# PSOC™ 4 MCU の低消費電力モードおよび消費電力低減技術

## 本書について

### 適用範囲と目的

本アプリケーションノートは、重要な機能を保持しながら超低消費電力レベルで動作するための、PSOC™ 4 MCU の低消費電力モードと機能の使用方法について説明します。5 つの消費電力モードを含み、PSOC™ Creator 電力管理機能、および他の省電力技術や考慮事項が本書の主なトピックです。また、本書には 3 つの PSOC™ Creator 用のサンプルプロジェクトも含まれており、低消費電力プログラミングの様々な面を明らかにします。

*注: PSOC™ Creator は一部の PSOC™ 4 デバイスをサポートします。ModusToolbox™ ソフトウェアは、最新の PSOC™ 4 デバイスをサポートしており、推奨ツールとなっています。サポートされているデバイスは、4000S, 4100S, 4100S Plus, 4100S Plus 256 KB, 4100S Max, 4000T, および 4500S が含まれます。*

### 対象者

このアプリケーションノートは、PSOC™ 4 MCU デバイスを使用するすべての人を対象とします。

# 目次


**本書について** . . . 1

**目次** . . . 2

**1 はじめに** . . . 4

**2 消費電力モードの概要** . . . 5

**3 低消費電力モードの詳細** . . . 7

3.1 スリープモード . . . 7

3.1.1 スリープモードのウェイクアップソース . . . 7

3.1.2 スリープモードの遷移 . . . 8

3.1.3 スリープモードのユースケース . . . 8

3.2 ディープスリープモード . . . 8

3.2.1 ディープスリープモードのウェイクアップソース . . . 8

3.2.2 ディープスリープモードの遷移 . . . 8

3.2.3 ペリフェラルディープスリープ設定 . . . 9

3.2.4 ディープスリープモードのユースケース . . . 9

3.3 PSOC™ Creator でのハイバネートモード . . . 9

3.3.1 ハイバネートモードのウェイクアップソース . . . 9

3.3.2 ハイバネートモードの遷移 . . . 10

3.3.3 ハイバネートモードのユースケース . . . 10

3.4 PSOC™ Creator でのストップモード . . . 10

3.4.1 ストップモードのウェイクアップソース . . . 10

3.4.2 ストップモードの遷移 . . . 11

3.4.3 ストップモードのユースケース . . . 11

**4 消費電力モードウェイクアップのまとめ** . . . 12

**5 消費電力低減技術** . . . 13

5.1 ModusToolbox™ ソフトウェアでのペリフェラルのオフ . . . 13

5.2 PSOC™ Creator でのコンポーネントのオフ . . . 13

5.3 低速でコンポーネントを動作 . . . 14

5.4 電源電圧の低下 . . . 14

5.5 PSOC™ 4 MCU デバイスによる電流パスの開閉 . . . 14

5.6 DMA を用いてデータを移動 . . . 15

**6 低消費電力モードのその他の注意事項** . . . 16

6.1 クロック . . . 16

6.2 WDT . . . 17

6.3 GPIO . . . 18

6.4 TCPWM . . . 18

6.5 SAR ADC . . . 19

6.6 ディープスリープおよびハイバネート レギュレータ . . . 19

## 目次


6.7 デバッグインターフェース . . . . . 20
6.7.1 PSOC™ Creator . . . . . 20
6.7.2 ModusToolbox™ソフトウェア . . . . . 20

**7 サンプルコード** . . . . . 21
7.1 ModusToolbox™ ソフトウェアサンプルコード . . . . . 21
7.2 PSOC™ Creator プロジェクト . . . . . 21
7.3 PSOC™ 4 MCU サンプルコード . . . . . 23
7.3.1 プロジェクト 1: 消費電力モード . . . . . 23
7.3.2 プロジェクト 2: 低消費電力コンパレータ . . . . . 25
7.3.3 プロジェクト 3: ディープスリープ ADC . . . . . 25
7.3.3.1 平均電力 . . . . . 26

**8 ModusToolbox™ ハードウェア設定** . . . . . 28

**9 PSOC™ Creator ハードウェア設定** . . . . . 29
9.1 CY8CKIT-042 キット (PSOC™ 4200) . . . . . 29
9.2 CY8CKIT-049-42xx キット (PSOC™ 4200) . . . . . 30
9.3 CY8CKIT-043 キット (PSOC™ 4200M) . . . . . 31

**10 DMM による電流測定** . . . . . 32
10.1 消費電力量の概算 . . . . . 32

**11 サンプルの再利用** . . . . . 33

**12 まとめ** . . . . . 34

**13 関連アプリケーションノート** . . . . . 35

**改訂履歴** . . . . . 36

**Trademarks** . . . . . 38

**免責事項** . . . . . 39

# 1 はじめに

PSOC™ 4 MCU の低消費電力モードは、特に他の省電力機能と技術を用いて実行する場合に、重要な機能を維持しながら全体の消費電力を削減できます。本アプリケーション ノートは、デバイスの低消費電力モードを説明しており、アクティブモードでの省電力方法にかかわる情報を提供する他、低消費電力の考慮事項についても論じています。

本書では、ModusToolbox™ ソフトウェア環境または PSOC™ Creator を使用して PSOC™のアプリケーション開発に慣れていることを想定しています。PSOC™ 4 MCU が初めての場合、AN79953 – Getting started with PSOC™ 4 MCU を参照してください。ModusToolbox™ ソフトウェアまたは PSOC™ Creator が初めての場合、ModusToolbox™ ソフトウェアホームページ または PSOC™ Creator ホームページを参照してください。その他のリソースについては関連アプリケーションノート を参照してください。

# 2 消費電力モードの概要

PSOC™ 4 MCU は、5 つの動作電力モードを特長としています。5 つの消費電力モードは消費電力と機能の順に、アクティブ, スリープ, ディープスリープ, ハイバネート, およびストップです。表 1 にそれぞれの電力モードの標準電流とウェイクアップ時間を示します。その他の値および特定の条件については、製品のデータシートを参照してください。

- アクティブモードは、すべてのペリフェラルが使用可能で、CPU が命令を実行している通常の動作モードです。
- スリープモードは、CPU を除くすべてのペリフェラルが使用可能です。
- ディープスリープモードは、CPU, ほとんどのペリフェラル, および MHz クロックが無効です。
- ハイバネートモードは、クロックは使用できませんが、論理状態は保持されます。
- ストップモードは、CPU, クロック, およびすべてのペリフェラルが停止し、論理状態は保持されませんが、GPIO 状態は保持またはフリーズされます。

*注: 表 1 は、一部の PSOC™ 4 MCU デバイスが 5 つの電力モードすべてをサポートしていないことも示しています。*

**表 1 消費電力モードの仕様**

| 消費電力モード | 電流範囲 (typ) (VDD = 3.3 V ～ 5.0 V) | ウェイクアップ時間 | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | PSOC™ 4000/4000S/4000S 4100S Plus/4100S Plus 256k/4100S Max/4000T/4100T Plus | PSOC™ 4100 Bluetooth® LE | PSOC™ 4200 Bluetooth® LE | PSOC™ 4200DS | PSOC™ 4500S | PSOC™ 4700S | PSOC™ アナログコプロセッサ | PSOC™ 4100PS | PSOC™ 4100/4200 | PSOC™ 4100M/4200M | PSOC™ 4200L |
| アクティブ | 1.3 mA～14 mA | - | - | - | - | - | - | - | - | - | - | - |
| スリープ | 1.0 mA～3 mA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ディープスリープ | 1.3 µA～15 µA | 35 µs | 25 µs | 35 µs | 35 µs | 35 µs | 35 µs | 35 µs | 35 µs | 25 µs | 25 µs | 25 µs |
| ハイバネート | 150 nA～1 µA | 該当なし | 2 ms | 0.7 ms | 該当なし | | | | | 2 ms | 0.7 ms | 0.7 ms |
| ストップ | 20 nA～80 nA | 該当なし | 2 ms | 2.2 ms | | | | | | 2 ms | 2 ms | 1.9 ms |

低消費電力モードでは、CPU とペリフェラルの利用可能性, ウェイクアップとリセットの発生源, 消費電力モードの遷移, および電力消費量に関して違いがあります。表 2 に PSOC™ 4 MCU リソースおよび異なる消費電力モードでのそれらの利用可能性を示します。

## 2 消費電力モードの概要

**表 2　PSOC™ 4 MCU の消費電力モードとリソースの利用可能性**

| サブシステム | アクティブ | スリープ | ディープスリープ | ハイバネート | ストップ |
|---|---|---|---|---|---|
| CPU | オン | 保持 1) | 保持 | オフ | オフ |
| SRAM | オン | オン | 保持 | 保持 | オフ |
| 高速ペリフェラル (SPI, UART, etc.) | オン | オン | 保持 | オフ | オフ |
| 汎用デジタルブロック (UDB) | オン | オン | 保持 2) | オフ 3) | オフ |
| VDAC | オン | オン | 保持 2) | オフ | オフ |
| SPI スレーブおよび $I^2C$ スレーブ (SCB ベース) | オン | オン | オン | オフ | オフ |
| 高速クロック (IMO, ECO, および PLL) | オン | オン | オフ | オフ | オフ |
| 低速クロック (32 kHz) (ILO および WCO) | オン | オン | オン | オフ | オフ |
| 電圧低下検出 | オン | オン | オン | オン | オフ |
| 連続時間ブロック (CTB) (オペアンプおよびコンパレータ) | オン | オン | オン | オフ | オフ |
| 連続時間ブロック mini (CTBm) (オペアンプおよびコンパレータ) | | | | | |
| ADC | オン | オン | オフ | オフ | オフ |
| 低消費電力コンパレータ | オン | オン | オン | オン | オフ |
| GPIO (出力状態) | オン | オン | オン | オン | 凍結 4) |

1) 保持: ペリフェラルのコンフィギュレーションと状態は保持されます。デバイスがアクティブモードへ遷移すると、ペリフェラルは動作を再開します。
2) UDB に基づく関数のように VDAC の状態は、ディープスリープに遷移する前に_Sleep() 関数を呼び出すことで保持され、またディープスリープモードの終了後に_WakeUP() 関数を呼び出すことによって復元されます。
3) ハイバネートモードの終了時に PSOC™ 4 MCU デバイスがリセットするため、すべての UDB を基にした機能は再初期化されます。
4) 凍結: すべての GPIO の設定, モード, および状態はロックされます。デバイスがアクティブモードへ遷移し、GPIO のロックが解除されるまで GPIO 状態の変更はできません。

# 3 低消費電力モードの詳細

ModusToolbox™ ソフトウェア環境は現在、スリープモードとディープスリープ電力モードを備えた PSOC™ 4 MCU デバイスをサポートします。PSOC™ Creator は、すべての消費電力モードで、すべての PSOC™ 4 MCU デバイスをサポートします。ModusToolbox™ ソフトウェアと PSOC™ Creator の両方に、これらの消費電力モードを示すサンプルコードがあります。プロジェクト 1: 消費電力モードを参照してください。

図 1 に、PSOC™ MCU の消費電力モード間の遷移を示します。表 3 に、各モードで使用可能なウェイクアップソースを示します。

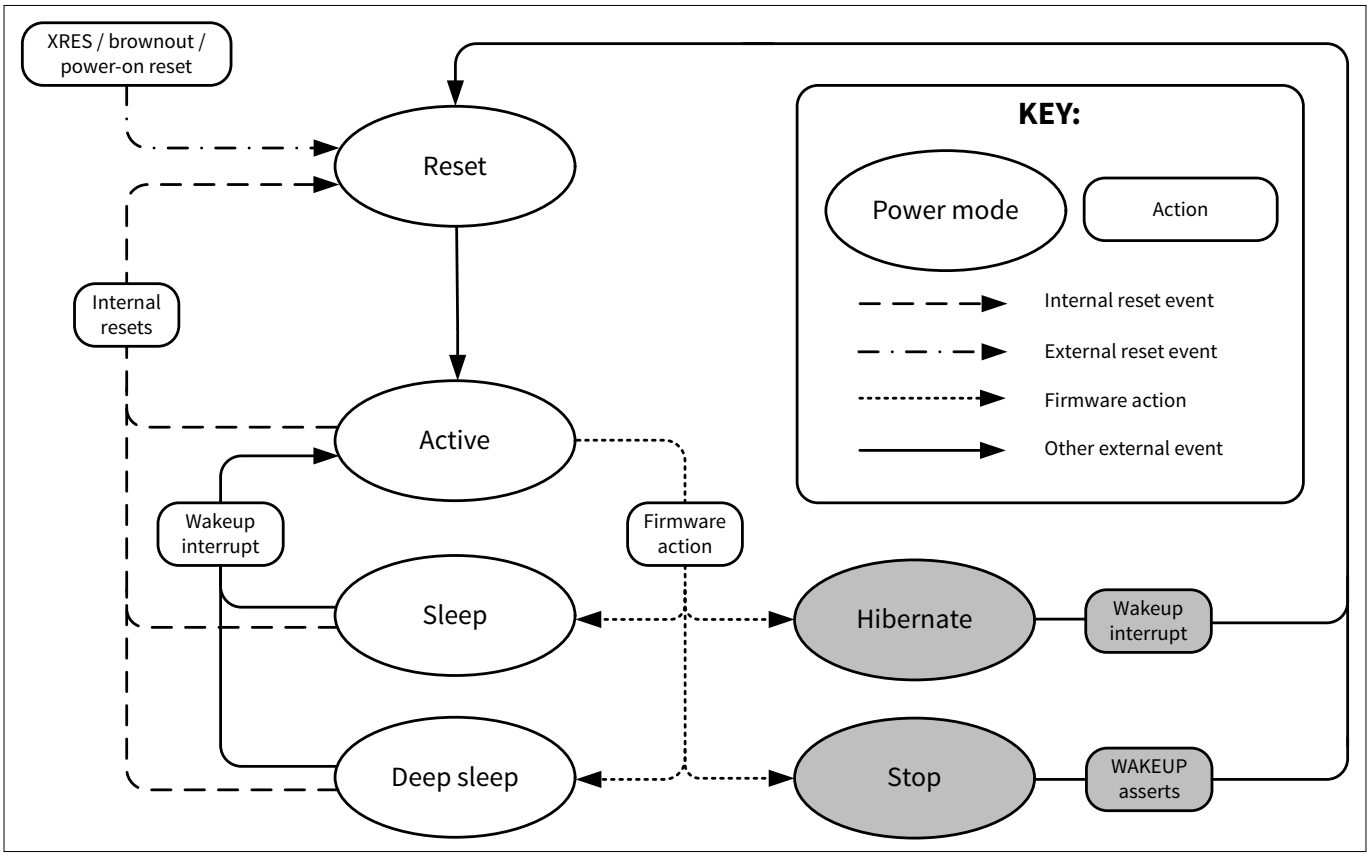


**図 1 PSOC™消費電力モードの遷移図**

*注: 灰色のブロックは、PSOC™ 4000, 4000S, 4100S, 4100S Plus, 4100S Plus 256k, 4100S Max, 4200DS, 4500S, 4700S, 4100PS, 4000T, 4100T Plus, およびアナログコプロセッサ製品では使用できません。*

