AN219528

PSoC® 6 MCU 低消費電力モードおよび節電技術

著者: Brian Lee

関連製品ファミリ: すべての PSoC 6 MCU 製品

関連サンプル コード: 詳細は、こちらをクリックしてください。

関連アプリケーション ノート: 完全な一覧については、こちらをクリックしてください。

その他のサンプル コードが必要な場合は、以下の通り対応いたします。

PSoC サンプル コードのリストにアクセスするには、サンプル コードのウェブページをご覧ください。
PSoC ビデオ ライブラリについてはここからご覧ください。

本アプリケーション ノート (AN219528) は、PSoC 6 MCU の電力を最適化する電力モードの使用方法について説明します。主な内容には、PSoC 6 MCU デバイスの低消費電力モードと、PSoC 6 MCU 機能を使用した電力管理手法を含みます。関連する 3 つのサンプル コードでは、異なる低消費電力技術を説明します。

目次

1 はじめに

PSoC 6 MCU は、低消費電力モードが他の節電機能と技術で組み合わされたら、重要な性能を犠牲にすることなく最大限に省電力の利点を引き出します。本アプリケーション ノートでは、一般的な節電方法だけでなく、PSoC 6 MCU 独自の低消費電力モードについて説明します。また、その他の低消費電力の注意点についても説明します。本書では、PSoC アーキテクチャに関する基本知識と、PSoC Creator™を使用した PSoC 6 MCU アプリケーションの開発能力が必要です。もし、PSoC 6 MCU を初めてお使いの場合は、「AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity」を参照してください。PSoC Creator を初めてご使用の場合は、「PSoC Creator™ Integrated Design Environment (IDE)」を参照してください。

2 電力モード

2.1 電力モードの遷移

PSoC 6 MCU は 6 つの電力モードで動作します。これらのモードは、アクティブ、低消費電力アクティブ (LPACTIVE)、スリープ、低消費電力スリープ (LPSLEEP)、ディープ スリープ、ハイバネートです。表 1 にデバイス動作の消費電力モードを示します。

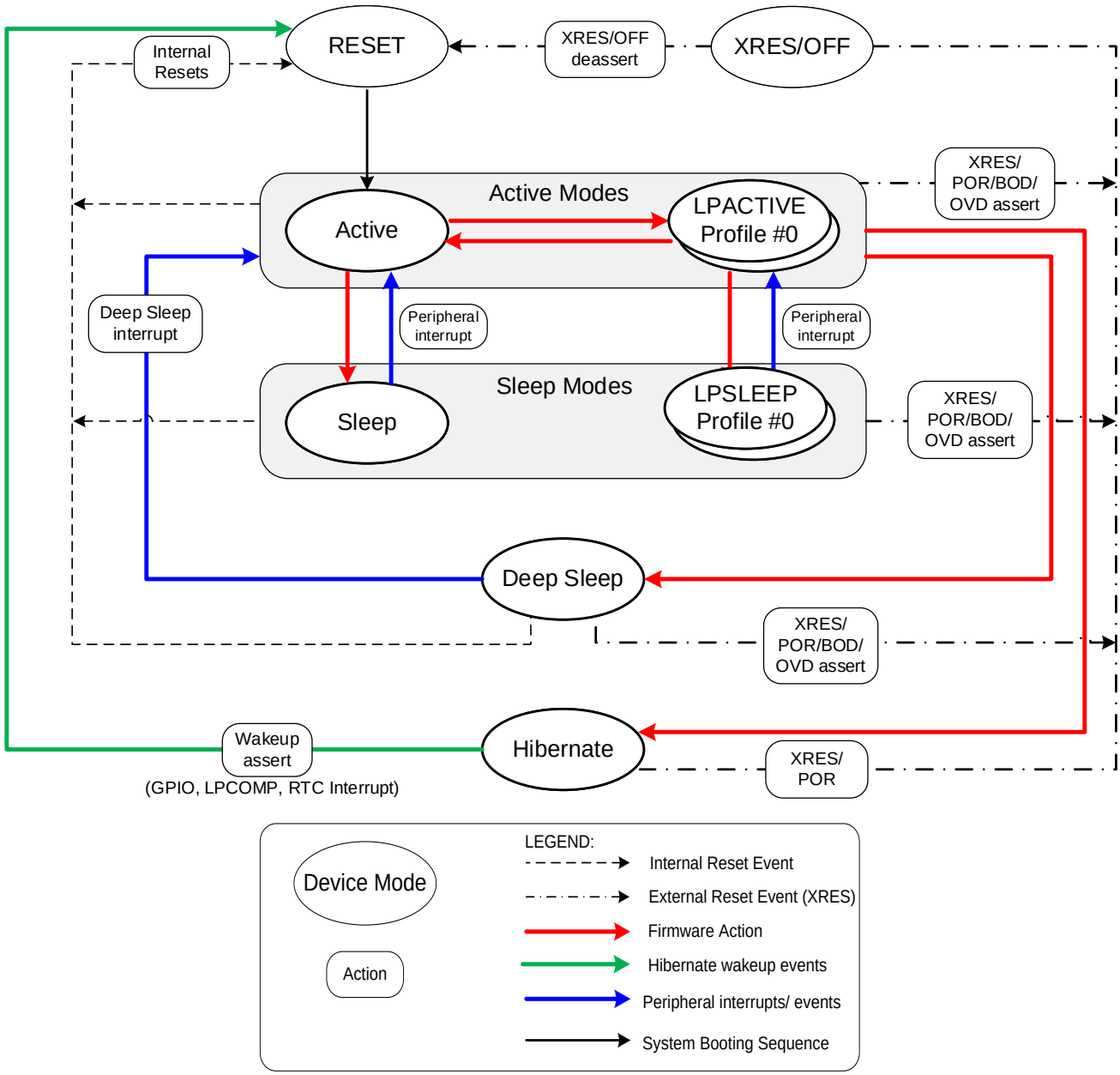
表 1. 電力モードの説明

電力モード	説明
アクティブ	・ CPU コアのいずれかまたは両方によるコード実行 ・ すべてのブロックは最大の消費電力および速度で使用可能
低消費電力アクティブ	・ CPU コアのいずれかまたは両方によるコード実行 ・ すべてのブロックは使用可能。但し、システム消費電流を抑えるために高周波クロック ソースは制限される ・ レギュレータとクロックのコンフィギュレーション
スリープ	・ コード実行なし ・ すべてのペリフェラルは使用可能
低消費電力スリープ	・ コード実行なし ・ すべてのペリフェラルは制限された電流とクロック周波数で使用可能 ・ レギュレータとクロックのコンフィギュレーション用のマルチ オプション
ディープ スリープ	・ コア、ほとんどのペリフェラルおよび高周波クロックはオフ ・ 低速クロックはオン ・ 低消費電力アナログと一部のデジタル ペリフェラルは動作可能で、ウェイクアップ ソースとして使用可能 ・ SRAM はデータ保持される
ハイバネート	・ コアとクロックはオフ ・ GPIO 状態は凍結 ・ 低消費電力コンパレータ、RTC アラーム、専用ウィクアップ ピンはシステムを復帰させるために使用可能 ・ バックアップ ドメインは使用可能。バックアップ ドメインの詳細は外部 PMIC 制御節および PSoC 6 MCU アーキテクチャ テクニカル リファレンス マニュアルを参照してください

図 1 電力モードの遷移が割込み、ファームウェア アクションおよびリセット イベントを含む異なるイベントおよびアクションに基づいていることを示します。スリープから LPSLEEP などの一部のケースでは、モード遷移は複数のモード間で行われます。

