PSOC™ 6 MCU でのクアッド SPI (QSPI)/シリアルメモリインターフェース (SMIF)の使用法

本書について

適用範囲と目的

AN228740 は、PSOC™ 6 MCU で QSPI/SMIF を使用するためのガイドラインを提供します。PSOC™ 6 MCU QSPI は、外部シリアルメモリデバイスと通信するためのインターフェースを提供します。このアプリケーションノートでは、eXecute In Place (XIP) モードまたはコマンドモードで QSPI をアプリケーションに組み込む方法について説明します。このアプリケーションノートでは、QSPI ブロックで実装している機能と、ユーザーのアプリケーションとして設定する方法について説明します。

関連製品ファミリ

PSOC™ 6 MCU

その他のサンプルコードが必要な場合は、以下を参照してください。

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。ビデオ トレーニング ライブラリについてはここからご覧ください。

目次

1 はじめに

MCU は、多くの組込みシステムにおいて、センサーやその他の外部デバイスからのデータを処理するために使用されます。MCU は、実行中のファームウェア以外のデータを保存するためのオンチップメモリが限られていることが多くあります。これにより、画像やオーディオファイルなどのより重要なデータタイプの処理が複雑になり、ファイルが MCU のメモリに収まらない可能性があります。さらに、アルゴリズムが複雑になり、ファームウェアイメージのサイズが大きくなると、MCU のオンチップメモリの大きさが十分でなくなる可能性があります。このメモリ不足を解消するために、外部シリアルメモリデバイスを組込みシステムに追加して、使用可能なストレージを大幅に増やすことができます。

PSOC™ 6 MCU には、外部シリアルメモリデバイスへのアクセスを簡素化するシリアルメモリインターフェース (SMIF) ハードウェアブロックが含まれます。このブロックは、標準的な SPI, デュアル SPI, クアッド SPI, デュアルクアッド SPI, オクタル SPI など、さまざまな SPI ベースのシリアルインターフェースをサポートします。このブロックは、CPU コアが大容量の外部メモリから少ない遅延で直接コードを実行できる eXecute-In-Place (XIP) モードもサポートしています。

このアプリケーションノートでは、QSPI を使用してインフィニオン PSOC™ 6 MCU デバイスがシリアルメモリデバイスと通信する方法を示します。このドキュメントは、QSPI を使用するための 2 つの簡単なソフトウェアアプローチで始まります。これには、QSPI の使用をすぐに開始するためのリファレンスとして使用できるサンプル プログラム コードが含まれます。QSPI とその機能の詳細については、ソフトウェアの例の後で説明します。本アプリケーションノートで説明する内容は、以下のとおりです。

- QSPI の特長
- PSOC™ 6 MCU QSPI を外部メモリ デバイスと連携させるための設定
- QSPI におけるオンザフライ (OTF) 暗号化と復号化
- QSPI キャッシング
- QSPI eXecute-In-Place (XIP) およびコマンドモード

さらに、このアプリケーションノートでは、外部メモリの保護, プログラミングツールを使用した外部メモリのプログラミング, QSPI 設定ツールで使用する設定の作成など、いくつかのシステムレベルのトピックについても説明します。

市場に出回っているほとんどのシリアルメモリデバイスは Quad-SPI (QSPI) インターフェースをサポートしているため、このアプリケーションノートの残りの部分では、メモリインターフェースとそのすべての設定を指す一般的な用語として QSPI を使用します。

このアプリケーション ノートは、デバイスのデータシートまたはテクニカル リファレンス マニュアル (TRM) に記載されている基本的な PSOC™ 6 MCU アーキテクチャに精通していることを前提としています。

2 QSPI のスタートガイド

QSPI ブロックは、SPI, デュアル SPI, クアッド SPI, デュアルクアッド SPI, およびオクタル SPI モードでシリアルメモリデバイスにアクセスするための専用ハードウェアを提供します。このブロックは、ModusToolbox™ソフトウェア環境で完全にサポートされており、QSPI ハードウェアにアクセスするための複数の API レイヤーがあります。

2.1 シリアルフラッシュライブラリの使い方

シリアルフラッシュライブラリは、QSPI ブロックを簡単に使い始めるための方法です。このライブラリは、ほとんどのシリアルフラッシュメモリデバイスにアクセスするために必要なすべての機能をサポートしています。このライブラリは、ほとんどの設定手順を自動的に処理する単純な関数呼び出しも提供します。その結果、シリアルフラッシュライブラリは使いやすく、ほとんどのユースケースに適しています。シリアルフラッシュライブラリ関数は、ハードウェア抽象化レイヤー (HardwareAbstraction Layer, HAL) とペリフェラル ドライバ ライブラリ (Peripheral Driver Library, PDL) の関数の組み合わせを呼び出します。

次の手順に従って、ModusToolbox™の Eclipse IDE を使用してシリアルフラッシュライブラリの使用を開始します。

1. **Library Manager** でライブラリを有効にします (図 1 を参照)
   - **a.** Library Manager を選択してください
   - **b.** Libraries タブに移動してください
   - **c.** **Serial Flash Library** を選択してください
   - **d.** Apply を選択してください

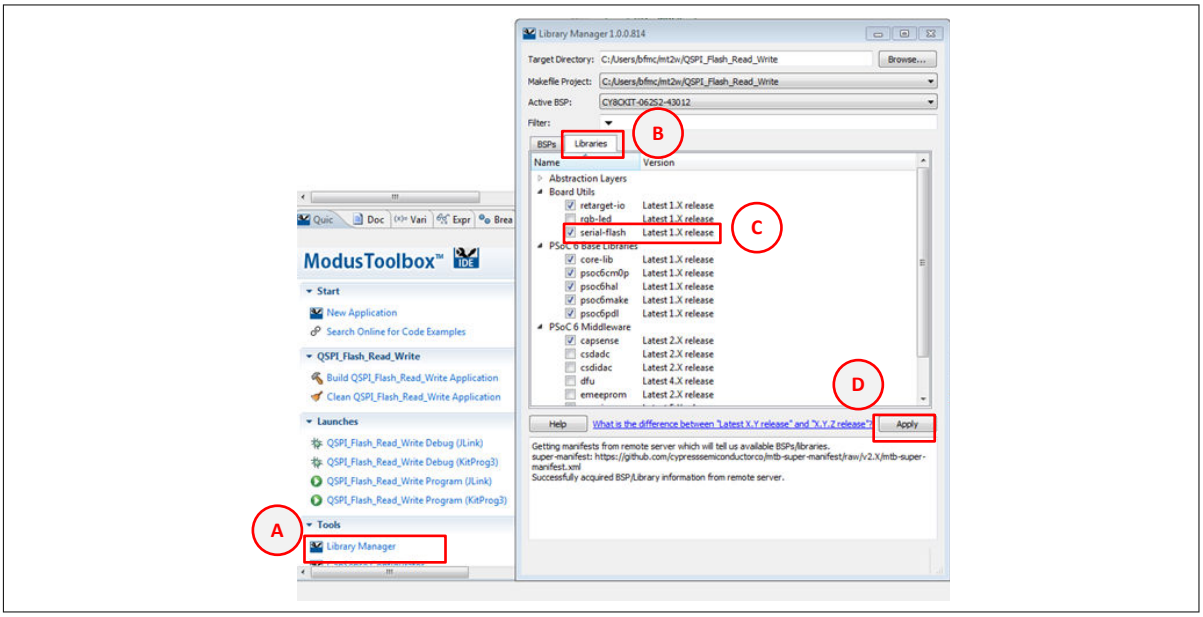

図 1 Serial Flash Library の追加

2. 必要なライブラリをアプリケーションの `main.c` に含めます。

```
#include "cyhal.h" /* HAL */
#include "cycfg.h" /* Auto-generated system configuration headers */
#include "cybsp.h" /* Board specific pin and peripheral definitions */
#include "cycfg_qspi_memslot.h" /*QSPI external memory configuration structures */
#include "cy_serial_flash_qspi.h" /* QSPI library functions */
```

3. main 関数で、ボード ペリフェラル, UART, および QSPI ブロックを初期化します。QSPI の設定は、キットのボードサポートパッケージ (BSP) に含まれています。

```
/* Initialize the device and board peripherals */
result = cybsp_init();
CY_ASSERT(result == CY_RSLT_SUCCESS);
/* Enable global interrupts */
 __enable_irq();
/* Initialize retarget-io to use the debug UART port */
cy_retarget_io_init(CYBSP_DEBUG_UART_TX, CYBSP_DEBUG_UART_RX, CY_RETARGET_IO_BAUDRATE);
/* Initialize qspi block and external memory device */
cy_serial_flash_qspi_init(smifMemConfigs[MEM_SLOT_NUM], CYBSP_QSPI_D0, CYBSP_QSPI_D1,
CYBSP_QSPI_D2, CYBSP_QSPI_D3, NC, NC, NC, NC, CYBSP_QSPI_SCK, CYBSP_QSPI_SS,
QSPI_BUS_FREQUENCY_HZ);
```

