用 16
4.9 GPIO でアナログとデジタル両方を使用 17
4.10 ハードウェアによるアナログスイッチングの制御 19
4.11 DSI をクロック ソースとして使用 21
4.12 ファームウェアを使用して PICU 設定を変更 24
4.13 より大きな駆動／シンク電流のためにピンを連動 25
4.14 レベルシフト信号 26
5 関連アプリケーション ノート 27
付録 A. GPIO の API およびレジスタの参照情報 28
A.1 コンポーネント API 28
A.2 ピン単位 API 28
A.3 GPIO レジスタ 29
A.4 不揮発性ラッチ 30
付録 B. PSoC Creator の設定とレジスタ 31
改訂履歴 42

1 はじめに

PSoC® 3 および PSoC 5LP の GPIO において可能な任意の信号とピン間のルーティングにより、プリント基板レイアウトの最適化と設計に必要な時間の短縮が可能になり、またはんだ付けを用いないプリント基板の大幅な改変が可能となります。しかしながら、この自由度により、従来のマイクロコントローラーの場合に比べて、急速な習得カーブが得られます。本アプリケーション ノートでは、PSoC 3 および PSoC 5LP の GPIO の基本情報について紹介し、設計時に効果的に使用する手法を説明します。

読者が PSoC Creator™、PSoC 3 および PSoC 5LP ファミリのデバイス アーキテクチャに精通していることを前提としています。PSoC が初めての方は、AN54181 – Getting Started with PSoC 3 および AN77759 – Getting Started with PSoC 5LP の序文をご参照ください。PSoC Creator が初めての方は、PSoC Creator ホームページを参照してください。

関連の PSoC デザイン リソースの一覧は関連アプリケーション ノート の節をご覧ください。

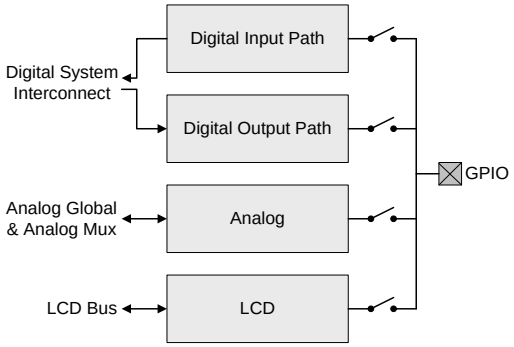
2 GPIO ピンの基本情報

PSoC 3 および PSoC 5LP デバイスでは、GPIO、SIO および USB ピンは類似です。しかしながら、GPIO ピンと違って、SIO および USB ピンは異なる駆動能力およびアプリケーション特有の機能を備えています。一部の GPIO ピンは、オペアンプ入力と出力、プログラミングおよびデバッグ インターフェースや DAC 出力などの、二次的専用機能も備えています。特別な機能に使用されない時、すべての GPIO ピンは同じ動作をします。パッケージ タイプによって、PSoC デバイスは最大 62 個の GPIO ピンを持ち得ます。

2.1 GPIO ピンの物理構造

GPIO ピンは 8 つの駆動モードを持ち、PSoC が提供する多数のアナログおよびデジタル I/O 機能をサポートします。GPIO 構造の詳細なブロック図は、PSoC 3 Architecture Technical Reference Manual (TRM)、および PSoC 3 と PSoC 5LP ファミリのデータシートにあります。図 1 は簡略化したバージョンを示します。

図 1. 簡略化した GPIO ブロック図

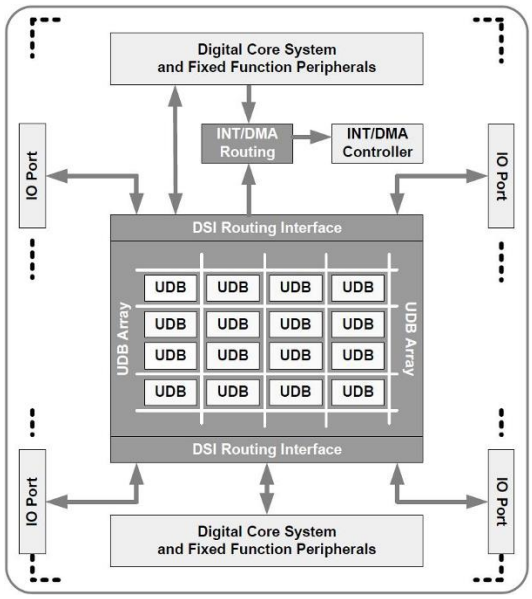


様々な駆動モードおよびそのカスタム設定はピン コンポーネント データシートで詳細に説明されています。このデータシートは PSoC Creator の一部として提供されており、サイプレスのウェブサイトから個別にダウンロードもできます。

2.2 デジタル システム相互接続の概要

PSoC 3 および PSoC 5LP デジタル サブシステムはプログラマブルな相互接続機能を持っており、内蔵ペリフェラル、カスタム論理機能 (ユニバーサル デジタル ブロック (UDB))と、任意の I/O ピンとの接続を可能にします。デジタル システム インターコネクト (DSI) ルーティング インターフェースにより、図 2 に示しているように、GPIO ピンをチップ内の任意のデジタル リソースに接続することができます。

図 2. DSI ブロック図

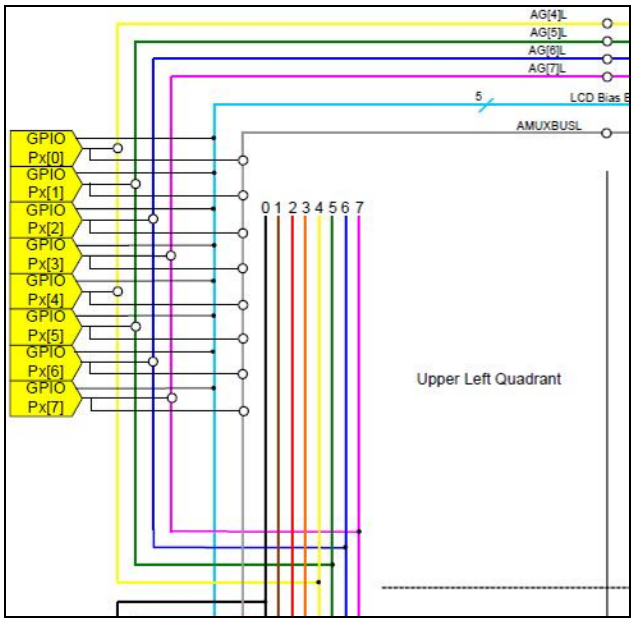


すべてのデジタル リソースは DSI につながり、お互いに接続するか、またはシステムコアに接続します。DSI 動作の詳細については、TRM の「UDB アレイおよびデジタル システム相互接続」の節を参照してください。

2.3 アナログ ルーティングの概要

GPIO ピンは、アナログ リソースに、または相互に接続されます。これはスイッチとマルチプレクサで結合される一連のアナログ ルーティング バスを介して行われます。2 つの主要なアナログ ルーティング バスは、アナログ グローバル (AG) バスおよびアナログマルチプレクサ (AMUX) バスです。AG バスは 4 つの象限 (AGL0～4、AGL4～7、AGR0～4 および AGR4～7) に分割され、AMUX バスは 2 つの象限 (AMUXL および AMUXR) に分割されます。図 3 に、TRM のアナログ配線図の一部を示しています。

図 3. 左上象限のアナログ配線



各 AGx は各象限内の関連ポートの 2 個のピンに接続できます。そして各 AMUX はチップの半分のすべてのピンに接続できます。アナログ バスは、コンパレータ、DAC および ADC などのような、様々なアナログリソースの入力と出力の両方に、あるいは片方に接続できます。それに加えて、スイッチは、左側のバスと右側のバスを相互に接続することを可能にします。

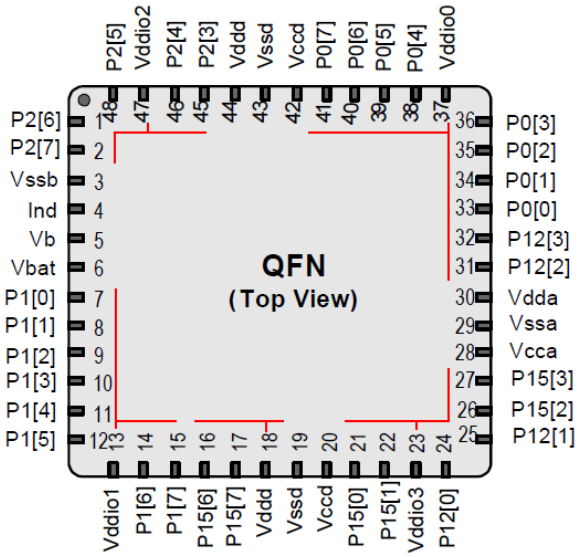
PSoC 3 および PSoC 5LP デバイスのアナログ配線システムの詳細な説明が TRM の「アナログ配線」節にあります。アプリケーション ノート AN58304 および AN58827 はアナログ配線およびピン選択を詳細に説明します。

2.4 GPIO 電源の構造および制限

一般的に、GPIO ピンはソース電流が 4mA で、シンク電流が 8mA です。これらのピンを集結 (短絡) することにより、単一のピンが供給できるより高いソース／シンク電流が可能になります。しかし、設計者は更なる電源の制限を考慮する必要があります。

PSoC 3 および PSoC 5LP デバイスは、V_{DDIO} ピンを通して最大 4 個の独立した I/O 電圧ドメインを提供しています。PSoC 3 および PSoC 5LP データシートでは、特定のピン セットに電源を供給する V_{DDIO} ピンは、ピン配置図に描かれた実線で示されています。図 4 に 48 ピンの PSoC 3 デバイスを示し、その中で V_{DDIO} 象限インジケータが赤色でハイライトされています。

図 4. 赤色でハイライトされた VDDIO 象限の例



V_{DDIO} ピンは、多くの場合、V_{DD} と同じ電源レールに接続されます。個々の V_{DDIO} 象限におけるソース電流とシンク電流がどれくらいであるかについては殆ど検討されていませんが、制限があります。表 1 に、PSoC ファミリとパッケージ タイプ別の制限を示します。

表 1. V_{DDIO} 象限の電源の制限

ファミリ	パッケージ	ソース	シンク
PSoC 3 PSoC 5LP	100 ピン 68 ピン	V_{DDIO} 毎に 100mA	V_{DDIO} 毎に 100mA
	48 ピン	100mA VDDIO0+VDDIO2 100mA VDDIO1+VDDIO3	100mA VDDIO0+VDDIO2 100mA VDDIO1+VDDIO3
PSoC 5	100 ピン 68 ピン	V_{DDIO} 毎に 20mA	V_{DDIO} 毎に 20mA
	48 ピン	20mA VDDIO0+VDDIO2 20mA VDDIO1+VDDIO3	20mA VDDIO0+VDDIO2 20mA VDDIO1+VDDIO3
注: ソース電流とシンク電流の合計は、任意の V_{DDIO} 象限 (あるいは 48 ピンパッケージの V_{DDIO} ペア) に対して、100mA を超えるべきではありません。			

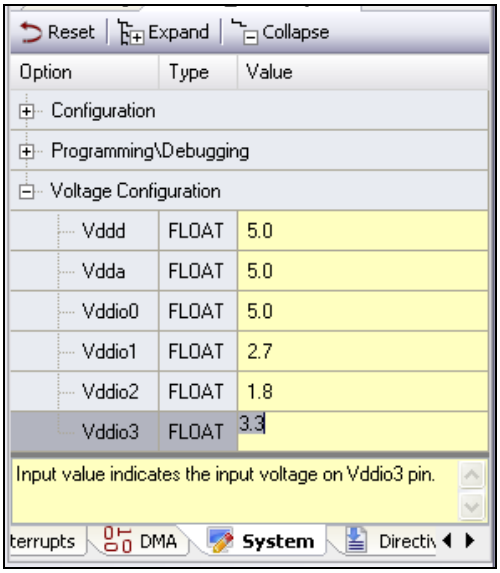
GPIO ピンによるソースとシンク電流の標準値が制限の 80%を超えると見込まれるアプリケーションでは、GPIO ピンのどの単一の象限も、最悪の動作条件下でも、その最大値を超えないことを確認します。そうすると、デザインは個別の VDDIO 象限内のピンを使用することが必要となり、電流が分散されることになります。

2.5 V_{DDA}、V_{DDD} および V_{DDIO} の相対電圧

V_{DDA} は PSoC 3 または PSoC 5LP デバイス上で最も高い電圧でなければなりません。他のすべての電源供給ピンは V_{DDA} 以下でなければなりません。 V_{DDD} と V_{DDIO} ピンは相互に対して、より低いか、より高いか、または等しいです。

Design-Wide Resource ファイルの **System** タブには電圧コンフィギュレーションセクションがあり、ユーザーは各々のパワードメインが動作する電圧を定義することになります。これらのフィールドに入力された値は、図 5 に示すように、PSoC Creator により使用されます。ただしそれはコンポーネントまたは機能が、動作する電圧に依存する場合です。

図 5. Design-Wide Resource での電圧コンフィグレーション



PSoC Creator での適正な電圧コンフィギュレーションは、すべての場合に推奨されます。それはどのコンポーネントまたは機能が使用されるかにかかわりません。

2.6 スタートアップと低電力での挙動

デフォルトで、すべての GPIO ピンはアナログ HI-Z 状態で起動し、リセットが解除されるまでその状態を維持します。各ピンの初期動作設定が起動中にロードされ、その時から有効になります。ユーザーは、不揮発性ラッチ アレイの PRTxRDM フィールドを使用して、GPIO のリセット動作を変更できます。このフィールドは、PSoC デバイスがプログラムされる時に、書き込まれます。

すべての低電力モードにおいて、GPIO ピンは、デバイスがリセットされるかまたはウェイクアップされるまで、その状態を保持します。ポート割り込み論理が、すべての低電力モードで、機能し続けますので、ピンをウェイクアップ ソースとして使用できます。

注: コントロール レジスタなどの UDB ベースのコンポーネントは、スリープ またはハイバネート モードの間は、通常非アクティブです。PSoC デバイスがこれらのモードに入る時または出る時に、そのコンポーネントにグリッチが発生する可能性があります。このグリッチにより、GPIO が望ましくない状態にセットされることもあります。それを回避するためには、PSoC デバイスが低電力モードに移行する前に、ピンを HIGH または LOW 論理状態に明確にセットします。

2.7 GPIO ピンへの DMA アクセス

PSoC デバイスには DMA コントローラーがあり、I/O インターフェースを含む異なる内部ペリフェラルに接続します。GPIO レジスタがメモリでアドレス指定されるため、DMA 転送が使われることにより、GPIO ピンを設定し、CPU による動作を必要とせずにデジタル出力パスにデータを書き込むことができます。

DMA 設定およびデータ転送は非常に複雑なので本アプリケーション ノートでは説明しません。他のいくつかのアプリケーション ノートとコード用例が用意されています。そこには AN52705 – PSoC 3 and PSoC 5LP – Getting Started with DMA が含まれています。

