PSoC™ 4 MCU - GPIO ピンの使用

本書について

適用範囲と目的

AN86439 は特長を実証するための様々な用途例を使用し、PSoC™ 4 MCU の GPIO ピンを効果的に使用する方法を説明します。本アプリケーションノートの主な内容には、GPIO の概要, 設定オプション, 混合信号の使用, 割込み, および低電力動作が含まれます。

対象者

本書は、PSoC™ 4 MCU GPIO ピンを使用するすべての人を対象とします。

目次

目次

本書について 1
目次 2
1 はじめに 4
2 PSoC™リソース 5
2.1 PSoC™ Creator 5
2.1.1 PSoC™ Creator サンプルコード 6
2.1.2 PSoC™ Creator ヘルプ 7
2.2 ModusToolbox™ ソフトウェア 8
2.2.1 ModusToolbox™サンプルコード 8
2.3 テクニカルサポート 10
3 GPIO ピンの基本情報 11
3.1 GPIO ピンの物理構造 11
3.2 ピン配線 13
3.2.1 デジタル配線 13
3.2.2 アナログ配線 15
3.3 スタートアップおよび低電力動作 24
3.4 GPIO 割込み 25
3.4.1 GPIO 割込みの制限 27
4 過電圧耐性 (OVT) ピン 28
5 PSoC™ Creator での GPIO ピン 29
5.1 ピン コンポーネント シンボル 29
5.2 ピンコンポーネントのカスタマイザ 29
5.3 ピンコンポーネント割込み 32
5.4 手動のピン割り当て 35
5.5 PSoC™ Creator の API 35
5.6 GPIO ピンのデバッグロジック 35
5.7 複数の GPIO ピンを論理ポートとして追加 36
5.8 オフチップ コンポーネントについて 38
6 ModusToolbox™での GPIO ピン 41
6.1 ModusToolbox™ Device Configurator を使用した GPIO ピンの設定 41
6.1.1 Device Configurator の使用 41
6.1.2 Device Configurator コードプレビュー 44
6.2 Peripheral Driver Library (PDL) を使用した GPIO 44
6.2.1 GPIO ピンの初期化-フル 45
6.2.2 GPIO ピンの初期化-高速 46
6.2.3 GPIO ポートの初期化 46
6.2.4 GPIO ピンからの読み出し 47
6.2.5 GPIO ピンへの書き込み 48
6.2.6 GPIO 割込み 48
7 PSoC™ Creator での GPIO のヒントおよびコツ 49
7.1 LED をトグル 49
7.2 入力の読み出しと出力への書き込み 51
7.3 デジタル ロジック ゲートから出力の駆動 52
7.4 双方向ピンの使用 52
7.5 GPIO 入力/出力同期の設定 54
7.5.1 GPIO 入力同期 57
7.5.2 GPIO 出力同期 58
7.6 データレジスタでの GPIO のより速いトグル 59

目次

7.7 GPIO 出力イネーブル ロジックの設定 63
7.8 ピン割込み 65
7.9 ファームウェアでの GPIO 割込みの設定 68
7.10 GPIO でのアナログとデジタルの両方の使用 70
7.11 より大きな駆動/シンク電流のためのピンの連動 74
7.12 ディープスリープにおける制御レジスタの取り扱い 77
8 ModusToolbox™での GPIO のヒントおよびコツ 81
8.1 入力の読み出しと出力への書き込み 81
8.2 ピン割込み 81
8.3 その他のサンプルコード 81
9 関連アプリケーションノート 82
10 PSoC™ 4 GPIO と PSoC™ 1, PSoC™ 3, および PSoC™ 5LP GPIO との比較 83
11 PSoC™ 4 開発ボード 84
改訂履歴 85
免責事項 86

1 はじめに

PSoC™は、従来の MCU と比べてより多くの機能を提供する柔軟な汎用 I/O (GPIO) アーキテクチャを有します。PSoC™ GPIO は従来の MCU のようにファームウェア レジスタの設定で制御することだけでなく、カスタム デジタル ロジックおよびアナログブロック信号からも駆動されます。本アプリケーションノートは PSoC™ 4 および PSoC™アナログ コプロセッサ GPIO ピンの基本情報および異なる機能のために、効果的な使用方法を説明します。

本アプリケーションノートは読者が PSoC™ Creator または ModusToolbox™および PSoC™ 4 アーキテクチャに精通していることを前提とします。PSoC™ 4 が初めてである場合、**AN79953 – PSoC™ 4 入門**を参照してください。PSoC™ Creator を初めて使用する場合は、**PSoC™ Creator ホームページ**を参照してください。ModusToolbox™を初めて使用する場合は、**ModusToolbox™ホームページ**を参照してください。デバイス パッケージまたは GPIO 仕様の詳細情報は **PSoC™ 4 データシート**を参照してください。デバイス [1] および開発環境に精通されている場合は、**PSoC™ Creator での GPIO のヒントおよびコツ**または **ModusToolbox™での GPIO のヒントおよびコツ**に進んでください。

このアプリケーションノートでは、PSoC™ 4 デバイスでの開発のための PSoC™ peripheral driver library (PDL) と ModusToolbox™の使用方法について説明します。これは現在、PSoC™ 4 S シリーズデバイスを使用している場合にのみ対応します。

Note: 「PSoC™」または「デバイス」はここで、特に指定しない限り、PSoC™ 4 を指します。

2 PSoC™リソース

インフィニオンは、**www.infineon.com** に大量のデータを掲載しており、ユーザーがデザインに適切なPSoC™デバイスを選択し、デバイスを設計に迅速で効果的に統合する手助けをします。リソースの包括的なリストについては **KBA86521 - How to Design with PSoC™ 3, PSoC™ 4, and PSoC™ 5LP** を参照してください。以下は PSoC™ 4 のリソースの要約です。

- **概要: MCU Portfolio, PSoC™, MCU Roadmap**
- **製品セレクター: PSoC™ 1**, **PSoC™ 3**, **PSoC™ 4**, **PSoC™ 5LP**。また、**PSoC™ Creator** にはデバイス選択ツールが含まれます。
- **データシート**は PSoC™ 3, PSoC™ 4, PSoC™ 5LP, および PSoC™ 6 MCU デバイスファミリの電気的仕様を説明し、提供します。
- **CAPSENCE™デザインガイド:** PSoC™ 4 および PSoC™ 6 MCU デバイスファミリを使用して、静電容量タッチセンシング アプリケーションを設計する方法について説明します。
- **アプリケーションノート:** 基本的なレベルから上級者レベルまで、幅広いトピックを提供します。
- **サンプルコード: PSoC™ 3, PSoC™ 4, および PSoC™ 5LP** 向け、または **PSoC™ 6 MCU** 向け。
- **テクニカルリファレンスマニュアル (TRM)** は、PSoC™ 3, PSoC™ 4, および PSoC™ 5LP (PSoC™ 6 デバイスを追加?) デバイスファミリのそれぞれにおける、アーキテクチャおよびレジスタの詳細な説明を提供します。
- **開発キット:**
  - **CY8CKIT-040**, **CY8CKIT-041**, **CY8CKIT-042**, **CY8CKIT-042-BLE**, **CY8CKIT-044**, および **CY8CKIT-046** Pioneer キットは使いやすく、安価な開発プラットフォームです。これらのキットには、Arduino 互換シールドおよび Digilent Pmod ドーターカード用のコネクタを搭載します。
  - **CY8CKIT-043**, **CY8CKIT-049**, **CY8CKIT- 145**, および **CY8CKIT-149** は、PSoC™ 4 デバイスをサンプリングする用途の低コスト プロトタイプ プラットフォームです。**CY8CKIT-001** はすべての PSoC™ファミリ デバイスの共通開発プラットフォームです。
- **MiniProg3** デバイスは PSoC™ Creator 使用時のフラッシュのプログラムとデバッグ用のインターフェースを提供します。
- **MiniProg4** デバイスは ModusToolbox™使用時のフラッシュのプログラムとデバッグ用のインターフェースを提供します。

2.1 PSoC™ Creator

PSoC™ Creator は Windows ベースの統合開発環境 (IDE) であり、無料で利用できます。これにより PSoC™ 3, PSoC™ 4, および PSoC™ 5LP に基づく、システムのハードウェアとファームウェアの同時設計が可能です。**Figure 1** に示すように、PSoC™ Creator により以下のことが可能です。

1. コンポーネントをドラッグアンドドロップして、メイン デザイン ワークスペースでハードウェア システム デザインを構築
2. PSoC™ハードウェアとアプリケーション ファームウェアを同時設計
3. コンフィギュレーション ツールを用いてコンポーネントを設定
4. 100 以上のコンポーネントを含むライブラリを利用
5. コンポーネント データシートの閲覧

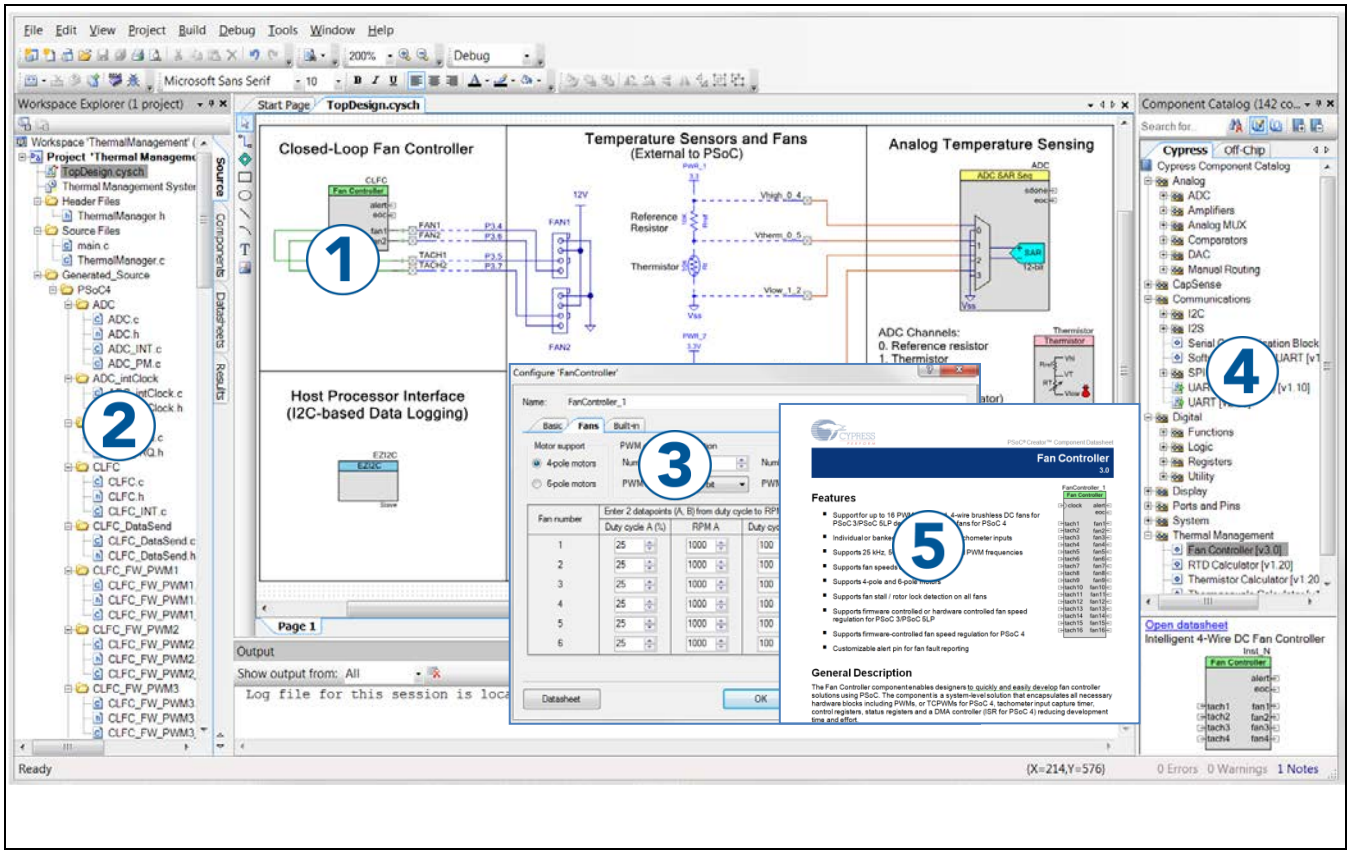


Figure 1 PSoC™ Creator の特長

2.1.1 PSoC™ Creator サンプルコード

PSoC™ Creator は多数のサンプルコードのプロジェクトを含みます。これらのプロジェクトは **Figure 2** に示すように、PSoC™ Creator のスタート ページからアクセスできます。PSoC™ Creator のサンプルコードは、**infineon.com** からダウンロードもできます。

サンプルプロジェクトにより、最初 (空のページ) からではなく完成した設計で始めるため、設計時間を短縮できます。サンプルプロジェクトは PSoC™ Creator コンポーネントを様々なアプリケーションに使用する方法も示します。**Figure 3** に示すように、サンプルコードおよびデータシートが含まれます。

Figure 3 に示す **Find Code Example Project** ダイアログにはいくつかのオプションがあります。

- デバイスファミリ (PSoC™ 3, PSoC™ 4, または PSoC™ 5LP など)、カテゴリやキーワードに基づいてサンプルプロジェクトをフィルターしてください。
- **Filter Options** に基づいてフィルターされたリストからサンプルプロジェクトを選択してください。
- 選択のためにデータシートをレビューしてください (**Documentation** タブ上で)。
- 選択したプロジェクトのサンプルコードをレビューしてください。コード開発時間を短縮するためにユーザーはこのウィンドウからコードをコピーして自身のプロジェクトに貼り付けることができます。

選択したものに基づいて新規プロジェクト (および必要な場合は新規ワークスペース) を作成してください。これにより、完成した基本設計で開始し、設計時間を短縮させることができます。それから設計を、所望のアプリケーションに適用できます。

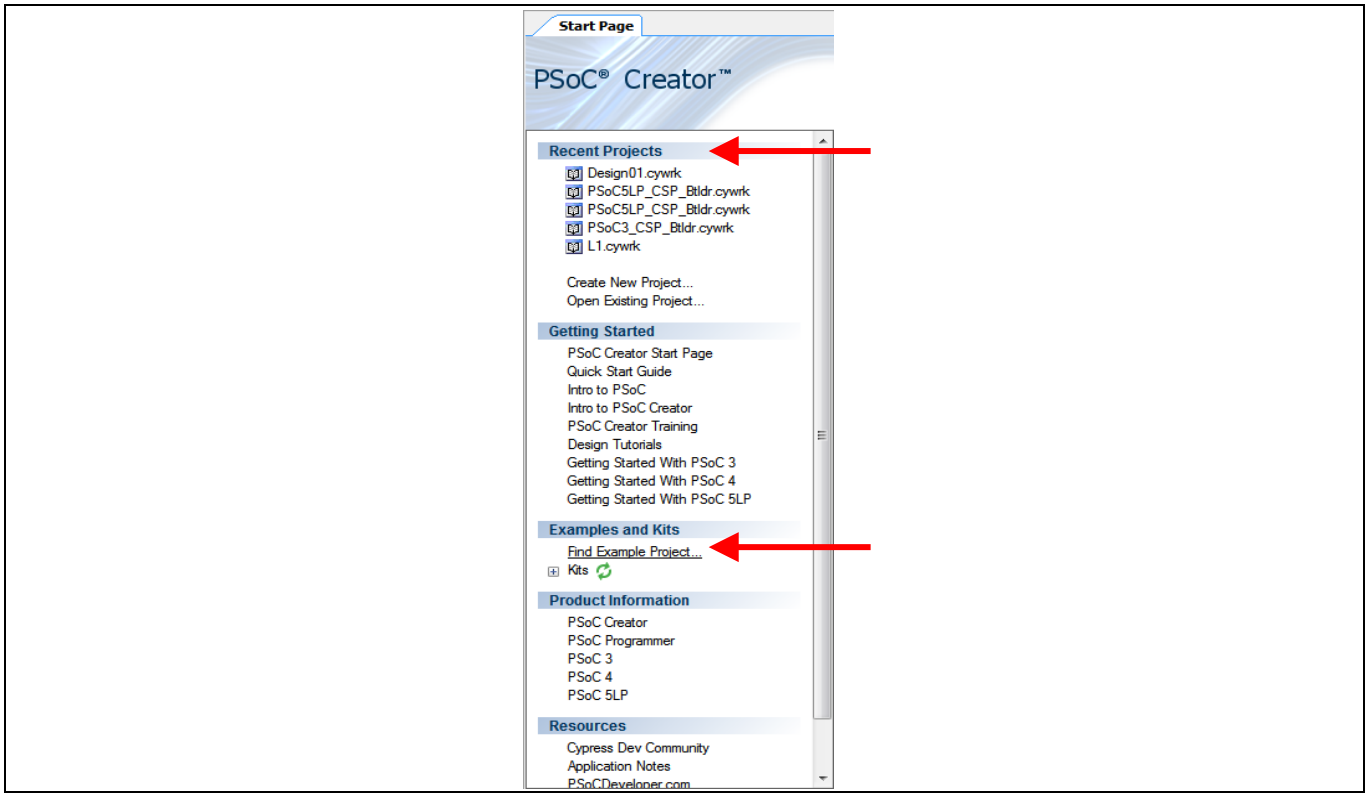


Figure 2 PSoC™ Creator のサンプルコード

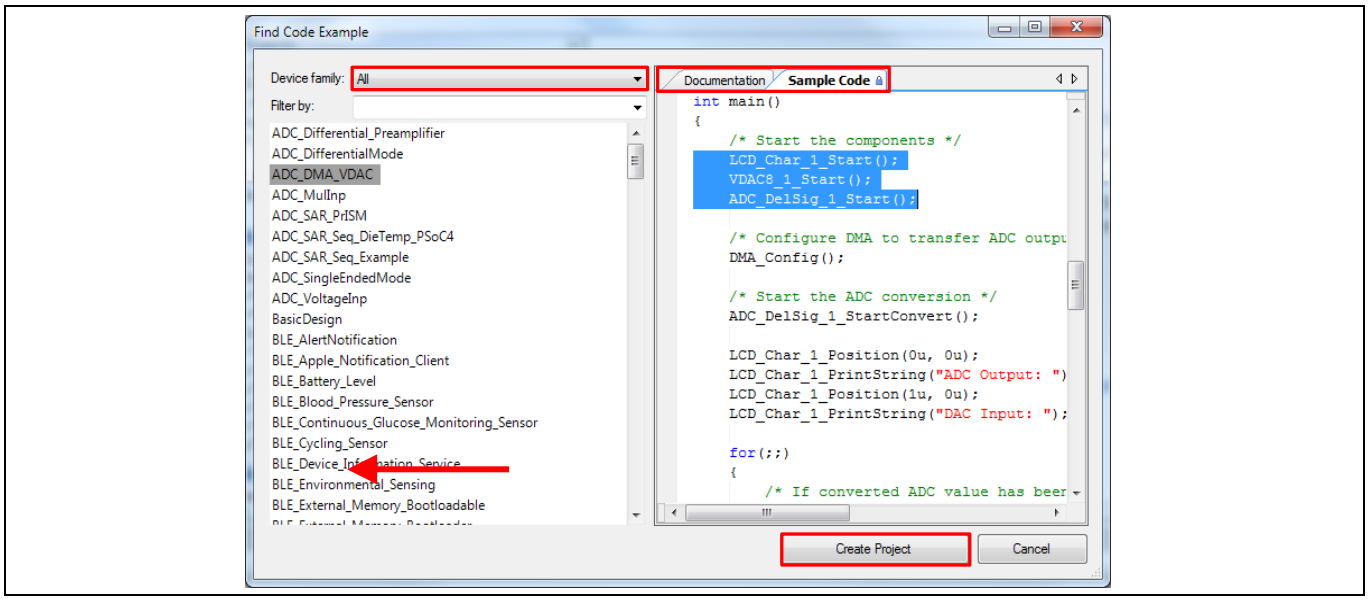


Figure 3 サンプルプロジェクトおよびサンプルコード

2.1.2 PSoC™ Creator ヘルプ

PSoC™ Creator ホームページへアクセスし、PSoC™ Creator の最新版をダウンロードしてください。次に、PSoC™ Creator を起動して以下のアイテムへ移動してください。

- **クイックスタートガイド**: **Help** > **Documentation** > **Quick Start Guide** を選択します。このガイドは PSoC™ Creator プロジェクトを開発するための基礎知識を提供します。
- **システム リファレンスガイド**: **Help** > **System Reference Guide** を選択します。このガイドは、PSoC™ Creator により提供されるシステム機能を一覧にして、説明します。
- **コンポーネント データシート**: コンポーネントを右クリックして「Open Datasheet」を選択してください。すべての PSoC™ 4 のコンポーネント データシートの一覧については **PSoC™ 4 Component Datasheets** ページをアクセスしてください。
- **ドキュメントマネージャー**: PSoC™ Creator が提供するドキュメントマネージャーにより、ドキュメントリソースを容易に検索し、レビューできます。ドキュメントマネージャーを開くためには、メニューアイテムの **Help** > **Document Manager** を選択します。

2.2 ModusToolbox™ ソフトウェア

ModusToolbox™は、マルチプラットフォーム開発ツールのセットであり、GitHub でホストされているファームウェアライブラリの包括的な一式です。これらを組み合わせることで、統合された MCU およびワイヤレスシステムを作成する顧客に没入型の開発エクスペリエンスを提供できます。

ファームウェアライブラリは、インフィニオン PSoC™ 6 MCU, PSoC™ 4, および Bluetooth® SoC (20xxx) キット用の簡単にカスタマイズ可能な board support package (BSP) と、業界をリードする機能を可能にするミドルウェアライブラリの包括的なセットで構成されます。

- CAPSENCE™
- Bluetooth® Low Energy と Mesh
- 市場で最も低電力で最も信頼性の高い Wi-Fi
- 徹底的にテストされた役立つサンプルコードアプリケーションの印象的なセット

ModusToolbox™ホームページにアクセスして、ModusToolbox™の最新バージョンをダウンロードしてください。以下は、ModusToolbox™の使用を開始するために役立つ項目です。

- **Quick Start Guide**: これは、特に Eclipse ベースの IDE を使用して ModusToolbox™のアプリケーションを作成および構築するための短いステップバイステップガイドです。
- **ModusToolbox™ User Guide**: このガイドでは、Eclipse IDE に焦点を当て、IDE とソフトウェア機能の詳細について説明します。
- **Documentation**: ModusToolbox™にあるクイックパネルのドキュメントセクションまでスクロールします。

*Note: ModusToolbox™は、KitProg3 および **MiniProg4** プログラミングデバイスと互換性があります。*

2.2.1 ModusToolbox™サンプルコード

ModusToolbox™には、増え続ける多くのサンプルコードプロジェクトが含まれます。これらのサンプルコードプロジェクトは、ModusToolbox™または **Infineon GitHub** の新しいアプリケーションウィザードで利用できます。

サンプルプロジェクトでは、空白のページではなく完成した設計から始めることで、設計プロセスをスピードアップできます。

新しいアプリケーションウィザードでは、**Figure 4** に示すように、board support package (BSP) を選択できます。BSP は、使用される特定のキットに対応します。**Figure 5** に示すように、BSP を選択すると、サンプルコードを表示できます。

PSoC™リソース

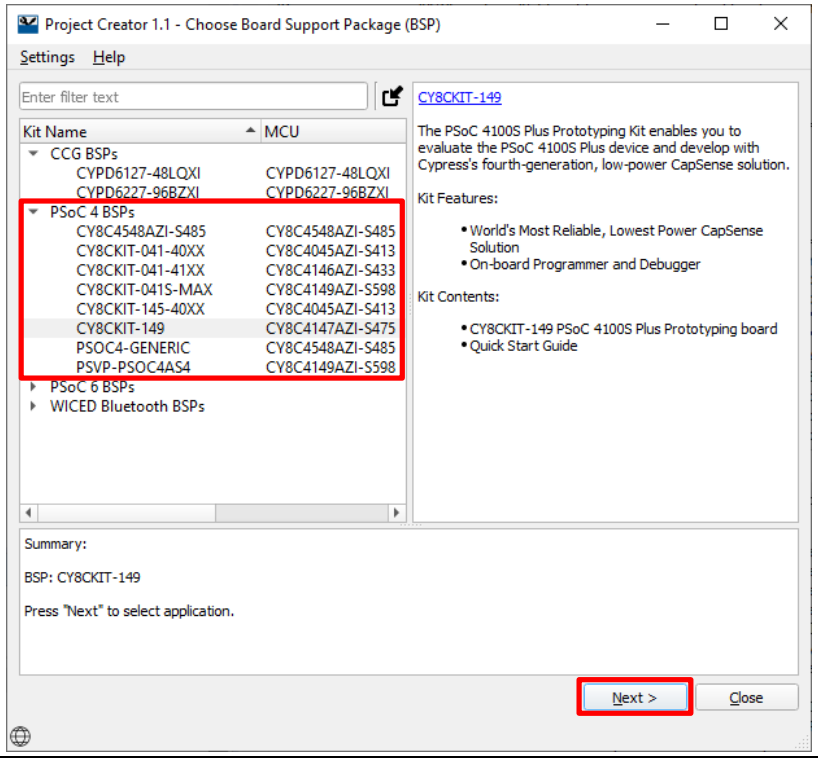


Figure 4　新しいアプリケーションウィザード

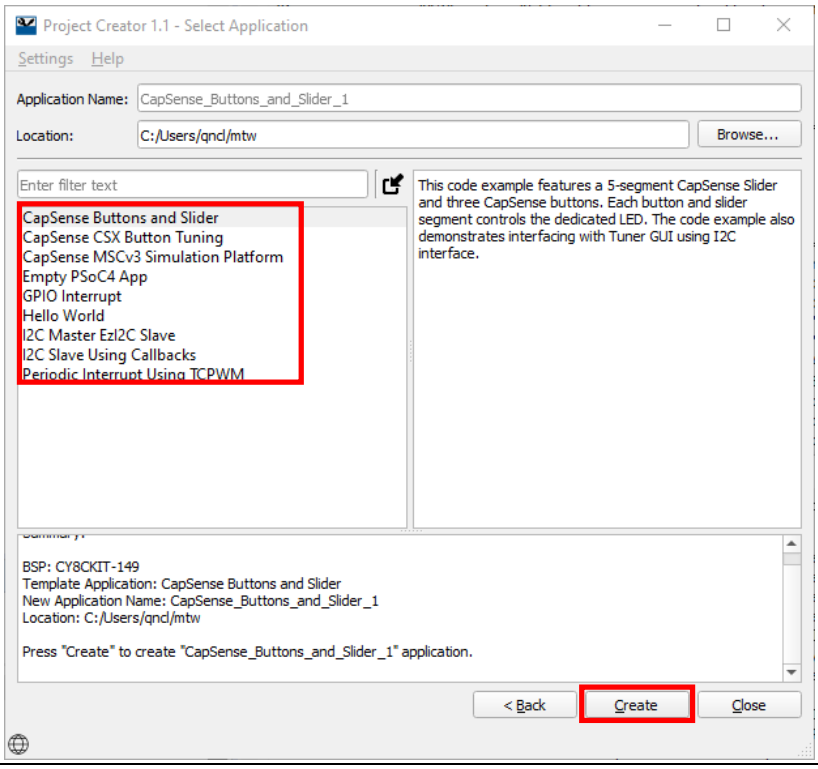


Figure 5　サンプルコード

2.3 テクニカルサポート

ご質問には弊社のテクニカルサポートチームが対応させて頂きますので、お気軽にご連絡ください。**Infineon Technical Support** ページにアクセスし、サポートリクエストを作成してください。

米国のお客様は、テクニカルサポートチームに連絡する際、以下の電話番号 (通話無料) にお問い合わせください。+1-800-541-4736 プロンプトでオプション「8」を選択してください。

早急な対応が必要な場合は、以下のサポートリソースをご利用ください。

- セルフヘルプ

3 GPIO ピンの基本情報

PSoC™ GPIO ピンは以下の機能を提供します。

- アナログ-デジタル入出力機能
- LCD セグメント駆動の対応 (PSoC™ 4000 および PSoC™ 4200DS では利用不可)
- CAPSENCE™の対応
- 立ち上りエッジ、立ち下りエッジ、または両方のエッジでのレベルの割込み
- スルーレート制御
- 入力閾値の選択 (CMOS/LVTTL)
- ホットスワップ機能を持つ過電圧耐性ピン (PSoC™ 4 Bluetooth® LE, PSoC™ 4 M シリーズ, および PSoC™ 4 L シリーズでのみ利用可能)

GPIO 機能は PSoC™ 4 デバイスで使用可能なペリフェラルに依存します。異なる PSoC™ 4 ファミリで利用可能な機能の横比較については **AN79953 – PSoC™ 4 入門**の表 1 を参照してください。

3.1 GPIO ピンの物理構造

Figure 6 に PSoC™デバイスでリソースとのピン接続を示します。

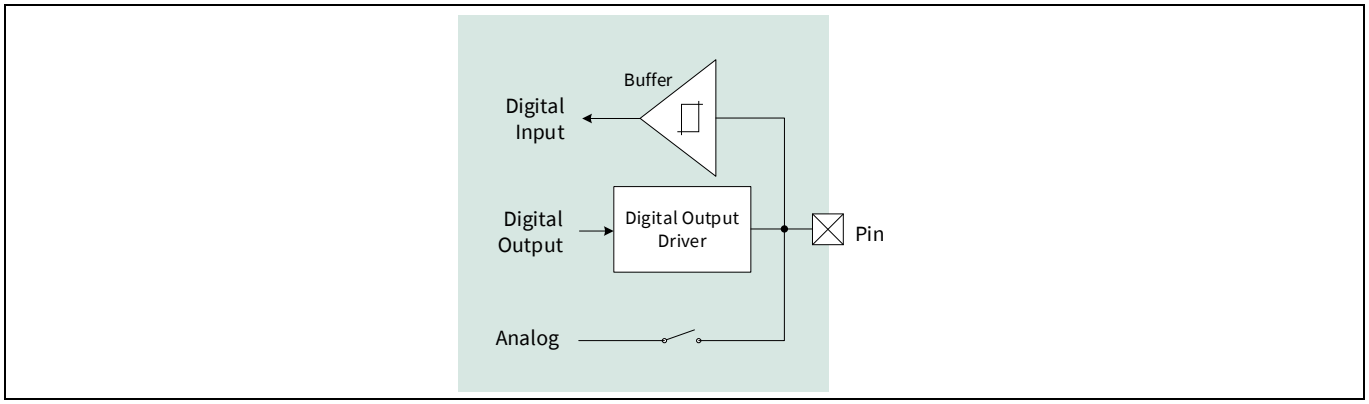


Figure 6 簡略化した GPIO ブロックダイヤグラム

GPIO 構造の詳細ブロックダイヤグラムは **PSoC™ 4 architecture TRM** の「I/O System」節に掲載されています。各ピンは CPU およびタイマー、PWM、または I^2C などのデジタルペリフェラルの入力または出力として動作できます。オペアンプおよび ADC 用のアナログピンとしても動作できます。任意の時点で、デジタル専用の入力, デジタル専用の出力, アナログ専用, またはそれら 3 つの組合せのピンを使用できます。例えば、デジタル出力およびデジタル入力の両方を有効にする場合、デジタル双方向ピンとなります。入力バッファは高インピーダンスを外部入力に提供します。それは CMOS, LVTTL に設定可能です。

入力閾値の値は**デバイス データシート**を参照してください。

デジタル出力ドライバーは異なる駆動モードおよびスルーレート制御をサポートします (**Figure 7** を参照してください)。

GPIO ピンの基本情報

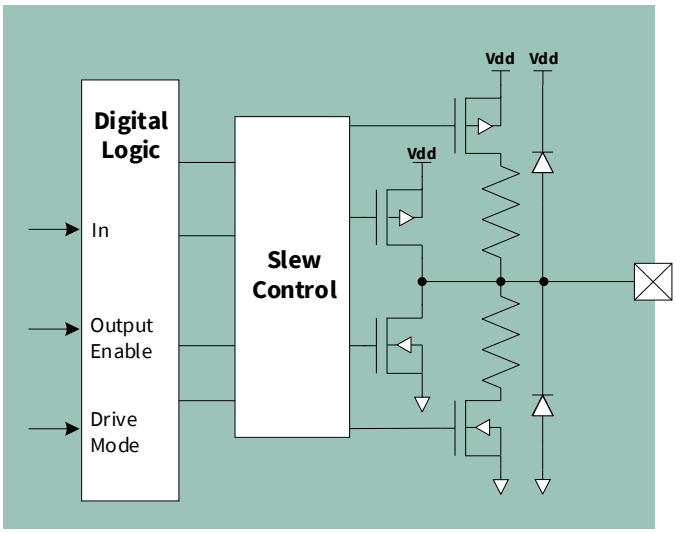


Figure 7　デジタル出力ドライバー

スルーレート制御は EMI およびクロストークを低減するために提供されます。高速と低速の 2 つのオプションがあります。スルーレートはデフォルトで高速に設定されます。速度が重要ではない信号の場合、低速オプションを使用します。

Table 1 に示すように、**Figure 7** に示す回路は 8 つの駆動モードに対応します。

Table 1　駆動モードおよびアプリケーション

#	駆動モード	応用例
1	高インピーダンス アナログ	アナログ入力/出力
2	高インピーダンス デジタル	デジタル入力
3	抵抗プルアップ (~5 kΩ)	モーターからのタコメーター出力またはグランドに接続されたスイッチのようなオープンドレイン LOW 入力へのインターフェースとして使用。LED の駆動にも使用可能。
4	抵抗プルダウン (~5 kΩ)	オープンドレイン HIGH 入力または VDD に接続されたスイッチへのインターフェースとして使用。電流吸い込み (シンク) モードでの LED とインターフェースするための出力として使用可能。
5	オープンドレイン, LOW に駆動	HIGH 状態で高インピーダンスおよび LOW 状態でストロングドライブを提供。この設定は I²C ピンのために使用される。このモードは、外部プルアップ抵抗と連動して動作する。
6	オープンドレイン, HIGH に駆動	HIGH 状態でストロングドライブおよび LOW 状態で高インピーダンスを提供。このモードは、外部プルダウン抵抗とともに動作。
7	ストロングドライブ	LOW 状態および HIGH 状態の両方で CMOS 出力ドライブを提供。
8	抵抗プルアップおよび抵抗プルダウン (~5 kΩ)	HIGH および LOW の両方の状態で直列抵抗を追加。