## 3.1 スリープモード

スリープモードでは PSOC™ 4 Arm® Cortex®-M0/M0+ CPU は命令を実行せずに、割込みの発生を待機します。SRAM は保持されますが、CPU によって読み書きはできません。DMA をサポートする製品では DMA によりこれにアクセスできます。すべての他のペリフェラルおよびクロックは動作し続けます。

### 3.1.1 スリープモードのウェイクアップソース

デバイスのいずれかの割込みソースを使用してデバイスをスリープモードから起こせます。すべてのペリフェラルはアクティブの状態を保持し、割込みを生成できます。

### 3.1.2 スリープモードの遷移

ModusToolbox™ および PSOC™ Creator は、スリープモードに入る API 関数を提供します。これらの関数は、デバイスをスリープモードに設定します。他の API 呼び出しは必要ありません。

**ModusToolbox™ソフトウェア:** Cy_SysPm_CPU_EnterSleep() – システムをディープスリープ電力モードに設定し、登録されたコールバック関数を呼び出します。これは CPU 中心の電力モードです。これは、CPU がスリープモードにあり、メインクロックを削除できることを示していることを意味します。**現在のスリープ状態を返します**。

**PSOC™ Creator**: `CySysPmSleep()` – システムをスリープ電力モードに設定し、登録されたコールバック関数を呼び出します。これは CPU 中心の電力モードです。これは、CPU がスリープモードにあり、メインクロックを削除できることを示していることを意味します。**戻り値はありません**。

割込みがトリガーされると、スリープモードが終了します。スリープモードの終了後、デバイスはアクティブモードに戻ります。スリープのウェイクアップソースの設定は割込みを有効にするだけで済みます。

### 3.1.3 スリープモードのユースケース

スリープモードは、ADC, CAPSENSE™, デジタル通信, またはその他の機器でペリフェラルのアクティブ状態を保持する必要がある時に使用しますが、CPU の動作は必要ありません。これにより、ADC 変換およびデジタル通信トランザクションなどのイベント間は消費電流を削減できます。

## 3.2 ディープスリープモード

ディープスリープ モードでは高周波数クロックおよび該当クロックを必要とするペリフェラルは無効です。高周波数クロックとは、内部主発振器 (IMO)、外部水晶発振器 (ECO) および位相同期回路 (PLL) です。すべての PSOC™ 4 MCU デバイスでは ECO と PLL がサポートされないため注意してください。

内部低速発振器 (ILO)クロックは依然として有効であり、ウォッチドッグ タイマ (WDT) にクロックを供給するために使用されます。このウォッチドッグ タイマはシステムをディープスリープ状態から復帰させるためのスリープタイマとして使用できます。一部の PSOC™ 4MCU デバイスは、ディープスリープ状態で動作可能な時計用水晶発振器 (WCO) を内蔵します。

$I^2C$ ブロックは $I^2C$ バスを監視するためにスレーブモードで引き続き動作し、$I^2C$ アドレスが一致するとウェイクアップが発生します。

### 3.2.1 ディープスリープモードのウェイクアップソース

$I^2C$ アドレスの一致, WDT, GPIO 割込み, CTB/CTBm コンパレータ割込み, および低消費電力コンパレータ割込みは、デバイスをディープスリープモードから復帰させます。WDT ブロックには複数のカウンターがあり、これらのカウンターは割込みまたはリセット、あるいはこれらの両方を生成するために独立して設定できます。これにより WDT はスリープタイマとして機能できます。

### 3.2.2 ディープスリープモードの遷移

ModusToolbox™ ソフトウェアと PSOC™ Creator は、ディープスリープモードに入るための API 関数を提供します。これらの関数は、デバイスをディープスリープモードに設定します。他の API 呼び出しは必要ありません。

**ModusToolbox™ソフトウェア:** Cy_SysPm_CpuEnterDeepSleep() – システムをディープスリープ電力モードに設定します。システムの準備が整う前にファームウェアがこのモードへ入ろうとすると、デバイスは代わりにスリープモードになり、ホールドオフの期限が切れると自動的に本来のモードになります。**現在のスリープ状態を返します**。

**PSOC™ Creator:** `CySysPmDeepSleep()` – システムをディープスリープ電力モードに設定します。システムの準備が整う前にファームウェアがこのモードへ入ろうとすると、デバイスは代わりにスリープモードになり、ホールドオフの期限が切れると自動的に本来のモードになります。**戻り値はありません**。

割込みがトリガーされると、ディープスリープモードが終了します。ディープスリープモードの終了後、PSOC™ 4 はアクティブモードに戻ります。ディープスリープのウェイクアップソースの設定は、割込みを有効にするだけで済みます。

### 3.2.3 ペリフェラルディープスリープ設定

デバイスがディープスリープに入るとき、いくつかのペリフェラルは、ディープスリープで動作し続けるか、デバイスをディープスリープから復帰させられるように設定する必要があります。

**ModusToolbox™ソフトウェア:** ペリフェラルがディープスリープ用に正しく設定されていることを確認するために、ModusToolbox™ には、特定の API 関数 `_DeepSleepCallback()` および `_DeepSleep()`があります。これらは、ディープスリープでの動作およびディープスリープからのウェイクアップのためにペリフェラルを設定します。詳細については、Peripheral Driver Library (PDL)を参照してください。

**PSOC™ Creator:** コンポーネントは、電力を節約するか、電流状態を節約するか、またはその両方を行うために、ディープスリープで動作するように設定できます。このような場合、`CySysPmDeepSleep()`; およびディープスリープからウェイクアップした後の`_WakeUp()`を呼び出す前に、コンポーネントの特定の API 関数`_Sleep()` を使用して現在のコンポーネントの状態を保存します。アプリケーションによっては、ディープスリープに入る前にコンポーネントの`_Stop()` 関数を呼び出し、ウェイクアップ後に`_Start()` を呼び出す方がはやい場合があります。

### 3.2.4 ディープスリープモードのユースケース

ディープスリープモードは、PSOC™ 4 の高性能アナログおよびデジタル ペリフェラルが不要な場合に使用する必要がありますが、デバイスが WDT を使用して定期的に、または $I^2C$ アドレス一致などのイベントでウェイクアップできる必要があります。

定期的なウェイクアップ間隔は、読み出しまたは CAPSENSE™ボタン入力のスキャニング目的で、ADC などのアクティブモードのペリフェラルを定期的に使用するために使われます。

## 3.3 PSOC™ Creator でのハイバネートモード

ハイバネートモードでは、PSOC™ 4 MCU のすべてのクロックおよび同期ペリフェラルが無効になります。ピンと低消費電力コンパレータはアクティブのままであり、SRAM と UDB のレジスタ状態は保持されます。**現在、ModusToolbox™ ソフトウェアは、ハイバネートモードをサポートする PSOC™ 4 MCU デバイスをサポートしていません。**

***注:*** *PSOC™ 4000, 4000S, 4100S, 4100S Plus, 4100S Plus 256k, 4100S Max, 4200DS, 4500S, 4700S, 4100PS, 4000T, 4100T Plus, およびアナログコプロセッサ製品は、ハイバネートモードをサポートしていません。*

### 3.3.1 ハイバネートモードのウェイクアップソース

ピンおよび低消費電力コンパレータの割込みは、デバイスをハイバネートモードから復帰させられます。ハイバネートモードからウェイクアップするとデバイスがリセットされますが、SRAM と一部のレジスタの状態は保持されるため、ウェイクアップリセットの原因を検出できます。

SRAM は保持されますが、リセット後、C スタートアップコードはすべてのグローバル変数と静的変数を初期値またはゼロに初期化します。ハイバネート状態から復帰した後に変数が再初期化されるのを防ぐには、変数の定義で `CY_NOINIT` 属性を使用します。

変数が確実に初期化されるようにするには、コードは、リセットがハイバネートモードからのウェイクアップによって引き起こされのではないかどうかを判断する必要があります。リセットからアクティブモードに戻った後、`CySysPmGetResetReason()` API 関数を呼び出して、リセットがハイバネート状態からのウェイクアップなのか、その

他のリセット状態からのウェイクアップなのかを判断します。次のコードは、初期化されていない変数を定義し、リセットがハイバネート状態のウェイクアップによるものではない場合に初期化する方法を示します。

```
/* Define non-initialized global variable */ int32 CY_NOINIT testVarNoInit; /* Inside of main()
*/ /* Initialize variable when PSoC™ is not reset from hibernation wakeup */
if( CySysPmGetResetReason() != CY_PM_RESET_REASON_WAKEUP_HIB ) { testVarNoInit = 0; }
```

### 3.3.2 ハイバネートモードの遷移

CySysPmHibernate() API 関数を使用してハイバネート モードに入ります。この関数は、デバイスをハイバネートモードに設定します。クロックを含むすべてのコンポーネントの電源が切断され、リセットしてハイバネート モードを終了したとき、再初期化されるため、他の関数呼び出しは必要ありません。唯一の例外は、ハイバネートモードからウェイクアップするために、低消費電力コンパレータがアクティブのままとなる可能性があることです。

ピンまたは低消費電力コンパレータ割込みがトリガーされると、ハイバネートモードが終了します。ハイバネートモードを終了すると、PSOC™ MCU はリセットされます。リセットからアクティブモードに戻った後、CySysPmGetResetReason()を呼び出して、ハイバネートのウェイクアップリセットを検出できます。特定のピンまたはコンパレータ割込みは、それらのレジスタの状態が保持されるので、コンポーネント API を使用して検出できます。

必須ではありませんが、ハイバネートモードに入る前に CySysPmfreezeIo()を呼び出すことにより、すべての I/O セル (GPIO) の状態をロックするオプションがあります。CySysPmUnfreezeIo()関数は、ハイバネートモードからウェイクアップした後、ピンが再び状態を変更する前に呼び出す必要があります。これらの関数を使用すると、リセット中およびリセット後の予期しない GPIO 遷移を防げます。

### 3.3.3 ハイバネートモードのユースケース

定期的なウェイクアップが不要で、デバイスが使用する電流が 1 µA 未満である場合は、ハイバネートモードを使用する必要があります。また、最小限の電流でアナログまたはデジタル信号遷移のウェイクアップにも役立つ場合があります。

コードがステートマシンとして構成されている場合、ハイバネートモードを効果的に使用でき、デバイスがハイバネートモードからウェイクアップすると、CPU はウェイクアップ前に以前の状態からコードの実行を開始できることに注意してください。状態変数の定義で CY_NOINIT 属性を使用して、ハイバネートモードからウェイクアップ後に状態変数が再初期化されないようにする必要があります。前のセクションで説明したように、CySysPmGetResetReason() を呼び出して、ハイバネートのウェイクアップリセットを検出できます。

## 3.4 PSOC™ Creator でのストップモード

ストップモードは PSOC™ 4 MCU 電源ピンから電源を取り外さないで可能な限り低い消費電流となるようにします。すべてのペリフェラルは無効になり、SRAM およびレジスタの状態は保持**されません**。デバイスピンは「凍結」され、駆動モードと論理状態を保持します。デバイスをストップモードからウェイクアップするための専用のウェイクアップピン P0[7] があります。**現在、ModusToolbox™ ソフトウェアは、ストップモードをサポートする PSOC™ 4 MCU デバイスをサポートしていません。**

***注**: PSOC™ 4000, 4000S, 4100S, 4100S Plus, 4100S Plus 256k, 4100S Max, 4200DS, 4500S, 4700S, 4100PS, 4000T, 4100T Plus, およびアナログコプロセッサ製品は、ストップモードをサポートしていません。*

### 3.4.1 ストップモードのウェイクアップソース

専用のウェイクアップピン P0[7] は、ストップモードで使用できる唯一のウェイクアップソースです。CySysPmSetWakeupPolarity()関数を呼び出すことにより、入力ウェイクアップ極性を立ち上りエッジまたは立ち下りエッジに設定できます。

### 3.4.2 ストップモードの遷移

ストップモードに入るには、CySysPmStop() 関数を呼び出してください。この関数は、I/O 状態の凍結を含む、デバイスをストップモードに設定します。専用のウェイクアップピンを使用する場合は、ストップモードに入る前に CySysPmSetWakeupPolarity()関数を使用して、入力ウェイクアップ極性を設定してください。

専用ピンのウェイクアップがトリガーされ、リセット信号が LOW になった場合、または電源を入れ直した場合に、ストップモードが終了します。ストップモードが終了すると、PSOC™ 4 MCU はリセットされます。リセット後にアクティブ モードに戻った後、CySysPmGetResetReason()関数を呼び出して、ウェイクアップピンをトグルするか電源を入れ直してストップモードが終了したかを判断できます。ウェイクアップピンのリセット後、GPIO 状態は凍結されたままであり、ピンの状態を変更する前に、CySysPmUnfreezeIo() 関数を使用して凍結を解除する必要があります。CySysPmStop() は I/O セルを暗黙的に凍結するため、ストップ モードに入る前に CySysPmfreezeIo()関数を呼び出す必要はありません。

### 3.4.3 ストップモードのユースケース

最小限の消費電力で最小限の機能を確保したいとき、ストップモードを使用してください。これは、ホストコントローラーまたはボタンを押すなどのユーザー入力によって専用のウェイクアップピンをトリガーでき、電源トポロジでデバイスから電源を切断できないアプリケーションで特に役立ちます。

消費電力モード移行 API の詳細については、system reference guide を参照してください。

# 4 消費電力モードウェイクアップのまとめ

すべての消費電力モードは、電源の入れ直しまたはリセット入力の LOW 駆動という 2 つの条件で終了します。いずれの場合も、**すべての**消費電力モードでの状態が失われます。表 3 に、現在の低消費電力モードを終了してアクティブモードに戻るために使用できるイベントとハードウェアの概要を示します。

**表 3 低消費電力モードおよびウェイクアップソース**

| 低消費電力モード | ウェイクアップソース | ウェイクアップ動作 |
|---|---|---|
| スリープ | 任意の割込みソース | 割込み |
| | 任意のリセットソース | デバイスリセット |
| ディープスリープ | GPIO 割込み | 割込み |
| | 低消費電力コンパレータ | 割込み |
| | CTB/CTBm コンパレータ | 割込み |
| | SCB ($I^2C$ アドレス一致) | 割込み |
| | WDT[1)] | 割込みまたはデバイスリセット |
| | XRES (外部リセットピン)[2)] | デバイスリセット |
| ハイバネート[3)] | GPIO 割込み | デバイスリセット |
| | 低消費電力コンパレータ | デバイスリセット |
| | XRES (外部リセットピン)[2)] | デバイスリセット |
| ストップ[3)] | WAKEUP ピン (P0[7]) | デバイスリセット |
| | XRES (外部リセットピン)[2)] | デバイスリセット |