2.8 ポート割り込み制御ユニット

PSoC 3 および PSoC 5LP は、I/O 割り込みを管理するポート割り込み制御ユニット (PICU) を備えています。各 GPIO ピンは、立ち上がりエッジ、立ち下がりエッジ、またはどちらのエッジ状態上にも、割り込みを生成することができます。レベル センシティブな割り込みは、cy_isr コンポーネントをピン コンポーネントの割り込み端子に接続することで、実行されます。

GPIO 割り込みがトリガされると、その GPIO のステータス レジスタ内の対応するビットが「1」にセットされます。レジスタが読み出されるか、またはチップ リセットが発生するまで、このビットは「1」のままです。PSoC Creator により供給される API は、GPIO 割込みの設定および通知を管理します。

1 つのポート内にある個々の GPIO 割り込み信号は一緒に OR されて、1 つの PICU 要求信号が割り込みコントローラーに送られます。ポート割り込み要求がデイジー チェーンで繋がって、単一のウェイクアップ信号を生成します。この信号は PSoC パワー マネージャに送られます。PICU はすべての低電力モードでアクティブの状態を維持しますが、個々の GPIO 割り込みはウェイクアップ後も管理されたままでなければなりません。

3 PSoC Creator の GPIO ピン

本節は、PSoC Creator を使用して GPIO ピンを設定・操作する方法について説明します。PSoC Creator はテキストとグラフィカル エディッティング インターフェースを組み合わせます。それにより設計者はハードウェア構成とファームウェアへの書き込みを同時に設定できます。

3.1 PSoC Creator の API

サイプレスは、ユーザーがファームウェアを通して GPIO を動的に制御できるような、API のセットを提供します。ピン コンポーネントの API は、コンポーネント単位とピン単位の両方で、アクセスできます。

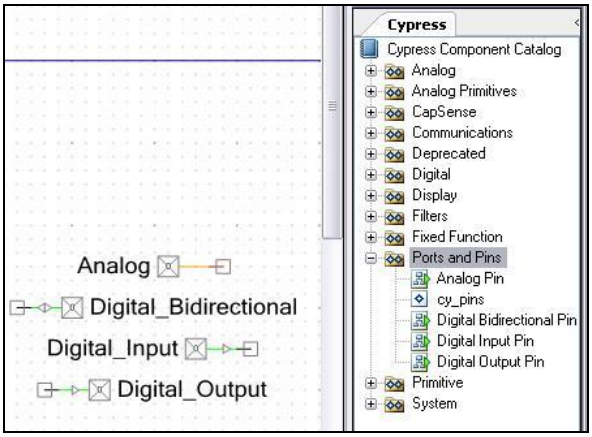
cy_boot コンポーネントもチップ リソースにアクセスする機能を提供します。cy_boot の機能は個々のコンポーネント ライブラリの一部ではありませんが、ライブラリはそれを使用できます。ピン単位の API は、 *cypins.h* ファイルの cy_boot の一部として提供されますが、PSoC Creator System Reference Guide のピンのセクションで記述されます。ユーザーは、これらの API を使って、各物理ピンのコンフィギュレーション レジスタを制御できます。

GPIO に関連する API の要約と簡単な コード用例については、付録 A GPIO の API およびレジスタの参照情報を参照してください。

3.2 ピン コンポーネント シンボルおよびマクロ

cy_pins コンポーネントは、内部 PSoC リソースを物理ピンに接続するための推奨方法です。これにより、PSoC Creator が、指定されたピン設定に基づき、信号を PSoC 内に自動的に配置・配線できます。標準のサイプレス コンポーネント カタログは、シンボルのポートおよびピンのクラスに、4 種類の定義された GPIO 設定 (マクロ) があります。それらはアナログ、デジタル双方向、デジタル入力、およびデジタル出力です。図 6 に示しているように、これらのコンポーネント マクロのひとつを回路図にドラッグして、ピンをプロジェクトに加えます。

図 6. PSoC Creator でのピン コンポーネント シンボル タイプ



どのマクロ シンボルを選択するかによって、ピン設定が限定されるわけではありません。回路図にピン シンボルを配置した後、本書で説明するコンポーネント カスタマイザー オプションを使って、ピンの動作を設定できます。

3.3 ピン コンポーネント割り込み

図 7 に示しているように、PSoC Creator の cy_pins 設定ダイアログを用いて、ピン コンポーネントでの割り込みを有効化できます。ピンコンポーネントをダブルクリックして、開きます。

図 7.PSoC Creator での割り込み設定

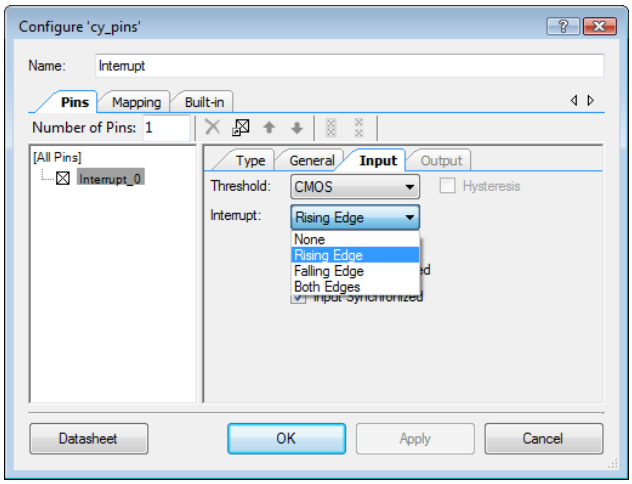
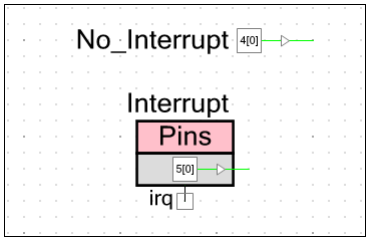


図 8 に示しているように、割り込みが有効になると、ピン コンポーネント シンボルが変わります。ピン割り込みがトリガされると、ピン コンポーネントの IRQ 信号がトグルします。ピン割り込みを有効にするために irq 端子を isr コンポーネントに接続する必要はありません。

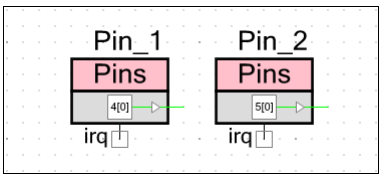
図 8. 割り込みの有効化に伴うピン コンポーネント シンボルの変化



割り込みが有効になった場合、1 つの物理 GPIO ポートには 1 つだけのピン コンポーネントを使えます。この制限の理由は、1 個のポートの全てのピン割り込みが OR されて、1 つの IRQ 信号のみが回路図に表示されるためです。

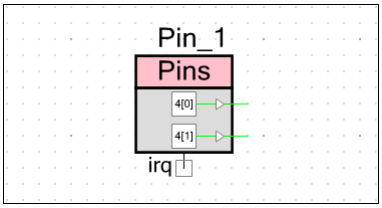
例えば、図 9 に示すように、割り込みが有効化された 2 つのピン コンポーネントを考えましょう。これらのコンポーネントは同じ物理ポートのピンにマッピングされることができません。理由は、PSoC Creator 回路図上に 2 つの個別 IRQ 信号があるのに、ポート全体に物理 PICU 割り込みが 1 つしか生成されないためです。

図 9. 割り込みを有する 2 つのピン コンポーネント



この 2 つのコンポーネントを同じポートに割り当てようとすると、PSoC Creator はエラーを発生します。容認される方法は、図 10 に示すように、同じコンポーネントに複数のピンを割り当てることです。これにより、回路図上でその物理ポートに対して 1 つのみの IRQ 信号があることが確保されます。さらに各々のピンにその独自の割り込みエッジ タイプを割り当てることができます。唯一の制限はピンがつながっていなければならないことです。

図 10. 異なる割り込み エッジ タイプのピンの選択

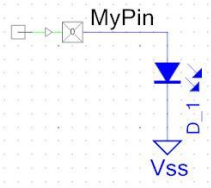


ユーザーはまた、任意の GPIO ピンへの割込みを有効化または変更するために、コンポーネント設定に関わらず、PICUx_INTTYPEy レジスタを使うことができます。このレジスタの詳細については付録を参照してください。

3.4 外部端子

cy_pins 設定ダイアログは外部端子を表示するオプションを提供します。それによって、ユーザーはオフチップのコンポーネントを回路図に追加して、そのコンポーネントとピンの接続を表示することができます。図 11 に、ピン コンポーネントがオフチップの LED を駆動する例を示します。

図 11. オフチップのコンポーネント接続例

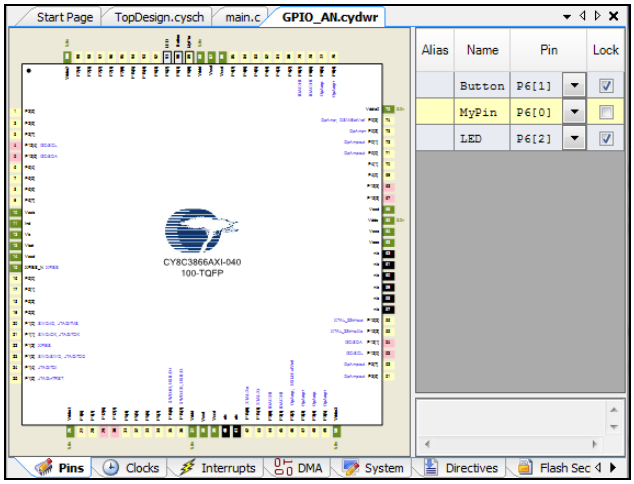


3.5 手動のピン割り当て

あるピン コンポーネントは、Design-Wide Resources インターフェース (*cydwr*) の **Pins** タブを介して、ひとつの物理ピンに割り当てられます。ユーザーが指定しない場合は、PSoC Creator は自動的にピンを割り当てます。しかしこれにより、プリント基板上に配線しにくいようなピン配置につながる可能性があります。また、一部の GPIO ピンはアナログまたはデジタル リソースに直接接続されます。

図 12 に、3 つの割り当てられたピンを示しています。灰色にハイライトされたピンは手動で、そして黄色にハイライトされたピンは自動で割り当てられました。**Lock** オプションを選択することで、ピンが PSoC Creator により再割り当てられることを回避できます。

図 12. cydwr ウィンドウでのピン割り当て



PSoC Creator は必要に応じてのピンの再割り当てを容易にしますが、設計者は基板を設計する前にピンの選択を考慮する必要があります。TRM、AN58304 および AN58827 での「アナログ相互接続」図は、最適なアナログ ピン選択を決める際に価値のあるリソースとなります。

4 GPIO の例、ヒントと要領

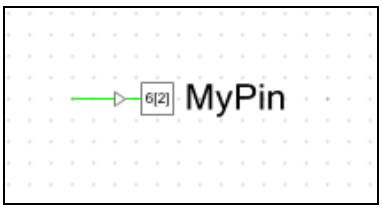
本節は GPIO ピン使用法の実例を述べています。例は PSoC 3 デバイス用に作成されますが、同じ技術が PSoC 5LP に適用されます。基本の例とより高度な技術が両方とも含まれています。

4.1 GPIO「Hello World」プロジェクト

GPIO の最も単純な用途は、ピンの出力状態を HIGH か LOW に設定することです。この例は、ピン コンポーネント API を使って出力を設定する方法を示します。

1. ストロング駆動モードに設定された 1 個のデジタル出力ピン コンポーネントを、プロジェクトの回路図に入れます。図 13 を参照ください。
2. コンポーネントに「MyPin」と名前を付けて、そのコンポーネントを P6[2]に割り当てます。

図 13.Hello World 例示回路図



3. 以下のように *main.c* でコンポーネント API を使って出力をトグルします。

```
for(;;)
{
   /* Set MyPin output state to HIGH */
   MyPin_Write(1);

   /* Delay for 500 ms */
   CyDelay(500);

   /* Set MyPin output state to LOW */
   MyPin_Write(0);

   /* Delay for 500 ms */
   CyDelay(500);
}
```

4. プロジェクトをビルドして、PSoC デバイスをプログラムします。

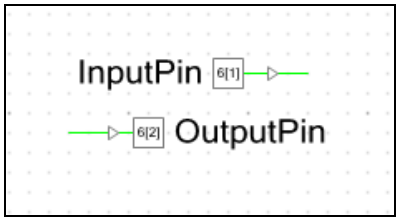
   その結果として、500 ミリ秒おきに HIGH と LOW の間でトグルする出力が出ます。

4.2 入力の読み出しと出力への書き込み

この例は、コンポーネント API を使って、GPIO の読み出し／書き込み方法を示します。出力ピンは入力ピンの状態の反転を駆動します。

1. 2 本のピン (1 本のデジタル入力ピンと 1 本のデジタル出力ピン) をプロジェクト回路図に配置します。図 14 を参照ください。

図 14.入力と出力の回路図例示



2. 次のように、コンポーネント API を使って入力ピンに基づいて出力ピンの状態を設定します。

```
for(;;)
{
   /* Set OutputPin state to the
      inverse of the InputPin state */
   OutputPin_Write( ~InputPin_Read() );
}
```

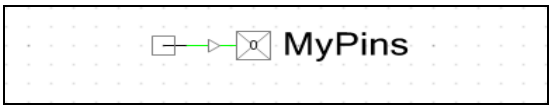
その結果として、出力ピンは常に入力ピンの逆の状態となります。

4.3 複数の GPIO ピンを論理ポートとして追加

PSoC Creator では、ユーザーは最大 64 個のピンのグループを 1 個の論理ポートに配置できます。これらのグループは、ポートの定義した名前としてコード中で参照されます。全てのピンは 1 つの同じ物理ポートに属することもあり、もしくは幾つかの別々のの物理ポートからくることもあります。

1. 図 15 に示しているように、単一のピン シンボルを配置します。

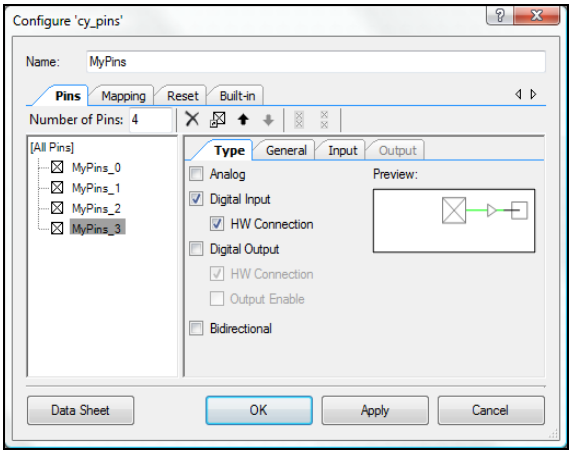
図 15. 回路図に配置された単一のピン シンボル



2. ピン シンボルをダブルクリックして、ピン カスタマイザ ウィンドウを開きます。
3. コンフィギュレーション ウィンドウの **Number of Pins** フィールドにピン数を入力します。

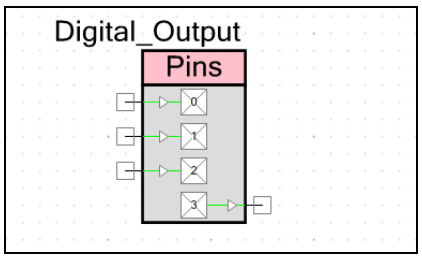
   それらのピンは、フィールドの下にあるリストに表示されます。リスト内のそれぞれのピンを個別に選択し、各ピンがお互いから独立してカスタマイズされるようにします。「**All Pins**」を選択し、ポート内のすべてのピンに作用するようにします。
4. この例では、これらのピンの内の 3 個をデジタル出力ピンとして設定します。図 16 に示すように、残りの 1 個のピンをデジタル入力として設定します。