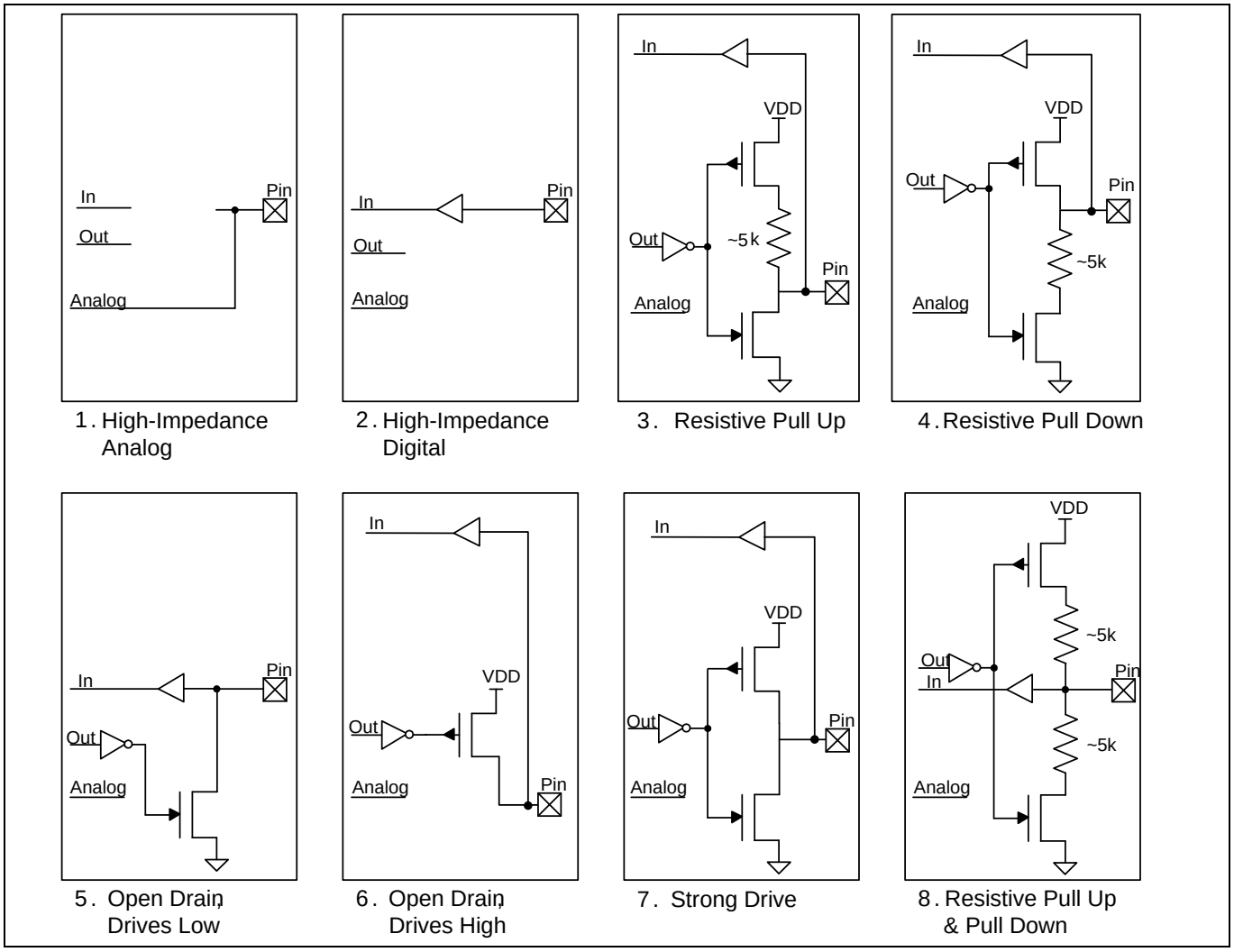


Figure 8　駆動モード

Note: ***Figure 8*** *に示すプルアップおよびプルダウン駆動モード用の抵抗値は近似値です。抵抗値の仕様については、****デバイス データシート****を参照してください。高い精度が必要となる場合、外部抵抗を使用します。この場合では、ピンをオープンドレイン HIGH 駆動またはオープンドレイン LOW 駆動として設定する必要があります。*

Note: *常にデバイスの VDD は ESD クランプダイオードを介したピンで外部電圧からの電力を及ばせないようにします。PSoC™ 4 デバイスに電力供給されず、外部電圧が GPIO に適用される場合、または GPIO での外部電圧がデバイス VDD より高い場合、この問題は発生する可能性があります。しかし、クランプダイオードがないため****過電圧耐性 (OVT) ピン****に適用できません。*

3.2　ピン配線

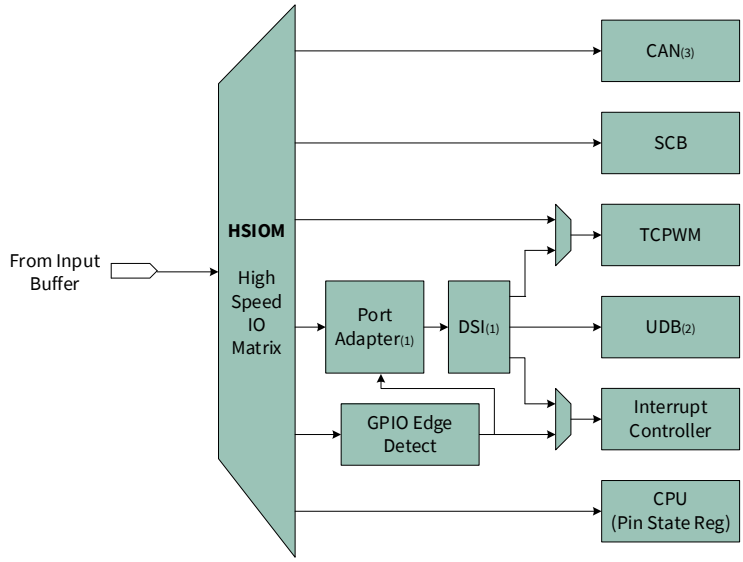
3.2.1　デジタル配線

1 つのピンは、CPU によって読み書きされるデータレジスタと同様に、汎用デジタルブロック (UDB), シリアル通信ブロック(SCB), タイマー/カウンター/パルス幅変調器 (TCPWM) ブロック, LCD ドライバー, CAN ブロック, および割込みコントローラなどのさまざまなデジタルペリフェラルに接続できます。**Figure 9** に入力ピンの配線を示し、**Figure 10** に出力ピンの配線を示します。これらの図に示すように、デバイス

GPIO ピンの基本情報

の周辺機器は高速 I/O マトリクス (HSIOM) でピンに接続されます。このマトリクスは異なるペリフェラルからの信号を多重化し、特定のピンに接続します。

PSoC™では、HSIOM を介して配線される専用 I/O と、デジタルシステム相互接続 (DSI) を使用する柔軟な配線の 2 つの配線が可能です。DSI の使用はペリフェラル入力および出力をピンに配線することに限定されません。デジタルリソース間に信号を配線するためにも使用されます。ポートアダプタは HSIOM と DSI を繋ぎ、ピン入力および出力信号を同期化するためのハードウェアも提供します。



1. Not applicable to PSoC™ 4000, PSoC™ 4000S, PSoC™ 4100S, 4100S Plus, PSoC™ 4100PS
2. Not applicable to PSoC™ 4000, PSoC™ 4000S, PSoC™ 4100PS, PSoC™ 4100, PSoC™ 4100S, PSoC™ 4100S Plus, PSoC™ 41xx-BL, or PSoC™ 4100M
3. Only available in PSoC™ 4200M , PSoC™ 4200L, and 4100S Plus devices

Figure 9　デジタルピン入力パス

SCB (I^2C, UART と SPI) および TCPWM はいくつかの I/O への専用配線があります。柔軟性のある配線のオプションは UDB 入出力に使用可能であり、ピンからの割込みを生成します。そのオプションは TCPWM にも使用可能です。LCD ドライバーは PSoC™デバイスのすべての I/O (PSoC™ 4000 および PSoC™ 4200DS 以外) にあります。いずれかの I/O は LCD 用のセグメントまたは共通ドライバーとして動作します。

GPIO エッジ検出ブロックは立ち上りエッジ, 立ち下りエッジ, および両方のエッジでピン割込みを有効にします。詳細は **GPIO 割込み**を参照してください。

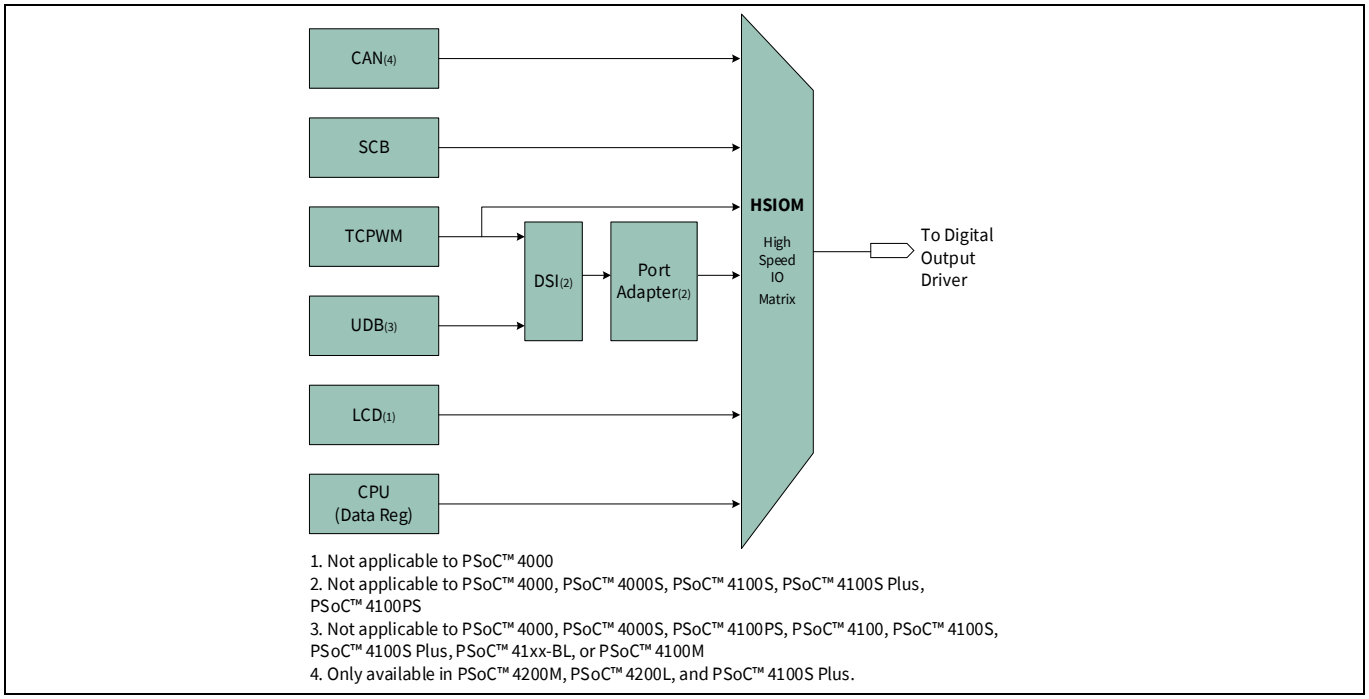


Figure 10　デジタルピン出力パス

Note: *PSoC™ 4 は複数のポートがあり、ポートごとに最大 8 つのピンがあります。PSoC™ 4200L デバイスでは、ポート 7, 8, および 9 のピンにはポートアダプタがありません。他のデバイスでは、ポート 4 以上はポートアダプタがありません。これらのポートは以下の制限があります。*

- DSI を介した配線ができないこと。したがって、UDB ベースのデジタル信号はこれらポートのピンに配線されることが不可能
- SAR ADC, オペアンプ - 連続時間ブロック mini (CTBm), 低電力コンパレータ (PSoC™ 4100, PSoC™ 4100PS, および PSoC™ 4200 のみに適用可能), および 連続時間ブロック(PSoC™ 4100PS のみに適用可能) などのアナログブロックのために使用できないこと
- 入力/出力の同期化なし

しかし、これらポートは以下の方法で有用です。

- ファームウェアで制御される GPIO として使用
- TCPWM, SCB または CAN に直接接続
- LCD および CAPSENCE™™ピン
- 割込みの生成

Note: *PSoC™ デバイスのピンは異なるペリフェラルへの専用接続のために共有されます。各ピンでの機能については、それぞれのデバイス データシートの「Pinouts」節を参照してください。*

3.2.2　アナログ配線

Figure 11～**Figure 16** に示すように、高インピーダンス アナログ (HI-Z) モードで設定された GPIO ピンは直接接続または、アナログスイッチおよびアナログマルチプレクサ (AMUX) バスを介して、アナログリソースに接続されます。

GPIO ピンの基本情報

以下は、**Figure 11** に示した PSoC™ 4000 デバイスでのアナログ配線のキーポイントです。

- すべてのピン (ポート 3 以外) はファームウェアで制御される AMUX バスに接続できます。AMUXBUS_A および AMUXBUS_B の 2 つのバスがあります。
- CAPSENCE™ IDAC0 が AMUXBUS_A に、IDAC1 が AMUXBUS_B に接続されます。
- CAPSENCE™ CMOD が P0[4]に、シールド タンク コンデンサが P0[2]に接続されます。
- CAPSENCE™ブロックが AMUX バスを介してセンサーに接続するため、いずれかのピンは静電容量タッチセンサー (ポート 3 以外) のために使用できます。

Note: *CMOD コンデンサをピンに近く配置します。レイアウトガイドラインについては、**AN85951 - PSoC™ 4 CAPSENCE™ Design Guide** を参照してください。*

以下は、他の PSoC™ 4 デバイスのアナログ配線のキーポイントです。**Figure 13**～**Figure 17** を参照してください。

- 2 つの AMUX バスがあります。すべてのピンは AMUXBUS_A と AMUXBUS_B に接続できます。AMUX バスの接続はファームウェアまたは DSI 信号の使用により制御されます。ポート 4 以上のピンは、DSI 接続が利用できず、AMUX はファームウェアでのみ接続されることに注意してください。
- 直接接続はオペアンプの入出力に利用可能です。これにより、配線抵抗および寄生容量が低くなるため、より良い性能となります。直接接続はスイッチなしで低電力コンパレータ (LPCOMP) 入力にも利用可能です。
- これらは CAPSENCE™ CMOD およびシールド タンク コンデンサのための専用ピンです。ピンの詳細は **Figure 13**～**Figure 17** を参照してください。
- CAPSENCE™ IDAC0 が AMUXBUS_A に、IDAC1 が AMUXBUS_B に接続されます。
- CAPSENCE™ブロックが AMUX バスを介してセンサーに接続するため、いずれかのピンは静電容量タッチセンサーのために使用できます。
- **Figure 14**～**Figure 17** に示すように、AMUXBUS_A および AMUXBUS_B はスイッチ (青色でマーク) を使用して分割できます。これは、AMUX バスが、入力と出力がシステムで CAPSENCE™と共に配線されるオペアンプ/コンパレータなどの非 CAPSENCE™アプリケーションに必要とされる場合、有用です。
- SAR シーケンサーは SAR ADC 入力を下記に接続します。
  - PSoC™ 4100, PSoC™ 4100S, PSoC™ 4100S Plus, PSoC™ 4200, PSoC™ 4100M, PSoC™ 4200M, および PSoC™ 4200L のポート 2
  - PSoC™ 41xx-BL, PSoC™ 42xx-BL, および PSoC™ 41xxPS のポート 3
  - CTBm, CTB 出力
  - 温度センサー出力

Figure 13～**Figure 17** の赤色で示されるように、スイッチを制御することにより多重化ができます。SAR ADC はシーケンサーなしで AMUXBUS を使用して任意のピンから入力を受け取ることもできることに注意してください。

Note: *オペアンプ出力はスイッチなしで専用ピンに接続されます。AMUX バスへの接続が必要な場合、専用ピンに関連付けられる AMUX スイッチはアクティブにします。これにより、関連する AMUX スイッチがアクティブになる場合、他のピンはオペアンプ出力ピンとして動作できます。*

Note: *SAR ADC はシーケンサーモードの差動入力で動作する場合、正の入力は偶数ピンである必要があり、負の入力は隣接する奇数ピンである必要があります。例えば、PSoC™ 4200 で、P2[0]と P2[1]のペアでは、P2[0]は正の入力で、P2[1]は負の入力です。これはアナログ配線図にリング形 を使用して示します。*