4. 読み出しデータと書き込みデータを格納するためのデータ バッファを作成します。データを外部メモリに転送します。データを読み戻し、検証のために UART 端末に表示します。次の例では、外部メモリのアドレス 0x00040000 へのデータの書き込みと読み出しを行っています。このアドレスは、PSOC™ 6 MCU のローカルアドレス範囲からではなく、外部メモリのベースアドレスからオフセットされます。

```
/* Read/write buffers */
uint8 rxBuffer[PACKET_SIZE];
uint8 txBuffer[PACKET_SIZE];
/* Write the content of the txBuffer to the memory */
cy_serial_flash_qspi_write(0x00040000, PACKET_SIZE, txBuffer);

/* Read back after Write for verification */
cy_serial_flash_qspi_read(0x00040000, PACKET_SIZE, rxBuffer);
```

2.2 Peripheral Driver Library の使用

PDL には、QSPI ブロックのレジスタに直接アクセスする一連の関数と構造体が用意されているため、QSPI ブロックを介して実行されるトランザクションを完全に設定できます。こうした設定が可能であることにより、PDL は、より細かい制御が必要なアプリケーションでの使用に適しています。

Eclipse IDE for ModusToolbox™ソフトウェアを使用して QSPI の PDL の使用を開始するためには、次の操作を行います。

1. デバイスコンフィギュレータを使用して QSPI ハードウェアを有効にします。図 2 に、外部フラッシュデバイスを備えた PSOC™ 6 MCU キットでの使用に適した設定例を示します。
   a. **Quick Panel** で、**Device Configurator** をクリックしてください
   b. **Quad Serial Memory Interface (QSPI)** 0 のチェックボックスをオンにして有効にしてください
   c. 図 2 に示されているように、**QSPI** ブロックを設定してください
   d. **Interface Clock** の横にある「Go to Signal」ボタンを選択して、クロック設定タブを開いてください
   e. クロックの **Divider** を **2** に変更してください
   f. 変更を保存して、Device Configurator を閉じてください

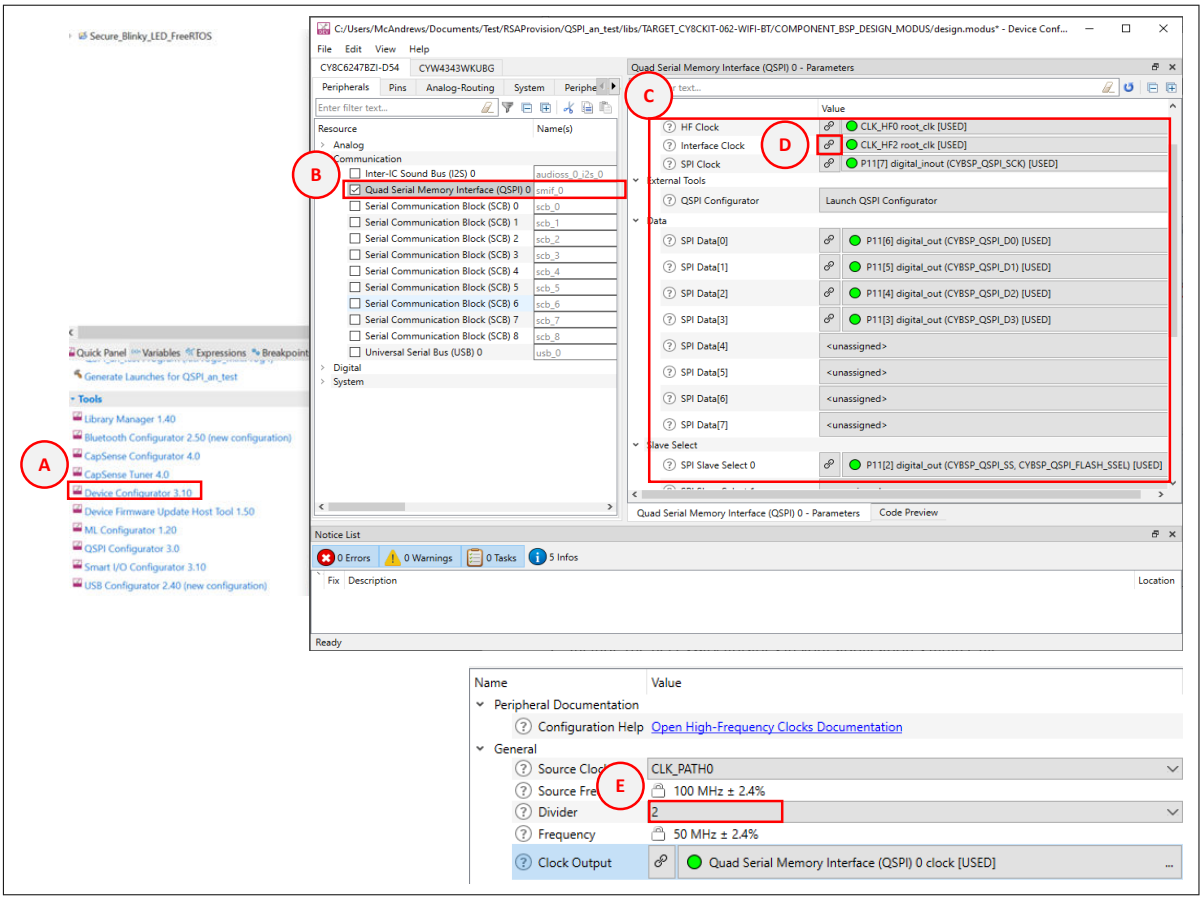

図 2　　QSPI ブロックを設定する手順

2. QSPI Configurator を使用して QSPI コンフィギュレーション構造体を生成します。図 3 に、外部フラッシュデバイスを備えた PSOC™ 6 MCU キットでの使用に適した設定例を示します。

注:　次の手順では、SFDP 準拠デバイスを使用するプロセスについて説明します。特定のメモリ設定のコードを生成する方法など、QSPI Configurator の使用方法の詳細については、QSPI コンフィギュレーター ガイドを参照してください。

a. **QSPI Configurator** を選択してください
b. ドロップダウンメニューから、**Auto detect SFDP** を選択してください
c. 設定を保存してください

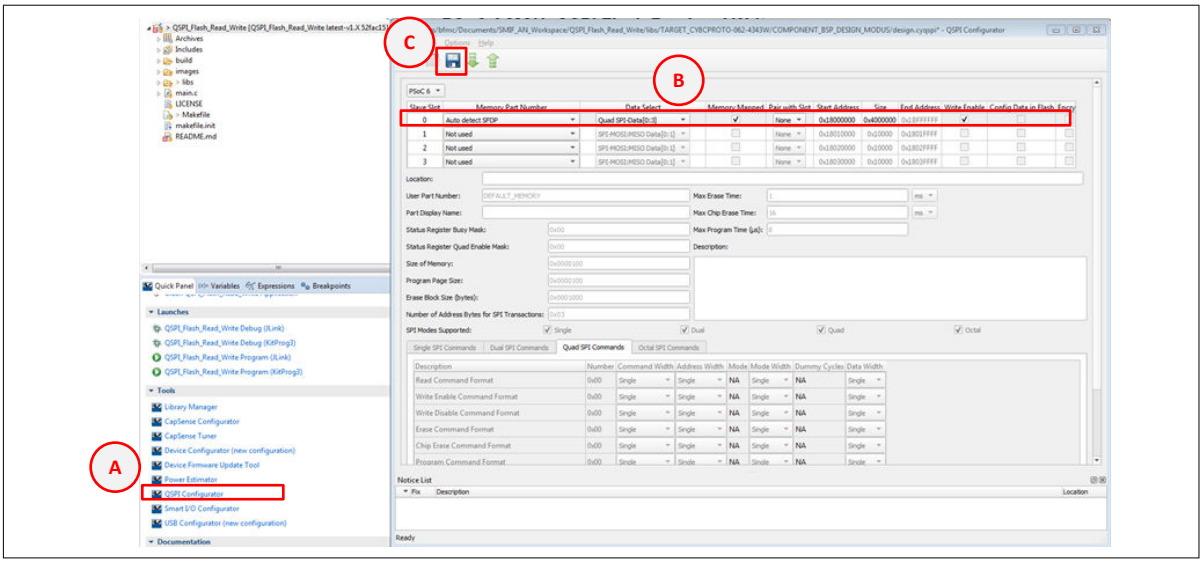

図 3　　QSPI Configurator の使用手順

3. 必要なライブラリをアプリケーションの `main.c` に含めます。

```
#include "cy_pdl.h" /* Peripheral Driver Library */
#include "cybsp.h" /* Board specific pin and peripheral definitions */
#include "cycfg_qspi_memslot.h" /*QSPI external memory configuration structures */
```