図 16. デジタル入力として配置された 4 ピンの内の 1 本



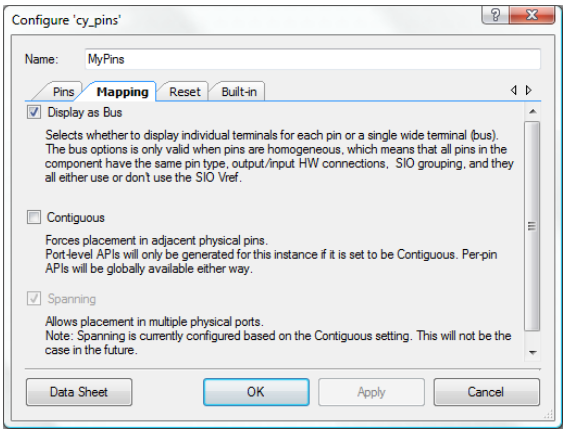
5. **OK** をクリックして変更を適用します。
   ピンの数とタイプを定義した後、回路図のシンボルは図 17 のようになります。

図 17. ポート コンフィギュレーション中のピン コンポーネント



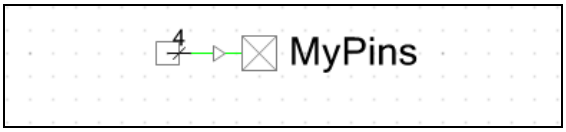
6. (任意) 図 18 と図 19 に示しているように、 ピン コンフィギュレーション ウィンドウの **Mapping** タブの **Display as Bus** を選択して、ポートをバス シンボルとして表示します。

図 18. バス オプションとして表示



この機能はポートの挙動に影響を与えません。
注: すべてのピンは、バスとして表示するために、同じタイプでなければなりません。

図 19. ポート バス シンボルとして表示された 4 個のピン



7. (任意) 図 20 に示しているように、**Mapping** タブの **Contiguous** を選択して、ピンを物理的に隣接させます。

図 20. 隣接ピン配置のオプション

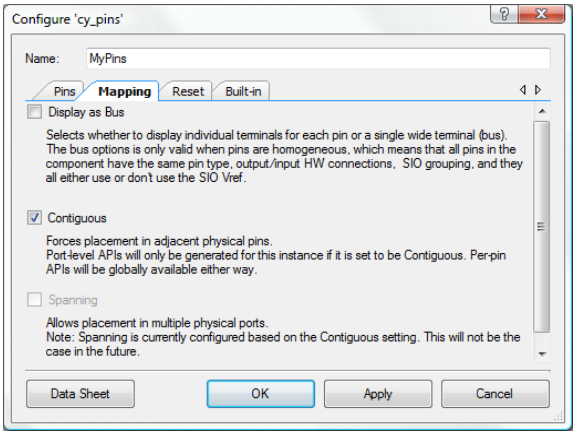
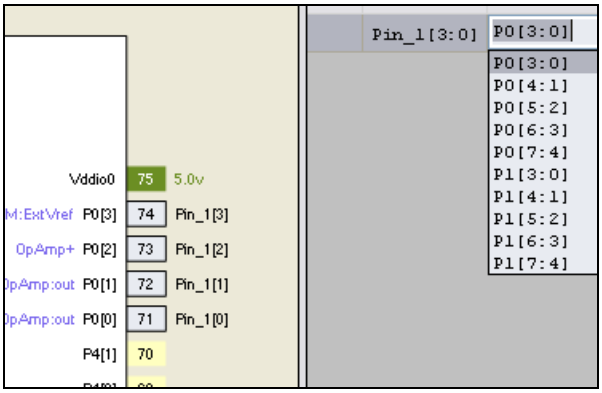


図 21 に示しているように、Contiguous を選択した時、PSoC Creator は使用可能なピン配置オプションのリストを、ポートの設定とマッチするよう調整します。

図 21. 隣接したポートピンのピン配置



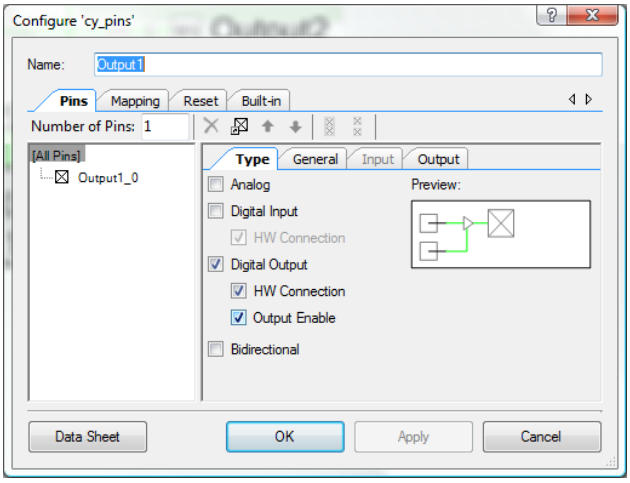
これらの機能は、ピン コンフィギュレーション ウィンドウとピン コンポーネント データシートに、詳述されています。

4.4 GPIO 出力イネーブル ロジックを設定

この例は、GPIO ピンの出力イネーブル ロジックを設定・使用する方法を示しします。

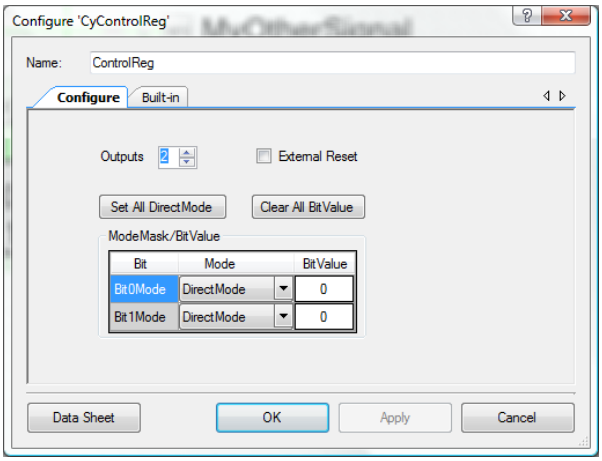
1. 2 個のデジタル出力ピンをプロジェクト回路図に配置します。
2. 図 22 に示しているように、それぞれのピンのコンフィギュレーション ダイアログを開いて、**Output Enable** ボックスにチェックを入れます。

図 22. 出力イネーブルの選択



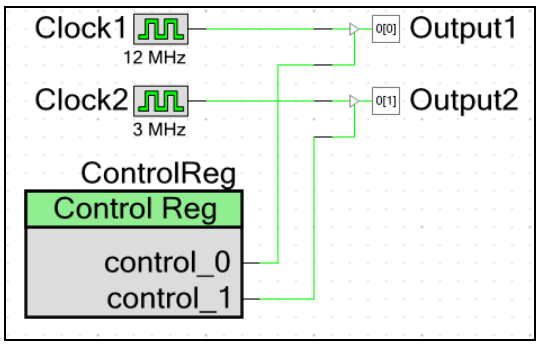
3. コントロール レジスタを回路図に配置します。
4. 図 23 に示しているように、2 個の出力を持つようにコントロール レジスタを設定します。

図 23. 2 個の出力で設定されたコントロール レジスタ



5. 任意の方法で設定された 2 個のクロック コンポーネントを加えます。
6. 図 24 に示すように、クロックをピンに接続します。

図 24. 出力イネーブルを駆動するコントロール レジスタ



7. *main.c* ファイルに次のコードを加えます:

```
for(;;)
{
    for( i=0; i<=3; i++ )
    {
        ControlReg_Write(i);
        CyDelay(500);
    }
}
```

8. PSoC 3 または PSoC 5LP デバイスをコンパイルしプログラムします。

   その結果は、ControlReg の状態によってゲートで制御された 2 個のピンの出力となります。

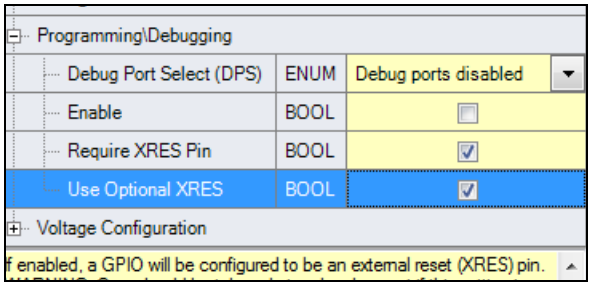
   これと同じ方法は、双方向データバスの作成に使用できます。 データバスに必要な数のピンを配置し、各ピン コンポーネントの出力イネーブル ボックスをチェックします。 すべての出力イネーブル信号を 1 つのコント ロールレジスタ出力に接続します。 出力イネーブルが有効の場合、GPIO ピンはストロング駆動出力されます。 出力イネーブルが無効の場合、GPIO ピンの入力ロジック状態を読み取れます。

4.5 設定可能な XRES 機能を有効化

この例は設定可能な XRES 機能を有効化する方法を示します。P1[2]ピンを随意 (選択自由) の XRES ピンとして設定し、小型パッケージの外部リセットをサポートします。この機能はより大きいパッケージでも使用可能です。

1. 図 25 に示しているように、Design-Wide Resources ファイル内の **System** タブを開きます。
2. **Use Optional XRES** オプションを選択し、随意の XRES ロジックを有効化します。このボックスが選択された場合、P1[2]は GPIO ピンとしての機能を停止し、内部プルアップを有するアクティブ LOW 入力として設定されます。

図 25. 随意の XRES ピンの有効化



3. PSoC デバイスをプログラムして、不揮発性アレイに設定を書き込みます。この設定は次の電源投入後に有効になります。
4. **Use Optional XRES** オプションの選択を解除して、通常の GPIO 機能を復元するために、PSoC デバイスを再プログラムします。

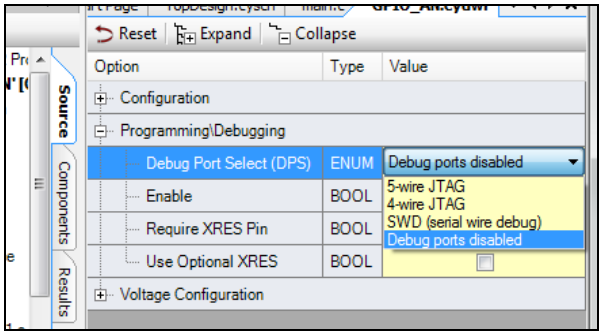
すべての PSoC 3 および PSoC 5LP デバイスは、工場出荷時には随意の XRES 機能が無効にされていることに注意してください。設定可能な XRES ピンの使用は専用 XRES ピンの機能を変更しません。

4.6 GPIO ピンのデバッグ ロジックを無効化

この例はポート 1 のピンに対応するデバッグ ロジックを無効にする方法を示します。デバッグ ポート機能が有効化された時、起動時にこれらのピンに動作を検出すれば、PSoC デバイスはデバッグ モードに入ります。

1. Design-Wide Resources ファイルを開いて、**System** タブをクリックします。
2. 図 26 に示すように、ドロップダウン メニューから **Debug ports disabled** を選択します。

図 26. 無効化されたデバッグ ポート



3. PSoC 3 または PSoC 5LP デバイスをコンパイルしプログラムします。

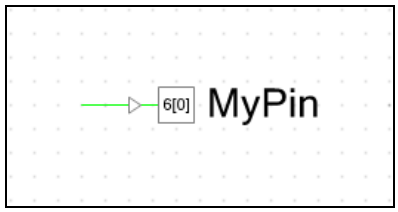
デバッグ処理が求められる場合、デバッグ ポートを手動で再び有効にする必要があることに注意してください。デバッグ インターフェースの無効化は、デバイスのプログラム能力に影響しません。

4.7 データ レジスタで GPIO をより速くトグル

この例はポート データ レジスタおよびマスクを使ってピンを速くトグルする方法を示します。コンポーネント API は GPIO ピンを制御する最も簡単な方法である一方、ピンを更新するのに必要なプロセッサ サイクル数はどれだけ速くトグルが生じるかに影響します。それぞれのコンポーネントのために作成された、*<pin_name>.h* ファイルでのレジスタ 定義及びマスクは、より速くピンを更新するのに使用されます。

1. デジタル出力ピンを回路図に配置して、便宜のために「MyPin」と名前を付けます。
2. ハードウェア接続なしにコンポーネントを設定し、このコンポーネントを物理ピン(この例では P6[0]) に割り当てます。図 27 にを参照ください。

図 27. 回路図に配置されたピン

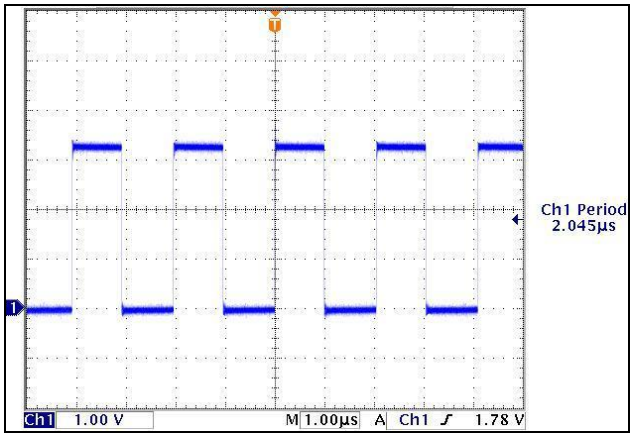


3. *main.c* ファイルに次のコードを追記します:

```
for(;;)
{
   // These are API functions
   MyPin_Write(1); //set MyPin output
   MyPin_Write(0); //clear MyPin output
}
```

4. 図 28 に示すように、API を使用して P6[0]の出力を観察します。

図 28. API 切り替え方法を用いたピン トグル

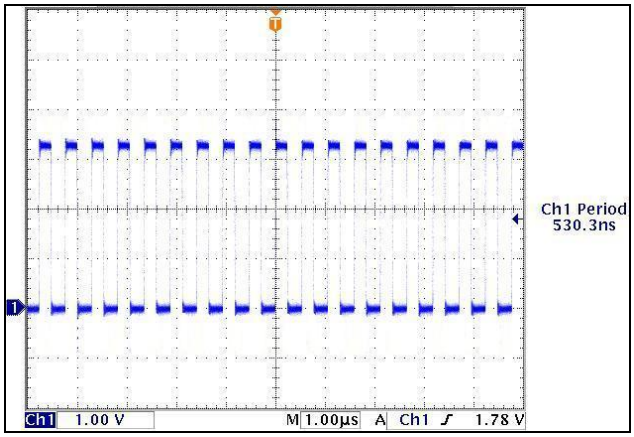


5. *main.c* の中の前のコードをこのコードで置き換えます:

```
for(;;)
{
    MyPin_DR |= MyPin_MASK; //Set MyPin
    MyPin_DR &= ~MyPin_MASK; //Clear
}
```