詳細は PSoC 6 MCU アーキテクチャ テクニカル リファレンス マニュアルおよび Appendix A を参照してください。

図 1. PSoC 6 MCU デバイス電力モードの遷移図

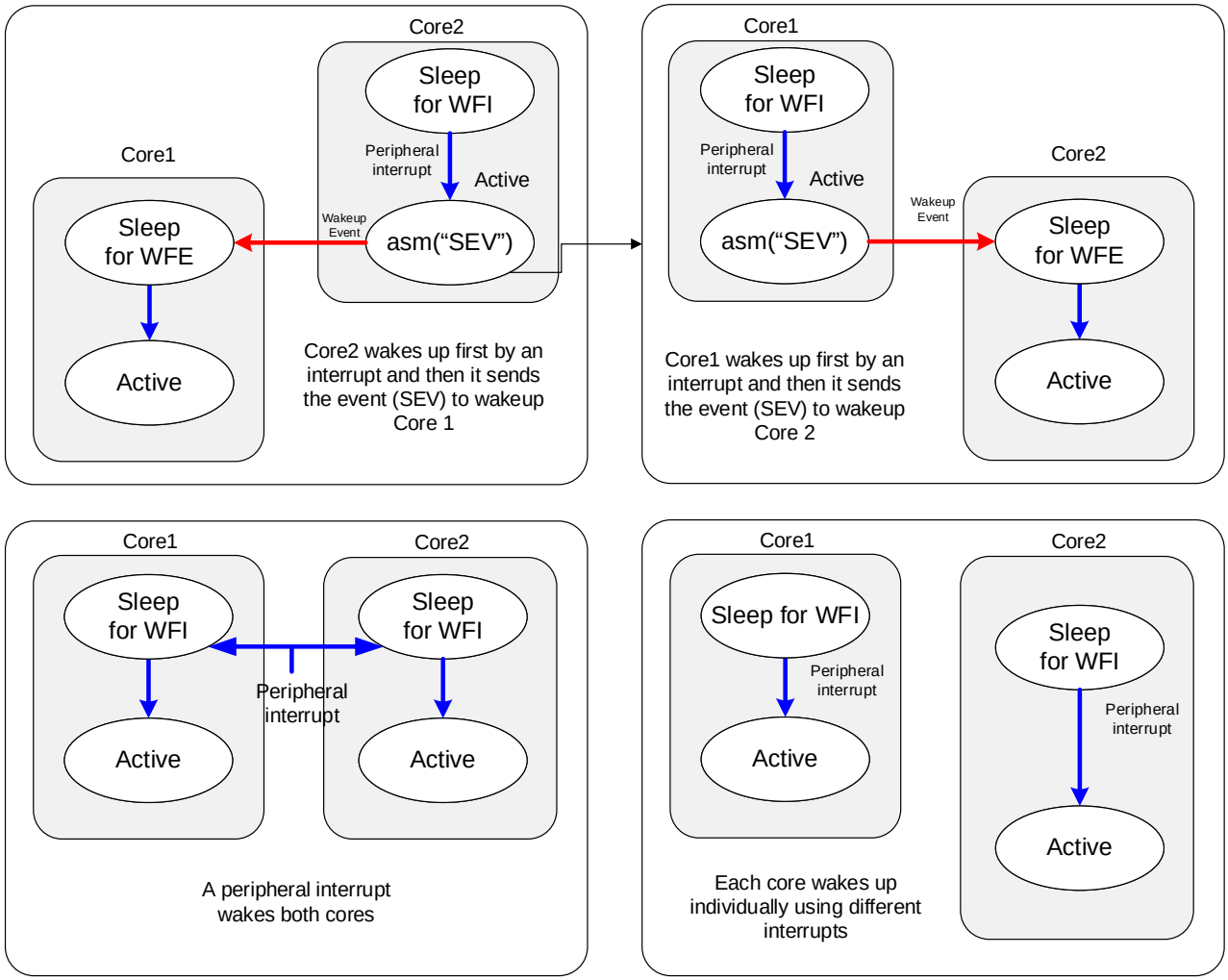


2.2 コアのスリープとウェイクアップ命令

Arm®コアそれぞれは、スリープとウェイクアップ間で独立して遷移します。図 2 にスリープからウェイクアップのいくつかのシナリオを示します。

Wait-for-Interrupt (__WFI) はコア スリープ命令です。コアが__WFI を実行した後、コアはスリープに入り、いずれかの割込みがアサートされるまでスリープに留まります。Wait-for-Event (__WFE) は__WFI と似ていますが、割込みに代わるウェイクアップ イベントが受信されるとウェイクアップします。Set Event (__SEV) は__WFE でスリープ モードに入った他のコアをウェイクアップさせるために使用されます。ディープ スリープはスリープとウェイクアップに同じ命令を使用しますが、スリープ命令の前に Arm システム制御レジスタ (SCR) の SLEEPDEEP ビット[2]がセットされます。SCR の詳細は Arm システム制御レジスタ ユーザー ガイドを参照してください。

図 2. マルチ コアのスリープとウェイクアップ ケース



CPU の電力状態はデバイスの電力モードと異なります。図 2 は、それぞれの CPU コアが他のコアの状態とは独立し、個別のスリープ モードに対応していることを示します。両方のコアがスリープまたはディープ スリープになると、デバイスはそれぞれスリープまたはディープ スリープ モードになります。詳細は「AN215656 – PSoC 6 MCU Dual-Core CPU System Design」を参照してください。

2.3 サブシステムの可用性と電力

2.3.1 サブシステムの可用性

各システム リソースの動作は、電力モードによって異なります。例えば、CPU はオン、オフおよびデータ保持モードのいずれかになります。電力モードが正しく動作するためには、適切なペリフェラルを選択することが重要です。表 5 に、異なる電力モードで使用できるリソースの一覧を示します。

2.3.2 電力消費量の概算

デバイス データシートは、特定条件下での消費電力データを提供します。最高の電力消費を達成するためにさまざまな組合せがあるため、アプリケーションの実際の電力消費量はデータシートと異なる場合があります。

2.4 サンプル ケース シナリオ

適切な電力モードの選択により、性能を低下させずに消費電力を節約できます。表 2 に、電力モードのサンプル ケース シナリオを示します。一部のサンプル ケースでは、わずかな電力モードだけが効果的に使用されます。

表 2. 電力モードのサンプル ケース シナリオ

	ウェアラブル デバイス	エアコン	リモコン	温度計
アクティブ	ユーザーによる GUI 対話	モーター使用	-	BLE 経由で通信
低消費電力アクティブ	心拍数を処理	-	コマンドを送信	温度を読み取る LCD に結果を更新
スリープ	-	-	-	-
低消費電力スリープ	アナログ ブロックは心拍を検出	-	-	-
ディープ スリープ	デバイスが 30 秒間心拍を検出しない (デバイスが未使用) 場合、ディープ スリープに遷移	コマンド入力まで待機 モーター未使用 赤外線 (IR) トリガによるウェイクアップ	-	ウォッチドッグ タイマー (WDT) を使用して 1 秒毎にウェイクアップ
ハイバネート	低バッテリ – 何もしない 充電開始でデバイスをリセット	-	ボタンが押されるまで待機	-

2.5 システム電源管理 (SysPM) ライブラリ

2.5.1 概要

サイプレスのペリフェラル ドライバー ライブラリ (PDL) は、ペリフェラルとシステム リソースを構成し、目的の機能を実装するための API を含んだ完全なソフトウェア ツールです。これにより、レジスタに関する情報を理解する必要性が減ります。

図 1 に示すように、PDL では SysPm API が電力モード変更の関数を提供します。また図 4 に示すように、API は電力モード遷移の前後にペリフェラル機能を実行するコールバック関数を登録することもできます。

2.5.2 モード遷移関数

図 1 に、電力モードのファームウェア遷移を示します。SysPm は、スリープ、ディープ スリープ、ハイバネート、LP 開始、LP 終了のデフォルトの 5 つの遷移関数を提供します。

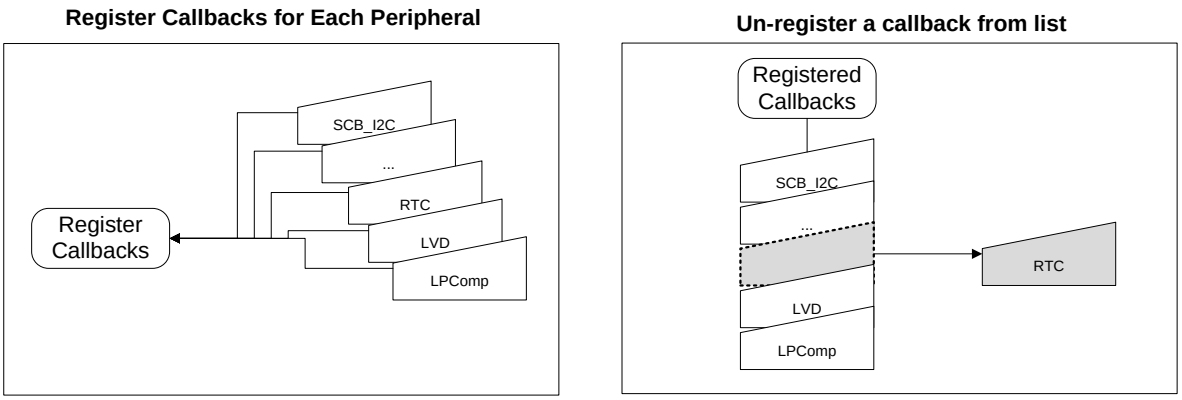
電力モード変更関数は、4 つの異なるコールバック動作を提供し、各ペリフェラルに必要なアクションを実行します。

- `CY_SYS_PM_CHECK_READY`: 他のモードに遷移する準備完了状態をチェックします。`CY_SYSPM_FAIL` を返した場合、遷移なしで終了します。
- `CY_SYSPM_BEFORE_TRANSITION`: コールバックは、モード遷移前に必要なアクションを実行して設定します。
- `CY_SYSPM_AFTER_TRANSITION`: コールバックは、モード遷移または設定後に実行します。
- `CY_SYS_CHECK_FAIL`: コールバックは、`CY_SYSPM_CHECK_READY` が失敗した時にのみ実行します。これはロールバック アクションを実行する

SysPm は登録、登録解除、実行という 3 つのコールバック用の関数を提供します。これらの関数は電力の最適化を支援するだけでなく、モード遷移後の異常なペリフェラル状態の防止にも有効です。図 3 に示すように、.PDL はユーザーが各電力モードのコールバックを登録することを期待しています。ほとんどのペリフェラル ドライバーは、各電力モードに関連する事前定義されたコールバックがあります。ユーザーは、定義されたペリフェラルコールバックを登録、またはカスタム コールバックの作成が選択できます。SysPm の遷移関数は、登録されたコールバックを 1 つずつ実行します。最初に登録された関数が最初に実行されます。