1) WDT は異なる間隔で割込みとリセットの両方を発生させるように設定できます。対応する PSOC™ 4 MCU デバイスファミリについては、リファレンスマニュアルを参照してください。
2) XRES はシステム全体のリセットを発生させます。凍結 I/O を含むすべての状態は失われます。この場合、ウェイクアップの原因はデバイスが再起動した後で検出されません。
3) ModusToolbox™ のソフトウェア環境は、ハイバネートモードまたはストップモードをサポートしていません。

# 5 消費電力低減技術

多くのアプリケーションでは、PSOC™ 4 MCU ブロックを適切に管理し、外部コンポーネントの電源を切ることで、さらに消費電力を削減できます。ここでは、そのためのいくつかのテクニックを紹介します。

## 5.1 ModusToolbox™ ソフトウェアでのペリフェラルのオフ

未使用のブロックを無効にすることで、不要な消費電流を節約できます。節約される電力は、無効になっているブロックによって異なります。

アクティブモードまたはスリープモードで無効にできるブロックは、_Enable() および _Disable() 関数によってサポートされます。_Disable() 関数は、ペリフェラルのすべての操作を即座に停止します。ペリフェラルはアクティブにタスクを実行している可能性があるため、無効にする前にそのステータスを確認してください。次のプログラムコードは、例として TCPWM ブロックのタイマ/カウンターを使用しています。

```
/* <Check task status here.> */ /* Disable the peripheral. */ Cy_TCPWM_Counter_Disable(TCPWM,
MY_TCPWM_CNT_NUM);
```

_Enable() 関数を呼び出して、ペリフェラルを再起動します。有効にする前に、ペリフェラルを再初期化することを推奨します。

```
/* Initialize the peripheral */ Cy_TCPWM_Counter_Init(TCPWM, MY_TCPWM_CNT_NUM, &config); /*
Enable the peripheral. */ Cy_TCPWM_Counter_Enable(TCPWM, MY_TCPWM_CNT_NUM);
```

TCPWM などの一部のペリフェラルは、トリガーで開始する必要がある場合があります。これは、すべての TCPWM カウンターが同時に開始されることを保証するソフトウェアトリガーです。1 つ以上の TCPWM カウンターを開始する必要がある場合は、この関数を呼び出します。

```
/* Trigger start the peripheral. */ Cy_TCPWM_TriggerStart(TCPWM, MY_TCPWM_CNT_MASK);
```

*注: すべてのペリフェラルをトリガーで開始する必要はありません。詳細については、Peripheral Driver Library (PDL) を確認してください。*

## 5.2 PSOC™ Creator でのコンポーネントのオフ

アクティブモードで電力を削減する最も簡単な方法は、未使用の PSOC™ Creator コンポーネントをオフにすることです。

アクティブモードまたはスリープモードで無効にできるコンポーネントには、API に _Stop() 関数が含まれています。この関数は、コンポーネントのすべての動作を即座に停止し、コンポーネントを最低消費電力状態に設定します。コンポーネントはアクティブにタスクを実行している可能性があるため、停止する前にそのステータスを確認してください。

```
/* <Check task status here.> */ /* Stop the Component. */ MyComponent_Stop();
```

start 関数を呼び出して、コンポーネントを再起動します。

```
/* Start the Component. */ MyComponent_Start();
```

電源を切る前に設定データを保持する必要があるコンポーネントには、API に _Sleep() 関数が含まれています。_Sleep() 関数は、必要なすべてのコンポーネント設定を保存してから、_Stop() 関数を呼び出します。場合によっては、_Sleep() 関数は _Stop()を呼び出すだけです。

```
/* < Check task status here.> */ /* Sleep the Component. */ MyComponent_Sleep();
```

コンポーネントがスリープ状態になったら、その _Wakeup() 関数を呼び出してウェイクアップする必要があります。これにより、コンポーネントがスリープ前の状態に復元されます。_Start() 関数もコンポーネントを動作に戻しますが、デフォルトの状態に再初期化されます。

```
/* Wake the Component. */ MyComponent_Wakeup();
```

_Sleep() と _Stop() はどちらも、同じ量の電力を節約します。違いは、コンポーネントが終了した状態から再開する必要があるかどうかです。

## 5.3 低速でコンポーネントを動作

クロック供給された集積回路は、クロックレートが上がるにつれて、より多くの電流を消費します。これは、寄生容量と設計容量がより急速に充放電され、より多くの電流が必要になるためです。PSOC™ MCU コンポーネントの動作周波数を下げると、消費電流を大幅に削減できます。この技術は、CM0/CM0+ CPU, SAR ADC, デジタルコンポーネントなどに適用できます。

## 5.4 電源電圧の低下

全体的な消費電力を削減するためには、おそらく電源電圧を下げるのが最も簡単な方法です。電流が同じでも、電源電圧を 5 V から 3 V に下げると、消費電力が 40%削減されます。PSOC™ MCU デバイスは 1.8 V 未満で動作できますが、システム内の他のデバイスの電圧要件を考慮する必要があります。

## 5.5 PSOC™ 4 MCU デバイスによる電流パスの開閉

PCB には、電力を消費する他のコンポーネントが含まれている場合があります。PSOC™ 4 MCU デバイスを使用して、それらが消費する電流を制御できます。データシートに記載されているピンソースおよびシンク能力の最大値を超えてはいけないことに注意してください。次の例は、PSOC™ Creator コンポーネントを使用して示されています。同じ理論が、ModusToolbox™ ソフトウェアにも当てはまります。

このシナリオの良い例は、図 2 に示すように、サーミスタのアプリケーションです。この場合、PSOC™ 4 MCU デバイスは、サーミスタ抵抗の変化に応じて変化するアナログピンの電圧を使用して温度を測定します。

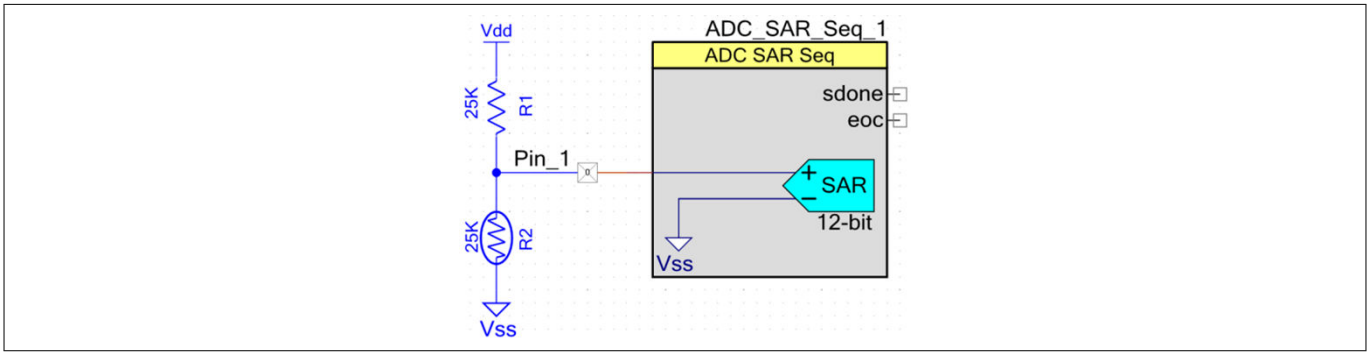


**図 2 一般的なサーミスタのアプリケーション**

ADC は使用しない時にオフにできますが、抵抗とサーミスタを通る電流パスが停止しないため、外部コンポーネントは依然として電力を消費します。PSOC™ 4 MCU での簡単な解決策は、図 3 に示すように 2 番目のピンをグランドに接続したスイッチとして使用することです。

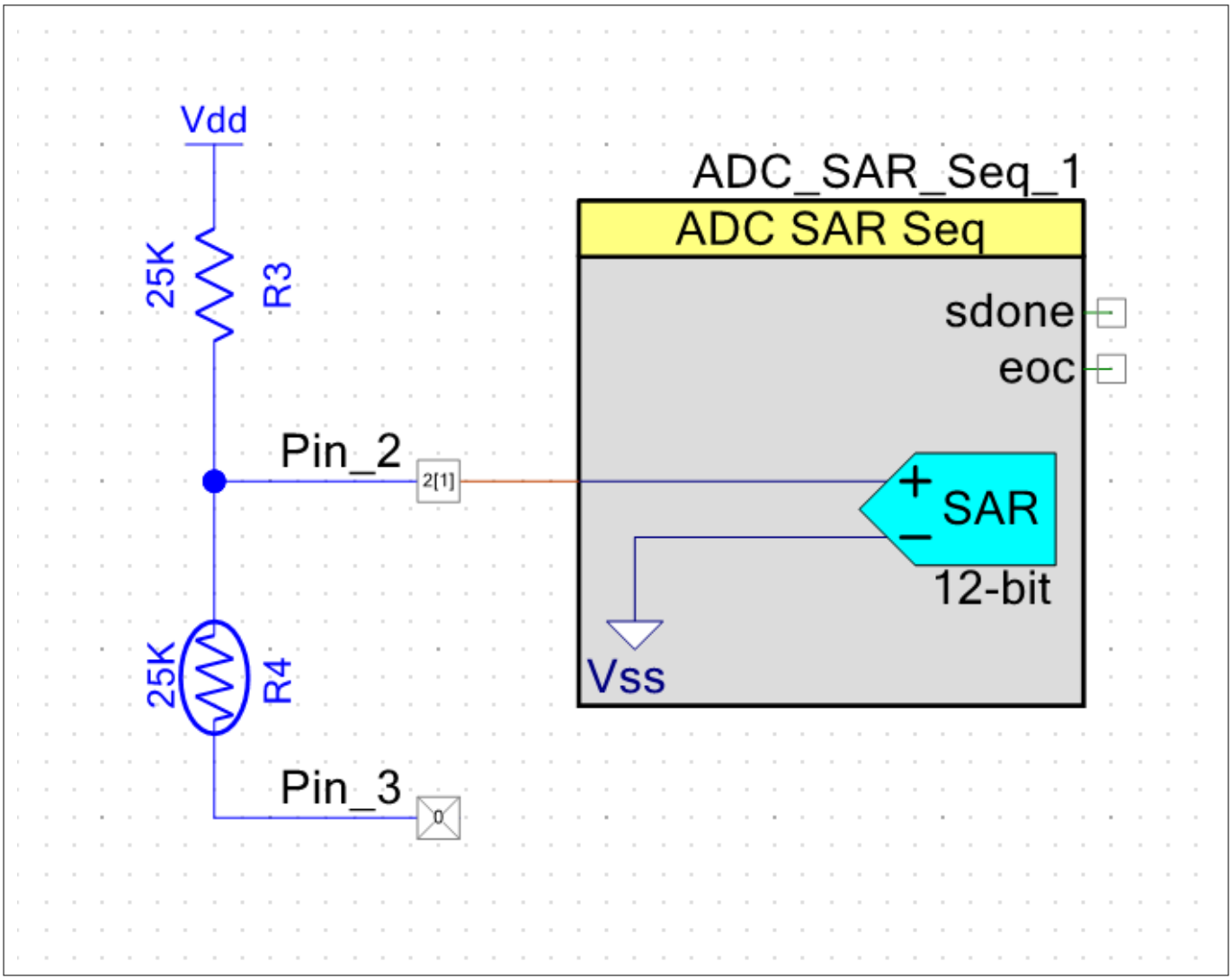


**図 3　　グランドに接続したスイッチとして GPIO を使用**

この設定では、Pin_3 に「1」を書き込んでピンをフロートさせることにより、電流の流れを停止できます。これにより、2 つの抵抗間の電圧差が 0 V に減少するため、消費電流が除去されます。「0」を書き込むと、電流の流れが再開されます。この省電力機能のリソース使用は、1 つのピンと数行のコードのみです。

また、測定が必要ない場合は、Pin_3 駆動モードを High-Z アナログに変更して、電流が流れる経路を無くせます。測定するときは、Pin_3 駆動モードを Strong に設定し、ピンにロジック 0 を書き込んでください。

## 5.6　　DMA を用いてデータを移動

CPU からタスクをオフロードし、CPU を停止するか、他の処理を並行して実行させると、いつでも電力を節約できます。DMA エンジンは、アクティブモードとスリープモードの両方で使用して、CPU を使用せずにデータを転送できます。節約される電力は、CPU を停止できる場合は CPU アクティブモードと CPU ストップ電力モードとの差であり、CPU をより遅い周波数でクロックし、同じ作業を実行できる場合は CPU アクティブ電流を低くします。

次のデバイスは DMA をサポートします。4000DS, 4100 Bluetooth® LE, 4100M, 4100PS, 4100S Plus, 4100S Max, 4200 Bluetooth® LE, 4200DS, 4200L, 4200M, 4100S Plus 256k, 4500S, 4100T Plus, およびアナログコプロセッサ。

# 6 低消費電力モードのその他の注意事項

ここでは PSOC™ の低消費電力モードの使用に関するヒントや推奨事項について説明します。

## 6.1 クロック

場合によっては、高速でクロックを実行することによって、平均消費電流が少なくなることがあります。例えば、毎秒に 1 回センサーを読み出し、いくつかの計算を行って結果を別のデバイスに送信する PSOC™ MCU デザインを検討してみてください。

PSOC™ MCU デバイスがアイドル状態の時、スリープまたはディープスリープモードを使用して消費電力を削減できますが、アクティブモードで費やす時間に起因して平均消費電流が高くなってしまいます。図 4 に、システムクロックが 3 MHz に設定されたこの例での消費電流を示します。

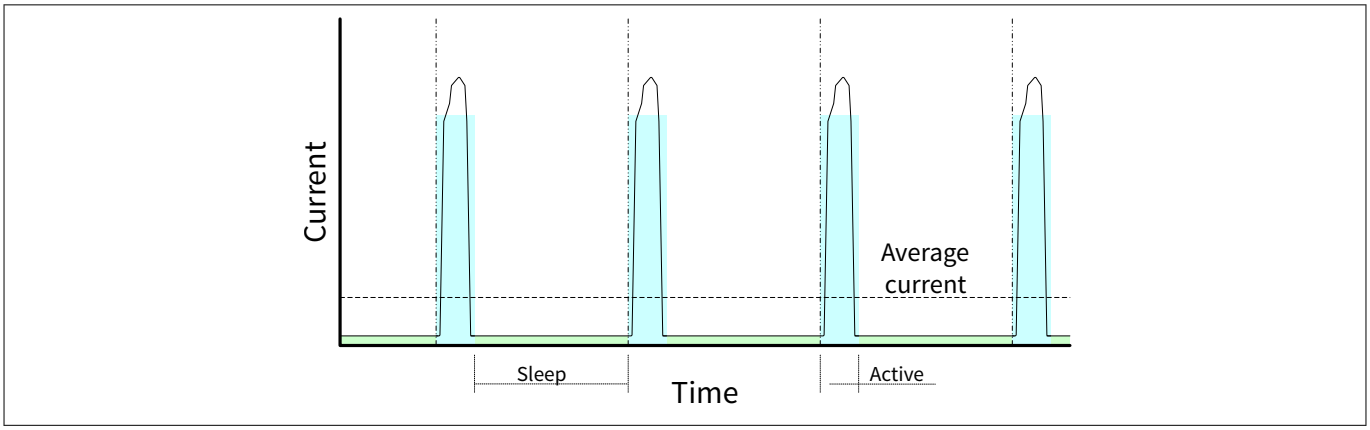


**図 4 3 MHz クロックを使った電流分布の例**

PSOC™ 4 MCU デバイスがウェイクアップ状態にある時に実行しているタスクや計算によって、システムクロックを速くすることで早く完了できる場合があります。これにより、PSOC™ 4 MCU デバイスはアクティブモードにある時間が少なくなるため、平均の消費電流を削減できます。図 5 で、タスクに分割されたアクティブモードのタイミングを説明します。