6. 図 29 に示すように、高速のスイッチング方法を使用して P6[0]の出力を観察します。

図 29. 高速スイッチング方法を用いたピン トグル



ピンは、高速スイッチング方式を使用して、API 機能よりもほぼ 4 倍速くトグルできます。また、このコードはポータブルというメリットもあります。ピン割り当てが開発中に変更される場合は、特定の物理ピンのレジスタへ書き込む必要がありません。

4.8 8051 特殊機能レジスタを使用

PSoC 3 の 8051 は一式の特殊機能レジスタを有し、限定される PSoC レジスタへの高速アクセスを可能にします。これらのレジスタのうちの 2 つを使用して迅速に GPIO ピンをトグルすることができます。

1. 前の例で行ったように、デジタル出力ピン コンポーネントをプロジェクトの回路図に配置し、物理ピンに割り当てます。また、この例では、P6[0]を使用しています。
2. *main.c* ファイルに次のコードを加えます:

```
/* Enable SFR access for P6[0]. */
/* Only done once in the beginning. */
SFRPRT6SEL |= 0x01;
/* Toggle GPIO pin. */
for(;;)
{
     /* Switch on P6[0] */
     SFRPRT6DR |= 0x01;
     /* Switch off P6[0] */
     SFRPRT6DR &= ~0x01;
}
```

3. または次の方法を使用します:

```
for(;;)
{
     /* Toggle P6[0] */
     SFRPRT6DR ^= 0x01;
}
```

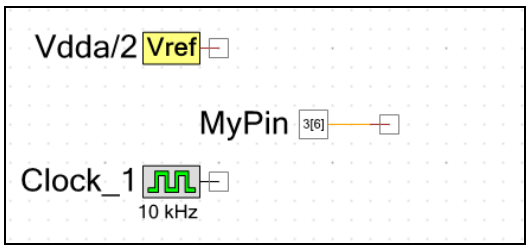
どちらの方法も、非常に高速にピンをトグルできます。SFR の詳細については、PSoC 3 のアーキテクチャ TRM を参照してください。

4.9 GPIO でアナログとデジタル両方を使用

この例では、アナログとデジタルの両方の機能に対してピンを設定し使用する方法を示します。GPIO ピンは短時間に 10kHz のクロック信号を出力し、短時間に基準電圧に切り替えて、それからまた 10kHz の信号に切り替える必要があることを想定します。

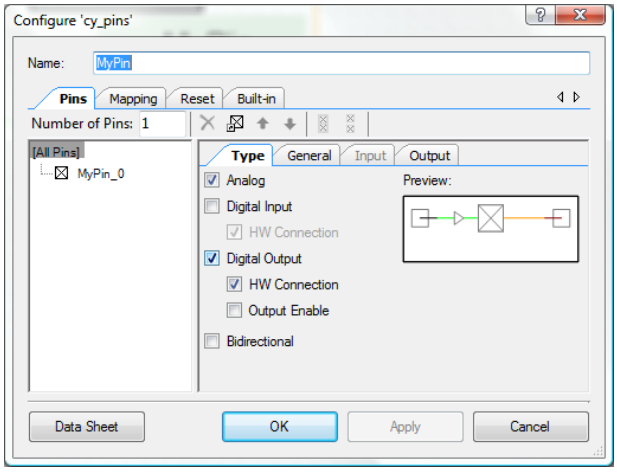
1. 回路図にアナログ ピン、V_{REF}、およびクロックを配置します。
2. 図 30 に示すように、ピン コンポーネントを物理ピンに割り当てます (この例では P3[6]を使用)。

図 30. 回路図に配置される基本的なコンポーネント



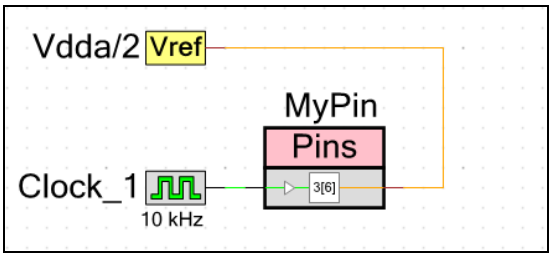
3. 図 31 に示すように、アナログとデジタル出力セッティングの両方を用いて、ピンをコンフィギュレーションします。

図 31. アナログとデジタルの両方としてコンフィギュレーションされた MyPin



4. 図 32 に示すように、クロックをデジタル端末に、V_{REF} をアナログ端末に接続します。

図 32. PSoC Creator のアナログおよびデジタルスイッチングスキームの回路図



5. プロジェクトをコンパイルし、PSoC Creator が使用するアナログ ルーティングを決定するために必要な API を作成します。
6. *cyfitter_cfg.c* ファイルを開き、`CYREG_PRT3_AG` (アナログ グローバル イネーブル) と `CYREG_PRT3_AMUX` (アナログ マルチプレクサ バス イネーブル) のどちらかを探します。この場合、ルーティング ツールは、以下のように、ポート 0 に AG バスを使用することを選択しました。

   ```
   CY_SET_REG8(CYREG_PRT3_AG, 0x40);
   ```

   プロジェクトが再びビルドされると、アナログ ルーティングが変化し得ることに注意してください。プロジェクトにどのような変更が行われた場合にも、ルーティングをチェックしなければなりません。
7. *main.c* ファイルに次のコードを加えます:

```
for(;;)
{
    /* Set pin to Analog */
    // Set P3[6] to Analog HI-Z
    CyPins_SetPinDriveMode(CYREG_PRT3_PC6, PIN_DM_ALG_HIZ);

    // Make AG connection for P3[6]
    CY_SET_REG8(CYREG_PRT3_AG, CY_GET_REG8(CYREG_PRT3_AG) | 0x40);

    // Wait for 100 ms while driving signal
    CyDelay(100);

    /* Set pin to digital */
    // Break AG connection for P3[6]
    CY_SET_REG8(CYREG_PRT3_AG, CY_GET_REG8(CYREG_PRT3_AG) & 0xBF);

    //Set P3[6] to Strong Drive mode
```

```
        CyPins_SetPinDriveMode(CYREG_PRT3_PC6, PIN_DM_STRONG);

        // Wait for 100 ms while driving signal
        CyDelay(100);
    }
```

8. PSoC 3 または PSoC 5LP デバイスをコンパイルおよびプログラムします。

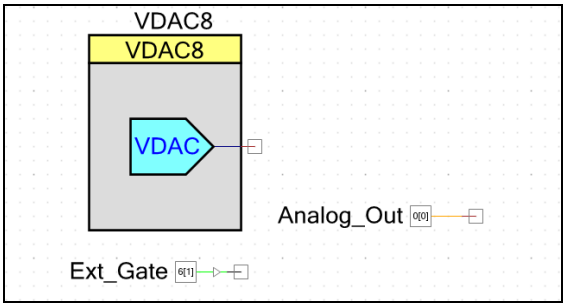
その結果は、クロック信号とリファレンス電圧間を毎 100 ms で切り替わる出力です。

4.10 ハードウェアによるアナログスイッチングの制御

この例では、外部信号が、CPU の介入なしにアナログ ピンの出力をゲート制御するために、使用される方法を示します。

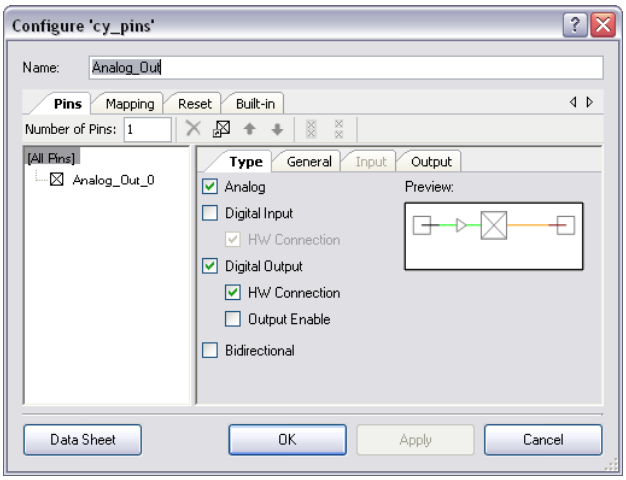
1. 図 33 に示すように、デジタル入力ピン (この例では Ext_Gate)、アナログ ピン (Analog_Out) とアナログ ソース (VDAC8) を、プロジェクトの回路図に配置します。

図 33. ハードウェア制御のゲート用コンポーネント



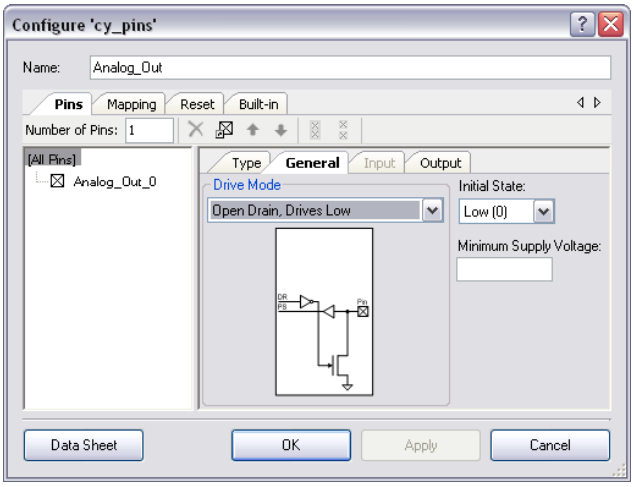
2. 図 34 に示すように、Analog_Out ピンをアナログとデジタルの両方の特性を持つようにコンフィギュレーションします。

図 34. Analog_Out ピン コンフィギュレーション



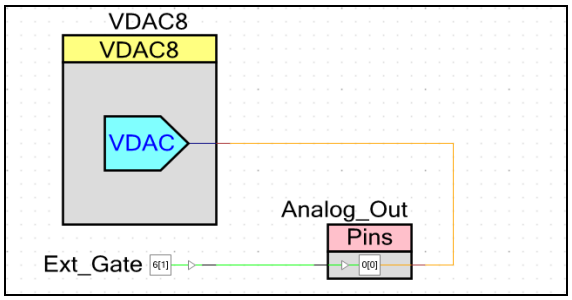
3. 図 35 に示すように、Analog_Out ピンの駆動モードを「Open Drain Drives Low」として設定します。

図 35. Analog_Out ピンの駆動モード



4. 図 36 に示すように、コンポーネントを接続します。

図 36. ハードウェア ゲートのあるアナログ ピン



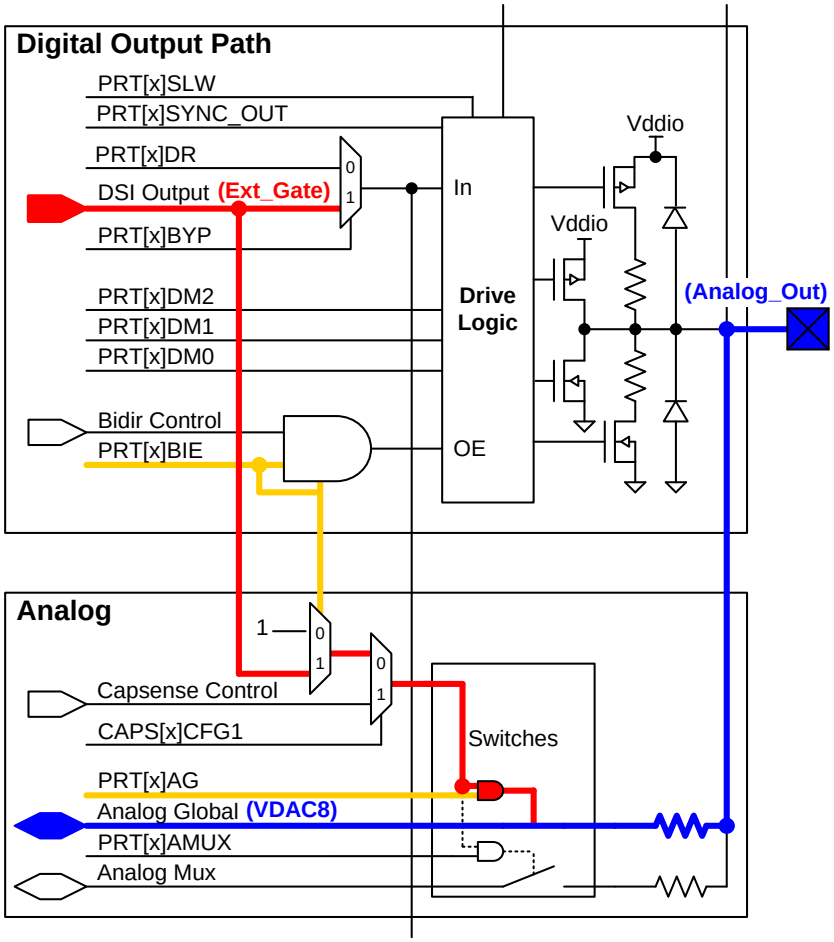
5. *main.c* ファイルに次の一連のコードを加えます。この例では、ポート 0 コンフィギュレーション レジスタのピン 0 を双方向ビットに設定します。

```
// Set P0[0] to bidirectional mode
CY_SET_REG8(CYDEV_IO_PRT_PRT0_BIE, 0x01);
```

6. Analog_Out ピンを P0[0]に割り当て、前のコードに一致させます。
7. PSoC 3 または PSoC 5LP デバイスをコンパイルおよびプログラムします。

PSoC 3 と PSoC 5LP データシートの詳細 GPIO ブロック図から取られた図 37 に、GPIO 制御論理を使用してこの技術を実行する方法を示します。

図 37. PSoC 3 と PSoC 5LP データシートから取られたハイライトされた GPIO ブロック図



Ext_Gate 信号は、DSI を通じて、Analog_Out ピンのデジタル部分まで、ルーティングされます。DSI (赤) からの信号はアナログ スイッチにルーティングされます。理由はポートの双方向ビットとアナログ グローバル選択ビットがセットされる (黄) からです。VDAC 出力 (青) は、Ext_Gate 信号の論理状態に応じてオン／オフにされます。

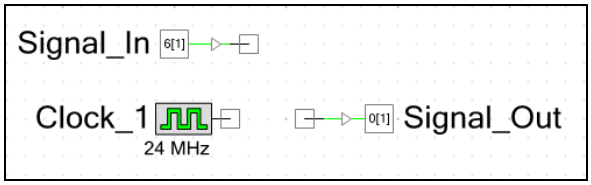
PSoC 3 と PSoC 5LP デバイスで可能なアナログ切り替えの詳細については、TRM の「アナログ ルーティング」節を参照してください。他にはアプリケーション ノート N58827 – PSoC 3 and PSoC 5LP Internal Analog Routing Considerations を参照することもできます。

4.11 DSI をクロック ソースとして使用

この例では、DSI を通じてルーティングされたデジタル信号をクロック ソースとして使用する方法を示します。最大 8 つのデジタルと 4 つのアナログ クロックを、任意の DSI 信号から作ることができます。また、PSoC デバイスは任意のデジタル信号を PLL の入力ソースとして使用することができます。