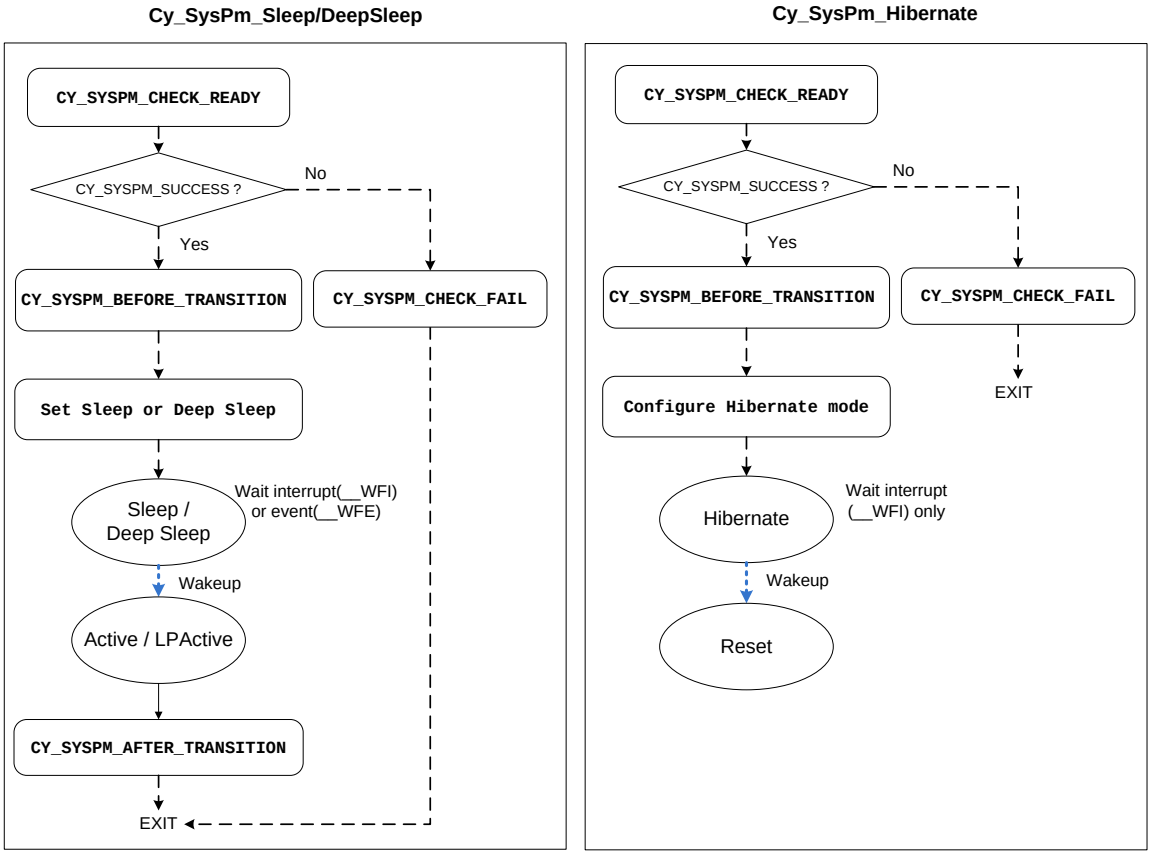
コールバックの登録と実装の詳細は、Appendix D および「E.1 CE219881 - PSoC 6 MCU の電力モード間の切換えこれはアクティブ、LPACTIVE、スリープ、LPSLEEP およびディープ スリープ用のモード遷移例です。

図 3. 電力モード コールバックの登録と登録解除



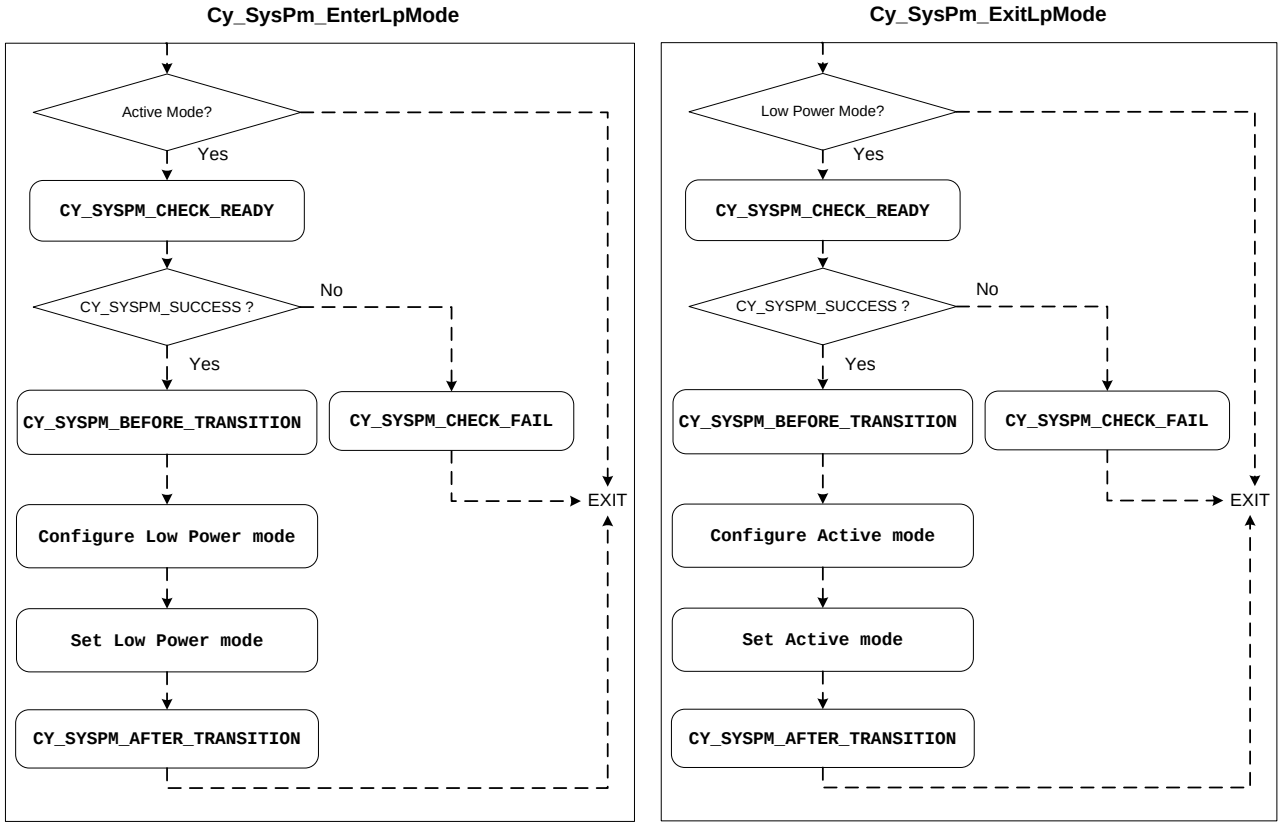
モード遷移関数の呼び出しで、デバイスは 4 つのコールバック動作で遷移を開始します。スリープ、ディープ スリープ、ハイバネートの各モードは、Arm スリープ命令が使用されます。コード実行は停止し、電力モード間の割込みを待ちます。図 4 は、WFI() または__WFE() スリープ命令の呼び出しによりウェイクアップ ソースを待機する CPU を示します。このため、スリープ、ディープ スリープおよびハイバネートへの実際のモード遷移は、スリープ命令が実行された後に行われます。ウェイクアップ後、デバイスは自動でアクティブ、LPACTIVE またはリセットに遷移します。

図 4. スリープ／ディープ スリープ／ハイバネート モード遷移フローチャート



アクティブと LPACTIVE モードでは、すべてのシステム リソースが実行を継続します。従って、LP モードの開始／終了は、電力モード制御レジスタを設定することによって行われ、スリープはなく、何も割込みを待機しません。図 5 に示すように、SysPm PDL は関連するドライバー関数を提供します。最高の消費電力効率のためには、コア電圧レギュレータとシステム クロックを設定する必要があります。詳細は「3.1 コア電圧選択」、「3.2 低消費電力モード クロック」およびペリフェラル ドライバー ライブラリのドキュメント (**PSoC Creator** > **Help** > **Documentation** > **Peripheral Driver Library**) を参照してください。

図 5. 低消費電力モードの遷移フローチャート



3 PSoC 6 MCU 消費電力の管理技術

3.1 コア電圧選択

3.1.1 リニア レギュレータと降圧レギュレータ

表 3 に示すように、PSoC 6 MCU はコア電源用に複数のオンチップ レギュレータ (低ドロップ アウト (LDO) レギュレータおよびシングル入力マルチプル出力 (SIMO) 降圧レギュレータ) をサポートしています。LDO は、出力電圧が 1.1V の時に最大 300mA、出力電圧が 0.9V の時に最大 25mA を供給することができます。SIMO 降圧レギュレータは、1 出力に 20mA、組合せ出力に 30mA を供給できます。SIMO 降圧レギュレータは、通常の負荷条件でより優れた効率を発揮します。SIMO に切り替えると、リセットなしに LDO にスイッチバックすることは推奨されません。