4. QSPI ブロックのグローバル コンテキスト変数を作成します。

```
cy_stc_smif_context_t smif_context;
```

5. トランザクション パケット サイズを定義

```
#define PACKET_SIZE             (64u)
```

6. QSPI API 割込み関数 (`Cy_SMIF_Interrupt`) の呼び出しを含む割込みルーチンを作成します。すべての FIFO 操作でこの割込みが使用されます。

```
void SMIF_Interrupt_User(void)
{
    Cy_SMIF_Interrupt(SMIF0, &smif_context);
}
```

7. `main` 関数では、周辺機能とグローバル割込みを初期化します。

```
/* Initialize the device and board peripherals */
result = cybsp_init();
CY_ASSERT(result == CY_RSLT_SUCCESS);
/* Enable global interrupts */
__enable_irq();
```

8. main 関数で、SMIF 割込みを設定します。

```
cy_stc_sysint_t smifIntConfig =
{
    #if (CY_CPU_CORTEX_M0P)
        /* .intrSrc */ NvicMux7_IRQn,
        /* .cm0pSrc */ smif_interrupt_IRQn,
    #else
       /* .intrSrc */ smif_interrupt_IRQn, /* SMIF interrupt number */
    #endif
    /* .intrPriority */ 7u
};
(void) Cy_SysInt_Init(&smifIntConfig, SMIF_Interrupt_User);
```

9. main 関数では、QSPI ブロックを初期化して有効にします。

```
/* SMIF initialization */
Cy_SMIF_Init(SMIF0, &smif_0_config, TIMEOUT_1_S, &smif_context);
Cy_SMIF_Enable(SMIF0, &smif_context);  /* Enable the SMIF Interrupt */
#if (__CORTEX_M == 0)
   NVIC_EnableIRQ(NvicMux7_IRQn);
#else
   NVIC_EnableIRQ(smif_interrupt_IRQn);
#endif
```

10. 外部メモリが Serial Flash Discoverable Parameters (SFDP) をサポートしている場合は、パラメータを検出します。手動メモリ設定の場合、この手順はオプションであり、デバイスを XIP モードで使用する場合にのみ必要となります。

```
/* Memslot level initialization */
Cy_SMIF_MemInit(SMIF0, &smifBlockConfig, &smif_context);
```

11. 外部メモリデバイスがクアッドモードをサポートしている場合は、クアッドモードを有効にします。

```
bool isQuadEnabled = false;
Cy_SMIF_MemIsQuadEnabled(SMIF0, smifBlockConfig.memConfig[0],
                                                    &isQuadEnabled,
                                                    &smif_context);
Cy_SMIF_MemEnableQuadMode(SMIF0,
                          smifBlockConfig.memConfig[0],
                          5000,
                          &smif_context);
```

12. 外部メモリデバイスとのデータのトランザクションを開始します。以下は、`txBuffer` からデータを送信する書き込みシーケンスの例です。

```
uint8_t txBuffer[PACKET_SIZE];
uint8_t rxBuffer[PACKET_SIZE];
/* Erase before write */
Cy_SMIF_MemEraseSector(SMIF0, smifMemConfigs[0],
             0x00,
             smifMemConfigs[0]->deviceCfg->eraseSize,
             &smif_context);

/* Read to rxBuffer after Erase to confirm that all data is 0xFF */
Cy_SMIF_MemRead(SMIF0, smifMemConfigs[0],
               smifMemConfigs[0]->deviceCfg->eraseSize,
               rxBuffer,
               PACKET_SIZE,
               &smif_context);

/* Write txBuffer to the external memory */
Cy_SMIF_MemWrite(SMIF0, smifMemConfigs[0],
               smifMemConfigs[0]->deviceCfg->eraseSize,
               txBuffer,
               PACKET_SIZE,
               &smif_context);
```

3 QSPI の特長

QSPI は、PSOC™ 6 MCU と外部シリアルメモリデバイス間の高度に設定可能なインターフェースを提供します。QSPI ブロックには、キャッシング, XIP モード, コマンド モード, および暗号化を可能にするサブコンポーネントが含まれています。

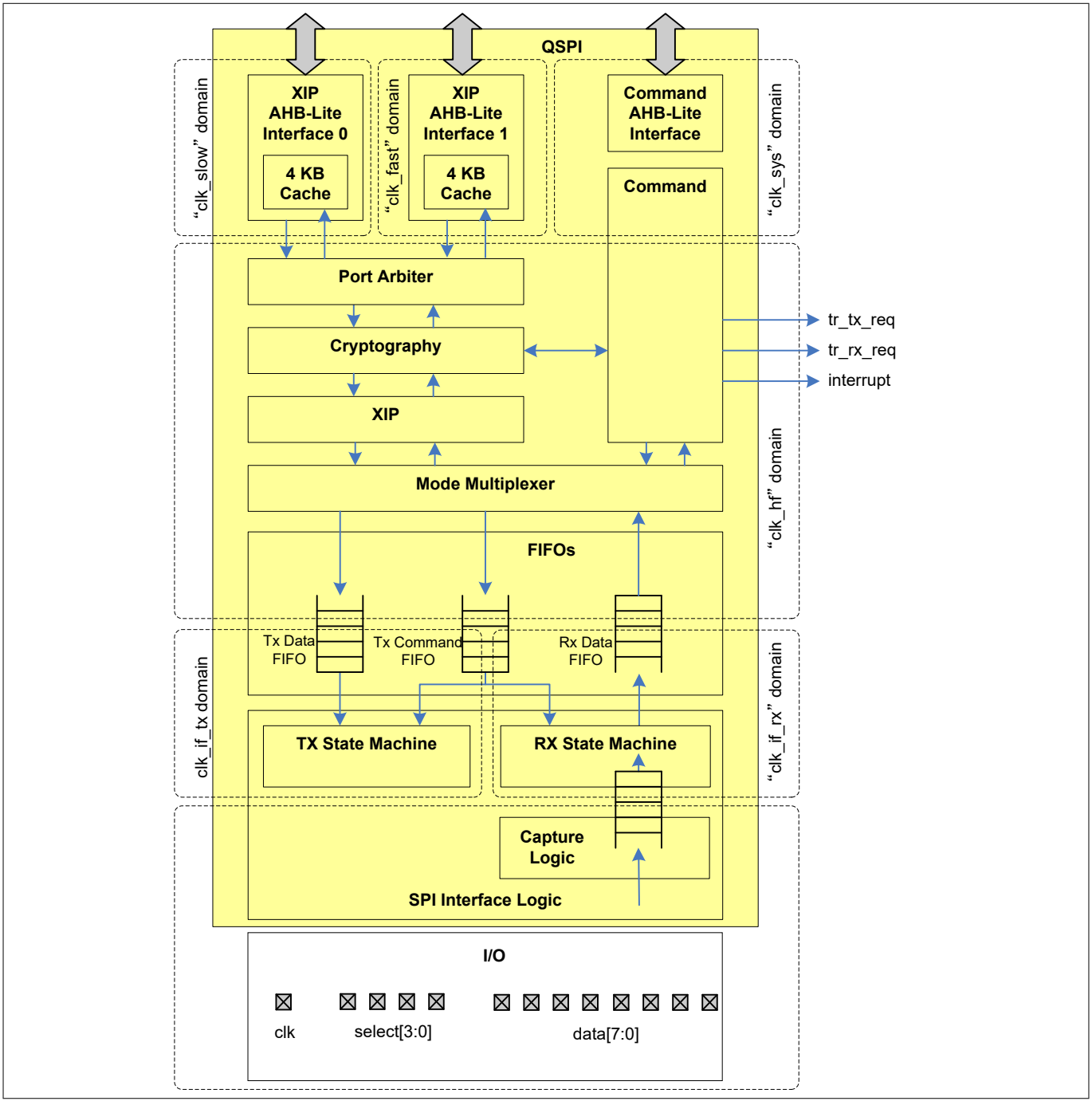

図 4 QSPI ブロック アーキテクチャ

3.1 クロックドメイン

QSPI ブロックには 3 つの AHB-Lite バス インターフェースがあり、2 つは XIP モード用、1 つは Command モード用です。XIP モード インターフェースは、1 つの高速ドメインと 1 つの低速ドメインで構成されます。Arm® Cortex®-M4 は、高速ドメインで唯一のバスマスターです。低速ドメインでは、Cortex®-M0, Crypto, Datawire0, および Datawire1 をバスマスターにできます。Command モード動作の場合、バス インターフェースは `clk_hf` からの分