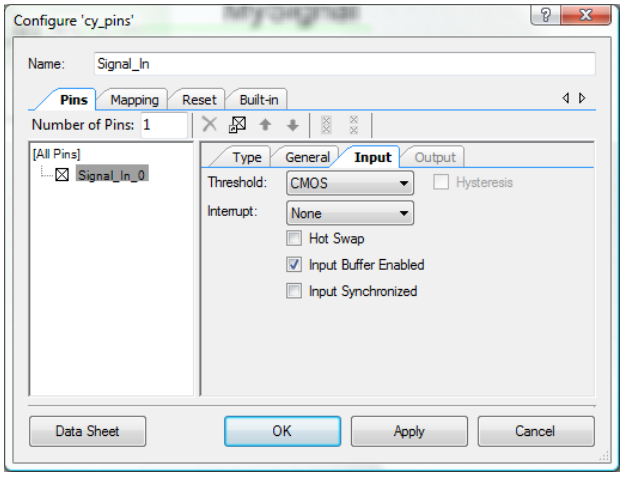
1. 図 38 に示すように、デジタル入力ピン、デジタル出力ピンとクロックを、回路図に配置します。

図 38. DSI クロック例の基本的なコンポーネント



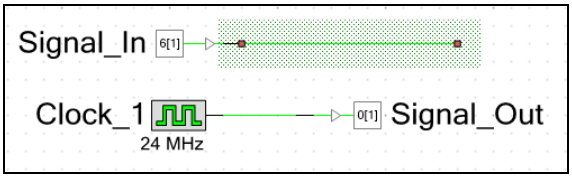
2. 図 39 に示すように、Signal_In 用のコンフィギュレーション ダイアログを開き、ピン コンフィギュレーション ウィンドウで **Input Synchronized** オプションの選択を解除します。これは、信号がそれ自体に同期化しようとすることを防止するために必要です。

図 39. 「Input Synchronized」が無効化された



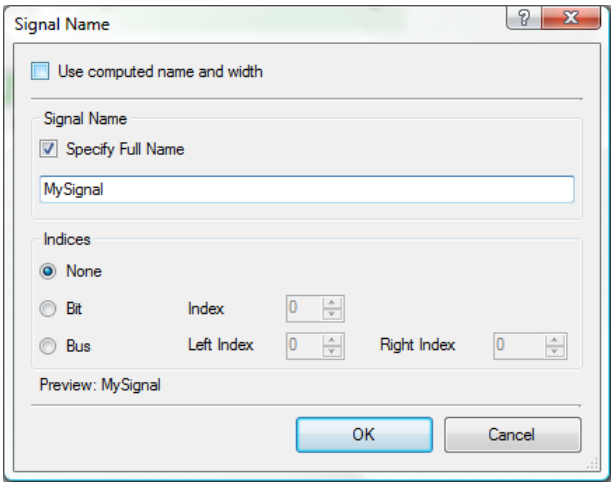
3. ワイヤ ツールでクロックを Signal_Out に接続します。
4. ワイヤ ツールを使用して、Signal_In のみに接続する DSI ソースの信号を生成します。図 40 に示すように、Signal_In の端子から離れて始め、ワイヤを生成します。

図 40. DSI クロック ソース信号



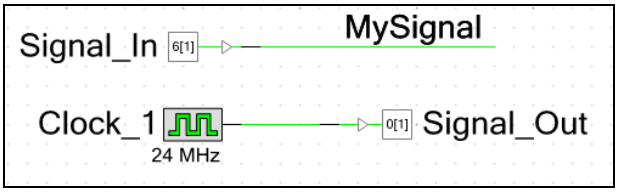
5. ワイヤを右クリックし、表示するポップアップ メニューから **Edit Name And Width** を選択します。
6. 図 41 のように、**Use computed name and width** オプションの選択を解除して、**Signal Name** フィールドに固有の名前 (この例では「MySignal」) を入力します。

図 41. 信号名コンフィギュレーション ウィンドウ



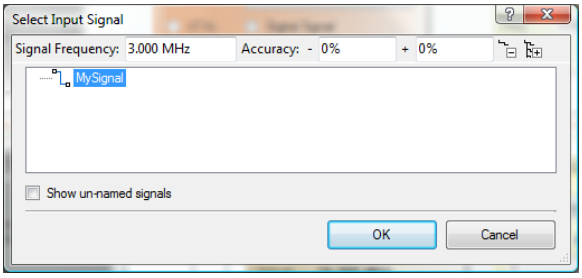
これらのコンフィギュレーションを設定した後では、回路図は図 42 のようになります。

図 42. DSI クロック例のために変更されたコンポーネント



7. プロジェクトの Design-Wide Resources ファイル (*<Project Name>.cydwr*) を開き、**Clocks** タブを選択します。
8. システム クロックのいずれかをダブルクリックして **Configure Systems Clocks** ウィンドウを開きます。
9. **Digital Signal** オプションを選択し、「**…**」ボタンをクリックして、「**Select Input Signal**」ウィンドウを開きます。
10. 図 43 に示すように、「MySignal」を選択し、信号周波数 (この例では 3MHz) と精度を入力します。

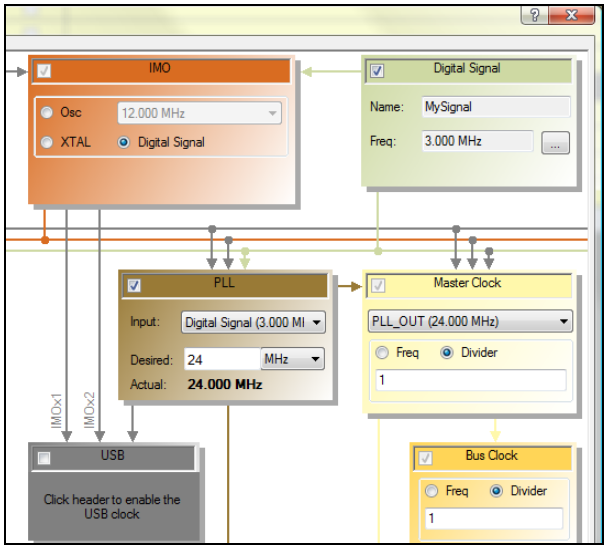
図 43. デジタル信号コンフィギュレーション ウィンドウ



PSoC Creator が、システム クロックをコンフィギュレーションするために必要な計算を行うために、この情報を使用することに注意してください。信号情報が正確であることを確認します。

11. 図 44 に示すように、内部メイン発振器 (**IMO**) と **PLL** への入力ソースを「Degital Signal」に設定します。

図 44. IMO と PLL へのソースとしての DSI 信号



PLL は 3MHz の入力を使用して 24MHz の出力を生成し、Signal_Out クロックを生成するためにマスターとバス クロックにルーティングされます。

DSI システム クロッキングの詳細については、データシート、TRM、および「AN60631 – PSoC 3 and PSoC 5LP Clocking Resources」アプリケーション ノートを参照してください。

4.12 ファームウェアを使用して PICU 設定を変更

PICU の動的コンフィギュレーションは、PICUx_INTTYPEy レジスタの[1:0]ビットへの書き込みにより行われます: 「x」がポート番号で、「y」がピン番号です (表 2 を参照)。ユーザーはいつでもコンフィギュレーションを変更して、ピン割り込みを有効または無効にすることができます。

表 2. PICU 割り込みのタイプおよびビット設定

1:0 ビット	文書名	説明
00	無効	割り込みが無効
01	立ち上がりエッジ	立ち上がりエッジでトリガー
10	立ち下がりエッジ	立ち下がりエッジでトリガー
11	モード変更	任意のエッジでトリガー

この例では、P6[0]ピンが立ち上がりエッジ割込みとしてコンフィギュレーションされ、P6[1]ピンが立ち下がりエッジ割込みとしてコンフィギュレーションされています。

1. 2 本のデジタル入力ピンをプロジェクト回路図に配置します。
2. それらのピンを P6[0]と P6[1]に割り当てます。
3. P6[0]を抵抗プルダウン ピンとしてコンフィギュレーションする、あるいは外部プルダウンを加えます。
4. P6[1]を抵抗プルアップ ピンとしてコンフィギュレーションする、あるいは外部プルアップを加えます。
5. *main.c* ファイルに次のコードを追記します:

```
//Set P6[0] to PICU rising-edge trigger
CY_SET_REG8(CYREG_PICU6_INTTYPE0, 0x01);

//Set P6[1] to PICU falling-edge trigger
CY_SET_REG8(CYREG_PICU6_INTTYPE1, 0x02);

//Sleep and wait for PICU interrupt
//Sleep again if not P6[1] PICU wakeup
do
{
    //Save clocks and enter sleep
    CyPmSaveClocks();
    CyPmSleep(PM_SLEEP_TIME_NONE, PM_SLEEP_SRC_PICU);
    CyPmRestoreClocks();

    //Stay awake for two seconds
    CyDelay(2000);
}
while (!(CY_GET_REG8(CYREG_PICU6_INTSTAT) & 0x02));

//Disable P6[1] PICU trigger
CY_SET_REG8(CYREG_PICU6_INTTYPE1, 0x00);
```

6. PSoC 3 または PSoC 5LP デバイスをコンパイルおよびプログラムします。

PSoC デバイスは、任意の PICU 割り込み時にスリープからウェイクしますが、P6[1]がトリガーの場合を除き、スリープ状態に戻ります。ウェイクアップ後に割り込みを無効化する必要はありません。他のいずれの割り込みソースのように、通常の動作で使用できます。

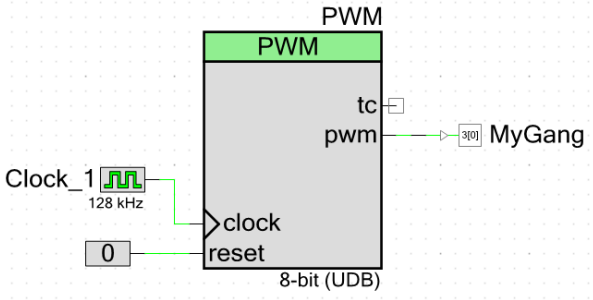
PSoC 3 または PSoC 5LP TRM は、ブロック図と機能説明を含む、PICU の詳細の情報を含みます。他に「AN54460 – PSoC 3 and PSoC 5LP Interrupts」アプリケーション ノートを参照することもできます。

4.13 より大きな駆動／シンク電流のためにピンを連動

回路の全ソース／シンク能力を向上するために、GPIO ピンを連動させます (互いに短絡させます）。V_{DDIO} 象限の制限がこの場合でも適用されます。この例は、4 本の GPIO ピンを使用した PWM 信号の駆動を示します。

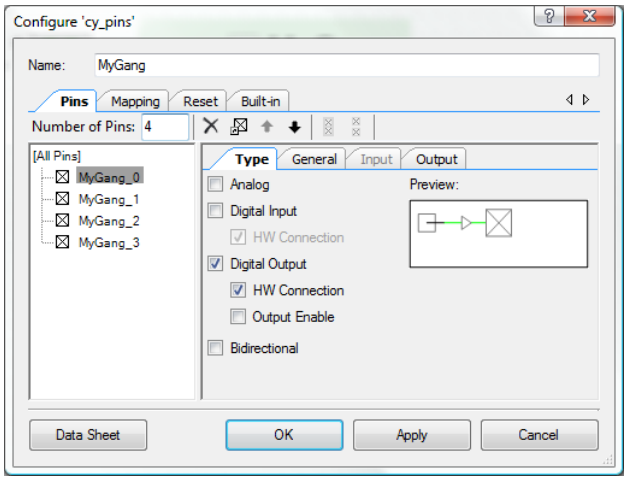
1. PWM とクロック コンポーネントを配置しコンフィギュレーションします。
2. 図 45 のように、単一のデジタル出力ピン コンポーネントを配置し、それを PWM 出力端子に接続します。

図 45. 回路図に配置された単一ピン



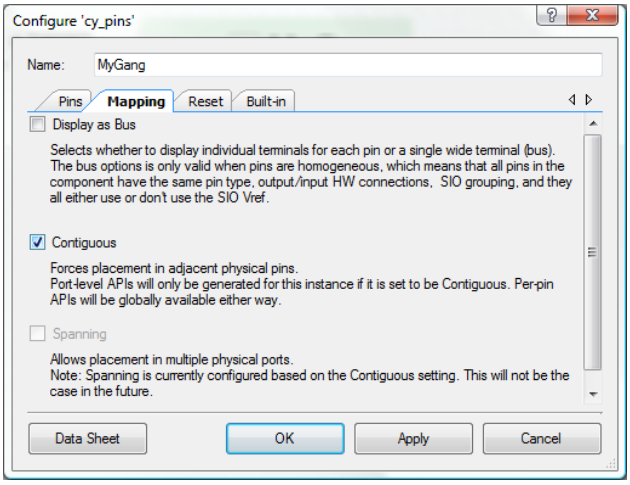
3. 図 46 のように、ピン コンフィギュレーション ダイアログを開き、 ピンの数を適切に設定します。この例では、4 つの GPIO ピンを使います。

図 46. コンポーネントで複数のピンをコンフィギュレーション



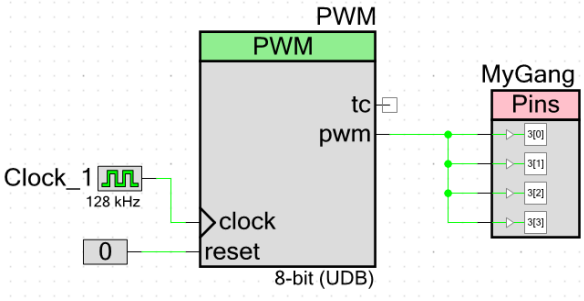
4. (オプション) 図 47 のように、より簡単なプリント基板ルーティングのために、ピン マッピングを **Contiguous** に設定します。

図 47. 隣接マッピングを有効化



5. ピン コンポーネントを物理ピンに割り当てます。
6. 図 48 のように、信号ソース (この例では PWM) をコン|ポーネントの各ピン ターミナルに接続します。

図 48. 連動されたピン シンボル



7. PSoC 3 または PSoC 5LP デバイスをコンパイルおよびプログラムします。

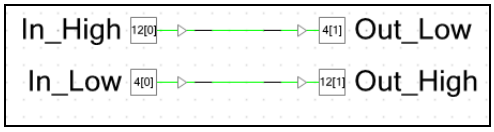
PWM の出力はの 4 本の GPIO 全ての上で駆動されます。これらのピンは、プリント基板で外部に短絡し、また必要な場合は外部回路に接続することができます。

4.14 レベルシフト信号

GPIO ピンは、異なる電圧で V_{DDIO} ピンに電源を供給することにより、信号をレベルシフトするために使用することができます。唯一の制限は、V_{DDA} よりも高い電圧にあるような V_{DDIO} 象限がないことです。この例では、PSOC 3 または PSoC 5LP デバイスで簡単な 5V／1.8V のレベルシフト コンフィギュレーションを作成する方法を示します。

1. プロジェクト回路図に、2 本の High-Z デジタル入力ピンと 2 本のストロング駆動デジタル出力ピンを配置します。
2. 入力のいずれかを出力のいずれかに接続します。残りの入力を残りの出力に接続します。
3. 便宜上、ピン シンボルを図 49 に示しのと同様の意味のある名前を付けます。

