PSOC™ 6 MCU 低消費電力モードおよび節電技術

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 (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。CY8C62x4 ファミリデバイスの低電力アナログ周辺機能の詳細については、AN230938 – PSOC™ 6 MCU low- power analog を参照してください。

さらにサンプル コードをお求めでしょうか？以下のとおり対応いたします。

数百に増え続ける PSOC™ MCU のサンプルコード リストにアクセスするにはサンプルコード ウェブページをご覧ください。ここでビデオ トレーニング ライブラリも探せます。

目次

	本書について	1
	目次	2
1	**はじめに**	4
2	**電力モード**	5
2.1	電力モードの遷移	5
2.2	CPU スリープとウェイクアップ命令	7
2.3	低消費電力アシスタント	7
2.3.1	低消費電力アシスタント機能	7
2.3.2	PSOC™ 6 MCU デバイスの LPA 設定	8
2.3.3	接続デバイスの LPA ソフトウェア設定	8
2.4	サブシステムの可用性と消費電力	11
2.4.1	サブシステムの可用性	11
2.4.2	電力消費量の概算	11
2.5	サンプル ケース シナリオ	12
2.6	システム電源管理 (SysPm) ライブラリ	13
2.6.1	概要	13
2.6.1.1	HAL	13
2.6.1.2	PDL	13
2.6.2	モード遷移関数	13
3	**PSOC™ 6 MCU 消費電力管理**	17
3.1	コア電圧選択	17
3.1.1	リニア レギュレータと降圧レギュレータ	17
3.2	ULP モード クロック	18
3.3	バックアップ電源ドメイン	19
3.3.1	V_{Backup} 電源	19
3.3.2	バックアップドメインリセット	23
3.3.3	外部 PMIC 制御	23
3.3.4	ウェイクアップソース	25
3.3.5	バックアップデータレジスタ	26
4	**他の節電技術**	27
4.1	PSOC™ 6 MCU による電流経路の開閉	27
4.2	未使用ブロックの無効化	28
4.3	DMA によるデータ移動	28
4.4	周期的ウェイクアップ タイマー	28
4.5	CPU の無効化	29
4.6	CPU 間のタスクの分割	29
4.7	クロック	30

目次

4.8 GPIO 32
4.9 SRAM 33
4.10 TCPWM 33
4.11 SCB 34
4.12 オーディオサブシステム 35
4.13 USB 36
4.14 低消費電力コンパレータ 36
4.15 SAR ADC 36
4.16 電圧 DAC 37
4.17 オペアンプ 37

5 電力測定 38
5.1 DMM による電力測定 38
5.2 消費電力の概算 38

6 電源保護システム 39
6.1 ハードウェア制御 39
6.1.1 電圧低下検出 (BOD) 39
6.1.2 低電圧検出 (LVD) 39
6.1.3 過電圧検出 (OVD) 39

7 まとめ 40

A 電力モードのまとめ 41
A.1 電力モードとウェイクアップ ソース 41

B サブシステムの可用性 43
B.1 異なる電力モードで使用できるリソース 43

C コールバック関数例 45
C.1 HAL を使用したコールバック関数の登録 45
C.2 HAL を使用したカスタム コールバック関数の実装 46

D サンプルコード 47
D.1 CE219881 – PSOC™ 6 MCU の電力モード間の切り替え 47
D.2 CE231030 – 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 47
D.3 CE237278 - HAL: RTC alarm periodic wakeup 48
D.4 CE226306 – PSOC™ 6 MCU 電力測定 49

12 関連ドキュメント 52

改訂履歴 53

Trademarks 54

免責事項 55

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™ MCU アーキテクチャに関する基本知識と、ModusToolbox™ software 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、AN221774 - PSOC™ Creator での PSOC™ 6 MCU 入門または AN228571 – ModusToolbox™ソフトウェアを使用する PSOC™ 6 MCU 入門を参照してください。

モノのインターネット (IoT) システムを設計している場合、システム全体の電力に影響する接続性に関連した電力に関する追加の考慮事項があります。これらの考慮事項はこのアプリケーションノート (AN) の範囲外ですが、低消費電力 IoT システムに焦点を当てた特定のアプリケーションノートがあります。AN227910 – AIROC™ Wi-Fi & Bluetooth®コンボチップおよび PSOC™6 MCU を使用した低電力システム設計では、Wi-Fi, Bluetooth®, および PSOC™ 6 MCU システムの低消費電力の最適化について説明します。

2 電力モード

2.1 電力モードの遷移

PSOC™ 6 MCU は、デバイス全体に影響を与えるシステムモードと、1 つの CPU のみに影響を与える標準 Arm® CPU モードに分けられる 7 つの電力モードを備えます。システムの電力モードは、低消費電力 (LP)、超低消費電力 (ULP)、ディープスリープ、およびハイバネートです。Arm® CPU の電力モードは、アクティブ, スリープ, ディープスリープです。これらはシステム LP および ULP 電力モードで利用可能です。表 1 にデバイス動作の消費電力モードを示します。

表 1 PSOC™ 6 MCU 電力モード

電力モード	説明
システム LP	• すべてのリソースが最大の電力と速度で使用可能 • すべての CPU 電力モードに対応
システム ULP	• すべてのブロックが利用可能だが、コア電圧が低下し、高周波クロック周波数が低下 • すべての CPU 電力モードに対応
CPU アクティブ	• 通常の CPU コード実行 • システム LP および ULP 電力モードで使用可能
CPU スリープ	• CPU はコード実行を停止 • システム LP および ULP 電力モードで使用可能
CPU ディープスリープ	• CPU はコード実行を停止 • システムのディープスリープ開始を要求 • システム LP および ULP 電力モードで使用可能
システム ディープスリープ	• すべての CPU が CPU ディープスリープ状態のときに発生 • CPU、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
システム ハイバネート	• コアとクロックはオフ • GPIO 出力状態は凍結 • 低消費電力コンパレータ、RTC アラーム、および専用 WAKEUP 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。 • SRAM はデータ保持されない

図 1 は、割込み、ファームウェア アクション、およびリセット イベントを含むさまざまなイベントとアクションに基づいて電力モードの遷移がどのように行われるかを示します。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 architecture technical reference manual および付録 A を参照してください。

2 電力モード

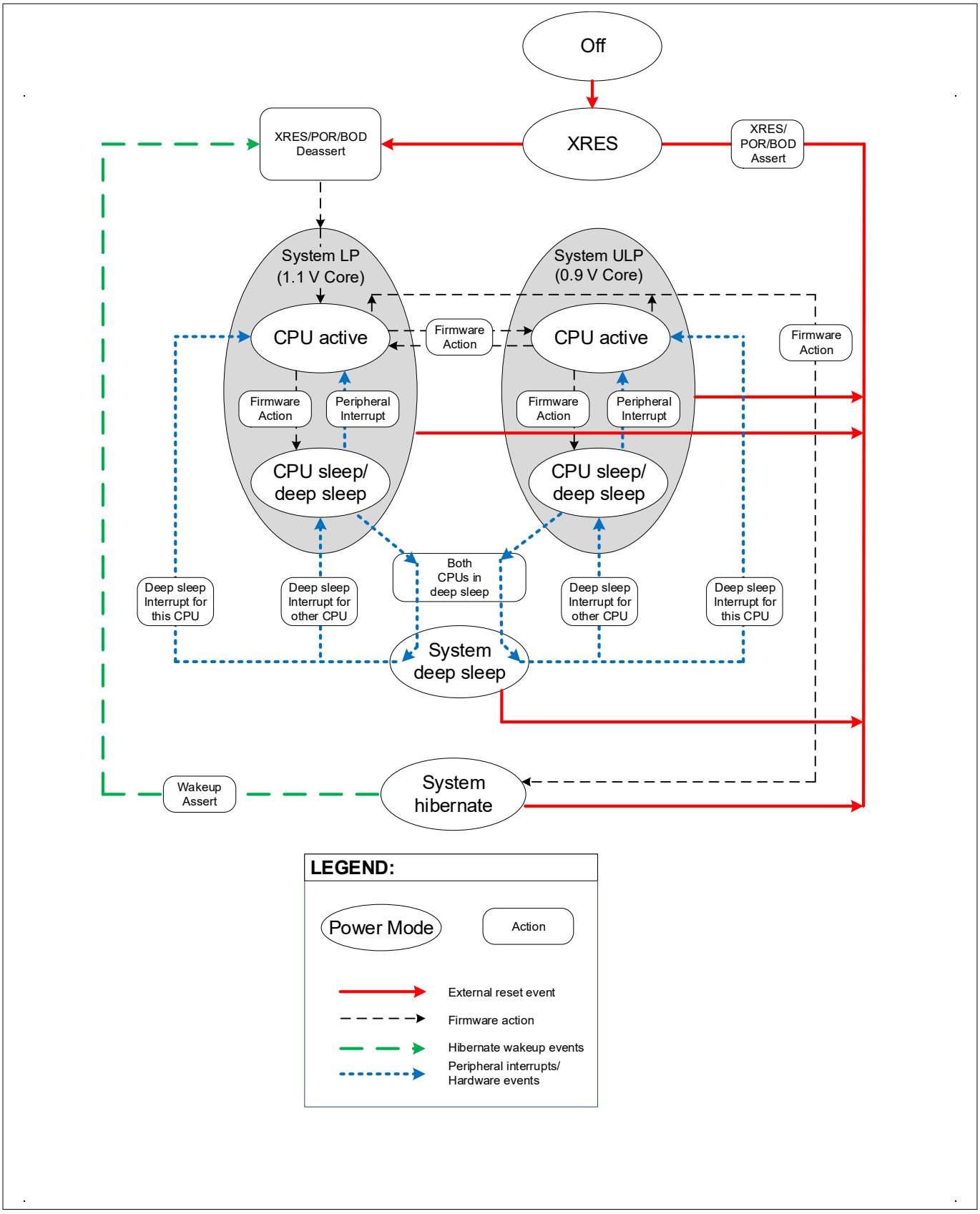


図 1　　PSOC™ 6 MCU デバイス電力モードの遷移

2.2 CPU スリープとウェイクアップ命令

Arm® Cortex® CPU 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。CPU が__WFI を実行した後、CPU 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 CPU をウェイクアップさせるために使用されます。CPU 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® system control register user guide を参照してください。このプロセスは、SysPm PDL ライブラリおよび HAL API に実装されます。

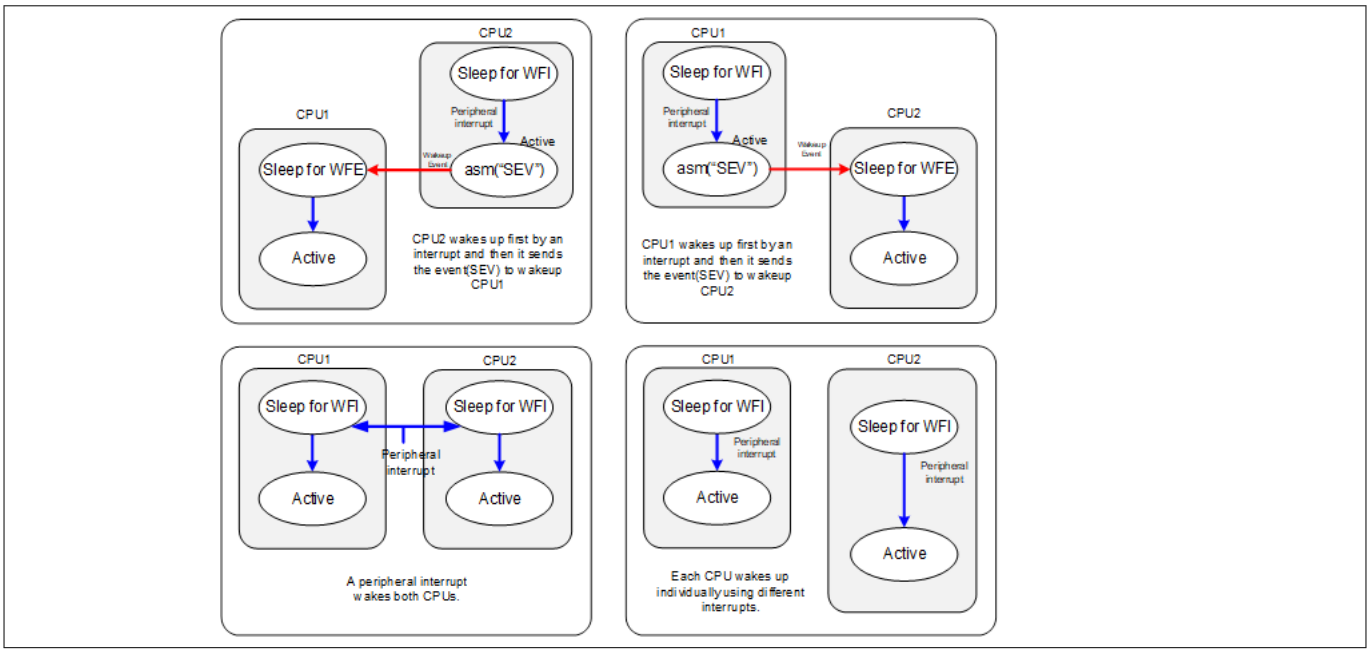


図 2 マルチ CPU のスリープとウェイクアップ ケース

CPU の電力モードはシステムの電力モードと異なります。図 2 に、それぞれの CPU が他の CPU の状態とは独立し、個別のスリープモードに対応することを示します。両方の CPU がディープスリープになると、デバイスはシステムディープスリープモードになります。詳細は AN215656 – PSOC™ 6 MCU デュアルコア システム設計を参照してください。

2.3 低消費電力アシスタント

2.3.1 低消費電力アシスタント機能

低消費電力アシスタント (LPA) は、デバイスとシステムの両方の電源オプションをセットアップするための使いやすい GUI を提供します。LPA の目標は、実世界のユーザーアプリケーションでデータシートの電力値を迅速に達成できるようにすることです。デバイスでサポートされている各電源オプションについて、ModusToolbox™ device configurator には対応するセクションがあり、他の周辺機器の設定と同じ方法で電源リソースを設定できます。

PSOC™ 6 MCU デバイスのアシスタントを起動するためには、図 3 に示すように、PSOC™ 6 MCU パーツ番号タブおよび System サブタブから、Power リソースを選択します。接続デバイスの場合、図 4 に示すように、Wi-Fi または Bluetooth®デバイスの部品番号タブの下で、Power セクションで BT または Wi-Fi リソースを選択します。Power Configurator の上部にあるドキュメントセクションの LPA ドキュメントを参照してください。

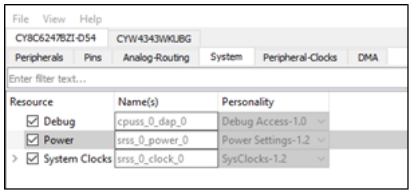


図 3　　PSOC™ MCU の LPA ソフトウェア選択

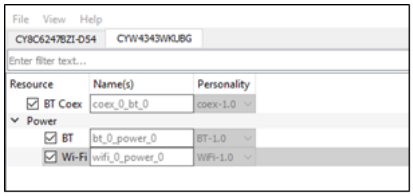


図 4　　接続デバイス LPA 選択

2.3.2　PSOC™ 6 MCU デバイスの LPA 設定

LPA の PSOC™ 6 MCU デバイスセクションを使用して、図 5 に示すようにデバイスの初期電力設定を構成します。LPA の General セクションを使用して、システム電力モード, コアレギュレータ, および Vbackup ドメイン機能の設計に必要な最高の電力設定を構成します。最高電力モードを設定したら、RTOS セクションを使用して、システムが自動的に移行する最低電力モードを設定します。RTOS は通常、すべての動的モード遷移を処理するため、多くの設計では、これが必要なすべての電力モード構成です。より高度なユースケースでは、HAL または SysPm PDL ライブラリの電源機能をいつでも呼び出して、電源設定を動的に変更したり、電力モードの遷移を開始したりできます。

スリープおよびディープスリープ電力モードからの CPU およびシステムウェイクアップは、設定されたペリフェラル割込みがトリガされると発生します。ModusToolbox™ device configurator 内の周辺機器のコンフィギュレーターで目的のウェイクアップ周辺機器の割込みを有効にすることで、スリープおよびディープスリープウェイクアップ割込みを構成できます。システムが実行時に HAL または SysPm 呼び出しを使用して休止状態に入る場合、LPA Wakeup pins セクションは休止状態ウェイクアップソースの選択を簡素化します。

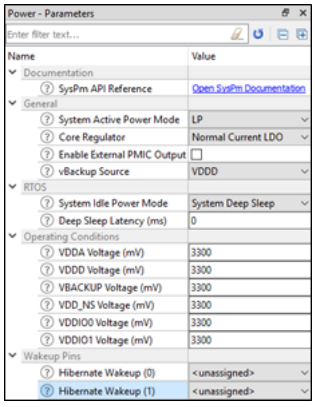


図 5　　電源パラメーター: PSOC™ MCU の LPA ソフトウェア

2.3.3　接続デバイスの LPA ソフトウェア設定

LPA ソフトウェアの「接続デバイス」セクションを使用して、Wi-Fi および Bluetooth®電源オプションを構成します。AIROC™ Bluetooth®接続デバイスは、もっとも低電力のモードで自動的に動作するように設計されます。主な電源オプションは、図 6 に示すように、Bluetooth®デバイスホストウェイクアップトリガを有効にすることです。

2 電力モード

Bluetooth®ホストウェイクアップにより、PSOC™ 6 MCU ホストデバイスはシステムディープスリープまたはハイバネートモードに入り、Bluetooth®操作を処理する必要があるときに GPIO ピン割込みを生成することにより、Bluetooth®接続デバイスが MCU をウェイクアップするのを待ちます。

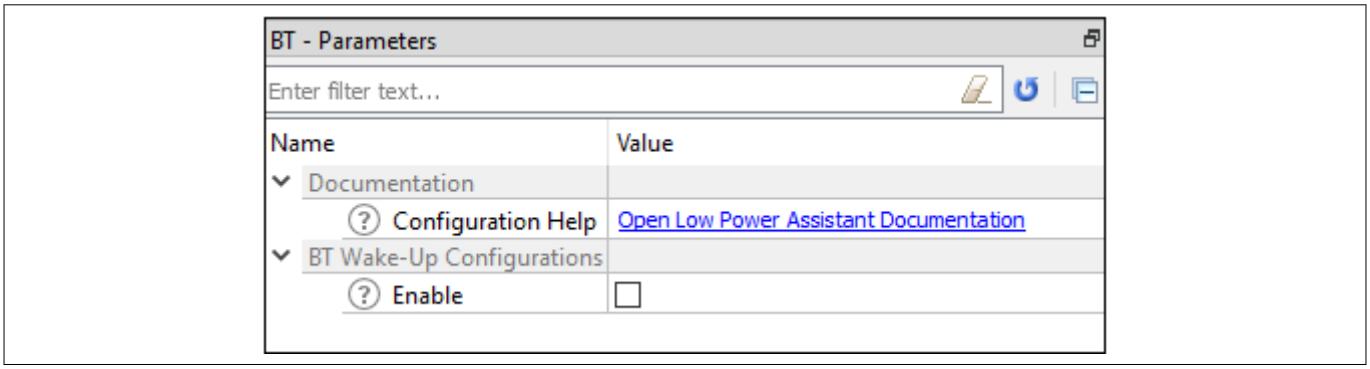


図 6　　LPA ソフトウェア: Bluetooth®電源パラメーター

インフィニオンの AIROC™ Wi-Fi & Bluetooth®接続デバイスは、図 7 に示すように、サポートされるアプリケーションおよびネットワーク機能のニーズに基づいて構成できる幅広い電力最適化手法をサポートします。Wi-Fi の 3 つの主要な低電力機能は、ホストウェイクアップ, オフロード, およびフィルターです。