周クロックである clk_sys ドメインにあります。Command モードでは、バス マスターは XIP モードの任意のバス マスターにできます。

ブロック内の FIFO は、SPI インターフェースのクロックドメインで動作します。暗号化, モードマルチプレクサ, ポートアービターなどの残りのブロックコンポーネントは、高周波数クロック領域で動作します。

3.2 モード

QSPI ハードウェアにはモード マルチプレクサが用意されており、Command モードまたは XIP モードのいずれかで QSPI ブロックを操作できます。PDL では、Cy_QSPI_SetMode()関数を呼び出すことによって実行時にこのモードを変更できます。モードは、進行中の転送が完了した後にのみ変更するべきであることに注意してください。Cy_SMIF_BusyCheck()関数を呼び出して、QSPI ブロックがビジー状態にならないようにする必要があります。

3.2.1 Command モード

これは QSPI ブロックのデフォルト モードであり、通常は大規模なデータ ストレージに使用されます。このモードでは、データ転送は FIFO にアクセスすることによって開始されます。ソフトウェアは、コマンド バイトを TX コマンド FIFO に転送し、データ バイトを TX および RX FIFO に転送できます。このモードでは、使用可能な FIFO エントリまたは使用されている FIFO エントリの数に応じてトリガーが生成されます。トリガー tr_tx_req は、TX データ FIFO のエントリが TX_DATA_FIFO_CTL. TRIGGER_LEVEL フィールド で指定されているエントリよりも少ない場合にアクティブになります。トリガー tr_rx_req は、RX データ FIFO に RX_DATA_FIFO_CTL. TRIGGER_LEVEL フィールドで指定された数よりも多くのエントリがある場合にアクティブになります。

Command モードは、外部メモリを設定または消去するための転送を含む、任意の SPI 転送を柔軟に実装できます。ただし、Command モードでは、アプリケーション コードでオペコード, スレーブ アドレス, ダミー サイクル, およびデータを生成する必要があります。その結果、トランザクションごとに多くの CPU サイクルが消費されます。これらの余分なサイクルは、コードの実行など、少量のデータにアクセスする場合には望ましくありません。ただし、このモードでは、1 つのトランザクションで最大 65535 バイトを転送できます。したがって、Command モードの利用は、一括データ転送や、画像やその他の大きなデータ タイプなど、アクセス頻度の低いデータに推奨されます。

3.2.2 XIP モード

このモードは、通常、外部メモリ デバイスからコードを実行するために使用されます。このモードでは、QSPI ブロックはソフトウェアの介入なしに SPI 転送を自動的に生成します。外部メモリ空間は、2 つの XIP AHB-lite インターフェースのいずれかを介して、PSOC™ 6 MCU のアドレス空間内の設定可能なアドレス範囲にマッピングされます。その結果、XIP モードでの外部メモリアクセスは個別のソフトウェアの介入を必要とせず、外部メモリに格納されたデータには他の変数と同様にアクセスできます。

3.3 キャッシュ

QSPI ブロックには、XIP インターフェースごとに専用の 4 KB キャッシュもあります。1 つは CM4 用で、もう 1 つは CM0 と DMA 用です。これらのキャッシュは初期設定でイネーブルです。キャッシュにヒットした読み出し転送はキャッシュによって処理されますが、キャッシュにヒットしなかった場合は、キャッシュミスしたサブセクターを補充するために 16 バイトの XIP メモリ読み出し転送が発生します。また、次の 16 バイトのデータを取得してキャッシュを補充するプリフェッチ バッファもあります。これは、XIP 転送が 32 バイトのチャンクで発生することを意味し、キャッシュ用に 16 バイト、プレフェッチバッファ用に 16 バイトの SPI 転送が発生してキャッシュサブセクターとプレフェッチバッファを再充填します。

その結果、XIP モードは、外部メモリからコードを実行するなどの小さな転送に便利で効率的です。ただし、大きな読み出しの場合、キャッシュおよびプリフェッチ バッファのリフィルでは、オペコード, デバイス アドレス, およびダミー サイクルを 32 バイトごとに繰り返し送信する必要があります。これらのリフィルは、コマンド モードには存在しない転送の遅延を追加します。

RAM デバイスの場合、XIP モードは外部メモリに書き込むことができます。XIP モードでの書き込み転送では、キャッシュがバイパスされ、SPI 転送が発生します。バス マスターが外部メモリ デバイスに対して読み出しと書き込みを行うと、書き込みによってキャッシュは自動的に無効になります。

XIP アドレッシング範囲内の場所にデータが書き込まれると、キャッシュ内のデータが無効になる可能性があります。キャッシュから古いデータを読み出さないようにするには、新しいデータが XIP メモリ領域に書き込まれるときにキャッシュを無効にする必要があります。例えば、アドレス `0x18000000` で外部メモリからコードを実行していたときに、QSPI ブロックがコマンド モードに移行し、同じアドレスで書き込みが発生した場合です。ブロックを XIP モードに戻し、そのアドレスから実行すると、エラーが発生する可能性があります。

また、異なる AHB クロックドメインの 2 つのマスターが、DMA ハードウェアや CM4 などの SMIF ブロックを介して外部メモリにアクセスすることもできます。このユースケースでは、各マスターがアクセスする領域が重複しないようにすることを推奨します。これにより、キャッシュに無効なデータが含まれる可能性が最小限に抑えられます。アクセスされた領域がオーバーラップしている場合は、PDL 関数呼び出し `Cy_QSPI_CacheDisable()`を使用してキャッシュを無効にするか、`Cy_QSPI_CacheInvalidate()`を使用して各書き込み後にキャッシュを無効にする必要があります。

3.4 メモリ デバイス信号インターフェース

QSPI ブロックは、外部メモリデバイスと通信する際に SPI マスターとして機能します。QSPI は、クロック極性 (CPOL) が 0 でクロック位相 (CPHA) が 0 の SPI 設定 0 のみをサポートします。さらに、標準的な SPI に加えて、QSPI ブロックはデュアル SPI, クアッド SPI, デュアルクアッド SPI, およびオクタル SPI モードでも動作できます。すべてのモードで、QSPI ブロックはシングル データ レート (SDR) モードを使用して動作します。

QSPI は、データセレクトラインの数によって制限される最大 4 つのメモリデバイスを同時にサポートできます。例えば、QSPI は 8 つのデータラインをサポートしているため、4 つのシングルまたはデュアル SPI メモリデバイスが 8 つのデータラインすべてと、使用可能なすべてのデータセレクトラインを使用するか、もしくは同じ 4 つのメモリデバイスが同じデータラインで異なるデータセレクトラインを使用できることを意味します。同様に、4 つのクアッド SPI またはオクタル SPI デバイスを同時に使用して、データラインを共有しますが、データセレクトラインは別のラインを使用することになります。特定のメモリデバイスの場合、使用するデータラインは隣接している必要があります。特定のメモリデバイスタイプで使用される信号については、表 1 を参照してください。

表 1 さまざまなメモリデバイスに使用される SPI クロック, セレクト, およびデータライン

メモリ デバイス	I/O 信号
シングル SPI メモリ	SCK,$\overline{CS}$, SI, SO。このメモリデバイスには、SI と SO の 2 つのデータ信号があります。
デュアル SPI メモリ	SCK,$\overline{CS}$, IO0, IO1。このメモリデバイスには、2 つのデータ信号 (IO0 と IO1) があります。
クアッド SPI メモリ	SCK,$\overline{CS}$, IO0, IO1, IO2, IO3。このメモリデバイスには、4 つのデータ信号 (IO0, IO1, IO2, IO3) があります。
オクタル SPI メモリ	SCK,$\overline{CS}$, IO0, IO1, IO2, IO3, IO4, IO5, IO6, IO7。このメモリデバイスには、8 つのデータ信号 (IO0, IO1, IO2, IO3, IO4, IO5, IO6, IO7) があります。

各メモリデバイスは、QSPI ブロックの 4 つの「スロット」のいずれかにマッピングする必要があります。メモリデバイスのスロットには、CS ラインを制御するために対応する I / O ピンがあります。ファームウェアが正しいデバイスにアクセスするようにするには、QSPI 設定で外部メモリ CS が接続されているのと同じ CS ラインを使用していることを確認してください。