GPIO ピンの基本情報

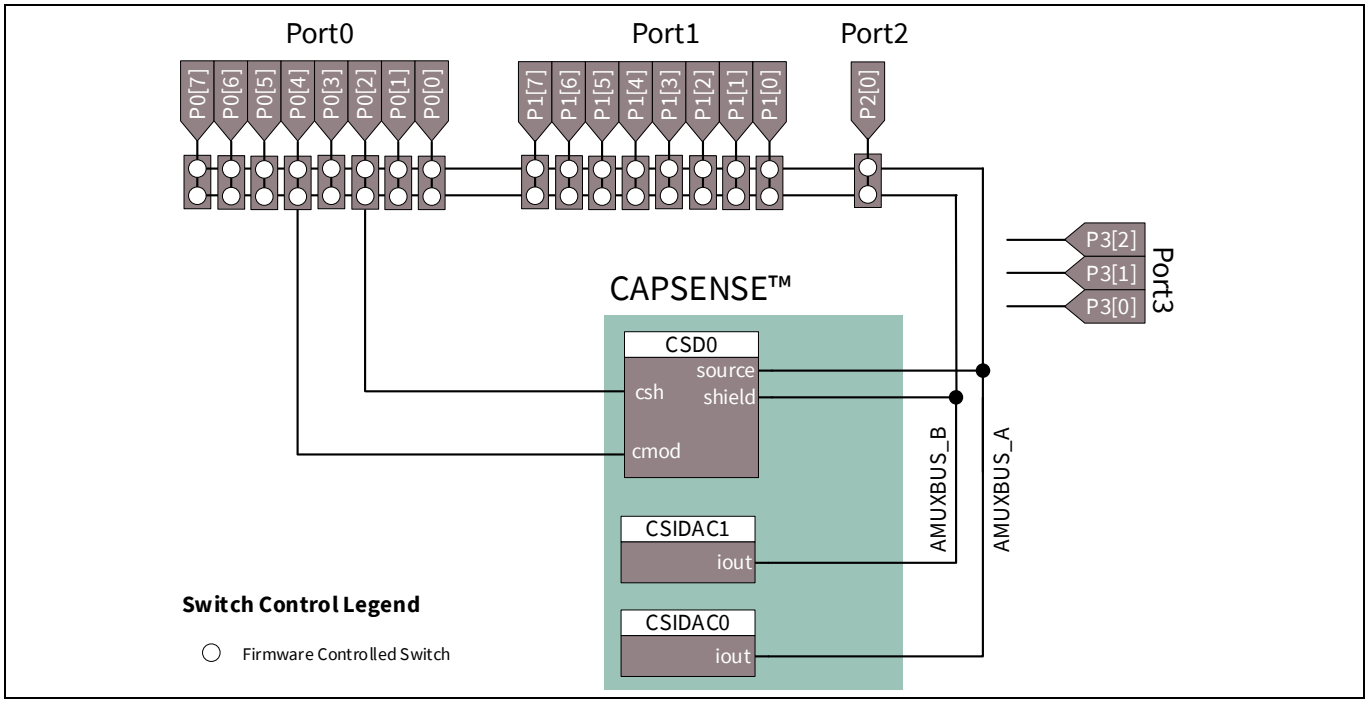


Figure 11 PSoC™ 4000 のアナログ配線図

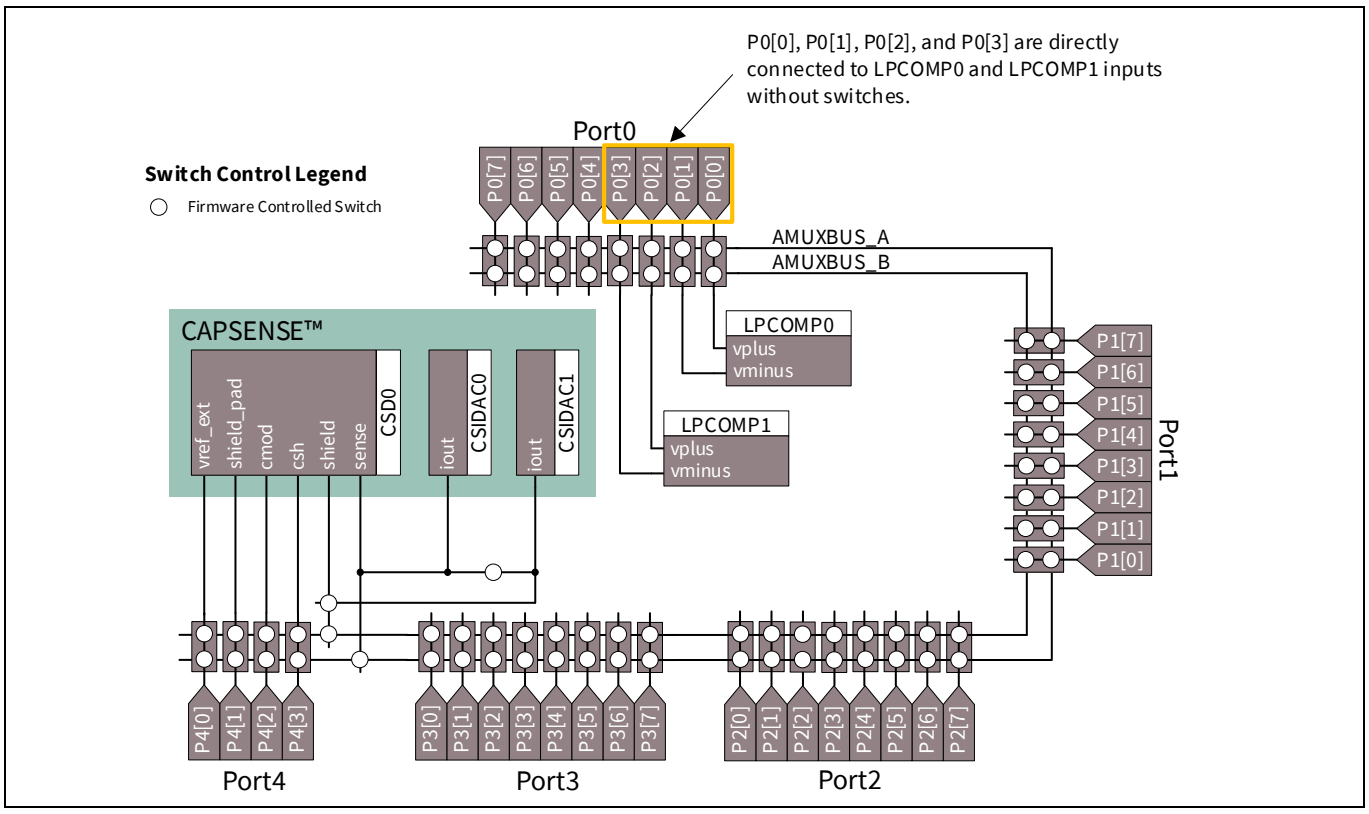


Figure 12 PSoC™ 4000S アナログ配線図

GPIO ピンの基本情報

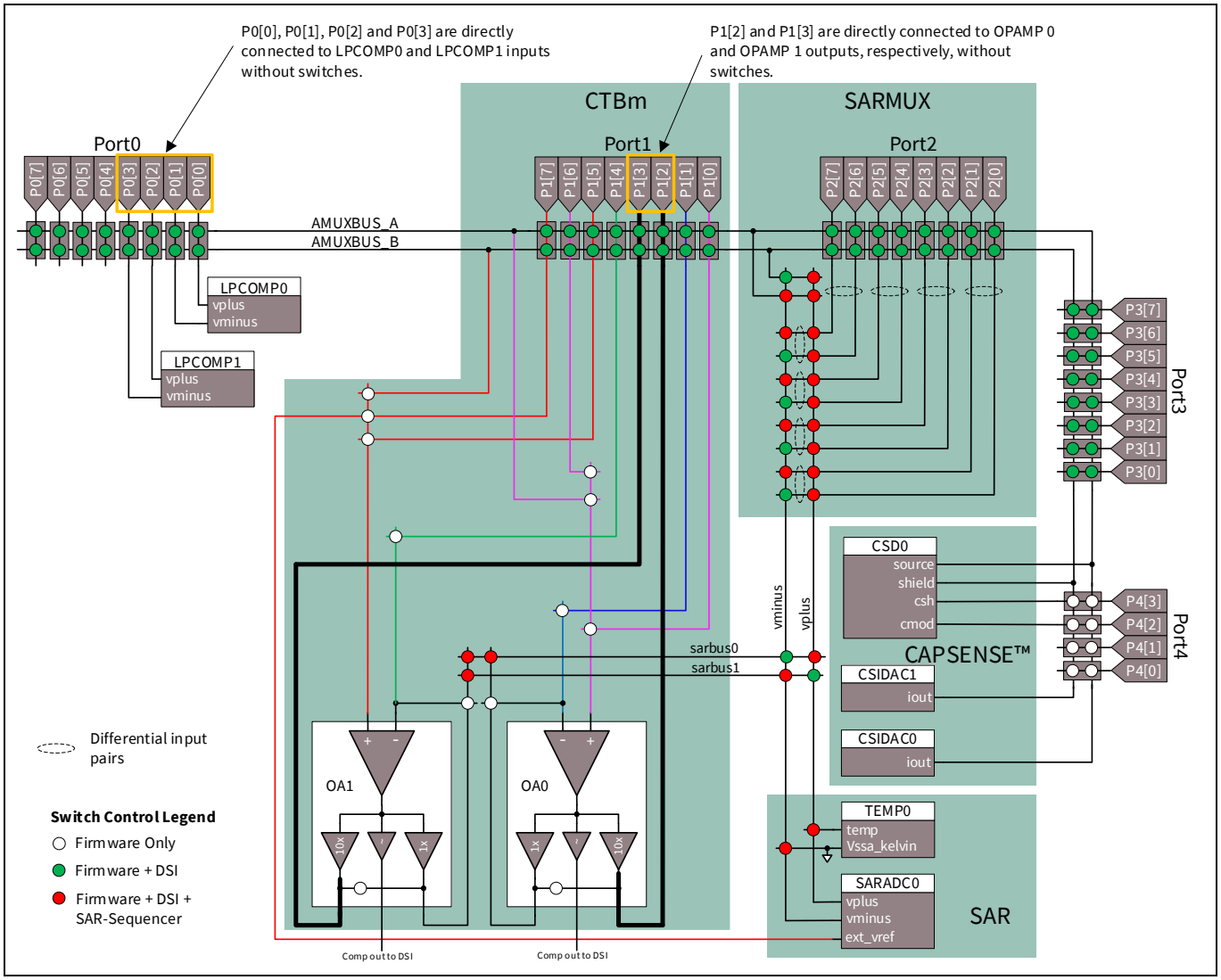


Figure 13　PSoC™ 4200/PSoC™ 4100 のアナログ配線図

GPIO ピンの基本情報

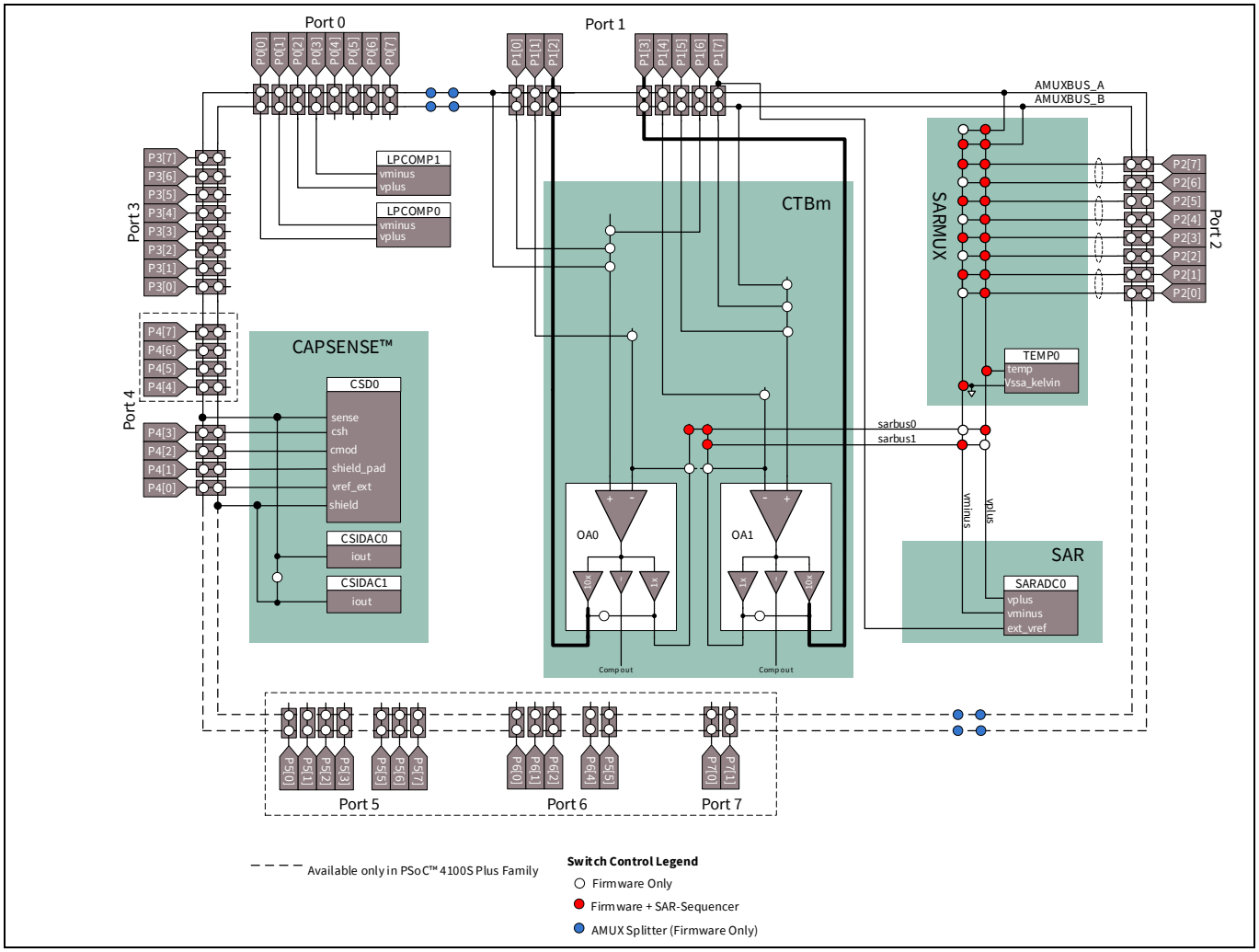


Figure 14 PSoC™ 4100S/4100S Plus アナログ配線図

GPIO ピンの基本情報

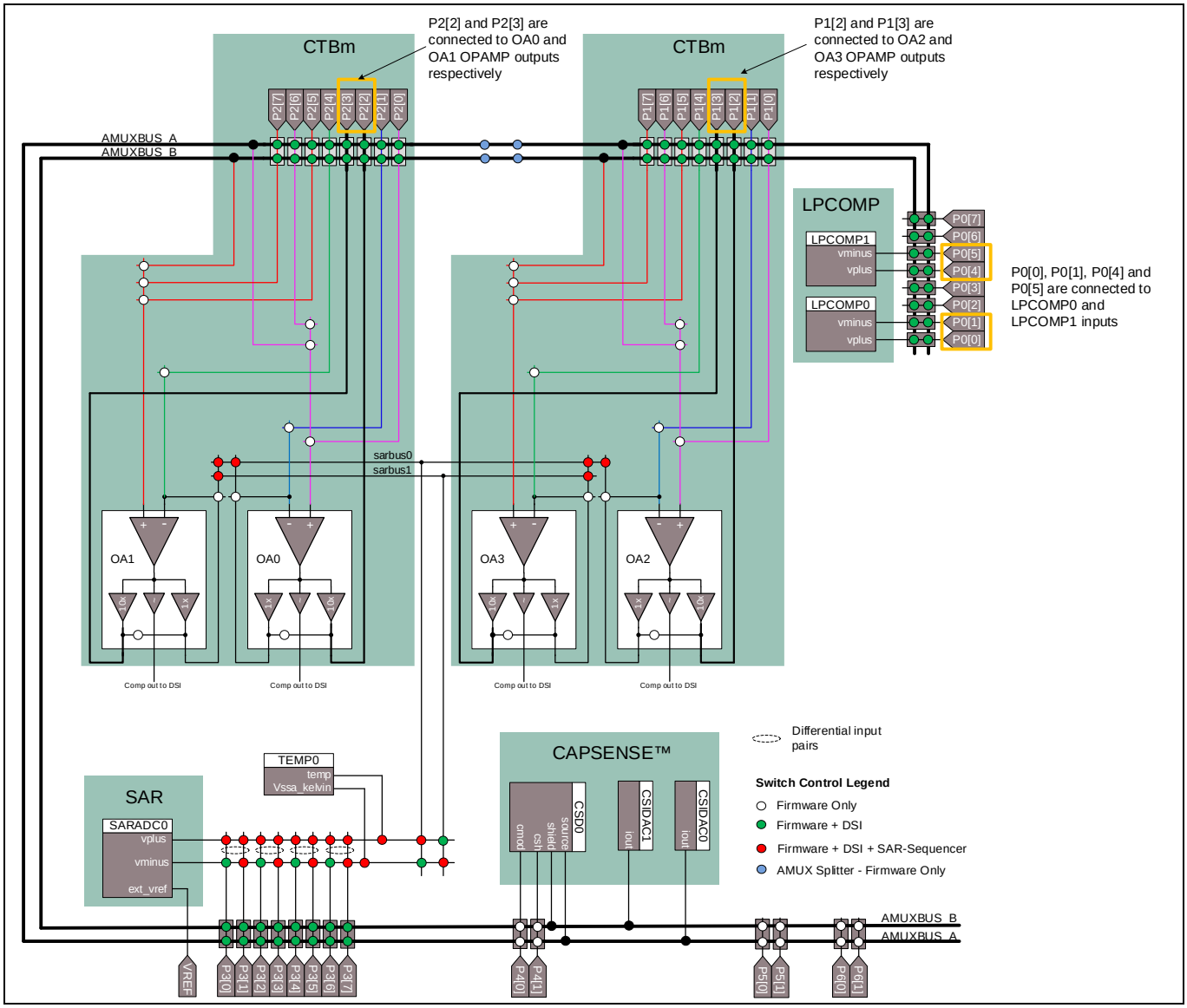


Figure 15 PSoC™ 41xx-BL/PSoC™ 42xx-BL のアナログ配線図

GPIO ピンの基本情報

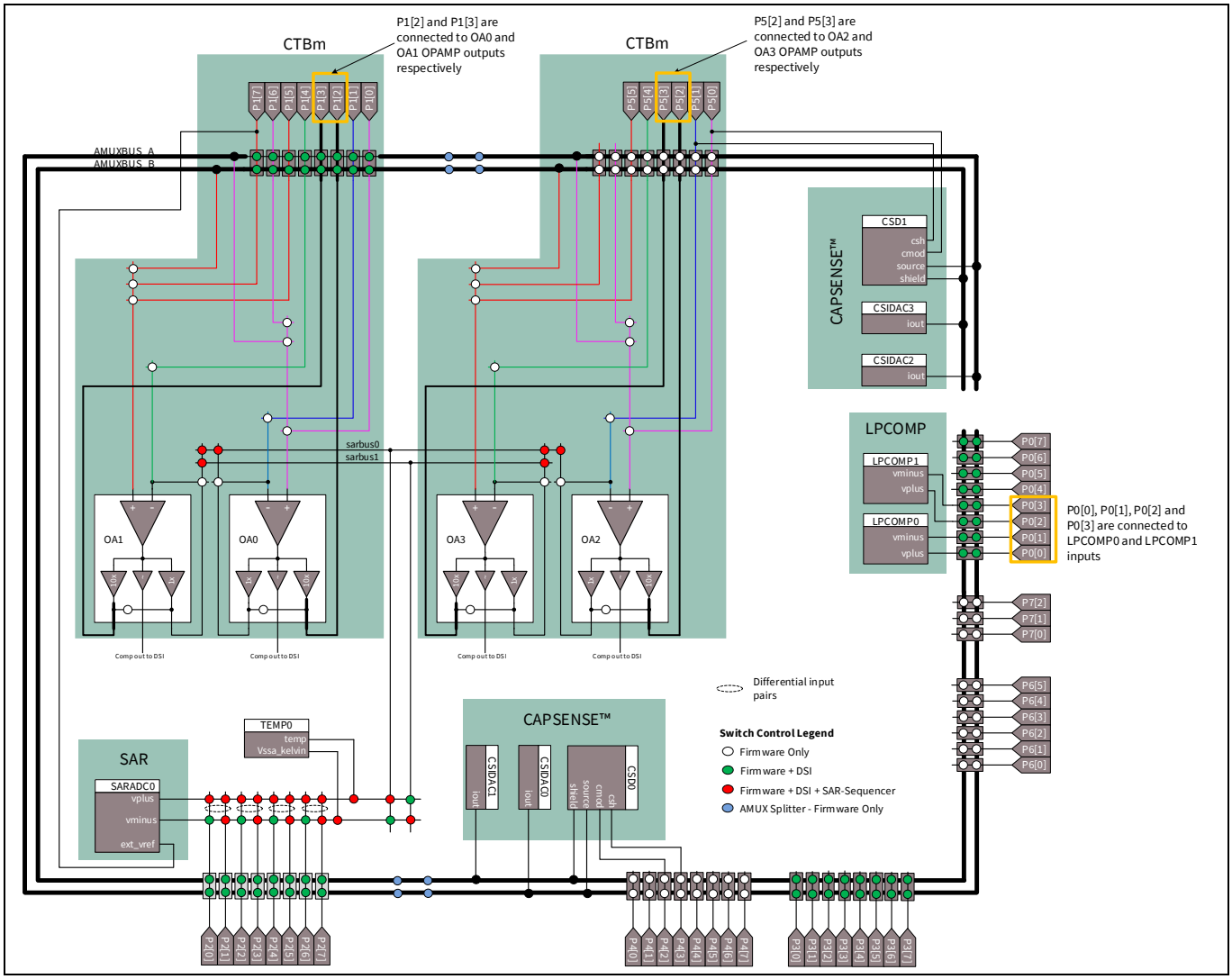


Figure 16 PSoC™ 4100M/PSoC™ 4200M のアナログ配線図

GPIO ピンの基本情報

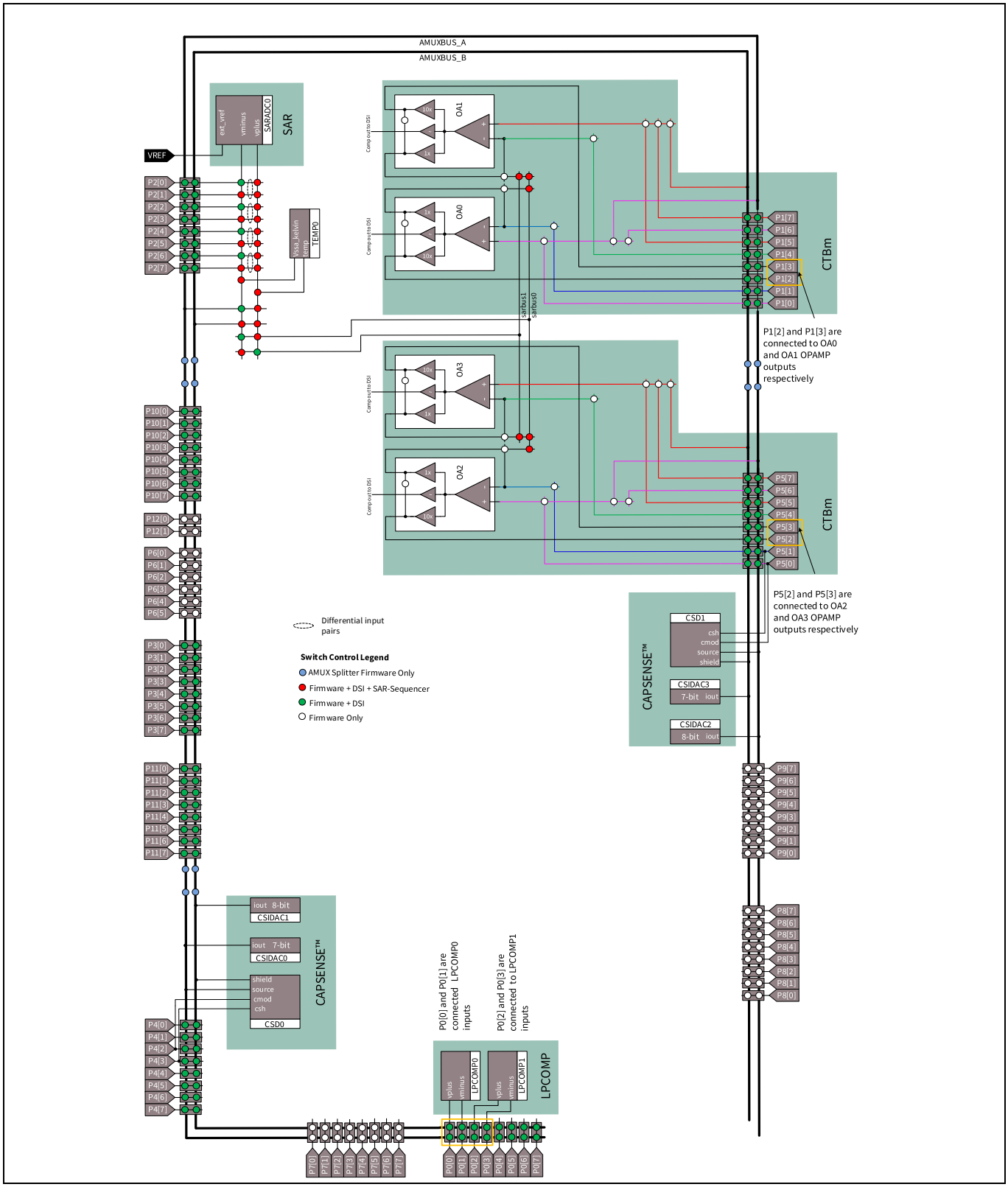


Figure 17　PSoC™ 4200L アナログ配線図

GPIO ピンの基本情報

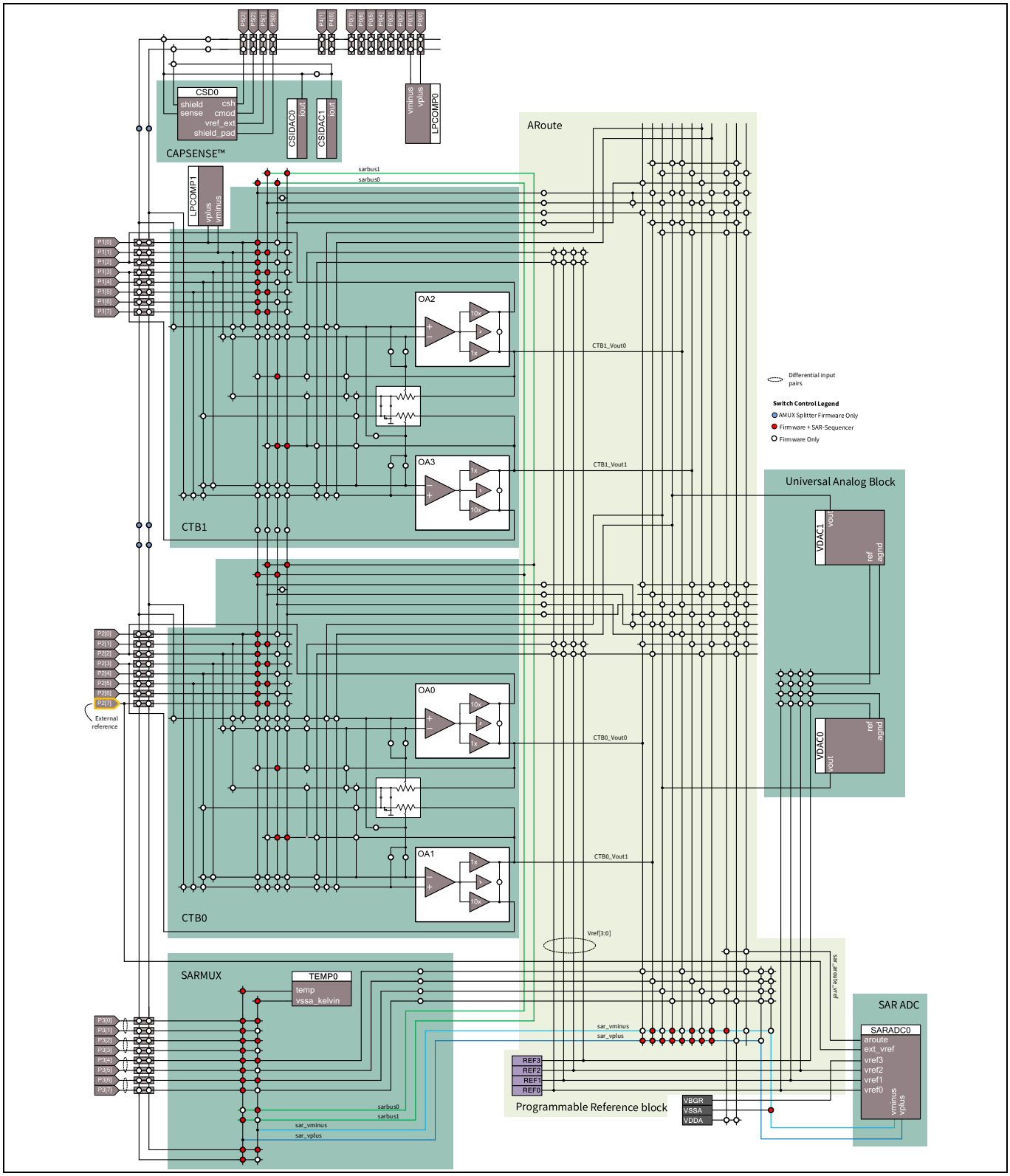


Figure 18 PSoC™ 4100PS アナログ配線図

*Note: PSoC™ Creator IDE ツールは **Figure 11** ～**Figure 18** に示されるものと同様な設計のためにアナログ配線図を提供します。**PSoC™ Creator** でのプロジェクトの.cydwr ファイルの Analog タブをご覧ください。*

3.3 スタートアップおよび低電力動作

リセット/電源投入時、すべての GPIO ピンは高インピーダンス アナログ モードで起動します。すなわち、入力バッファおよび出力ドライバーは無効です。これらの GPIO ピンはリセットが解除されるまでこのモードのままです。各 GPIO ピンの関連レジスタ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。実行中、GPIO は関連レジスタに書き込むことで設定できます。

Note: PSoC™ 4000 デバイスでは、ピン P1[6] は電源投入の間にデバイスが起動コードを実行するまで XRES として暫定的に設定されます。電源投入の間にこのピンをプルダウンすると、デバイスがリセットに保持されるため、そうしないでください。この P1[6] によるリセットは量産テスト目的専用であり、ユーザーアプリケーションでの使用は対象外です。

詳細は **PSoC™ 4000 ファミリの I/O システムの制約 - KBA91258** を参照してください。

PSoC™には、以下のように最大 4 つの消費電力モードがあります。

Table 2 PSoC™ 4 ファミリの低消費電力モード

デバイス	スリープ	ディープスリープ	ハイバネート	ストップ
PSoC™ 4000	✓	✓	×	×
PSoC™ 4000S	✓	✓	×	×
PSoC™ 4100/4200	✓	✓	✓	✓
PSoC™ 4100S	✓	✓	×	×
PSoC™ 4100S Plus	✓	✓	×	×
PSoC™ 4100S Max	✓	✓	×	×
PSoC™ 4100S Plus 256KB	✓	✓	×	×
PSoC™ 4200DS	✓	✓	×	×
PSoC™ 4500S	✓	✓	×	×
PSoC™ 4700S	✓	✓	×	×
PSoC™ analog coprocessor	✓	✓	×	×
PSoC™ 4 Bluetooth® LE	✓	✓	✓	✓
PSoC™ 4 M	✓	✓	✓	✓
PSoC™ 4 L	✓	✓	✓	✓
PSoC™ 4100PS	✓	✓	×	×

スリープモードで、GPIO はアクティブであり、ペリフェラルにより積極的に駆動されることができます。このモードで CPU だけが非アクティブです。ディープスリープモードで、I^2C, LCD ドライバー, オペアンプおよびコンパレータなどのディープスリープペリフェラルにより駆動されるピンは動作可能です。I^2C ピンはアドレス一致イベントでデバイスを起こすことができます。デバイスピンに接続されるセグメント LCD はディープスリープモードでも周期的にリフレッシュされます。

PSoC™ 4 デバイス (PSoC™ 4000 以外) は GPIO をディープスリープ, ハイバネートおよびストップモードでフリーズする追加機能があります。低消費電力モードが終了すると、GPIO は自動的にフリーズ解除されます。しかし、ディープスリープペリフェラルにより駆動される GPIO はディープスリープモードでアクティブであり、フリーズされないことに注意してください。

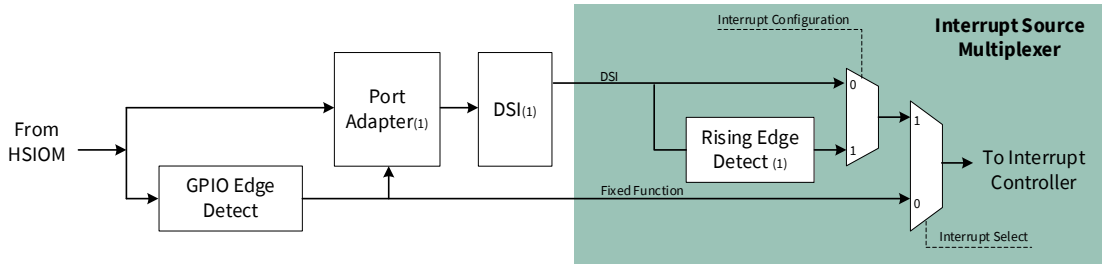
ハイバネートおよびストップモードの場合、ウェイクアップはデバイスリセットによって発生し、GPIO 設定およびピン状態をクリアします。ピン状態を保持するために、`CySysPmFreezeIo()`および `CySysPmUnfreezeIo()`API 関数を使用してください。ユーザーが `CySysPmStop()`API 関数を使用し

てストップモードを起動すると、CySysPmFreezeIo()関数が自動的に呼び出されるので、ストップモードのためにこの関数の呼び出しが不要であることに注意してください。しかし、ハイバネートモードに入る関数呼び出しの直前に、CySysPmFreezeIo()API を呼び出す必要があります。GPIO は CySysPmUnfreezeIo()の呼び出しにより、ロック解除されます。ストップモードを終了するときにも、この関数の呼び出しが必要です。フリーズされたピンの状態と設定は外部リセット (XRES) イベントの時に維持されないことに注意してください。

CySysPmFreezeIo()および CySysPmUnfreezeIo()は、ディープスリープモードでも利用可能です。**ディープスリープにおける制御レジスタの取り扱い**に、この特長の使用例を示します。制御レジスタなどの UDB ベースのコンポーネントはディープスリープ, ハイバネート, およびストップモードで非アクティブであり、データを失います。制御レジスタがピンを駆動している場合、ピンの最終状態が「1」ですと、PSoC™がこれらのモードに入る時、または終了する時にグリッチが発生する可能性があります。このグリッチを回避するためには、低消費電力モードに入る前に GPIO を停止する必要があります。

低消費電力モードの詳細は、**AN86233 – PSoC™ 4 low-power modes and power reduction techniques** を参照してください。

3.4 GPIO 割込み



(1) Not applicable to PSoC™ 4000, PSoC™ 4000S, PSoC™ 4100S, PSoC™ 4100S Plus, PSoC™ 4100PS

Figure 19 割込みコントローラへの GPIO 割込み信号の配線

プロセッサ コア内の割込みコントローラの 32 割込みラインでは、それぞれ 1 つの割込みソース マルチプレクサがあります。このマルチプレクサ ブロックは割込みソースを選択し、立ち上りエッジ検出か割込みコントローラへの直接接続のオプションを提供します。2 つの割込みソースがあります。

1. 固定機能ソース
2. DSI ソース

割込み選択ラインは DSI ソースまたは固定機能ソースを選択します。割込み設定は割込みコントローラに接続するために、直接接続または立ち上りエッジ検出のロジック ルートを選択します。

固定機能の割込みソースは固定割込みベクタがあります。すなわち、割込みソースは Cortex® M0/Cortex® M0+ CPU の 32 割込みラインの 1 つへの専用接続があります。このルート上の割込みソースは割込みコントローラに直接接続します。割込みソースが DSI を通して配線される時には、ベクタ選択は固定されません。この配線は直接接続に加えて、立ち上りエッジ検出のオプションも提供します。

*Note: 割込みベクタ表は**テクニカルリファレンスマニュアル**の「割込み」章を参照してください。*

割込みソース マルチプレクサの使用は GPIO 割込みに限定されません。他のすべてのソースにも使用されます。他の割込みソースについては **AN90799 - PSoC™ 4 Interrupts** の「割込みソース」を参照してください。

GPIO ピンの基本情報

Figure 19 に示すように、GPIO 割込みは割込みソース マルチプレクサに存在しているリソースに加えて、自身の GPIO エッジ検出ブロックも使用します。

HSIOM からの GPIO 割込み信号は以下の方法で配線されます。

- 経路 1: GPIO エッジ検出ブロックを通じる固定機能配線 (割込みソース マルチプレクサが直接接続に設定)。
- 経路 2: GPIO エッジ検出ブロックを通じる DSI 配線 (割込みソース マルチプレクサが立ち上りエッジ検出に設定)。
- 経路 3: GPIO エッジ検出ブロックを通じる DSI 配線 (割込みソース マルチプレクサが直接接続に設定)。
- 経路 4: GPIO エッジ検出ブロックをバイパスする DSI 配線 (割込みソース マルチプレクサが立ち上りエッジ検出に設定)。
- 経路 5: GPIO エッジ検出ブロックをバイパスする DSI 配線 (割込みソース マルチプレクサが直接接続に設定)。

異なるルートを設定する方法については、**ピンコンポーネント割込み**を参照してください。

以下の図に GPIO エッジ検出ブロックを示します。このブロックは着信 GPIO 信号の立ち上りエッジ, 立ち下りエッジ, および両方のエッジを検出します。1 つのポート内の個々の GPIO 割込み信号は OR され、単一の割込み要求を生成します。したがって、各ポートには 1 つの割込みベクタがあります。

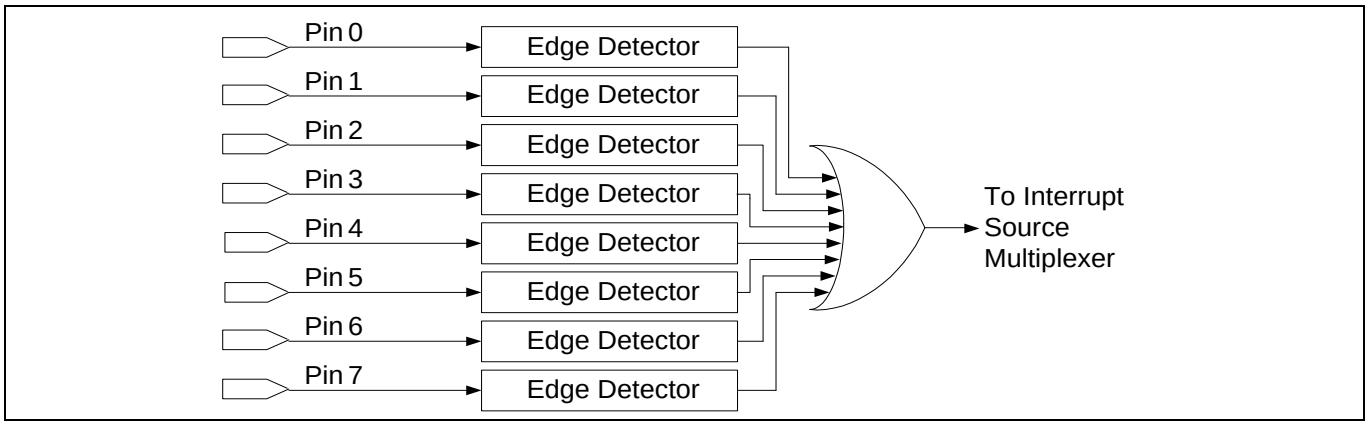


Figure 20 GPIO エッジ検出

Figure 20 に示すように、割込みがトリガーされると、割込みソースを識別する必要があります。PSoC™ 4 は割込みピンを識別するために、ステータスレジスタを提供します。GPIO エッジ検出ロジックを使用する度に、下記の状況を避けるために、ステータスレジスタは読み出した後、クリアすることが重要です。

1. 割込みソース マルチプレクサが立ち上りエッジに設定される場合のシングル割込みトリガーおよび次の割込みに対する非反応性。経路 2 が使用される場合、このシナリオが発生します。
2. 割込みソース マルチプレクサが直接接続に設定される場合の、単一の要求に対する繰り返しの割込み。経路 1 または経路 3 が使用される場合、このシナリオが発生します。
3. GPIO 割込みが GPIO エッジ検出ブロックを通じない場合、割込みをクリアすることが不要です。しかし、割込みソース マルチプレクサの立ち上りエッジ検出ロジックもバイパスされると、レベルタイプの割込み (経路 5) が発生します。この場合は、ピン信号が HIGH になるかぎり、割込みは繰り返しトリガーされます。そのため、GPIO エッジ検出ブロックがバイパスされる場合 (経路 4)、割込みソース マルチプレクサを立ち上りエッジ割込みに設定することが推奨されます。

Note: *任意のピンがウェイクアップ ソースとして使用できるように、GPIO 割込み論理はスリープモード, ディープスリープモード, およびハイバネートモードで機能し続けます。専用のウェイクアップピン(P0[7]) は PSoC™ 4200/PSoC™ 4100, PSoC™ 4M, および PSoC™ 4L 製品のストップモードでデバイスをウェイクアップするために使用可能です。PSoC™ 4 Bluetooth® LE デバイスの場合は、ウェイクアップピンは P2[2] です。*

3.4.1 GPIO 割込みの制限

- ポート 4 以上はポートアダプタがありません。したがって、DSI ルーティングを介するピン割込みはこれらのポートピンに対して不可能です。
- PSoC™ 4000 および PSoC™ 4100/PSoC™ 4200 は各ポートに 1 つの割込みベクタがあります。PSoC™ 4 Bluetooth® LE はポート 5 以上、PSoC™ 4M はポート 4 以上のポートに専用割込みベクタを持っていませんが、共有ポート割込みベクタがあります。この割込みベクタは任意のポート割込みがアクティブになると、トリガーされます。この共有ポート割込みの使い方については、**Pins Component datasheet** を参照してください。
- 共通割込みベクタを持つポートについては、**architecture テクニカルリファレンスマニュアル** (TRM) の「Interrupts」章を参照してください。

サンプルプロジェクトは**ピン割込み**に示され、GPIO 割込みの使い方を説明します。割込みを一般的に理解するためには、アプリケーションノート **AN90799 – PSoC™ 4 Interrupts** を参照してください。

4 過電圧耐性 (OVT) ピン

PSoC™ 4 Bluetooth® LE でのピン P5[0]と P5[1]、および PSoC™ 4M でのポート 6 は OVT ピンです。PSoC™ 4L では、ポート 6 およびポート 8 が OVT ピンです。それらは普通の GPIO ピンと同様であり、以下の特長が追加されています。

1. 過電圧耐性 - OVT ピンと電源の間には ESD クランプ ダイオードがありません。これにより OVT ピンには VDDIO、VDDD、または VDDA 電圧よりも高い最大 5.5 V の外部電圧への電圧耐性があります。
2. 普通の GPIO と比べると、より良いプルダウン駆動能力を提供します。
3. シリアル通信ブロック (SCB) : I^2C として設定し、自体のラインを OVT ピンにルートされる場合、SCB は下記の I^2C 仕様を満たします。
   a) 高速モード プラス LOW レベル出力電流 (IOL) 仕様
   b) 高速モードおよび高速モード プラス ヒステリシスおよび最小立ち下り時間の仕様

I/O ハードウェアの詳細はテクニカルリファレンスマニュアルの「I/O System」章を参照してください。

5 PSoC™ Creator での GPIO ピン

ここでは、PSoC™ Creator を使用して GPIO ピンを設定および使用する方法について説明します。

5.1 ピン コンポーネント シンボル

ピン 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される方法です。これにより、PSoC™ Creator は選択済みのピン設定に基づいて PSoC™デバイス内に自動的に信号を配置し、配線できます。

標準的インフィニオン コンポーネント カタログでは、ポートおよびピンの 4 個のシンボルタイプ (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 4 個の定義済みの GPIO 設定が含まれます。以下の図のように、これらのコンポーネントの 1 つを回路図にドラッグして、ピンをプロジェクトに加えてください。

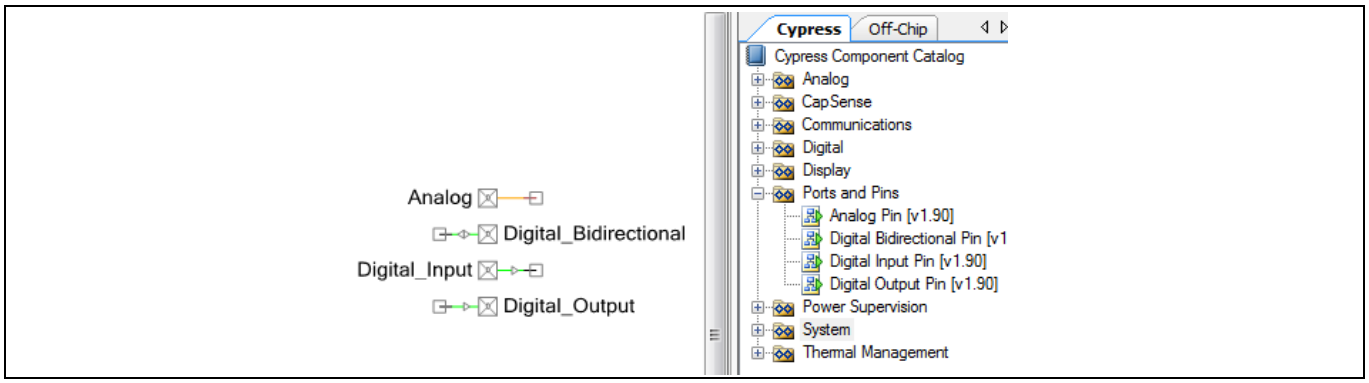


Figure 21 PSoC™ Creator でのピンコンポーネント シンボル タイプ

5.2 ピンコンポーネントのカスタマイザ

PSoC™ Creator では、各コンポーネントはそのカスタマイザにより設定されます。**Figure 22** に、コンポーネントをダブルクリックしてアクセスできるピンコンポーネント カスタマイザを示します。

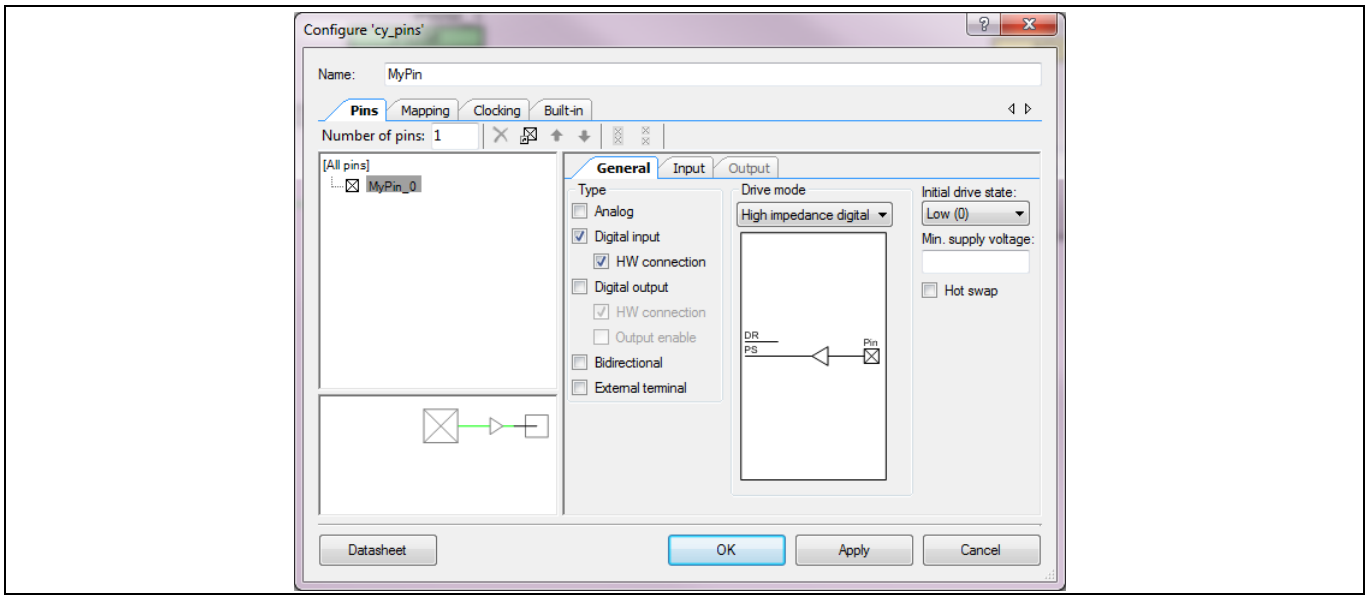


Figure 22 ピンコンポーネント カスタマイザ

PSoC™ Creator での GPIO ピン

以下の表にピンコンポーネント カスタマイザのいくつかのパラメーターを示します。すべてのパラメーターの詳細は**ピン コンポーネント データシート**を参照してください。

Table 3　　ピン コンポーネントの設定

設定	説明
General タブ > Type	このパラメーターはピンタイプを設定します。利用可能なオプションは以下のとおりです。 • アナログ • ハードウェア (HW) 接続有り, または無しのデジタル入力 • HW 接続と出力イネーブル有り, または無しのデジタル出力 • 双方向ピン デジタル入力または出力がハードウェア接続なしで設定される場合、ピン状態は CPU により制御されることを意味します。複数の選択を一度に実行できることに注意してください。例えば、ピンは同時にアナログとデジタル入力の両方に設定できます。
General タブ > Drive Mode	このパラメーターは、**GPIO ピンの基本情報**で説明されている 8 つの駆動モードの 1 つにピンを設定します。**Figure 23** にピン カスタマイザの駆動モードのオプションを示します。
General タブ > Initial Drive State	Initial drive state パラメーターはデータレジスタの値を設定します。ピンがソフトウェア駆動であり、適切なモードに設定されている場合、この値はピンに反映されます。ピンが出力モードにある場合 (HW 接続が有効, および出力イネーブルが無効)、Initial drive state はイネーブル制御として機能します。**Figure 23** に示すように、初期状態を「1」に設定すると、PSoC™ Creator によりデフォルト値として実行されるピンが有効になります。ピンが入力として設定される場合でも、Initial drive state は有用です。例えば、抵抗プルアップが入力ピンに必要な場合、抵抗を通じるプルアップ パスを有効にするために、駆動モードを Initial drive state = HIGH で Resistive pull up に設定します。同様に、Resistive pull down の場合、Initial drive state はプルダウン パスを有効にするために、LOW に設定する必要があります。
Input タブ > Threshold	CMOS および LVTTL 入力閾値の設定はポート全体で適用可能です。**Figure 24** に示すように、3 つのオプションがあります。「CMOS or LVTTL」オプションにより、ポート内の他のピンの閾値に基づいて、PSoC™ Creator ツールは CMOS または LVTTL を選択できます。
Input タブ > Interrupt	**GPIO 割込み**で説明するように、このパラメーターは GPIO エッジ検出ブロックを設定します。詳細については**ピンコンポーネント割込み**を参照してください。