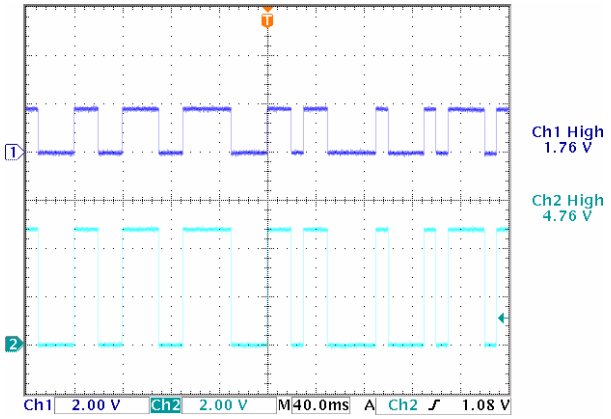
図 49. GPIO ピンを使用したレベルシフト



4. 1 つの V_{DDIO} 象限にあるピンに 5V 信号を、もう一つの象限のピンに 1.8V 信号を割り当てます。この例では、$V_{DDIO}3$ (5.0 V ～ P12[1:0]) と $V_{DDIO}0$ (1.8 V ～ P4[1:0]) が選択されました。V_{DDIO} の分配については、デバイスのデータシートを参照してください。
5. 開発基板において、V_{DDD}、V_{DDA}、$V_{DDIO}1$、$V_{DDIO}2$ と $V_{DDIO}3$ を 5V 電源に接続します。
6. $V_{DDIO}0$ を 1.8V 電源に接続します。
7. PSoC 3 または PSoC 5LP デバイスをコンパイルおよびプログラムします。

high-side 入力に加えられたすべての 5V 信号は、low-side 出力では 1.8V で現れます。同様に、図 50 のように、low-side 入力に加えられたすべての 1.8V 信号は、high-side 出力では 5V で現れます。

図 50. レベルシフトされた信号



信号の電圧をシフトすることに加えて、PSoC デバイスは、データが通過する際に、データを読み取り操作することができます。1 つの V_{DDIO} 象限内のすべての GPIO ピンが同じ電圧になることに注意してください。

5 関連アプリケーション ノート

- AN54181 – Getting Started with PSoC 3
- AN77759 – Getting Started with PSoC 5LP
- AN60631 – PSoC 3 and PSoC 5LP Clocking Resources
- AN58304 – PSoC 3 and PSoC 5LP – Pin Selection for Analog Designs
- AN58827 – PSoC 3 and PSoC 5LP Internal Analog Routing Considerations
- AN54460 – PSoC 3 and PSoC 5LP Interrupts
- AN60580 – SIO Tips and Tricks in PSoC 3/PSoC 5LP
- AN77900 – PSoC 3 and PSoC 5LP Low-power Modes and Power Reduction Techniques
- AN52705 – PSoC 3 and PSoC 5LP – Getting Started with DMA

著者について

氏名: Greg Reynolds

経歴: Greg Reynolds は、10 年以上にわたってサイプレスでいろいろな役割を果たしてきました。

付録A. GPIO の API およびレジスタの参照情報

A.1 コンポーネント API

ピン コンポーネントの API は、コンポーネントに関連したすべての物理ピンにアクセスするために使用されます。コンポーネントのインスタンス名は、ユーザーにより割り当てられるかあるいは PSoC Creator により生成され、関数名の接頭辞として使用されます。コンポーネント API は、ピン コンポーネントのデータシートに詳細に記載されています。表 3 にコンポーネント関数の一覧を示します。

表 3. コンポーネント API の参照情報

API	説明	用例 (MyPin のコンポーネント名を使用)
<Pin_Name>_Read	コンポーネントのすべてのピンの現在の値を返す	`myVar = MyPin_Read();`
<Pin_Name>_Write	コンポーネント ピンに値を書き込む	`MyPin_Write(1);`
<Pin_Name>_ReadDataReg	コンポーネントのすべてのピンの現在の値を返す	`myVar = MyPin_ReadDataReg();`
<Pin_Name>_SetDriveMode	コンポーネントの各ピンの駆動モードを設定する	`MyPin_SetDriveMode(MyPin_DM_ALG_HIZ);`
<Pin_Name>_ClearInterrupt	コンポーネントがマップされているポート上のアクティブな割り込みをクリアする。割り込みステータス レジスタの値を返す	`myVar = MyPin_ClearInterrupt();`

A.2 ピン単位 API

グローバルのピン単位 API マクロを使用して、個々の物理ピンにアクセスすることができます。マクロがピン コンフィギュレーション レジスタに直接アクセスするため、物理ピンはピン コンポーネントに関連付けられる必要はありません。パーピンの API を使用すると、コンポーネント コンフィギュレーションとの矛盾によりピンコンポーネントに関連した物理ピンに未定義の動作が発生する可能性があります。ピン単位 API は、PSoC Creator System Reference Guide に詳細に記載されています。表 4 は、ピン単位関数の一覧を示します。

表 4. パーピン API の参照情報

API	説明	用例 (P1[2] を使用)
CyPins_ReadPin	ピンの現在の値を読み出す	`myVar = CyPins_ReadPin(CYREG_PRT1_PC2);`
CyPins_SetPin	ピンの出力値を論理 HIGH に設定する	`CyPins_SetPin(CYREG_PRT1_PC2);`
CyPins_ClearPin	ピンの出力値を論理 LOW に設定する	`CyPins_ClearPin(CYREG_PRT1_PC2);`
CyPins_SetPinDriveMode	ピンの駆動モードを設定する	`CyPins_SetPinDriveMode(CYREG_PRT1_PC2, PIN_DM_ALG_HIZ);`
CyPins_ReadPinDriveMode	ピンの駆動モードを読み出す	`myVar = CyPins_ReadPinDriveMode(CYREG_PRT1_PC2);`
CyPins_FastSlew()	ピンのスルー レートを高速エッジ レートに設定する	`CyPins_FastSlew(CYREG_PRT1_PC2);`

API	説明	用例 (P1[2] を使用)
CyPins_SlowSlew()	ピンのスルー レートを低速エッジ レートに設定する	CyPins_SlowSlew(CYREG_PRT1_PC2);

A.3 GPIO レジスタ

以下のレジスタは、GPIO のコンフィギュレーションに使用されます。通常の動作中にファームウェアでアクセスできます。詳細説明とレジスタ マップはサイプレスのウェブサイトから無料で入手できる PSoC 3 と PSoC 5LP レジスタの TRM に記載されます。表 5 は GPIO レジスタの一覧を示します。

表 5. GPIO レジスタ

レジスタ	名称	説明
PRTx_PCy[7:0]	ポート ピン コンフィギュレーション	このレジスタは、いくつかのコンフィギュレーションやシングル I/O ポート ピンのステータス ビットに、一度にアクセスできる
PRTx_DR[7:0]	ポート データ出力	このレジスタは、対応する GPIO ポートの出力データの状態を設定するために使用される
PRTx_PS[7:0]	ポート ピン状態	このレジスタは、対応する GPIO ポートの論理ピンの状態を保持するピンの駆動モードが High-Z アナログに設定された場合、状態は常に 0 を読み出すことになる
PRTx_DM2[7:0] PRTx_DM1[7:0] PRTx_DM0[7:0]	ポート駆動モード	これらのレジスタの合成値は、GPIO ポートの各ビットの固有の駆動モードを決定する
PRTx_SLW[7:0]	ポート スルー レート制御	このレジスタは、任意のストロング駆動モードの GPIO ピンに対して、高速または低速のエッジ レートを設定するために使用する
PRTx_BYP[7:0]	ポート バイパス イネーブル	このレジスタは、対応する GPIO の出力データが、DSI とポート論理データ レジスタのどちらから供給されるかを選択する
PRTx_BIE[7:0]	ポート双方向イネーブル	このレジスタは、DSI を通じて動的な双方向制御を有効化するために使用される
PRTx_INP_DIS[7:0]	入力バッファ ディスエーブル オーバーライド	このレジスタは、入力バッファをオフに強制するために使用される
PRTx_CTL[0]	ポート ワイド制御信号	このレジスタは、内部バッファのトリップ ポイントを選択するために使用される
PRTx_PRT[7:5,3:1]	ポート ワイド コンフィギュレーション	このレジスタは、シングル ビットの書き込みを用いたポート ワイドベースにおいて、いくつかの可能なコンフィギュレーション レジスタにアクセスする
PRTx_BIT_MASK[7:0]	エイリアス レジスタアクセス用のビットマスク	このレジスタは、エイリアス レジスタ アドレス空間からデータ レジスタへのアクセスを許可またはブロックする
PRTx_AMUX[7:0]	ポート アナログ グローバル マルチプレクサ バス イネーブル	このレジスタは、対応する GPIO ポートに対するアナログ グローバル マルチプレクサ スイッチを制御する

レジスタ	名称	説明
PRTx_AG[7:0]	ポート アナログ グローバル イネーブル	このレジスタは、対応する GPIO ポートに対してアナログ グローバルスイッチを制御する
PICUx_INTTYPE[1:0]	ポート割り込み制御タイプ	このレジスタは、対応する GPIO ピンの割り込みタイプを設定する
PICUx_INTSTAT[7:0]	ポート割り込み制御ステータス	このレジスタは、対応する GPIO ポートの掲示された割り込みを表示する
PICUx_SNAP[7:0]	ポート割り込み制御スナップショット	このレジスタは、PICUx_INTSTAT レジスタの最後の読み出しでの入力ピンの状態を示す
PICUx_DISABLE_COR[0]	ステータス レジスタのクリア オン リード機能のディスエーブル	このレジスタは、PICUx_INTSTAT レジスタの「クリア オン リード」機能を無効化する

A.4 不揮発性ラッチ

PSoC 3 と PSoC 5LP は、不揮発性ラッチ (NVL) のアレイを備えていて、リセット時のデバイスの動作をコンフィギュレーションするために使用します。以下のラッチは、GPIO のコンフィギュレーションに使用されます。通常の動作中はファームウェアからアクセスできません。NVL アレイのさらなる情報とレジスタ マップは、PSOC3 と PSoC5LP のデータシートと TRM の「不揮発性ラッチ」節に記載されています。表 6 は、不揮発性ラッチの一覧を示します。

表 6. GPIO に関連した不揮発性ラッチ

不揮発性ラッチ	名称	説明
PRTxRDM[1:0]	ポート リセット駆動モード	対応する I/O ポートのリセット駆動モードを制御する
XRESMEN[0]	オプションの XRES イネーブル	ピン P1[2]が GPIO として使用されるか、外部リセットとして使用されるかを制御する
DPS[1:0]	デバッグ ポート選択	複数の Port1 ピンを、デバッグ ポートとして使用することを制御する

付録B. PSoC Creator の設定とレジスタ

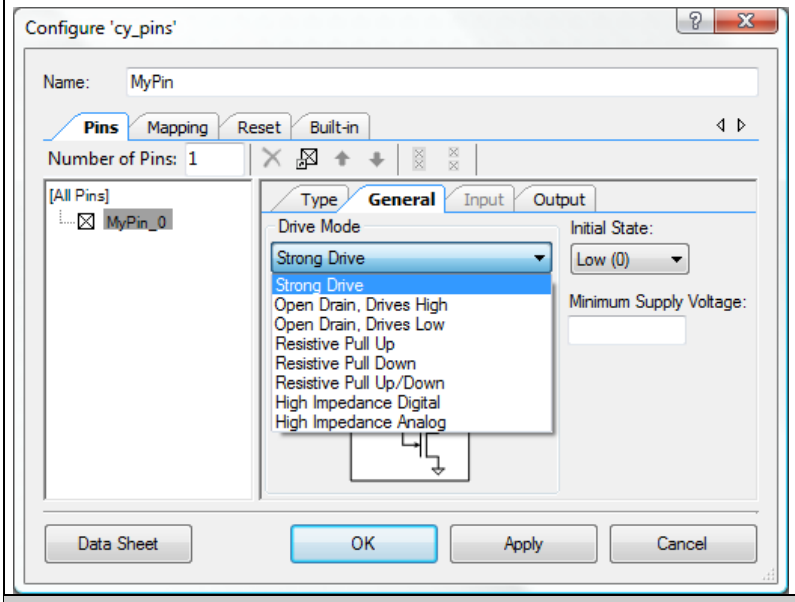
PSoC Creator で確立した GPIO 設定は、cy_boot スタートアップ コードの一部で、デバイスの初期コンフィグレーション時に有効です。本付録の表は、ピン コンポーネントのコンフィギュレーション ウィンドウ内の設定と GPIO レジスタとの関係を示します。また、簡単なコード用例により、 (該当する場合) 通常動作時にファームウェアで同じ機能を実行する方法を示します。コンフィギュレーション ウィンドウは、使用している PSoC Creator のバージョンによって、やや異なって見えることがあります。

表 7. 駆動モード パラメーター

駆動モード

このパラメーターは、8 つの可能なピン駆動モードのいずれか一つを供給して、ピンをコンフィグレーションします。ピン タイプはデフォルト設定を決定します。

PSoC Creator コンフィギュレーション ウィンドウ



コンポーネント API

<PinName>_SetDriveMode()

ピン単位 API

CyPins_SetPinDriveMode()
CyPins_ReadPinDriveMode()

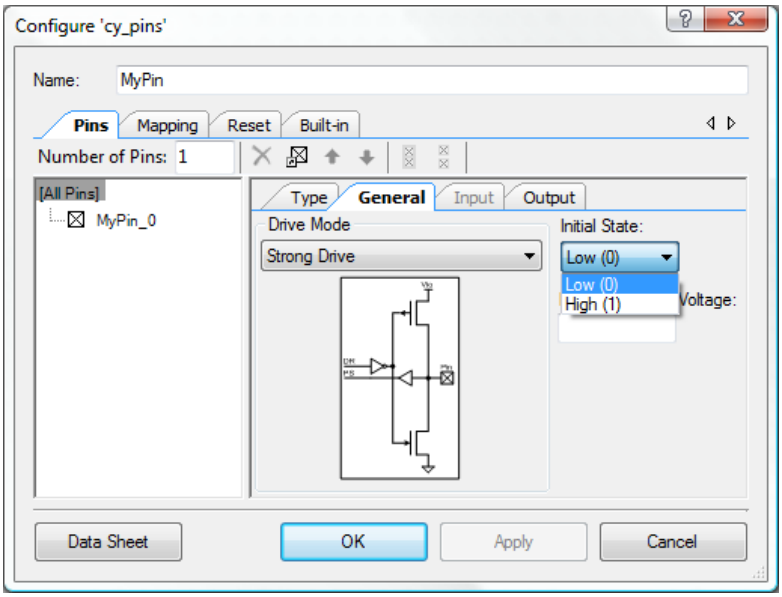
関連レジスタ

PRTx_PCy[3:1]
PRTx_DMy[7:0]
PRTx_BIE[7:0]

コード用例

```
/* Set pin to Resistive Pull-up Using Component API */
MyPin_SetDriveMode(MyPin_DM_RES_UP);

/* Set pin to Resistive Pull-up Using Per-Pin API */
CyPins_SetPinDriveMode(CYREG_PRT1_PC2,PIN_DM_RES_UP);

/* Read Drive Mode Using Per-Pin API */
myVar = CyPins_ReadPinDriveMode(CYREG_PRT1_PC2);

/* Set pin to Resistive Up/Down Using Register Write */
CY_SET_REG8(CYREG_PRT1_PC2, CY_GET_REG8(CYREG_PRT1_PC2) | 0x07);
```