4 つのセレクトラインのそれぞれについて、正しい設定と不正な設定が存在します。表 2 に、`cy_smif.h` で定義されている正しい設定と、対応するデータ選択列挙型を示します。このデータセットは、QSPI Configurator を使用してデバイスを設定するときに自動的に使用されます。

表 2　　データセレクトラインと使用可能なデバイス設定

cy_en_smif_data_select_t	シングル SPI デバイス	デュアル SPI デバイス	クアッド SPI デバイス	オクタル SPI デバイス
CY_SMIF_DATA_SEL0	spi_data[0] = SI spi_data[1] = SO	spi_data[0] = IO0 spi_data[1] = IO1	spi_data[0] = IO0 … spi_data[3] = IO3	spi_data[0] = IO0 … spi_data[7] = IO7
CY_SMIF_DATA_SEL1	spi_data[2] = SI spi_data[3] = SO	spi_data[2] = IO0 spi_data[3] = IO1	不正	不正
CY_SMIF_DATA_SEL2	spi_data[4] = SI spi_data[5] = SO	spi_data[4] = IO0 spi_data[5] = IO1	spi_data[4] = IO0 … spi_data[7] = IO3	不正
CY_SMIF_DATA_SEL3	spi_data[6] = SI spi_data[7] = SO	spi_data[6] = IO0 spi_data[7] = IO1	不正	不正

spi_data 値は、PSOC™ 6 MCU デバイスの QSPI I/O ピンに対応しています。デバイスの spi_data ピンとして使用できるピンを確認するには、デバイス データシートの代替ピン機能セクションを参照してください。

デュアルクアッド設定も QSPI でサポートされています。デュアルクアッド設定では、2 つのクアッド SPI デバイスが同時に使用され、各デバイスは転送ごとに 1 バイトのニブルを提供します。デバイスはインターフェース クロック信号を共有しますが、異なる CS ラインと個別の I/O ラインを使用します。表 3 に、デュアルクアッド モードの設定を示します。

表 3　　デュアルクアッド SPI 設定

DATA_SEL[1:0]		デュアルクアッド SPI 設定	
CY_SMIF_DATA_SEL0	CY_SMIF_DATA_SEL2	spi_data[0] = IO0 … spi_data[3] = IO3	spi_data[4] = IO4 … spi_data[7] = IO7

正しい設定の説明と適切な設定の図例の解説については、PSOC™ 6 MCU アーキテクチャ TRM を参照してください。

3.5　暗号技術

QSPI ブロックには、データを外部メモリに安全に保存できるようにするための暗号化コンポーネントが含まれています。暗号化と復号化は、AES-128 フォワードブロック暗号に基づいています。書き込み専用 QSPI レジスタ SMIF_CRYPTO_KEY3, ..., SMIF_CRYPTO_KEY0 に格納された 128 ビット キーは、128 ビットのプレーンテキストと共に使用され、暗号テキストが生成されます。暗号テキストの生成方法は、QSPI ブロックが Command モードか XIP モードかによって異なります。

シリアルフラッシュライブラリは暗号化をサポートしていないため、暗号化用の PDL 関数を使用する必要があります。設定とデータ転送にシリアルフラッシュライブラリを使用する場合は、シリアルフラッシュライブラリと PDL を組み合わせて使用できます。

3.5.1　XIP モードでの暗号化

XIP モードでは、暗号化コンポーネントはオンザフライの暗号化と復号化をサポートし、外部メモリから実行されるコードまたは外部 RAM に書き込まれるデータまたは外部 RAM から読み出されるデータに自動的に適用されます。暗号化は、プレーンテキストと呼ばれる入力データのキーを使用した AES-128 暗号化アルゴリズムを使用し

ます。XIP モードのプレーンテキストは、SMIF0_CRYPTO_INPUT レジスタの内容を含む 128 ビットに拡張された 28 ビット XIP アドレスです。

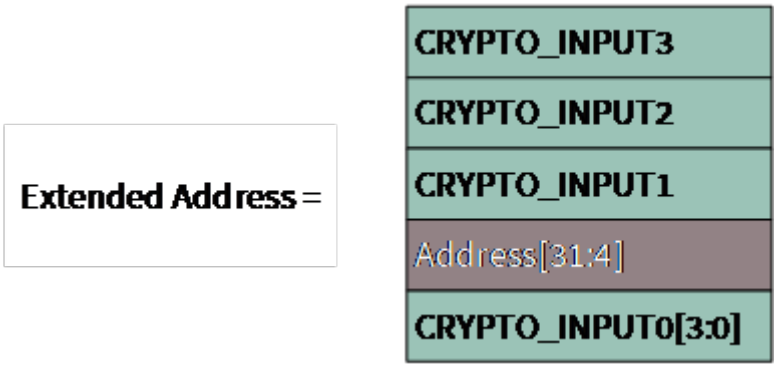

図 5 に、拡張アドレスのフォーマットを示します。

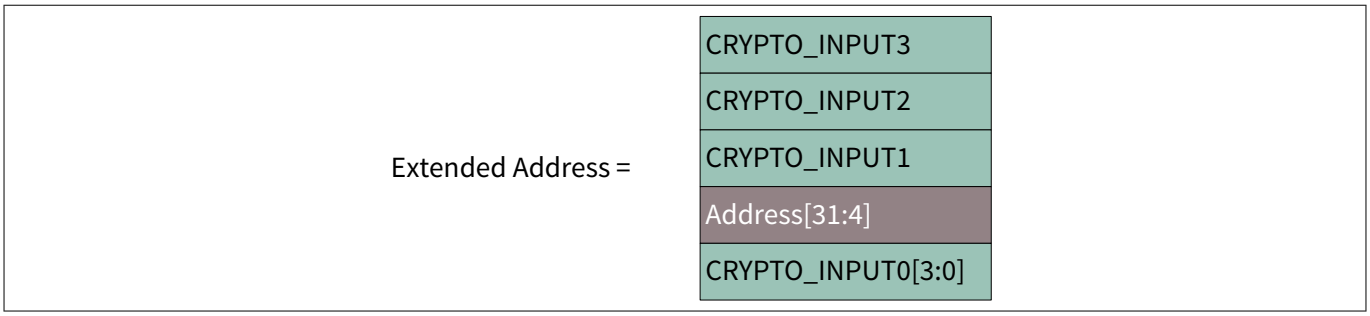

図 5　　XIP 暗号化で使用される拡張アドレス

拡張アドレスにキーを使用して AES-128 を適用した後、結果の暗号テキストは、転送の読み出しまたは書き込みデータと XOR されます。転送されるデータではなくアドレスに AES-128 を適用することにより、暗号化と復号化はオンザフライで行われ、遅延は発生しません。図 6 に、XIP モードでの暗号化プロセス全体を示します。

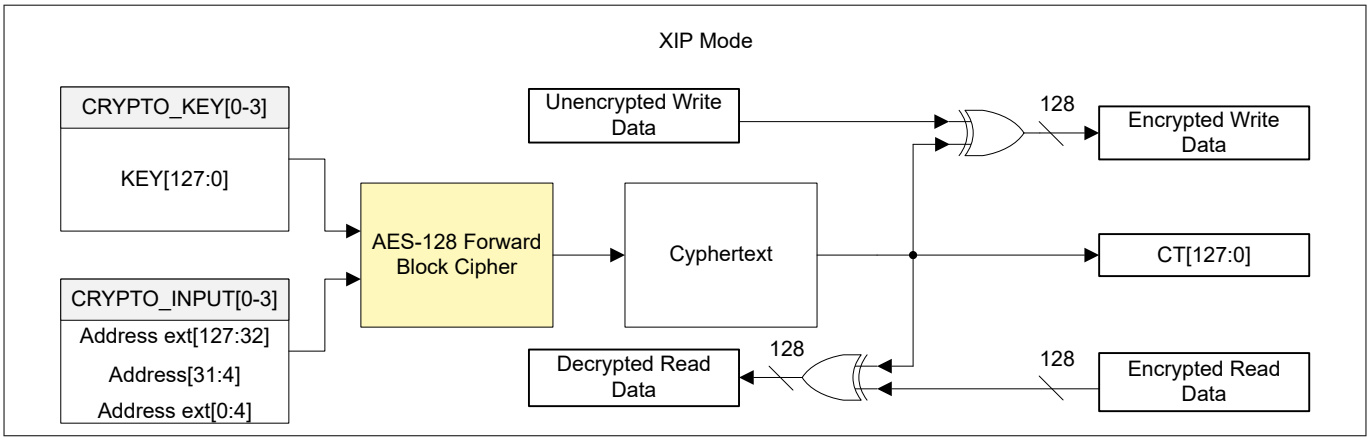

図 6　　XIP モードのオンザフライ暗号化プロセス

XIP モードで暗号化を有効にするには、SMIF0_DEVICEn_CTL.CRYPTO_EN ビットを設定します。ここで DEVICEn は外部メモリ デバイスのメモリ スロットを指します。例えば、スロット 2 のメモリ デバイスで暗号化を有効にするには、次のコード行を使用します。

```
SMIF0->DEVICE[2].CTL |= (1 << SMIF_DEVICE_CTL_CRYPTO_EN_Pos /* 8U */);
```