- **Host wakeup** により、PSOC™ 6 MCU ホストは、Wi-Fi デバイスが MCU 処理を必要とするまで、システムディープスリープまたはハイバネートモードに移行できます。ホスト MCU がウェイクアップする必要がある場合、Wi-Fi デバイスは GPIO ピン割込みをトリガします。他のほとんどの低消費電力 Wi-Fi 機能を使用するためには、ホストウェイクアップが必要です。
- **Offload** により、MCU のデバイススタック処理を Wi-Fi デバイスの内部プロセッサに移動 (オフロード) できます。これにより、MCU ホストが他のタスクに専念したり、低電力モードで費やせる時間が増加します。
- **Filter** は Wi-Fi デバイス上で実行され、フィルター条件が満たされるまでホスト MCU の起動を回避します。例は、MQTT パケットにのみ応答する必要がある IoT デバイスです。デバイスは、MQTT フィルターを有効にして、他のすべてのネットワークトラフィックを無視し、ホスト MCU デバイスを低消費電力モードで長く保てます。

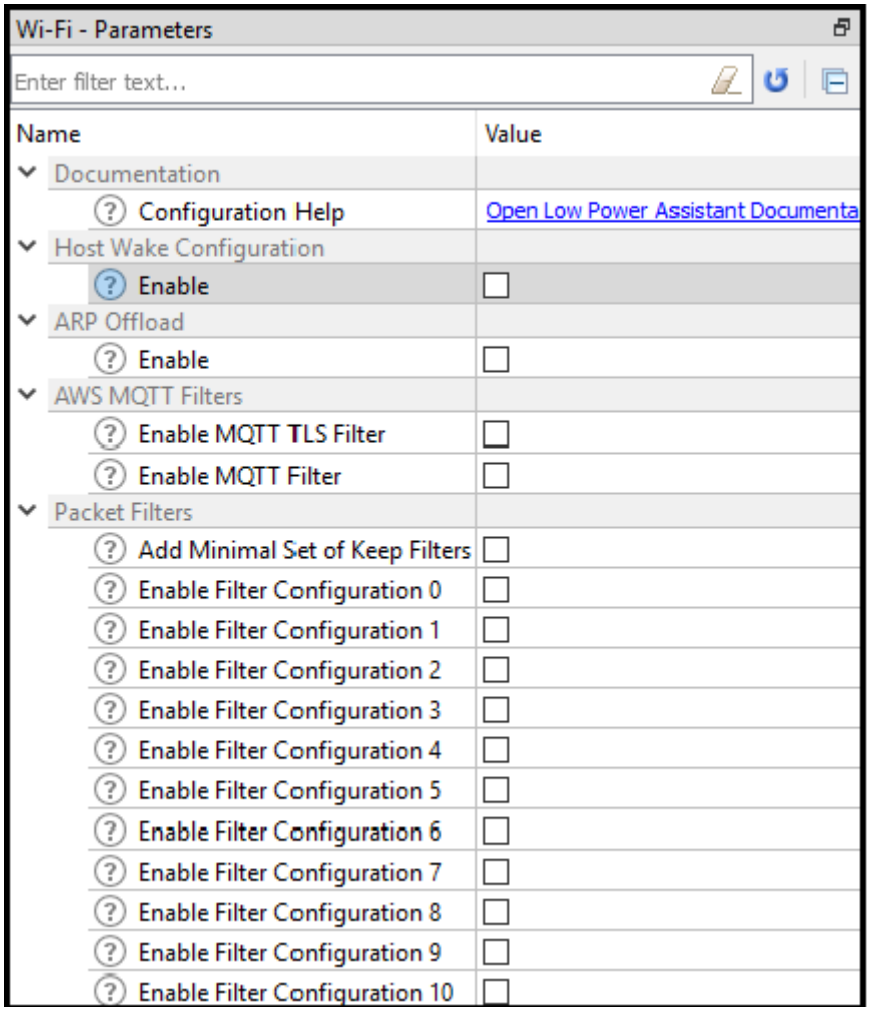


図 7　　LPA ソフトウェア: Wi-Fi 電源パラメーター

2.4 サブシステムの可用性と消費電力

2.4.1 サブシステムの可用性

各サブシステム リソースは、さまざまなシステム電力モードによって異なる動作をします。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 2 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。

2.4.2 電力消費量の概算

特定の条件の消費電力データについては、デバイスのデータシートを参照してください。最高の消費電力を実現するための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消費電力はデータシートと異なる場合があります。

2.5 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削減できます。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、効果的に使用される電力モードはわずかです。

表 2 電力モードのサンプル ケース シナリオ

電力モード	ウェアラブル デバイス	エアコン	リモコン	温度計
システム LP CPU アクティブ	ユーザーによる GUI 対話	モーター使用	–	Bluetooth® LE 経由で通信
システム ULP CPU アクティブ	心拍数を処理	–	コマンドを送信	温度を読み出す LCD に結果を更新
システム LP CPU スリープ	–	–	–	–
システム ULP CPU スリープ	アナログ ブロックは心拍を検出	–	–	–
システム 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 秒ごとにウェイクアップ
システム 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	–

2.6 システム電源管理 (SysPm) ライブラリ

2.6.1 概要

2.6.1.1 HAL

ハードウェアアブストラクションレイヤ (HAL) は、電源状態を変更するためのインターフェースを提供します。

HAL ライブラリの API を使うことで、システム電源状態 (ノーマル、低電力、ハイバネート) と同じように、ユーザーはサポートされているすべての CPU 電源状態 (アクティブ、スリープ、ディープスリープ) を実現できます。

ライブラリはまた、要求された電源状態の遷移時に呼び出されるコールバック関数を登録 (または解除) するオプションも提供します。

ディープスリープに入る機能は、API を使ってロックすることができ、クリティカルブロックの実行が完了すると解除されます。

2.6.1.2 PDL

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 レジスタを構成し、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。PDL は、ターゲットデバイスのほぼすべてのハードウェアリソースへの直接アクセスを提供します。これにより、レジスタとビット構造を理解して直接アクセスする必要が減ります。

図 1 に示すように、PDL では system power management (SysPm) API が電力モード変更の関数を提供します。また図 8 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も登録できます。ModusToolbox™で Project Creator ツールを使ってアプリケーションを作成する際に、PDL は GitHub からダウンロードされますが、ModusToolbox™ IDE と一緒に PDL はインストールされません。

2.6.2 モード遷移関数

図 1 に、電力モードのファームウェアの遷移を示します。SysPm は、CPU スリープ、CPU ディープスリープ、システム ハイバネート、システム LP、システム ULP 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。

コールバック機能	説明
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 アクションを実行します。

SysPm ドライバーは登録、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 8 に示すように、PDL および HAL 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。定義されたペリフェラルコールバックを登録するか、またはカスタム コールバックの作成することを選択できます。SysPm 遷移関数は、登録されたコールバックを順番に実行します。最初に登録された関数が最初に実行されます。

2 電力モード

HAL を使用したコールバックの登録と実装の詳細は、付録 C および付録 D.1 のサンプルコード CE219881 - PSOC™ 6 MCU switching between power modes を参照してください。これは CPU アクティブ, CPU スリープ, システム LP, システム ULP, およびシステム ディープスリープのモード遷移例です。

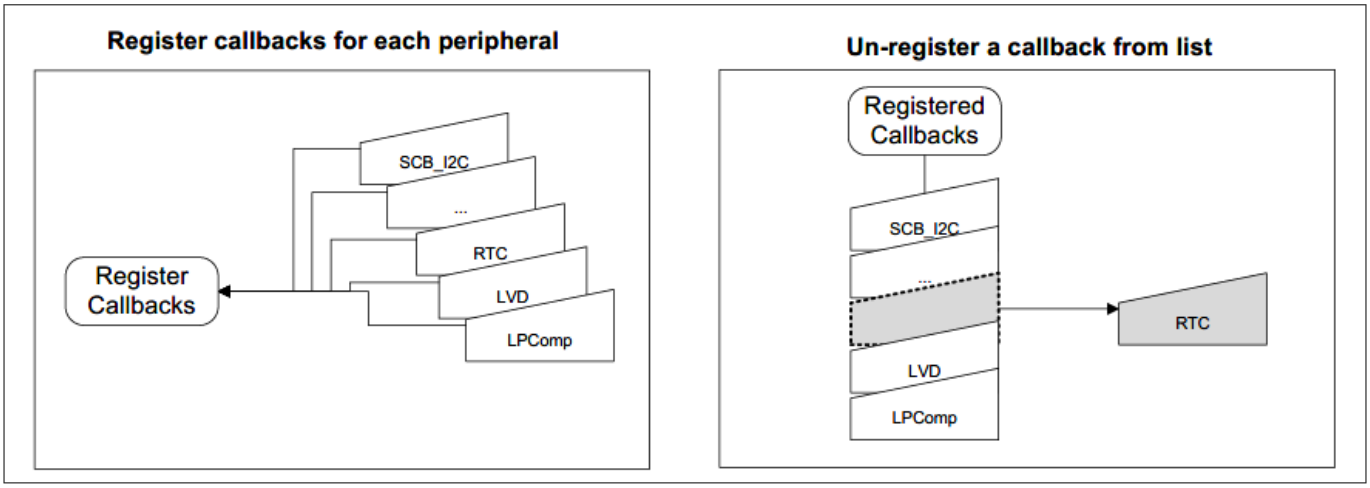


図 8　　電力モード コールバックの登録と登録解除

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。CPU スリープおよび CPU ディープスリープの各モードは、Arm®スリープ命令が使用されます。コード実行は停止し、CPU スリープ電力モード間の割込みを待ちます。図 8 は、スリープ命令__WFI()または__WFE()を呼び出した後、ウェイクアップ ソースを待機している CPU を示します。ウェイクアップ後、デバイスは自動的に CPU アクティブに遷移します。

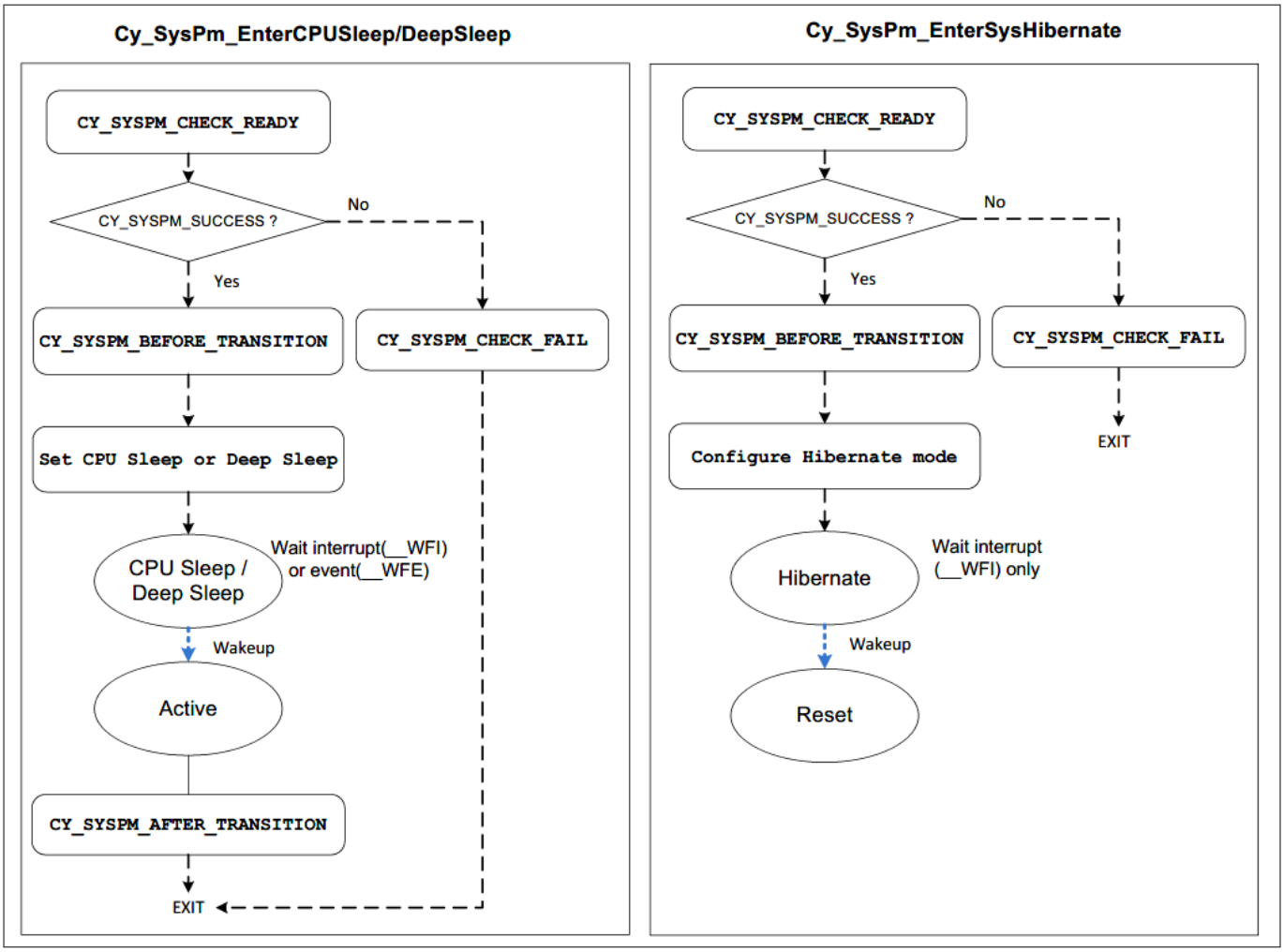


図 9　　スリープ/ディープスリープ/ハイバネート モード遷移

システム LP とシステム ULP 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。システム LP と ULP モードの開始は、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、遷移は遅滞なく行われます。図 10 に示すように、SysPm PDL および HAL 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。詳細はコア電圧選択、ULP モード クロックおよびペリフェラル ドライバ 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。

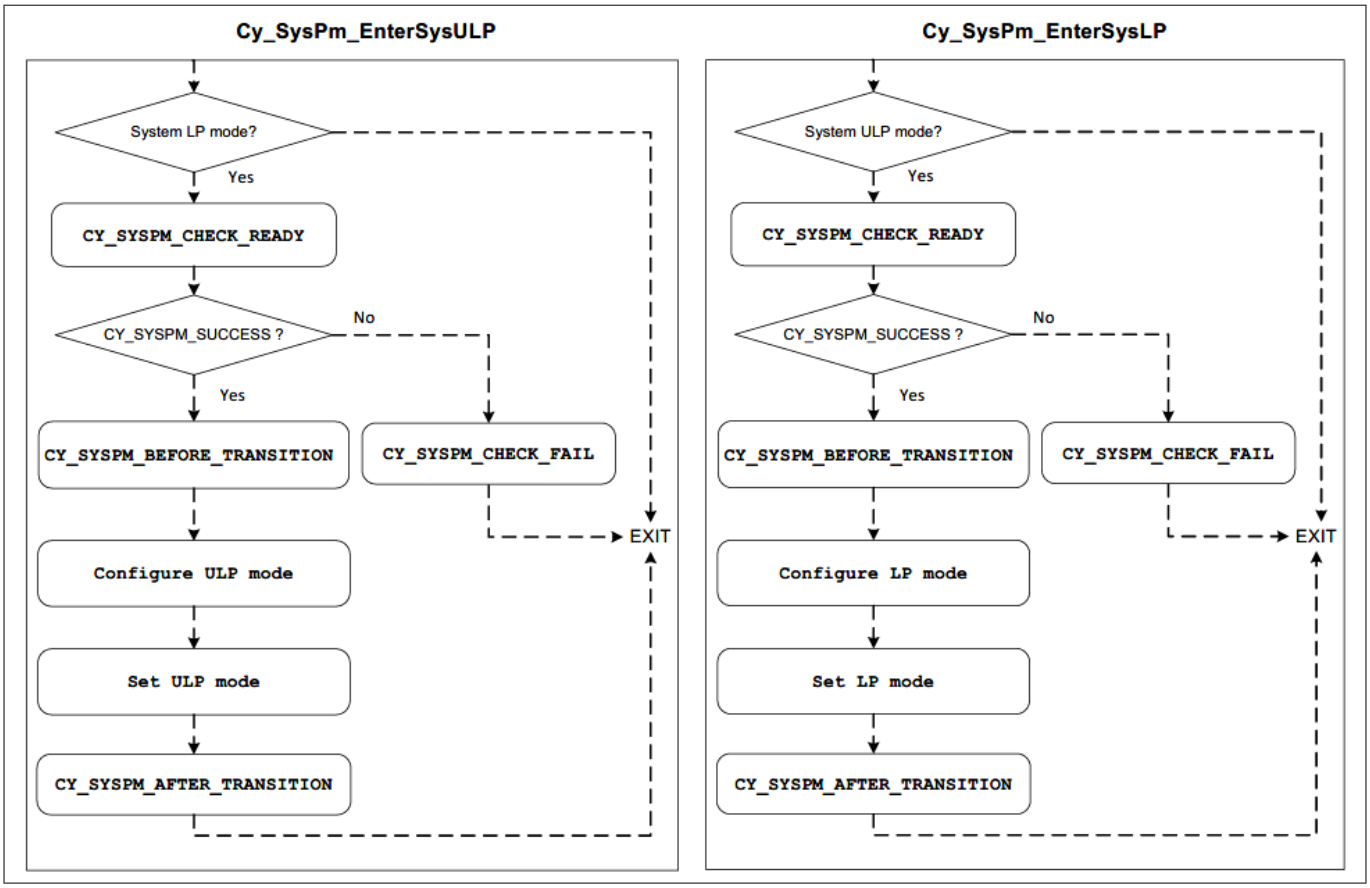


図 10　　低消費電力モードの遷移

3 PSOC™ 6 MCU 消費電力管理

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、低ドロップアウト (LDO) およびシングル入力マルチ出力 (SIMO)、またはシングル入力シングル出力 (SISO) 降圧を含む複数のオンチップレギュレータをサポートして、コア電源用の VCC を生成します。LDO は、高電流モード (通常) で最大 300 mA、低電流モードで 25 mA を供給できます。降圧レギュレータは、1 つの出力に最大 20 mA、SIMO 降圧の両方の出力に合計 30 mA を供給できます。SIMO 降圧レギュレータは、通常の負荷条件で効率を向上させます。SIMO 降圧レギュレータに切り替えた後、デバイスをリセットせずに LDO に切り替えられません。