PSoC™ Creator での GPIO ピン

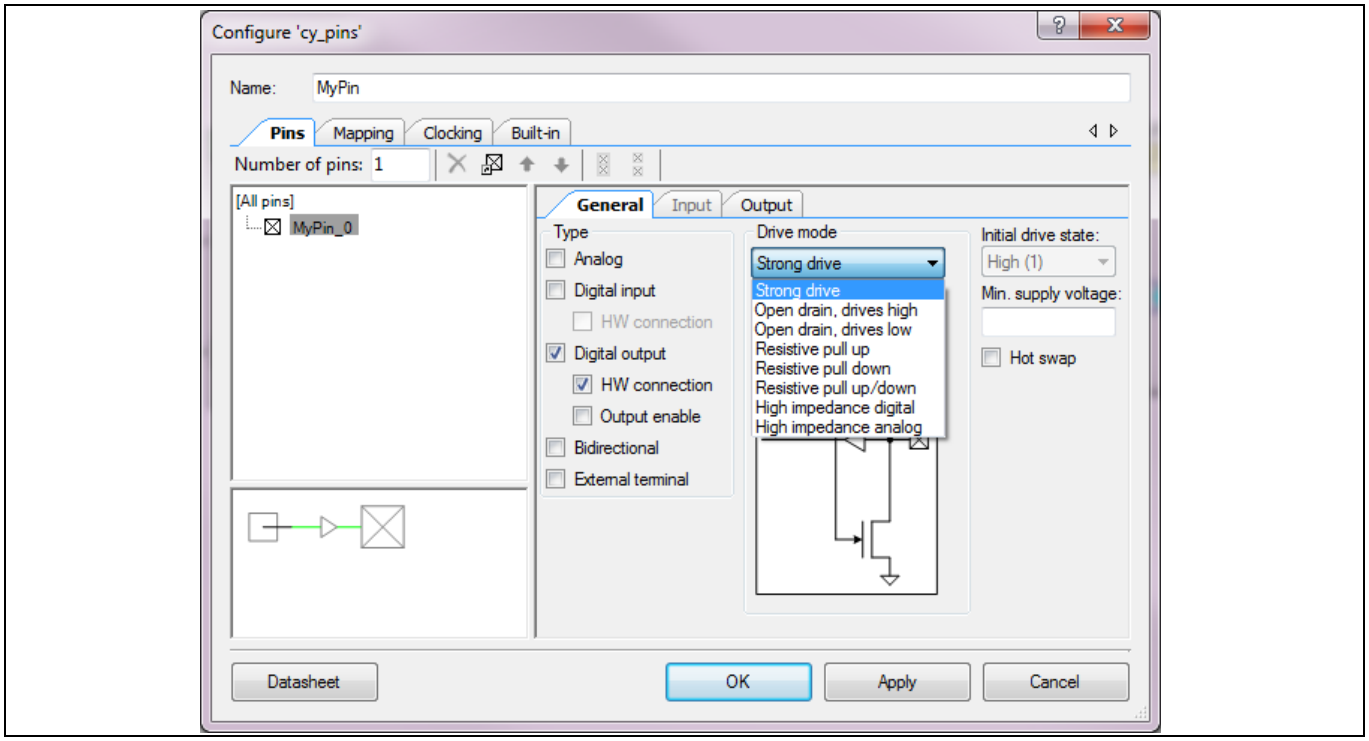


Figure 23　ピン駆動モードの設定および初期駆動状態

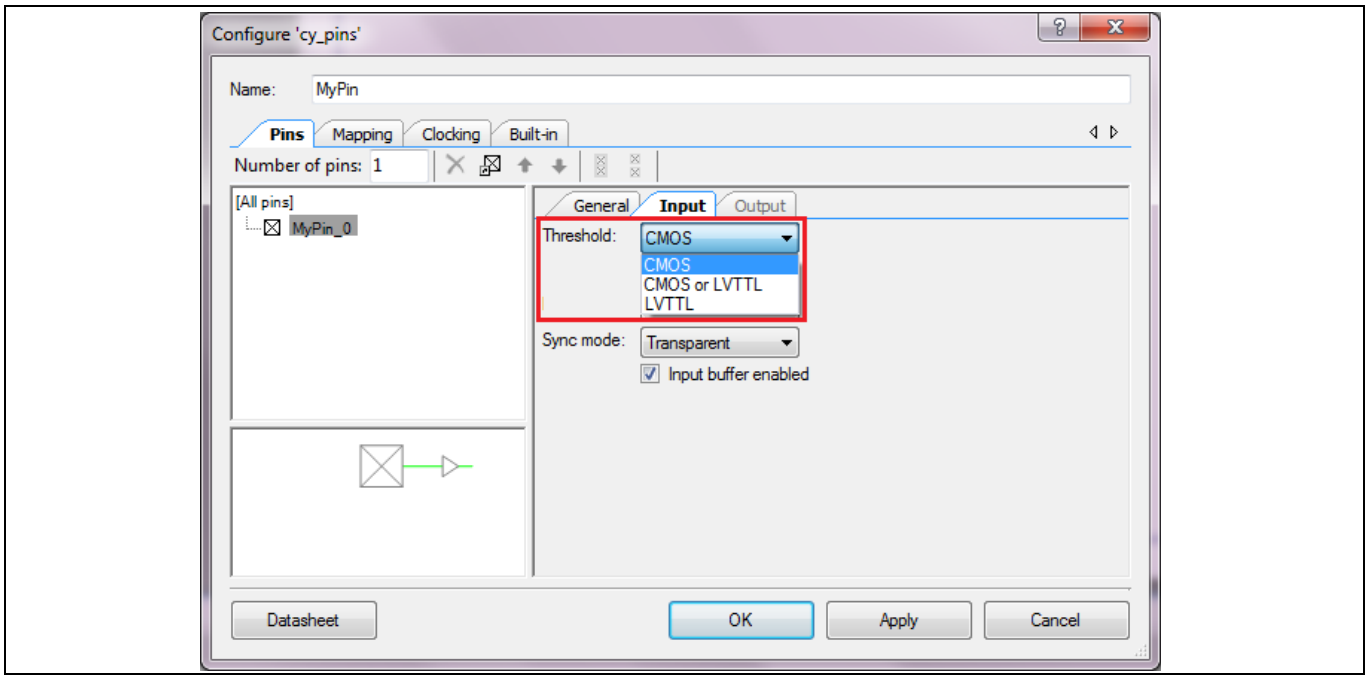


Figure 24　ピン入力閾値の選択

5.3 ピンコンポーネント割込み

GPIO 割込みで説明するように、ピン カスタマイザの Interrupt パラメーターは GPIO エッジ検出ブロックを設定します。

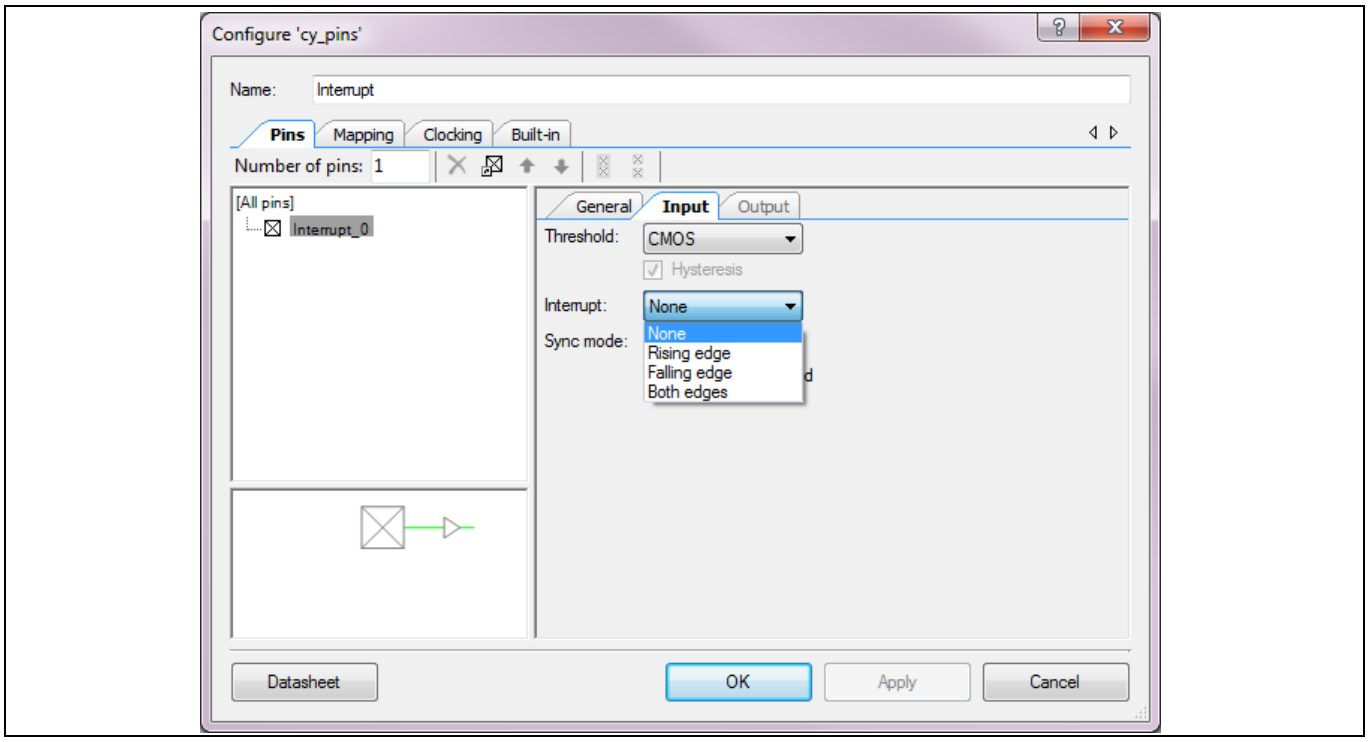


Figure 25 PSoC™ Creator の割込み設定

Figure 26 に示すように、割り込みが有効になると、ピン コンポーネントのシンボルが変化します。

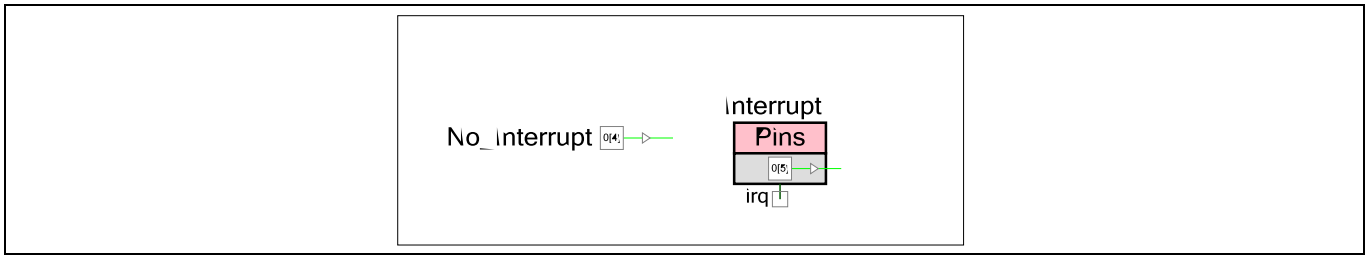


Figure 26 割込みが有効時のピン コンポーネント シンボル

割込みが有効になる場合、各物理 GPIO ポートに 1 つだけのピンコンポーネントを使用できることに注意してください。GPIO 割込みで説明するように、この制限の理由としては、1 個のポートですべてのピン割込みは OR されます。そのため、ポートごとに 1 つのみの IRQ 信号を回路図に示します。例えば、割込みが有効化された 2 つのピンコンポーネントを考えてください。これらのコンポーネントは同じ物理ポートでのピンにマップできません。

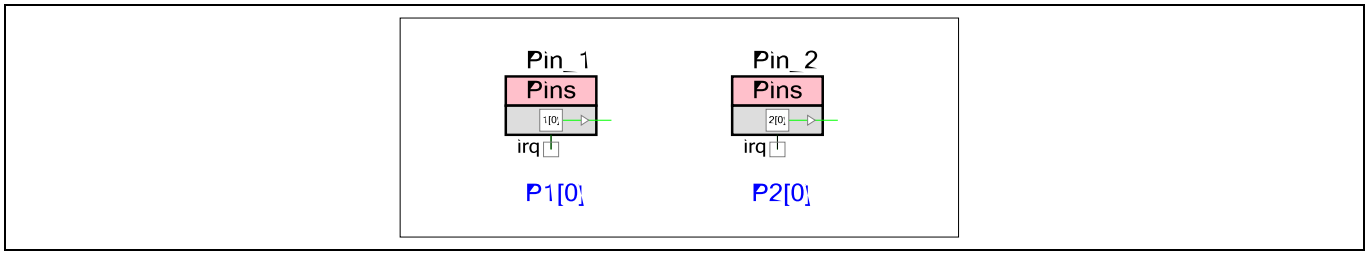


Figure 27　割込みが有効時の 2 つのピン コンポーネント

PSoC™ Creator は 2 つのコンポーネントを同じポートに割り当てることを許可しません。許容方法は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。これは、回路図上でその物理ポートに対して 1 つのみの IRQ 信号があることを保証します。各ピンにそれ自身の割込みエッジタイプを割り当てることもできます。唯一の制限はピンが同じポートで隣接しなければならないことです。割込みソースは ISR で識別する必要があります。**ピン割込み**を参照してください。

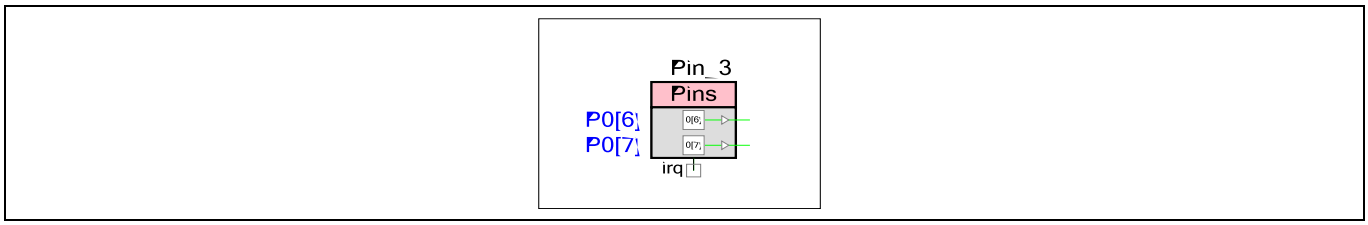


Figure 28　有効な割込みがある同じポートの複数ピン

ピンコンポーネントの IRQ は割込みコンポーネントに接続する必要があります。これは GPIO 割込み信号を割込みコントローラに配線します。

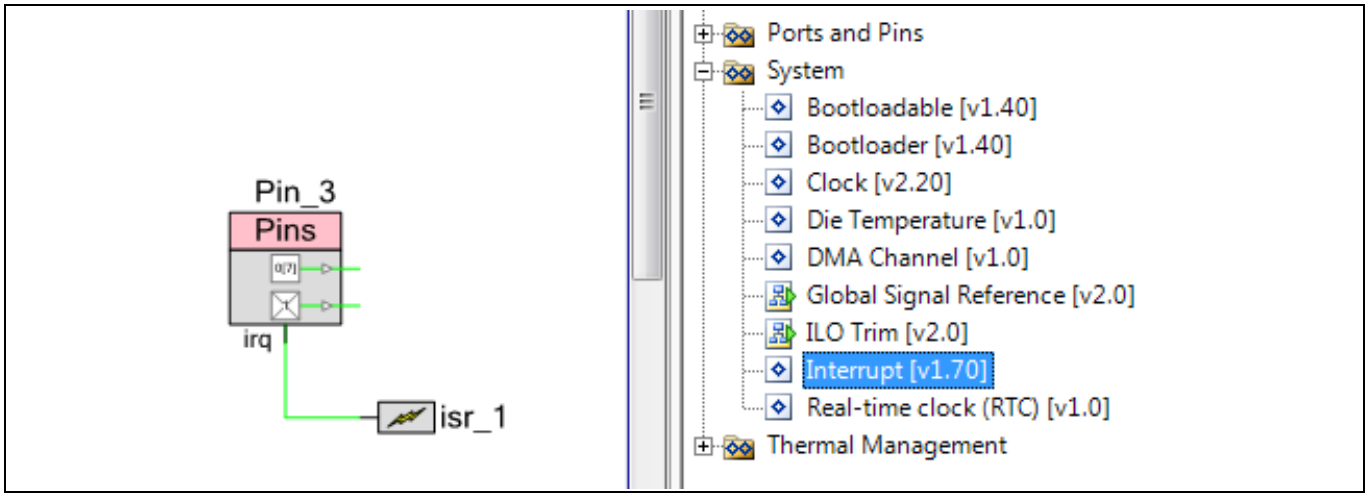
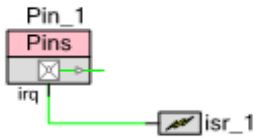
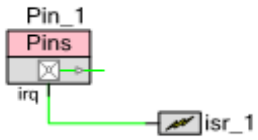
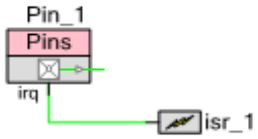


Figure 29　カタログにある割込みコンポーネント

割込みコンポーネントは割込みソース マルチプレクサを直接接続 (割込みコンポーネント カスタマイザで「レベル」として表示) または立ち上りエッジに設定します。GPIO 割込みアーキテクチャは、割込み信号に使用可能な異なる配線とともに **GPIO 割込み**で説明されます。異なる配線は、以下にまとめるピンコンポーネントおよび割込みコンポーネント カスタマイザの設定により設定されます。

Table 4　　GPIO 割込みの設定

回路図	ピンコンポーネントでの割込み設定	割込みコンポーネントの設定	配線	詳細
	立ち上りエッジまたは立ち下りエッジまたは両方のエッジ	レベル	経路 1	エッジの割込みはピンコンポーネントの設定に依存します。選択されたポートによって固定割込みベクタを使用します。この設定では、GPIO 割込みステータスレジスタをクリアする必要があります。そうしないと、割込みは 1 つの割込み要求で繰り返しトリガーされます。この設定は、どんな低消費電力モードからのデバイス起動に使用可能。ただし、ストップモードから復帰するためには、選択したデバイスに応じた特定のピンの使用が必要です。
	立ち上りエッジまたは立ち下りまたは両方のエッジ	立ち上りエッジ	経路 2	エッジの割込みはピンコンポーネントの設定に依存します。この設定では、GPIO 割込みステータスレジスタがクリアされる必要があります。そうしないと、割込みが 1 回のみトリガーされます。割込みベクタが固定されません。この設定は、スリープモードからのみ CPU を起動可能です。低消費電力モードでは機能しません。
	立ち上りエッジまたは立ち下りまたは両方のエッジ	レベル	経路 3	これは経路 1 と同様ですが、経路 3 は割込みベクタが希望の DSI ベクタラインに強制される場合のみに実行されます。割込みベクタの強制方法はアプリケーションノート **AN90799 – PSoC™ 4 Interrupt** を参照してください。この設定は、スリープモードからのみ CPU を起動可能です。低消費電力モードでは機能しません。
	無効	立ち上りエッジ	経路 4	この設定は立ち上りエッジ割込みを提供します。この場合は、割込みをクリアする必要がありません。この設定は、スリープモードからのみ CPU を起動可能です。低消費電力モードでは機能しません。
	無効	レベル	経路 5	この設定はレベル割込みを提供します。ピン信号が HIGH になる限り、割込みが繰り返してトリガーされることに注意してください。この場合でも、割込みをクリアする必要がありません。この設定は、スリープモードからのみ CPU を起動可能です。低消費電力モードでは機能しません。

5.4 手動のピン割り当て

Design-Wide Resouce (DWR) ウィンドウの **Pins** タブを介してピンコンポーネントを物理ピンに割り当てることができます。ユーザーが全くピンの選択をしない場合、PSoC™ Creator は自動的にピンの割り当てをしますが、その場合、PCB でピン配置がより難しくなるという結果になります。

以下の図に 3 つの割り当てられたピンを示します。濃青色にハイライトされたピンは手動で、そして薄青色にハイライトされたピンは自動で割り当てられたものです。**Lock** オプションを選択することで、ピンが PSoC™ Creator により再度割り当てられることを回避できます。

必要な時、PSoC™ Creator を使ってピンをより簡単に割り当てできますが、ピンの割り当てを考慮してボードを設計する必要があります。

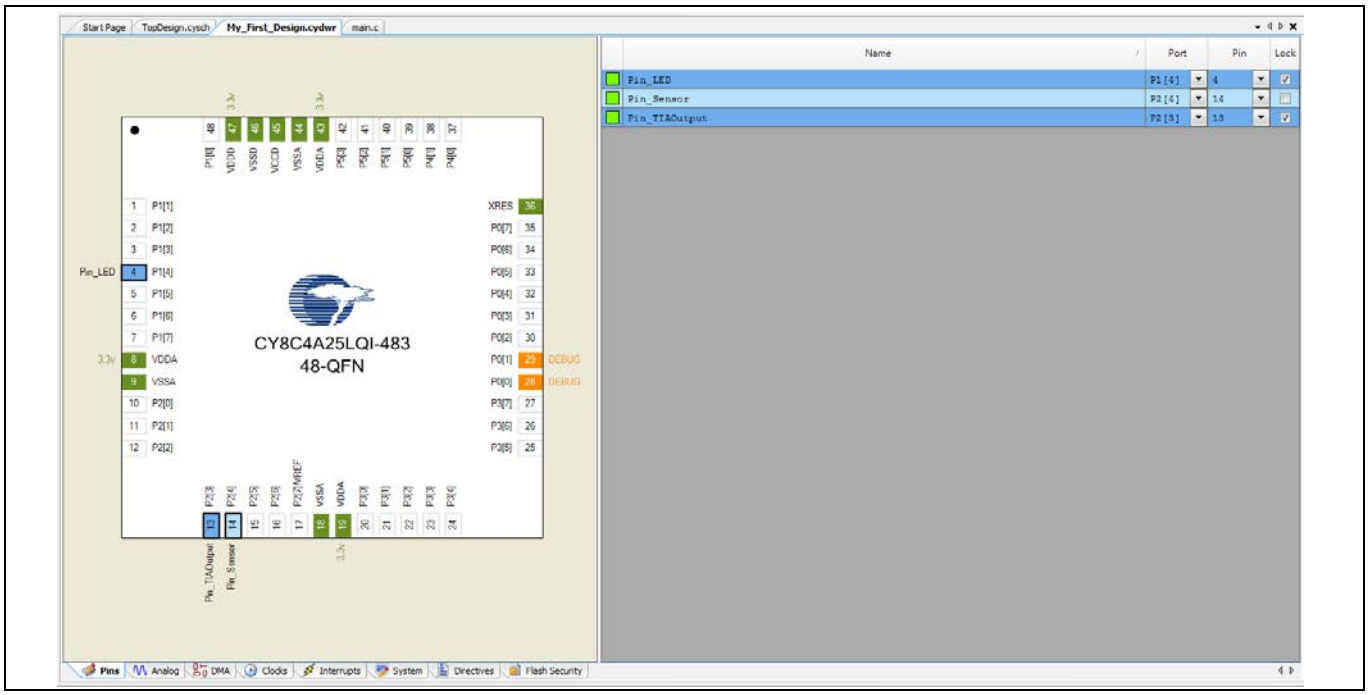


Figure 30 DWR ウィンドウでのピン割り当て

*Note: PSoC™ Creator はアナログ信号を配線するために未使用ピンスイッチを使用できます。これは.cydwr ファイルの **System** タブの **Unused Bonded I/O** パラメーターを使って設定されます。詳細については PSoC™ Creator ヘルプを参照してください。*

5.5 PSoC™ Creator の API

インフィニオンはファームウェアを通じて動的に GPIO を制御するための API 関数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。詳細については、ピン データシートで「API」節を参照してください。

cypins.h ファイルの cy_boot の一部として提供されるピン単位 API 関数は、PSoC™ Creator システム リファレンスガイドの「Pin」節 (**Help** > **Documentation** > **System Reference**) で記述されます。これらの関数を使って、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。

5.6 GPIO ピンのデバッグロジック

PSoC™ 4 シリアル ワイヤ デバッグ (SWD) ピンはポートピンで共有されます。デバッグ ポート ピンの詳細は関連するデバイス データシートを参照してください。デバッグ機能は無効にでき、ピンは DWR ウ

ィンドウの **System** タブで「Debug Select」オプションを「GPIO」に設定することにより普通の GPIO として使用できます。

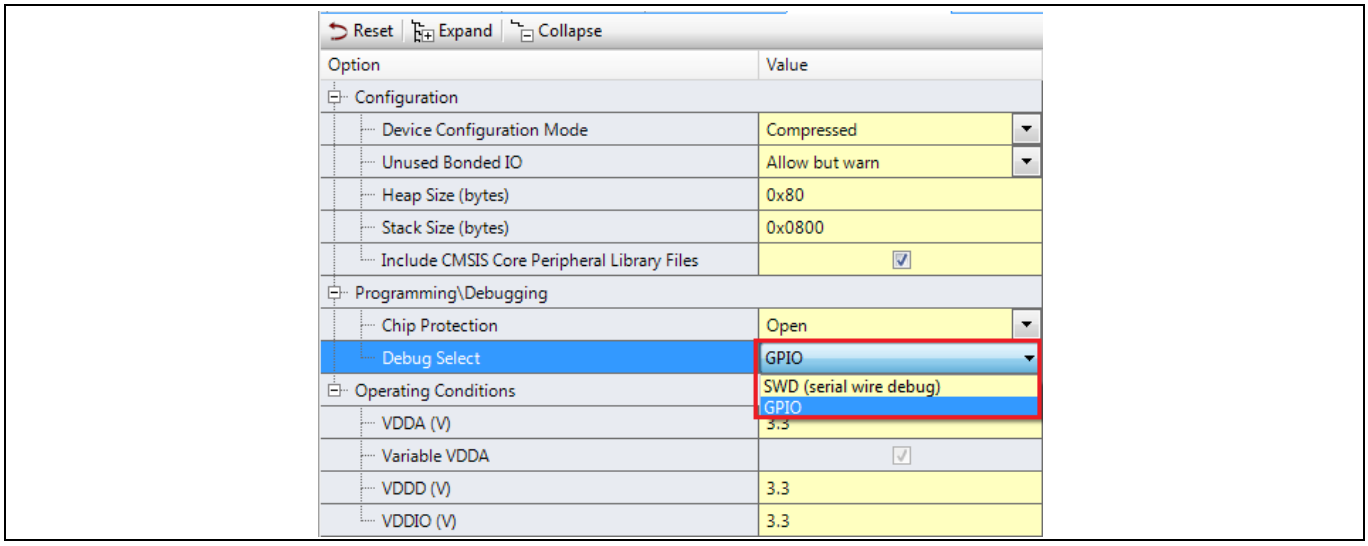


Figure 31　デバッグ ポートが無効

デバッグ インターフェースを無効にすることは、デバイスのプログラム可能性に影響を及ぼさないことに注意してください。

5.7　　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 36 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。論理ポートは、ポートの定義された名前によってコード内で参照できます。すべてのピンは 1 つの同じ物理ポートに属するか、もしくはいくつかの個別の物理ポートに属することもあります。ピン コンポーネント カスタマイザでは、ポートに必要な **Number of Pins** を設定します。**Figure 32** に示すように、ピンはフィールドの下にあるリストに表示されます。各ピンは別々に設定できます。**[All Pins]**を選択して、コンポーネントのすべてのピンを一括設定します。

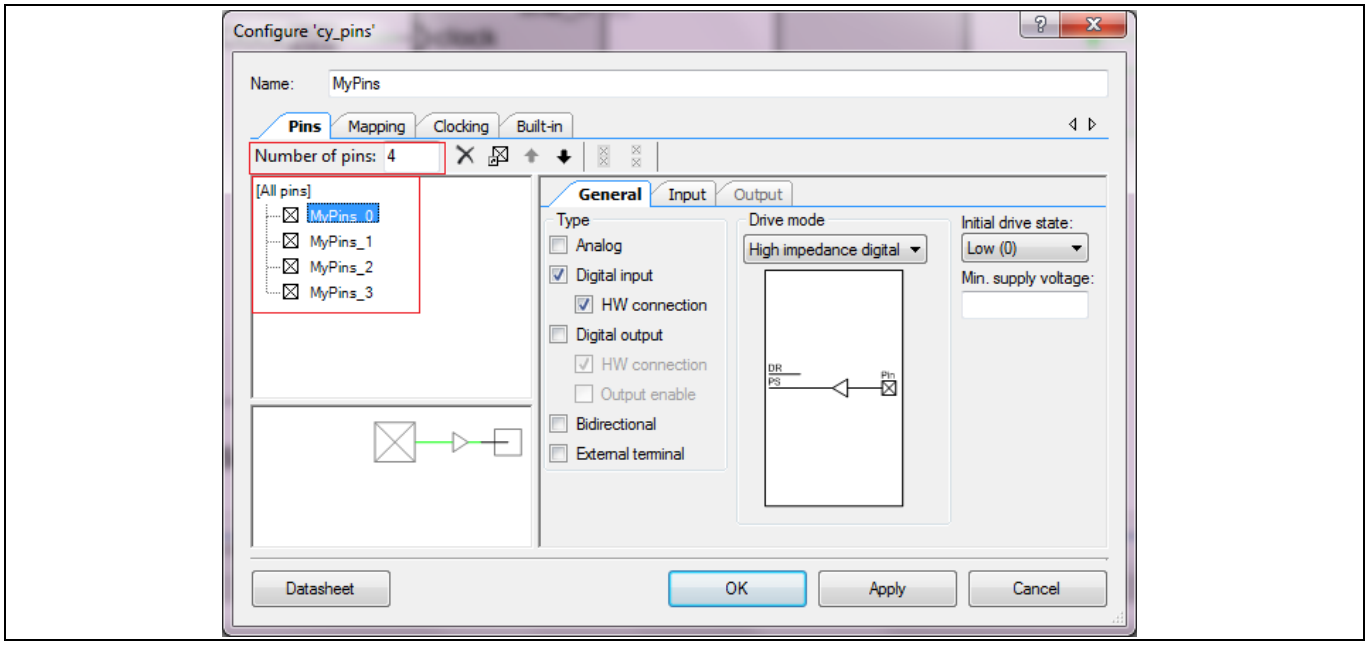


Figure 32　4 ピンの内 1 つをデジタル入力として設定

PSoC™ Creator での GPIO ピン

ピン数を「4」(3 つのデジタル入力および 1 つのデジタル出力) に設定した場合、回路図シンボルは **Figure 33** のように表示されます。

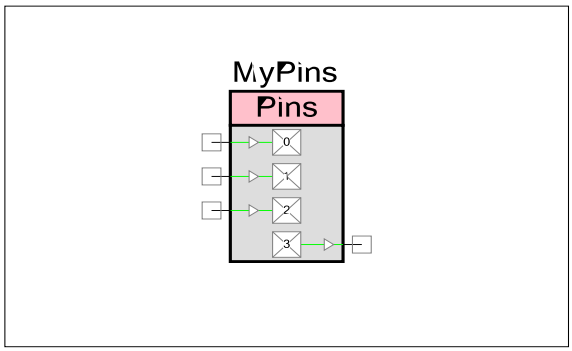


Figure 33　ポート設定のピンコンポーネント

個々のピン端子の代わりに、ポートをバスとして表示するオプションがあります。ポート設定ウィンドウの **Mapping** タブの **Display as Bus** を選択し、ポートをバスとして表示します。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いけないことに注意してください。

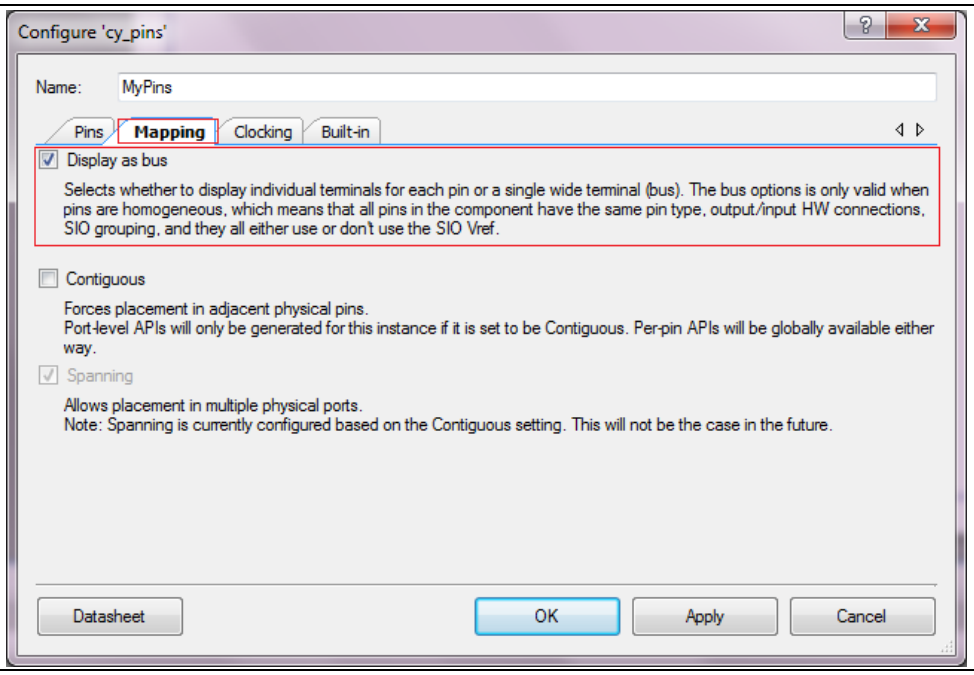


Figure 34　バスとして表示

Number of Pins が 4 つのデジタル出力に設定されると、回路図シンボルは以下のように表示されます。

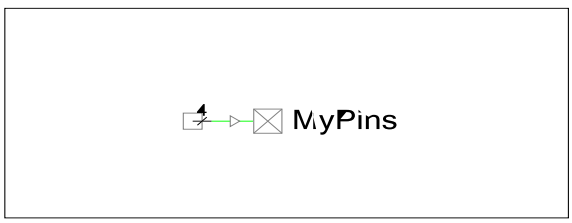


Figure 35　4 つのピンはバス付きで表示

バス付きで表示されるピンは **Mapping** タブの **Contiguous** を有効にすることにより隣接ピンに強制的にマップできます。

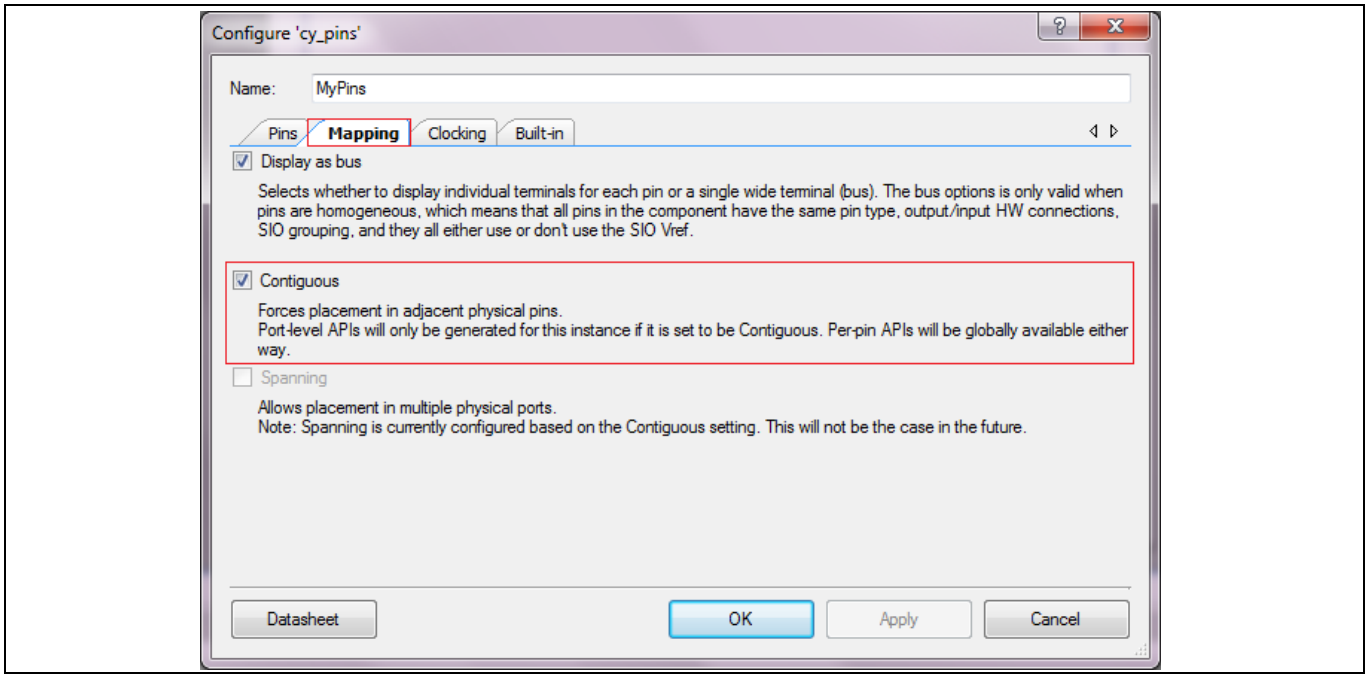


Figure 36　隣接したピン配置オプション

Contiguous を選択した場合、PSoC™ Creator はピンの使用可能なオプションのリストをポートのコンフィギュレーションとあわせて調整します。Contiguous オプションを無効にすると、任意のピンを選択できます。Contiguous オプションが有効になると、隣接ピンのみを選択できます。

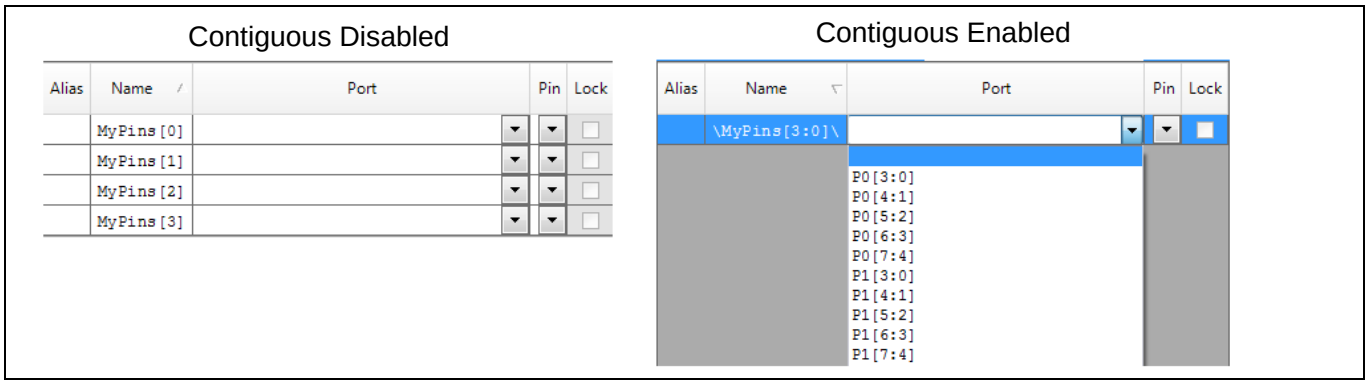


Figure 37　Contiguous オプションが無効/有効の場合のピン配置

これらの機能は、ピン設定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。

5.8　オフチップ コンポーネントについて

オフチップ コンポーネントカタログは、外部と内部コンポーネントを同一の回路図上に組み合わせる方法を提供します。これにより、ドキュメントを改善し、内部回路図がどのように全体の設計に合わせるかをより良く理解できます。オフチップ コンポーネントはコード内のコメントと同じ機能を提供します。それは PSoC™設計の機能を変更せず、システム全体を明確化します。