暗号化を無効にするには、CRYPTO_EN ビットをクリアします。

3.5.2 Command モードでの暗号化

Command モードの暗号化を使用して、ブートローダ, アプリケーションイメージ, または外部フラッシュメモリ内のバルクデータを暗号化できます。Command モードでの暗号化では、データ転送ごとに PDL 暗号化関数 (Cy_SMIF_Encrypt) を個別に呼び出す必要があります。結果の cyphertext は CRYPTO_OUTPUT レジスタに格納され、データを外部メモリに書き込む前にアンパックする必要がありますが、これは Cy_SMIF_Encrypt 関数内で自動的に処理されます。XIP オンザフライ復号化との互換性を保つために、Cy_SMIF_Encrypt 関数は XIP モードの暗号化で使用されるのと同じ暗号化方式を使用します。

Command モードの場合、暗号化フローは通常、次の手順に従う必要があります。

1. 暗号化キーを SMIF_CRYPTO_KEY レジスタにロードしていることを確認してください。
2. PDL 機能 Cy_SMIF_Encrypt を使用して転送するデータを暗号化してください。
3. データを外部メモリに書き込んでください。

復号化の場合:

1. 外部メモリから暗号化されたデータを読み出してください。
2. PDL 関数 Cy_SMIF_Encrypt を使用してデータを復号化してください。

4 エコシステム

QSPI ブロックには、以下で構成されるツールとファイルのエコシステムがあります。

- ModusToolbox™アプリケーション ソフトウェア ライブラリ
- QSPI 設定ツール
- プログラミング ツール

図 7 に、エコシステムのさまざまなコンポーネントと、それらのコンポーネントによって使用または生成されるファイルを示します。

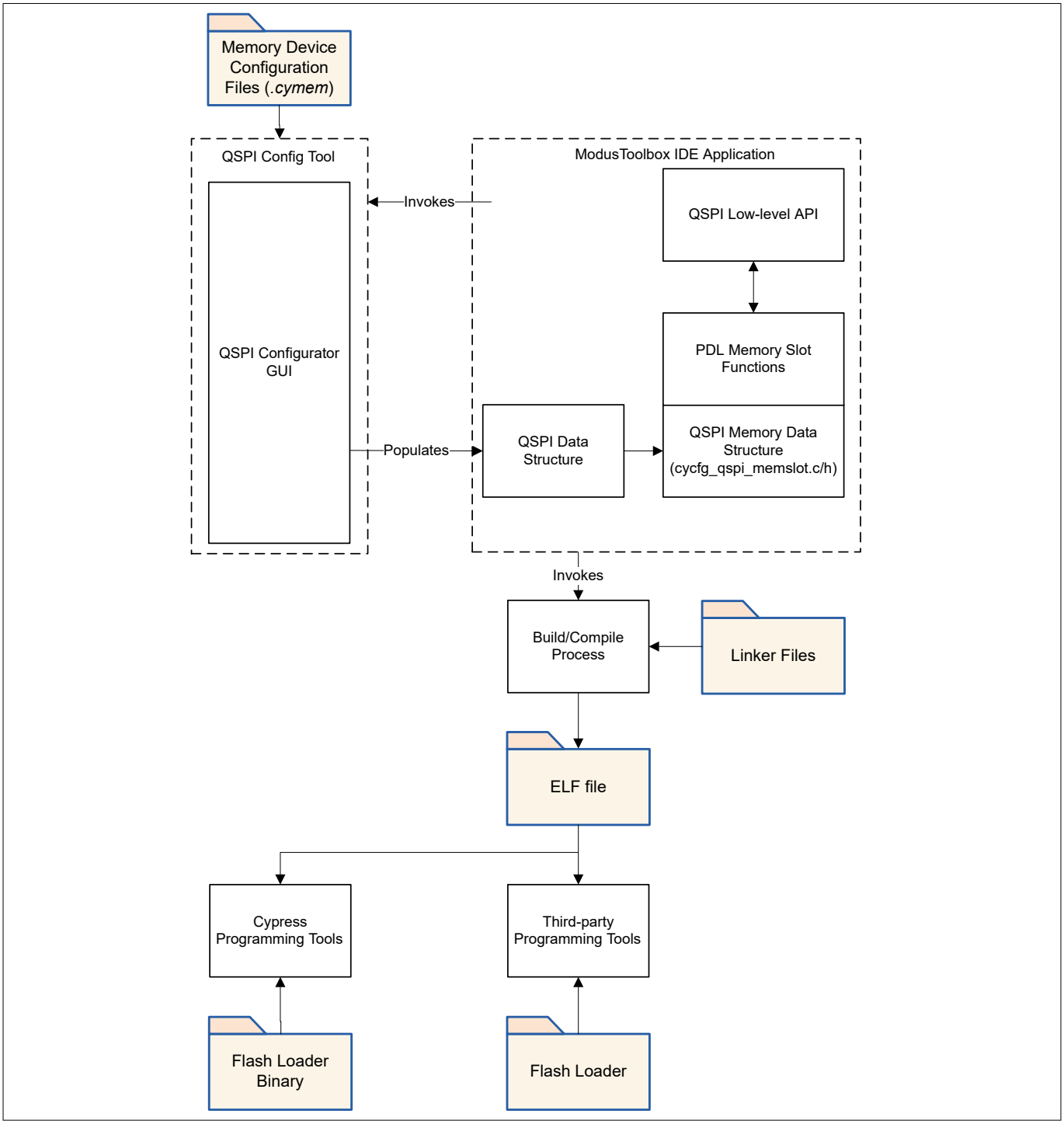

図 7 ツールの QSPI エコシステム

4.1 ModusToolbox™アプリケーション ソフトウェア ライブラリ

ModusToolbox™ソフトウェア環境には、ハードウェアレジスタに直接アクセスすることなく QSPI を使用するために必要なライブラリとファイルが含まれています。PDL は、QSPI ブロックの低レベル設定と、メモリ デバイスにデータを転送するために必要な API を提供します。シリアル フラッシュ ライブラリと HAL は、PDL の上のアクセス レイヤーを提供します。表 4 に、これらのライブラリで使用されるファイルと、それらが属するライブラリを示します。

表 4 ソースファイルとライブラリ

ファイル	説明	ライブラリ
`cy_smif.c/h`	QSPI ハードウェアを設定し、転送を開始するための低レベル API を提供します。QSPI ブロックを直接設定するために使用されます。	PDL
`cy_smif_memslot.c/h`	`cy_smif.c` の 1 つ上のレベル外部メモリにアクセスするための低レベル API (ステータス レジスタや SFDP パラメータなど) を提供します。通常、コマンドモードまたは SFDP 検出に使用されますが、メモリデバイスのステータスチェック関数も定義します。	PDL
`cyhal_qspi.c/h`	チップ固有の設定関数を C 言語で抽象化します。HAL の関数は、ブロックが使用するピン、必要な割込み、および各データ転送のタイムアウトを自動的に設定します。通常、シリアルフラッシュライブラリと組み合わせて使用され、高レベルのメモリアクセスを実現します。	HAL
`cy_serial_flash_qspi.c/h`	QSPI の使用を容易にするために、`cy_smif.c` と `cy_smif_memslot.c` の両方の関数のラッパーを提供します。HAL を使用して、チップ固有の設定をします。これらのファイルに含まれる関数は、可能な設定を制限しますが、使いやすく、ほとんどのメモリ アクセスに必要な機能を提供します。	シリアル フラッシュ ライブラリ
`cy_serial_flash_prog.c`	接続されたシリアルフラッシュメモリのプログラミング方法をプログラミングツールに指示するために必要な変数を提供します。QSPI Configurator ツールで生成されたファイルによって異なります。通常、XIP モードと共に使用して、プログラミング用の外部メモリ アドレスを公開します。	シリアル フラッシュ ライブラリ