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オプション

	出力	最大負荷	最大クロック周波数
LDO	0.9 V	25/300 mA (低電流モード/通常モード)	Arm® Cortex®-M4 (CM4) 用は 50 MHz Arm® Cortex®-M0+ (CM0+) 用は 25 MHz
	1.1 V	25/300 mA (低電流モード/通常モード)	サポート可能な最大クロック周波数を許可
降圧	0.9 V	20 mA	CM4 用は 50 MHz CM0+用は 25 MHz
	1.1 V	20 mA	サポート可能な最大クロック周波数を許可

3.2 ULP モード クロック

ULP モードへの遷移は、電力モード制御レジスタの設定によって行えます。前述したように、ULP モードの最大クロック速度の制限があるため、ULP 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 registers technical reference manual を参照してください。

図 11 に、登録されたコールバック関数で PDL 関数を使用し、LP と ULP モード間の遷移方法を示します。ULP モードのクロック制限のため、モード遷移前に Frequency-locked loop (FLL) クロック速度または HFClk のいずれかを有効な周波数に調整する必要があります。FLL 周波数を変更すると、FLL から派生した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、コールバックを使用したクロック調整の例を提供します。

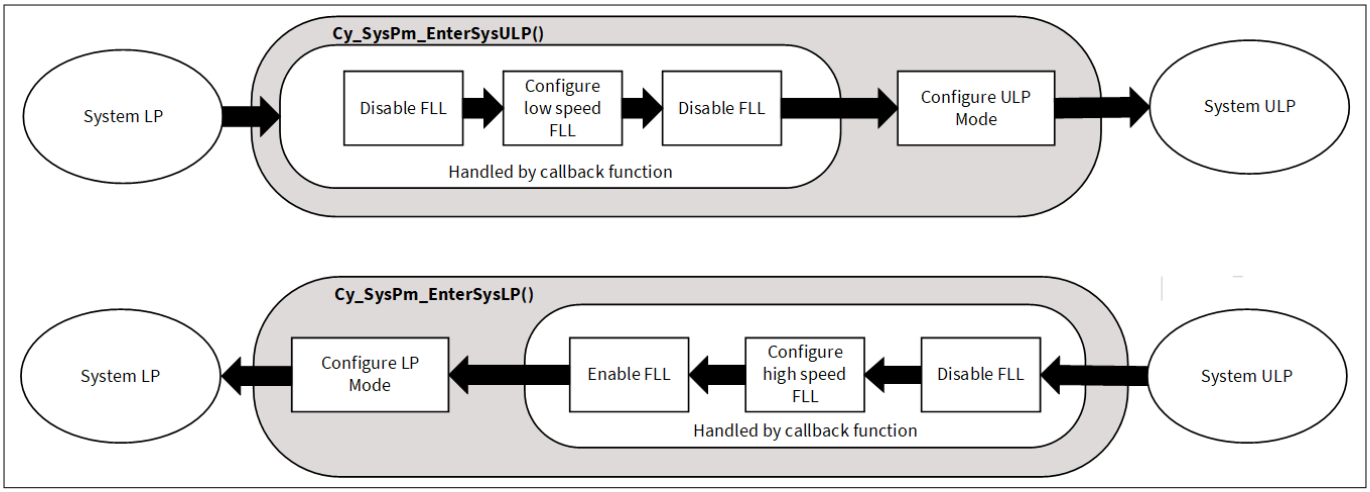


図 11 LP モード開始/終了の遷移

3.3 バックアップ電源ドメイン

バックアップドメインは電力モードではなく、デバイスの電力モードのいずれかでアクティブにできる常時オンのリソースグループです。

図 12 に、メインデバイスの電源が取り外されている間は、別の電源が限られた数の機能へ電力を供給することを示します。これにより、基本レベルの機能を維持しながら、可能な限り低い電力状態が得られます。バックアップドメインには pmic_wakeup_in ピンが含まれます。このピンは、システムで生成された信号がデバイスの再電力供給をトリガし、CPU の有効電力状態に戻します。

リアルタイムクロック (RTC) により、定期的かつ正確な時間ベースのウェイクアップトリガが可能です。V_{BACKUP} 電源を使用して RTC に継続的に電力を供給することにより、デバイスの全電力状態に関係なく正確な時間を維持できます。ウェイクアップは、pmic_wakeup_out ピンに HIGH 論理レベルを出力することによって実現されます。pmic_wakeup_out ピンは、完全な PSOC™ 6 MCU デバイスに電力を供給する外部パワーマネジメント集積回路 (PMIC) を有効化するために、最も一般的に使用されます。

64 バイトの SRAM ストレージも維持されるため、デバイスの電源を入れ直したときに主要なシステム状態情報を保持できます。バックアップドメインの電力は、別のシステム電源、バッテリ、またはスーパーキャパシタから電力を供給できる V_{BACKUP} 電源から供給されます。他の PSOC™ 6 MCU デバイスとは別にバックアップドメインに電力を供給する必要がない設計では、V_{BACKUP} を V_{DDD} 電源に直接接続する必要があります。

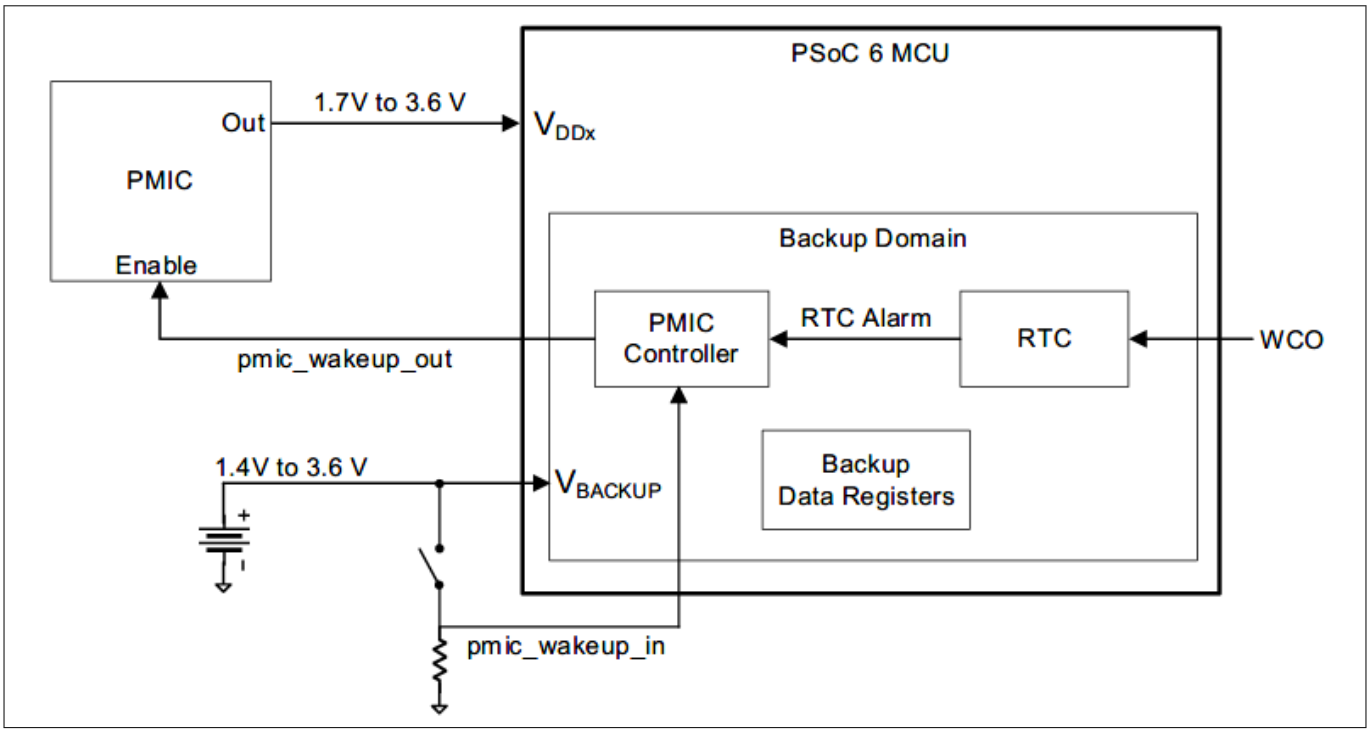


図 12 バックアップドメインのブロックダイヤグラム

3.3.1 V_{Backup} 電源

バックアップドメインに個別の V_{BACKUP} 電源を提供することにより、デバイスの大部分の電源の切断と、システムのハイバネート電力モードで発生する可能性のあるリークの排除を実現できます。バックアップドメインにデバイスの他の部分とは別に電力が供給されていない場合は、V_{BACKUP} と V_{DDD} を相互に接続する必要があります。

V_{DDBAK} という名前の内部バックアップ電源は、内部回路を使用して、V_{DDD} と V_{BACKUP} の間で自動的に切り替えられます。V_{DDD} が存在し、1.6 V を超える場合、バッテリまたはスーパーキャパシタの容量を節約するために、V_{BACKUP} 電源は無効になります。V_{DDD} が 1.6V を下回ると、V_{BACKUP} または V_{DDD} 電源の大きい方が選択され、低電圧電源接続が無効になります。V_{BACKUP} と V_{DDD} の間にシーケンスの制限はありません。ほとんどの設計では、外部電源の絶縁およびスイッチング回路は必要ありません。

4 つの一般的な V_{BACKUP} 電源接続オプションがあります。

1. **V_{DDD} に直接接続された V_{BACKUP}**: バックアップドメインが完全な PSOC™ 6 MCU デバイスでのみ使用される場合、個別の電源を使用する理由はありません。図 13 に示すように、V_{BACKUP} と V_{DDD} の電源ピンを直接接続する必要があります。

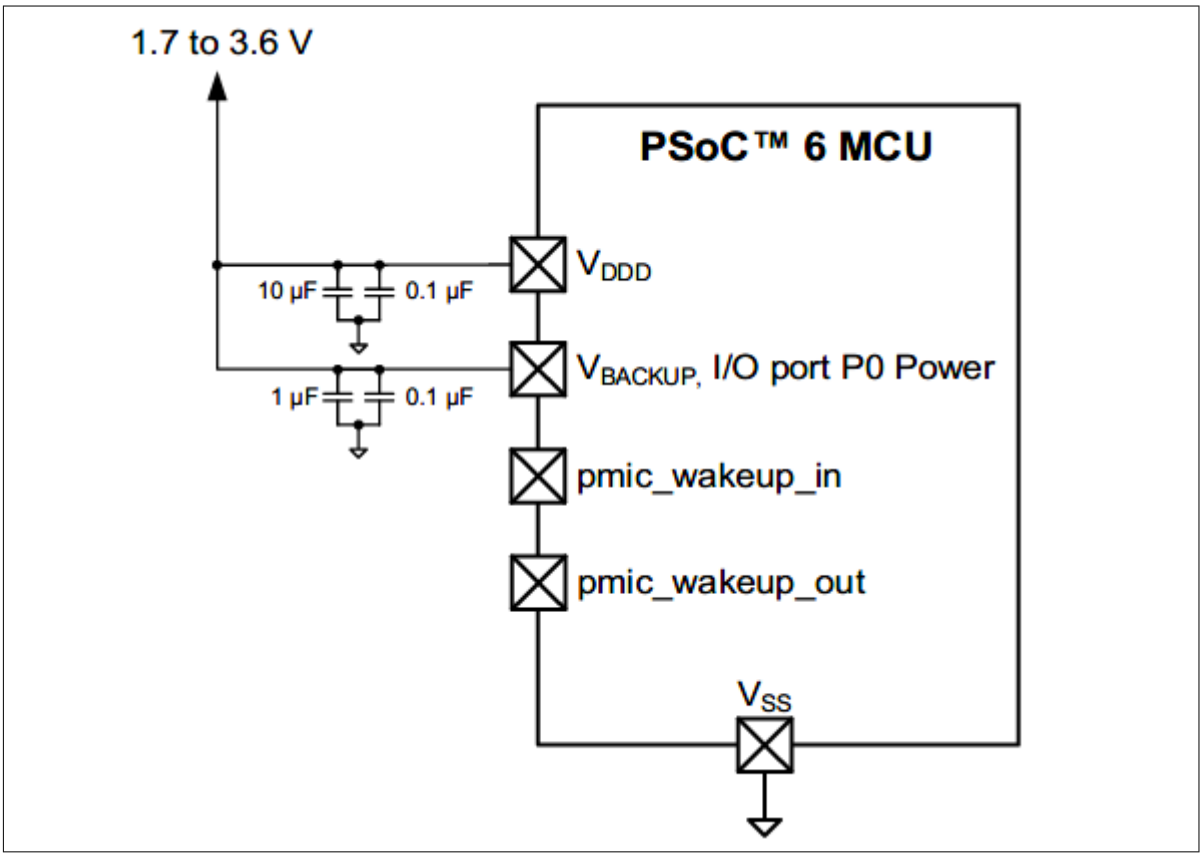


図 13　　バックアップドメイン結合接続

2. **スーパーキャパシタを搭載した V_{BACKUP}**: 図 14 に示すように、スーパーキャパシタを V_{BACKUP} 電源として使用する場合、再充電には 2 つのオプションがあります。`Cy_SysPm_BackupSuperCapCharge()` PDL 関数を呼び出すことにより、V_{DDD} 電源がアクティブな V_{DDBAK} 電源として選択されるたびに内部充電回路が有効になります。

   誤って充電回路を有効にしないように、キー値 0x3C を関数に渡す必要があります。この充電回路は正確ではなく、電流は V_{DDD} と V_{BACKUP} の差に基づいて 20 µA の範囲で変化します。充電時間は数時間程度です。

   2 番目のオプションは、充電時間の短縮など、独自の設計上の課題に最適化できる外部充電回路を使用することです。両方の充電方法を並行して使用できます。リーク電流を制限するために、外部電荷コンポーネントを選択する必要があります。スーパーキャパシタは V_{DDD} まで充電されるため、適切な電圧定格を備えていることを確認してください。

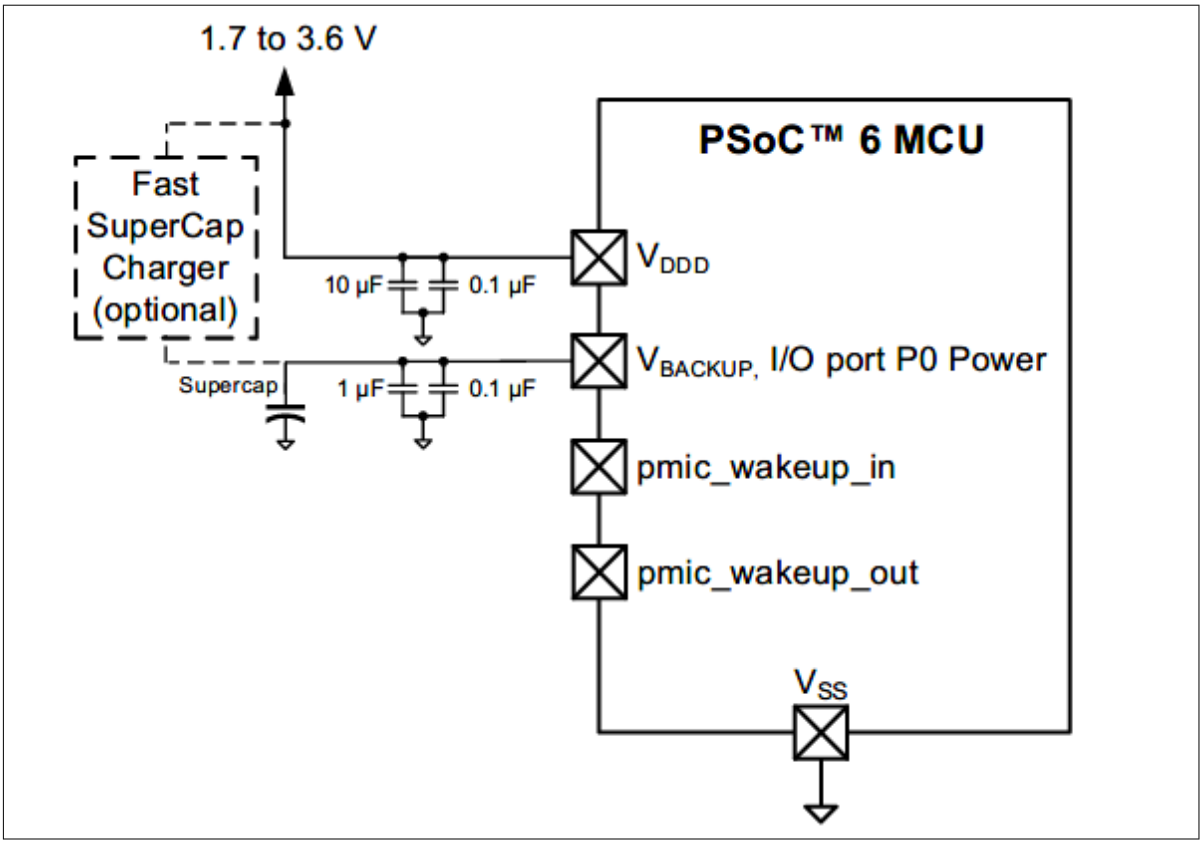


図 14　　バックアップドメイン スーパーキャパシタ接続

3. **バッテリ駆動の V_{BACKUP}**: 図 15 に、バッテリ駆動の V_{BACKUP} を示します。バッテリが V_{BACKUP} を供給する場合、スーパーキャパシタ充電回路はバッテリの安全メカニズムを提供しないため、使用しないことが重要です。バッテリが再充電可能な場合は、バッテリ固有の充電要件に合わせて、外部ボードレベルの再充電回路を提供できます。