PSoC™ Creator での GPIO ピン

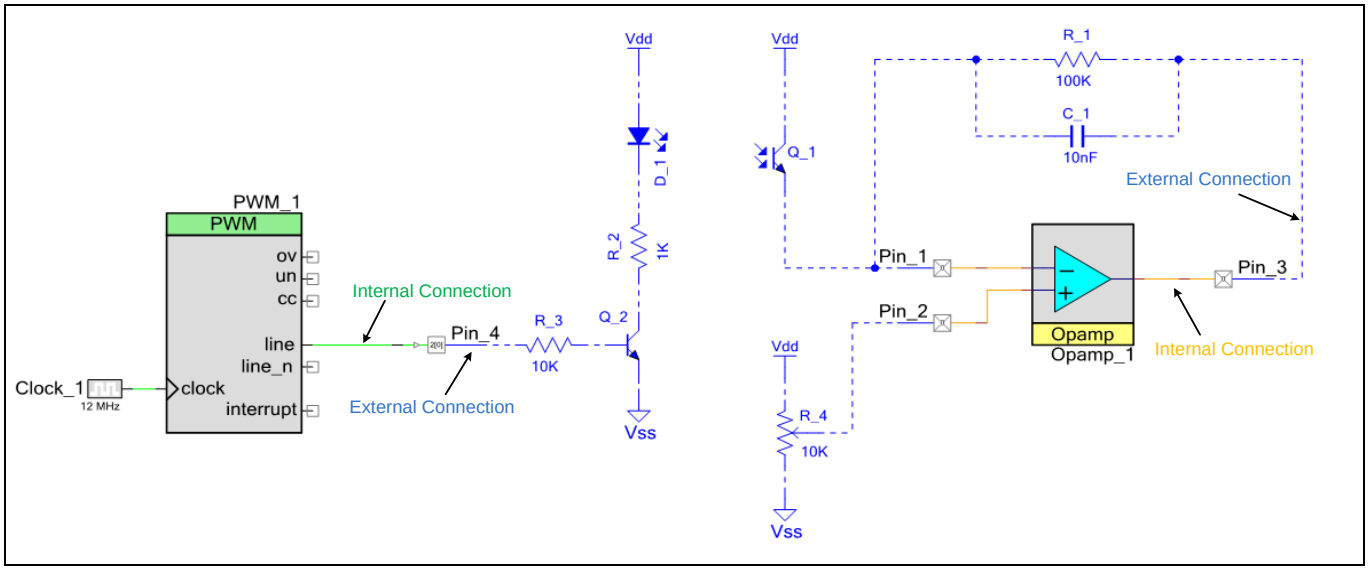


Figure 38　オフチップ コンポーネントのある設計

上で示した設計で、PWM_1 および Opamp_1 はデバイスの内部ブロックです。これらのブロックは Pin_1 ～Pin_4 を使用して外部コンポーネントに接続します。緑色とオレンジ色の線は内部接続です (緑色はデジタル信号、オレンジ色はアナログ信号)。一方、青色の線およびコンポーネントはデバイスの外部接続です。回路図で外部コンポーネントへ接続するために、ピンコンポーネント カスタマイザの「External Terminal」パラメーターを有効にしてください。これにより、回路図上に追加端子が現れます。

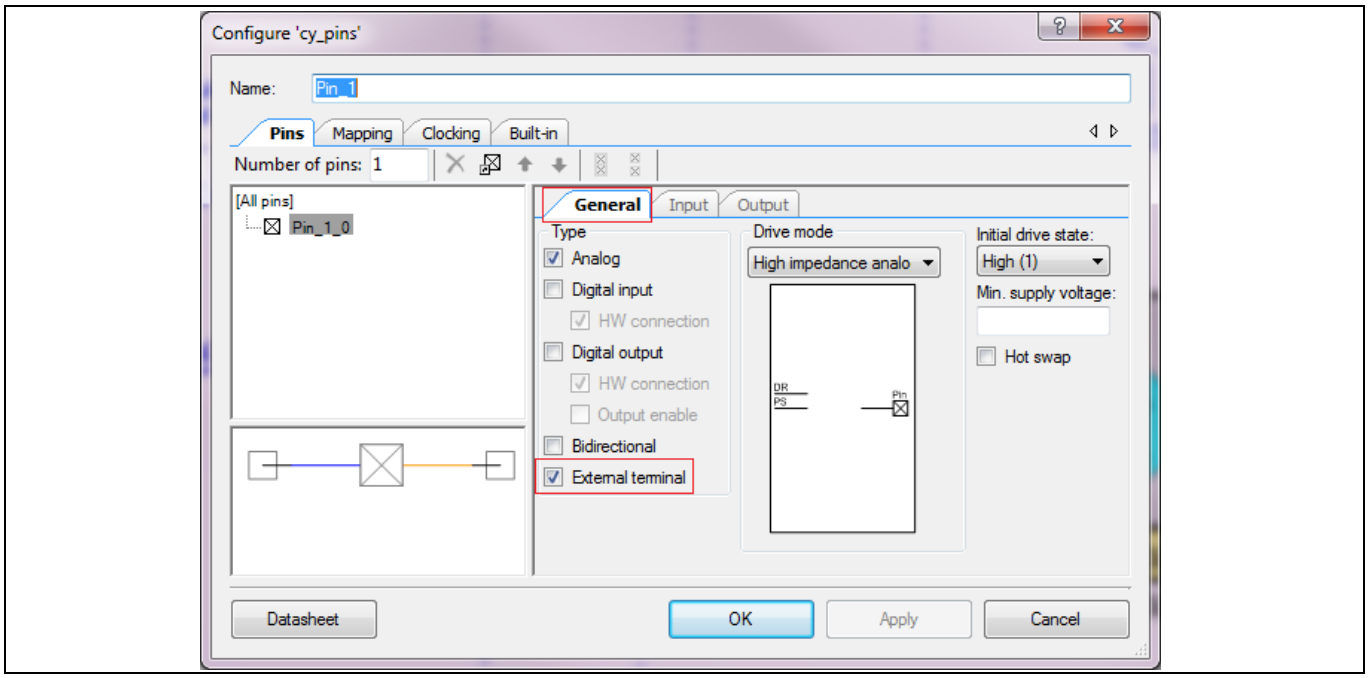


Figure 39　外部端子を有効にする

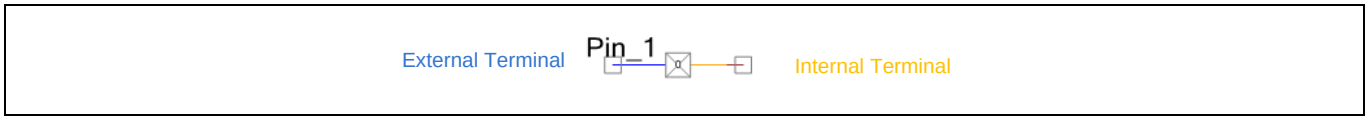


Figure 40　内部端子と外部端子があるピンコンポーネント

回路図で外部端子に接続するコンポーネントは、Components Catalog の Off-Chip タブにあります。

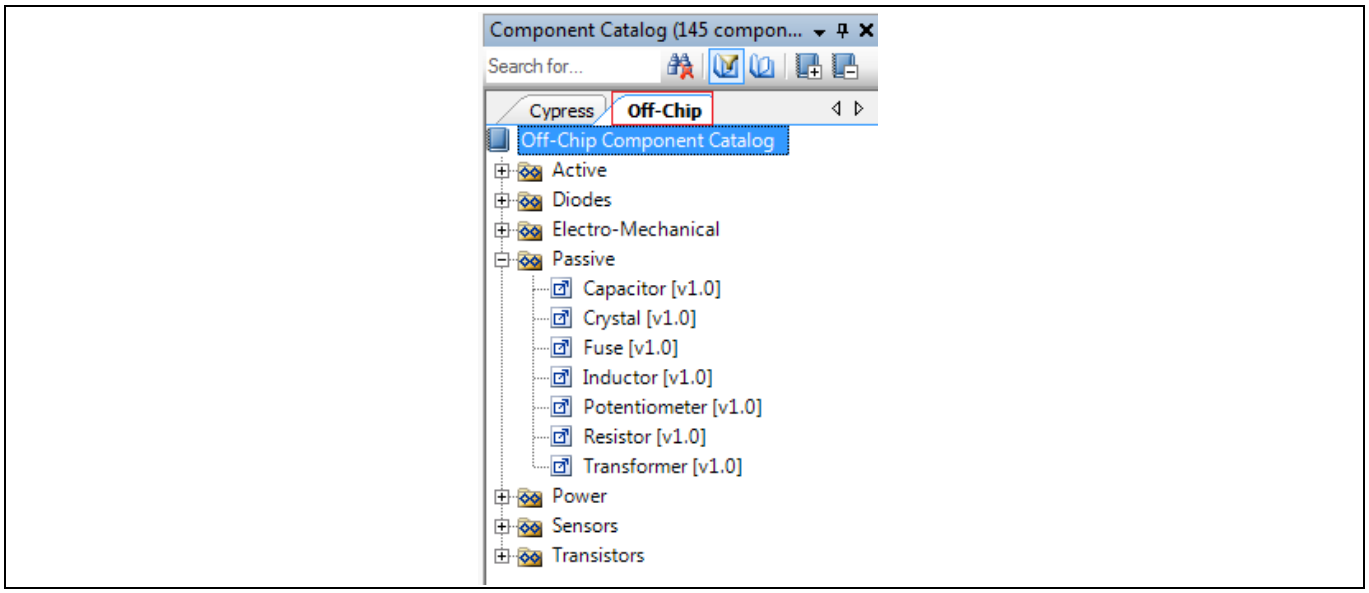


Figure 41　オフチップ コンポーネント カタログ

このカタログのコンポーネントは、基板で PSoC™デバイスのピンに接続される可能性が高いものです。これらのコンポーネントには抵抗, コンデンサ, トランジスタ, インダクタ, スイッチ, およびその他を含みます。内部コンポーネントと同様に、コンポーネントをドラッグして回路図上に配置します。

6 ModusToolbox™での GPIO ピン

ここでは、ModusToolbox™を使用して GPIO ピンの設定および使用方法について説明します。

6.1 ModusToolbox™ Device Configurator を使用した GPIO ピンの設定

6.1.1 Device Configurator の使用

GPIO ピンの初期化は、ここに示すように Device Configurator を使用する方法と、**次のセクションに示す PDL を使用する方法**の 2 つの方法を使用して実行できます。PSoC™ Creator に精通している場合、Device Configurator は、PSoC™ Creator トップ設計のコンポーネントに対して行える構成に似ています。Device Configurator の Pins タブでは、ユーザーは GPIO ピンを初期化し、個々のピンのパラメーターを設定できます。

ModusToolbox™でプロジェクトを右クリックし、ModusToolbox™を選択してから、Device Configurator 2.20 を選択すると、Device Configurator にアクセスできます。Device Configurator には、左下の ModusToolbox™クイックパネルからもアクセスできます。

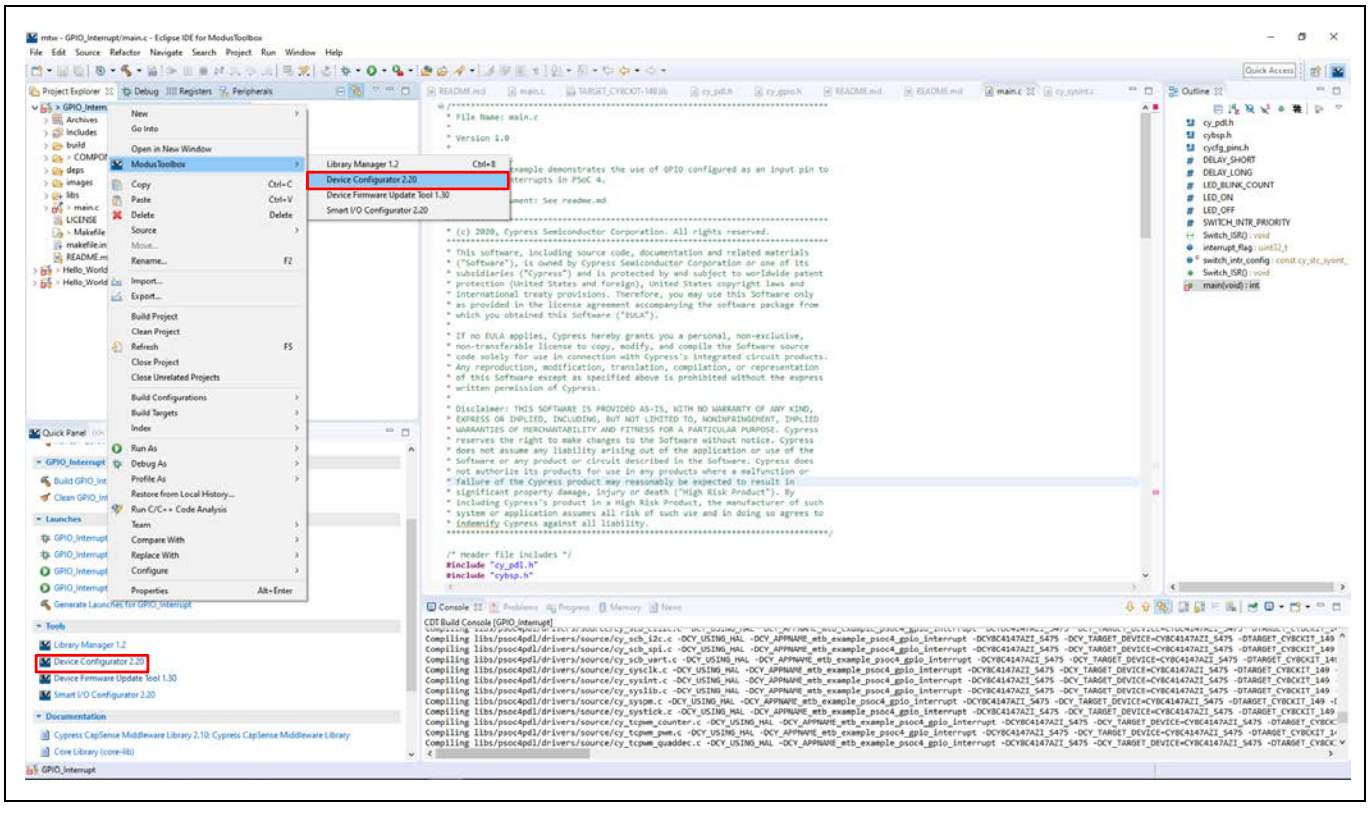


Figure 42 Device Configurator のアクセス

Device Configurator の Pins セクションでは、デバイス上の個々の GPIO ピンを設定できます。左側のパネルは、それぞれのポートで区切られたピンを示します。ピンを設定するためには、ピンの横にあるチェックボックスをオンにします。これにより、初期化コードが作成されます。ピンのパラメーターは、Device Configuration ウィンドウの右側で設定できます。



Figure 43 Device Configurator

Table 5 ピンコンポーネント設定

設定	説明
General > Drive Mode	この設定は、**Figure 44** に示すように、8 つのドライブモードのいずれかでピンを構成します。これらのドライブモードの詳細については、**GPIO ピンの基本情報**を参照してください。
General > Initial Drive State	初期ドライブ状態パラメーターは、データレジスタ値を設定します。**Figure 44** に示すように、この値は、ソフトウェア駆動の場合はピンに反映され、ピンは適切なドライブモードに設定されます。ピンが入力として設定されている場合でも、初期ドライブ状態は有用です。たとえば、入力ピンで抵抗プルアップが必要な場合、抵抗をとおるプルアップパスをオンにするためには、ドライブモードを初期状態が HIGH の **Resistive pull up** に設定する必要があります。同様に、抵抗性プルダウンの場合、プルダウンパスを有効にするためには、初期ドライブ状態を LOW に設定する必要があります。
Input > Threshold	CMOS および LVTTL 入力閾値設定は、ポート全体用です。ポートのすべてのピンが同じように設定されていない場合、Device Configurator の通知リストにエラーが表示されることに注意してください。詳細については、**ピンのデータシート**を参照してください。
Input > Interrupt Trigger Type	この設定は、**GPIO 割込み**で説明されている GPIO エッジ検出ブロックを設定します。割込みの詳細については、**PSoC™ 4 割込みアプリケーションノート**を参照してください。
Output > Slew Rate	スルーレートパラメータは、出力論理レベルを変更する際のピンの立ち上りおよび立ち下りランプレートを決定します。この設定の詳細については、**GPIO ピンの物理構造**を参照してください。
Internal Connection > Analog	この設定により、ピンをアナログ信号に接続できます。

設定	説明
Internal Connection > Digital Output	この設定により、デジタル出力信号に接続できます。
Internal Connection > Digital InOut	この設定により、デジタル入力信号に接続できます。入力信号は主に I²C インターフェースに使用されます。
Advanced > Store Config in Flash	この設定は、構成構造をフラッシュ (const, true) または SRAM (const ではない, false) のどちらに保存するかを制御します。

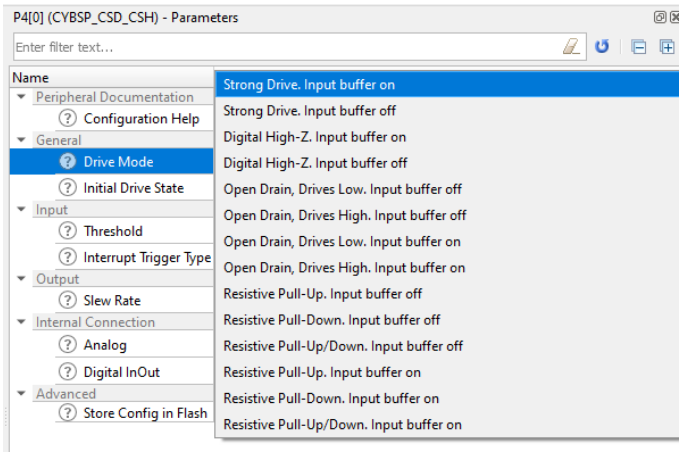


Figure 44　駆動モード

Device Configurator は、ピン, ペリフェラル, およびその他のデバイス設定をセットアップします。Device Configurator を閉じるときは、**File** > **Save** に移動して、現在のプロジェクトの更新を保存します。Device Configurator を使用してリソースを設定した後、PDL を使用してピンおよび割込みと呼応します。初期化後の GPIO ピンの使用の詳細については、**Peripheral Driver Library (PDL) を使用した GPIO** および **PSoC™ 4 PDL API Reference** を参照してください。

Note: *GPIO ピンの名前は、BSP によって提供されるデフォルトの名前から変更できます。これは、設計の目的に対応するようにピンに名前を付けるために役立ちます。ピンの名前は、**Figure 43** に示すように、ピンを選択し、「Name(s)」列のテキスト入力にカスタム名を入力することで変更できます。*

6.1.2 Device Configurator コードプレビュー

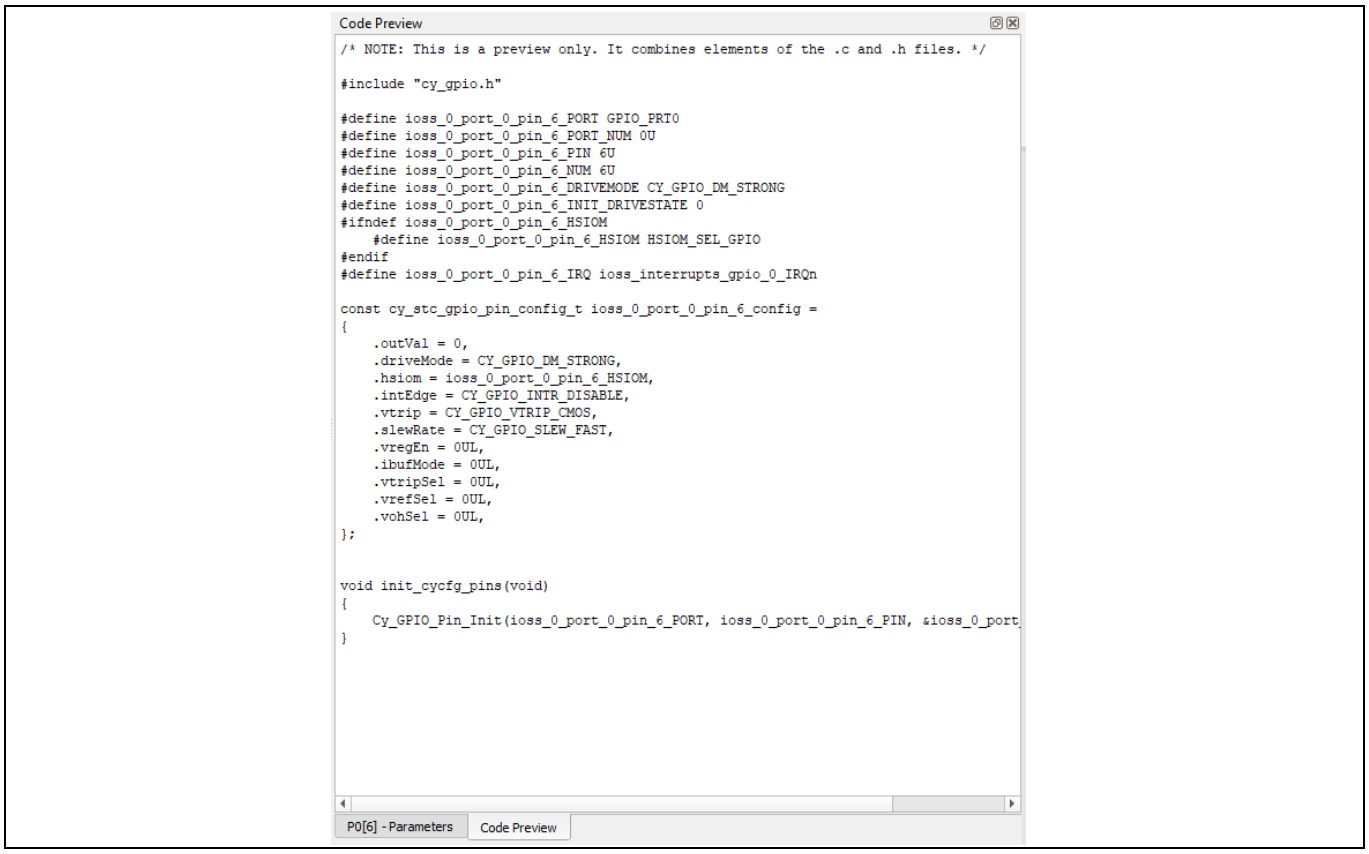


Figure 45 Device Configurator コードのプレビュー

Device Configurator は、**Figure 45** に示すように GPIO ピンを設定するために作成された PDL 定義と関数を表示する Code Preview ウィンドウを備えます。Device Configurator 設定が保存されると、このコードはプロジェクトに自動的に追加されます。このウィンドウから定義をコピーして、このコードをプロジェクトに手動でも追加できます。

6.2 Peripheral Driver Library (PDL) を使用した GPIO

PDL は、デバイスヘッダーファイル、スタートアップコード、および周辺機器ドライバーを 1 つのパッケージに統合します。PDL は、PSoC™ 4 シリーズデバイスファミリに対応します。ドライバーは、ハードウェア関数を一連の使いやすい API に抽象化します。これらは、**PSoC™ 4 PDL API Reference** に完全に文書化されています。PDL は、レジスタの使用法とビット構造を理解する必要性を減らし、PSoC™ 4 シリーズの広範な周辺機器セットのソフトウェア開発を容易にします。アプリケーションのドライバーを設定してから、API 関数呼び出しを使用して周辺機器を初期化して使用します。

このドライバーで使用される Peripheral Driver Library (PDL) GPIO 関数およびその他の宣言は、*cy_gpio.h* にあります。オプションで *cy_pdl.h* (ModusToolbox™のみ) を含めて、PDL 内のすべての関数と宣言にアクセスできます。

初期化は、ポートレベルで、または個々のピンを構成することによって実行できます。サポートされているポートとピンのリストについては、製品デバイスのヘッダーファイルを参照してください。ピンやその他のデバイスリソースを初期化するためには、**Device Configurator** を使用することを推奨します。これにより、不適切な初期化やリソースのオーバーフローのリスクが軽減されます。

シングルピン設定は、Cy_GPIO_Pin_FastInit (特定の引数値を提供) または Cy_GPIO_Pin_Init (充填された cy_stc_gpio_pin_config_t 構造を提供) を使用して実行されます。

充填された cy_stc_gpio_prt_config_t 構造を提供することにより、Cy_GPIO_Port_Init を使用してポート全体を設定できます。構造体の値は、ポートの各ピンの目的の値を表すビットフィールドです。

次の例では、Device Configurator で定義された名前を使用できます。「base」には「user_defined_name」_PORT を使用し、「pin_num」には「user_defined_name」**_NUM** を使用します。ユーザー定義の名前は、**ModusToolbox™ Device Configurator** を使用して簡単に設定および表示できます。

6.2.1 GPIO ピンの初期化-フル

個々の GPIO の初期化は、以下に示す形式で、すべてのピン設定値を定義することから始まります。**ModusToolbox™ Device Configurator** を使用してピンがすでに設定されている場合、この初期化は必要ありません。

Code Listing 1 GPIO ピン設定構造

```
cy_stc_gpio_pin_config_t pinConfig = {
/*.outVal */ 1UL, /* Output = High */
/*.driveMode */ CY_GPIO_DM_PULLUP, /* Resistive pull-up, input buffer on */
/*.hsiom */ P0_3_GPIO, /* Software controlled pin */
/*.intEdge */ CY_GPIO_INTR_RISING, /* Rising edge interrupt */
/*.vtrip */ CY_GPIO_VTRIP_CMOS, /* CMOS voltage trip */
/*.slewRate */ CY_GPIO_SLEW_FAST, /* Fast slew rate */
/*.vregEn */ 0UL, /* SIO-specific setting - ignored */
/*.ibufMode */ 0UL, /* SIO-specific setting - ignored */
/*.vtripSel */ 0UL, /* SIO-specific setting - ignored */
/*.vrefSel */ 0UL, /* SIO-specific setting - ignored */
/*.vohSel */ 0UL /* SIO-specific setting - ignored */
};
```

このピン設定は、GPIO ピンのすべてのパラメーターを定義します。これは、ピンを初期化するときの 3 番目の引数です。

ピンを初期化するためには、以下に示すように、**Cy_GPIO_Pin_Init** (base, pinNum, config) を次の引数で使用します。

Base ピンのポートレジスタのベースアドレスへのポインタ

pinNum ポートレジスタ内のピンビットフィールドの位置

config ピン構成構造のベースアドレスへのポインタ

Code Listing 2　単一 GPIO ピンの初期化 – フルの例

```
/* Initialize pin P0.3 */
if(CY_GPIO_SUCCESS != Cy_GPIO_Pin_Init(P0_3_PORT, P0_3_NUM, &pinConfig))
{
/* Insert error handling */
}
```

6.2.2　GPIO ピンの初期化-高速

GPIO ピン高速初期化すべてのピンタイプの最も一般的な設定を初期化します。これらには、駆動モード, 初期出力値, および HSIOM 接続が含まれます。以下に示すように、次の引数を使用して `CY_GPIO_PIN_FastInit(base, pinNum, driveMode, outVal, hsiom)` でピンを初期化してください。この初期化は、ピンが **ModusToolbox™ Device Configurator** を使用してすでに設定されている場合は必要ありません。

base　ピンのポートレジスタのベースアドレスへのポインタ

pinNum　ポートレジスタ内のピンビットフィールドの位置

driveMode　ピン駆動モード。オプションについては、ピン駆動モードマクロで詳しく説明します。

outVal　ピンを駆動する出力バッファの論理状態 (1 または 0)

hsiom　HSIOM (High-Speed Input Output Multiplexer) 入力選択

Code Listing 3　単一 GPIO ピンの初期化 – 高速の例

```
/* Quickly initialize pin P0.3 (e.g. quickly set up a test LED) */
Cy_GPIO_Pin_FastInit(P0_3_PORT, P0_3_NUM, CY_GPIO_DM_PULLUP, 1UL, P0_3_GPIO);
```

6.2.3　GPIO ポートの初期化

単一の初期化構造からピンの完全なポートを初期化します。

この関数で使用される設定構造には、以下に示す GPIO および HSIOM レジスタへの 1:1 マッピングがあります。それらを設定する方法のレジスタの詳細については、デバイスのテクニカルリファレンスマニュアル (TRM) を参照してください。

Code Listing 4　単一 GPIO ポートの設定構造の例

```
cy_stc_gpio_prt_config_t portConfig = {
/*.dr =*/ 0x00000008u, /* PX.3 output = 1 */
/*.intrCfg =*/ 0x00000080u, /* PX.3 rising edge interrupt */
/*.pc =*/ 0x00000400u, /* PX.3 resistive pull-up */
/*.pc2 =*/ 0x00000000u, /* PX.3 input buffer on */
/*.sio =*/ 0x00000000u, /* PX[7:0] ignored */
/*.selActive =*/ 0x00000000u, /* PX[7:0] software controlled */
};

/* Initialize GPIO port 0 */
if(CY_GPIO_SUCCESS != Cy_GPIO_Port_Init(GPIO_PRT0, &portConfig))
{
/* Insert error handling */
}
```

6.2.4　GPIO ピンからの読み出し

ピンが **Device Configurator** または PDL を使用して設定されている場合、GPIO ピンからの読み出しは同じです。以下に示すように、ポートとピンは CY_GPIO_Read 関数を使用する場合の引数です。

Code Listing 5　単一 GPIO ピンの読み出し

```
/* Scenario: P0.3 was initialized and input buffer enabled */
/* Read the input state of P0.3 */
if(1UL == Cy_GPIO_Read(P0_3_PORT, P0_3_NUM))
{
/* Insert logic for High pin state */
}
else
{
/* Insert logic for Low pin state */
}
```

6.2.5 GPIO ピンへの書き込み

GPIO ピンへの値の書き込みは、ピンが **Device Configurator** または PDL を使用して設定されている場合と同じです。以下に示すように、`CY_GPIO_Write` 関数を使用する場合、ポートとピンは引数です。

Code Listing 6 単一 GPIO ピンの書き込み

```
uint32_t pinState = 0UL;

/* Control P0.3 based on the pinState variable */

Cy_GPIO_Write(P0_3_PORT, P0_3_NUM, pinState);
```

6.2.6 GPIO 割込み

ピンの GPIO 割込みは、**Figure 43** に示すように Device Configurator を使用するか、PDL を使用するかの 2 つの方法のいずれかを使用して設定されます。Device Configurator を使用してピンを設定する場合は、**Figure 43** に示すように、サイドパネルで割込みタイプを選択できます。割込みタイプは、GPIO PDL 設定構造でも設定できます。割込みトリガータイプを設定する方法については、**PSoC™ 4 PDL API Reference** を参照してください。

ModusToolbox™の PSoC™4 S シリーズデバイスでの割込みの使用の詳細については、**PSoC™ 4: GPIO Interrupt Code Example on GitHub** を参照してください。このサンプルコードは、ModusToolbox™の New Application Wizard にもあります。詳細については、**ModusToolbox™サンプルコード**を参照してください。

7 PSoC™ Creator での GPIO のヒントおよびコツ

ここでは PSoC™ Creator を使用する際の GPIO ピンの使用方法の実例を提供します。

Table 6 PSoC™ Creator プロジェクト

#	セクション	PSoC™ 4000, 4000S, 4100S, 4100S Plus, 4100PS, 4500S, 4700S	PSoC™ 4100_BLE, 4100, 4100M	PSoC™ 4200, 4200M, 4200L, 4200-BLE
1	LED をトグル	✓	✓	✓
2	入力の読み出しと出力への書き込み	✓	✓	✓
3	デジタル ロジック ゲートから出力の駆動			✓
4	双方向ピンの使用		✓	✓
5	GPIO 入力/出力同期の設定		✓	✓
6	データレジスタでの GPIO のより速いトグル	✓	✓	✓
7	GPIO 出力イネーブル ロジックの設定			✓
8	ピン割込み	✓	✓	✓
9	ファームウェアでの GPIO 割込みの設定	✓	✓	✓
10	GPIO でのアナログとデジタルの両方の使用		✓	✓
11	より大きな駆動/シンク電流のためのピンの連動			✓
12	ディープスリープにおける制御レジスタの取り扱い			✓

PSoC™ 4 開発ボードに記載されるインフィニオン開発キットは、これらのプロジェクトのテストに使用できます。

7.1 LED をトグル

GPIO の最も簡単な使用法は、ファームウェアでピンの出力を HIGH か LOW にセットすることです。この例は、ピンコンポーネント API 関数を使って LED をトグルする出力を設定する方法を示します。

1. プロジェクト回路図にデジタル出力ピンコンポーネントを配置してください。
2. コンポーネントに「Pin_LED」と名前を付け、ハードウェア接続を無効にしてください。

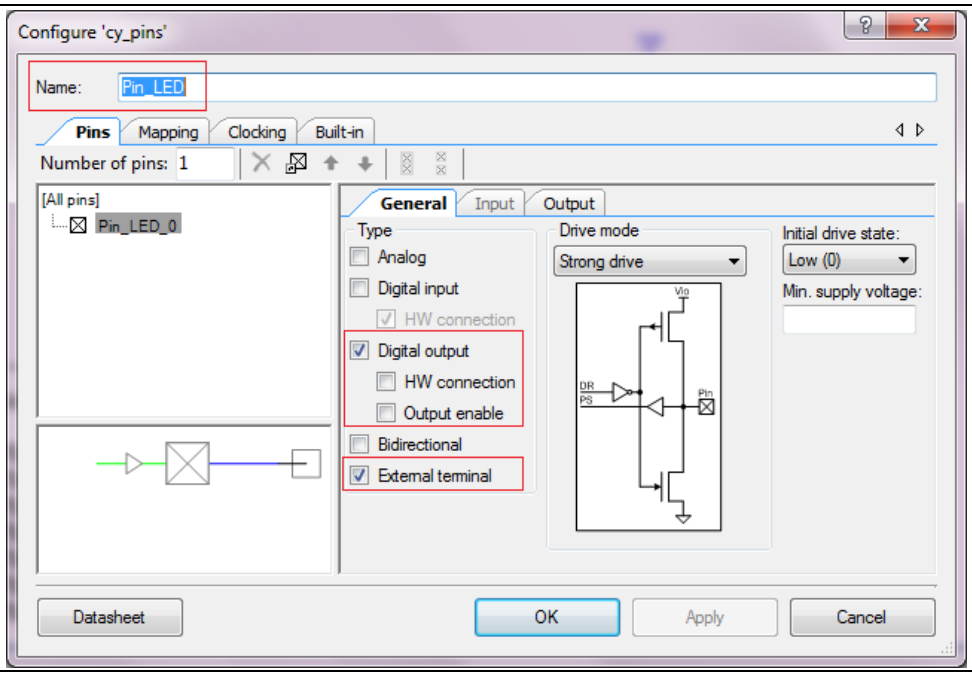


Figure 46　Pin_LED の設定

3. 回路図で外部コンポーネントに接続するため、外部端子を有効にしてください。
4. DWR ウィンドウの Pins タブで、物理ピン (この例では P1[6]を使用)の P1[6]) に割り当ててください。
5. 物理ピンを LED に接続してください。LED および抵抗「R」はオフチップ コンポーネントであることに注意してください。

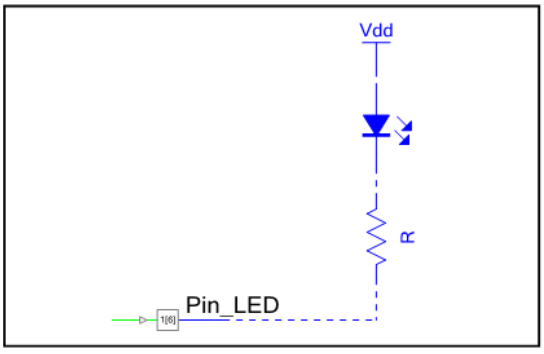


Figure 47　LED トグル例の回路図

6. *main.c* で、以下のようにコンポーネント API を使って出力を設定してください。

```
for(;;)
{
   /* Set LED output to logic HIGH */
   Pin_LED_Write(1u);

   /* Delay of 500 ms */
```

```
    CyDelay(500u);


    /* Set LED output state to logic LOW */
    Pin_LED_Write(0u);


    /* Delay of 500 ms */
    CyDelay(500u);
}
```

7. プロジェクトをビルドし、PSoC™ 4 デバイスをプログラムしてください。

結果として、LED が 1Hz の周波数で点滅します。

7.2 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 関数を使用した GPIO ピンの読み出し/書き込みの方法を説明します。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 (ハードウェア接続が無効になる 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てください。