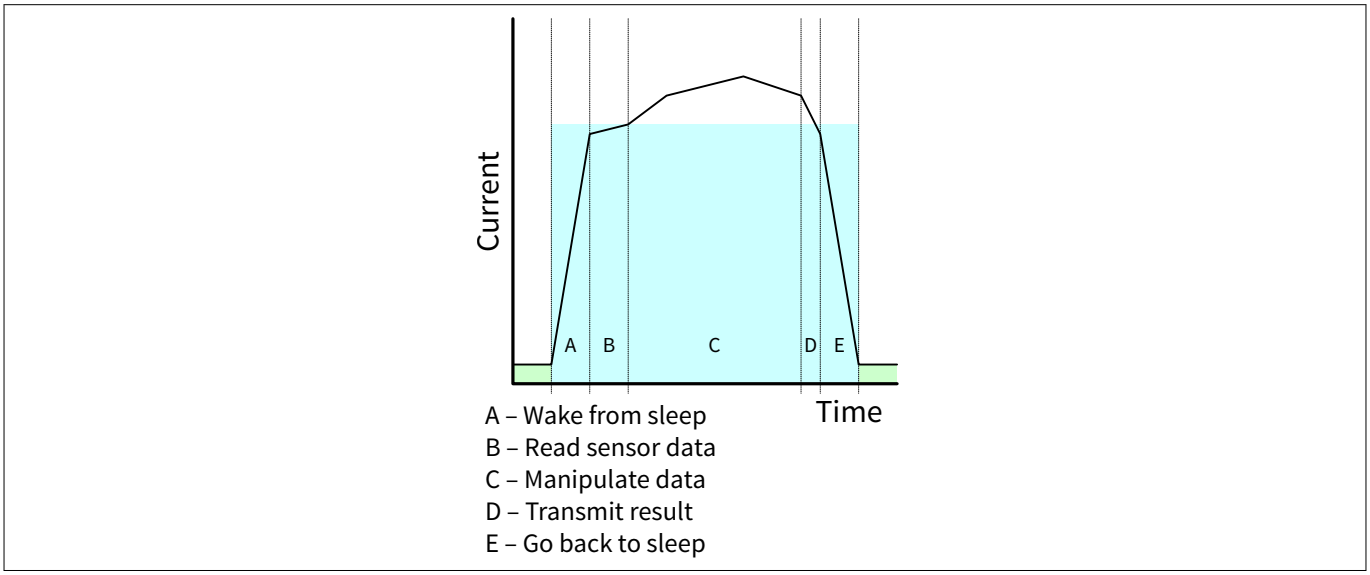


**図 5 3 MHz でのアクティブモードのタスク解析**

一部のタスクに要する時間は、システムクロックの周波数が増加しても変わりません。センサー読み出しとデータ転送はこの種類に属しています。しかし他のタスクでは、CPU の動作周波数が高くなると、処理時間が少なくて済みます。

ある時点では、アクティブ時間が短い利点よりも、高速でクロックを駆動するために必要となるエネルギーが高い弱点のほうが優ります。図 6 に示すように、最適速度が 12 MHz であると仮定します。12 MHz のクロック動作では、アクティブモードで費やす時間は 3 MHz のクロック動作の時と比べて約半分です。図 7 に、より速いクロックの時にピーク時の消費電流はより多くなりますが、全体の平均消費電流は少なくなることを示します。

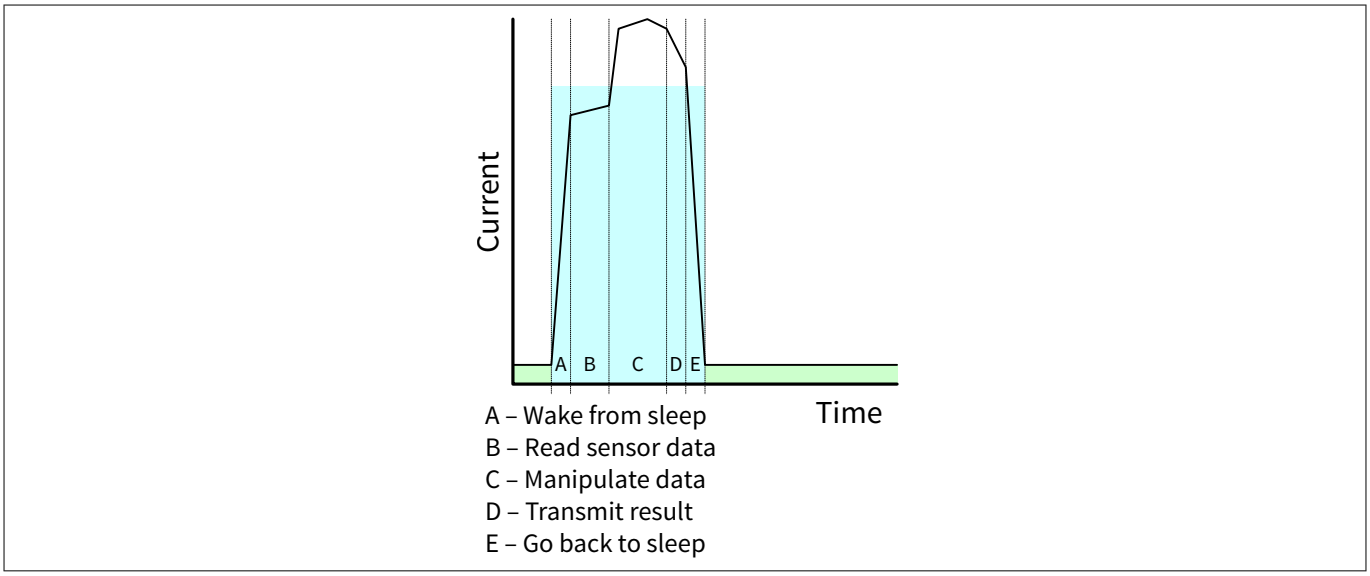


**図 6　　12 MHz でのアクティブモードのタスク解析**

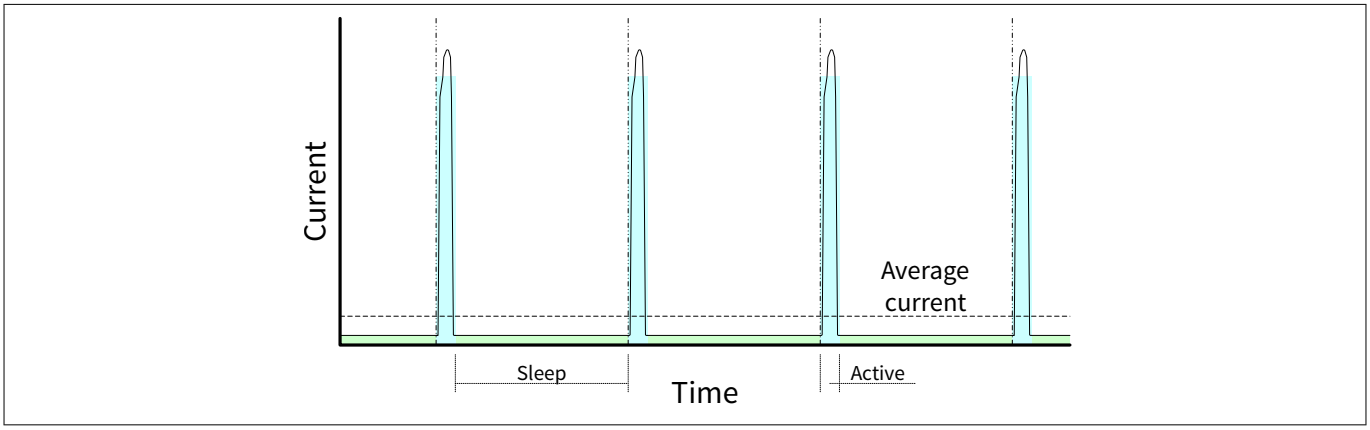


**図 7　　12 MHz クロックを使った電流分布の例**

本アプリケーションノートで説明した他の手法を適用することでも、ピーク時のアクティブ電流を低減できることがあります。プロジェクト 3: ディープスリープ ADC は、毎秒 1 回ウェイクアップし、ADC で温度を読み出し、UART を介してデータを送信してから、ディープスリープモードに戻るサンプルプロジェクトです。

## 6.2　　WDT

WDT はアクティブ, スリープ, およびディープスリープモードで動作できます。WDT のカウンターはコンフィギュレーションや動作条件によって割込みやリセットを生成できます。これにより、WDT は信頼性の高い動作を保証しながら、従来のスリープタイマに置き換えられます。

低消費電力モードに遷移する前に WDT の間隔を長くするか、または WDT を完全に無効にすることで、アクティブモードに費やす時間を減らし、総消費電力を削減できます。WDT はハイバネートまたはストップモードでは使用できません。

WDT の動作と関連 API の詳細情報は、PSOC™ 4 reference manuals, Peripheral Driver Library (PDL), および PSOC™ Creator system reference guide を参照してください。

## 6.3 GPIO

GPIO は、PSOC™デバイスが低消費電力モードのときに外部回路を駆動し続けられます。これは、外部ロジックを固定レベルに保持する必要がある場合に役立ちますが、ピンが不必要に電流をソースまたはシンクすると、電力が無駄になる可能性があります。この技術による特定の省電力は、特定の GPIO ピンに接続されている回路によって異なります。

設計を分析し、低消費電力動作中の GPIO に最適な状態を判断する必要があります。デジタル出力ピンをロジック 1 または 0 に保持するのが最適な場合は、GPIO または Pin_Write()関数を使用して設定してください。

**ModusToolbox™ソフトウェア:**

```
/* Set MyPin to ‘0’ for low power. */ Cy_GPIO_Write(MYPIN_0_PORT, MYPIN_0_NUM, 0u);
```

別の駆動モードを使用する特別な理由がないかぎり、未使用のすべての GPIO をアナログ High-Z に設定します。ポート全体の駆動モードは、_SetDriveMode() 関数を使用して設定できます。

```
/* Set MyPin to Alg HI-Z for low power. */ Cy_GPIO_SetDriveMode(MYPIN_0_PORT, MYPIN_0_NUM,
CY_GPIO_DM_HIGHZ);
```

**PSOC™ Creator:**

```
/* Set MyPin to ‘0’ for low power. */ MyPin_Write(0);
```

別の駆動モードを使用する特別な理由がないかぎり、未使用のすべての GPIO をアナログ High-Z に設定します。ポート全体の駆動モードは、_SetDriveMode() 関数を使用して設定できます。

```
/* Set MyPin to Alg HI-Z for low power. */ MyPin_SetDriveMode(MyPin_DM_ALG_HIZ);
```

PSOC™ 4 MCU の柔軟性により、GPIO の駆動モードを簡単に管理して、不要な電流リークを防げます。GPIO ピン設定の詳細については、AN86439 – PSOC™ 4 – Using GPIO pins を参照してください。

## 6.4 TCPWM

カウンタ, タイマ, または PWM を使用する場合は、周波数と精度の要件を満たしながら、チャネルをソースするクロックをできるだけ低い周波数になるように設定する必要があります。例えば、タイマで 1 秒の割込みを生成する必要がある場合は、周期が 1,000,000 カウントの 1 MHz のクロック周波数よりも、周期が 1,000 カウントの 1 kHz のクロック周波数を使用する方が適切です。TCPWM クロックを削減することによる省電力は、クロック周波数に基づいてほぼ線形です。図 8 に、TCPWM ブロックのクロック設定の比較を示します。

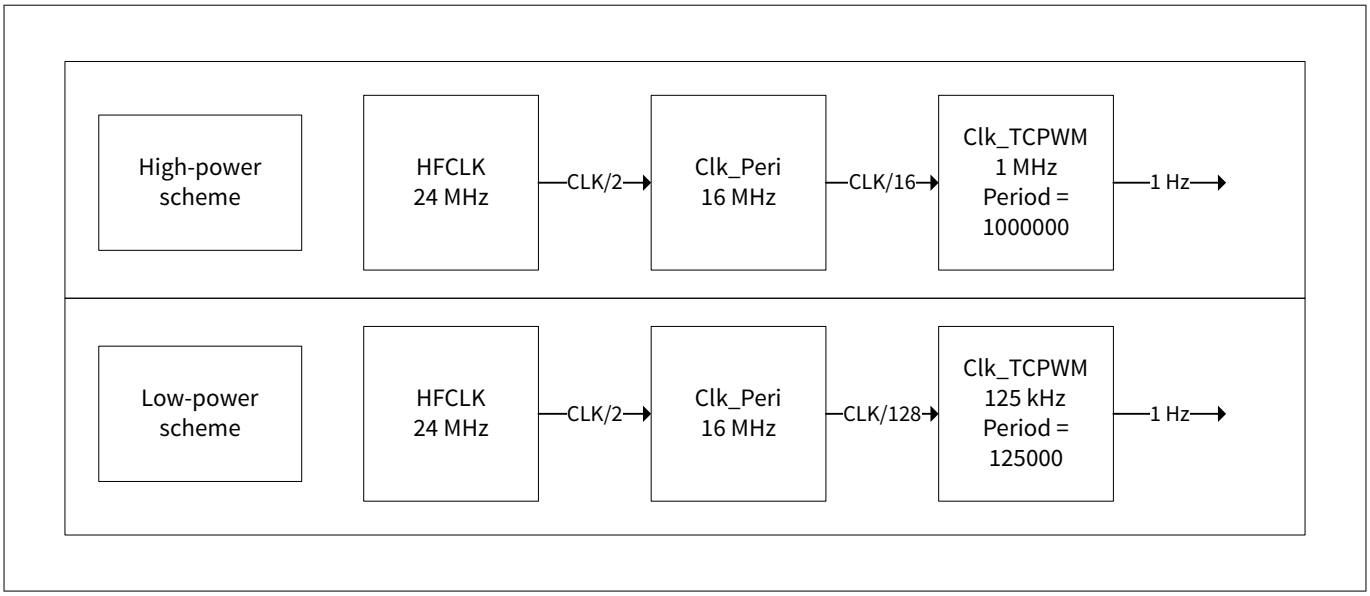


**図 8　　TCPWM クロック設定の比較**

TCPWM ブロックには、クロックプリスケーラ機能があります。消費電力を最小限に抑えるためには、TCPWM クロックプリスケーラを使用する前に、ペリフェラルクロック分周器 Clk_Peri を最大化します。