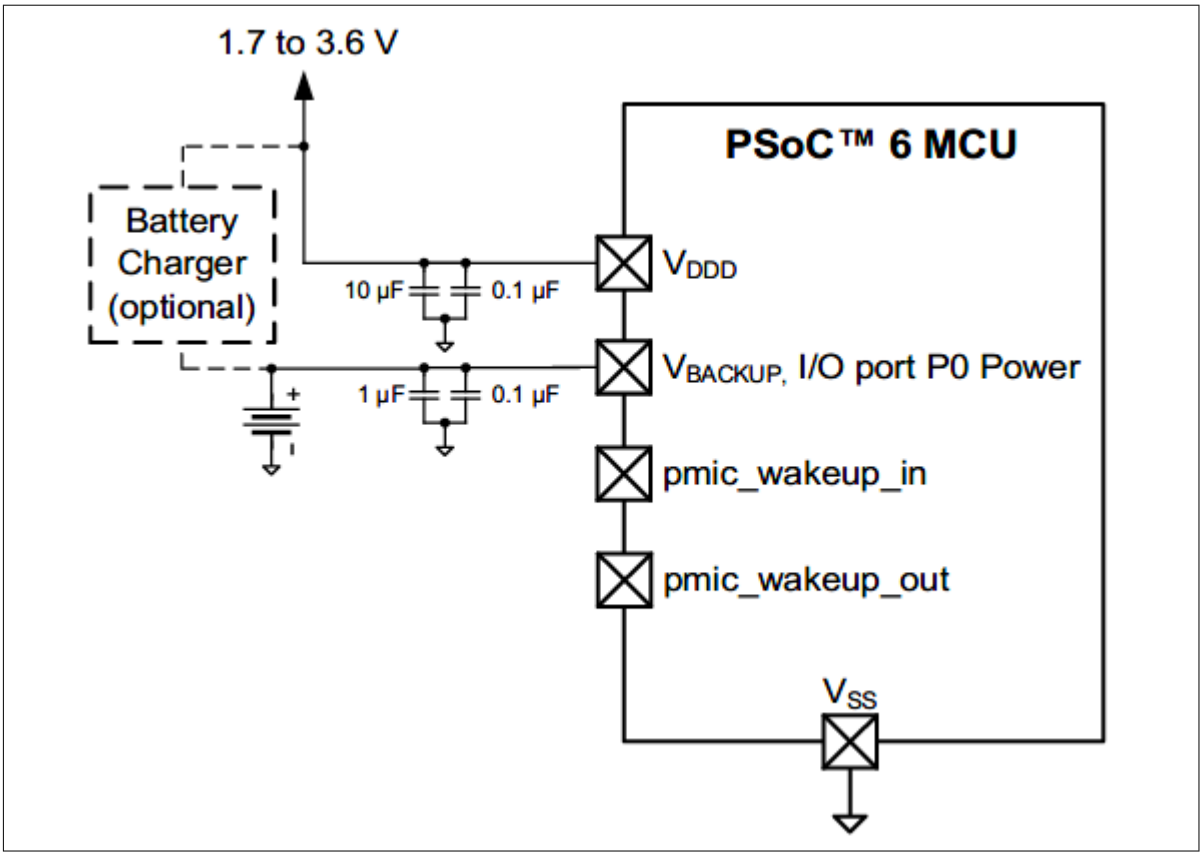


図 15　　バックアップドメインバッテリ接続

4. **システム提供の電源装置から電力を供給される V_{BACKUP}**: 常時オンの電源装置がシステムにすでに存在する場合は、V_{BACKUP} 電源装置として使用できます。

最大 512 KB, 1 MB, または 2 MB のフラッシュを備えた PSOC™ 6 MCU デバイスでは、Vmax 電源スイッチングロジックでは、V_{BACKUP} 電源を接続した後、少なくとも 50 µs の間、V_{DDD} の電源を少なくとも 1 回必要とすることに注意してください。これにより、スイッチング回路が正しい状態に設定され、リークパスがないことを確認して V_{DDD} 電源を V_{BACKUP} 電源から分離します。コードを実行する必要はありません。初期化されていない場合、V_{BACKUP} ドメインはバッテリからかなり多くの電流を消費し、早い段階でバッテリを消耗します。最大 256 KB のフラッシュを搭載した PSOC™ 6 MCU デバイス、およびすべての新しい PSOC™ 6 MCU デバイスでは、リークの増加は発生しません。このリークの問題が発生する可能性が高い 2 つのケースがあります。

1. PCB の製造中に、バックアップバッテリが取り付けられ、デバイスは高リーク状態で保管されます。その後、製品が完全に組み立てられ、テストされ、PSOC™ 6 MCU デバイスがプログラムされると、ロジック状態は正しく設定されますが、この期間中にバッテリが消耗した可能性があります。
2. 最終製品にユーザーが取り外し可能なバッテリがあり、V_{DDD} の電源がオフになっている場合は、ユーザーがバッテリを交換してスイッチロジックを低電流状態にリセットした後、PSOC™ 6 MCU の電源を再投入することが重要です。

V_{BACKUP} ドメインには低電圧検出回路が含まれないため、バッテリまたはスーパーキャパシタが 1.4 V 未満で放電しないようにすることが重要です。V_{BACKUP} 電源が切れた場合は、システムを再度有効にするためにデバイスの完全な電源を入れ直す必要があります。V_{BACKUP} は、AMUXBUSA を介して 10%スケーリングされたバージョンをルーティングすることにより、ADC によって直接測定できます。これにより、システムはスーパーキャパシタの高速再充電を開始するか、ユーザーにバッテリの交換を通知します。`Cy-SysPm_EnableBackupVMeasure()` PDL ドライバー関数を呼び出すことにより、測定が有効になります。

V_{BACKUP} は Port0 のすべてのピンにも電力を供給するため、V_{BACKUP} バッテリまたはスーパーキャパシタベースの電源から電力を供給しながら、設計での電流とリークを最小限に抑えるようにピンを設計することが重要です。

3.3.2 バックアップドメインリセット

V_{BACKUP} または V_{DDD} が存在する限り、パワーオンリセット (POR)、ブラウンアウト検出 (BOD)、ウォッチドッグタイマー (WDT)、および外部リセット (XRES) イベントではバックアップドメインはリセットされません。バックアップドメインは、V_{BACKUP} と V_{DDD} の両方が 0 V の場合、または特にファームウェアを介してリセットされた場合にのみリセットされます。ほとんどの場合、バックアップドメインをリセットしないでください。RTC 時間とバックアップデータレジスタに保存されているデータが失われる可能性があります。

バックアップドメインのリセットは、V_{DDD} 電源レールからの電力供給中に発生する POR、BOD、XRES、またはシステムの休止状態のウェイクアップには必要ありません。バックアップドメインのリセットは、V_{BACKUP} 電源から電源が切断された場合、または電圧低下が許可された場合にのみ発生する必要があります。

バックアップ電源ドメインは、`Cy_SysLib_ResetBackupDomain()` PDL 関数を呼び出すことにより、ファームウェア制御下でリセットされます。次に、ファームウェアは BACKUP_RESET.RESET レジスタフィールドの読み出し値が「0」であることを確認する必要があります。これは、バックアップドメインロジック回路のクロックレートが遅いため、バックアップドメインレジスタに書き込む前にリセットが完了したことを示します。バックアップドメインがリセットされた後、ILO も無効にしてから再度有効にして、確実にリセットする必要があります。`Cy_SysClk_IloDisable()` および `Cy_SysClk_IloEnable()` PDL 関数は ILO リセットを実行します。`init_cycfg_platform()`関数は、ModusToolbox™-software が提供する初期化コードを使用すると、デフォルトでバックアップドメインと ILO をリセットします。

バックアップドメインをリセットする手順は、V_{BACKUP} 電源接続によって異なります。

1. VDDD に直接接続された V_{BACKUP}: V_{BACKUP} と V_{DDD} が相互に接続される場合、V_{BACKUP} 電源は電源を入れ直すたびに削除されるため、すべての POR および BOD イベントでバックアップドメインをリセットする必要があります。RTC とバックアップデータレジスタを使用しない場合は、すべてのリセットイベントの後にバックアップドメインをリセットしても安全です。
2. スーパーキャパシタから電力を供給される V_{BACKUP}: V_{BACKUP} がスーパーキャパシタに接続される場合、ファームウェアは、ADC を使用してリセットから抜け出した直後にスーパーキャパシタの電圧を測定する必要があります。スーパーキャパシタの電圧が 1.4 V 未満の場合は、RTC とバックアップデータが信頼できないことを意味し、ファームウェアはバックアップドメインをリセットし、スーパーキャパシタが再充電されるまでの時間を確保する必要があります。
3. バッテリ駆動の V_{BACKUP}: バッテリ電圧は ADC を使用して定期的に監視する必要があります。障害点の近くで放電する場合、ファームウェアはフラッシュに永続的なフラグを設定する必要があります。リセットから起動した後、ファームウェアはフラグをチェックし、RTC とバックアップデータが信頼できないため、バックアップドメインをリセットする必要があることを認識できます。バッテリ電圧が再び良好であることをファームウェアが検出すると、最終的なバックアップドメインのリセットを実行してからフラグをクリアできます。
4. システム提供の電源から電力を供給される V_{BACKUP}: 外部 V_{BACKUP} 電源が無効になっていて、ファームウェアがバックアップドメインをリセットできる場合、システムは PSOC™デバイスファームウェアに通知する方法を提供する必要があります。

3.3.3 外部 PMIC 制御

PSOC™ MCU バックアップドメインは、図 16 に示すパワーマネジメント IC (PMIC) 制御を提供し、PMIC によって提供されるメインの PSOC™ 6 MCU デバイス電源 (V_{DDD}, V_{DDA}, V_{DDIO}) の電源投入と電源切断を支援します。pmic_wakeup_out ピンは、PMIC デバイスのイネーブルピン (アクティブ HIGH) に直接接続するように設計されています。pmic_wakeup_out ピンはロジック HIGH にリセットされ、PMIC およびフル PSOC™ 6 MCU デバイスに最初に電力が供給されるようにします。フルパワーの状態で、ファームウェアは pmic_wakeup_in ピンまたは RTC ウェイクアップソースを構成してから、PMIC、つまりメインデバイスの電源を切れます。pmic_wakeup_out ピンを LOW に駆動すると、PMIC が無効になります。HIGH 論理レベルが pmic_wakeup_in ピンに適用されるか、RTC アラームが発生すると、pmic_wakeup_out が HIGH に設定され、PMIC が再度有効になり、メインデバイスの電源が再投入されます。

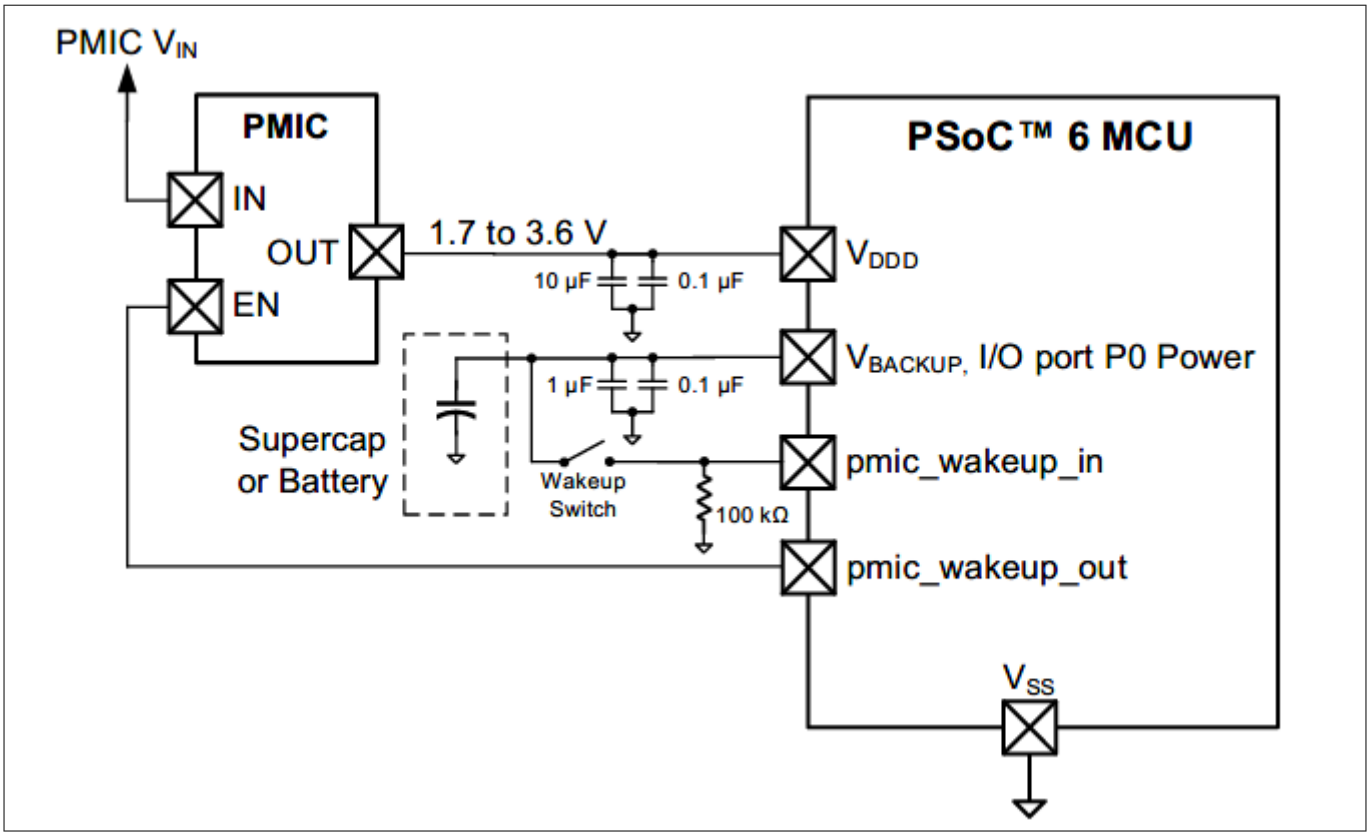


図 16　　PMIC コントローラーのインターフェース

誤ったコードが PMIC コントローラーを無効にしないようにするためには、他のレジスタに書き込む前にロック解除キーが必要です。BACKUP_PMIC_CTL.UNLOCK レジスタに書き込まれた 0x3A の値は、PMIC コントローラーを無効にし、pmic_wakeup_out ピンを GPIO として使用できるようにします。UNLOCK フィールドに他の値が書き込まれると、PMIC コントローラーが有効になります。

特定のレジスタ書き込み制限があるため、制限を適用する SysPm PDL ライブラリを使用する必要があります。直接レジスタアクセスの詳細については、デバイスのアーキテクチャ TRM を参照してください。PMIC の有効化は 3 段階のプロセスであり、他のレジスタが書き込む前にロックを解除する必要があります。

```
Cy_SysPm_PmicUnlock();
Cy_SysPm_PmicEnable();
Cy_SysPm_PmicLock();
```

`Cy_SysPm_PmicAlwaysEnable()`を実行することにより、pmic_wakeup_out ピンを永続的に HIGH に設定できます。この機能を呼び出した後、V_{BACKUP} 電源が切断されるか、バックアップドメインがリセットされるまで、PMIC 出力を無効にできません。

pmic_wakeup_out ピンを LOW に設定するためには、BACKUP_PMIC_CTL.PMIC_EN レジスタのビットフィールドをクリアします。PDL を使用して PMIC を無効にするためには、2 つの PDL 命令が必要です。PDL の使用を無効にした後は、PMIC コントローラーを再ロックしないでください。再ロックすると PMIC が自動的に再度有効になります。`Cy_SysPm_PmicDisable()`を呼び出してから V_{DDD} が減衰するまでの時間は、外部電源の設計によって異なります。

```
Cy_SysPm_PmicUnlock();
Cy_SysPm_PmicDisable();
```

専用 PMIC への直接接続以外にも多くのオプションがあります。この柔軟性により、設計のニーズに合わせて電源スイッチングを最適化できます。一般的な pmic_wakeup_out ピン接続のいくつかは次のとおりです。

1. V_{DDD}, V_{DDA}, V_{DDIO} に複数の電源レールを提供する高度な PMIC デバイス
2. すべての PSOC™ 6 MCU デバイスの電源ピンに供給するイネーブルピンを備えたシンプルな LDO またはスイッチングレギュレータ
3. 既存の安定化電源の制御またはバッテリへの直接接続を可能にする低リーク N-FET または同等のスイッチのゲート
4. システム電源を管理する外部 MCU またはコントローラーへのデジタル信号

3.3.4 ウェイクアップソース

pmic_wakeup_in ピンは、高ロジックレベルを生成できる任意の回路を使用できるため、PSOC™ 6 MCU をウェイクアップおよびリパワリングするための最も柔軟で最小電流の方法です。最も一般的な方法の 1 つは、ピンをプルダウン抵抗に接続し、ユーザーの「電源」ボタンを HIGH に接続することです。複数のウェイクアップソースが必要な場合は、有線の OR 回路でウェイクピンをトリガできます。デバイスは、オプションで、電源投入後に追加のピンを使用してウェイクアップのソースを照会できます。図 17 に、製品の電源ボタンの押下または USB ケーブルの挿入を検出するための有線の OR ウェイクアップを示します。

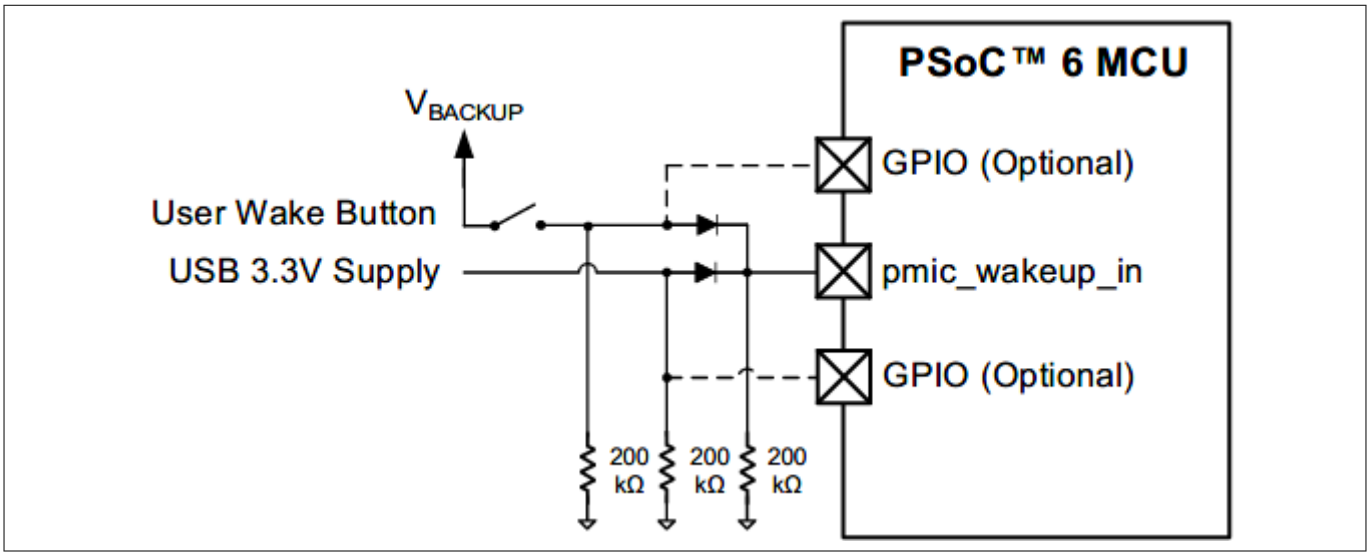


図 17 複数の PMIC ウェイクアップ入力

pmic_wakeup_in 信号と hibernate_wakeup[1]信号は、Port0 で同じ物理ピンを共有するため、システムの休止状態と PMIC 制御の両方を設計で使用する場合は制限があります。hibernate_wakeup[1]ピンは、レジスタ構成によって設定された LOW または HIGH の論理レベルのいずれかでデバイスをウェイクアップできます。pmic_wakeup_in ピンは、高い論理レベルでのみアクティブです。同じデザインで両方のピン信号を使用する場合は、HIGH 論理レベルを使用して休止状態からウェイクアップし、PMIC をウェイクアップしてください。別のオプションは、2 番目の hibernate_wakeup[0]ピンを備えたデバイスで使用でき、バックアップドメインウェイクアップ信号を 2 番目のピンの休止状態ウェイクアップから分離できます。