表 3. 低消費電力プロファイル用のコア電圧レギュレータの異なるオプション

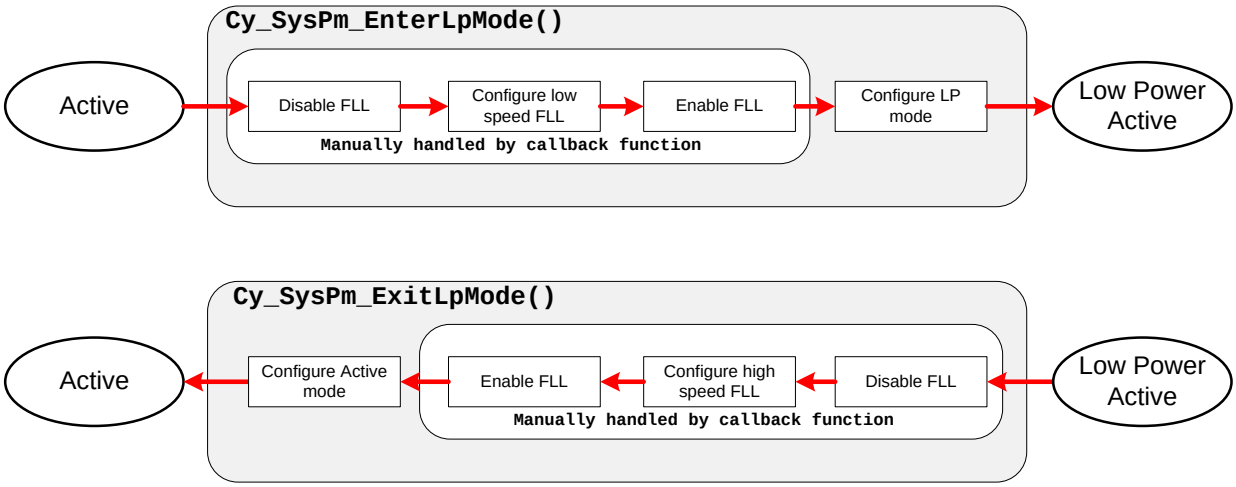
	出力	最大負荷	最大クロック周波数
LDO	0.9V	25mA	Cortex®-M4 用の 50MHz Cortex-M0+用の 25MHz
	1.1V	300mA	サポート可能な最大クロック周波数を許可
SIMO 降圧	0.9V	20mA	CM4 用は 50MHz CM0+用は 25MHz
	1.1V	20mA	CM4 用は 50MHz CM0+用は 25MHz

3.2 低消費電力モード クロック

低消費電力モードへの遷移は、Arm コア コンフィギュレーション (_WFI、_WFE、_SEV) の代わりに、電力モード制御レジスタの設定によって行うことができます。前述したように、低消費電力モードの最大クロック速度の制限があるため、LP モードの開始／終了時にレギュレータ出力に基づいてクロック コンフィギュレーションを調整する必要があります。PDL は、PWR_CTL レジスタを設定するための関連する関数を提供します。詳細は、PSoC 6 MCU レジスタ テクニカル リファレンス マニュアルを参照してください。

図 6 に、登録されたコールバック関数で PDL 関数を使用し、アクティブと LPACTIVE 間の遷移方法を示します。LP モードのクロック制限のため、HFClk が制限よりも速い場合、モード遷移前に FLL クロック速度または HFClk のいずれかを調整する必要があります。FLL の変更は、FLL クロックを使用するブロックに影響します。このため、すべてのアクティブなペリフェラルは、変更する周波数に対応するために独自のコールバックを登録する必要があります。「CE219881 – PSoC 6 MCU Switching between Power Modes」はクロック調整のコールバック例を提供します。

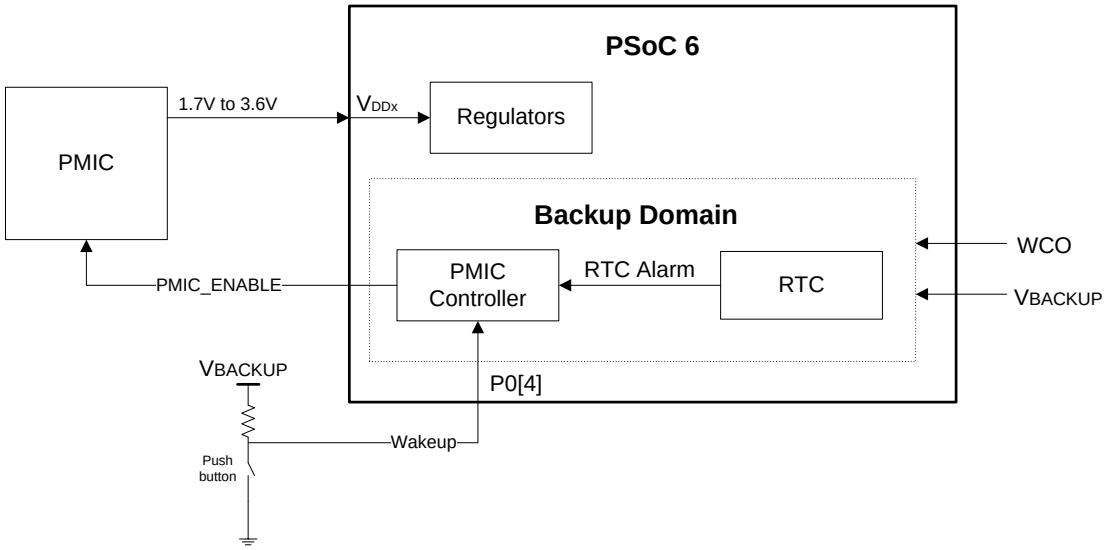
図 6. LP モード開始／終了の遷移



3.3 外部 PMIC 制御

PSoC 6 MCU バックアップ ブロックは電源管理 IC (PMIC) 制御機能を提供します。図 7 に、システム電源 (V_{DDD}) を供給する外部 PMIC を示します。PSoC 6 MCU は PMIC により完全にシャットダウンできますが、PMIC を制御するバックアップ電源はバックアップ ドメインを動作させます。

図 7. 外部 PMIC 制御ブロック図



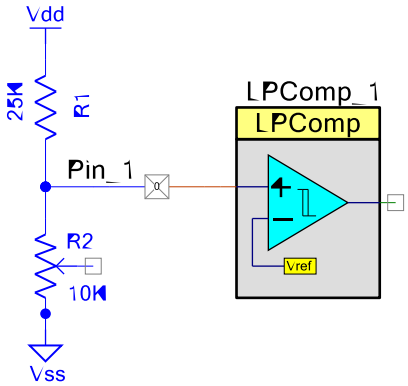
4 他の電力節約技術

4.1 PSoC による電流経路の開閉

PCB は、電力を消費する他の部品が含まれている場合があります。 PSoC 6 MCU はそれらの電流を制御することができます。 例えば、GPIO は電流を引き出すことができます。データシートに記載されているピンのソースとシンク電流の最大値を超えないように注意してください。

図 8 に示すように、このシナリオの良い例は低消費電力コンパレータ (LPComp) アプリケーションです。この場合、PSoC デバイスはポテンショメータ抵抗値の変化に伴って変化するアナログ ピンの電圧を比較します。

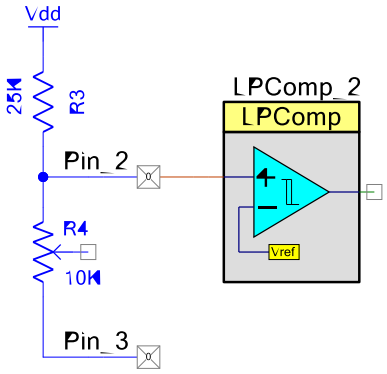
図 8. 代表的な LPComp アプリケーション



LPComp は使用していない時にオフにすることができますが、抵抗とポテンショメータを通る電流経路が残っているため、外付け部品は電力を消費します。PSoC での簡単な解決策は、図 9 に示すように 2 番目のピンをグランドへのスイッチとして使用することです。

この構成では Pin_3 に「1」を書き込むことで電流を止め、ピンを開放にすることができます。これにより、2 個の抵抗間の電圧差を 0V に減少させることで電流を消費しません。「0」を書き込むと、電流が流れます。この節電機能のリソース使用は 1 本のピンおよび数行のコードのみです。

図 9. グランド スイッチとして GPIO を使用



4.2 未使用ブロックの無効化

未使用ブロックを無効化し、不要な電流消費を節約します。

4.3 DMA によるデータ移動

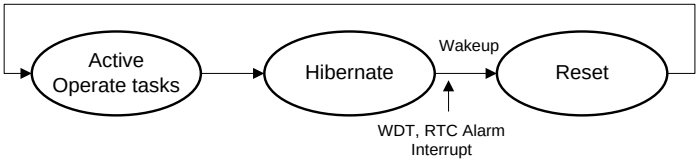
CPU 負荷を軽減し、CPU を停止させたり他のタスクを並行して実行させたりすると、いつでも消費電力を節約できます。DMA エンジンは CPU を使用せずにアクティブまたはスリープ モードでデータを転送するために使用できます。