## 6.5　SAR ADC

ADC 結果のフルレートの精度が必要ない場合は、より低い分解能を使用し、平均化を使用しないでください。これにより、同じサンプルレートに必要な ADC クロックの数が減少します。

最大サンプルレートが必要ない場合は、連続モードではなくシングルショットモードの使用を検討してください。これにより、SAR ADC が常に動作することを回避できます。シングルショットモードでは、ADC はソフトウェアまたはハードウェアによってトリガされた場合にのみサンプリングします。

## 6.6　ディープスリープおよびハイバネートレギュレータ

PSOC™ 4 MCU には、ディープスリープおよびハイバネートモードで論理状態を保持するために使用される 2 つの低消費電力レギュレータがあります。

- ディープスリープレギュレータは、ILO や SCB など、ディープスリープモードで電力が供給されたままの回路に電力を供給します。ディープスリープレギュレータは、ハイバネートモードを除くすべての消費電力モードで使用できます。アクティブおよびスリープ電力モードでは、このレギュレータのメイン出力はアクティブデジタルレギュレータ ($V_{CCD}$) の出力に接続されます。このレギュレータには、ILO に安定した電圧を提供する個別のレプリカ出力もあります。この出力はアクティブモードとスリープモードでは $V_{CCD}$ に接続されていません。
- ハイバネートレギュレータは、スリープコントローラ, 低消費電力コンパレータ, SRAM など、ハイバネートモードで電力が供給されたままの回路に電力を供給します。ハイバネートレギュレータは、すべての消費電力モードで使用できます。アクティブモードとスリープモードでは、このレギュレータの出力はデジタルレギュレータの出力に接続されます。ディープスリープモードでは、このレギュレータの出力はディープスリープレギュレータの出力に接続されます。

どちらのレギュレータも $V_{CCD}$ 電源に電力をを供給しません。各レギュレータは内部電源ドメインに電源を供給し、デバイスのピンには引き出されません。

***注**:　ハイバネートレギュレータは、PSOC™ 4000, 4000S, 4100S, 4100S Plus, 4100S Plus 256k, 4100S Max, 4200DS, 4500S, 4700S, 4100PS, 4000T, 4100T Plus, およびアナログコプロセッサ製品では使用できません。これらのデバイスはハイバネートモードをサポートしていません。*

## 6.7 デバッグインターフェース

PSOC™ 4 MCU はオンチップデバッグをサポートしています。デバッグモードの間、予想より高い消費電流を観測することがあります。プログラミングおよびデバッグインターフェースがすべての低消費電力モードでアクティブのままなので、これは正常です。

PSOC™ 4 MCU デバイスがデバッグモードでない場合であっても、デバッグピンが SWD モードに設定され、MiniProg3 プログラマ/デバッガが接続されると、消費電力の測定も偏りが生じる場合があります。

### 6.7.1 PSOC™ Creator

デバッグインターフェースピンは工場出荷時にすべてのチップで GPIO モードに設定されていますが、新しい PSOC™ Creator プロジェクトではデフォルトで SWD モードに設定されます。デバッグ インターフェースを制御するレジスタは、プログラミング時にのみ変更可能です。図 9 に示すように、PSOC™ Creator プロジェクトの .cydwr ファイル内の System タブを使用してピンを GPIO モードに設定します。

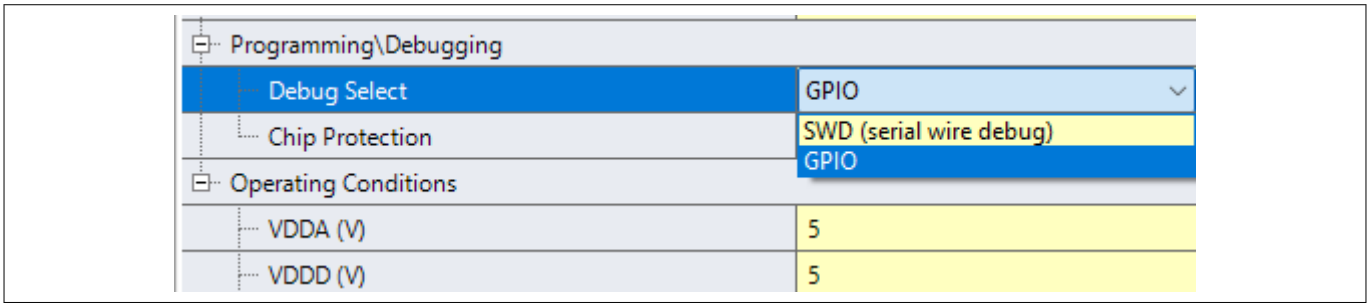


**図 9 消費電力削減のためにデバッグインターフェースをディセーブル**

### 6.7.2 ModusToolbox™ソフトウェア

デバッグインターフェースピンは、工場出荷時のすべてのチップで GPIO モードに設定されていますが、新しい ModusToolbox™ プロジェクトでは、デフォルトで SWD モードに設定されています。

デバッグインターフェースピンを GPIO モードに変更するには、次の手順を実行します。

1. **Quick panel** > **Tool** > **Device configurator** に移動してください。
2. デバイスコンフィギュレータで、**Pins** タブに移動し、図 10 に示すように、「CYBSP_SWDIO」および「CYBSP_SWDCK」というラベルの付いたピンを探してください。
3. これらのピンの選択を解除して、デバイスを GPIO モードに設定してください。

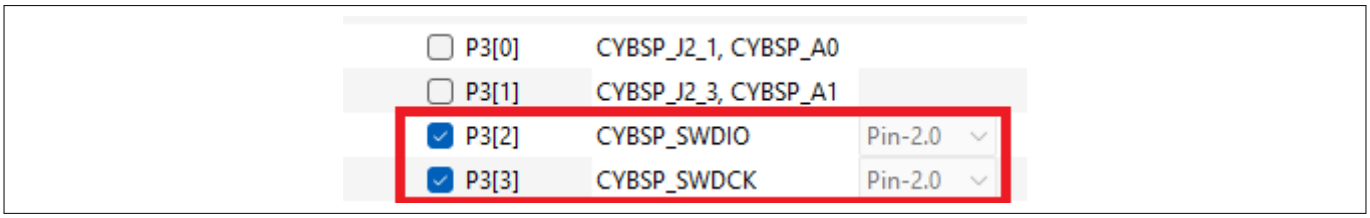


**図 10 GPIO へのデバイスモード変更**

ピンを GPIO モードに設定してデバッグインターフェースを無効にしている時、PSOC™ MCU デバイス内のデバッグコントローラーにアクセスするためにリセットを行う必要があります。つまり、ユーザーができない唯一のことは、実行中のプロジェクトにデバッガを接続することです。ピンが SWD モードのとき、消費電力を最小限にするため、低消費電力モードに遷移する前に次のことのいずれかを行う必要があります。外部コンポーネントを使ってピンをプルアップ/プルダウンする、またはピンモードを High-Z に変更。

すべてのリリースされた製品に対して、デバッグインターフェースピンを GPIO モードに設定することを推奨します。プログラミングとデバッグの詳細については、デバイスのデータシートおよび TRM を参照してください。

# 7 サンプルコード

次のプロジェクトは、ModusToolbox™ ソフトウェアと PSOC™ Creator の両方の低消費電力モードのさまざまな側面を示すために設計されました。

## 7.1 ModusToolbox™ ソフトウェアサンプルコード

ModusToolbox™ ソフトウェア環境は、以下のさまざまなプロジェクトに対応しています。mtb-example-psoc4-power-modes, mtb-example-psoc4-wakeup-from-deepsleep-using-low-power-comparator, mtb-example-psoc4-msclp-capsense-low-power など。これらのプロジェクトは、ModusToolbox™ ソフトウェアでサポートされているすべてのデバイス (該当する場合) をサポートしています (現在、4000S, 4100S, 4100S Plus, 4100S Plus 256k, 4100S Max, 4000T, 4100T Plus, および 4500S)。

ModusToolbox™ ソフトウェアでこれらのプロジェクトを開くには、ModusToolbox™ ソフトウェアの **Quick panel** で **New application** を選択します。ボードサポートパッケージ (BSP) のリストから選択できます。BSP は、使用されている特定のデバイスに対応しています。BSP を選択すると、その BSP に対応するサンプルコードのリストを選択できます。mtb-example-psoc4-power-modes を選択できます。

## 7.2 PSOC™ Creator プロジェクト

このアプリケーションノートには、すべての PSOC™ 4 MCU デバイス用の 3 つのプロジェクトと、PSOC™ 4100PS 用の 1 つのプロジェクトが含まれており、このアプリケーションノートで説明されている概念のいくつかを示します。各サンプルのすべてのソースコードは、`main.c` ファイルに含まれており、コメントが多く含まれています。時間をかけて各サンプルのソース コードを確認し、基本的な操作を学ぶことを推奨します。

PSOC™ Creator プロジェクトのデフォルトのターゲットデバイスは CY8C4245AXI-483 デバイスであり、CY8CKIT-042 および CY8CKIT-049 キットで使用されます。プロジェクトは、他の PSOC™ 4 MCU デバイスを使用するキットに簡単に適合させられます。外部コンポーネントに接続するピンとデバイスの選択を変更する必要がある場合があります。接続とデバイス タイプについては、特定のキットのドキュメントを確認してください。

他の PSOC™ 4 MCU デバイス用にプロジェクトを設定する手順は次のとおりです。

1. PSOC™ Creator でプロジェクトを開いてください。
2. 図 11 に示すように、コンテキスト メニューから **Device selector** オプションを選択してください。

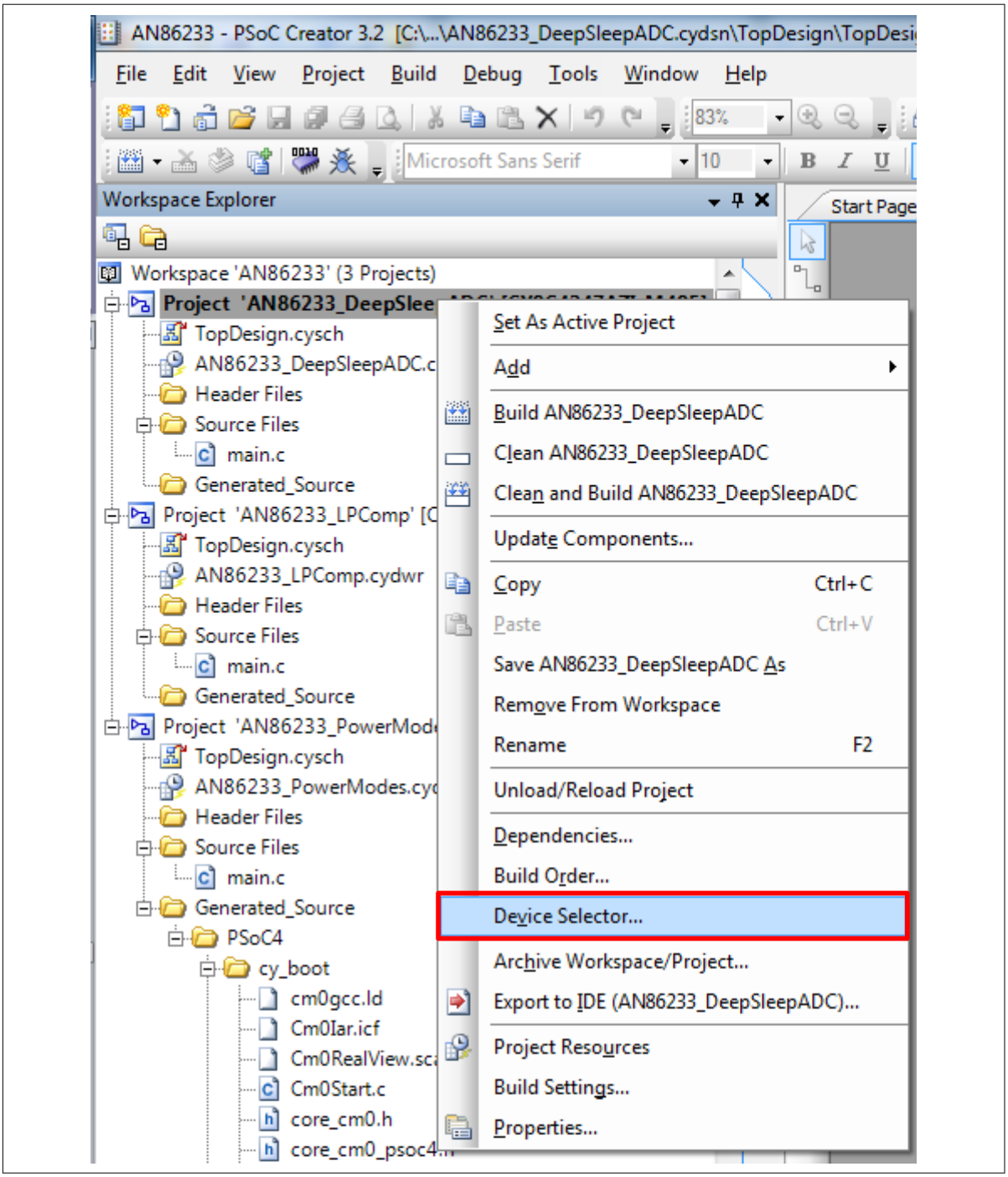


**図 11　　Device Selector オプションの選択**

3. 図 12 に示すように、表示されるポップアップメニューから適切なデバイスを選択します。

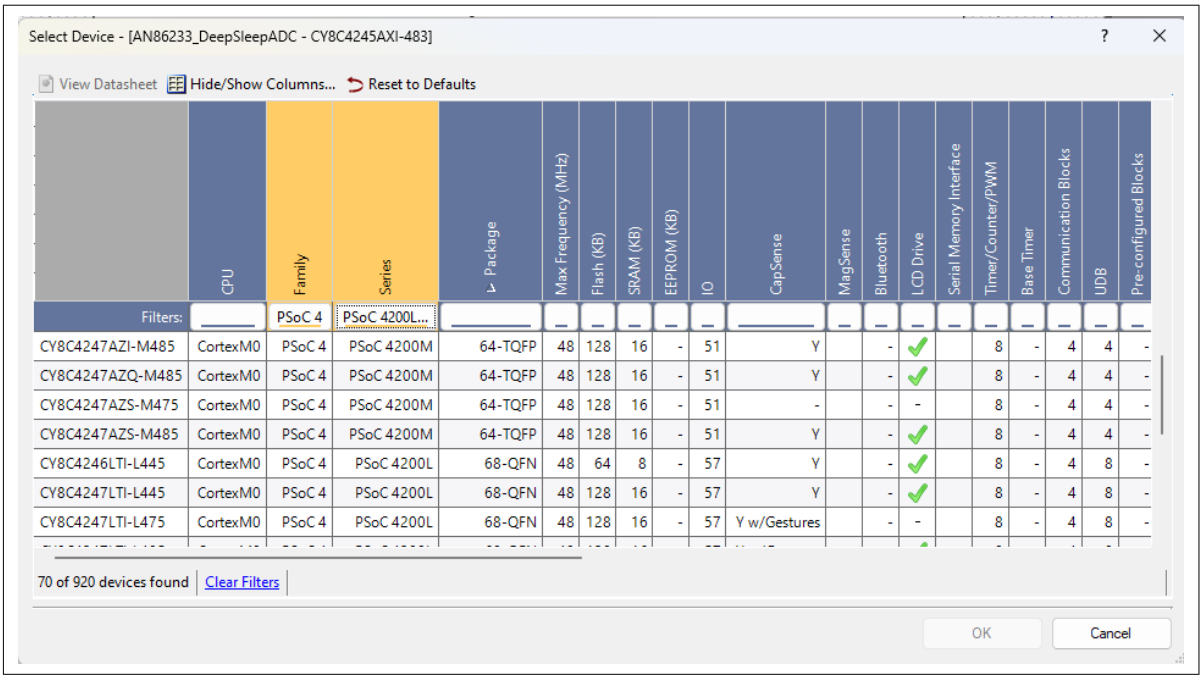


**図 12　　Device Selector ウィンドウ**

## 7.3　PSOC™ 4 MCU サンプルコード

### 7.3.1　プロジェクト 1: 消費電力モード

**ModusToolbox™ソフトウェア:** このサンプルコードは、GitHub の mtb-example-psoc4-power-modes にあります。すべてのスリープモードとディープスリープモードをアクティブモードと簡単に比較できます。