WDT はバックアップドメインで使用できないため、必要に応じて、RTC を使用してデバイス全体を定期的にウェイクアップする必要があります。RTC を使用するためには、WCO をインストールして、コンポーネント数と消費電流を増やす必要があります。RTC を使用したウェイクアップは、2 つの個別のアラームを使用して特定のアラーム時間にトリガすることも、日, 時間, 分, または秒に基づく定期的な時間のロールオーバーごとにもトリガできます。RTC は、時計の水晶発振器や 50/60 Hz の主電源などの外部ソースからの外部クロック入力も可能にします。正確な時間を維持するために、キャリブレーションレジスタは WCO 周波数の+/- 1 ppm トリム分解能を可能にします。RTC には特定のデータアクセス制限があります。そのため、それらを適切に処理する Cy_RTC PDL ライブラリを使用する必要があります。RTC への直接レジスタアクセスの詳細については、デバイスアーキテクチャ TRM を参照してください。

3.3.5 バックアップデータレジスタ

バックアップドメインには、16 個の 32 ビットレジスタ BACKUP_BREG[15:0] が含まれており、V_{BACKUP} 電源が有効である限り、その内容を保持します。各レジスタは、4 バイトのデータを合計 64 バイト保持します。これらのレジスタは、デバイス全体の電源を切ったときに重要なシステム情報とフラグを保持するために使用されます。また、システムハイバネートモード中に使用して、デバイスのウェイクアップおよびリセット後にデータを読み出せます。

4 他の節電技術

4.1 PSOC™ 6 MCU 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る場合があります。PSOC™ 6 MCU は、ファームウェアでオンおよびオフできる GPIO ピンで電源を供給することにより、それらを介して電源を制御するために使用できます。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。GPIO が直接供給できる電流よりも高い電流が必要な場合は、GPIO によって制御される外部電源デバイスを使用してください。

図 18 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。

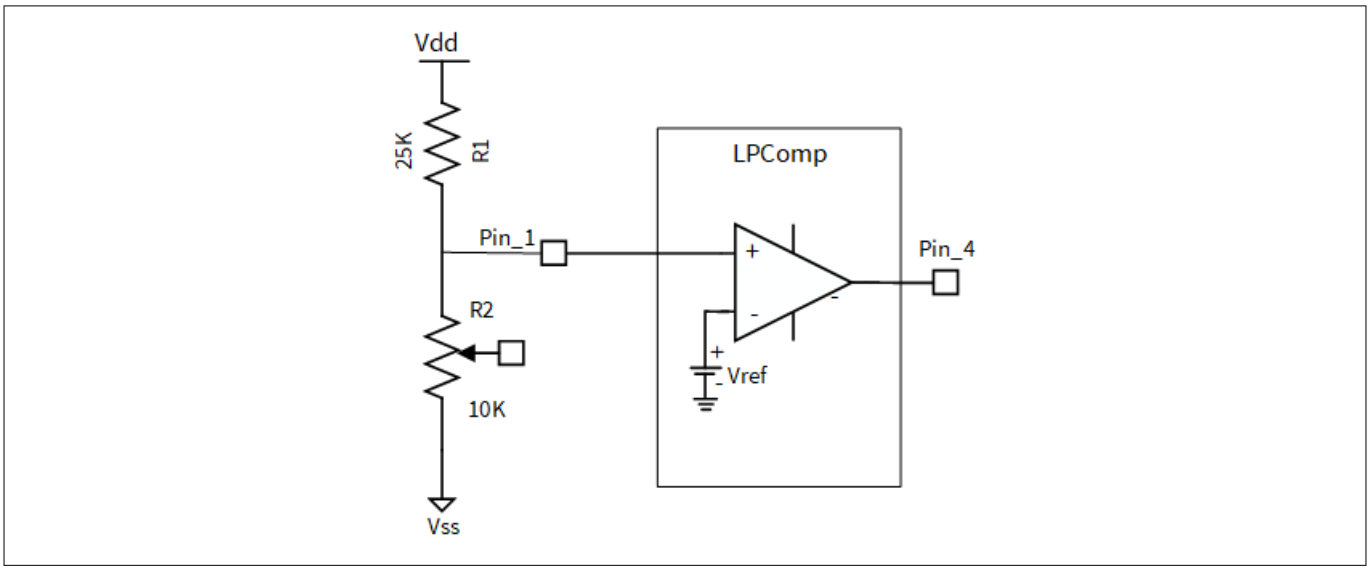


図 18 標準的な LPComp アプリケーション

LPComp は使用しない時にオフにできますが、抵抗とポテンショメータを通る電流経路が残るため、外付け部品は電力を消費します。PSOC™ 6 MCU での簡単な解決策は、図 19 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、ピンを開放にできます。これにより、2 個の抵抗間の電圧差を 0 V 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。この節電機能を実装するためには、1 本のピンおよび数行のコードのみが必要です。

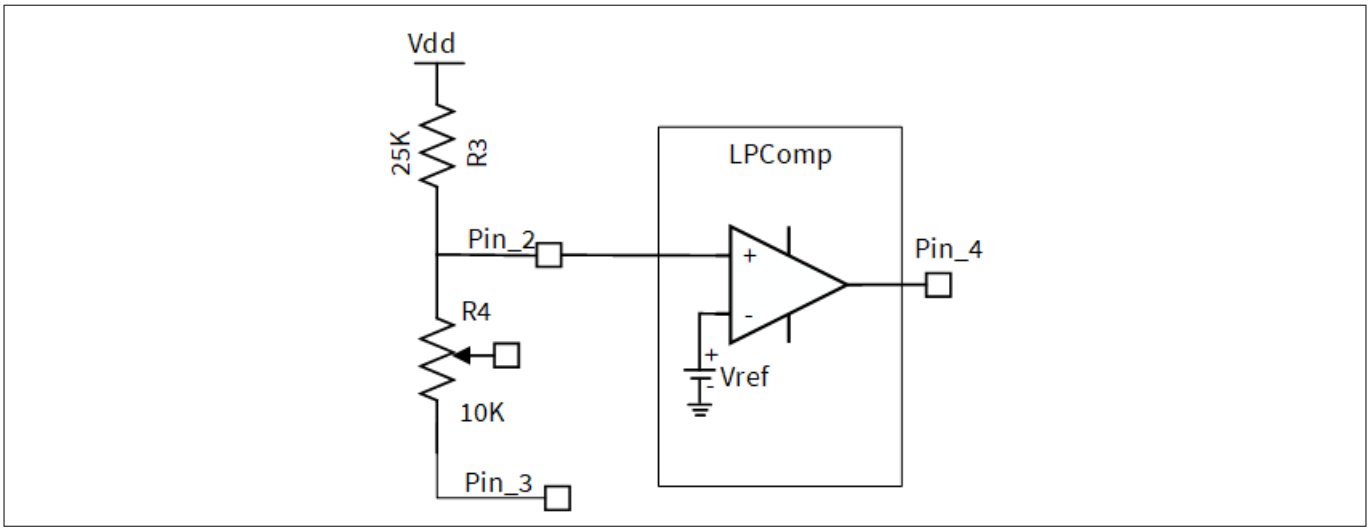


図 19　　グランド スイッチとして GPIO を使用

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。節約される電力は、無効化されたブロックによって異なります。

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 エンジンをシステム LP モードと ULP 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 のアクティブモードと CPU 停止電力モードとの差、CPU がより遅い周波数でクロックされても同じ作業を行える場合はより低い CPU アクティブ電流との差です。

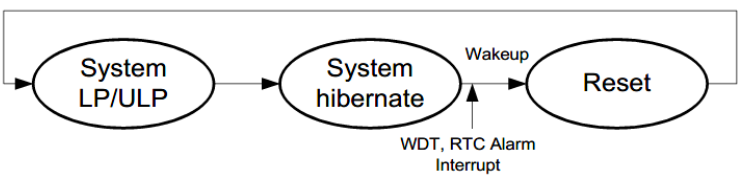
4.4　周期的ウェイクアップ タイマー

CPU スリープモードからの定期的なウェイクアップは、消費電力を削減する最も一般的な方法です。平均消費電力は、CPU アクティブ期間の消費電力と CPU スリープ期間の消費電力の比率によって決まります。最良の結果を得るためには、スリープ期間をできるだけ長くし、アクティブ期間をできるだけ短くする必要があります。CY8C61x6 データシートによる CPU スリープモードの省電力の例は、CM4 アクティブ = 1.7 mA (SIDF5) および CM4 スリープ= 0.76 mA (SIDS7) です。

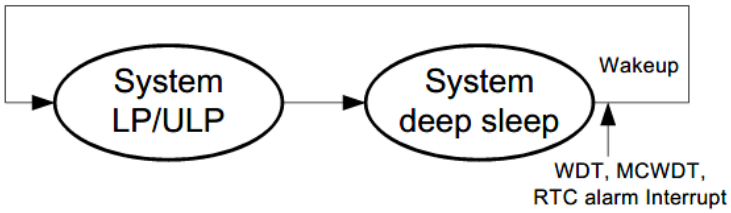
WDT およびマルチカウンター WDT (MCWDT) は、システムディープスリープモードおよびシステム休止モードでの効果的な定期的なウェイクアップソースです。アプリケーションでより長い、またはより正確なウェイクアップ期間が必要な場合は、RTC アラームが適切な定期的なウェイクアップソースになります。ウェイクアップに RTC 周期タイマーを使用したサンプルコードは、付録 D.3 を参照してください。CY8C61x6 データシートによるシステムディープスリープモードの省電力の例は、CM4 アクティブ = 1.7 mA (SIDF5) およびシステムディープスリープ = 7 µA (SIDDS1) です。

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。

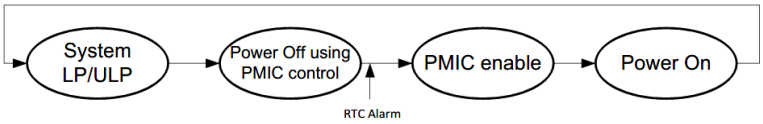
- システム ハイバネート



- システム ディープスリープ



- 外部 PMIC による電源オフ



4.5 CPU の無効化

CM4 CPU がアプリケーションで使用されない場合は、CM0+ファームウェアから `Cy_SysDisableCM4()`関数を呼び出して無効にしてください。また、Clk_Fast 分周器を 256 に設定しても、このクロックからのファンアウトリークを最小限に抑えられます。

CM0+ CPU がアプリケーションで使用されない場合は、システム ディープスリープ電力モードにし、Clk_Slow 分周器を 256 に設定して、このクロックからのファンアウトリークを最小限に抑えてください。

***注**: Clk_Slow は、DMA エンジンの実行速度にも影響します。*

4.6 CPU 間のタスクの分割

特定のタスクを実行するために、CM4 または CM0+を選択するいくつかの理由があります。例を示します。

1. 浮動小数点または DSP 操作を必要とするタスクは、CM4 でより効率的に実行されます。
2. 最大クロック速度 (150 MHz) を必要とするタイムクリティカルなタスクは、CM4 でのみ実行する必要があります。
3. 他のすべてのタスクから独立した単一のタスクで確定的なタイミングが必要な場合、タイミングを単純化するために、そのタスクを独自の CPU に移動すると多くの場合有益です。通常、この目的には CM0+ CPU が使用されます。
4. クロックで説明されるようにシステムのアイドル状態の時間を最大化する場合、CM0+と CM4 の間にタスクを分散すると、タスクの同時処理により低電力のアイドル状態に早く入れます。
5. 低消費電力状態で費やされる時間を制限するリアルタイムハードウェアインタラクションのためにシングル CPU 処理を長期間にわたって分散させる必要がある場合、CM0+は同じアクティブな時間間隔で CM4 よりも低電力になることがよくあります。
6. 1 つの CPU が提供できる能力を超える処理能力が必要な場合、タスクを両方のコアに分散する必要があります。各コアには独自のバスコントローラーがあり、個別のメモリブロックにアクセスできるため、バスまたはメモリの競合による待機状態が最小限になります。競合によるパフォーマンスの低下は通常 2%未満ですが、2 番目のコアからのパフォーマンスの増加は 50%を超えます (CM0+を CM4 に追加すると仮定) 。
7. アクティブモードのままで、単一のコアで必要な処理能力を実現できるアプリケーションでは、ほとんどの場合、必要なタスクをコア間で共有しても電力上の利点はありません。

4.7 クロック

場合によっては、CPU クロックを高速で実行すると、平均消費電流が少なくなることがあります。例えば、センサーから 1 秒ごとに読み出し値を取得し、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、CPU スリープまたはシステム 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、CPU アクティブ モードで費やされる時間が長くなるため、平均消費電流が高くなる可能性があります。図 20 に、システム クロックが 8 MHz に設定されたこの例での消費電流を示します。

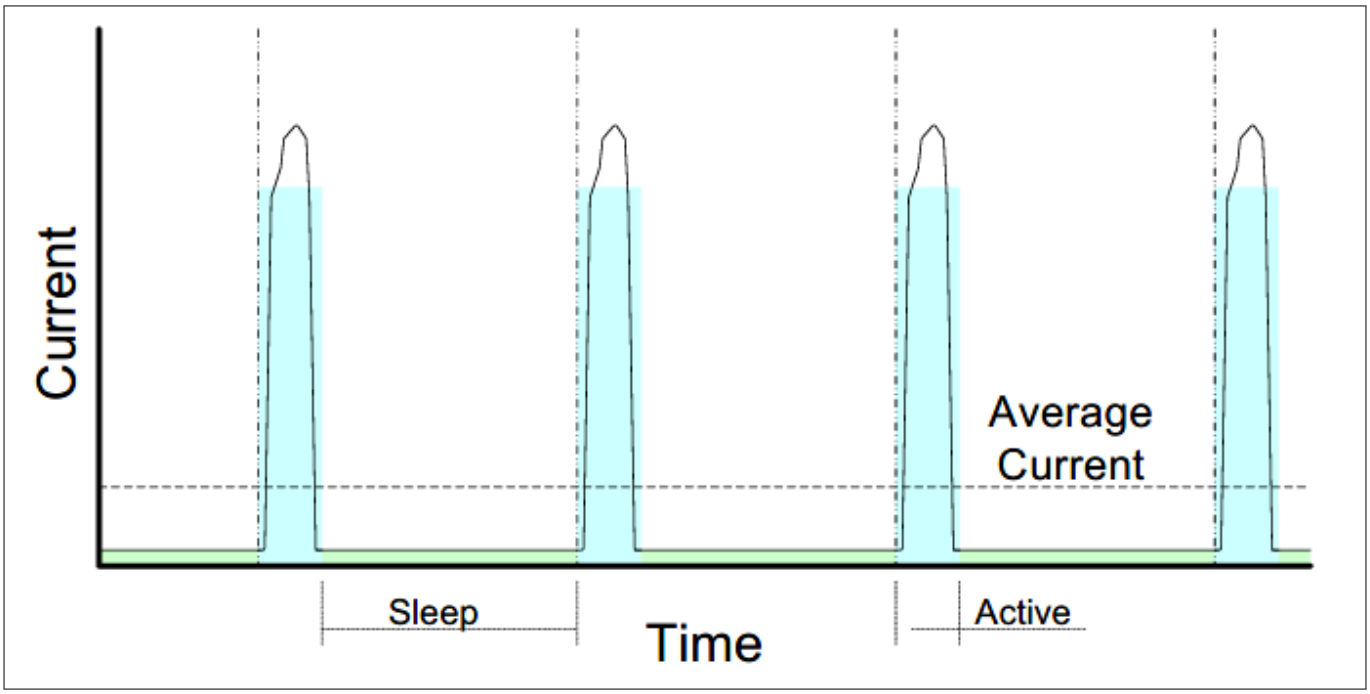


図 20 8 MHz クロックを使った電流分布の例

PSOC™デバイスが起動しているときに実行されるタスクまたは計算によっては、CPU クロックをより速く実行することにより、それらをより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 CPU アクティブモードになり、CPU スリープまたはシステムディープスリープが長くなるため、平均消費電流を削減できます。図 21 は、タスクに分割された CPU アクティブモードのタイミングを表します。

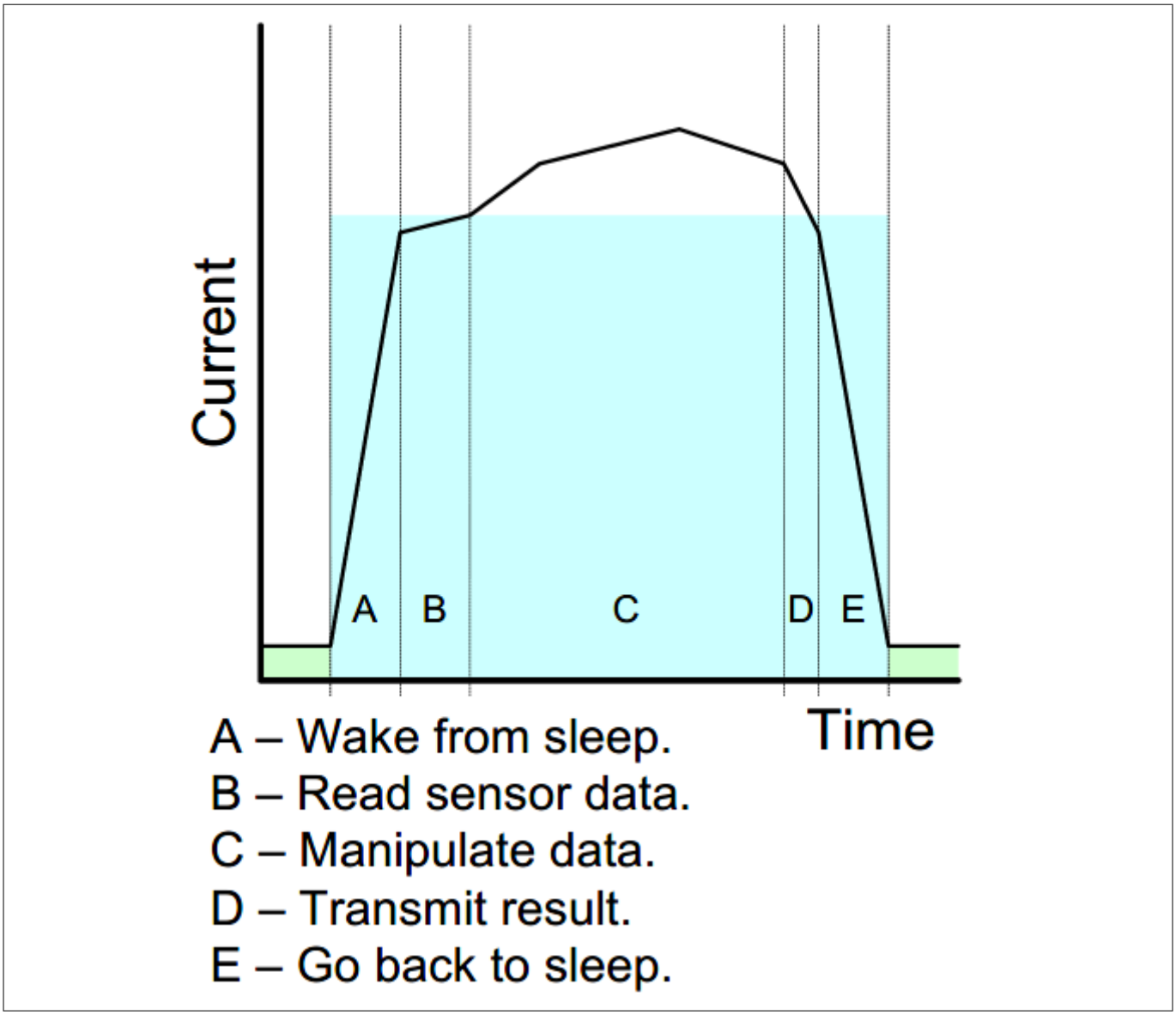


図 21　　8 MHz での CPU アクティブ モードのタスク解析

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カテゴリーに属します。データ処理などの他のタスクは、CPU がより高い周波数で動作する場合、必要な時間が短くなります。