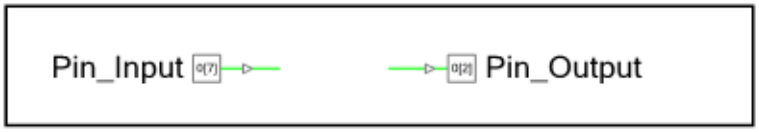


Figure 48 入力と出力の回路図例

2. .cydwr ウィンドウでピンを Pin_Input と Pin_Output に割り当ててください。
3. 以下のように、Pin_Input に基づき、コンポーネント API を使って Pin_Output の状態を設定してください。

```
  for(;;)
    {
/* Set the output pin with an inverted value of input pin */
      Pin_Output_Write(~Pin_Input_Read());
    }
```

4. プロジェクトをビルドし、PSoC™ 4 デバイスをプログラムしてください。

信号発生器から方形波を Pin_Input に供給することにより、このプロジェクトをテストできます。Pin_Output での信号は Pin_Input での信号の反転となる必要があります。

7.3 デジタル ロジック ゲートから出力の駆動

前述の例では、ピンを読み出し、他のピンをその読み出した値の反転で設定するためにプロセッサ コアを使用する方法を示しました。この例は同様ですが、汎用デジタルブロック (UDB) として知られる設定可能なデジタルリソースを使用します。この例では、入力ピン信号は NOT ゲートに、NOT ゲートの出力は他のピンに配線されます。以下のステップに従ってプロジェクトを作成してください。

1. 2 本のピン (ハードウェア接続が有効になる 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てください。



Figure 49 ハードウェア接続が有効になる入力および出力ピン

2. **Figure 50** に示すように、NOT ゲートを配置し、ピンに接続してください。

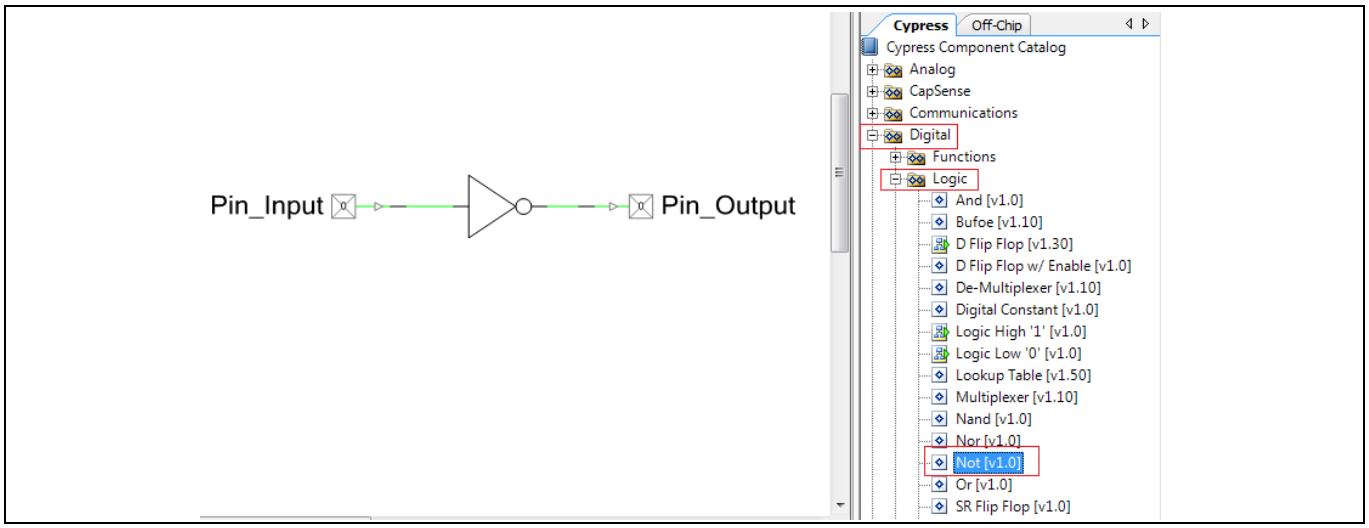


Figure 50 NOT ゲートの接続

3. .cydwr ウィンドウでピンを Pin_Input と Pin_Output に割り当ててください。
4. このプロジェクトはいずれのコードを必要としません。プロジェクトをビルドし、PSoC™ 4 デバイスをプログラムしてください。
5. 前述のプロジェクトと同じように、信号発生器から方形波を Pin_Input に供給することにより、このプロジェクトをテストできます。Pin_Output での信号は Pin_Input での信号の反転となります。

7.4 双方向ピンの使用

この例は、デジタル入力とデジタル出力の両方がアクティブになる双方向モードでピンを使用することを示します。PSoC™ Creator は双方向設定でのピンコンポーネント (デジタル双方向ピン) を提供します。



Figure 51 Pin_Bidirectional

しかし、このピンコンポーネントは入出力の単一の端子を示します。この使用は I²C SDA および SCL ラインに限定されます。多くのアプリケーションでは、2 つの端子 (入力用 1 つと出力用 1 つ) を持っていると便利です。これはピンコンポーネント カスタマイズの Digital Input および Digital Output オプションの両方を有効にすることで実現できます。そのようなピンを設定する例は以下に示します。この例では、スイッチは入力側で接続され、ピンを論理 LOW にプルダウンします。このピンは論理「1」で連続的に駆動される抵抗プルアップに設定します。双方向ピンの状態をチェックするためには、その信号を他のピンに配線します。以下のステップに従って、プロジェクトを作成します。

回路図にデジタル入力ピンを配置してください。

Digital Output を有効にし、Drive mode を Resistive pull up に設定してください。

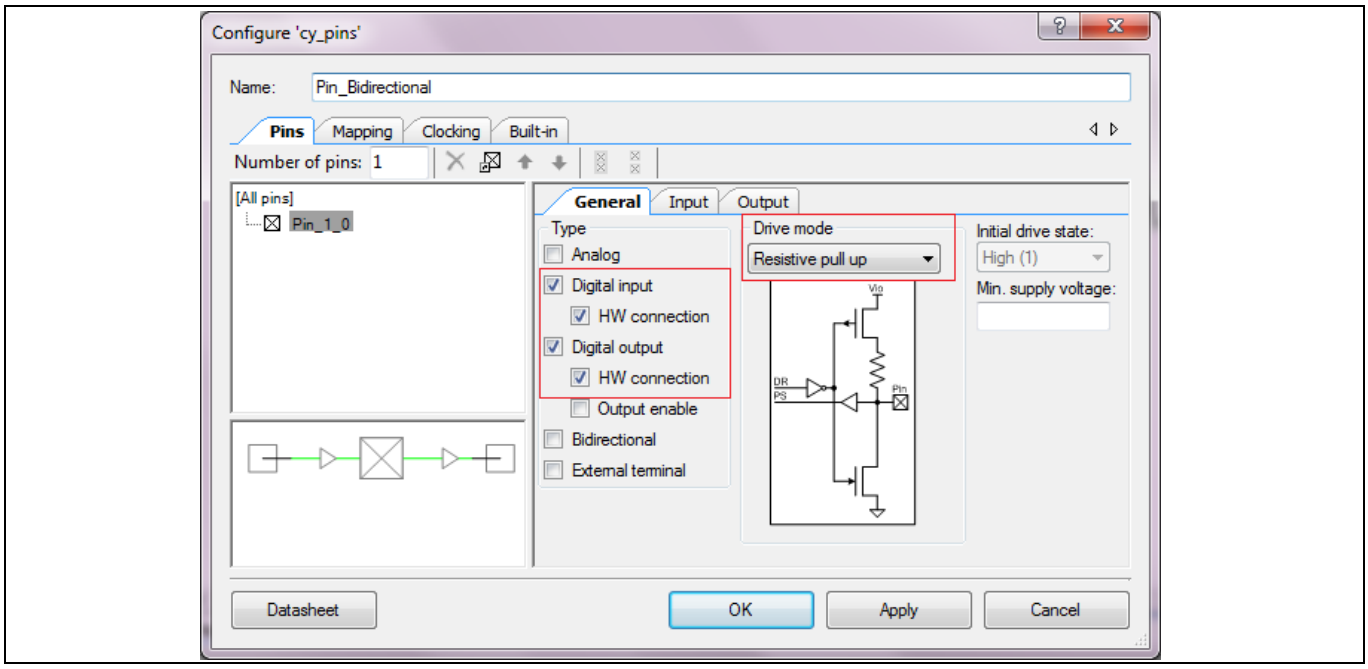


Figure 52 Pin_Bidirectional の設定

コンポーネントが以下に示す回路図のようになることを確認してください。

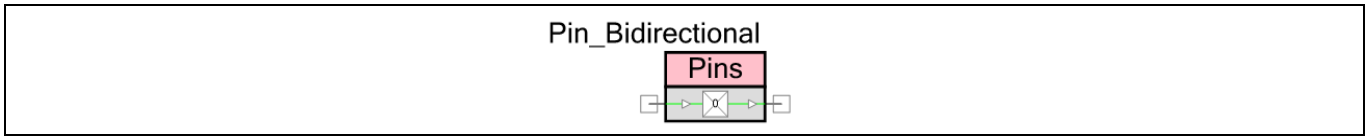


Figure 53 回路図上の Pin_Bidirectional

論理 HIGH に接続して、抵抗プルアップに設定されるピンを連続的に駆動してください。

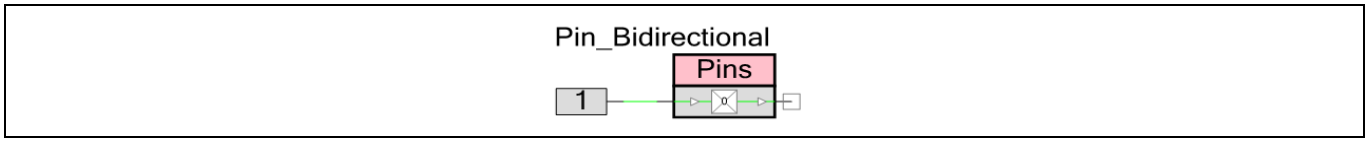


Figure 54 論理 HIGH を Pin_Bidirectional に接続

現在、入力側に接続される他のピン (Pin_Status) を使用して、ピンの状態を見ることができます。デジタル出力ピンを配置し、Pin_Bidirectional の入力バッファ側に接続してください。

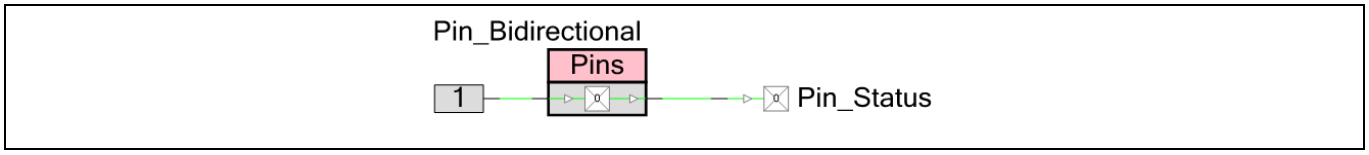


Figure 55　Pin_Status を Pin_Bidirectional に接続

外部接続が有効になると、回路図は以下の図のようになります。

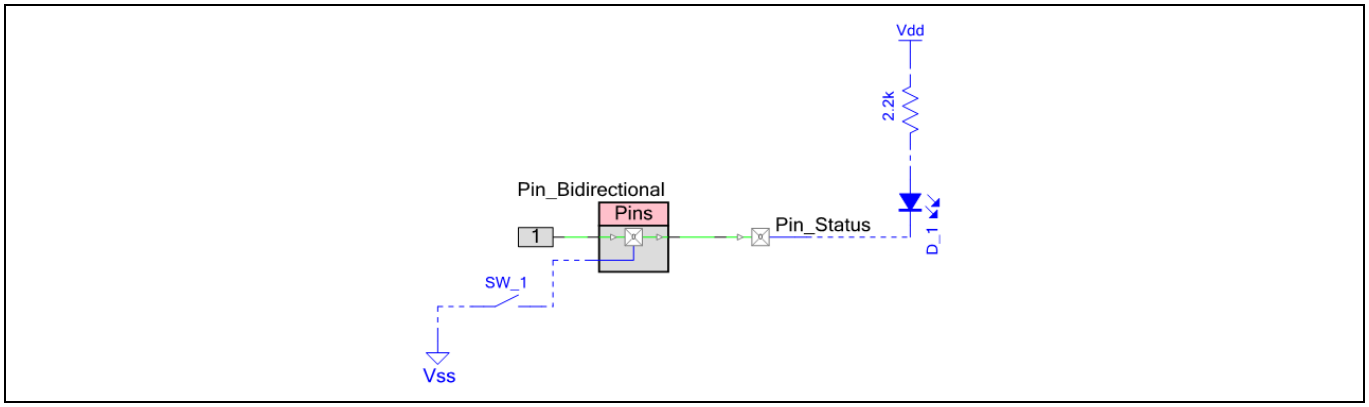


Figure 56　完全な回路図

.cydwr ウィンドウでピンの割り当ててください。

このプロジェクトは、いずれのコードも必要ありません。プロジェクトをビルドし、PSoC™ 4 デバイスをプログラムしてください。

Pin_Status に接続される LED は Pin_Bidirectional の入力バッファの状態を示します。スイッチを押さない場合、論理「1」は抵抗プルアップを介して Pin_Bidirectional を駆動します。これは LED をオフにします (LED がアクティブ LOW モードで接続されるため)。スイッチを押す場合、ストロング論理「0」は Pin_Bidirectional に現れます。これは LED を ON にします。したがって、このプロジェクトは同じピン (Pin_Bidirectional) で 2 つのドライバーを表示します (1 つは内部 (論理 1) であり、もう 1 つは外部 (入力として動作するスイッチ) です)。

7.5　GPIO 入力/出力同期の設定

デジタル入力および出力信号に対して、GPIO は内部クロック (HFCLK) との同期を提供し、PSoC™ 4 デバイス (PSoC™ 4000 を除く) でクロックのようなデジタル信号との同期を提供します。また、クロック イネーブルと同期ロジック リセットのコンフィギュレーションを提供します。以下の 2 つの図に示すように、ポート アダプタ ロジックは入出力の同期のために使用されます。

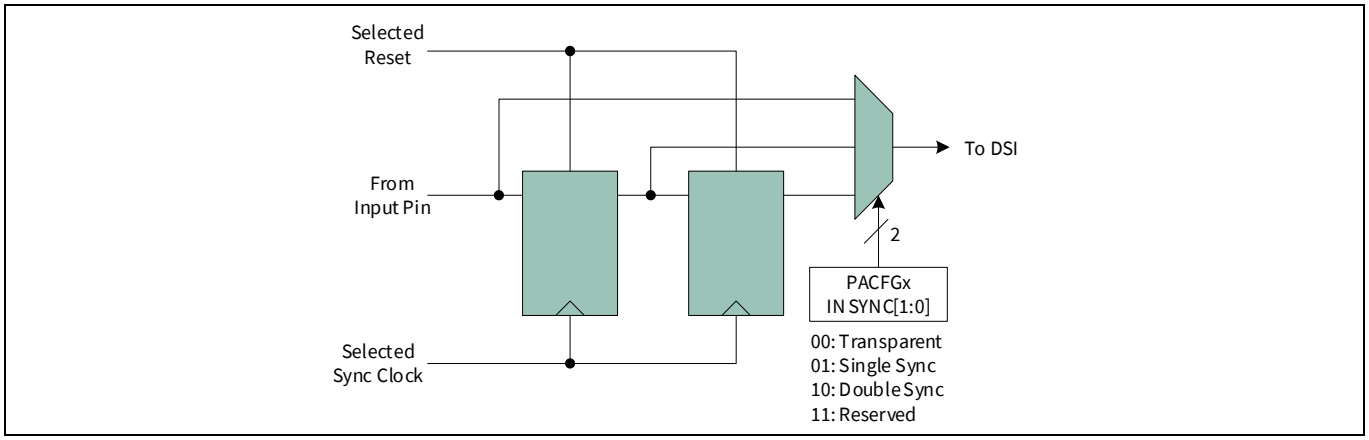


Figure 57　PSoC™ 4 での入力同期

Figure 57 に示すように、入力同期回路には透過, シングル同期およびダブル同期のオプションがあります。

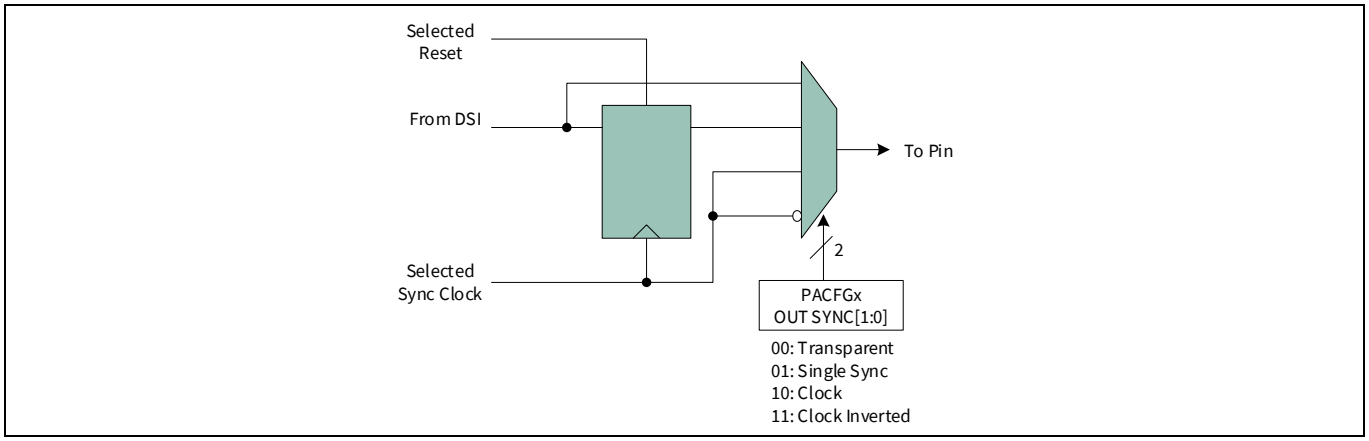


Figure 58　PSoC™ 4 の出力同期

Figure 58 に示すように、出力同期回路には透過, シングル同期, クロック, および反転クロックのオプションがあります。クロックと逆転クロックは同期クロックを出力ピンに配線します。**Figure 59** に示すように、これらはピンコンポーネント カスタマイザで設定されます。

同期クロックは HFCLK、外部信号 (DSI から)、またはいずれかのピン信号に設定できます。同期ブロック リセット信号は外部信号 (DSI から) またはいずれかのピン信号に設定できます。**Figure 60** に示すように、これらはピンコンポーネント カスタマイザの Clocks タブで設定されます。

ピンコンポーネント カスタマイズの Clocking タブのパラメーターについては、**Pins Component datasheet** を参照してください。

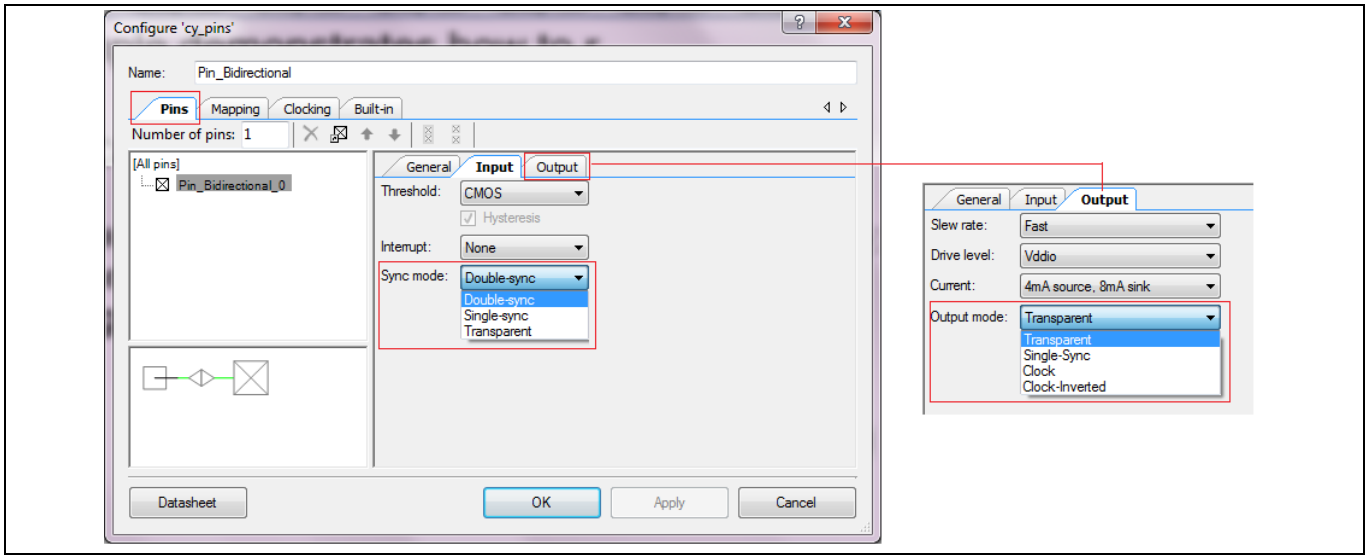


Figure 59　同期モードの設定

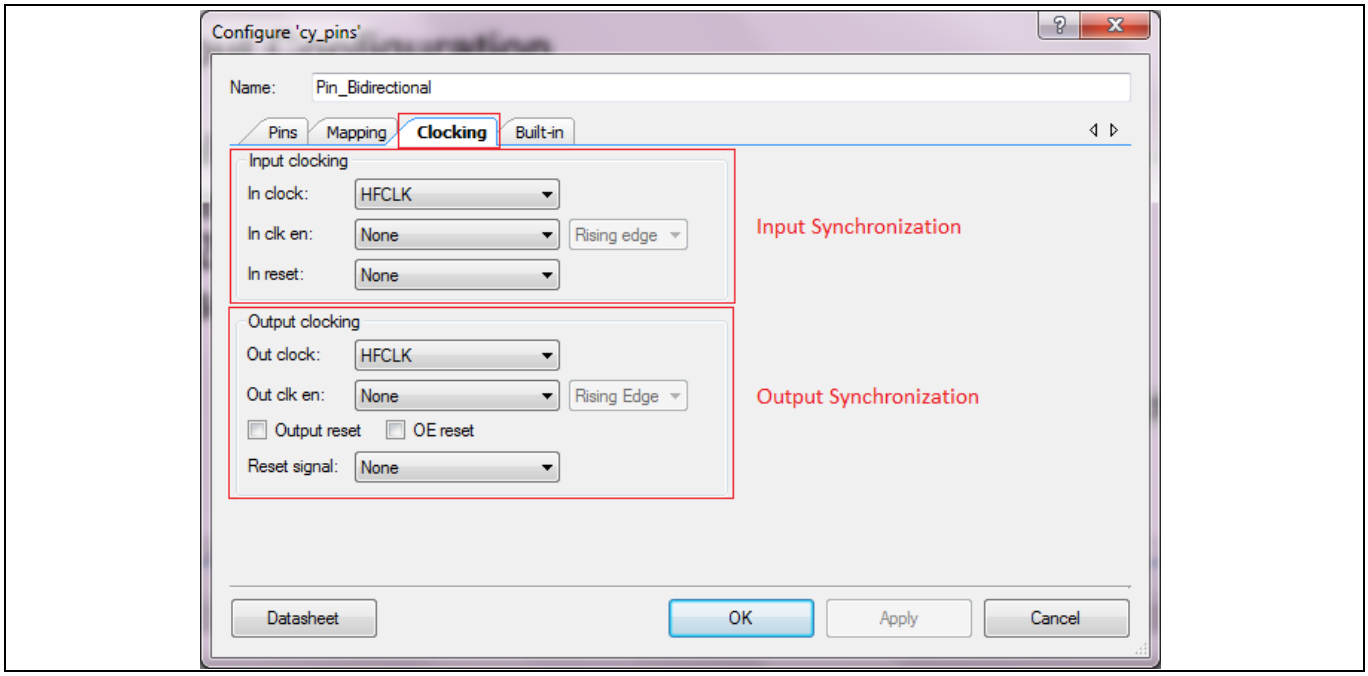


Figure 60　クロック設定

ピン信号は、UDB ポートアダプタと GPIO ブロックとの組合せで同期されます。また、クロックは内部クロック同期のために、ポートのすべてのピンは共有されます。詳細は **PSoC™ 4 architecture TRM** を参照してください。

Note: *ポート 4 以上のピンは UDB ポートアダプタを持っていないため、それらのピン信号を同期化できません。そのため、これらのポートピンはビルドの時にエラーを回避するために透過モード (Transparent mode) で使用する必要があります。*

次の 2 つの例では、入力/出力同期の設定方法を説明します。

7.5.1 GPIO 入力同期

1. 1 本のデジタル入力ピン、2 本のデジタル出力ピンおよび 1 本のクロックコンポーネントをプロジェクト回路図に配置してください。以下に示すように、それらを設定してください。

Table 7 コンポーネント コンフィギュレーション

コンポーネント	名称	設定
デジタル入力ピン	Pin_Input_DoubleSync	駆動モード: 抵抗プルアップ 同期モード: ダブル同期 クロック内 (入力同期用): 外部
デジタル出力ピン	Pin_Output_LFCLK_1	出力モード: 透過
デジタル出力ピン	Pin_Output_Transparent_1	出力モード: 透過
クロック	LFCLK	クロックタイプ: 現存のクロック ソース: LFCLK

2. ピンを接続して、オフチップ コンポーネントを加えてください。

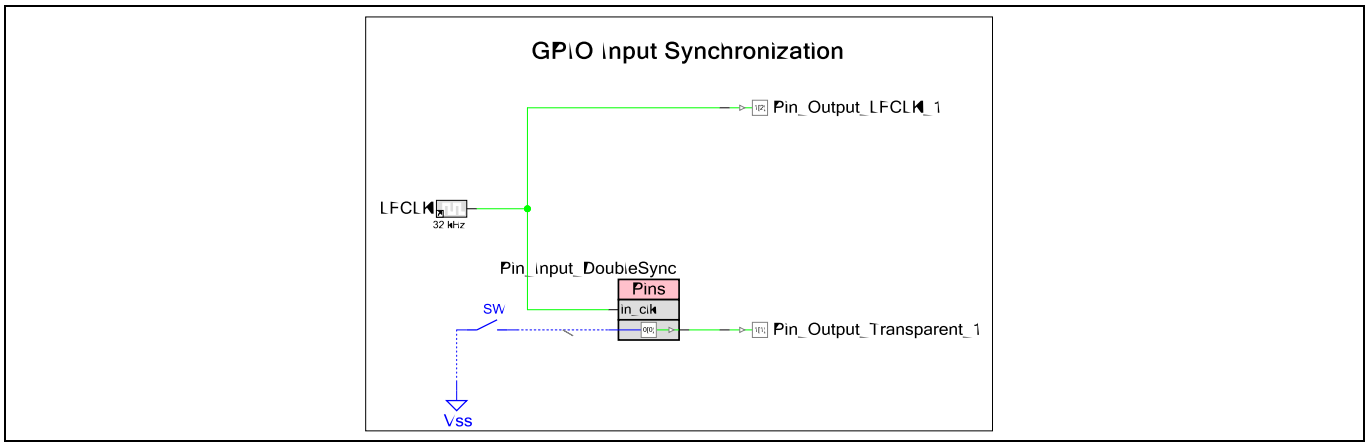


Figure 61 GPIO 入力同期回路図の例

*Note: PSoC™ 4 のクロックは、SYSCLK と LFCLK を除き、直接的にピン端子に接続できません。詳細については、**PSoC™ 4 architecture TRM** の「Clocking System」を参照してください。*

3. ピンを割り当てて、Pin_Input_DoubleSync ピンをグランドと接続されているスイッチに接続してください。
4. プロジェクトをビルドし、PSoC™ 4 デバイスをプログラムしてください。
5. Pin_Input_DoubleSync に接続したボタンが押される時、信号波形は発生します。入力が LFCLK と二重同期されるため、Pin_Output_Transparent_1 ピンは LFCLK の 2 番目の立ち上りエッジで LOW になります。

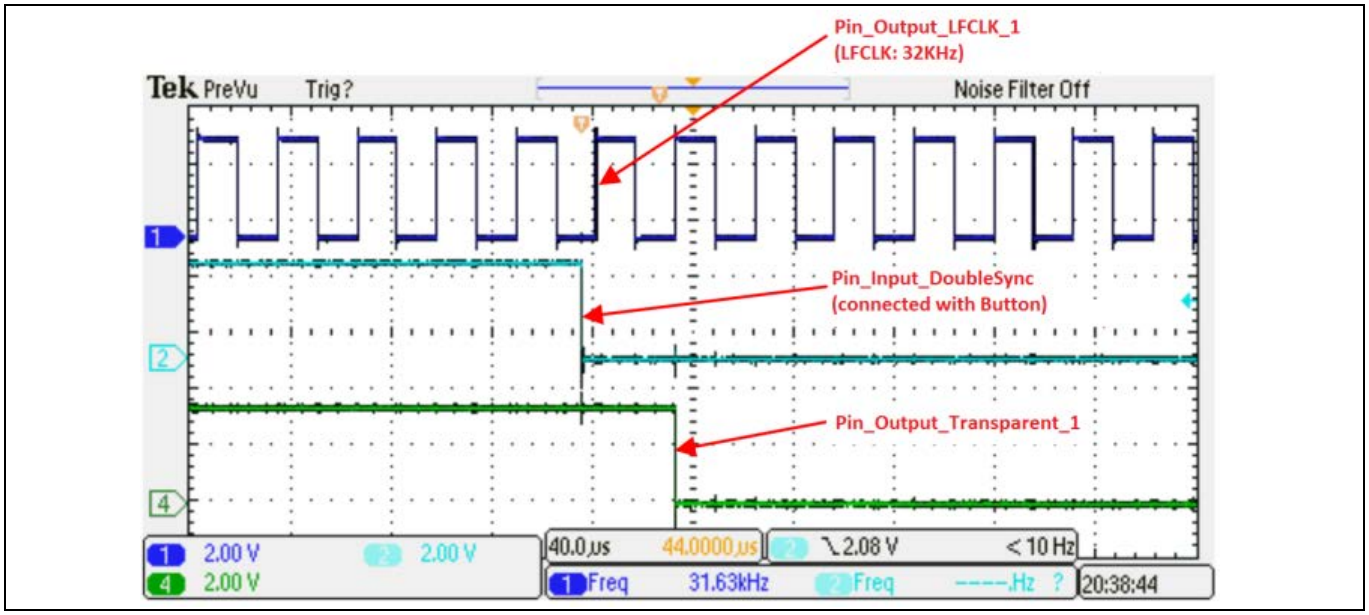


Figure 62　入力/出力信号波形

7.5.2　GPIO 出力同期

1. 1 本のデジタル入力ピン、3 本のデジタル出力ピンと 1 つのクロック コンポーネントをプロジェクト回路図に配置して、それらを **Table 8** のように設定してください。

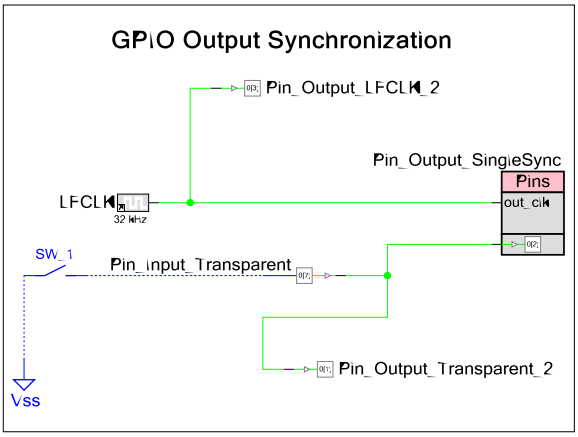


Figure 63　GPIO 出力同期回路図の例

Table 8　ピンコンフィギュレーション

コンポーネント	名称	設定
デジタル入力ピン	Pin_Input_Transparent	駆動モード: 抵抗プルアップ 同期モード: 透過
デジタル出力ピン	Pin_Output_LFCLK_2	出力モード: 透過
デジタル出力ピン	Pin_Output_SingleSync	出力モード: シングル同期 出力クロック (出力同期用): 外部
デジタル出力ピン	Pin_Output_Transparent_2	出力モード: 透過
クロック	LFCLK	クロックタイプ: 現存のクロック ソース: LFCLK

2. **Figure 63** に示すように、ピンを接続してください。
3. ピンを割り当てて、Pin_Input_Transparent ピンをグランドと接続されているスイッチに接続してください。ポート 4 以上のピンは同期機能をサポートしないため、Pin_Output_SingleSync に選択できないことに注意してください。
4. プロジェクトをビルドし、PSoC™ 4 デバイスをプログラムしてください。

Figure 64 に、ボタン押しに対応する波形を示します。

出力が LFCLK と同期化されるため、Pin_Output_SingleSync ピンは LFCLK の次の立ち上りエッジで LOW になります。同期がないため、Pin_Output_Transparent_2 ピンは Input_Transparent ピンと同時に LOW になります。

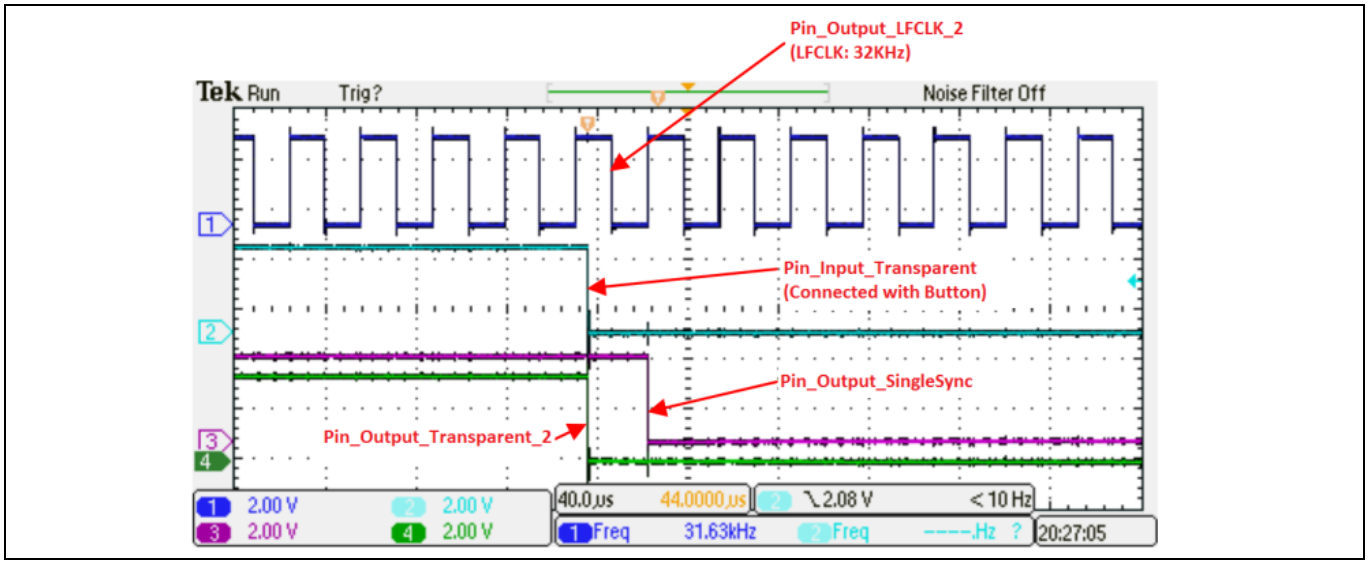


Figure 64　入力/出力信号波形

7.6　データレジスタでの GPIO のより速いトグル

コンポーネント API 関数を呼び出すことは、GPIO ピンを制御するために最も簡単ですが、最も速い方法ではありません。

ピン P5[2]にマップされた Pin_1 に論理「1」を書き込む例を参照してください。以下は API 関数呼び出しです。

```
Pin_1_Write(1);
```

同等のアセンブリコードは、プロジェクト ワークスペースの Results タブのリストファイル (*main.lst*) で見ることができます。

```
mov     r0, #1      ;load the value in r0
bl Pin_1_Write      ;call Pin_1_Write
```