4.2 QSPI Configurator ツール

ModusToolbox™ソフトウェア環境には、QSPI Configurator が含まれています。Configurator ツールは、外部メモリ デバイスを使用するように QSPI を設定するためのシンプルなグラフィカル インターフェースを提供します。Configurator は、Peripheral Driver Library の使用のステップ 1 に従って、ModusToolbox™ IDE 内から起動できます。スタンドアロンの Configurator ツールを起動して、ほとんどの IDE で使用できるソース ファイルを生成することもできます。ModusToolbox™の `installation` フォルダに移動し、次のパスに従います。

```
{Install Dir}\ModusToolbox\tools_2.0\qspi-configurator\qspi-configurator.exe
```

Configurator では、メモリデバイス, 接続先のスロットまたは選択ライン, SPI 幅, メモリアドレス範囲, および暗号化 (XIP モードの場合) を選択できます。選択した内容を設定すると、QSPI Configurator ツールは、外部メモリデバイスのパラメータを含むソースファイルとヘッダーファイルを自動的に生成します。デバイスのパラメータは設定用の構造体に格納され、PDL またはシリアルフラッシュライブラリ関数に引数として渡すことで、メモリデバイスに簡単にアクセスできます。

表 5 QSPI コンフィギュレータによって生成されたソースファイル

ファイル	説明
cycfg_qspi_memslot.c/h	QSPI Configurator から自動的に生成されます。QSPI メモリ デバイス設定の定義を提供するか、SFDP 検出用のデフォルトの構造体を設定します。

Configurator ツールは、.cymem ファイルと呼ばれる XML 形式のメモリ ファイルのデータベースから設定値を取得します。これらのファイルと編集可能なテンプレート メモリ ファイルは、通常、ModusToolbox™ IDE とともにインストールされます。これらのファイルの詳細と QSPI Configurator ツールの使用方法については、ユーザ ガイドを参照してください。

4.3 プログラミング ツール

QSPI ブロックは、インフィニオンのキットに付属する KitProg3 などのプログラミング ツールを通じて、外部メモリのプログラミングをサポートします。外部メモリのプログラミングをサポートするようにアプリケーションを設定する方法の詳細については、外部メモリにデータを書き込むを参照してください。

5 コンフィギュレーション

5.1 QSPI コンフィギュレーション構造体のアーキテクチャ

外部メモリデバイスにアクセスするには、使用しているメモリデバイスに対して QSPI ブロックが正しく設定されていることを確認する必要があります。QSPI Configurator ツールによって生成された `cycfg_qspi_memslot.c/h` ファイルには、システムの各設定パラメータを含むネストされた構造体のセットが用意されています。これにより、コマンドリストの作成や各転送の転送パラメータの設定に費やす時間が短縮されます。図 8 に、生成された構造体の構成を示します。

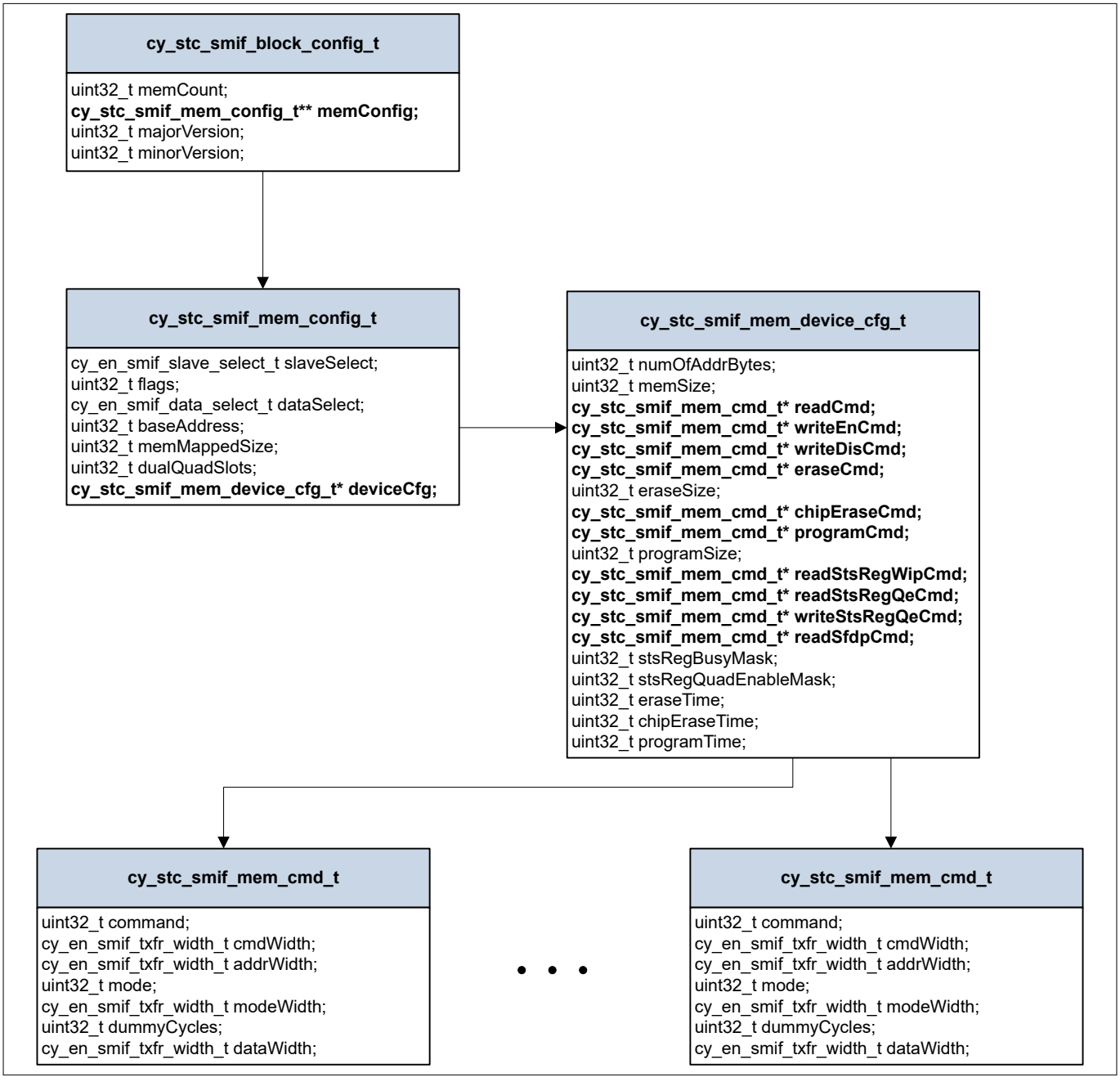

図 8 QSPI コンフィギュレーション構造体のアーキテクチャ

最上位にあるのは `cy_stc_smif_block_config_t` 構造体です。この単純な構造体には、ブロックに接続されているメモリ デバイスの数, QSPI ドライバのバージョン, およびメモリ設定構造体へのダブル ポインタが含まれています。

ブロック設定構造体の中にネストされているのは、メモリ設定構造体です。この構造体には、SPI スレーブ選択スロット, データラインのスロット, ベースアドレス, 外部アドレスのサイズ, デュアルクアッド SPI スロットの数, メモリデバイス構成構造体へのポインタなど、外部メモリデバイスに関するアプリケーションレベルの情報が含まれています。

メモリデバイス設定構造体 `cy_stc_smif_mem_device_cfg_t` には、メモリデバイスにアクセスするために必要なデフォルトのメモリコマンドなど、デバイス固有の情報が含まれています。通常、この構造体の詳細は、メモリ デバイスのデータシートの情報を使用して入力できます。

QSPI アーキテクチャ全体の最下位レベルの構造体は、メモリ デバイスのコマンド構造体 `cy_stc_smif_mem_cmd_t` です。メモリ デバイス構成構造体にリストされている各コマンドには、対応するコマンド構造体があります。これらの構造体は、コマンド幅, アドレス幅, モード, モード幅, データ幅, ダミー サイクル数など、コマンドの要件を指定します。

5.2 コンフィギュレーションの手順

5.2.1 SFDP 検出

シリアル フラッシュ検出可能パラメータ (SFDP) 標準は、シリアル フラッシュ デバイスの機能とアクセス仕様を定義する一連の標準パラメータ テーブルを提供します。これらのテーブルは、SFDP 標準を使用するシリアル フラッシュ デバイスの内部にあり、デバイスへのアクセスに必要な設定を決定するために読み出すことができます。

QSPI ブロックは、SFDP 検出をサポートします。この機能をサポートするすべてのデバイスで、設定を簡素化するために SFDP 検出を有効にすることを推奨します。SFDP 検出を使用すると、QSPI コンフィギュレーション構造体のアーキテクチャに記載されている QSPI コンフィギュレーション構造体が自動的に入力されます。