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 22 に示すように、最適速度が 48 MHz であると想定します。48 MHz クロックでは、アクティブモードで費やされる時間は、8 MHz クロックで費やされる時間の約半分です。図 22 は、クロックが高速になるとピーク電流消費が大きくなることを示していますが、CPU スリープまたはシステムディープスリープに費やされる時間が長いため、全体の平均が低くなります。PSOC™ 6 MCU デバイスでは、通常 CPU をできる限り高速で実行して処理を完了し、低消費電力モードでより多くの時間を費やすことを推奨します。

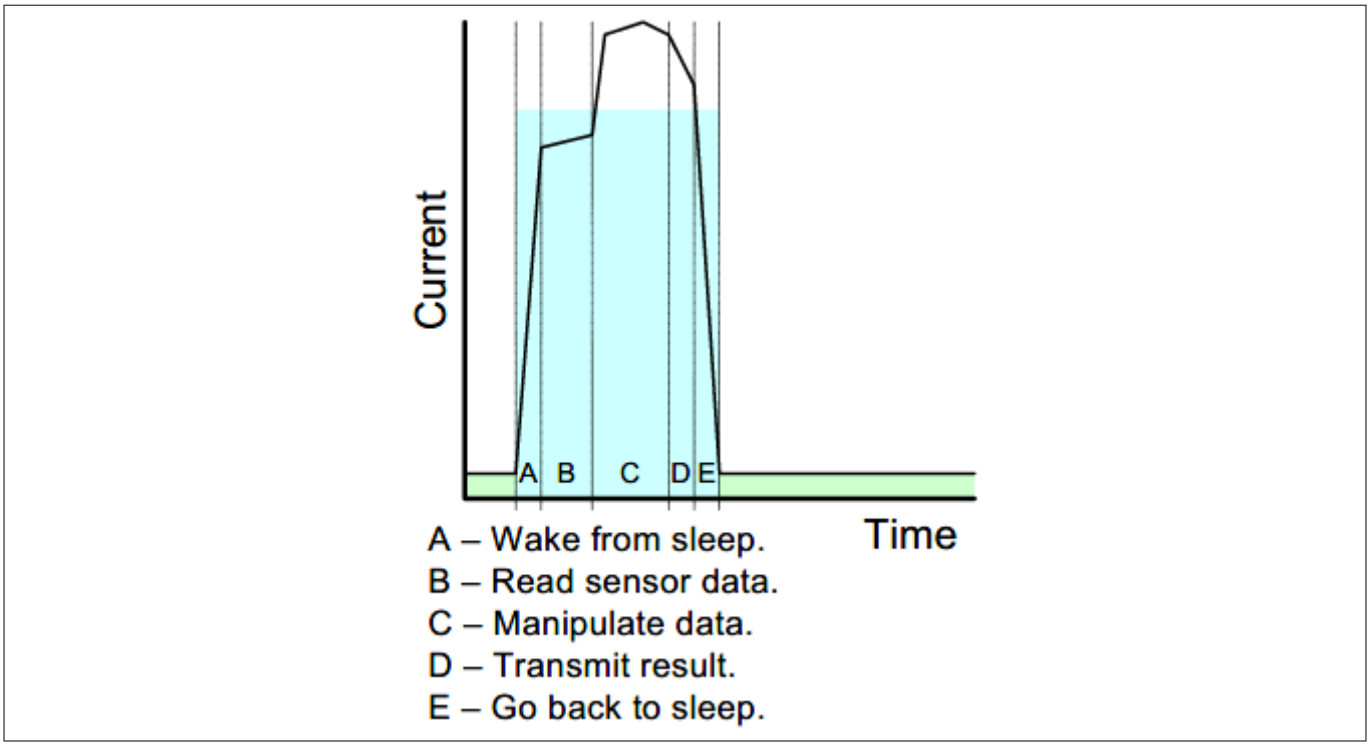


図 22　　48 MHz でのアクティブ モードのタスク解析

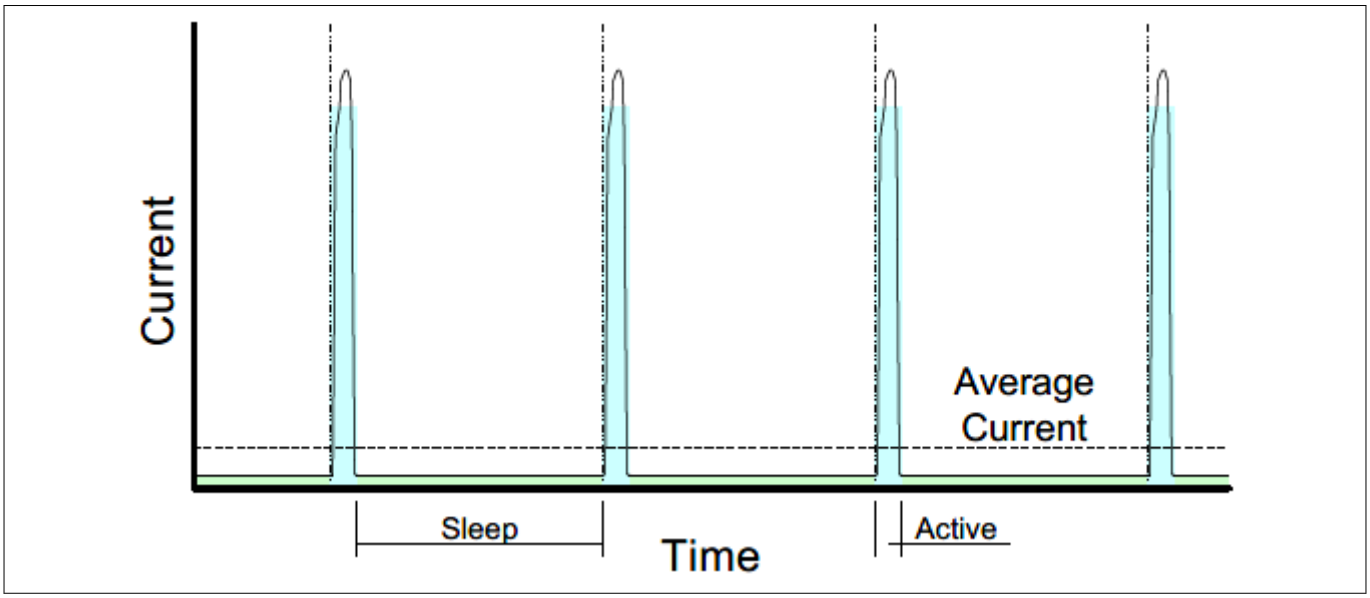


図 23　　48 MHz クロックを使った電流分布の例

4.8　GPIO

PSOC™デバイスが低消費電力モードにある間、GPIO は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場合、消費電力の無駄になってしまう場合があります。この手法の特定の電力節約は、特定の GPIO ピンに接続された回路に依存します。

設計を分析して、低消費電力動作での GPIO に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」に保持すると電流が最小になる場合、cyhal_gpio_write()を使用してそのデジタルレベルを一致させてください。

```
/* Set MYPIN_0_NUM to ‘0’ for low power using HAL APIs.*/
cyhal_gpio_write(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、すべての未使用 GPIO をアナログ High-Z に設定してください。High-Z 駆動モードでは、GPIO ピンの電流が最小になります。ピンの駆動モードは、cyhal_gpio_configure() HAL 関数を使って設定できます。

```
/* Set MYPIN_0_NUM to Analog HI-Z for low power using HAL APIs. */
cyhal_gpio_configure(P0_0, CYHAL_GPIO_DIR_INPUT, CYHAL_GPIO_DRIVE_NONE);
```

PSOC™デバイスの柔軟性により、電流漏れを防止するように GPIO 駆動モードを簡単に管理できます。システム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自動的に「凍結」されます。ウェイクアップリセット後の GPIO 出力の遷移を回避し、実行時に状態を変更できるようにするために、「凍結解除」する前に既知の状態に再構成する必要があります。GPIO ピンを設定し、出力駆動状態を設定した後、Cy_SysPm_IoUnfreeze()を呼び出してください。ハイバネート状態と I/O 制御のサンプル コードについては、付録 D.2 を参照してください。

4.9 SRAM

PSOC™ 6 MCU デバイスは、個々の SRAM バンクまたはバンク内のページの電源をオフにします。バンク内のページサイズは、デバイスデータシートに詳述されているように、特定のデバイスとバンクに依存します。特定のデバイスには、1 つ以上の SRAM バンクがあります。ほとんどのデバイスには、有効な SRAM の量をきめ細かく制御するために、ページサイズが小さい (通常 32 KB) バンクが 1 つあります。未使用のページは、CPUSS 電力制御レジスタに書き込むことで無効にできます。この手法は、CY8C61x6 データシートに記載されているように、64K SRAM = 7 µA (SIDDS1) を保持しながら 256K SRAM = 9 µA (SIDDS2) を保持するシステムディープスリープモードでもっとも役立ちます。