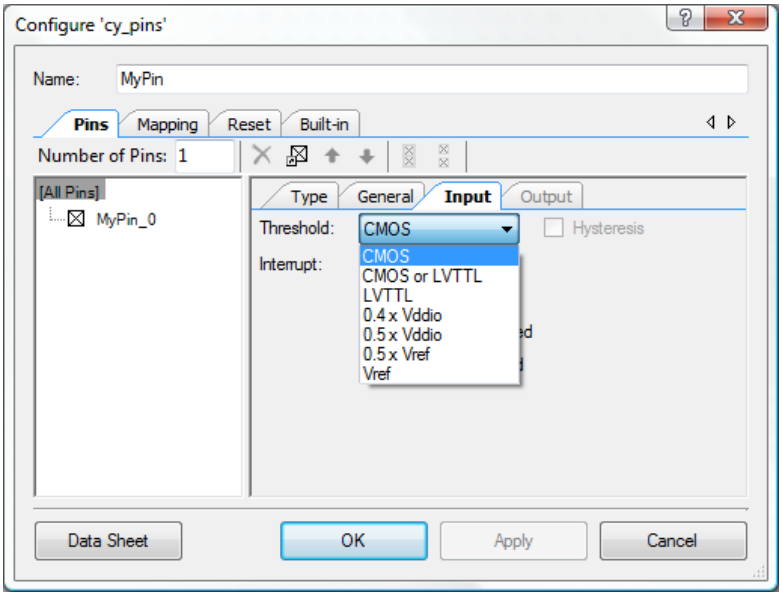
表 8. 初期状態パラメーター

初期状態	
このパラメーターは、パワーオン リセット (POR) 後にピンのデータ レジスタに書き込まれる値を指定します。	
PSoC Creator コンフィギュレーション ウィンドウ	**コンポーネント API**
	<PinName>_Write() **ピン単位 API** CyPins_SetPin() CyPins_ClearPin() **関連レジスタ** PRTx_DR[7:0]

コード用例

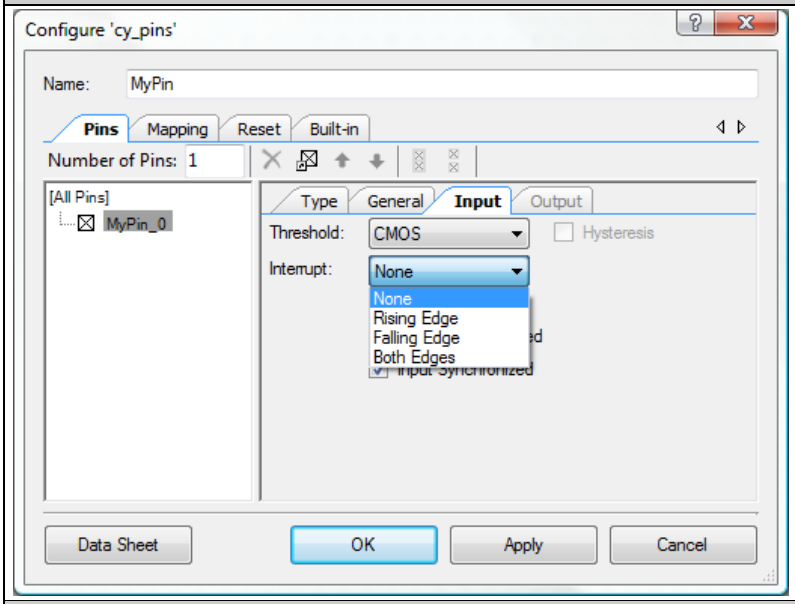
```
/* Set pin to logic state HIGH output */
       MyPin_Write(1);

/* Set pin P1[2] to logic state HIGH output */
       CyPins_SetPin(CYREG_PRT1_PC2);

/* Set pin P1[2] to logic state LOW output */
       CyPins_ClearPin(CYREG_PRT1_PC2);

       /* Set pin P1[2] output to logic state LOW using a register write */
       CY_SET_REG8(CYREG_PRT1_DR, CY_GET_REG8(CYREG_PRT1_DR) | 0xFB);
```

表 9. 閾値パラメーター

<table>
<tr><th colspan="2">閾値</th></tr>
<tr><td colspan="2">このパラメーターは論理 HIGH (1) と論理 LOW (0) を定義するしきい値を選択します。この設定は、ポート内の全ての物理ピンに適用される。CMOS と LVTTL 設定のみが PSoC 4 GPIO ピンに対して有効です。</td></tr>
<tr><th>PSoC Creator コンフィギュレーション ウィンドウ</th><th>コンポーネント API</th></tr>
<tr><td rowspan="4">
</td><td>該当なし</td></tr>
<tr><th>ピン単位 API</th></tr>
<tr><td>該当なし</td></tr>
<tr><th>関連レジスタ</th></tr>
<tr><td>PRTx_CTL[0]</td></tr>
<tr><th colspan="2">コード用例</th></tr>
<tr><td colspan="2"><pre>
/* Set port 1 logic threshold to CMOS using a register write */
      CY_SET_REG8(CYREG_PRT1_CTL, 0);

/* Set port 1 logic threshold to LVTTL using a register write */
      CY_SET_REG8(CYREG_PRT1_CTL, 1);
</pre></td></tr>
</table>

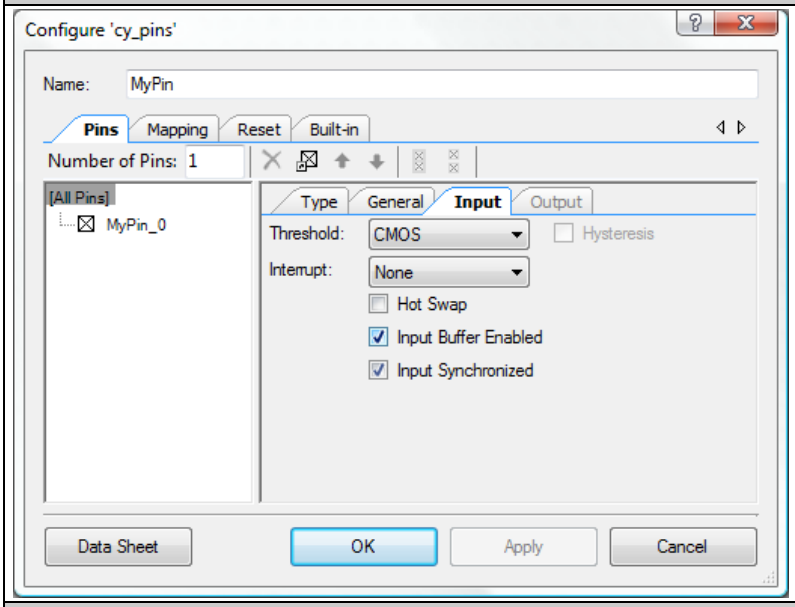
表 10. 割り込みパラメーター

割り込み	
このパラメーターは GPIO の割込みタイプを選択します。	
PSoC Creator コンフィギュレーション ウィンドウ	**コンポーネント API**
	該当なし
	ピン単位 API
	該当なし
	関連レジスタ
	PICUx_INTTYPEy[1:0]

コード例

```
/* Disable interrupt on P1[2] */
CY_SET_REG8(CYREG_PICU1_INTTYPE2, 0x00);

/* Enable rising-edge interrupt on P1[2] */
CY_SET_REG8(CYREG_PICU1_INTTYPE2, 0x01);

/* Enable falling-edge interrupt on P1[2] */
CY_SET_REG8(CYREG_PICU1_INTTYPE2, 0x02);

/* Enable any edge interrupt on P1[2] */
      CY_SET_REG8(CYREG_PICU1_INTTYPE2, 0x03);
```

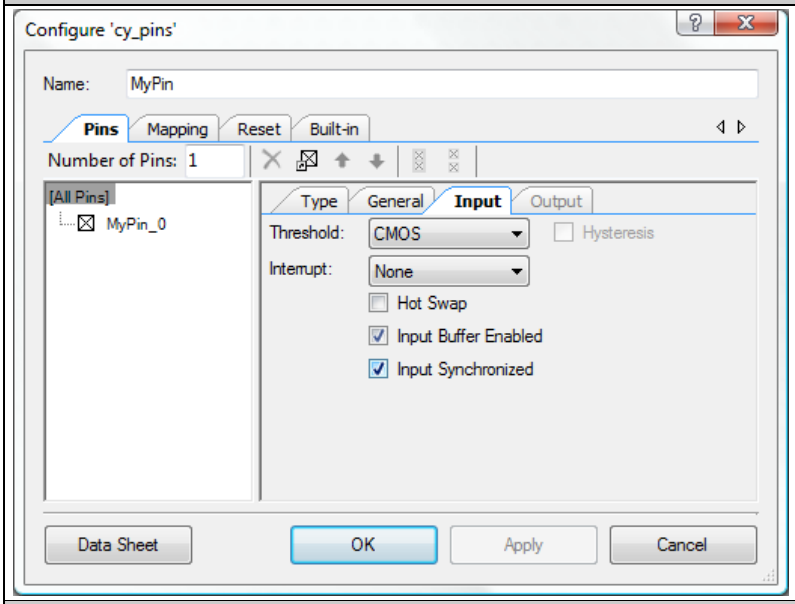
表 11. 入力バッファ イネーブル パラメーター

<table>
<tr><th colspan="2">入力 バッファイネーブル</th></tr>
<tr><td colspan="2">このパラメーターはピンのデジタル入力バッファが有効化されるかを決定します。</td></tr>
<tr><th>PSoC Creator コンフィギュレーション ウィンドウ</th><th>コンポーネント API</th></tr>
<tr><td rowspan="4">
</td><td>該当なし</td></tr>
<tr><th>ピン単位 API</th></tr>
<tr><td>該当なし</td></tr>
<tr><th>関連レジスタ</th></tr>
<tr><td>PRTx_INP_DIS[7:0]</td></tr>
<tr><th colspan="2">コード用例</th></tr>
<tr><td colspan="2">

```
/* Disable Input Buffer on P1[2] using register write. */
CY_SET_REG8(CYREG_PRT1_INP_DIS, CY_GET_REG8(CYREG_PRT1_INP_DIS) | 0x04);
```

</td></tr>
</table>

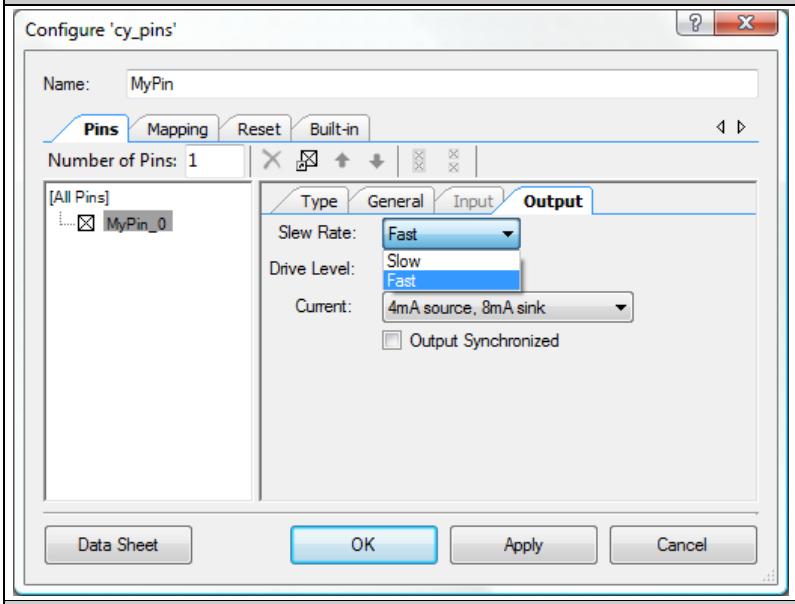
表 12. 入力同期化パラメーター

<table>
<tr><th colspan="2">入力同期化</th></tr>
<tr><td colspan="2">このパラメーターは、ピンの入力をバス クロックと同期化することを可能にします。</td></tr>
<tr><th>PSoC Creator コンフィギュレーション ウィンドウ</th><th>コンポーネント API</th></tr>
<tr><td rowspan="4">
</td><td>該当なし</td></tr>
<tr><th>ピン単位 API</th></tr>
<tr><td>該当なし</td></tr>
<tr><th>関連レジスタ</th></tr>
<tr><td>PRTx_DBL_SYNC_IN[7:0]</td></tr>
<tr><th colspan="2">コード用例</th></tr>
<tr><td colspan="2">

```
/* Sync input of P1[2] to bus_clk using register write. */
      CY_SET_REG8(CYREG_PRT1_DBL_SYNC_IN, CY_GET_REG8(CYREG_PRT1_DBL_SYNC_IN) | 0x04);
```

</td></tr>
</table>

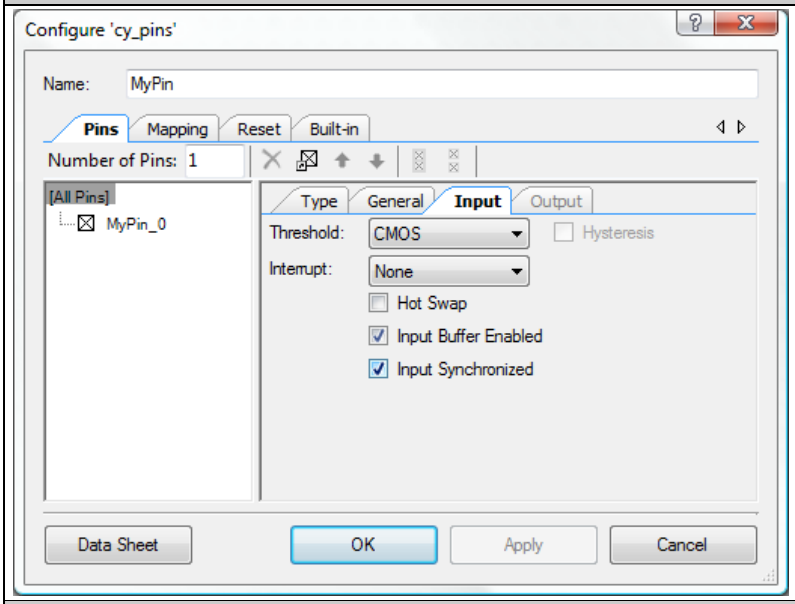
表 13. スルー レート パラメーター

<table>
<tr><th colspan="2">スルー レート</th></tr>
<tr><td colspan="2">このパラメーターは、出力論理レベルが変化する時に、ピンの立ち上がりと立ち下がりのランプ レートを決定します。</td></tr>
<tr><th>PSoC Creator コンフィグレーション ウィンドウ</th><th>コンポーネント API</th></tr>
<tr><td rowspan="4">
</td><td>該当なし</td></tr>
<tr><th>ピン単位 API</th></tr>
<tr><td>CyPins_FastSlew(CYREG_PRTx_PCy)
CyPins_SlowSlew(CYREG_PRTx_PCy)</td></tr>
<tr><th>関連レジスタ</th></tr>
<tr><td>PRTx_SLW[7:0]</td></tr>
<tr><th colspan="2">コード例</th></tr>
<tr><td colspan="2">

```
/* Set fast edge rate for P1[2] */
CyPins_FastSlew(CYREG_PRT1_PC2);

/* Set slow edge rate for P1[2] */
CyPins_SlowSlew(CYREG_PRT1_PC2);

/* Set slow edge rate for P1[2] */
       CY_SET_REG8(CYREG_PRT1_SLW, CY_GET_REG8(CYREG_PRT1_SLW) |= 0x02);
```