デバイスの消費電力モードを変更するには、オンボードボタンを 2 秒以上長押して、デバイスをディープスリープモードにします。デバイスをスリープモードに切り替えるには、200 ミリ秒から 2 秒の間にボタンを押します。これにより、デバイスで消費電力モードを簡単に切り替えて、比較できるようになります。モードを切り替える方法については、図 13 を参照してください。

## 7 サンプルコード

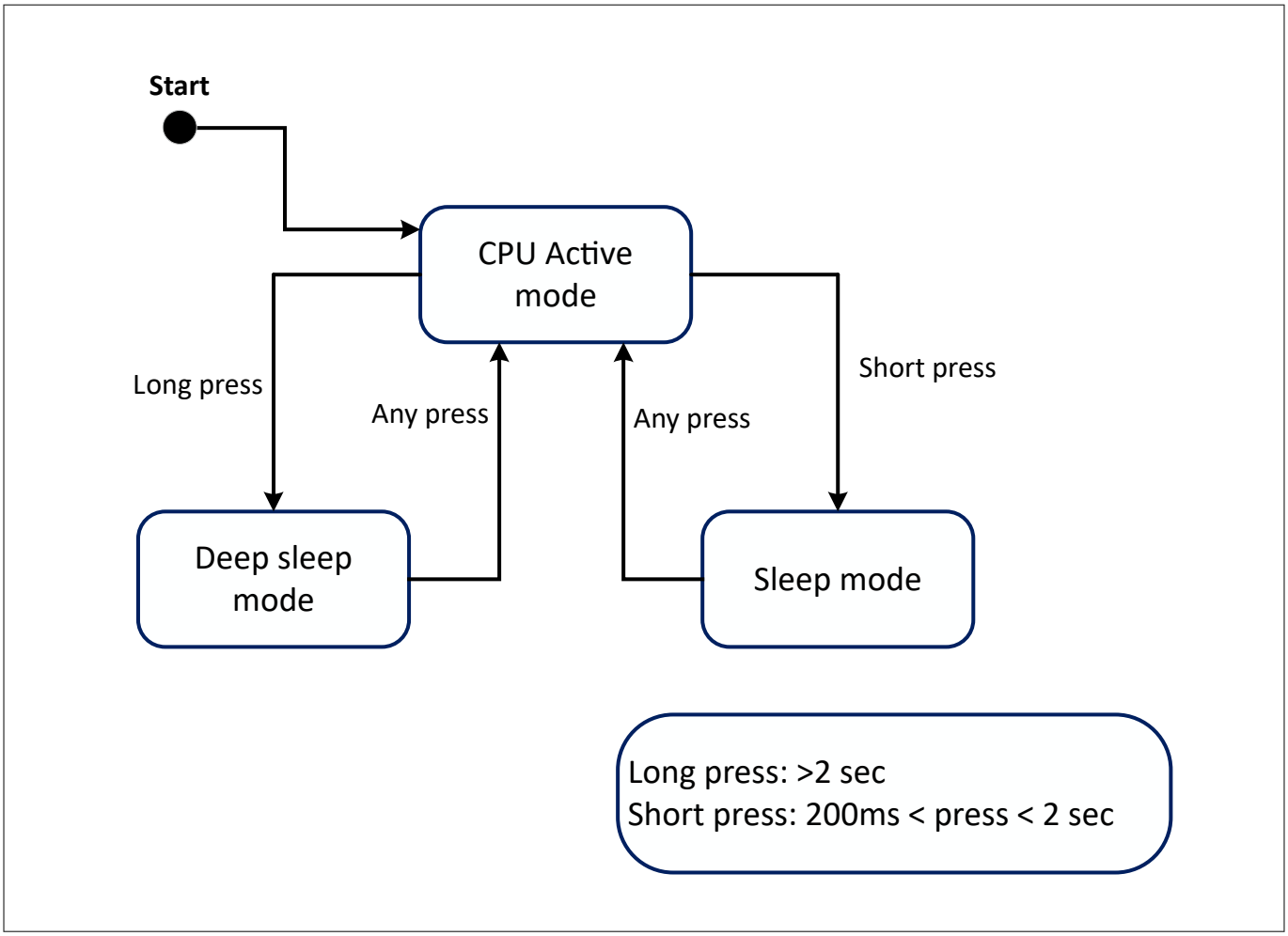


**図 13　　消費電力モードの切り替え**

PSOC™ 4 開発キットで電流を測定できます。詳細については、キットガイドを参照してください。

**PSOC™ Creator:** プロジェクト (AN86233_Power Modes) は、このアプリケーションノートに関連付けられた zip file で入手できます。このプロジェクトでは、4 つの低消費電力モードすべてをアクティブモードと簡単に比較できます。コードを 1 か所で編集して、PM_TO_DEMO 定数を目的の消費電力モード定数に設定できます。次のプログラムコードに、ディープスリープモードが選択されているコードの例を示します。

```
/* Power Modes Constants */ #define PM_SLEEP 1u #define PM_DEEP_SLEEP 2u #define PM_HIBERNATE
3u #define PM_STOP 4u /* Set PM_TO_DEMO to power mode */ #define PM_TO_DEMO PM_DEEP_SLEEP
```

この選択が行われた後、プロジェクトをビルドし、デバイスをプログラムします。プロジェクトは、開発ボードのスイッチが押されるまでアクティブモードで開始されます。消費電力モード定数に割り当てられた回数に応じて、LED がすばやく点滅します。この例では、ディープスリープモードで LED が 2 回点滅します。

スイッチが再度押されるまで、プロジェクトは低消費電力モードのままです。次に、同じ回数ゆっくり点滅して、プロジェクトがアクティブモードに戻ったことを通知します。電流計を使用すると、各モードの電流を測定できます。他の消費電力モードを評価するためには、PM_TO_DEMO を他の消費電力モード定数に設定し、プロジェクトをリビルドして、PSOC™ 4 MCU デバイスを再プログラムします。

*注: ハイバネートモードとストップモードのないデバイスでは、別のプロジェクト (AN86233_PSoC_4100PS) を使用できます。これは zip ファイルで入手できます。ハイバネートモードまたはストップモードをサポートしないデバイスは、PSOC™ 4000, 4000S, 4100S, 4100S Plus, 4100S Plus 256k, 4100S max, 4200DS, 4500S, 4700S, 4100PS, 4000T, 4100T Plus, およびアナログコプロセッサ製品です。*

### 7.3.2 プロジェクト 2: 低消費電力コンパレータ

**ModusToolbox™ソフトウェア:** このサンプルコードは、GitHub の PSOC™ 4: Wakeup from Deep Sleep using low-power comparator に掲載されており、LPComp ペリフェラルを低消費電力モードで動作させる方法を示しています。

PSOC™ Creator プロジェクト (AN86233_LPComp) は、このアプリケーションノートに関連付けられている zip file で入手できます。

このプロジェクトでは、低消費電力コンパレータ (LPComp) を使用して、PSOC™ 4 MCU デバイスをハイバネートモードから復帰させ、アクティブ モードに戻す方法を示します。プロジェクトを完了するには、2 つの 1 MΩ の外部抵抗とポテンショメータ、または電圧源を開発キットに追加する必要があります。2 つの抵抗は分圧器を形成して、$V_{DDD}$ の半分の基準電圧を提供します。

ポテンショメータは、割込みを発生させることにより、ハイバネートからアクティブ モードに戻るトリガーとなる外部変化電圧をシミュレートします。ポテンショメーターの電圧がリファレンスより高い電圧からリファレンスより低い電圧に、またはその逆に遷移すると、割込みが発生します。

リセット後、プロジェクトはアクティブモードになります。プロジェクトがアクティブモードのときはいつでも LED が点灯し、ハイバネートモードでは消灯します。キットボタンを押すと、入力電圧が基準電圧 (この場合は $V_{DDD}/2$) の上から下に、またはその逆に変化していることを LPComp が検出するまで、PSOC™ 4 MCU デバイスをハイバネートモードにします。

キットボタンが押されたときにハイバネートモードになる代わりに、デバイスがスリープまたはディープスリープになるようにコードを変更できます。操作は同じである必要がありますが、消費電力は低消費電力モード間で異なります。

### 7.3.3 プロジェクト 3: ディープスリープ ADC

このプロジェクトは、PSOC™ Creator のみで機能します。

**PSOC™ Creator:** プロジェクト (AN86233_DeepSleepADC) は、このアプリケーションノートに関連付けられている zip file で入手できます。

このプロジェクトは、周期的にウェイクアップし、SAR ADC で読み出しを行い、データを変換し、データをシリアルポート (UART) から送信し、ディープスリープモードに戻す方法の例です。このシーケンスは毎秒 1 回実行されますが、サンプル間の時間の 99.9%はディープスリープモードのままであり、消費電力を約 3 桁も低減させます。

このプロジェクトでは、ILO と WDT (WDT0) を使用して、毎秒 1 回ディープスリープモードからウェイクアップします。ILO は低消費電力発振器ですが、32Hz −50%～+100%の仕様で、あまり正確ではありません。一方、IMO は ±2%とはるかに正確ですが、より多くの消費電力を必要とし、ディープスリープモード中は動作しません。PSOC™ 4 MCU デバイスがアクティブモードのときに常に動作します。IMO を使用して、システム ILO 機能で ILO を調整できます。

ILO トリムは、WDT0 カウンター周期を±10%以内になるように調整するために使用されます。ILO の大きなドリフトは主に電圧と温度の関数であるため、トリムを頻繁に調整する必要はありません。この例では、コンポーネントがディープスリープモードから復帰する 60 回ごと、または毎分 1 回トリミングされます。これは、60 秒の固定更新から、測定されたダイ温度または電源電圧が設定値を超えて変化するたびに変更できます。サンプルプロジェクトのプログラムフローについては、図 14 を参照してください。

*注: main.c ファイルの WDT_IRQN マクロを使用して、ターゲットデバイスに応じたプロジェクトで、WDT 割込みベクタ番号を更新する必要があります。ベクタ番号 (IRQx) については、対象デバイスの architecture reference manual の「Interrupts」章を参照してください。*

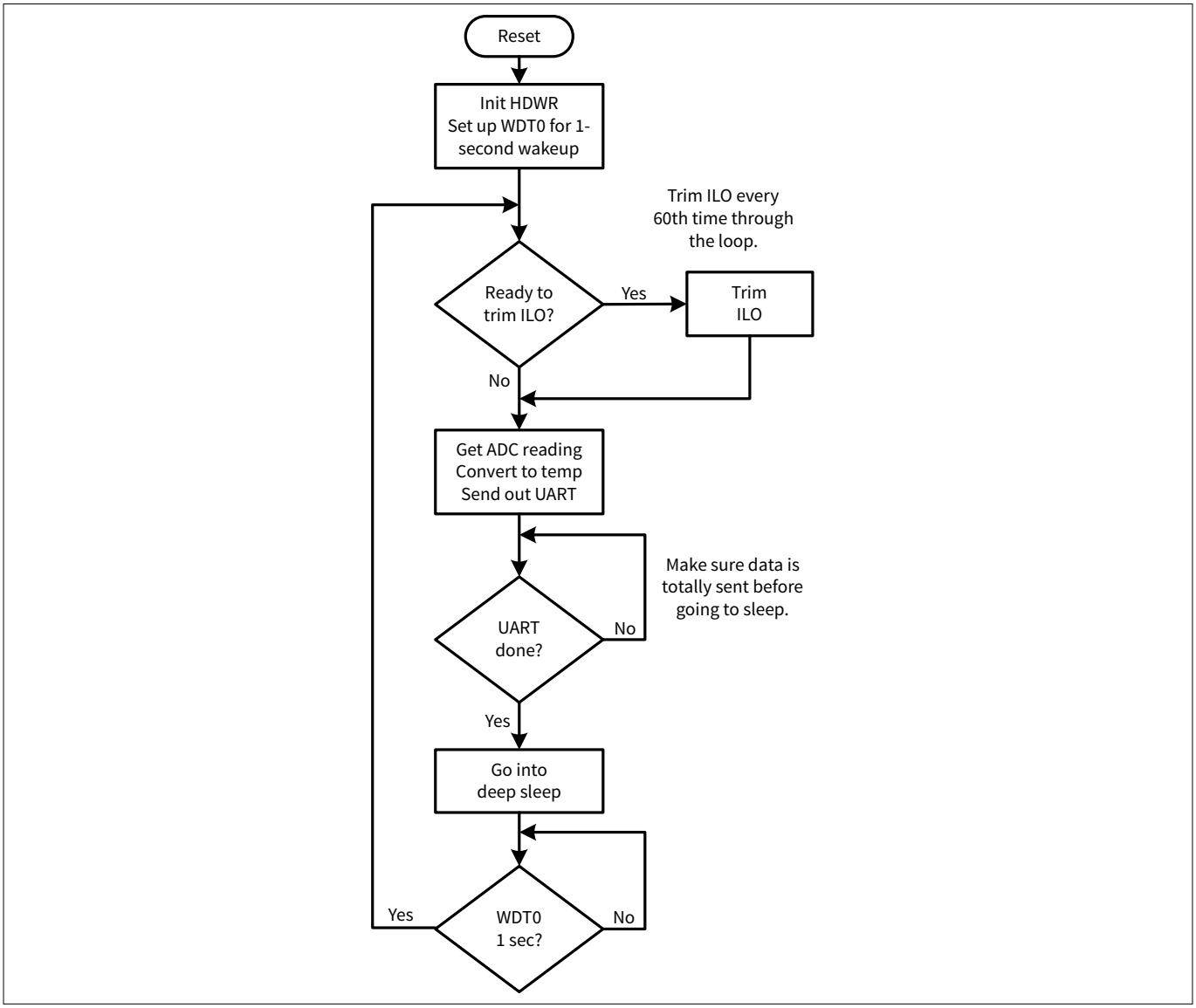


**図 14 AN86233_DeepSleepADC プログラミングフロー**

### 7.3.3.1 平均電力

本例の平均消費電力を判定するためには、それぞれのモードでの時間と電流の両方を測定する必要があります (電力モードのデューティサイクルが同じであることを前提としています)。このファームウェアの周期は約 1 秒です。したがって、PSOC™ 4 MCU デバイスが 1 周期内にアクティブにある時間とディープスリープモードにある時間を合計すると、約 1 秒となります。