```
/* Power off all except the first two pages of RAM0 */
for (uint32_t i = 2; i < NUM_RAM_PAGES; i++)
   {
      CPUSS->RAM0_PWR_MACRO_CTL[i] = 0x05FA0000;
   }
/* Additional SRAM banks */
CPUSS->RAM1_PWR_CTL = 0x05FA0000;
CPUSS->RAM2_PWR_CTL = 0x05FA0000;
```

すべての電力制御レジスタについては、デバイスの Register TRM を参照してください。

高度なユースケースとして、プロジェクトのリンカスクリプトの変更を SRAM のパワーダウンとともに使用して、必要な SRAM の合計量を最適化するか、ダイナミック SRAM の電源供給をサポートするカスタムデータ配置を許可できます。

4.10 TCPWM

カウンタ, タイマー, 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、チャネルを可能なかぎり低くするクロックを設定する必要があります。たとえば、タイマーで 1 秒の割込みを生成する必要がある場合

は、1,000,000 カウントに等しい周期の 1 MHz のクロック周波数よりも、1,000 カウントに等しい周期の 1 kHz のクロック周波数を使用する方が適切です。TCPWM クロックの削減による電力節約は、クロック周波数に基づいてほぼ線形です。CY8C61x6 データシートによると TCPWM 動作電流は、100 MHz = 540 µA (SID.TCPWM.2B) であり、8 MHz = 70 µA (SID.TCPWM.1) です。

PWM を使用して LED を調光する場合も同じ考え方が当てはまります。LED が点滅していることを人間の目で認識させない、できるかぎり最小のクロック周波数を使用してください。60 Hz は、ほとんどのアプリケーションに適した周波数です。図 24 に、TCPWM ブロックのクロック設定の比較を示します。

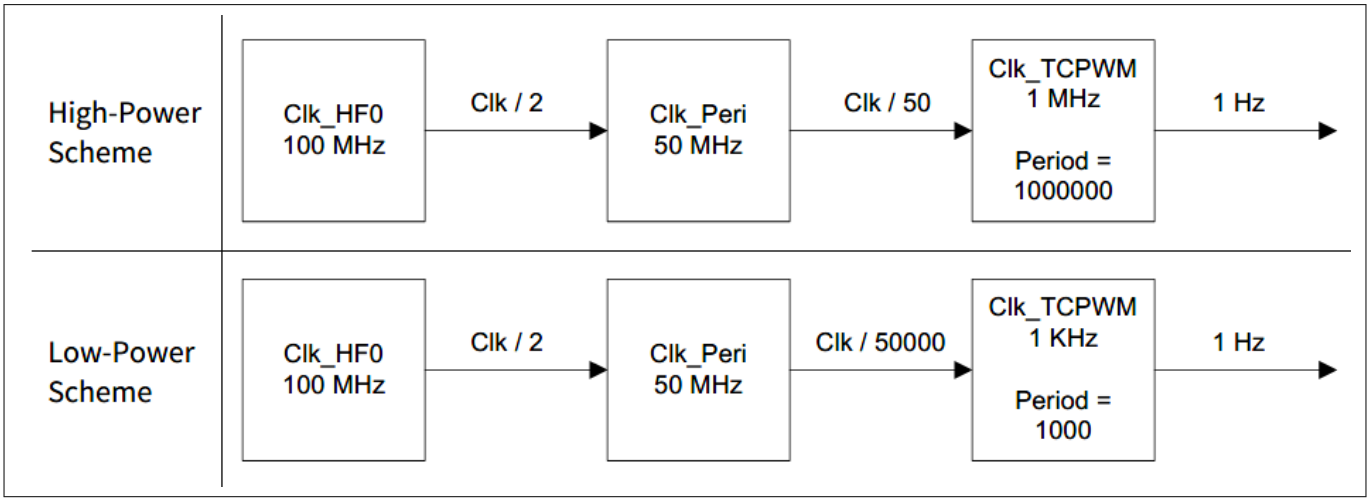


図 24　　TCPWM クロック設定の比較

TCPWM ブロックには、クロック プリスケーラ機能があります。消費電力を最小にするためには、TCPWM クロック プリスケーラを使用する前に、まず周辺クロック分周器を最大化します。

TCPWM ブロックがピン接続を使用する場合、ブロックが無効のときにピンを Analog High-Z に設定してください。PSOC™デバイスがデジタル システム インターコネクト (DSI) をサポートしている場合、目的のピン割当てに直接接続する TCPWM チャネルを選択して、DSI の使用を避けてください。PSOC™デバイスが UDB (universal digital block) をサポートする場合、UDB ベースの TCPWM の代わりにすべての固定機能 TCPWM チャネルを最初に使用してください。

4.11　　SCB

データを送受信する際にブロック機能を使用しないでください。割込みベースのイベントまたは RTOS を使用して、CPU を他のタスクに譲りながらデータを転送してください。この戦略の背後にある考え方は、送信の状態をポーリングするのではなく、CPU をスリープ状態に長く保つことです。図 25 に、ブロッキング機能と非ブロッキング機能の例を示します。

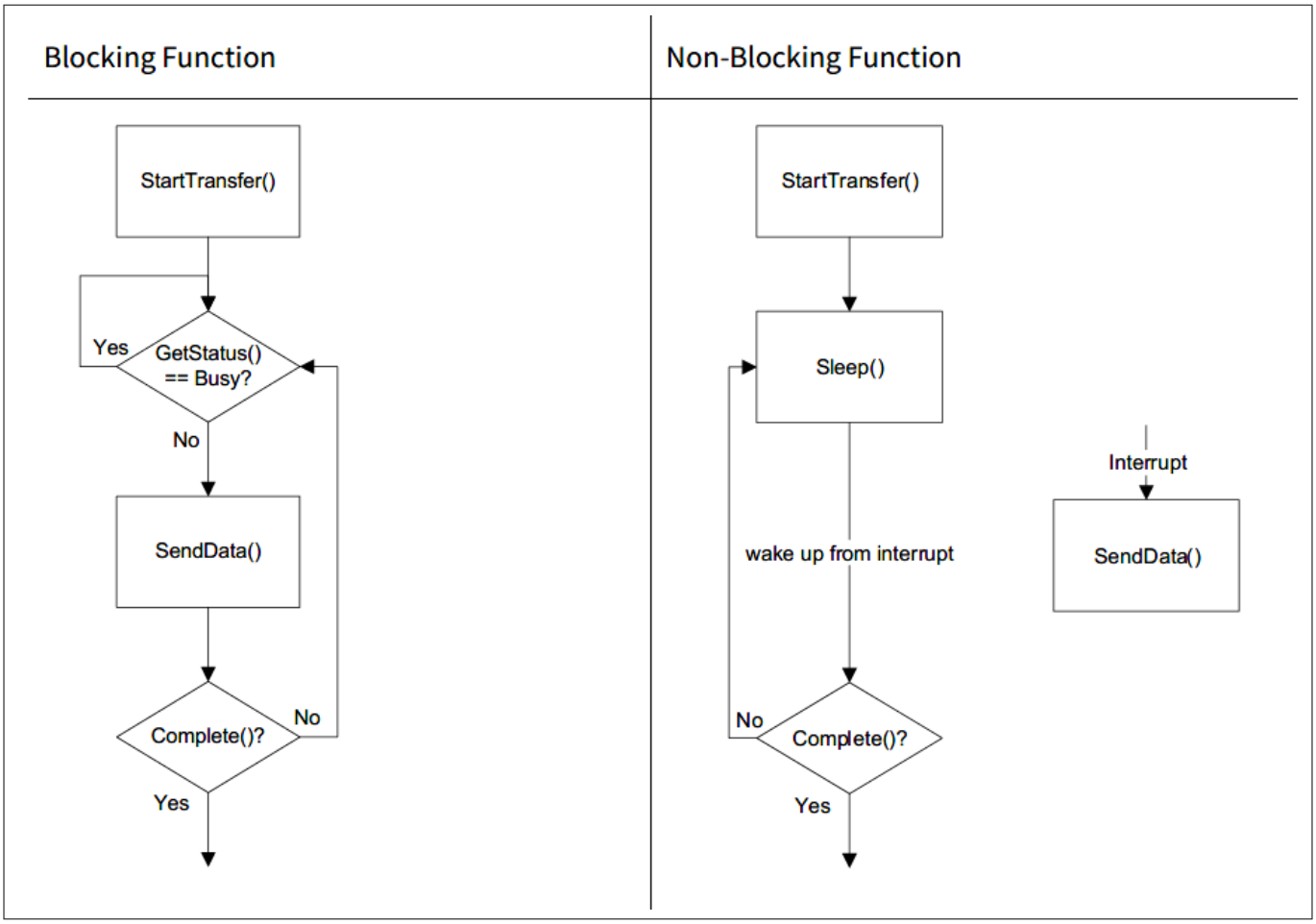


図 25　　ブロッキング機能と非ブロッキング機能の例

SCB 割込みを使用してファームウェアの FIFO にアクセスする代わりに、FIFO レベルで制御される DMA を使用して、アプリケーションで必要な CPU サイクルの量を減らします。そのため、CPU はスリープ状態を長く維持したり、他のタスクを実行したりできます。

SCB ブロックがピン接続を使用する場合、ブロックが無効のときにピンを Analog High-Z に設定してください。PSOC™デバイスが DSI をサポートしている場合、目的のピン割当てに直接接続する SCB チャネルを選択することにより、DSI の使用を避けます。PSOC™デバイスが UDB をサポートしている場合、UDB ベースの通信インターフェースを使用する前に、すべての固定機能 SCB チャネルを使用してください。

システムがサポートできる最低のデータレートで動作することも、SCB の電力がクロック周波数に比例して増加するため、電力の削減に役立ちます。例えば、CY8C61x6 データシートに示されているように、100 kbps = 30 µA (SID149) で実行しながら 1 Mbps = 180 µA (SID152) で I2C を実行します。

4.12　　オーディオサブシステム

I2S および PDM/PCM ブロックは、通常 PLL から生成された高周波クロックによって供給されます。高精度のフレームレートの場合、ECO は PLL をソースとします。ECO の周波数を選択するときは、目的のオーディオサンプルレートを検討してください。

- すべての標準オーディオサンプルレート (8/16/22.05/32/44.1/48 kHz) のサポートが必要な場合は、使用できる最小 ECO 周波数は 17.2032 MHz です。
- サンプルレート 8/16/48 kHz のみが必要な場合は、PLL を 12.288 MHz に設定してください。
- サンプルレート 22.05/44.1 Hz のみが必要な場合は、PLL を 22.5792 MHz に設定してください。

オーディオサブシステムが使用されていない場合は、I2S または PDM/PCM ブロックを無効にします。また、高電流消費の主な理由である PLL と ECO も無効にする必要があります。

フレームレートの精度がアプリケーションに対し重要でない場合は、ECO や PLL を使用しないでください。たとえば、アプリケーションがサウンド検出器を実装し、より高いエラーを許容できる場合、FLL をオーディオサブシステムのソースとして使用してください。音が検出され、システムを完全に動作させる必要がある場合は、PLL と ECO を有効にしてより高い精度でサンプリングしてください。

オーディオサブシステムの割込みを使用してファームウェアの FIFO にアクセスする代わりに、FIFO レベルトリガによって制御される DMA を使用して、アプリケーションで必要な CPU サイクル数を減らし、CPU をスリープ状態に長くしたり、他のタスクを実行できるようにしてください。

4.13 USB

USB バスにアクティビティがない場合、USB ブロックを一時停止して、消費電流を減らせます。サンプルコード CE223305 – PSOC™ 6 MCU: USB suspend and resume を参照して、低消費電力状態に出入りするデバイスをセットアップする方法を参照してください。

4.14 低消費電力コンパレータ

低消費電力コンパレータでピン接続とシステムのディープスリープ動作が必要な場合は、常にコンパレータへの専用ピン接続を選択してください。これにより、コンパレータはシステムのディープスリープで動作し、アクティブな状態を維持するために追加の電流を消費するグローバルアナログマルチプレクサの使用を回避できます。

コンパレータのステータスを確認するためには、2 つの方法があります。`cyhal_comp_read()`機能を呼び出すか、割込みを設定してください。ほとんどの場合、CPU 使用率を削減する割込み方式を使用することを推奨します。

コンパレータは、速度と電力に影響する 3 つの動作モードをサポートします。

1. **超低消費電力/遅い**: I_{CMP1} = 0.85 µA (SID259) – 応答時間を 20 µs (SID92) に遅くします。このモードは、システムのディープスリープまたはハイバネート状態のときにのみ使用します。
2. **低消費電力/低い**: I_{CMP2} = 10 µA (SID248) – システムのディープスリープまたはハイバネート状態が不要で、1 µs (SID258) の遅い応答時間が許容される場合の優先モード。これにより入力オフセット電圧が 25 mV (SID85A) に増加します。
3. **通常電力/速い**: I_{CMP1} = 150 µA (SID89) - 100 ns (SID91) の最速応答時間、または 10 mV (SID84) の低オフセット電圧が必要な場合に使用します。

コンパレータが消費する電流量を減らす別のオプションは、ヒステリシスを無効にすることです。ヒステリシスは、ModusToolbox™のデバイス コンフィギュレーター ツールで無効にできます。

4.15 SAR ADC

SAR ADC に接続するピンを選択するときは、SAR MUX に接続される専用ポートを最初に選択することを推奨します。そのポートのピンがなくなった後にのみ、他のポートからピンを選択してください。専用ポートを使用するだけで余分な電流を消費するグローバルアナログマルチプレクサを使用する必要はありません。また、入力容量が小さいため、専用ポートを使用して高いサンプルレートを実現できます。

ADC の結果のフルレート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 クロックの数が削減されます。

最大サンプルレートが不要な場合は、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 が常に動作しなくなります。シングルショットモードでは、ADC は、アプリケーションに応じて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。

メインアプリケーションが ADC サンプリング中に処理を実行する必要がない場合、CY8C62x4 ファミリのデバイスは、システムディープスリープパワーモードでの SAR ADC 動作をサポートします。CY8C62x4 ファミリの低電力アナログ機能の詳細については、AN230938 – PSOC™ 6 MCU low-power analog を参照してください。

4.16 電圧 DAC

DAC 電圧出力を所定の値で定期的に変更する必要がある場合、DMA を使用して電圧値を更新してください。これにより、CPU サイクルを使用して DAC レジスタに書き込む必要がなくなります。

DAC 出力電圧を必要とする外部デバイスが定期的にのみ使用される場合、出力が不要なときは DAC を無効にしてください。

システムのディープスリープ中に DAC 出力電圧を供給する必要がある場合、サンプルホールドストラテジを使用して、システムのディープスリープ中に電圧 DAC が無効になっている間、電圧出力を維持してください。詳細は、サンプルコード CE220925 – PSOC™ 6 MCU VDAC sample and hold を参照してください。CY8C61x6 データシートによれば、DAC 動作電流 = 125 µA (SID100D) であり、DAC 電流停止 = 1 µA (SID101D) です。

4.17 オペアンプ

オペアンプは、低, 中, および高の 3 つの異なる動作モードをサポートします。これらの動作モードは、システムの電力モード (システム LP/ULP およびディープスリープ) の影響を受けます。システム ディープスリープでは、2 つの追加モードから選択できます。表 4 に、適切な動作モードの選択方法を示します。

表 4 オペアンプの動作モード条件

仕様	高消費電力動作モード		中消費電力動作モード		低消費電力動作モード	
	LP/ULP	ディープスリープ	LP/ULP	ディープスリープ	LP/ULP	ディープスリープ
最小ゲイン幅	6 MHz	M1: 4 MHz M2: 0.5 MHz	4 MHz	M1: 2 MHz M2: 0.2 MHz	1 MHz	M1: 0.5 MHz M2: 0.1 MHz
最大出力電流 (無負荷時)	1500 µA	M1: 1500 µA M2: 120 µA	600 µA	M1: 600 µA M2: 60 µA	350 µA	M1: 350 µA M2: 15 µA
応答時間	150 ns		400 ns		2000 ns	

M1: モード 1 は、GBW が高く、電流が最大です。

M2: モード 2 は GBW が低く、電流が最小です。オペアンプに接続するピンを選択するときは、オペアンプに接続する専用ピンを選択することが望ましいです。専用ピンのみを使用することにより、余分な電流を消費するグローバルアナログマルチプレクサを使用する必要がありません。

5 電力測定

5.1 DMM による電力測定

デジタルマルチメーター (DMM) を使用してデバイスの電流を測定する場合、DMM 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。これらの抵抗の範囲は、1 Ω 未満から 10 kΩ 以上です。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、シャント抵抗値を確認することが重要です。つまり、PSOC™デバイスには、供給していると思われるのと同じ電圧が表示されません。メーターのシャント抵抗が 1 Ω 以下の場合、PSOC™ MCU 電流を測定するときに数ミリボルトの降下しか見られませんが、これは無視できます。一部のベンダーが低電流測定に使用しているシャント抵抗が 1 kΩ の場合、1 mA の電流は 1 V の低下になります。

シャント抵抗値が高いと、大電流での低電圧のためにデバイスがリセットされる可能性があります。また、測定電流範囲を変更する際、ご使用の DMM が「ブレークビフォア メーク」を行わないようにしてください。そうしないと、電源が切断されて再投入され、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な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 シャントに依存する代わりに、ほとんどの PSOC™ 6 MCU キットには、シャント抵抗または電流測定ヘッダーの場所が含まれます。特定の電流測定機能を決定するためには、それぞれのキットのユーザーガイドを参照してください。これらは、電圧計を使用してシャントの両端の電圧を測定できる小さな抵抗に交換または接続できます。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。

5.2 消費電力の概算

デバイスのデータシートには、プロジェクト固有のユーザー構成の PSOC™ 6 MCU の消費電力を推定するための情報が記載されます。このプロセスを簡素化するために、広範な内部コンポーネントとモードの一般的な電力要件を含むスプレッドシートが提供されます。スプレッドシートには、*AN230938 – PSOC™ 6 MCU low-power analog* で詳細に説明されている低電力アナログ周辺機能を含む CY8C62x4 デバイスが含まれます。このスプレッドシート (PSOC™ 6 Power Calculator (version 3) - AN219528.xlsx) は、32-bit PSOC™ 6 Arm® Cortex®-M4/M0+ Web ページの Documents セクションにあります。すべてのプロジェクトは異なるため、このスプレッドシートで提供される消費電力の計算は、典型的な値の推定値に過ぎませんが、設計が完了する前にフィードバックを提供するために十分なものでなければいけません。

スプレッドシートにはいくつかのタブがあります。データを入力する前に、「Instructions」タブを必ず読んでください。5 つの「Config」タブを使用すると、実行時にデバイスを移行できるさまざまなハードウェア設定と電力モードを構成できます。「Summary」タブには、各構成の電流と、各構成で費やされた時間に基づいたすべてのモードのピークおよび平均電流が表示されます。「Summary」タブのオプションのバッテリ寿命推定セクションは、平均電流に基づいてバッテリ寿命を推定します。

6 電源保護システム

6.1 ハードウェア制御

6.1.1 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、フラッシュに有効です。BOD 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。

6.1.2 低電圧検出 (LVD)

低電圧検出 (LVD) は BOD に似ていますが、しきい値電圧は設定可能です。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。この割込みにより、アプリケーションは、BOD リセットがトリガされる前に重要なデータを処理できます。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。

6.1.3 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。

7 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。

A 電力モードのまとめ

A.1 電力モードとウェイクアップ ソース

表 5 電力モードとウェイクアップ ソース

システム電力モード	MCU 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
LP	アクティブ	プライマリモードの操作。1.1V コア電圧。すべての周辺機器が利用可能です (プログラム可能)。最大周波数のクロック。	• 外部リセットからのリセット、ブラウンアウト、パワーオンリセットシステム、および休止状態モード • システム ULP モードからの手動レジスタ書き込み • システム LP モードでの CPU スリープまたは CPU ディープスリープからのウェイクアップ • LP モードから入った後のシステムディープスリープからのウェイクアップ	適用できません	該当なし
	スリープ	1.1V コア電圧。スリープモードの 1 つ以上の CPU (実行が停止)。すべての周辺機器が利用可能です (プログラム可能)。最大周波数のクロック。	システム LP モードでは、CPU はディープスリープを無効にして WFI/WFE 命令を実行します。	CPU への割込み	割込み
	ディープスリープ	1.1V コア電圧。ディープスリープモードの 1 つの CPU (実行が停止)。アクティブまたはスリープモードの他の CPU。すべての周辺機器が利用可能です (プログラム可能)。最大周波数のクロック。	システム LP モードでは、CPU はディープスリープを有効にして WFI/WFE 命令を実行します。	CPU への割込み	割込み

(続く)

A 電力モードのまとめ

表 5 (続き) 電力モードとウェイクアップ ソース

システム電力モード	MCU 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
ULP	アクティブ	0.9V コア電圧。すべての周辺機器が利用可能です (プログラム可能)。クロック周波数は制限されます。	システム LP モードからの手動レジスタ書き込み。システム ULP モードでの CPU スリープまたは CPU ディープスリープからのウェイクアップ。ULP モードから入った後のシステムのディープスリープからのウェイクアップ。	適用できません	該当なし
	スリープ	0.9V コア電圧。スリープモードの 1 つ以上の CPU (実行が停止)。すべての周辺機器が利用可能です (プログラム可能)。クロック周波数は制限されます。	システム ULP モードでは、CPU はディープスリープを無効にして WFI/WFE 命令を実行します。	CPU への割込み	割込み
	ディープスリープ	0.9V コア電圧。ディープスリープモードの 1 つの CPU (実行が停止)。アクティブまたはスリープモードの他の CPU。すべての周辺機器が利用可能です (プログラム可能)。クロック周波数は制限されます。	システム ULP モードでは、CPU はディープスリープを有効にして WFI/WFE 命令を実行します。	CPU への割込み	割込み
ディープスリープ	ディープスリープ	すべての高周波クロックと周辺機器はオフになります。低周波数クロック (32 kHz) と低電力アナログおよびデジタル周辺機器は、動作用およびウェイクアップソースとして利用できます。SRAM は保持されます (プログラム可能) 。	CPU ディープスリープモードで同時に両方の CPU	GPIO 割込み、低電力コンパレータ、SCB、CTBm、ウォッチドッグタイマー、および RTC アラーム	割込み
ハイバネート	該当なし	GPIO の状態は凍結されます。低電力コンパレータとバックアップドメインを除き、デバイスのすべての周辺機器とクロックは完全にオフになります。ウェイクアップは、WAKEUP ピン、XRES、低電力コンパレータ (プログラム可能)、および RTC アラーム (プログラム可能) を通じて可能です。ウェイクアップ時にデバイスがリセットされます。	LP または ULP モードからの手動レジスタ書き込み	WAKEUP ピン、低電力コンパレータ、ウォッチドッグタイマー [1)]、および RTC[2)] アラーム。	リセット

1) ウォッチドッグタイマーは、ハイバネートのウェイクアップを生成できます。
2) RTC (WCO とともに) はバックアップドメインの一部であり、デバイスの電力モードに関係なく使用できます。RTC アラームは、あらゆる電力モードからデバイスを起動できます。

B サブシステムの可用性

B.1 異なる電力モードで使用できるリソース

表 6 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 6 異なる電力モードで使用できるリソース

コンポーネント	システム電力モード							
	LP		ULP		ディープスリープ	ハイバネート	XRES	バックアップ付き電源オフ
	CPU アクティブ	CPU スリープ/ディープスリープ	CPU アクティブ	CPU スリープ/ディープスリープ				
コア機能								
CPU	On	スリープ	On	スリープ	データ保持	Off	Off	Off
SRAM	On	On	On	On	データ保持	Off	Off	Off
フラッシュ	読み出し/書き込み	読み出し/書き込み	読み出しのみ	読み出しのみ	Off	Off	Off	Off
高速クロック (IMO, ECO, PLL, FLL)	On	On	On	On	Off	Off	Off	Off
LVD	On	On	On	On	Off	Off	Off	Off
ILO	On	On	On	On	On	On	Off	Off
ペリフェラル								
SMIF	On	On	On	On	データ保持	Off	Off	Off
UDB	On	On	On	On	Off	Off	Off	Off
SAR ADC	On	On	On	On	Off	Off	Off	Off
CTBm	On	On	On	On	On (下位 GBW)[1]	Off	Off	Off
LPCMP	On	On	On	On	On[1]	On[2]	Off	Off
TCPWM	On	On	On	On	Off	Off	Off	Off
CSD	On	On	On	On	データ保持	Off	Off	Off
Bluetooth® LE	On	On	On	On	データ保持	Off	Off	Off
LCD	On	On	On	On	On	Off	Off	Off
SCB	On	On	On	On	データ保持 (I2C/SPI ウェイクアップ可)[3]	Off	Off	Off

(続く)

B サブシステムの可用性

表 6　(続き) 異なる電力モードで使用できるリソース

コンポーネント	システム電力モード							
	LP		ULP		ディープスリープ	ハイバネート	XRES	バックアップ付き電源オフ
	CPU アクティブ	CPU スリープ/ディープスリープ	CPU アクティブ	CPU スリープ/ディープスリープ				
GPIO	On	On	On	On	On	フリーズ	Off	Off
ウォッチドッグタイマー	On	On	On	On	On	On	Off	Off
マルチカウンター WDT	On	On	On	On	On	Off	Off	Off
リセット								
XRES	On	On	On	On	On	On	On	Off
POR	On	On	On	On	On	On	Off	Off
BOD	On	On	On	On	On	Off	Off	Off
ウォッチドッグリセット	On	On	On	On	On	On[4)]	Off	Off
バックアップドメイン								
WCO, RTC, アラーム	On	On	On	On	On	On	On	On

1) システムのディープスリープモードでは、オプションで低電力コンパレータと CTBm を有効にして、ウェイクアップを生成できます。
2) ハイバネート状態モードでは、ウェイクアップを生成するためにオプションで低電力コンパレータを有効にできます。
3) システムのディープスリープ電力モードでは、ディープスリープをサポートする SCB が 1 つしか使用できません。他の SCB は、システムのディープスリープ電力モードでは使用できません。
4) ウォッチドッグ割込みは、ハイバネート状態のウェイクアップを生成できます。詳細については、TRM の「ウォッチドッグタイマー」章を参照してください。

C コールバック関数例

C.1 HAL を使用したコールバック関数の登録

```
cyhal_syspm_callback_data_t callback_data_sleep =
    {
        .callback = syspm_app_callback,
        .states = (cyhal_syspm_callback_state_t)(CYHAL_SYSPM_CB_CPU_SLEEP |
CYHAL_SYSPM_CB_SYSTEM_LOW),
        .ignore_modes = (CYHAL_SYSPM_CHECK_READY | CYHAL_SYSPM_CHECK_FAIL),
        .args = NULL,
        .next = NULL,
    };

    cyhal_syspm_callback_data_t callback_data_hibernate =
    {
        .callback = syspm_app_callback,
        .states = (cyhal_syspm_callback_state_t)(CYHAL_SYSPM_CB_SYSTEM_HIBERNATE |
CYHAL_SYSPM_CB_SYSTEM_LOW),
        .args = NULL,
        .next = NULL,
    };

    cyhal_syspm_register_callback(&callback_data_sleep);
    cyhal_syspm_register_callback(&callback_data_hibernate);
```

C.2 HAL を使用したカスタム コールバック関数の実装