コンポーネント API の Pin_1_Write 関数のアセンブリコードもリストファイルで見ることができます。

```
ldr    r3, .L2      ;load the address of Pin_1_DR into r3
```

```
ldr   r1, [r3]     ;load the value of Pin_1_DR into r1
mov   r2, #251     ;load 251 into r2 (value depends on location of pin
                   ;in 8 bit wide port, in this case, Pin_1 is on port P5[2])
and r2, r1         ;AND the values of r2 and r1 and load result back in r2

lsl   r0, r0, #2   ;left shift r0 by two bits and load the result back in
           ;r0(this instruction is not present for the pin on LSB)
mov   r1, #4         ;load value of 4 into r1 (depends on the location of
           ;pin in 8 bit wide port)
and   r0, r1         ;and the value of r0(contains "value")and r1 and load
           ;the result in r0
orr   r0, r2         ;or the value of r0 with r2 and load the result back in
           ;r0
str   r0, [r3]     ;store the result back in Pin_1_DR
      bx    lr        ;return to calling function
```

このコードは Pin_1 に論理「1」を書き込むために 20 CPU サイクルかかります。

また、ピンを速く更新するために、それぞれのピン コンポーネントのために作成された*<pin_name>.h* ファイルでレジスタ定義およびマスクを使用できます。

以下のとおり、P5[2]にマップされた Pin_1 に論理「1」をセットします。Pin_1_DR は Pin_1 のデータレジスタです。

```
Pin_1_DR |= Pin_1_MASK
```

リストファイル (*main.lst*) では、上記の命令は以下のとおりにアセンブリコードになります。

```
ldr   r3, .L3;load the address of Pin_1_DR into r3
ldr   r1, [r3]   ;load value of Pin_1_DR into r1
mov   r2, #4     ;move value of 4 (Pin_1_MASK) into r2
orr   r2, r1     ;Set the bit in Pin_1_DR
strb  r2, [r3]   ;Store it back into Pin_1_DR
```

コンポーネント API 関数を使用して 20 サイクルかかる場合に比較して、このコードは 8 サイクルかかります。

コンポーネント API 関数はファームウェアを介して、直接レジスタ書き込みを必要としない以下のアクションを実行します。

- 関数の呼び出し

PSoC™ Creator での GPIO のヒントおよびコツ

- ピンを論理「1」または論理「0」にセットするために関数引数をチェック
- 関数から返す

直接レジスタ書き込みを使用してピンをセット、リセット、および読み出すために、PSoC™ Creator に以下のマクロが提供されます。

マクロ	説明
`CY_SYS_PINS_SET_PIN(portDR, pin)`	ピンの出力値を論理 HIGH に設定する portDR はポート データレジスタのアドレス pin はピン番号 (0〜7)
`CY_SYS_PINS_CLEAR_PIN(portDR, pin)`	ピンの出力値を論理 LOW にクリアする portDR はポート データレジスタのアドレス pin はピン番号 (0〜7)
`CY_SYS_PINS_READ_PIN(portPS, pin)`	ピン値を読み出す portPS はポート ステータスレジスタのアドレス pin はピン番号 (0〜7)

これらのマクロの詳細はシステム リファレンスガイド (PSoC™ Creator のヘルプメニューから利用できる) を参照してください。

以下の命令を実現して、API 関数呼び出しと直接レジスタ書き込みの性能を比較するために使用できる PSoC™ Creator プロジェクトを作成してください。

1. ハードウェア接続が無効になる 2 本のデジタル出力ピンをプロジェクト回路図に配置し、「Pin_Test」および「Pin_Index」と名前を付けてください。

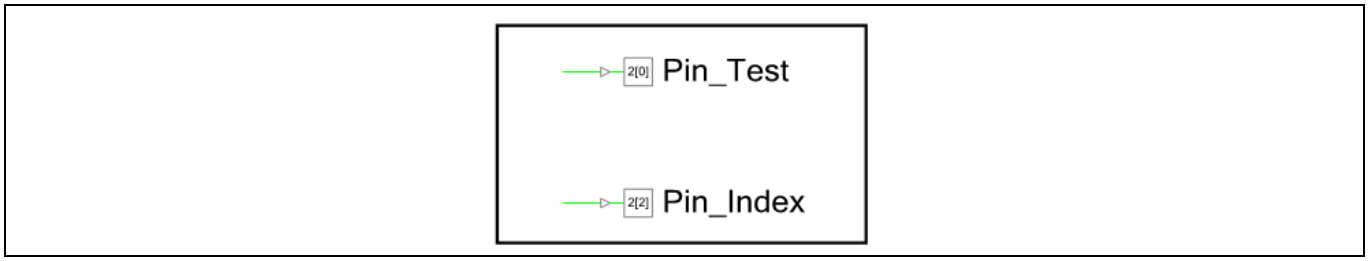


Figure 65　データレジスタ回路図例で GPIO をトグル

2. .cydwr ウィンドウでピンを割り当ててください。
3. *main.c* ファイルに以下のコードを追加してください。このコードは Pin_Index を HIGH に設定し、コンポーネント API 関数を使って Pin_Test をトグルします。その後、Pin_Index を LOW に設定し、データレジスタ (DR) を使用して Pin_Test をトグルします。

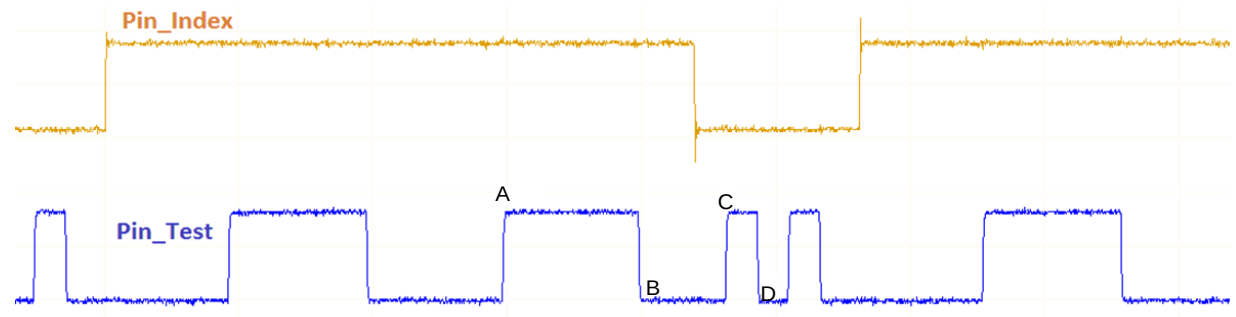
```
for(;;)
  {
      /* Set IndexPin */
      Pin_Index_Write(1);



      /* Set TestPin */
      Pin_Test_Write(1u);
```

```
        /* Clear TestPin */
        Pin_Test_Write(0u);

        /* do it again */
        Pin_Test_Write(1u);
        Pin_Test_Write(0u);

        /*************************************************/

        /* Clear IndexPin */
        Pin_Index_Write(0);

        /************* Direct Register Writes **************/

        /* Set TestPin */
        CY_SYS_PINS_SET_PIN(Pin_Test__DR, Pin_Test_SHIFT);

        /* Clear TestPin */
        CY_SYS_PINS_CLEAR_PIN(Pin_Test__DR, Pin_Test_SHIFT);

        /* do it again */
        CY_SYS_PINS_SET_PIN(Pin_Test__DR, Pin_Test_SHIFT);
        CY_SYS_PINS_CLEAR_PIN(Pin_Test__DR, Pin_Test_SHIFT);

        /*************************************************/
    }
```

Note: *`Pin_Test__DR` はデータレジスタのアドレスであり、一方 `Pin_Test_DR` もデータレジスタの値です。データレジスタ アドレス用のマクロの詳細はプロジェクト ワークスペースのソースファイル フォルダにある Pin Component.h ファイル (この場合では Pin_Test.h) を参照してください。*

ピンの割り当ては開発中に変更される場合、コードを変更する必要がないようにデータレジスタへ書き込まれるコードは、API 関数呼び出しと同じように移植可能なものです。

4. オシロスコープを使用して 2 本のピンの波形を観察してください。



A. Pin_Test_Write(1u)

B. Pin_Test_Write(0u)

C. CY_SYS_PINS_SET_PIN(Pin_Test__DR, Pin_Test_SHIFT)

D. CY_SYS_PINS_CLEAR_PIN(Pin_Test__DR, Pin_Test_SHIFT)

Figure 66　出力信号波形

Figure 66 に示すように、データレジスタへ直接に書き込むことにより、ピンは API 関数を呼び出す方法より速くトグルできます。

時間効率良く行うためのコーディング技術についてはアプリケーションノート **AN89610 – PSoC™ 4 and PSoC™ 5LP ARM Cortex Code Optimization** を参照してください。

7.7　GPIO 出力イネーブル ロジックの設定

この例は GPIO ピンの出力イネーブル ロジックを設定し、使用する方法を示します。このプロジェクトは PSoC™ 4200, PSoC™ 4200 Bluetooth® LE, PSoC™ 4200M, および PSoC™ 4200L 製品でのみ利用可能です。

1. 2 本のデジタル出力ピンをプロジェクト回路図に配置してください。
2. 各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** オプションを選択してください。

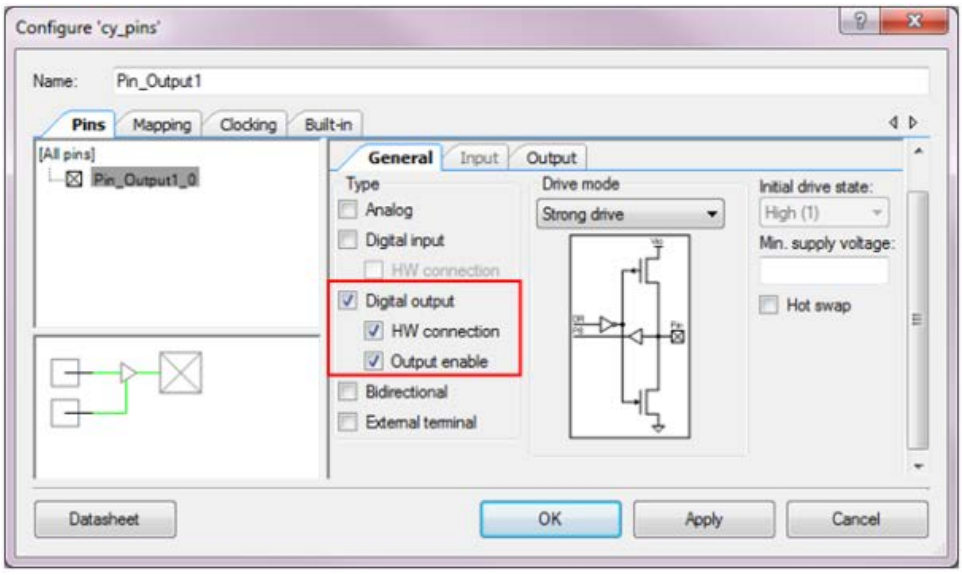


Figure 67　出力イネーブルの選択

3. 制御レジスタを回路図に配置してください。
4. 2 つの出力を持つように制御レジスタを設定してください。

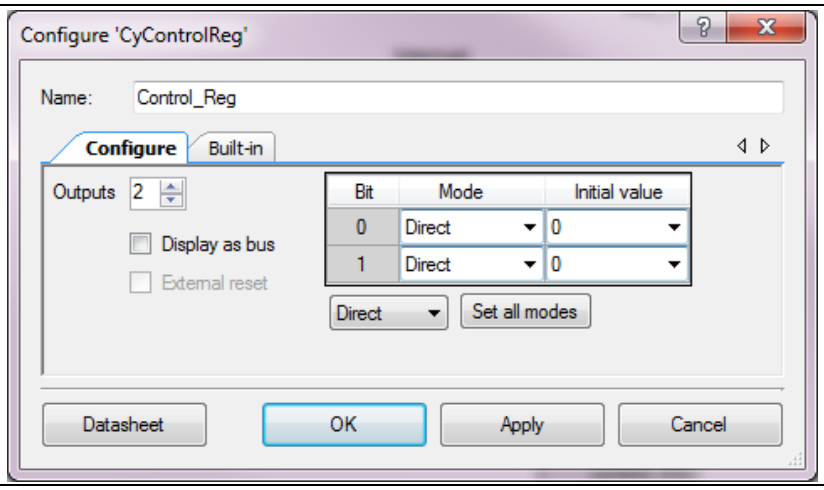


Figure 68　制御レジスタが 2 個の出力を持つように設定

5. 1 つの論理 LOW「0」コンポーネントを追加してください。
6. ロジック LOW をピンに接続して、LED 用のオフチップ コンポーネントを加えてください。

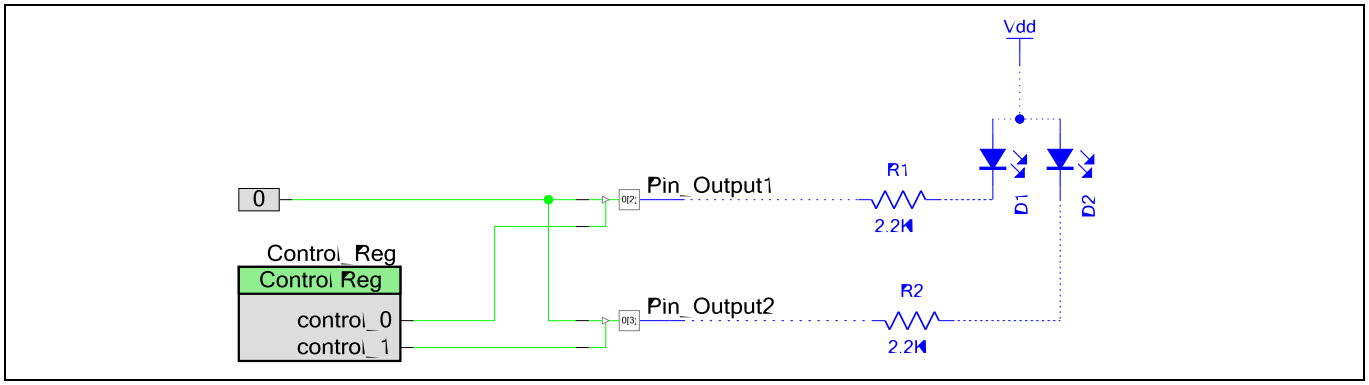


Figure 69　制御レジスタの駆動ピンの出力イネーブル

7. ピンを割り当て、LED に接続してください。
8. *main.c* ファイルに以下のコードを追加してください。

```
uint8 count;


for(;;)
{
   for(count = 0u; count < 4u; count++)
   {
       /* Set Control_Reg Value */
       Control_Reg_Write(count);
```

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

9. プロジェクトをビルドし、PSoC™ 4 デバイスをプログラムしてください。

結果として、2 本のピンの出力は、LED を 3 までカウントさせる Control_Reg の状態によって制御されます。

7.8 ピン割込み

この例では、コンポーネント API 関数を使って同じポートの 2 本のピンから生成された 1 つの割込みを使用する方法を示します。これらの 2 本のピンは 1 つだけの IRQ 端末を使用できます。そのため、割込みソースは ISR で識別する必要があります。

1. プロジェクト回路図の 2 本のピンを配置してください (「Pin_Button」というデジタル入力ピンの 1 本および「Pin_LED」というデジタル出力ピンの 1 本)。
2. Pin_Button のピン数を「2」に、「Drive mode」を「Resistive Pull Up」に、「Interrupt」を「Falling-Edge」に設定してください。これにより、IRQ 端子を表示します。
3. 割込みコンポーネントを irq 端子に接続してください。

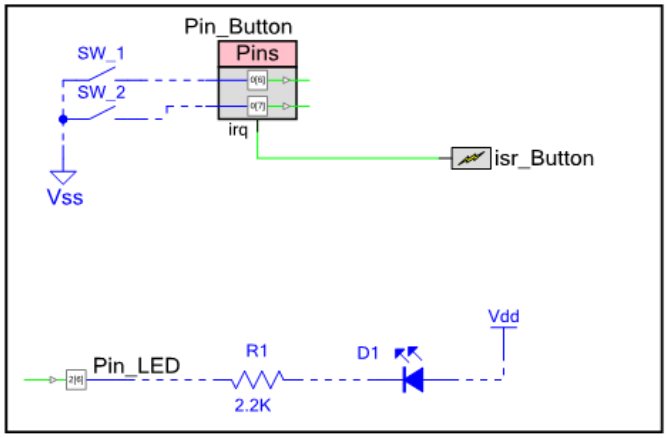


Figure 70 ピン割込みの回路図例

4. .cywdr ウィンドウでピンを割り当ててください。
5. コンポーネント API 関数を使用して、Pin_Button に応じて LED ピンの状態を設定してください。以下の *main.c* コードをコピーしてください。

```
#define LED_ON (0u)
#define LED_OFF (1u)

/* The flag to enter ISR_Button */
uint8 isrFlag = 0u;
```

```
/* The LED state */
uint8 ledState = LED_OFF;

/* ISR for ISR_Button */
CY_ISR(INT_ISR_Button)
{
   /* Set the flag */
   isrFlag = 1u;

   /* Check which pin caused interrupt by reading interrupt status register */
   if(Pin_Button_INTSTAT & (0x01u << Pin_Button_SHIFT))
    {
        /* Triggered by Pin_Button_0 */
        ledState = LED_OFF;
    }
    else
    {
        /* Triggered by Pin_Button_1 */
        ledState = LED_ON;
    }

    /* Clear interrupt */
    Pin_Button_ClearInterrupt();
}

int main()
{
    /* Start Pin ISR */
    isr_Button_StartEx(INT_ISR_Button);

     /* Enable global interrupt */
```

```
    CyGlobalIntEnable;


    for(;;)
    {
        /* Check the flag */
        if(0u != isrFlag)
        {
            /* Clear the flag */
            isrFlag = 0u;

/* Drive the LED with ledState. Led State is updated in ISR */
            Pin_LED_Write(ledState);
        }

        /* Delay 1ms */
        CyDelay(1u);
    }
}
```

main.c コードでは、`CY_ISR(INT_ISR_Button)`はピン割込みのための割込みサービスルーチンです。

6. プロジェクトをビルドし、PSoC™ 4 デバイスをプログラムしてください。

その結果は、ボタンを離す時を除いて、Pin_Button_0 に接続されているボタンを押すと LED がオフになり、Pin_Button_1 に接続されているボタンを押すと LED がオンになります。(ボタンのスイッチバウンスにより、ボタンを 1 回押すと、いくつかの割込みが発生する場合があります。詳細は、AN60024 – Switch debouncer and glitch filter with PSoC™ 3, PSoC™ 4, and PSoC™ 5LP を参照してください)

main.c コードでは、Pin_Button_INTSTAT および Pin_Button_SHIFT はピンコンポーネントにより提供される関数と定数マクロです。これらは、どのピンが割込みを発生させるか検出するために使用されます。

`Pin_Button_ClearInterrupt()`関数は、割込みステータスレジスタをクリアします。

Note: *専用割込みを持たないポートもあります。上位ポートでは、共通の割込み信号が生成されます。*

対応するデバイスの **architecture テクニカルリファレンスマニュアル** (TRM) の「Interrupts」章を参照してください。

割込みと割込みハンドラの書き込みの詳細情報は **AN90799 – PSoC™ 4 Interrupts** を参照してください。

7.9 ファームウェアでの GPIO 割込みの設定

GPIO 割込みは、割込み設定レジスタ内の 2 ビットへ書き込むことによって動的に設定されます。

- PSoC™ 4000 用: GPIO_PRTx_INTR_CFG[2y+1:2y]
- その他の PSoC™ 4 製品用: PRTx_INTCFG[2y+1:2y]

以下の表において、ここで「x」はポート番号で、「y」はピン番号です。ピン割込みを有効または無効にするために設定を随時変更できます。

Table 9 GPIO 割込みのタイプおよびビット設定

PRTx_INTCFG [2y+1:2y]	エッジタイプ	説明
0	無効	割込みが無効
1	立ち上りエッジ	立ち上りエッジでトリガー
2	立ち下りエッジ	立ち下りエッジでトリガー
3	両方のエッジ	どちらのエッジでもトリガー

この例では、Pin_Button は、立ち上りエッジで発生する割込みとして設定されます。割込みが発生すると、それは立ち下りエッジで発生する割込みとして設定されます。割込みがトリガーされると、LED はトグルされます。

1. デジタル入力ピンとデジタル出力ピンをプロジェクト回路図に配置してください。オフチップ コンポーネントを LED とボタン用に追加してください。

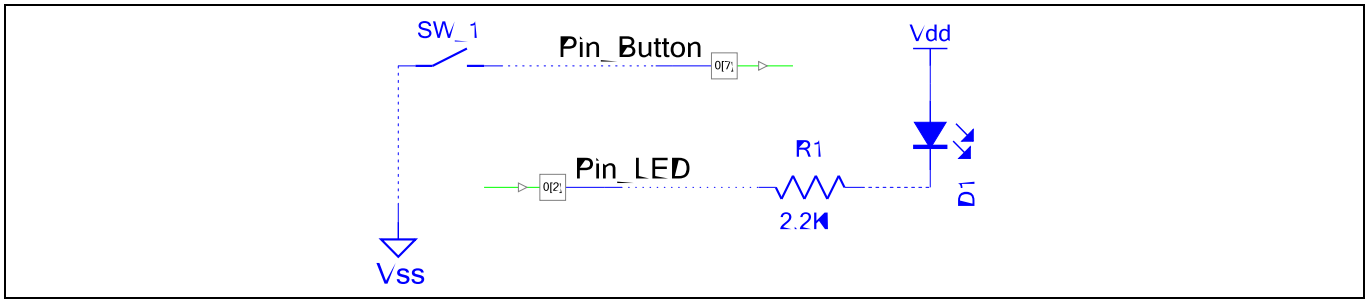


Figure 71 サンプル回路図

2. cydwr ウィンドウでピンを Pin_Button と Pin_LED に割り当ててください。
3. Pin_Button を抵抗プルアップピンとして設定し、1 つのボタンに接続してください。
4. Pin_LED をストロングドライブ ピンとして設定し、外部 LED に接続してください。
5. *main.c* ファイルに以下のコードを追加してください。このプロジェクトでは、デバイスのレジスタ名の代わりに、PSoC™ Creator が提供する Pin_Button__INTCFG (*cyfitter.h* ファイルに保存される) を割込みの設定に使用することに注意してください。選択したデバイスでレジスタの正確な名称を考慮する必要がありません。これにより、コードを変更せずプロジェクトを異なった PSoC™ 4 デバイスに移植できます。

```
#define INTERRUPT_MASK 0x03

#define RISING_EDGE 0x01

#define FALLING_EDGE 0x02
```

```
int main()
{
    /* Variable to save temporary data */
    uint32 regVal = 0x00u;

    /* Flag to switch interrupt type */
    uint8 edgeFlag = 0x00u;

    for(;;)
    {
/* Get value of port interrupt configuration register */
regVal = CY_GET_REG32(Pin_Button__INTCFG);

/* Clear the configuration bits for the Pin_Button. Pin_Button_SHIFT is multiplied
by 2 as two bits of the interrupt configuration register sets the configuration for
one pin */
             regVal &= ~(INTERRUPT_MASK << (Pin_Button_SHIFT * 2));

if(edgeFlag)
{
/* Set P0[7] to GPIO interrupt rising-edge trigger. Pin_Button_SHIFT is multiplied
by 2 as two bits of the interrupt configuration register sets the configuration for
one pin */
        CY_SET_REG32(Pin_Button__INTCFG, regVal | (RISING_EDGE << (Pin_Button_SHIFT
* 2)));
}
else
{
/* Set P0[7] to GPIO interrupt falling-edge trigger. Pin_Button_SHIFT is multiplied
by 2 as two bits of the interrupt configuration register sets the configuration for
one pin */
      CY_SET_REG32(Pin_Button__INTCFG, regVal | (FALLING_EDGE << (Pin_Button_SHIFT
* 2)));
}
```

```
/* Toggle edgeFlag */
edgeFlag ^= 0x01u;

/* Wait for Interrupt */
while(!(CY_GET_REG32(Pin_Button__INTSTAT) & (0x01u << Pin_Button_SHIFT))) {;}

/* Clear interrupt */
CY_SET_REG32(Pin_Button__INTSTAT, (0x01u << Pin_Button_SHIFT));

/* Toggle LED */
Pin_LED_Write(~Pin_LED_Read());
    }
}
```

6. プロジェクトをビルドし、PSoC™ 4 デバイスをプログラムしてください。

ボタンを押すか離すと、LED はトグルします。ボタンを押すと、割込みは立ち下りエッジでトリガーされ、ボタンを離すと割込みは立ち上りエッジでトリガーされます。

PSoC™ 4 architecture TRM は、ブロック図と機能説明を含む、GPIO 割込みの詳細を記載します。他に **AN90799 – PSoC™ 4 Interrupts** アプリケーションノートも参照してください。

7.10 GPIO でのアナログとデジタルの両方の使用

この例では、ピンをアナログとデジタルの機能に設定し使用する方法を示します。この例では、出力ピンは IDAC とファームウェアにより交互に制御されます。ファームウェアにより制御される時、LED が点滅します。IDAC により制御される時、LED が徐々に点灯します。

このような多重化は、単一のピンのアナログ機能とデジタル機能を利用したい場合は有用です。また、これにより、デザインに使用する GPIO ピンの数を削減できます。

ファームウェアではなく、ハードウェア接続を使用してデジタル出力も制御できます。プロジェクトの必要な修正点については、このセクションの終わりの説明を参照してください。

ピンの信号のソースを設定するためには、HSIOM_PORT_SELx レジスタを更新します。前の例のように、PSoC™ 4 デバイスファミリのすべてに容易に移植するために、このプロジェクトはピン コンポーネントで定義されるレジスタ名を使用します。

これらのステップに従って回路図とファームウェアを作成してください。

1. アナログピンと Current DAC を回路図に配置してください。
2. ピンコンポーネントを物理ピンに割り当ててください (この例では P0[2]を使用)。
3. **Analog** と **Digital Output** の両方を選択してピンを設定してください。

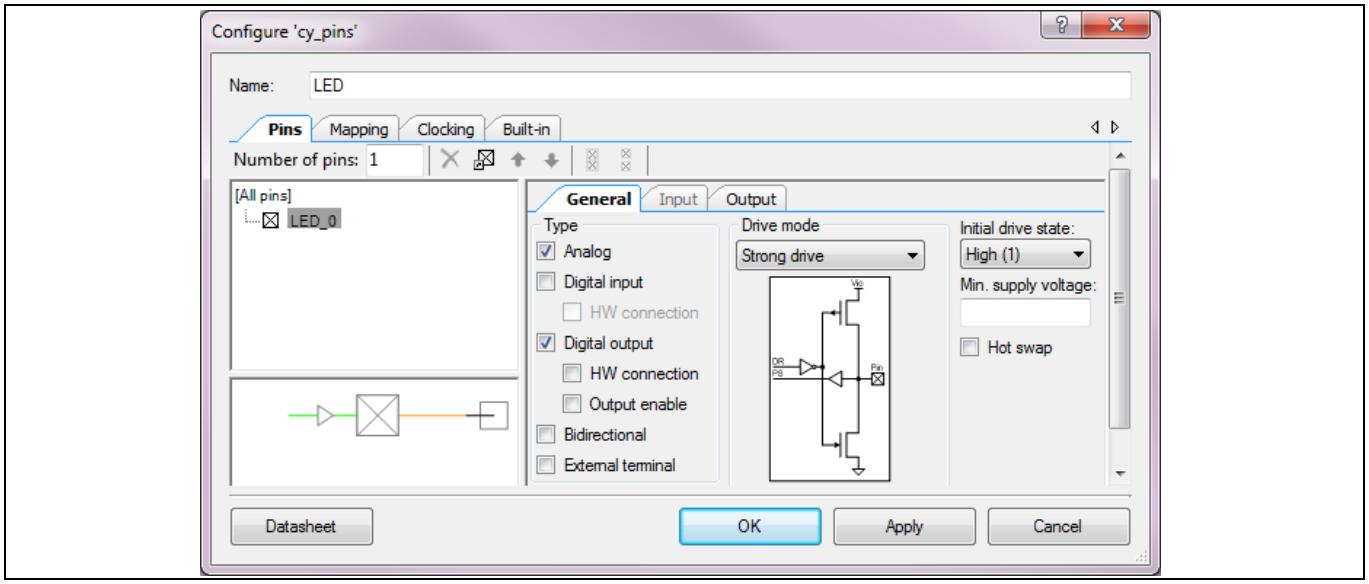


Figure 72　アナログとデジタルの両方として設定された LED ピン

4. **Figure 73** に示すように IDAC の **Polarity** を **Negative (Sink)** として設定してください。**Figure 74** に示すように IDAC をアナログ端子に接続してください。

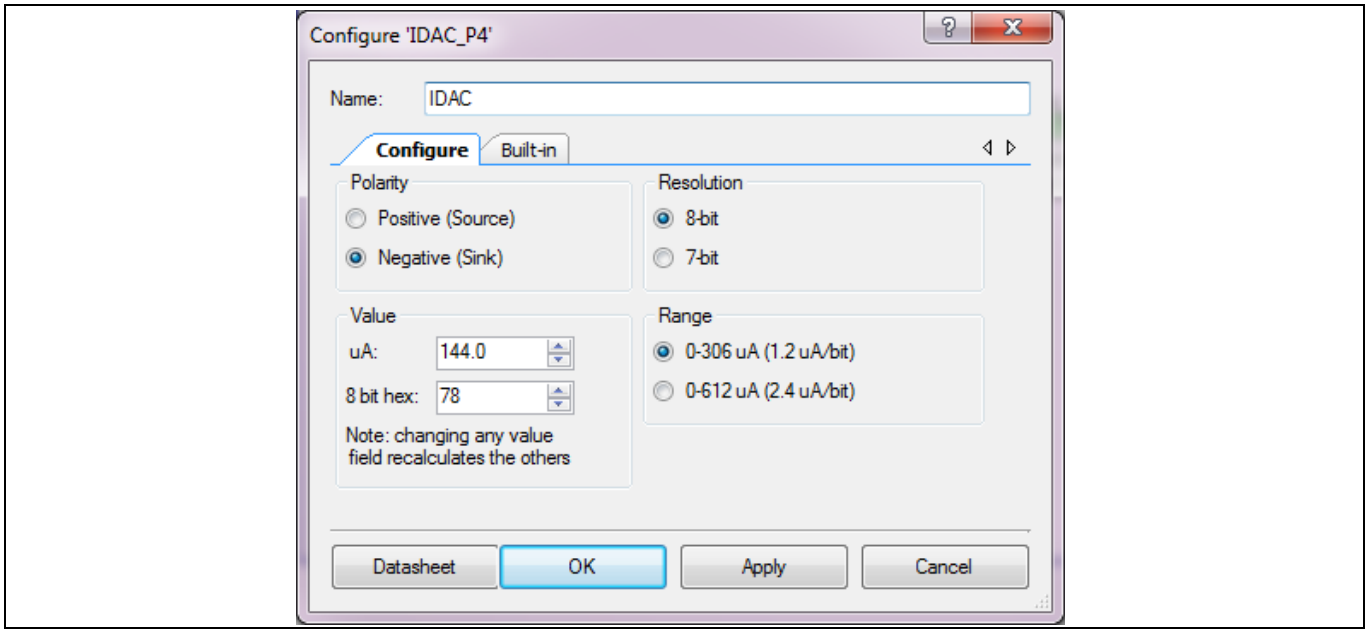


Figure 73　IDAC 設定

5. プロジェクトを構築して必要な API を作成してください。

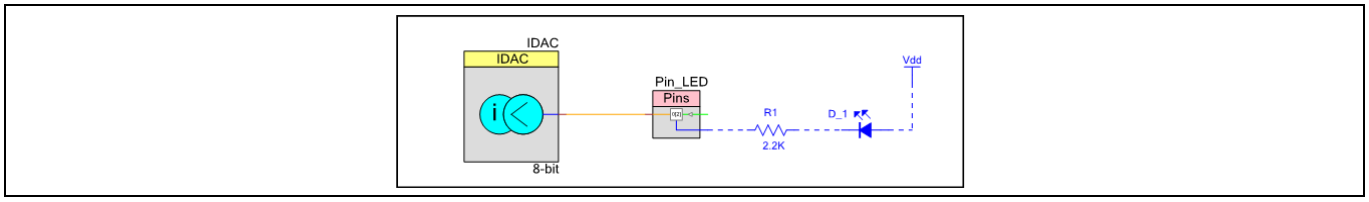


Figure 74　PSoC™ Creator のアナログおよびデジタルスイッチング スキームの回路図

6. *main.c* ファイルに次のコードを追加し、プロジェクトを再度ビルドしてください。生成された HEX ファイルを使ってデバイスをプログラムしてください。このコードでは、ピンコンポーネントで定義されるマクロと *Cyfitter.h* が使用されることに注意してください。

```
#define HSIOM_SW_GPIO 0x00

#define HSIOM_AMUX_BUS_A 0x06


int main()

{

        uint32 i = 0u;

        uint32 regVal = 0x00u;


        /* Disable Input Buffer */

        Pin_LED_INP_DIS |= (0x01u << Pin_LED_SHIFT);


        /* Start IDAC */

        IDAC_Start();


        for(;;)

        {

            /* Get the current value of HSIOM_PORT_SEL0 register */

            regVal = CY_GET_REG32(Pin_LED__0__HSIOM);

            regVal &= ~Pin_LED__0__HSIOM_MASK;


            /* Set LED Pin as GPIO controlled by firmware */

            regVal = CY_SET_REG32(Pin_LED__0__HSIOM, regVal |(HSIOM_SW_GPIO <<
Pin_LED__0__HSIOM_SHIFT));


            /* Set LED Pin to Strong Drive Mode */

            Pin_LED_SetDriveMode(Pin_LED_DM_STRONG);


            for(i= 0u; i < 5u; i++)

            {

              /* Toggle LED with 100-ms delay */
```

```
            Pin_LED_Write(0u);
            CyDelay(100u);
            Pin_LED_Write(1u);
            CyDelay(100u);
          }

          /* Get the current value of HSIOM_PORT_SEL0 register */
          regVal = CY_GET_REG32(Pin_LED__0__HSIOM);
          regVal &= ~Pin_LED__0__HSIOM_MASK;

          /* Connect LED Pin to AMUXBUS-A */
          CY_SET_REG32(Pin_LED__0__HSIOM, regVal | (HSIOM_AMUX_BUS_A <<
Pin_LED__0__HSIOM_SHIFT));

          /* Set LED Pin to High Impedance-Analog Drive Mode */
          Pin_LED_SetDriveMode(Pin_LED_DM_ALG_HIZ);

          for(i = 0u; i < 0x7fu; i++)
          {
            /* Adjust LED brightness */
              IDAC_SetValue(i);

            /* Delay 20 ms */
            CyDelay(20u);
          }
      }
}
```

この結果はファームウェアと IDAC が交代に制御する出力です。

このプロジェクトでは、ファームウェアの制御からデジタル出力へのハードウェア接続に簡単に変えられます。そうするためには、ステップ 3 で、ピン設定 ウィンドウの **HW connection** を有効にするだけです。その後、デジタルリソースをピンに配線できます。このデジタルリソースをピン出力として選択するためには、HSIOM_PORT_SEL レジスタを使って、そのピンを DSI 制御の GPIO またはピン固有のデジタルリソース接続として設定してください。詳細は **PSoC™ 4 architecture TRM** を参照してください。

7.11 より大きな駆動/シンク電流のためのピンの連動

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)。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。このプロジェクトは PSoC™ 4200, PSoC™ 42xx_BL, PSoC™ 4200M および PSoC™ 4200L 製品でのみ利用可能であることに注意してください。