4.4 周期的ウェイクアップ タイマー

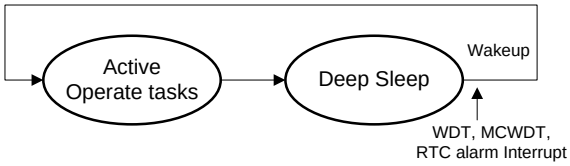
スリープ方法からの周期的なウェイクアップは、消費電力を節約するための従来方法です。平均消費電力は、アクティブ期間とスリープ期間の消費電力によって決まります。最高の結果を得るためにスリープ期間はできるだけ長くし、アクティブ期間はできるだけ短くする必要があります。WDT およびマルチカウンターWDT (MCWDT) は、ディープ スリープ モードで良好な周期的ウェイクアップ ソースとなります。WDT はハイバネート モードで動作します。アプリケーションがより長いウェイクアップ周期を必要とする場合、RTC アラームは良好な周期的ウェイクアップ ソースにすることができます。RTC 周期的タイマーのサンプル コードは、「E.3 CE218542 - RTC アラーム割込みを用いた PSoC 6 MCU カスタム ティック タイマー」を参照してください。

以下のシナリオは、モード状態を変更する方法を示します。

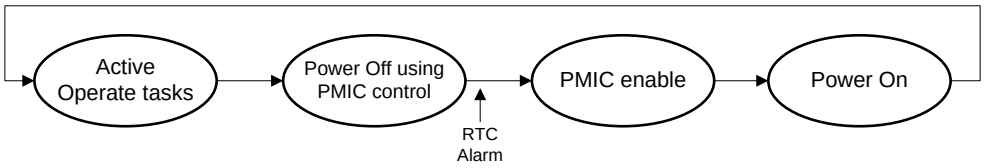
- アクティブとハイバネート



- アクティブとディープ スリープ



- アクティブと外部 PMIC による電源オフ

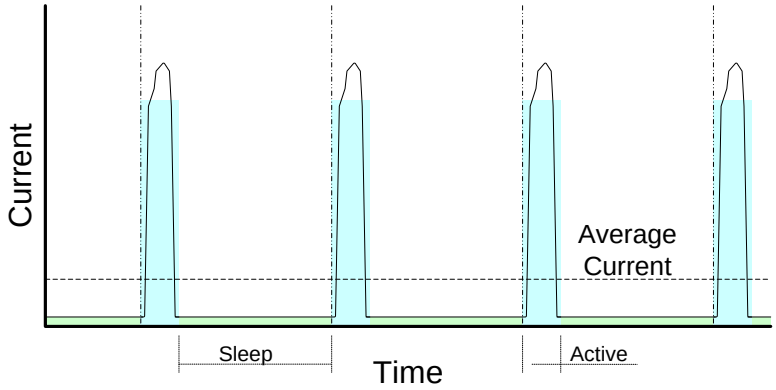


4.5 クロック

ケースによっては、高速でクロックを実行すると、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 回センサーを読み取り、幾つかの計算を行って結果を別のデバイスに送信する PSoC 設計を検討してみてください。

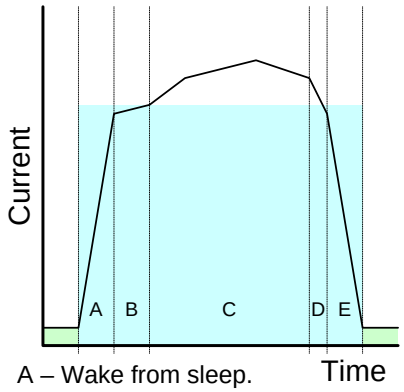
PSoC デバイスがアイドル状態の時、スリープまたはディープ スリープ モードを使用して消費電力を削減できますが、アクティブ モードで費やす時間に起因して平均消費電流が高くなってしまいます。図 10 に、システム クロックが 3MHz に設定されたこの例での消費電流を示します。

図 10. 3MHz クロックを使った電流分布の例



PSoC デバイスがウエ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 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 11 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 11. 3MHz でのアクティブ モードのタスク解析

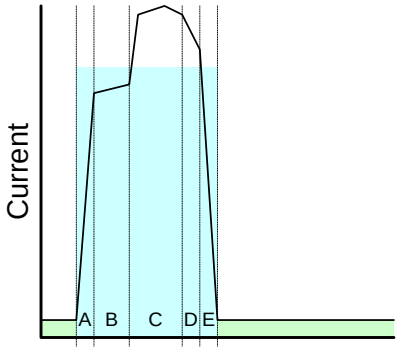


A – Wake from sleep.
B – Read sensor data.
C – Manipulate data.
D – Transmit result.
E – Go back to sleep.

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。

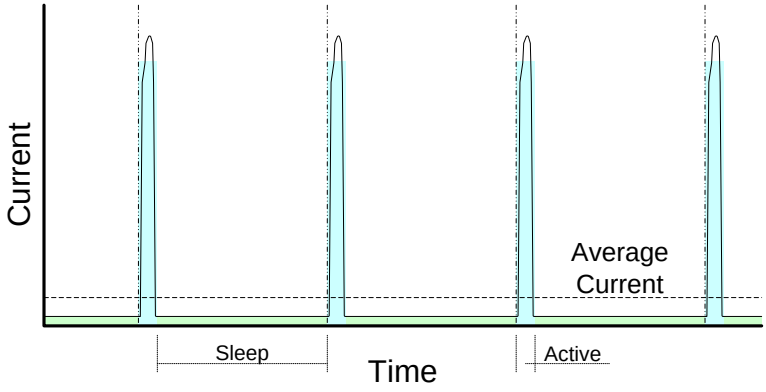
ある時点では、アクティブ時間が短い利点よりも、高速でクロックを駆動するために必要となるエネルギーが高い弱点のほうが優ります。図 12 に示すように、最適速度が 12MHz であると仮定します。12MHz クロックでは、アクティブ モードで費やされる時間は、3MHz のクロックで費やされた時間の約半分です。図 13 に、より速いクロックの時にピーク時の消費電流はより多くなりますが、平均消費電流は少なくなることを示します。

図 12. 12MHz でのアクティブ モードのタスク解析



A – Wake from sleep.
B – Read sensor data.
C – Manipulate data.
D – Transmit result.
E – Go back to sleep.

図 13. 12MHz クロックを使った電流分布の例



4.6 GPIO

PSoC デバイスが低消費電力モードにある時、GPIO は外部回路を駆動し続けることができます。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時に役立ちますが、ピンが電流を不必要に電流を吐き出したり吸い込んだりする時には、消費電力の無駄になってしまう場合があります。