</td></tr>
</table>

表 14. 出力同期化パラメーター

<table>
<tr><th colspan="2">出力同期化</th></tr>
<tr><td colspan="2">このパラメーターは、ピンの出力ドライバーをバス クロックと同期化します。</td></tr>
<tr><th>PSoC Creator コンフィギュレーション ウィンドウ</th><th>コンポーネント API</th></tr>
<tr><td rowspan="4">
</td><td>該当なし</td></tr>
<tr><th>ピン単位 API</th></tr>
<tr><td>該当なし</td></tr>
<tr><th>関連レジスタ</th></tr>
<tr><td>PRTx_SYNC_OUT[7:0]</td></tr>
<tr><th colspan="2">コード用例</th></tr>
<tr><td colspan="2">

```
/* Sync output of P1[2] to bus_clk using register write. */
      CY_SET_REG8(CYREG_PRT1_SYNC_OUT, CY_GET_REG8(CYREG_PRT1_SYNC_OUT) | 0x04);
```

</td></tr>
</table>

表 15. POR パラメーター

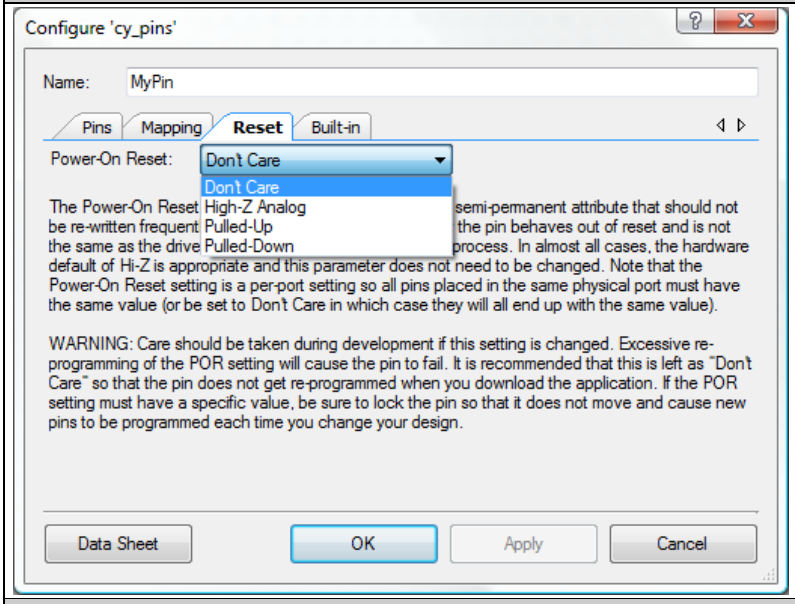
パワーオン リセット (NVL アレイ)	
このパラメーターは、リセット時にピンがどのように挙動するかを決定します。起動プロセス中にコンフィグレーションされる動作駆動モードとは異なります。POR 設定は per-port 設定であることに注意してください。そこでは、同じ物理ポートに配置されたすべてのピンが同じ値を持つ必要があります。このレジスタは NVL アレイの一部で、通常の動作中に変更することはできません。	
PSoC Creator コンフィギュレーション ウィンドウ	**コンポーネント API**
	該当なし
	ピン単位 API
	該当なし
	関連レジスタ
	PRTxRDM[1:0]
コード用例	
NVL アレイのプログラミングの手順と説明については、PSoC 3 Device Programming Specifications または PSoC 5LP Device Programming Specifications を参照してください。	

表 16. イネーブル オプション XRES のパラメーター

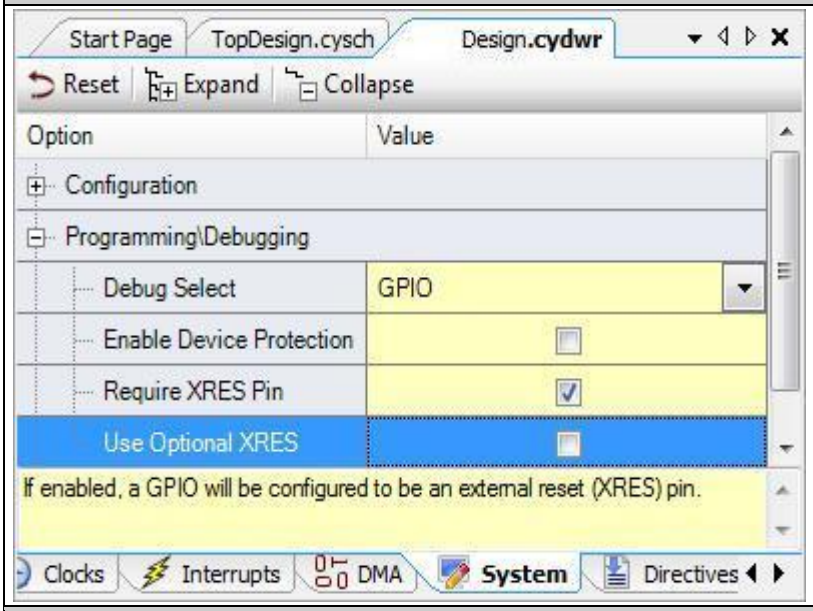
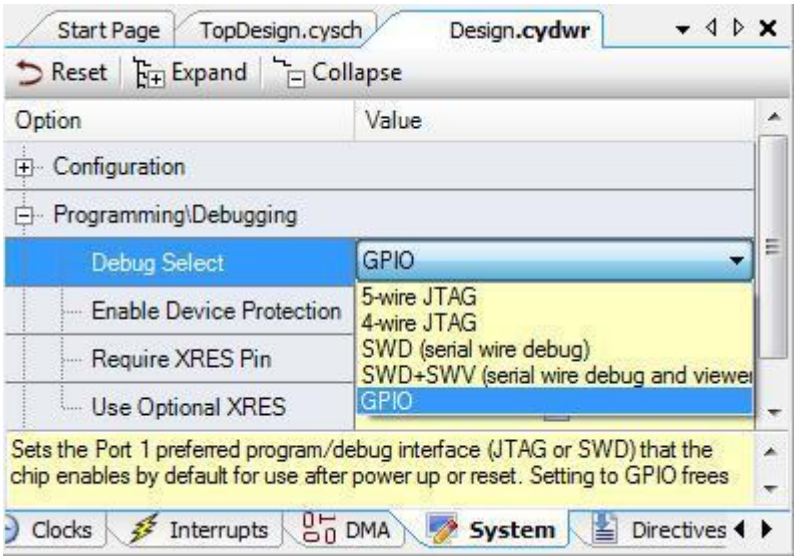
<table>
<tr><th colspan="2">イネーブル オプション XRES (NVL アレイ)</th></tr>
<tr><td colspan="2">このパラメーターは、P1[2]が外部リセット (XRES) ピンとしてコンフィギュレーションされるか否かを決定します。このレジスタは NVL アレイの一部で、通常の動作中に変更することはできません。</td></tr>
<tr><th>PSoC Creator コンフィギュレーション ウィンドウ</th><th>コンポーネント API</th></tr>
<tr><td rowspan="5">
</td><td>該当なし</td></tr>
<tr><th>ピン単位 API</th></tr>
<tr><td>該当なし</td></tr>
<tr><th>関連レジスタ</th></tr>
<tr><td>XRESMEN[0]</td></tr>
<tr><th colspan="2">コード用例</th></tr>
<tr><td colspan="2">NVL アレイのプログラミングの手順と説明については、PSoC 3 Device Programming Specifications または PSoC 5LP Device Programming Specifications を参照してください。</td></tr>
</table>

表 17. デバッグ ポート選択のパラメーター

<table>
<tr><th colspan="2">デバッグ ポート選択 (NVL アレイ)</th></tr>
<tr><td colspan="2">このパラメーターは、適切なプログラミングとデバッギング インターフェースを設定このレジスタは、NVL アレイの一部で、通常の動作中に変更することはできません。</td></tr>
<tr><th>PSoC Creator コンフィグレーション ウィンドウ</th><th>コンポーネント API</th></tr>
<tr><td rowspan="5">
</td><td>該当なし</td></tr>
<tr><th>ピン単位 API</th></tr>
<tr><td>該当なし</td></tr>
<tr><th>関連レジスタ</th></tr>
<tr><td>DPS[1:0]</td></tr>
<tr><th colspan="2">コード用例</th></tr>
<tr><td colspan="2">NVL アレイのプログラミングの手順と説明については、PSoC 3 Device Programming Specifications または PSoC 5LP Device Programming Specifications を参照してください。</td></tr>
</table>

改訂履歴

文書名: AN72382 – PSoC® 3 と PSoC 5LP の GPIO ピンの使用

文書番号: 001-97883

版	ECN	改版者	提出日	変更内容
**	4802470	HZEN	07/10/2015	これは英語版 001-72382 Rev. *F を翻訳した日本語版 001-97883 Rev. ** です。
*A	6364025	SSAS	10/25/2018	これは英語版 001-72382 Rev. *H を翻訳した日本語版 001-97883 Rev. *A です。

セールス, ソリューションおよび法律情報

ワールドワイドな販売と設計サポート

サイプレスは、事業所、ソリューション センター、メーカー代理店、および販売代理店の世界的なネットワークを保持しています。お客様の最寄りのオフィスについては、サイプレスのロケーション ページをご覧ください。

製品

Arm® Cortex® Microcontrollers	cypress.com/arm
車載用	cypress.com/automotive
クロック&バッファ	cypress.com/clocks
インターフェース	cypress.com/interface
IoT (モノのインターネット)	cypress.com/iot
メモリ	cypress.com/memory
マイクロコントローラ	cypress.com/mcu
PSoC	cypress.com/psoc
電源用 IC	cypress.com/pmic
タッチ センシング	cypress.com/touch
USB コントローラー	cypress.com/usb
ワイヤレス	cypress.com/wireless

PSoC® ソリューション

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

サイプレス開発者コミュニティ

コミュニティ | Projects | ビデオ | ブログ | トレーニング | Components

テクニカルサポート

cypress.com/support

Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2011-2018. 本書面は, Cypress Semiconductor Corporation 及び Spansion LLC を含むその子会社 (以下「Cypress」という。) に帰属する財産である。本書面 (本書面に含まれ又は言及されているあらゆるソフトウェア若しくはファームウェア (以下「本ソフトウェア」という。) を含む) は, アメリカ合衆国及び世界のその他の国における知的財産法令及び条約に基づき Cypress が所有する。Cypress はこれらの法令及び条約に基づく全ての権利を留保し, 本段落で特に記載されているものを除き, その特許権, 著作権, 商標権又はその他の知的財産権のライセンスを一切許諾しない。本ソフトウェアにライセンス契約書が伴っておらず, かつ Cypress との間で別途本ソフトウェアの使用方法を定める書面による合意がない場合, Cypress は, (1) 本ソフトウェアの著作権に基づき, (a) ソースコード形式で提供されている本ソフトウェアについて, Cypress ハードウェア製品と共に用いるためにのみ, かつ組織内部でのみ, 本ソフトウェアの修正及び複製を行うこと, 並びに (b) Cypress のハードウェア製品ユニットに用いるためにのみ, (直接又は再販売者及び販売代理店を介して間接のいずれかで) 本ソフトウェアをバイナリーコード形式で外部エンドユーザーに配布すること, 並びに (2) 本ソフトウェア (Cypress により提供され, 修正がなされていないもの) が抵触する Cypress の特許権のクレームに基づき, Cypress ハードウェア製品と共に用いるためにのみ, 本ソフトウェアの作成, 利用, 配布及び輸入を行うことについての非独占的で譲渡不能な一身専属的ライセンス (サブライセンスの権利を除く) を付与する。本ソフトウェアのその他の使用, 複製, 修正, 変換又はコンパイルを禁止する。

適用される法律により許される範囲内で, Cypress は, 本書面又はいかなる本ソフトウェア若しくはこれに伴うハードウェアに関しても, 明示又は黙示をとわず, いかなる保証 (商品性及び特定の目的への適合性の黙示の保証を含むがこれらに限られない) も行わない。いかなるコンピューティングデバイスも絶対に安全ということはない。従って, Cypress のハードウェアまたはソフトウェア製品に講じられたセキュリティ対策にもかかわらず, Cypress は, Cypress 製品への権限のないアクセスまたは使用といったセキュリティ違反から生じる一切の責任を負わない。加えて, 本書面に記載された製品には, エラッタと呼ばれる設計上の欠陥またはエラーが含まれている可能性があり, 公表された仕様とは異なる動作をする場合がある。適用される法律により許される範囲内で, Cypress は, 別途通知することなく, 本書面を変更する権利を留保する。Cypress は, 本書面に記載のある, いかなる製品若しくは回路の適用又は使用から生じる一切の責任を負わない。本書面で提供されたあらゆる情報 (あらゆるサンプルデザイン情報又はプログラムコードを含む) は, 参照目的のためのみに提供されたものである。この情報で構成するあらゆるアプリケーション及びその結果としてのあらゆる製品の機能性及び安全性を適切に設計, プログラム, かつテストすることは, 本書面のユーザーの責任において行われるものとする。Cypress 製品は, 兵器, 兵器システム, 原子力施設, 生命維持装置若しくは生命維持システム, 蘇生用の設備及び外科的移植を含むその他の医療機器若しくは医療システム, 汚染管理若しくは有害物質管理の運用のために設計され若しくは意図されたシステムの重要な構成部分としての使用, 又は装置若しくはシステムの不具合が人身傷害, 死亡若しくは物的損害を生じさせるようなその他の使用 (以下「本目的外使用」という。) のためには設計, 意図又は承認されていない。重要な構成部分とは, それの不具合が装置若しくはシステムの不具合を生じさせるか又はその安全性若しくは実効性に影響すると合理的に予想できるような装置若しくはシステムのあらゆる構成部分をいう。Cypress 製品のあらゆる本目的外使用から生じ, 若しくは本目的外使用に関連するいかなる請求, 損害又はその他の責任についても, Cypress はその全部又は一部をとわず一切の責任を負わず, かつ Cypress はそれら一切から本書により免除される。Cypress は Cypress 製品の本目的外使用から生じ又は本目的外使用に関連するあらゆる請求, 費用, 損害及びその他の責任 (人身傷害又は死亡に基づく請求を含む) から免責補償される。

Cypress, Cypress のロゴ, Spansion, Spansion のロゴ及びこれらの組み合わせ, WICED, PSoC, CapsSense, EZ-USB, F-RAM, 及び Traveo は, 米国及びその他の国における Cypress の商標又は登録商標である。Cypress のより完全な商標のリストは, cypress.com を参照すること。その他の名称及びブランドは, それぞれの権利者の財産として権利主張がなされている可能性がある。