```
bool syspm_app_callback(cyhal_syspm_callback_state_t state, cyhal_syspm_callback_mode_t mode,
void *callback_arg)
{
    (void)callback_arg;
    bool allow_transition = true;

    switch (mode)
    {
        case CYHAL_SYSPM_CHECK_READY:
            /* Check to see if it is OK to change power modes. */
            /* If it is OK the change, make sure nothing will invalidate this and return true.
*/
            /* If it is not OK the change, return false. */
            /* Example: prevent transition if requested mode is HIBERNATE. */
            allow_transition = (state != CYHAL_SYSPM_CB_SYSTEM_HIBERNATE);
            break;
        case CYHAL_SYSPM_CHECK_FAIL:
            /* Undo anything done as part of CHECK_READY since it is not actually chaning. */
            break;
        case CYHAL_SYSPM_BEFORE_TRANSITION:
            /* Do anything necessary to shut things down. */
            break;
        case CYHAL_SYSPM_AFTER_TRANSITION:
            /* Undo anything done as part of CHECK_READY or BEFORE_TRANSITION since it is
coming back up. */
            break;
    }

    return allow_transition;
}
```

D サンプルコード

D.1 CE219881 – PSOC™ 6 MCU の電力モード間の切り替え

このサンプル コードは、システム LP および ULP 電力モードの開始/終了の方法と、CPU を CPU アクティブからスリープまたはディープ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システム LP または ULP モードおよび CPU 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、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 26 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。

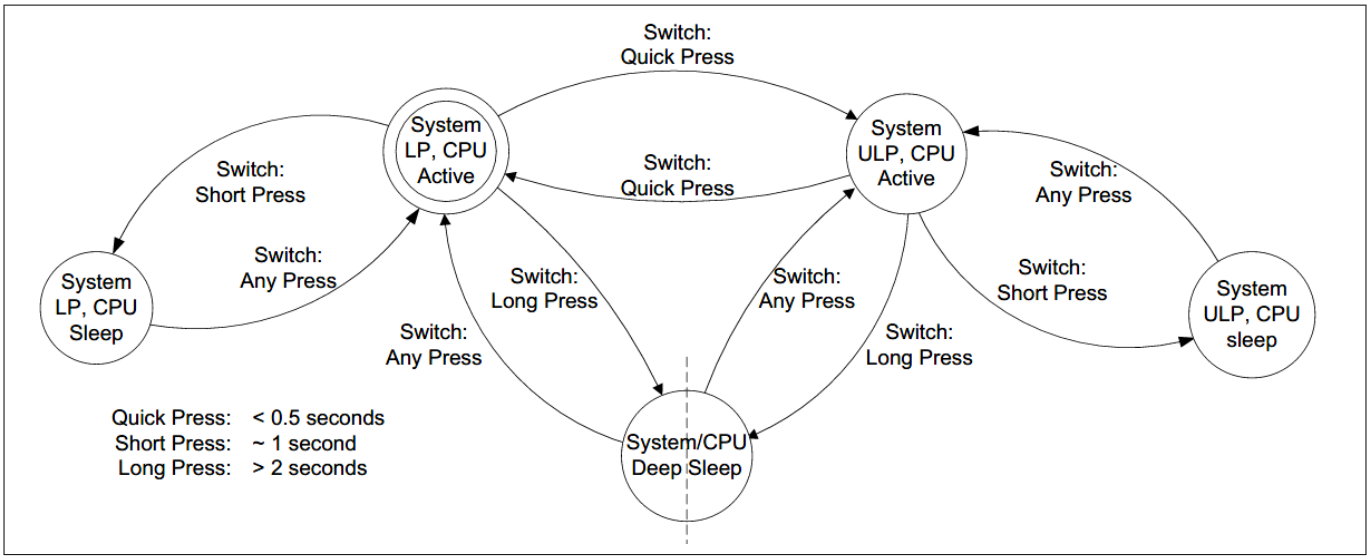


図 26 電力モード ステート マシン

D.2 CE231030 – 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、システムハイバネート電力モード遷移の良い例です。ここでは、システムハイバネートの前後で GPIO を取り扱う方法、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 27 に、このプロジェクトの基本フローを示します。詳細は、CE231030 - PSOC™ 6 MCU: wakeup from hibernate using a low-power comparator を参照してください。

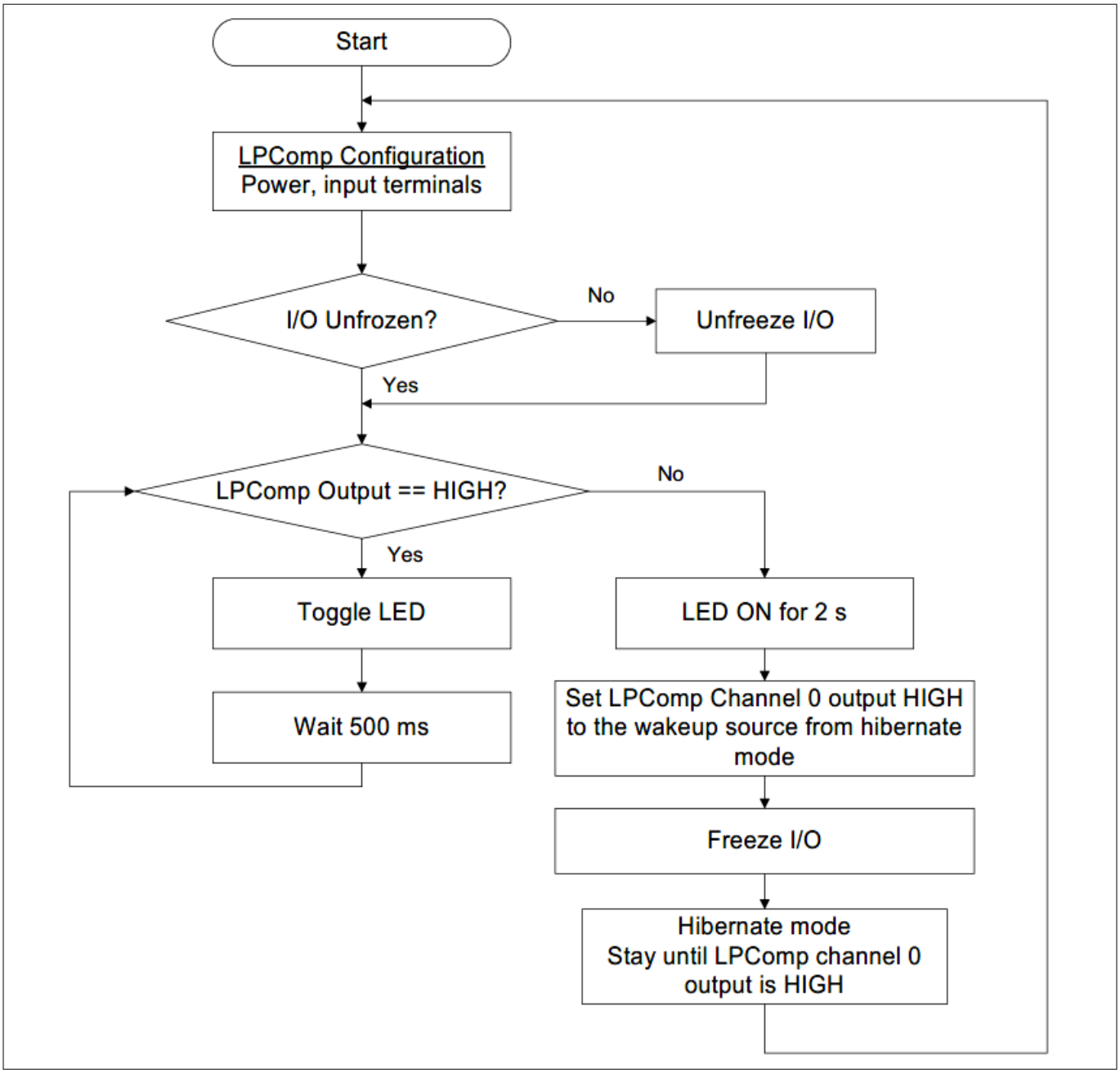


図 27　　LPComp 入力を使用するシステムハイバネート モードからのウェイクアップ

D.3　CE237278 - HAL: RTC alarm periodic wakeup

このサンプルコードでは、ディープスリープ モードとハイバネート モードに入る方法と、RTC を使用して RTC アラームを生成して、ディープスリープ モードとハイバネート モードから MCU を復帰させる方法を示します。

MCU は、ユーザー・ボタンが押されている時間に応じて、ディープスリープまたはハイバネート モードに入ります。RTC アラームはシステムを復帰させ、ウェイクアップ情報を UART 端子に出力します。詳細については、CE237278 - HAL: RTC alarm periodic wakeup を参照してください。

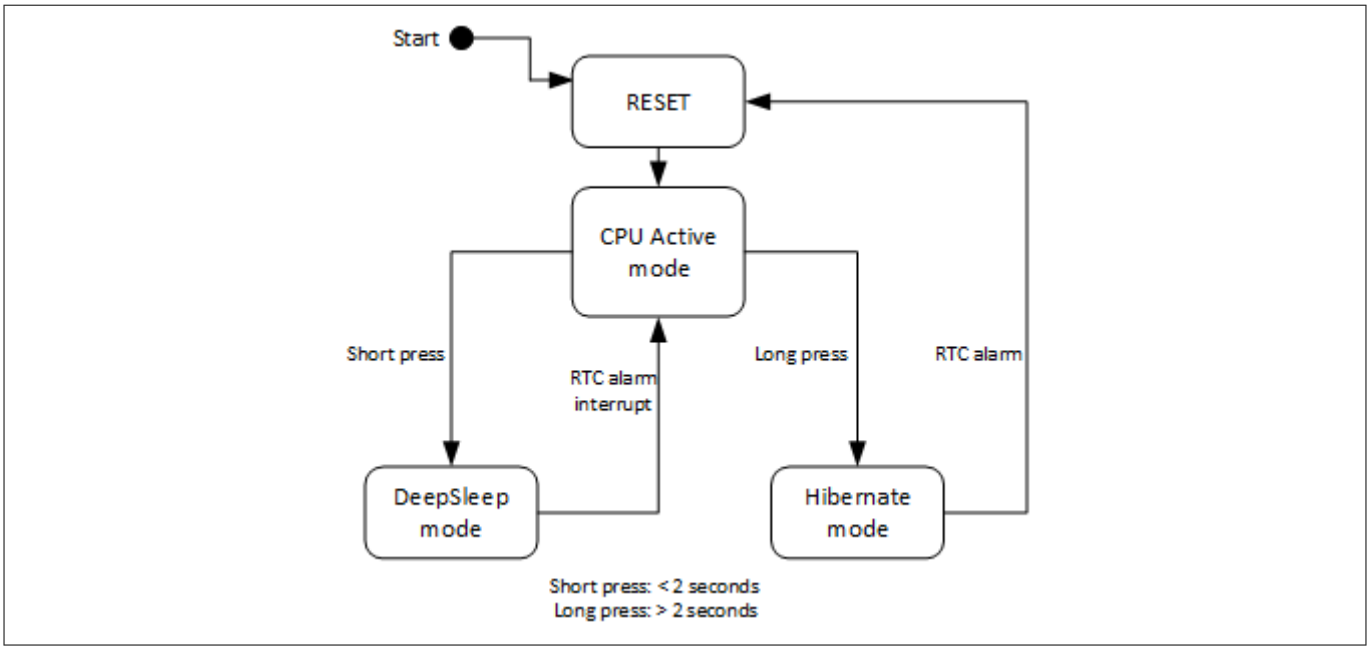


図 28　　アラーム割込みを使用した RTC 周期的ウェイクアップ タイマー

D.4　CE226306 – PSOC™ 6 MCU 電力測定

この例は、PSOC™ 6 MCU データシートのデバイスレベル仕様表で指定されたクロック周波数およびシステムモードを実行するように PSOC™ 6 MCU デバイスを設定する方法を示します。PSOC™ 6 MCU デバイスにアプリケーションを構築してプログラミングした後、PSOC™ 6 MCU デバイスが消費する電流を測定し、それをデータシートで指定された値と比較できます。ファームウェアの`#define` を変更することにより、電源設定を選択できます。

表 7　　設定オプション

設定	オプション	説明

システムモード	SYSTEM_LP
	SYSTEM_ULP
		ファームウェアが入力するシステムモード: システム低消費電力モードまたはシステム超低消費電力モードを定義します。以下の機能が使用されます。
		`Cy_SysPm_SystemEnterLp(); // Enter System Low Power mode`
		`Cy_SysPm_SystemEnterUlp(); // Enter System Ultra-Low Power mode`

コア電圧電源	VCCD_1V1
	VCCD_0V9
		BUCK または LDO のコア電圧電源を定義します– 0.9 V または 1.1 V。
		`Cy_SysPm_SwitchToSimoBuck();`
		`Cy_SysPm_SimoBuckSetVoltage1(CY_SYSPM_SIMO_BUCK_OUT1_VOLTAGE_1_1V);`

(続く)

表 7 (続き) 設定オプション

設定	オプション	説明

キャッシュサポート	RUN_FROM_FLASH
	RUN_FROM_CACHE
		キャッシュを使用するかどうかを定義します。キャッシュが無効の場合、次の手順が使用されます。
		`//Disable CM4 cache`
		`//Disable CM0+ cache`
		`CY_SET_REG32(CYREG_FLASHC_CM0_CA_CTL0,`
		`CY_GET_REG32(CYREG_FLASHC_CM0_CA_CTL0) & CACHE_DISABLE_MASK);`
		Cy_SysLib_SetWaitStates()機能は、キャッシュ サポートに基づいて呼び出されます。無効の場合、特定のシステムモードの最大許容周波数を設定します。キャッシュが有効の場合、CM4 CPU 周波数に基づいて設定します。

CM4 CPU モード	CM4_WHILE_LOOP
	CM4_DHRYSTONE
	CM4_SLEEP
	CM4_DEEP_SLEEP
		CM4 CPU が実行するものを定義します: while (1) ループ、Dhrystone アルゴリズム、CPU スリープに行く、または CPU ディープスリープに行く。以下の機能が使用されます。
		`while (1); // Runs a forever while loop dhrystone(); // Runs the Dhrystone instructions`
		`Cy_SysPm_CpuEnterSleep(CY_SYSPM_WAIT_FOR_INTERRUPT); // Sleep`
		`Cy_SysPm_CpuEnterDeepSleep(CY_SYSPM_WAIT_FOR_INTERRUPT); // Deep-Sleep`

CM4 CPU 周波数 [CM4_FREQ_MHZ]	FREQ_8_MHZ
	FREQ_25_MHZ
	FREQ_50_MHZ
	FREQ_100_MHZ
		CM4 CPU クロックにリンクされている HFCLK0 のクロック周波数を定義します。以下の機能が使用されます。
		`Cy_SysClk_FllConfigure(FREQ_8_MHZ, CM4_FREQ_MHZ, CY_SYSCLK_FLLPLL_OUTPUT_AUTO); // Configure the FLL`
		`Cy_SysClk_FllEnable(TIMEOUT_LOCK); // Enable the FLL`

CM0+ CPU モード	CM0P_WHILE_LOOP CM0P_DHRYSTONE CM0P_SLEEP CM0P_DEEP_SLEEP CM0P_HIBERNATE	CM0+ CPU の実行内容を定義します: While (1) ループ、Dhrystone アルゴリズム、CPU スリープへ行く、CPU ディープスリープまたはハイバネートへ行く。 CM4 CPU モードと同じ機能が使用されます。 `Cy_SysPm_SystemEnterHibernate(); // Go to Hibernate`

CM0+ CPU 周波数 [CM0P_FREQ_MHZ]	FREQ_8_MHZ
	FREQ_25_MHZ
	FREQ_50_MHZ
	FREQ_100_MHZ
		CM0+ CPU クロックのクロック周波数を定義します。次の機能が使用されます。
		`/* Set the PERI Clock Divider */`
		`Cy_SysClk_ClkPeriSetDivider((CM4_FREQ_MHZ/CM0P_FREQ_MHZ)-1);`

(続く)

D サンプルコード

表 7　　(続き) 設定オプション

設定	オプション	説明

クロックソース	USE_IMO
	USE_FLL
		HCLK0 – IMO または FLL のクロックソースを定義します。IMO が使用されている場合、FLL はバイパスされます。FLL を使用する場合、IMO は引き続き FLL のソースとして使用されます。
		以下の機能が使用されます。
		`Cy_SysClk_ClkPathSetSource(0UL,CY_SYSCLK_CLKPATH_IN_IMO);`
		`Cy_SysClk_ClkHfSetSource(0UL,CY_SYSCLK_CLKHF_IN_CLKPATH0);`

最小レギュレータ電流モード	MIN_CURRENT	ファームウェアが `Cy_SysPm_SystemSetMinRegulatorCurrent()`機能を呼び出すかどうかを決定します。

12 関連ドキュメント

アプリケーションノート

[1] AN215656 – PSOC™ 6 MCU デュアルコア システム設計
[2] AN227910 – AIROC™ Wi-Fi & Bluetooth®コンボチップおよび PSOC™ 6 MCU を使用した PSOC™ 6 低電力システム設計

サンプルコード

[1] CE219881 – PSOC™ 6 MCU: Switching between power modes
[2] CE226306 – PSOC™ 6 MCU: Power measurements
[3] CE218542 - HAL: RTC Alarm periodic wakeup
[4] CE218129 – PSOC™ 6 MCU: Wakeup from hibernate using a low-power comparator

改訂履歴

版数	発行日	変更内容
**	2018-07-13	これは英語版 002-19528 Rev. **を翻訳した日本語版 Rev. **です。英語版の改訂内容: New application note.
-	-	この版は英語版のみです。英語版の改訂内容: Updated power mode names to match TRM names; CPU active, CPU, sleep, CPU deep sleep, system LP, system ULP, system deep sleep, system hibernate.
*A	2020-03-25	• これは英語版 002-19528 Rev. *B を翻訳した日本語版 Rev. *A です。英語版の改訂内容: Added CE226306 PSOC™ 6 MCU Power Measurements • Added reference to AN227910 Low-Power System Design AIROC™ CYW43012 Wi-Fi & Bluetooth® combo chip and PSOC™ 6 MCU • Added sections on Low Power Assistant, Power Estimator, measuring current with a DMM, and approximating power consumption • Added design details for the following IP blocks: CPU の無効化、CPU 間のタスクの分割、SRAM、TCPWM、SCB 、オーディオサブシステム、USB、低消費電力コンパレータ、SAR ADC、電圧 DAC、オペアンプ • Added PSOC™ 6 Power Calculator Excel spreadsheet file download
*B	2021-03-15	これは英語版 002-19528 Rev. *C を翻訳した日本語版 Rev. *B です。英語版の改訂内容: Complete rewrite of Chapter 3.3 Backup Power Domain to add more details on its operation.
*C	2021-12-27	これは英語版 002-19528 Rev. *D を翻訳した日本語版 Rev. *C です。英語版の改訂内容: Added reference to AN230938 – PSOC™ 6 MCU low-power analog and added its content to the PSoC_6_Power_Estimator.xlsx power estimation spreadsheet.
-	-	この版は英語版のみです。英語版の改訂内容: Template update
-	-	この版は英語版のみです。英語版の改訂内容: No content change; added the "PSoC™ 6 Power Calculator (version 2).xlsx" file in the source of DMS.
*D	2025-01-08	• これは英語版 002-19528 Rev. *G を翻訳した日本語版 Rev. *D です。英語版の改訂内容: Updated to support ModusToolbox™ and removed PSoC™ Creator related content • Updated the power calculator spreadsheet to support the PSoC™ 6 devices with 512 KB flash (CY8C6xx5) • Replaced the PDL APIs with HAL APIs
*E	2026-08-10	• これは英語版 002-19528 Rev. *H を翻訳した日本語版 Rev. *E です。英語版の改訂内容: No content change • Updated the "PSOC™ 6 Power Calculator (version 4).xlsx" file to fix the issue • Updated PSoC™ to PSOC™ in adherence to the latest branding guidelines

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-yhc1648542360833

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られます。

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。