まず、消費電流およびデバイスがアクティブモードにある時間を測定します。図 15 に示すように、デバイスは約 630 µs アクティブ モードにあり、その間約 5000 µA (5.0 mA) の電流を消費します。図 15 に示す UART 通信を見ると、UART が通信を完了するまでは PSOC™ 4 MCU デバイスはディープスリープに戻らないことが分かります。LED 駆動 (アクティブ LOW) 信号の継続時間および電流測定により、デバイスがアクティブモードとディープスリープモードのどちらにあるかを判定します。ディープスリープ モードでは、電流は約 3.5 µA で、信号継続時間は約 999.4 ms です。

## 7 サンプルコード

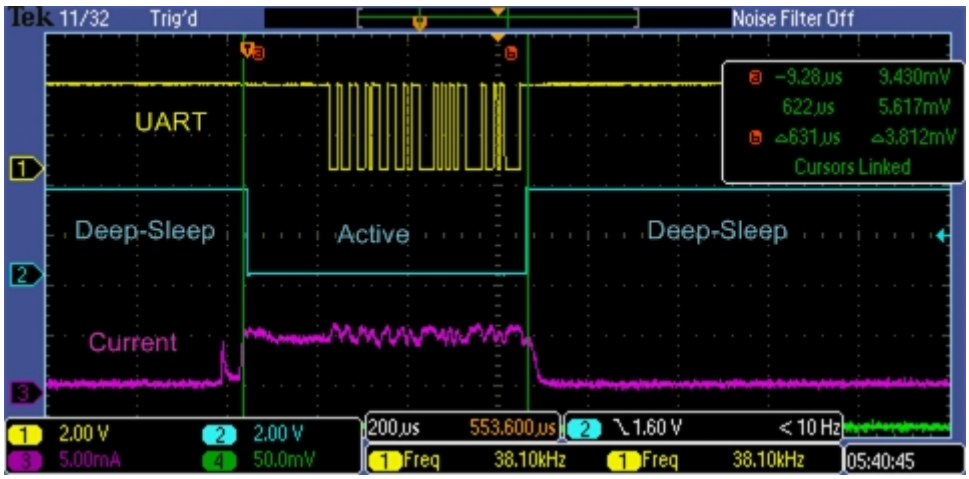


**図 15　　プロジェクトのアクティブモードパラメータの測定**

表 4 に示すように、アクティブモードおよびディープスリープモードでの消費電力を平均すると、低消費電力のディープスリープモードを使用すると、5 mA のアクティブ モード電流を 6.7 µA の平均電流に、約 3 桁も低減できます。

**表 4　　平均消費電力の計算**

| | ディープスリープ | アクティブ | 合計 | 単位 |
|---|---|---|---|---|
| 時間 (ms) | 999.4 | 0.63 | 1000 | ms |
| 電流 (µA) | 3.5 | 5000 | – | µA |
| 時間× I | 3498 | 3150 | 6648 | ms × µA |
| 平均電流 | 6.7 | | | µA |

総体的に見ると、低消費電力モードを使用せずにプロジェクトを 300 mAh のバッテリで実行すると、バッテリの持続時間は約 60 時間 (300mAh/5mA) となります。消費電力モード間の切り替えを利用すると、300 mAh バッテリの持続時間は約 44,800 時間、5 年以上となります。

もちろん、アクティブモードとディープスリープモードのデューティサイクル時間は平均消費電力に大きく影響しますが、各モードでの消費電力にも影響します。2 つの消費電力モードを切り替えると、デザインが少し複雑になりますが、バッテリ駆動アプリケーションにとっては大きな利益になります。

# 8 ModusToolbox™ ハードウェア設定

ModusToolbox™ ハードウェア設定については、サンプルコードの README を参照してください。ModusToolbox™ サンプルコードの README は、すべてのハードウェアをセットアップおよび設定する方法を示しています。

# 9 PSOC™ Creator ハードウェア設定

インフィニオンは、PSOC™ 4200 用の CY8CKIT-042, PSOC™ 4200M 用の CY8CKIT-044, PSOC™ 4200L MCU 用の CY8CKIT-046 など、PSOC™ MCU ファミリ用のさまざまな開発キットを提供しています。また、PSOC™ 4200 用の CY8CKIT-049, PSOC™ 4200M 用の CY8CKIT-043, PSOC™ 4100PS MCU 用の CY8CKIT-147 など、フォームファクタが小さい低コストのキット (プロトタイピングキット) も利用できます。

これらのキットはさまざまな機能を提供しますが、すべてに PSOC™ 4 MCU デバイスに接続されたユーザボタンと LED が含まれています。プロジェクトが正しく機能するには、キットごとに適切なピン割り当てを行う必要があります。表 5 に、PSOC™ パイオニアキット全体のピン配置を示します。表 6 に、PSOC™ 4 MCU プロトタイピングキット全体のピン配置を示します。

**表 5 PSOC™ 4 MCU パイオニアキットのピン配置**

| 機能 | CY8CKIT-042 (PSOC™ 4200) | CY8CKIT-044 (PSOC™ 4200M) | CY8CKIT-046 (PSOC™ 4200L) |
|---|---|---|---|
| 赤色 LED (アクティブ LOW) | P1[6] | P0[6] | P5[2] |
| スイッチ (アクティブ LOW) | P0[7] | P0[7] | P0[7] |
| UART Tx[1)] | P4[1] | P7[1] | P3[1] |

1) 他の互換性のあるピンも UART Tx に接続するために使用できます。CY8CKIT-044 および CY8CKIT-046 では、表にリストアップされているピンは基板搭載の KitProg USB-UART ブリッジに配線されています。

**表 6 PSOC™ MCU プロトタイピングキットのピン配置**

| 機能 | CY8CKIT-049 (PSOC™ 4200) | CY8CKIT-043 (PSOC™ 4200M) | CY8CKIT-147 (PSOC™ 4100PS) |
|---|---|---|---|
| 青色 LED (アクティブ HIGH) | P1[6] | P1[6] | P0[2] |
| スイッチ (アクティブ LOW) | P0[7] | P0[7] | P0[3] |
| UART Tx[1)] | P4[1] | P7[1] | P0[5] |

1) 他の互換性のあるピンも UART Tx に接続するために使用できます。CY8CKIT-049 および CY8CKIT-043 では、表にリストアップされているピンは基板搭載の USB-UART ブリッジに配線されています。

## 9.1 CY8CKIT-042 キット (PSOC™ 4200)

このキットには、多くの一般的な Arduino シールド基板と互換性がある基板搭載デバッガおよびコネクタを備えています。本アプリケーションノートおよび電流測定に関するこのキットの最も重要な機能は、電源ジャンパ J13 です。PSOC™ 4 MCU デバイスの電流を測定するためには、このジャンパを取り外し、ピン間に電流計を配置します。これにより、基板の残りの部分から独立して PSOC™ 4 MCU デバイスの電流のみを測定できます。この基板には、これらの例を実行するために必要な LED とボタンも備えています。

基板のもう 1 つの機能は、USB-シリアルポートアダプタです。図 16 に示すように、PSOC™ 4 MCU P4[1]から J11 ピン 9 をワイヤで接続することにより、お気に入りのターミナルエミュレータを使用して AN86233_DeepSleepADC プロジェクトのシリアル出力を監視できます。ボーレートは 115.2 kbps, 8 ビットデータ, パリティなしです。USB-シリアル ポート機能の使用の詳細については、CY8CKIT-042 – PSOC™ 4 Pioneer Kit guide を参照してください。

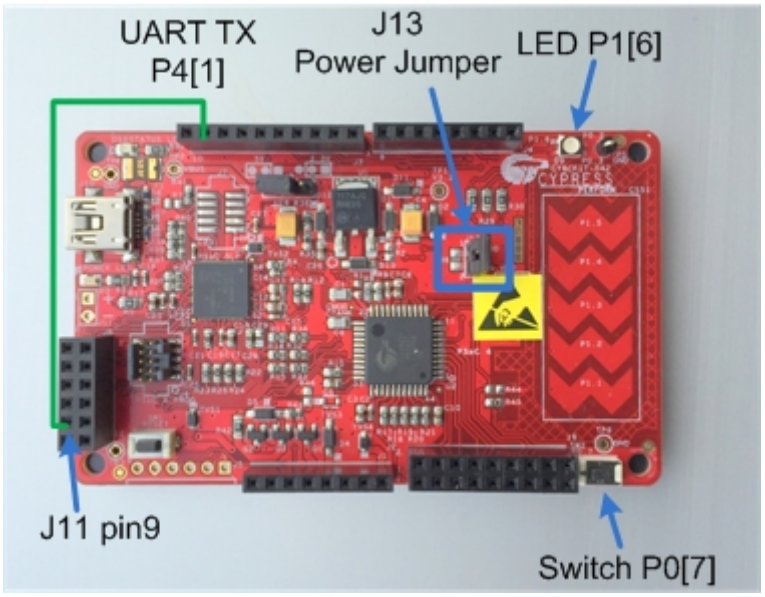


**図 16　　CY8CKIT-042 キット ハードウェア**

CY8CKIT-044 と CY8CKIT-046 キットのスイッチ, LED, および UART Tx ラインの接続は CY8CKIT-042 と同じです。これらのキットの詳細情報については、該当ウェブページに搭載されているキット ガイドを参照してください。

## 9.2　　CY8CKIT-049-42xx キット (PSOC™ 4200)

これらの低コストキットはすべての PSOC™ 4 MCU ピンへのアクセスを提供し、CY8CKIT-042 と互換性のあるユーザー LED およびスイッチを備えています。電源は、側面にある「P4VDD」ピンから、または端にある USB コネクタを介して供給されます。図 17 にキットハードウェアを示します。

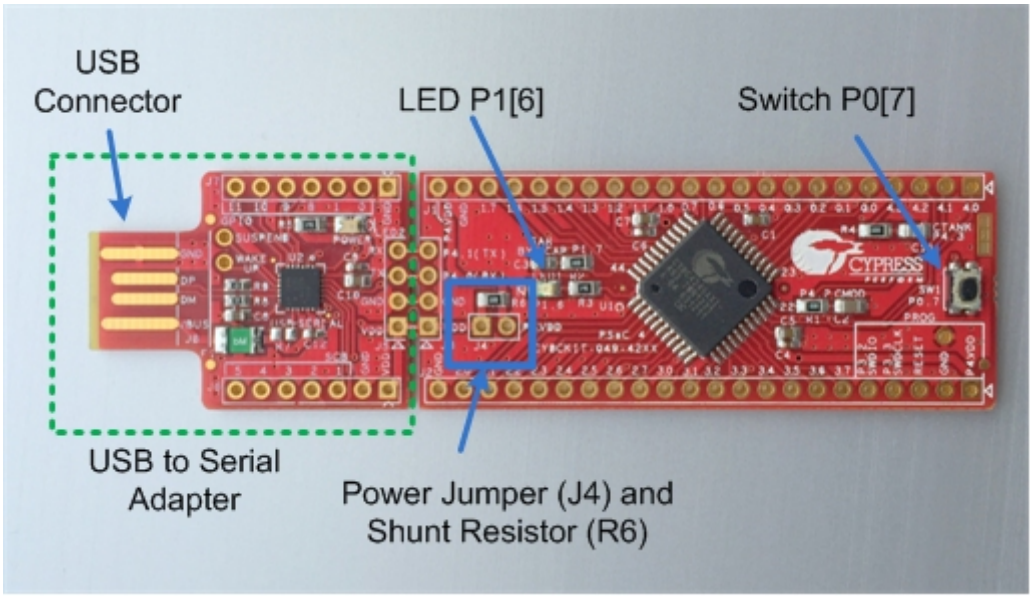


**図 17　　CY8CKIT-049-42xx キット ハードウェア**

このキットはデバッガを搭載していませんが、プロジェクトのブートロードに使用される USB-シリアル アダプタを備えています。まず、このインターフェースを使用するためにブートロード可能なコンポーネントを含むようにプロジェクトを変更する必要があります。MiniProg3 プログラムおよびデバッグキット (CY8CKIT-002) を使用する場合、

それを基板の右下隅にあるユーザーインストールのコネクタに直接差し込めます。ブートロード用に USB-シリアル アダプタを使用する方法については、ウェブサイトからダウンロードできるキットの資料を参照してください。

キットが USB ポートから電源供給されている時にキットでのプロジェクト電流を測定するために、0Ω 抵抗 R6 を取り外し、電流計やデジタルマルチメーター (DMM) に接続可能な J4 にピンまたはワイヤを追加します。

USB-シリアル アダプタはブートローダ インターフェースおよびシリアルポート アダプタを提供しており、サンプル プロジェクト AN86233_DeepSleepADC の出力を表示します。

## 9.3 CY8CKIT-043 キット (PSOC™ 4200M)

このキットは基板搭載デバッガを備え、CY8CKIT-049 と同じフォームファクターを持っています。電源は側面にある「P4VDD」ピンから、または端にある USB コネクタを介して供給されます。キットが USB ポートから電源供給されている時にキットでのプロジェクト電流を測定するために、0Ω 抵抗 R22 を取り外し、電流計や DMM に接続可能な J4 にピンまたはワイヤを追加します。図 18 にキットハードウェアを示します。