設計を分析して、低消費電力動作での GPIO の最適な状態を決める必要があります。デジタル出力ピンを論理 1 または 0 に保持することが最善の場合、`Cy_GPIO_Write()` を使用して同じデジタル レベルに一致させてください。

```
/* 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限り、すべての未使用 GPIO をアナログ HI-Z に設定します。ピンコンポーネントのポート幅駆動モードは `Cy_GPIO_SetDrivemode()` 関数を使って設定できます。

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

PSoC の柔軟性により、電流漏れを防止するように GPIO 駆動モードを簡単に管理することができます。ハイバネート モードでは、GPIO 駆動モードとデータ レジスタは「凍結」されることがあります。ピンの状態は `Cy_SysPm_IoFreeze()` の呼び出しで凍結されます。ウェイクアップ リセット後、その状態を変更できるようにするために `Cy_SysPm_IoUnfreeze()` の呼び出しで「凍結解除」する必要があります。ハイバネートでの IO 制御のサンプル コードは「E.2 CE218129 - 低消費電力コンパレータによる PSoC 6 MCU のハイバネートからのウェイクアップ

5 電源保護システム

5.1 ハードウェア制御による電源保護

5.1.1 電圧低下検出 (BOD)

電圧低下検出 (BOD) は、ロジックが V_{DDD} および V_{CCD} 電源の喪失に衝突する前にシステムをリセットすることができます。電圧低下検出システムは、V_{DDD} が最小のシステム動作電圧になる前にリセットを保証し、これはすべてのロジック、SRAM、フラッシュなどに有効です。これはハードウェアにより制御され、設定できるレジスタはありません。

5.1.2 低電圧検出 (LVD)

低電圧検出 (LVD) は BOD に似ていますが、設定可能です。V_{DDD} が設定されたトリップ電圧を下回ると、LVD は割込みを生成します。この割込みは、BOD リセットがトリガされる前に重要なデータを処理するために役立ちます。LVD は 15 の選択可能なトリップ電圧を提供します。ディープ スリープおよびハイバネート モードでは LVD は使用できません。

5.1.3 過電圧検出 (OVD)

過電圧検出 (OVD) は BOD の逆です。これらの回路は、V_{DDD} および V_{CCD} に安全でない過電圧状態が検出された時にリセットを生成します。ファームウェアの制御は不要です。OVD は、システムを高電圧の損傷から保護するために役立ちます。

6 まとめ

PSoC 6 MCU では、多くの消費電力管理オプションが使用できます。適切な方法に従うことで、設計を最適化し、PSoC 6 MCU の新しい電力モードと機能は、バッテリ駆動デバイスの性能劣化を招くことなく消費電力を最小限に抑える最適なオプションが得られます。

7 関連文書:

AN215656 – PSoC 6 MCU Dual-Core CPU System Design

CE219881 – PSoC 6 MCU Switching between Power Modes

CE218542 – PSoC 6 MCU Custom Tick Timer Using RTC Alarm Interrupt

CE218129 – PSoC 6 MCU Wakeup from Hibernate Using a Low-Power Comparator

著者について

氏名:	Brian Lee
役職:	アプリケーションエンジニア主任
経歴:	Brian Lee は韓国嶺南大学校から BSEE を取得します。彼は携帯電話や MCU ベースの組込みシステムを長年開発しています。

Appendix A. 電力モードの詳細

A.1 アクティブ

アクティブ モードは動作の主要モードです。一方または両方のコアがコードを実行し、すべてのロジックとメモリに電力が供給されます。このため、このモードは最大消費電力で最高のパフォーマンスを発揮します。最大の消費電力とスピードでは、ペリフェラルの動作に制限はありません。

A.1.1 コア クロックとリニア コア レギュレータ

リニア コア レギュレータは、アクティブ モード用 1.1V の V_{CCD} を生成します。これは最大 100mA の負荷電流を駆動します。コアは最大クロック速度で動作できます。

A.1.2 モード遷移

モード遷移は、割込みまたはアクティブ モードのファームウェア アクションを使用します。図 1 に示すように、アクティブ モードから別のモードへの遷移にファームウェア アクションを使用します。しかし、この反対の遷移は割込みだけを使用します。モードはファームウェア アクションなしで自動的にアクティブに遷移します。デバイスがリセットされると、デバイスはアクティブになります。

A.2 低消費電力アクティブ (LPACTIVE)

LPACTIVE はアクティブと同じ機能を持ちますが、消費電力を抑えるためのいくつかのトレードオフがあります。このモードの主な違いは、電流制限とクロック周波数制限です。

A.2.1 コアのクロックとレギュレータ

低消費電力アクティブ モードは 0.9V の V_{CCD} を使用します。負荷電流は 25mA に制限されます。CM4 のコア クロックは最大 50MHz、CM0+のコア クロックは最大 25MHz までです。

A.2.2 モード遷移

ファームウェア アクションがアクティブ モードで低消費電力コンフィギュレーションを設定すると、デバイスはアクティブから LPACTIVE に遷移します。低消費電力コンフィギュレーションが有効になると、スリープまたはディープ スリープからのウェイクアップ時に LPACTIVE に戻ります。しかし、低消費電力コンフィギュレーションがすでに有効になっていても、デバイスはデバイス リセットでアクティブ モードになります。

A.3 スリープ

スリープ モードでは、Arm コア (CM0 +および CM4) はコードを実行せず、Arm スリープ命令 (__WFE または__WFI) の呼び出しでスリープに留まりますが、その他のすべてのペリフェラルは使用可能です。各コアはスリープ状態から独立してスリープを解除することができます。

A.3.1 ペリフェラル

デバイスは、表 5 に示すアクティブおよび低消費電力アクティブのようにすべてのシステム リソースが使用できます。

A.3.2 コアの状態

CPU クロックがオフになり、CPU はスリープします。

A.3.3 ウェイクアップ ソース

CPU がスリープ モードになっている時、登録された割込み (CPU にマスクされる) とコアからの__SEV イベントは、ウェイクアップ ソースになる可能性があります。図 2 に__WFI、__WFE および__SEV を使用したウェイクアップ フローを示します。

A.3.4 モード遷移

両方のコアがスリープ モードに遷移するためには、ファームウェア アクションが必要です。スリープ モードに入るファームウェア関数を呼び出すと、アクティブがスリープに遷移します。図 1 に示すように、アクティブ以外の他のモードはスリープに遷移することはできません。

A.4 低消費電力スリープ (LPSLEEP)

LPSLEEP はスリープ モードと同じ機能を持ちますが、消費電力を抑えるためのいくつかのトレードオフがあります。このモードの主な違いは電流制限です。

A.4.1 ペリフェラル

表 5 に示すように、デバイスはスリープのようにすべてのシステム リソースを使用できますが、汎用デジタル ブロック (UDB) は遅くなります。デバイスの最大電流が 25mA を超えないようにする必要があるため、クロック周波数を下げなければなりません。

A.4.2 コアの状態

CPU クロックがオフになり、CPU はスリープします。

A.4.3 ウェイクアップ ソース

CPU が低消費電力スリープ モードになっている時、登録された割込み (CPU にマスクされる) とコアからの__SEV イベントは、ウェイクアップ ソースになる可能性があります。

A.4.4 モード遷移

ファームウェアが LPACTIVE でスリープ モード遷移関数を呼び出すと、電力モードは LPSLEEP に変わります。両方のコアがスリープである必要があります。そうでない場合、システムは LPACTIVE のままになります。図 1 に示すように、LPACTIVE 以外の他のモードは LPSLEEP に遷移することはできません。

A.5 ディープ スリープ

ディープ スリープは、高周波クロックを使用するシステム リソースをオフにします。しかし、低周波クロック (内部低速発振器 (ILO)、高精度 ILO、時計用水晶発振器 (WCO)) およびこれらの低周波クロックを使用するペリフェラルは引き続き動作します。ディープ スリープ中に、アクティブまたは LPACTIVE に高速で遷移するために SRAM と CPU は保持されます。

A.5.1 ペリフェラル

表 5 に示すように、内部主発振器 (IMO)、外部水晶発振器 (ECO)、位相ロック ループ (PLL)、周波数ロック ループ (FLL) などの高周波クロックはディープ スリープで動作しません。

ディープ スリープでは内部低速発振器 (ILO) と WCO を許可し、LCD、GPIO、ウォッチドッグ タイマー (MCWDT、EDT)、CTBm、RTC は極端な低消費電力状態でも正常に動作します。

SRAM、SMIF、CSD、SCB およびバックアップ レジスタは動作しませんが、高速ウェイクアップのために保持状態に維持されます。

A.5.2 コアの状態

各コアは割込みが発生するまでディープ スリープに留まり、CPU、SRAM およびすべてのレジスタは保持されます。

A.5.3 レギュレータ

ディープ スリープでは、ディープスリープ レギュレータ (V_{CCD}SLP) とリテンション レギュレータ (V_{CC}RET) は、保持中のリソース用とさまざまな動作中のペリフェラル用に電源を供給します。レギュレータは V_{DDD} または SIMO 降圧レギュレータによって電源供給されます。SIMO 降圧レギュレータは、低消費電力に効率的です。

A.5.4 ウェイクアップ ソース

GPIO エッジ検出ロジックの割込み、コアからの SEV イベント、(LPComp)、SCB I2C／SPI スレーブ、CTBm および RTC アラームはウェイクアップ ソースとなります。

A.5.5 モード遷移

ディープ スリープでは、両方のコアがファームウェア アクションによってディープ スリープにある必要があります。1 つのコアがウェイクアップした場合でも、他のコアは割込みが発生するまでディープ スリープに留まります。図 1 に示すように、デバイスはウェイクアップ後、前の状態に戻ります。

A.6 ハイバネート

ハイバネートは、ほとんどのリソースをオフにして、最小の消費電力になります。アプリケーションを休止状態にすることを目的としていますが、オフ状態ではないため、V_{DDD} は提供され、一部のペリフェラルは動作し続けます。システムがハイバネートである場合、システムおよびファームウェアはウェイクアップにより再起動する必要があります。

A.6.1 ペリフェラル

表 5 に示すように、ILO が使用可能であってもほとんどのペリフェラルは動作しません。WDT とバックアップ ドメイン (RTC とアラーム) は機能し続けます。システムの一部のハイバネート レジスタとバックアップ レジスタは保持されたままです。GPIO 状態は凍結されます。予期しないウェイクアップを防ぐために、LVD と BOD は自動的に無効になります。

A.6.2 コアの状態

CPU と SRAM は動作しません。しかし、2 つの 4 バイトのハイバネート レジスタが保持されます。

A.6.3 レギュレータ

すべての内部レギュレータはオフです。V_{DDD} と V_{BACKUP} は電源を供給します。

A.6.4 ウェイクアップ ソース

2 つのウェイクアップ ピン、ウォッチドッグ タイマー、LPComp および RTC アラームはウェイクアップ ソースにできます。

A.6.5 遷移

ハイバネート モードへの遷移には、ファームウェア アクションが必要です。1 つのコアがハイバネートに入るための関数を呼び出し、システム全体がハイバネート モードに入ります。図 1 に示すように、システムを再起動するためにウェイクアップした後は常にリセットに戻ります。そうでない場合、システム リソースが保持されていないため、システムは正しく動作しません。ファームウェアは、ハイバネートからリセットされた時に、凍結された GPIO の凍結を解除するためにリセット原因をチェックする必要があります。

A.7 リセット／XRES／オフ

外部リセット (XRES) とオフ モードは機能的に似ています。すべてのリソースが電力供給されず、I/O はリーク電流を防ぐためにトライステートになります。外部リセットのデアサート後、または POR／BOD のアサート後、リセットは開始されますが、明確な電力モードではありません。

Appendix B. 電力モードのまとめ

B.1 電力モードとウェイクアップ ソース

表 4. 電力モードとウェイクアップ ソース

電力モード	説明	遷移条件	ウェイクアップ ソース	ウェイクアップ アクション
アクティブ	・ 動作の主要モード ・ コード実行 ・ ペリフェラル用に最大消費電力とスピードをサポート	・ POR、XRES および BOD 後のブートシーケンス完了 ・ LPACTIVE からのファームウェア アクション: `Cy_SysPm_ExitLpMode()` ・ スリープ モードからのペリフェラル割込み ・ ディープ スリープ モードからの割込み	該当なし	該当なし
低消費電力アクティブ	・ アクティブと同じ機能 ・ コード実行 ・ 制限された電流とクロック周波数	・ アクティブからのファームウェア アクション: `Cy_SysPm_EnterLpMode()` ・ LPSLEEP モードからのペリフェラル割込み ・ ディープ スリープ モードからの割込み	該当なし	該当なし
スリープ	・ コアはスリープを継続 ・ コード実行なし ・ すべてのペリフェラルは使用可能	アクティブからファームウェア アクション: `Cy_SysPm_Sleep()`	CPU への任意の割込み	割込み
低消費電力スリープ	・ LPACTIVE と同じ機能 ・ コード実行なし ・ 制限された電流とクロック周波数	LPACTIVE からのファームウェア アクション: `Cy_SysPm_Sleep()`	CPU への任意の割込み	割込み
ディープスリープ	・ コア、ほとんどのペリフェラルおよび高周波クロックはオフ ・ 低速クロックはオン ・ 低消費電力アナログと一部のデジタル ペリフェラルは動作可能で、ウェイクアップ ソースとして使用可能 ・ SRAM はデータ保持される	アクティブまたは LPACTIVE からのファームウェア アクション: `Cy_SysPm_Deep Sleep()`	・ GPIO 割込み ・ 低消費電力コンパレータ ・ SCB ・ CTBm ・ ウォッチドッグ タイマー ・ RTC アラーム	割込み
ハイバネート	・ コアとクロックはオフ ・ GPIO 状態は凍結 ・ 低消費電力コンパレータ、RTC アラームおよび専用の WAKUP ピンは、システムをウェイクアップするために使用可能 ・ バックアップ ドメインは使用可能	アクティブまたは LPACTIVE からのファームウェア アクション: `Cy_SysPm_Hibernate()`	・ WAKEUP ピン ・ 低消費電力コンパレータ ・ RTC アラーム	リセット

Appendix C. サブシステムの可用性

C.1 異なる電力モードで使用できるリソース

表 5 に、さまざまな電力モードでのリソース可用性に関する情報を示します。

表 5. 異なる電力モードで使用できるリソース

部品	電力モード							
	アクティブ	低消費電力アクティブ [1]	スリープ	低消費電力スリープ [1]	ディープスリープ	ハイバネート	XRES アサート	バックアップ付き電源オフ [2]
コア機能								
CPU	オン	オン	スリープ	スリープ	データ保持 [3]	オフ	オフ	オフ
SRAM	オン	オン	オン	オン	データ保持 [3]	オフ	オフ	オフ
フラッシュ	オン	オン	オン	オン	オフ	オフ	オフ	オフ
高速クロック (IMO、ECO、PLL、FLL)	オン	オン	オン	オン	オフ	オフ	オフ	オフ
LVD	オン	オン	オン	オン	オフ	オフ	オフ	オフ
ILO	オン	オン	オン	オン	オン	オン	オフ	オフ
システム レジスタ (Arm レジスタ、サブシステム レジスタ)	オン	オン	データ保持 [3]	データ保持 [3]	データ保持 [3]	保持されるハイバネート レジスタ	オフ	オフ
ペリフェラル								
SMIF	オン	オン	オン	オン	データ保持 [3]	オフ	オフ	オフ
UDB	オン	オン (低速)[1]	オン	オン (低速)[1]	オフ	オフ	オフ	オフ
SAR ADC	オン	オン	オン	オン	オフ	オフ	オフ	オフ
CTBm	オン	オン	オン	オン	オン (より低い GBW)	オフ	オフ	オフ
TCPWM	オン	オン	オン	オン	オフ	オフ	オフ	オフ
CSD	オン	オン	オン	オン	データ保持 [3]	オフ	オフ	オフ
BLE	オン	オン	オン	オン	データ保持 [3]	オフ	オフ	オフ
LCD	オン	オン	オン	オン	オン	オフ	オフ	オフ
SCB	オン	オン	オン	オン	データ保持 (I^2C／SPI ウェイクアップ可能)[4]	オフ	オフ	オフ
GPIO	オン	オン	オン	オン	オン	凍結 [5]	オフ	オフ
LPComp	オン	オン	オン	オン	オン (ウェイクアップ割込み)[6]	オン (ウェイクアップ割込み) [6]	オフ	オフ

[1] 低消費電力アクティブ／スリープ: これらのモードでは、最大デバイス電流は 25mA を超えてはいけません (Min／Max 数値はデータシートを参照してください)。ファームウェアは、消費電力が制限を超えないようにブロックを有効／無効にし、クロック周波数を適切に下げる必要があります。

[2] バックアップ ドメイン電源はオンですが、PMIC はデバイスに電力を供給しません。

[3] データ保持: レジスタ値は低消費電力モードおよびウェイクアップを通じて保持されます。

[4] ディープ スリープ電力モードでは、ディープ スリープに対応する SCB のみが使用可能です。

[5] 凍結: システム内のすべての GPIO のコンフィギュレーション、モードおよび状態がロックされます。GPIO 属性の変更は、ハイバネートからのシステム リセット後にファームウェアが凍結解除するまで許可されません。

[6] これは、ローカル電圧リファレンス (0.4V～0.6V) に対する外部 GPIO 入力が可能です。

部品	電力モード							
	アクティブ	低消費電力アクティブ [1]	スリープ	低消費電力スリープ [1]	ディープスリープ	ハイバネート	XRESアサート	バックアップ付き電源オフ [2]
WDT	オン	オン	オン	オン	オン	オン	オフ	オフ
MCWDT	オン	オン	オン	オン	オン	オフ	オフ	オフ
リセット								
XRES	オン	オン	オン	オン	オン	オン	オン	オフ
POR	オン	オン	オン	オン	オン	オン	オフ	オフ
BOD	オン	オン	オン	オン	オン	オフ	オフ	オフ
ウォッチドッグ リセット	オン	オン	オン	オン	オン	オフ	オフ	オフ
バックアップ ドメイン								
WCO、RTC、アラーム	オン	オン	オン	オン	オン	オン	オン	オン
バックアップ レジスタ	オン	オン	オン	オン	データ保持 [3]	データ保持 [3]	データ保持 [3]	データ保持 [3]

Appendix D. コールバック関数例

D.1 コールバック関数の登録