1. PWM (TCPWM モード) およびクロックコンポーネントを回路図に配置し、設定してください。
2. シングルのデジタル出力ピンコンポーネントを配置してください。
3. コンポーネントを接続してください。

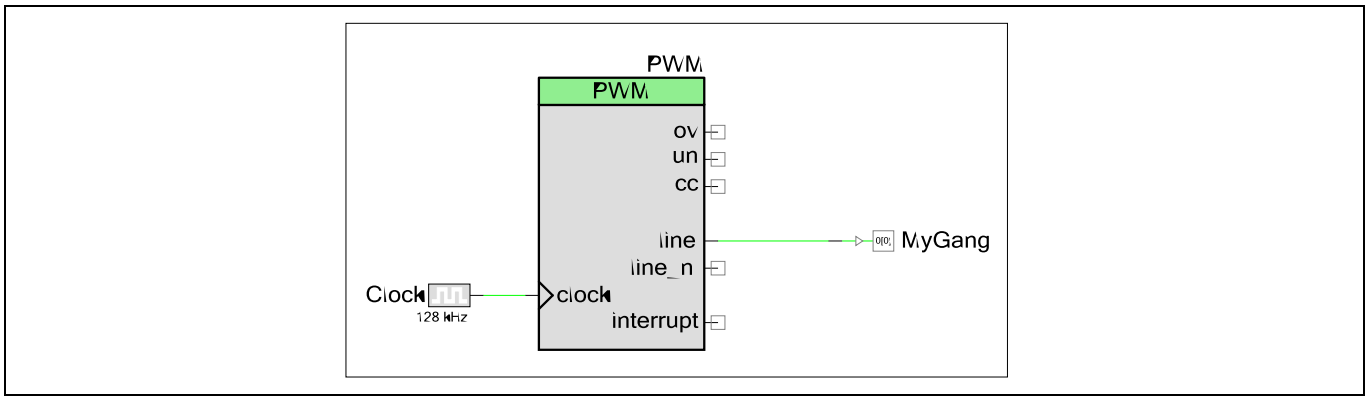


Figure 75 PWM は単一のピンに駆動

4. ピン設定ダイアログを開き、ピンの数を適切に設定してください。この例では、4 つの GPIO ピンを使います。**Output Mode** を **Single-Sync** および **Out Clock** を **External** に設定してください。

Note: 各ピンでの異なる出力信号遅延を回避するために出力を同期化してください。

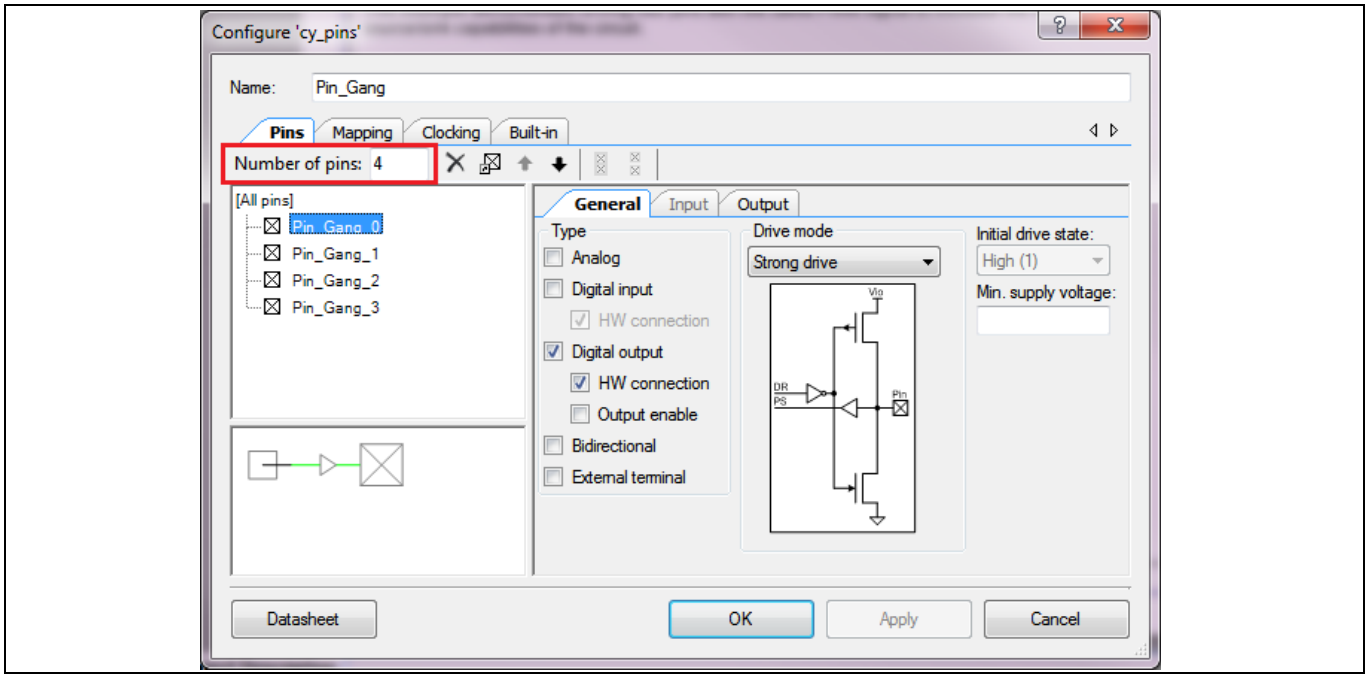


Figure 76 コンポーネントで複数のピンを設定

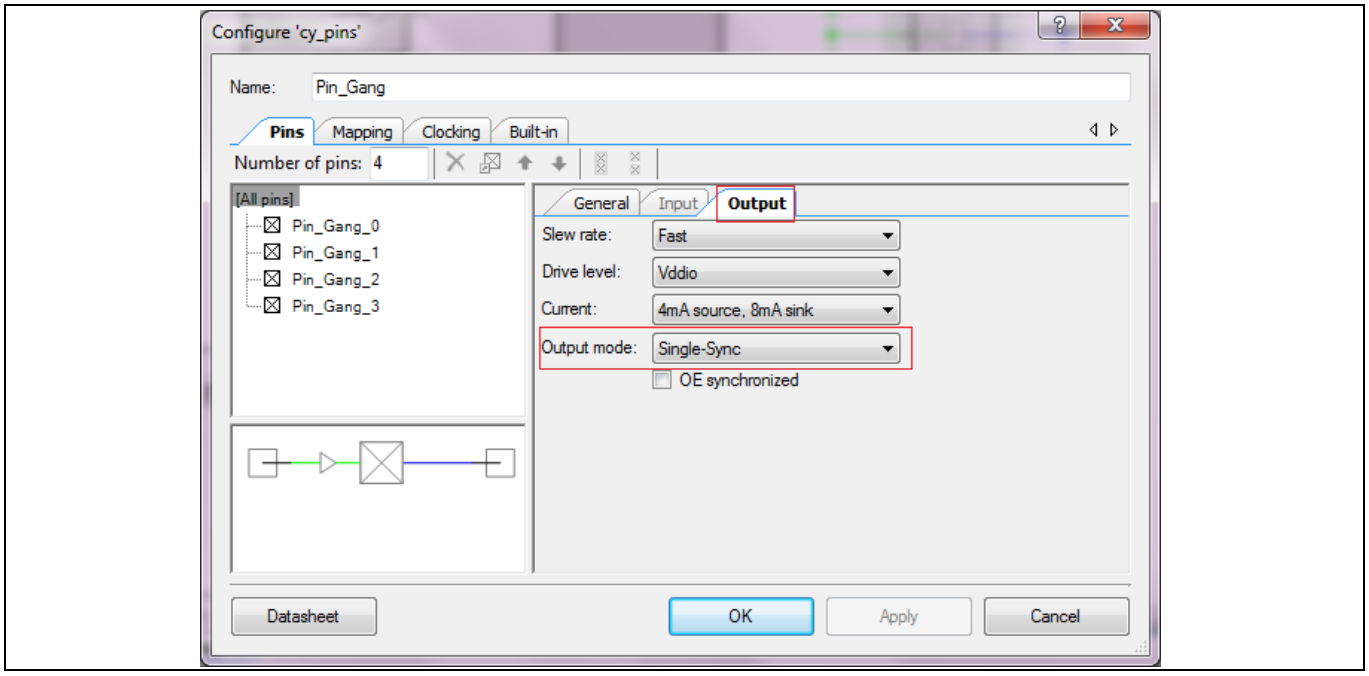


Figure 77　出力モード設定

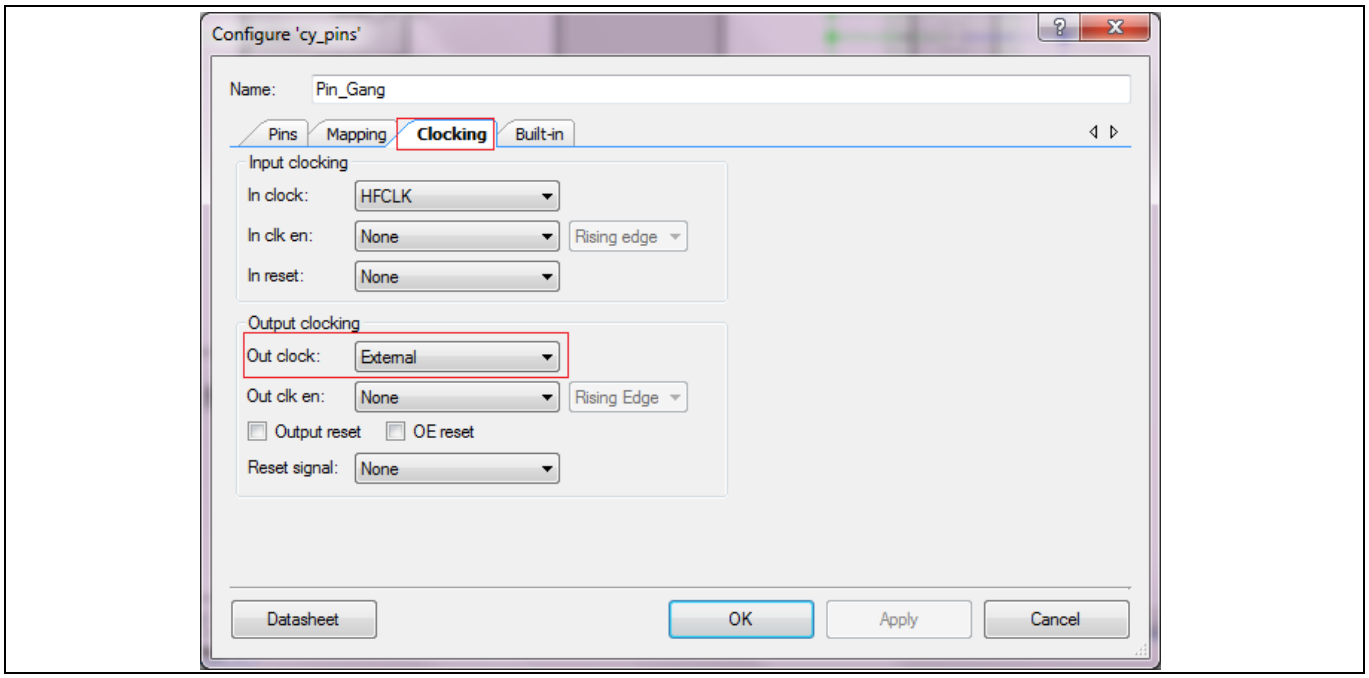


Figure 78　出力クロック設定

5. (任意) より簡単な PCB ルーティングのために、ピンマッピングを **Contiguous** に設定してください。

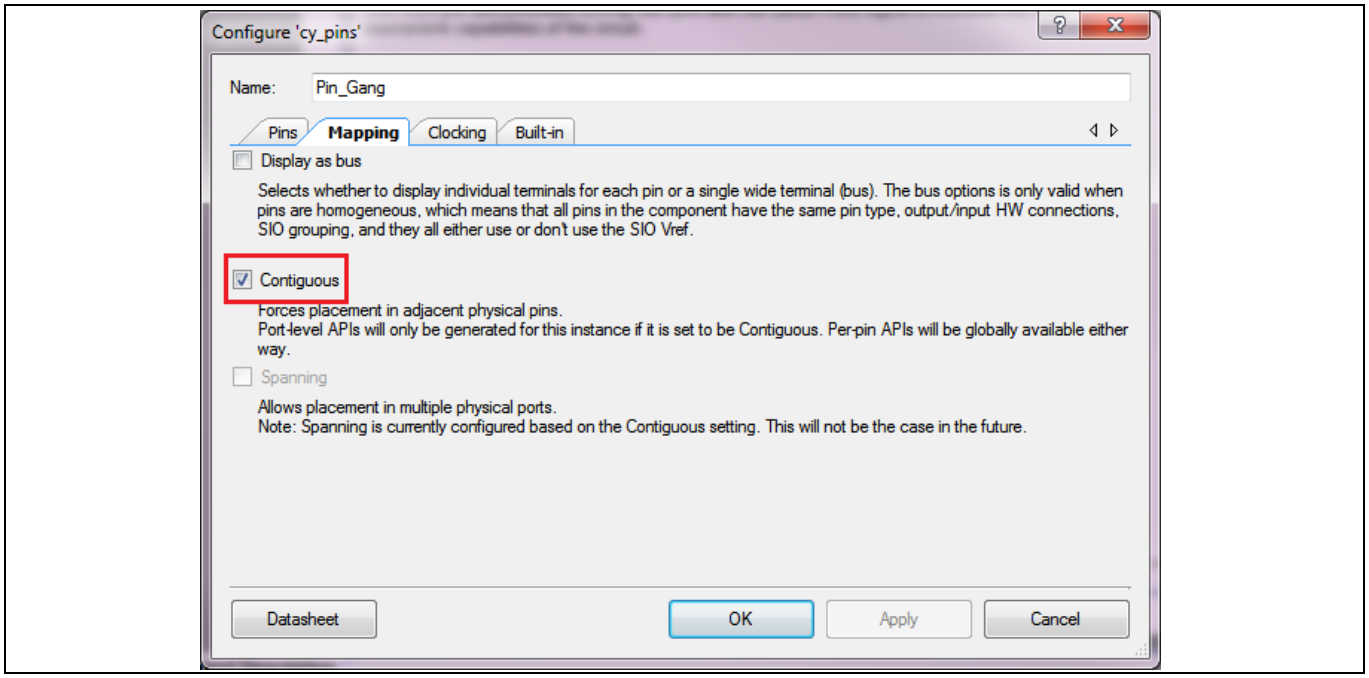


Figure 79　隣接マップを有効化

6. ピンコンポーネントを物理ピンに割り当ててください。
7. Sync コンポーネントを配置し、Sync コンポーネントを介して信号ソース (この例では PWM) を各ピン端子に接続してください。他のクロックコンポーネントを配置し、そのソースを高周波数クロック (HFCLK) に設定してください。ピンコンポーネントの out_clk 端子および Sync コンポーネントのクロック端子を HFCLK に接続してください。

ピン信号の遅延の差を低減するために、高周波数の同期クロックを選択することが重要です。Sync コンポーネントは 1 つのクロックドメインから他方にわたる信号を同期するために必要です。この場合、PWM 出力はクロック (1kHz) ドメインから HFCLK に移ります。

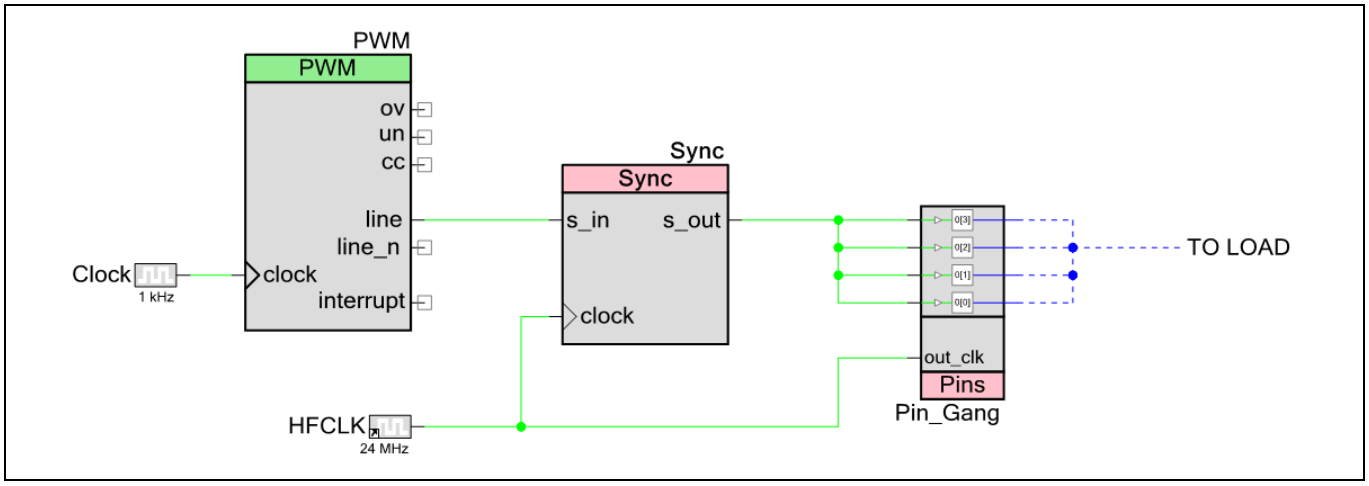


Figure 80　PWM は 4 つのピンを駆動

8. プロジェクトをビルドし、PSoC™ 4 デバイスをプログラムしてください。
9. PWM の出力は、4 本すべての GPIO で駆動されます。必要な場合、これらのピンは PCB の外部で短絡および外部回路に接続できます。

7.12 ディープスリープにおける制御レジスタの取り扱い

この例では、低消費電力モードで出力のグリッチを避けるために GPIO ピンを凍結する方法について示します。例として、制御レジスタがピンを駆動する場合を考えてください。デバイスがディープスリープ モードに入ると、すべての I/O はユーザーの介入なしで凍結されます。デバイスがウェイクアップすると、これらの I/O は自動的に元の設定に復元されます。しかし、制御レジスタはディープスリープ モードでデータを失います。I/O が凍結状態を解除する前に、それを復元する必要があります。そうしないと、出力にグリッチが発生します。PSoC™ 4 は `CySysPmFreezeIo()`および `CySysPmUnfreezeIo()` API 関数を使用して GPIO を凍結および解凍するための代替制御を提供します。以下のステップに従って、PSoC™ Creator プロジェクトを作成します。このプロジェクトは PSoC™ 4200, PSoC™ 42xx_BL, PSoC™ 4200M および PSoC™ 4200L でのみ利用可能であることに注意してください。

1. 1 つのデジタル入力ピン, 2 つのデジタル出力ピン, クロック, 制御レジスタ, および割込みコンポーネントを回路図に配置してください。
2. 以下の表のように各コンポーネントを設定してください。**Figure 81** に示すように、コンポーネントを接続してください。

Table 10 コンポーネントの設定

コンポーネント	名称	設定
デジタル入力ピン	Pin_Button	駆動モード: 抵抗プルアップ 割込み: 立ち上りエッジ
デジタル出力ピン	Pin_Clock	デフォルト設定
デジタル出力ピン	Pin_CtrlReg	デフォルト設定
クロック	SYSCLK	ソース: SYSCLK
割込み	isr_Buttton	デフォルト設定
コントロールレジスタ	Ctrl_Reg	出力: 1 初期値: 1

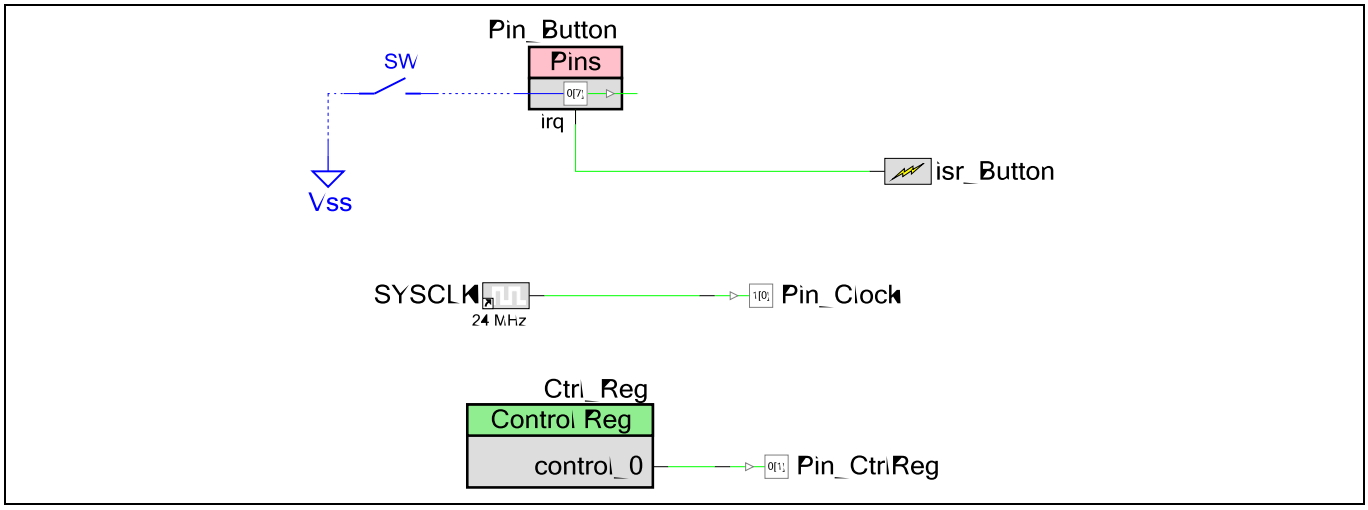


Figure 81 ディープスリープ終了時のグリッチを回避

3. *main.c* ファイルに以下のコードを追加してください。

```
/* Set FREEZE_IO to 0x01 to avoid glitch by enabling the GPIO freeze */
```

```
/* else set it to 0 */
#define FREEZE_IO 0x01

/* The flag to enter ISR */
uint8 isrFlag = 0u;
CY_ISR(ISR_Handle)
{
        /* Set the flag */
        isrFlag = 1u;

        /* Clear pin interrupt */
        Pin_Button_ClearInterrupt();
}

int main()
{
        /* This variable is used as backup for Control register value */
        uint8 ctrlRegVal = 0u;

        /* Clear the flag */
        isrFlag = 0u;

        /* Start the ISR */
        isr_Button_StartEx(ISR_Handle);

        CyGlobalIntEnable;

        /* Set Control register output as high */
        Ctrl_Reg_Write(1u);

        for(;;)
        {
```

```
                /* If freeze flag is set */
                if(0u != isrFlag)
                {
                        /* Clear isr flag set in GPIO Interrupt Handler */
                        isrFlag = 0u;

                        /* Rewrite the value */
                        Ctrl_Reg_Write(ctrlRegVal);

                        #if(FREEZE_IO)
                                /* Unfreeze I/O */
                                CySysPmUnfreezeIo();
                        #endif
                }

                /* Delay 200us */
                CyDelayUs(200u);

                /* Store the value of Control register */
                ctrlRegVal = Ctrl_Reg_Read();

                #if(FREEZE_IO)
                        /* Freeze I/O */
                        CySysPmFreezeIo();
                #endif

                /* Enter Deep-Sleep mode */
                CySysPmDeepSleep();
        }
}
```

フリーズ オプションを無効にするために、FREEZE_IO を「0」に設定します。デバイスをビルドし、プログラムします。**Figure 82** に示すように、ボタンを押して離す時、グリッチは Pin_CtrlReg で見ること

ができます。フリーズ オプションを有効にするために、FREEZE_IO を「1」に設定してください。デバイスをビルドし、プログラムしてください。**Figure 83** に示すように、この場合、I/O はフリーズされ、グリッチは観察されません。

Note: *専用割込みを持たないポートもあります。上位ポートでは、共通の割込み信号が生成されます。*

デバイスに対応した **architecture テクニカルリファレンスマニュアル** (TRM)の「Interrupts」章を参照してください。

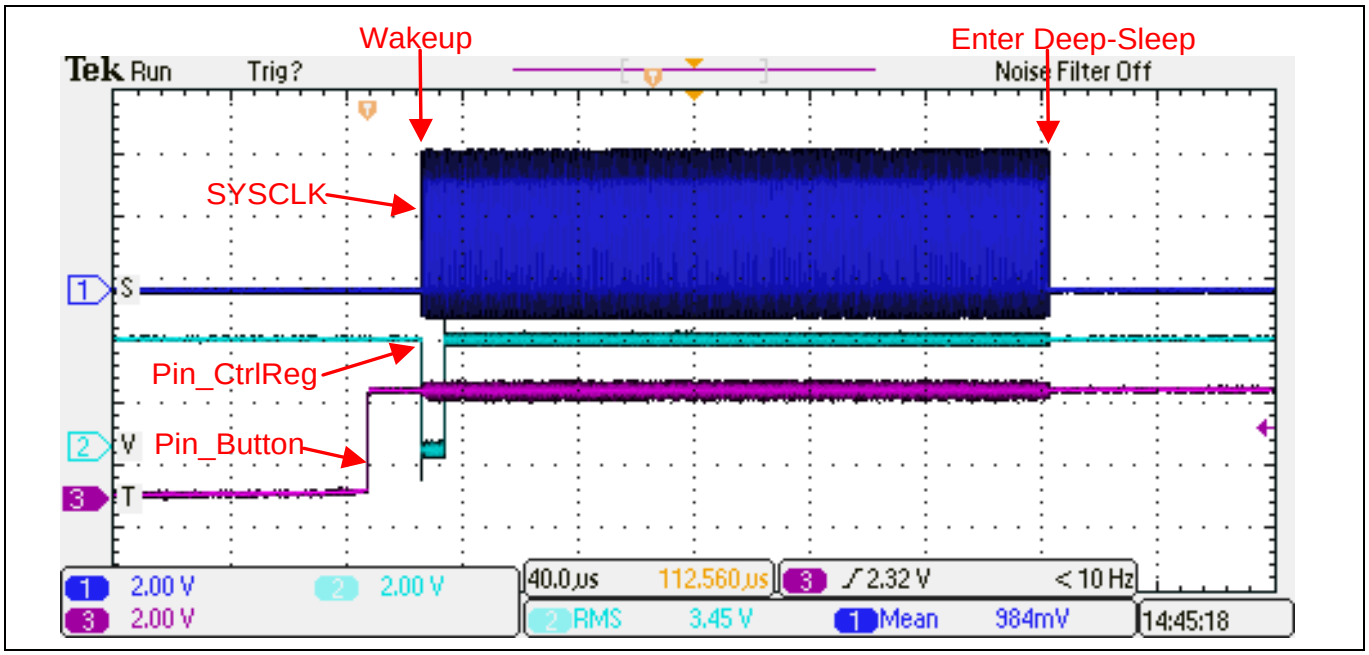


Figure 82　出力信号波形 (I/O フリーズなし)

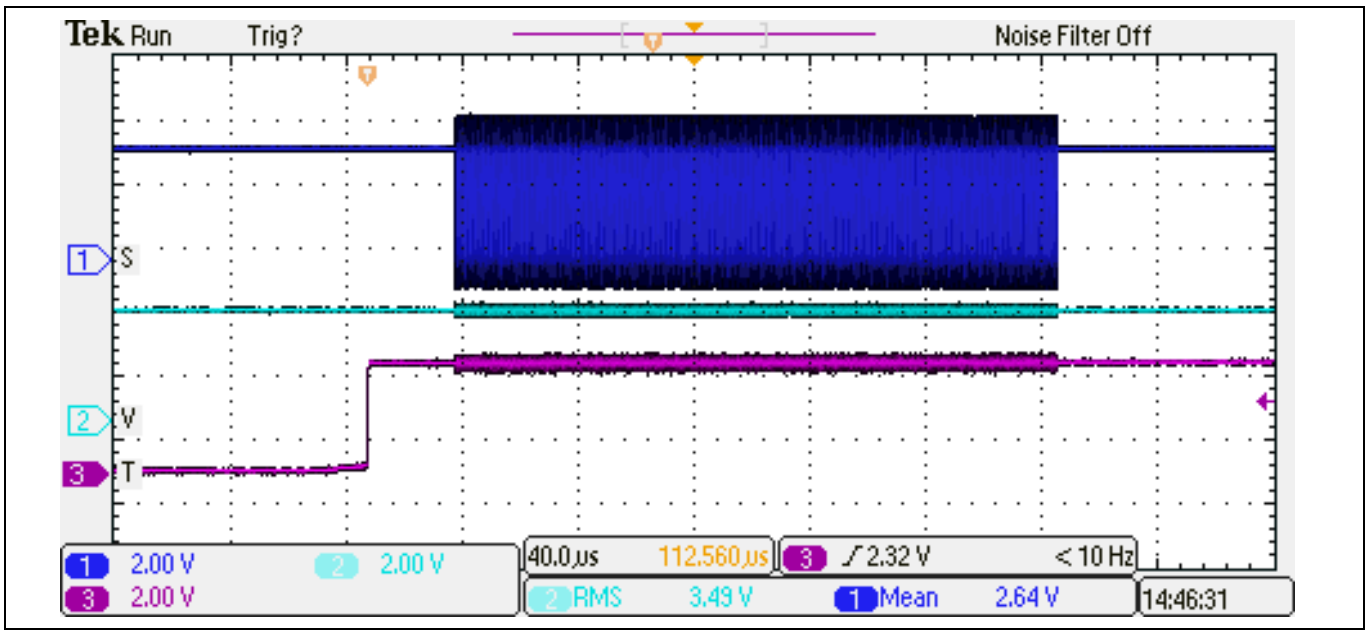


Figure 83　出力信号波形 (I/O フリーズあり)

8 ModusToolbox™での GPIO のヒントおよびコツ

ここでは、ModusToolbox™を使用するときに GPIO ピンを使用する方法の実際的な例を示します。

8.1 入力の読み出しと出力への書き込み

入力からの読み出しと出力への書き込みは、**GPIO PDL 関数**を使用して行われます。

PSoC™ 4 のサンプルコードには、**ModusToolbox™の新しいアプリケーションウィザード**からアクセスできます。サンプルコード CE231741 は、PSoC™ 4 汎用入力/出力 (GPIO) ピンを使用して割込みを設定、読み出し、書き込み、および生成する複数の方法を示します。

8.2 ピン割込み

ModusToolbox™の GPIO 割込みプロジェクトのサンプルコードについては、**mtb-example-PSoC™4-gpio-interrupt code example** を参照してください。このサンプルコードには、**ModusToolbox™の新しいアプリケーションウィザード**からもアクセスできます。

8.3 その他のサンプルコード

幅広いトピックをカバーする ModusToolbox™で開発された PSoC™ 4 のその他のサンプルコードは、**Infineon GitHub** または ModusToolbox™の中にあります。サンプルコードへのアクセスの詳細については、**ModusToolbox™サンプルコード**を参照してください。

9 関連アプリケーションノート

- **AN79953 – Getting started with PSoC™ 4**
- **AN86233 – PSoC™ 4 low-power modes and power reduction techniques**
- AN60024 – Switch debouncer and glitch filter with PSoC™ 3, PSoC™ 4, and PSoC™ 5LP
- AN72382 – Using PSoC™ 3 and PSoC™ 5LP GPIO pins
- **AN90799 – PSoC™ 4 interrupts**
- **AN2094 – PSoC™ 1 getting started with GPIO**
- **AN89610 - PSoC™ 4 and PSoC™ 5LP Arm® Cortex® code optimization**

10 PSoC™ 4 GPIO と PSoC™ 1, PSoC™ 3, および PSoC™ 5LP GPIO との比較

PSoC™ 4 GPIO は、PSoC™ 1, PSoC™ 3, および PSoC™ 5LP の GPIO とは異なります。詳細については、以下の表を参照してください。

Table 11 PSoC™ 4 GPIO と PSoC™ 1, PSoC™ 3, および PSoC™ 5LP GPIO

GPIO 機能	PSoC™ 1	PSoC™ 3	PSoC™ 4	PSoC™ 5LP
CAPSENCE™	√	√	√	√
LCD セグメント駆動	√	√	√ *	√
8 つの駆動モード	√	√	√	√
POR 状態設定	×	√	×	√
個別のポート DR とポート PS	×	√	√	√
入力/出力の同期	×	Bus_clk	HFCLK, 外部 *	Bus_clk

* PSoC™ 4000 では使用できません

11 PSoC™ 4 開発ボード

このアプリケーションノートで提供される PSoC™ Creator プロジェクトは、次のインフィニオン開発ボードでテストできます。

デバイスファミリ	開発ボード
PSoC™ 4000	**CY8CKIT-040 PSoC™ 4000 pioneer development kit**
PSoC™ 4000S / 4100S	**CY8CKIT-041 PSOC™ 4 S-Series pioneer kit**
PSoC™ 4100S Plus	**CY8CKIT-149 PSoC™ 4100S Plus prototyping kit**
PSoC™ 4100PS	**CY8CKIT-147 PSoC™ 4100PS prototyping kit**
PSoC™ 4200 / PSoC™ 4100	**CY8CKIT-042 PSoC™ 4 pioneer kit** **PSoC™ 4 CY8CKIT-049 4xxx prototyping kits**
PSoC™ 42x7_BL	**CY8CKIT-042-BLE Bluetooth® Low Energy pioneer kit**
PSoC™ 4200M	**CY8CKIT-044 PSoC™ 4 M-Series pioneer Kit**
PSoC™ 4200L	**CY8CKIT-046 PSoC™ 4 L-Series pioneer kit**
PSoC™ 4200DS	**PSoC™4 CY8CKIT-146 4200DS prototyping kits**
PSoC™ 4700S	**CY8CKIT-148 PSoC™ 4700S inductive sensing evaluation kit**
PSoC™ analog coprocessor	**CY8CKIT-048 PSoC™ analog coprocessor pioneer kit**

改訂履歴

版数	発行日	変更内容
**	2015-07-06	これは英語版 001-86439 Rev. *A を翻訳した日本語版 001-97886 Rev. **です。
*A	2016-05-10	これは英語版 001-86439 Rev. *D を翻訳した日本語版 001-97886 Rev. *A です。
*B	2017-07-17	ロゴと著作権を更新しました。
*C	2019-03-01	これは英語版 001-86439 Rev. *I を翻訳した日本語版 001-97886 Rev. *C です。
*D	2019-08-19	内容の変更なし。Sunset review を実施。
*E	2021-02-15	これは英語版 001-86439 Rev. *K を翻訳した日本語版 001-97886 Rev. *E です。
*F	2023-08-08	これは英語版 001-86439 Rev. *L を翻訳した日本語版 001-97886 Rev. *F です。

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2023-08-08
Published by
Infineon Technologies AG
81726 Munich, Germany

© 2023 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?
Go to www.infineon.com/support

Document reference
001-97886 Rev. *F

重要事項

本手引書に記載された本製品の使用に関する手引きとして提供されるものであり、いかなる場合も、本製品における特定の機能性能や品質について保証するものではありません。本製品の使用の前に、当該手引書の受領者は実際の使用環境の下であらゆる本製品の機能及びその他本手引書に記された一切の技術的情報について確認する義務が有ります。インフィニオンテクノロジーズはここに当該手引書内で記される情報につき、第三者の知的所有権の不侵害の保証を含むがこれに限らず、あらゆる種類の一切の保証および責任を否定いたします。

本文書に含まれるデータは、技術的訓練を受けた従業員のみを対象としています。本製品の対象用途への適合性、およびこれら用途に関連して本文書に記載された製品情報の完全性についての評価は、お客様の技術部門の責任にて実施してください。

警告事項

技術的要件に伴い、製品には危険物質が含まれる可能性があります。当該種別の詳細については、インフィニオンの最寄りの営業所までお問い合わせください。
インフィニオンの正式代表者が署名した書面を通じ、インフィニオンによる明示の承認が存在する場合を除き、インフィニオンの製品は、当該製品の障害またはその使用に関する一切の結果が、合理的に人的傷害を招く恐れのある一切の用途に使用することはできないこと予めご了承ください。