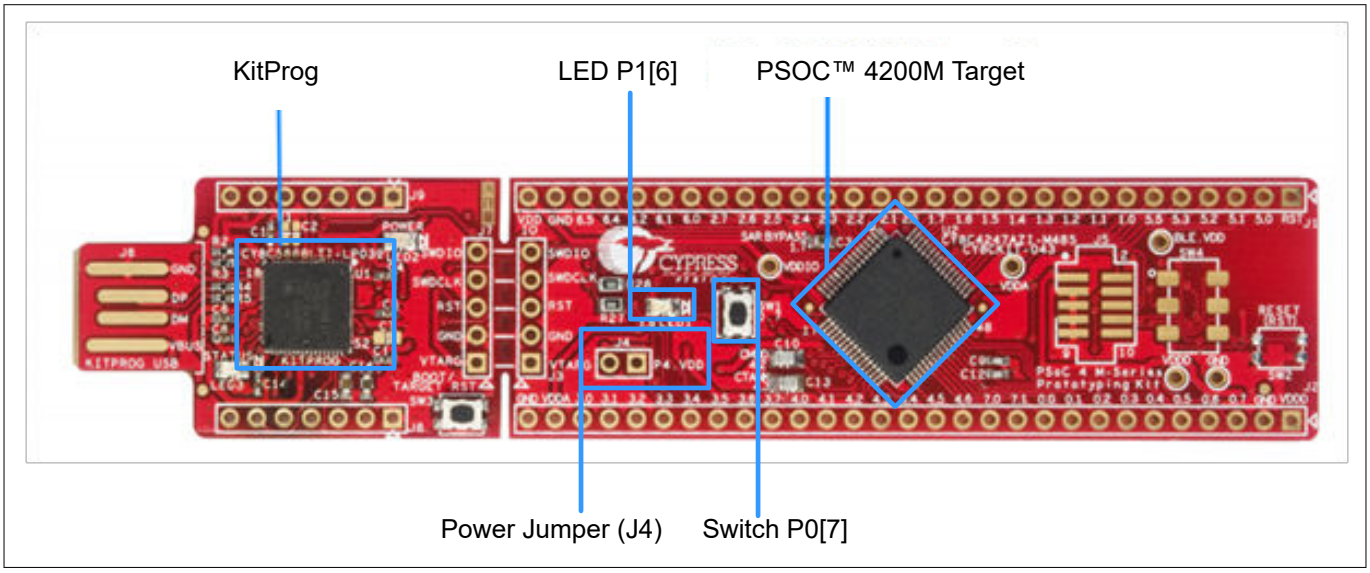


**図 18 CY8CKIT-043 ハードウェア**

この基板はこれらの例をデモするために必要な LED およびスイッチも備えています。もう 1 つの機能は USB-シリアルポートアダプタです。特殊なハードウェア接続を行わず、お気に入りの端末エミュレータを使って AN86233_DeepSleepADC プロジェクトのシリアル出力を監視できます。ボーレートは 115.2 kbps, データは 8 ビット, パリティ無しです。USB-シリアルポート機能の使用の詳細については、KitProg user guide を参照してください。

# 10 DMM による電流測定

DMM を使用して電流を測定する場合、DMM 内のシャント抵抗の値を知ることが重要です。DMM には、電流入力間に 1 つ以上の (シャント) 抵抗があります。これらの抵抗器の範囲は、1 Ω 未満～10 kΩ を超える場合があります。同じベンダーからのシャント抵抗でも、ブランドやモデルによって標準値が異なることがあります。これらのシャント抵抗をとおして常に電圧降下が発生するため、ご使用のメーターのマニュアルを参照し、シャント抵抗の値を確認することが重要です。これは、PSOC™ 4 MCU デバイスが受ける電圧は供給される電圧とは違うことを意味します。

メーターのシャント抵抗が 1 Ω 以下の場合、PSOC™ 4 MCU の電流を測定するとき、無視できるわずか数 mV の電圧降下があります。あるベンダーが低電流測定に使用するシャント抵抗が 1 kΩ の場合、1 mA の電流は 1 V の低下になります。また、測定電流範囲を変更するときは、ご使用の DMM がブレークしないように注意してください。そうしないと、電源が入れ直され、プロジェクトがリセットされます。

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の非常に低い電流の場合、デバイスが低消費電力モードに入るまで、ゼロまたは低抵抗のシャントを使用することを推奨します。低消費電力モードに入った後、コードはデバイスをそのモードに保ち、電流測定のために高抵抗シャントに切り替える必要があります。

DMM のシャント抵抗に依存しないために、両キットはシャント抵抗を取り付ける場所を含んでいます。CY8CKIT-049-42xx は 0 Ω シャント抵抗 (R6) を備えています。これは小さい値の抵抗で置き換え、電圧計を使ってシャント抵抗の電圧を測定できます。このように、電流を簡単に判定できます。1 Ω～100 Ω のシャント抵抗はほとんどのアプリケーションで使用できます。CY8CKIT-042 にも J13 の隣にシャント抵抗 (R6) 用の場所が含まれています。

## 10.1 消費電力量の概算

デバイスデータシートおよび PSOC™ Creator コンポーネントデータシートは、特定のプロジェクトの消費電力を見積もるために十分な情報を提供しています。このプロセスを簡単にするために、幅広い内部コンポーネントの一般的な消費電力要件を記載したスプレッドシートを提供しています。本スプレッドシート `PSoC4_Power_Estimator.xlsx` は AN86233.zip の中にあります。プロジェクトによって異なるため、本スプレッドシートが提供した消費電力計算はあくまでも目安ですが、デザインを完了する前の意味のあるフィードバックとしては十分近い数字のはずです。スプレッドシートにいくつかのタブがあります。データを入力する前に、「Instructions」タブを読むことを忘れないでください。

# 11 サンプルの再利用

前のセクションで説明したサンプルコードは、さまざまな PSOC™ 4 MCU デバイス, キット, またはその両方に移植できます。サンプルコードを移植する前に、次の点に注意してください。

1. PSOC™ 4 MCU デバイスは、すべて同じブロックを共有するわけではありません。
2. ピン配置はキットごとに異なります。一部のピンは移動する必要がある場合があります。PSOC™ Creator の Pin layout タブを参照してください。

コードを新しいデバイスに移植するためには、PSOC™ Creator で **Project** > **Device** selector を選択し、ターゲットデバイスに変更します。場合によっては、サンプルコードで使用されるリソース (IP ブロックなど) が別のデバイスでサポートされていないことがあります。その場合、サンプルは機能しません。このようなデバイスを対象としたコードを作成すると、エラーが発生します。特定のデバイスがサポートするものについては、デバイスのデータシートを参照してください。

# 12 まとめ

消費電力は、優れたアイデアと成功したデザインとの差異です。PSOC™ 4 MCU で使用可能な多くの省電力機能をうまく利用できれば、ご自身の設計を最適化し、最低の電力消費量を確保できます。

# 13 関連アプリケーションノート

これらのアプリケーションノートは、本書の範囲内では十分説明できていなかったトピックに関する詳しい情報を提供します。

**[1]** AN79953 – PSOC™ 4 MCU 入門

**[2]** AN85951 – PSOC™ 4 および PSOC™ 6 MCU CAPSENSE™ デザインガイド

**[3]** AN86439 – PSOC™ 4 MCU – GPIO ピンの使用

**[4]** AN77900 – PSOC™ 3 と PSOC™ 5LP の低電力モードおよび電力低減技術

**[5]** AN234231 – PSOC™ 4 CAPSENSE™ ultra-low-power capacitive sensing techniques

# 改訂履歴

| 版数 | 発行日 | 変更内容 |
|---|---|---|
| - | - | 本版は英語版のみの発行です。英語版の改訂内容: New application note |
| ** | 2013-09-23 | これは英語版 001-86233 Rev. *A を翻訳した日本語版 001-89291 Rev. **です。英語版の改訂内容: Updated links<br>Updated example project section |
| - | - | 本版は英語版のみの発行です。英語版の改訂内容: Replaced example project with three other projects<br>Added typical current and wakeup times<br>Updated content on UnFreezeIo() and FreezeIo() APIs<br>Updated low-power APIs<br>Updated the CY8CKIT-049-42xx キット (PSOC™ 4200) section<br>Included power calculation spreadsheet<br>Many minor changes<br>Sunset update |
| - | - | 本版は英語版のみの発行です。英語版の改訂内容: Clarified retention of UDBs when exiting deep sleep and hibernate in Table 2.<br>Cleaned up several minor typos |
| *A | 2015-11-20 | これは英語版 001-86233 Rev. *D を翻訳した日本語版 001-89291 Rev. *A です。英語版の改訂内容: Updated the document for PSOC™ 4200L MCU<br>Updated the power calculation spreadsheet |
| - | - | この版は英語版のみです。英語版の改訂内容: Updated the example projects to PSOC™ Creator 3.3 SP1. |
| *B | 2016-05-06 | これは英語版 001-86233 Rev. *F を翻訳した日本語版 001-89291 Rev. *B です。英語版の改訂内容: Updated the document for PSOC™ analog coprocessor<br>Updated the power calculation spreadsheet |
| *C | 2017-04-26 | Updated logo and copyright. この後に英語版の改訂あり。英語版の改訂内容: Updated projects to PSOC™ Creator 4.1 update 1<br>Removed the ILO calibration component from AN86233_DeepSleepADC project<br>Updated template |
| *D | 2018-09-07 | これは英語版 001-86233 Rev. *H を翻訳した日本語版 001-89291 Rev. *D です。英語版の改訂内容: Added references to PSOC™ 4100PS throughout the document.<br>Updated power estimator spreadsheet and PSOC™ Creator projects |
| *E | 2018-12-13 | 変更なし。Sunset Review を実施。この後に英語版の改訂あり。英語版の改訂内容: Added support for ModusToolbox™ software and the peripheral driver library<br>Updated to Infineon template |
| *F | 2022-06-15 | これは英語版 001-86233 Rev. *I を翻訳した日本語版 001-89291 Rev. *F です。 |

## 改訂履歴

| 版数 | 発行日 | 変更内容 |
|---|---|---|
| *G | 2026-08-10 | これは英語版 001-86233 Rev. *K を翻訳した日本語版 001-89291 Rev. *G です。英語版の改訂内容: Added support for PSOC™ 4000T and PSOC™ 4100T Plus<br>Updated code example links<br>Updated 表 1<br>Changed section 7.3.1 title from 'Power modes code example' to 'Project 1: Power modes'. |

# Trademarks

Bluetooth® のワードマークおよびロゴは、Bluetooth SIG, Inc.が所有する登録商標であり、インフィニオンはこれらのマークをライセンスに基づいて使用しています。

PSOC™ (旧称: PSoC™) は、インフィニオン テクノロジーズの商標です。本文書またはその他の文書において PSoC™ と記載されている箇所は、すべて PSOC™を指すものとみなされます。

**Trademarks**

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

**Edition 2026-08-10**

**Published by**

**Infineon Technologies AG**

**81726 Munich, Germany**

**© 2026 Infineon Technologies AG**

**All Rights Reserved.**

**Do you have a question about any aspect of this document?**

**Email: erratum@infineon.com**

**Document reference**

**IFX-cdk1741254842696**

**重要:注意事項**

製品(サンプルも含みます。また、ハードウェア、ソフトウェアまたはその両方で構成される製品も含みます。以下同じです。)の販売、提供または引渡しは、インフィニオンテクノロジーズ AG およびその関連会社(以下まとめて「インフィニオン」といいます)、お客様とインフィニオンによって締結された基本契約その他の書面により合意された条件、そのようなの条件がない場合はインフィニオンの適用ある販売条件に従います。お客様の一般的な取引条件またはインフィニオンの販売条件と乖離した条件は、インフィニオンが書面で明示的に同意した場合に限り、インフィニオンを拘束するものとします。

疑義を避けるため、インフィニオンは、第三者の権利を侵害していないことの保証、および特定の使用/目的に対する適合性または商品性の保証などの黙示的保証を負いません。

インフィニオンは、サンプル、アプリケーション、もしくはお客様による製品の具体的な使 用方法、または本書面に記載された例や標準値に関するいかなる情報についても責任いません。

本書に「Preliminary(予備)」「Target(目標)」「Draft(草案)」または「Proposal(提案)」と表示されている場合、インフィニオンは事前の通知なく、本書に記載されたすべての情報をいつでも変更する権利を留保します。インフィニオンによる量産向け製品の開発およびリリースを条件として、当該製品の技術仕様は、そのような表示のないインフィニオン提供の関連するデータシートに定めれます。

本書面に記載されているデータは、技術的資格を有する熟練したお客様のご担当者をのみを対象としています。お客様は、意図された用途およびお客様の特定用途に対する製品の適合性を評価し、意図された用途および顧客の特定用途において本書面に含まれるすべての関連技術データを検証する責任を負います。お客様は、意図されたアプリケーションの機能性および安全性を適切に設計、プログラミング、テストし、その使用に関連する法令を遵守してください。

インフィニオンの明示的な承認がない限り、製品は、製品の故障またはその使用による結果が人身傷害につながると合理的に予想されるいかなる用途にも使用しないでください。ただし、上記は、インフィニオンが明示的に設計し、販売した使用分野でお客様が製品を使用することを妨げるものではありません。

インフィニオンは、適用される法律、例えばドイツ著作権法(UrhG)第 44b 条に従って、そのコンテンツを商業的なテキストおよびデータのマイニング(TDM)のために使用する権利を明示的に留保します。

製品にセキュリティ機能が含まれている場合:

いかなるコンピューティング・デバイスも絶対的に安全であることはあり得ないため、製品にセキュリティ対策が施されているにもかかわらず、インフィニオンは、製品に侵入、データの盗難もしくは損失、またはその他の侵害(「セキュリティ侵害」といいます)がないことを保証せず、インフィニオンは、セキュリティ侵害に起因するいかなる責任も負わないものとします。

本書面がソフトウェアを含んでいるまたはソフトウェアに言及している場合:

ソフトウェアは、米国、ドイツ、および世界各国の知的財産権法および条約に基づきインフィニオンが所有しています。インフィニオンはすべての権利を留保し、無断複写、複製および転写を禁じます。お客様はソフトウェアに同梱されたソフトウェア使用許諾契約に従ってのみソフトウェアを使用できます。

ソフトウェア使用許諾契約が適用されない場合、インフィニオンはここに、ソフトウェアの知的所有権に基づき、(a)ソースコード形式で提供されたソフトウェアについて、インフィニオンのハードウェア製品と共に使用する目的のためにのみソフトウェアを修正および複製するための、そして(b)インフィニオンのハードウェア製品上での使用目的に限り、ソフトウェアをバイナリコード形式でエンドユーザーに外部配布するための、個人的、非排他的、譲渡不能なライセンス(サブライセンス不可)をお客様に付与します。それ以外のソフトウェアの使用、複製、変更、翻訳またはコンパイルは禁止されています。製品、技術、納品条件、および価格に関する詳細については、最寄りのインフィニオンオフィスに連絡するか、www.infineon.com にアクセスしてください。