```
cy_stc_syspm_callback_params_t myParams;
cy_stc_syspm_callback_t myAppSleep =

{
       &Application_Callback,              /* Callback function */
       CY_SYSPM_SLEEP,                     /* Select Power Mode */
       (CY_SYSPM_SKIP_CHECK_READY |
        CY_SYSPM_SKIP_CHECK_FAIL),         /* Skip CHECK_READY and CHECK FAIL */
       &myParams,                          /* Operation, contexts */
       NULL,                               /* Previous list callback */
       NULL                                /* Next list callback */

};

cy_stc_syspm_callback_t myAppHibernate =
{
       &Application_Callback,              /* Callback function */
       CY_SYSPM_HIBERNATE,                 /* Select Power Mode */
       0U,                                 /* Skip mode, no skip */
       &myParams,                         /* Operation, contexts */
       NULL,                               /* Previous list callback */
       NULL                                /* Next list callback */

};

/* Register Callback functions for each power mode */
Cy_SysPm_RegisterCallback(&myAppSleep);
Cy_SysPm_RegisterCallback(&myAppHibernate);
```

D.2 カスタム コールバック関数の実装

```
cy_en_syspm_status_t Application_Callback(
cy_stc_syspm_callback_params_t *callbackParams)
{
    cy_en_syspm_status_t retVal = CY_SYSPM_SUCCESS;

     switch(callbackParams->mode)
       {
         case CY_SYSPM_CHECK_READY:
          {
            if(Check_HW())
             {
                    /* Hardware is ready */
             }
           else
             {
                    retVal = CY_SYSPM_FAIL;
             }
         }
        break;

         case CY_SYSPM_CHECK_FAIL:
          {
          /* Rollback any configuration during CHECK_READY */
          Rollback_HW();
          retVal = CY_SYSPM_SUCCESS;
          }
```

```
                break;

            case CY_SYSPM_BEFORE_TRANSITION:
            {
                /* configure HW for new mode before transition */
                ConfigureHW_BeforeMode();
                retVal = CY_SYSPM_SUCCESS;
            }
            break;

            case CY_SYSPM_AFTER_TRANSITION:
             {
                /* configure HW after mode transition */
                ConfigureHW_AfterMode();
                retVal = CY_SYSPM_SUCCESS;
            }
            break;

            default:

                break;
        }

    return (retVal);
}
```

Appendix E. サンプル コード

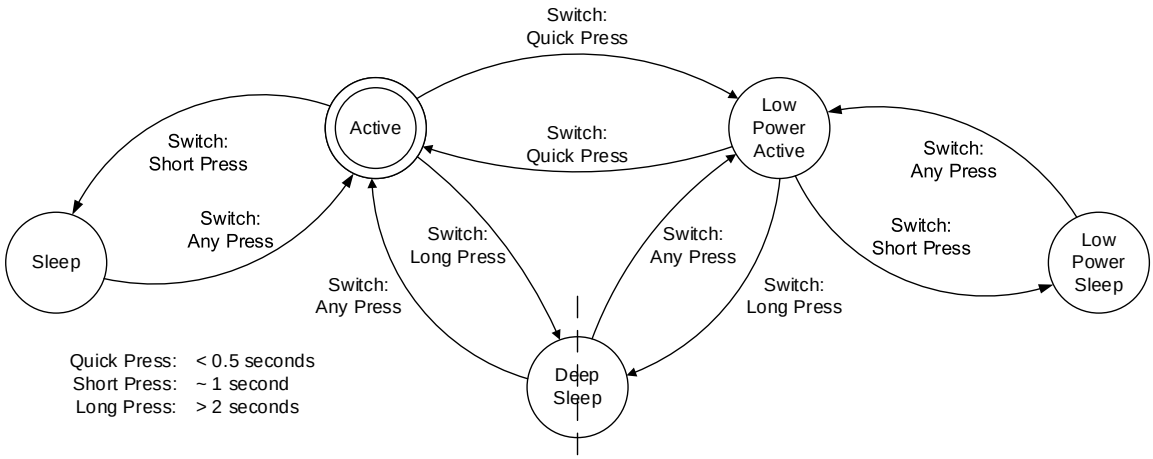
次のサンプル コードは、PSoC 6 MCU の電力モードと電力削減技術を実演するために、本アプリケーション ノートに含まれいます。

E.1 CE219881 - PSoC 6 MCU の電力モード間の切換え

このサンプル コードは、低消費電力モードの開始／終了の方法と、アクティブからディープ スリープまたはスリープへの遷移方法を示します。またこのサンプル コードは、どちらかのモードになった後、どのようにウェイクアップしてアクティブ モードのいずれかに戻るかも示します。

このプロジェクトでは、ボタン スイッチを使用し、電力モード間を遷移し、異なる LED カラーを表示して現在の電力モードを示します。図 14 は、遷移を実行するためにファームウェアで実装されたステート マシンを示します。詳細は「CE219881 - PSoC 6 MCU Switching between Power Modes」を参照してください。

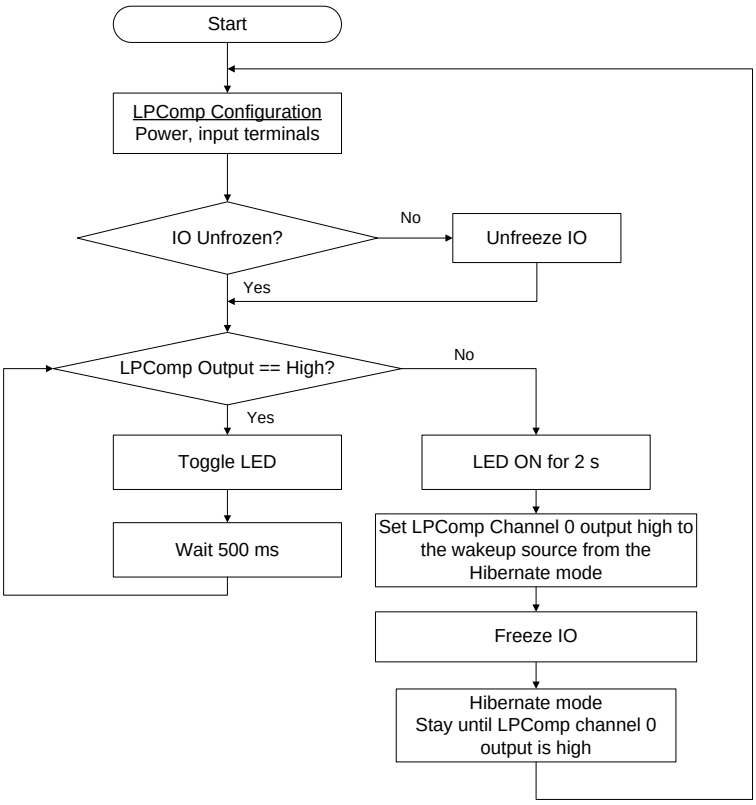
図 14. 電力モード ステート マシン



E.2 CE218129 - 低消費電力コンパレータによる PSoC 6 MCU のハイバネートからのウェイクアップ

このサンプル コードは、LPComp 内部リファレンス電圧のコンポーネント オプションを設定する方法、および LPComp ドライバーを使用して GPIO から外部入力を設定する方法について示します。このプロジェクトは、ハイバネート電力モード遷移の良い例です。ここでは、ハイバネートの前後で GPIO を取り扱うする方法、およびハイバネートのウェイクアップ ソースを登録する方法について示します。図 15 に、このプロジェクトの基本フローを示します。詳細は「CE218129 - PSoC 6 MCU Wakeup from Hibernate Using a Low-Power Comparator」を参照してください。

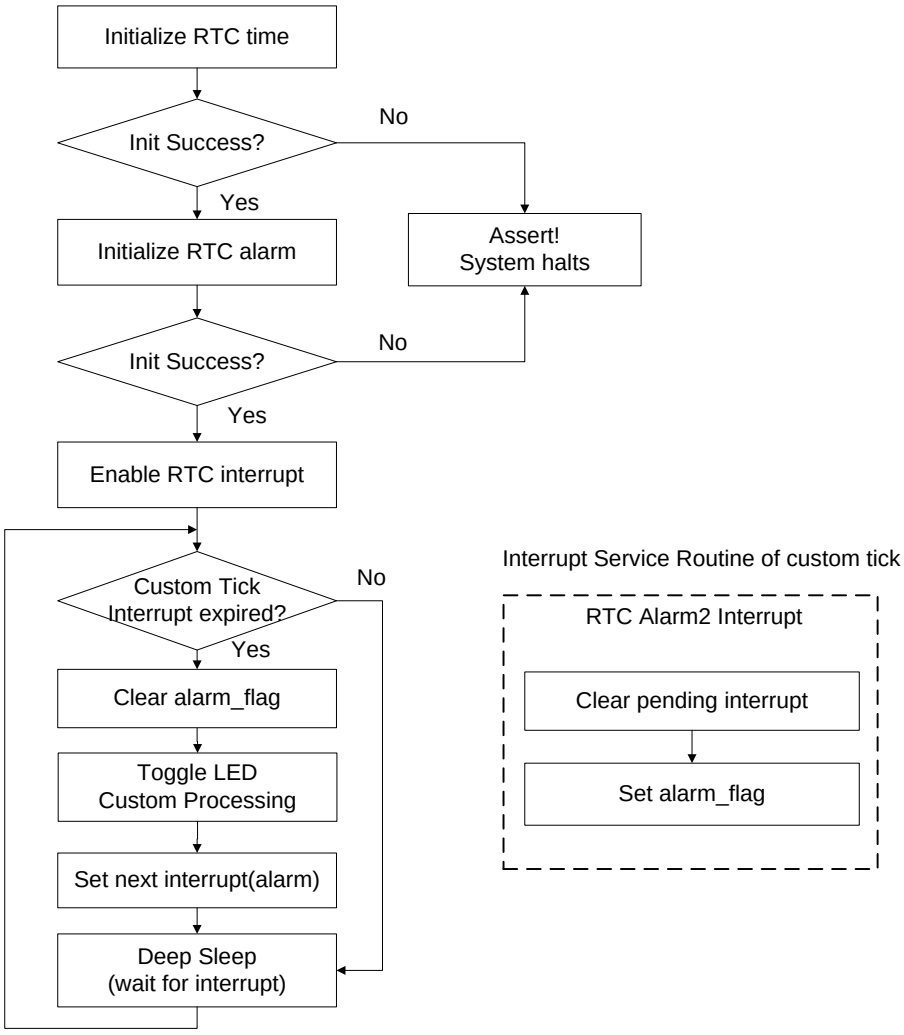
図 15. LPComp 入力を使用するハイバネート モードからのウェイクアップ



E.3 CE218542 - RTC アラーム割込みを用いた PSoC 6 MCU カスタム ティック タイマー

このサンプル コードは、PDL RTC ドライバーAPI を使用して周期的アラーム割込み用に RTC レジスタを設定する方法を示します。 このプロジェクトは、アクティブとディープ スリープ モードを使用し、消費電力を節約します。 LED をトグルし、割込み周期を表示するために GPIO 出力が含まれます。詳細は「CE218542 - PSoC 6 MCU Custom Tick Timer Using RTC Alarm Interrupt」を参照してください。

図 16. アラーム割込みを使用した RTC 周期的ウェイクアップ タイマー



改訂履歴

文書名: AN219528 - PSoC® 6 MCU 低消費電力モードおよび節電技術

文書番号: 002-23481

版	ECN	変更者	発行日	変更内容
**	6214780	IYM	07/13/2018	これは英語版 002-19528 Rev. **を翻訳した日本語版 Rev. **です。

ワールドワイドな販売と設計サポート

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

製品

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

PSoC®ソリューション

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

サイプレス開発者コミュニティ

フォーラム | WICED IOT フォーラム | プロジェクト | ビデオ | ブログ | トレーニング | コンポーネント

テクニカル サポート

cypress.com/support

本書で言及するその他すべての商標または登録商標は、それぞれの所有者に帰属します。

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