QSPI Configurator ツールで SFDP 検出を実行するように QSPI ブロックを設定できます。QSPI Configurator の手順に従って、QSPI Configurator ツールを起動します。図 9 に示すように、**Memory Part Number** ドロップダウンメニューから、**Auto detect SFDP** を選択します。選択内容がハードウェア接続に対応するスレーブスロットにあることを確認してください。

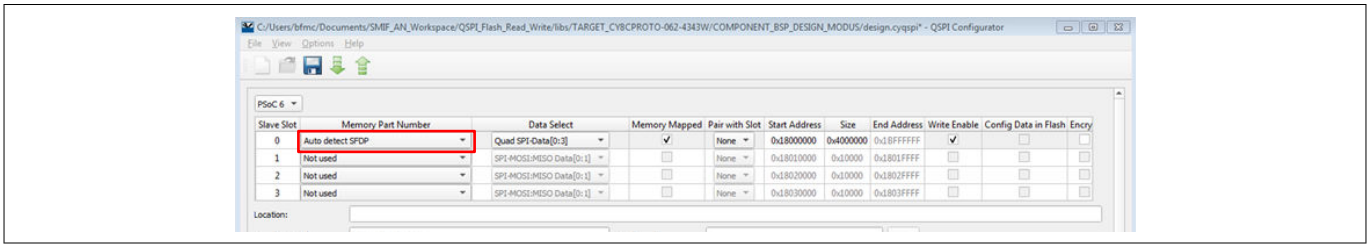

図 9 QSPI コンフィギュレータによる SFDP 検出

このオプションを選択すると、プレフィックス「Auto_detect_SFDP」が付いたコンフィギュレーション構造体と、Configurator ツールで設定されたデフォルト値が生成されます。

```
cy_stc_smif_mem_config_t Auto_detect_SFDP_SlaveSlot_0
```

アプリケーションでは、`Cy_SMIF_MemInit` 関数を呼び出すと SFDP 検出が実行され、検出されたパラメータが構造体に入力されます。

5.2.2 手動設定

お使いのデバイスが SFDP をサポートしていない場合、またはデバイス設定を手動で設定する場合は、**Memory Part Number** ドロップダウンメニューからサポートされている製品番号を選択できます。

カスタムデバイスまたはサポートされていないデバイスの場合は、Configurator ツールを使用して新しいメモリファイル (`*.cymem`) を作成するか、ソースファイル (`cycfg_qspi_memslot.c/.h`) の構造体にデバイスのパラメータを手動で入力することができます。

新しいメモリファイルを作成するには、QSPI Configurator Guide の「Create New Memory File」セクションに記載されている手順に従います。

6 動作の順序

PDL またはシリアルフラッシュライブラリを使用する場合、関数がいくつかの重要な転送ステップを自動的に処理していることに留意してください。書き込み転送の場合、これは通常、書き込みイネーブル コマンドとそれに続くプログラム コマンドの送信を含む 2 段階のプロセスです。読み出しアクセスの場合、通常、読み出しコマンドが唯一の必要な手順です。

データを転送するコマンドの後、QSPI ブロックを再度使用する前に、データ転送が完了し、外部メモリ デバイスが次のトランザクションの準備ができていることを確認することが重要です。これは、`cy_smif_memslot.c/h` の関数内で処理されます。ただし、下位レベルの PDL アクセスの場合は、関数 `Cy_SMIF_GetTransferStatus()`を使用して転送の現在のステータスを判別できます。また、`Cy_SMIF_TransmitCommand()`関数を使用してデバイス固有のステータス コマンドを送信し、外部メモリ デバイスのステータスを読み出すこともできます。お使いのデバイスの正しい読み出しステータス コマンドを確認するには、デバイスのデータシートを参照してください。

さらに、多くの外部メモリ デバイスでは、メモリ デバイスに書き込む前に全消去またはセクタ消去の操作が必要です。これは、デバイス固有の消去コマンドを送信するか、`cy_smif_memslot.c/h` ファイル, シリアル フラッシュ ライブラリ ファイル, または HAL で提供される機能を使用して実現できます。

書き込みデータ転送の一般的なフローは次の手順に従います。

1. 外部メモリがビジーでないことを確認します
2. 書き込み許可コマンドを送信します
3. 書き込み先の外部メモリデバイスのセクターを消去します
4. 外部メモリの消去が終了するのを待ちます。これには時間がかかる場合があります。消去時間の仕様については、メモリ デバイスのデータシートを参照してください。
5. 書き込み許可コマンドを外部メモリデバイスに送信します
6. データを外部メモリへ書き込みます
7. 転送が完了するのを待ちます
8. 外部メモリがレディーになるまで待機します

読み出し転送はより簡単で、読み出しコマンドと転送の完了のチェックのみ実行します。

1. 外部メモリがビジーでないことを確認します
2. 読み出されるデータのアドレスで読み出しコマンドを送信します
3. 転送が完了するのを待ちます

7 外部メモリにデータを書き込む

QSPI は、PSOC™ 6 MCU キットに含まれる OpenOCD や KitProg3 デバイスなどのプログラミングツールによる外部メモリのプログラミングをサポートします。この機能を有効にするには、いくつかの重要な手順に従う必要があり、そうしないとプログラミングが失敗する可能性があります。

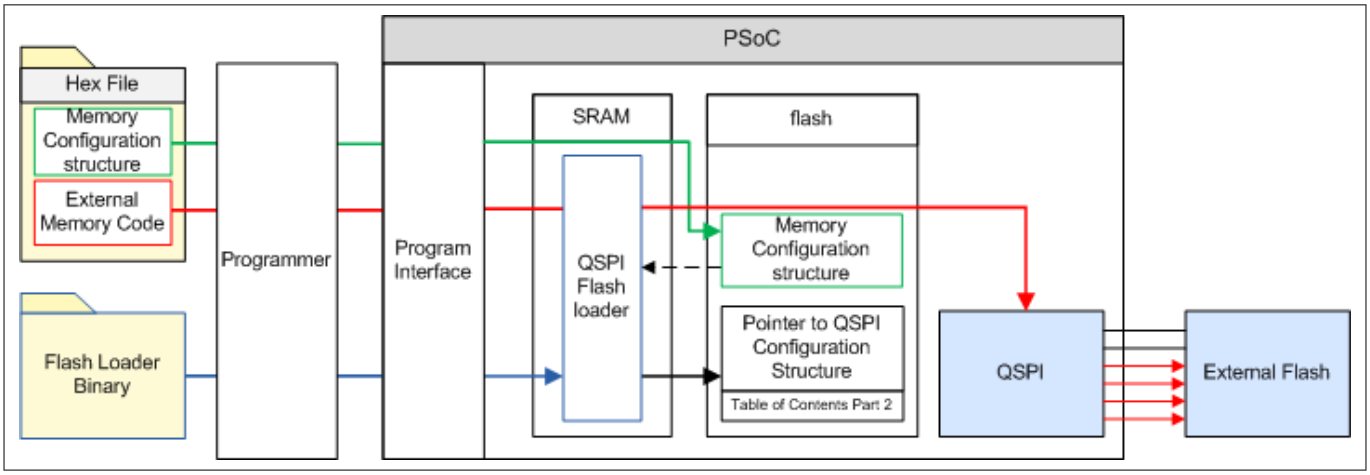

図 10 外部メモリへデータを書き込むときの手順

図 10 の赤い矢印は、QSPI インターフェースを介して外部メモリにプログラムされるコードまたはデータを表しています。コードまたはデータが外部フラッシュに到達する前に、プログラマ, プログラミング インターフェース, QSPI フラッシュローダを経由し、最後に QSPI インターフェースに移動する必要があります。

プログラミング プロセスの最後に、QSPI フラッシュ ローダが PSOC™ 6 MCU SRAM にプログラムされ、実行が開始されます。フラッシュ ローダは、Table of Contents パート 2 (TOC2) 構造体の一部として、フラッシュ内の固定位置から QSPI 設定構造体へのポインタを見つけようとします。この場所にポインタを置くと、フラッシュローダがポインタを見つけて、外部メモリをプログラミングするためのコマンド構造体を持つことができます。

上述の実行は、以下の処理を行ってください。

1. シリアル フラッシュ ライブラリを使用している場合は、`cy_serial_flash_prog.c` ファイルに移動し、定義 `#define CY_ENABLE_XIP_PROGRAM` を追加します。これにより、必要なポインタがフラッシュの正しい位置に自動的に含まれます。これ以上の手順は必要ありません。

   シリアル フラッシュ ライブラリを使用していない場合は、`cy_stc_smif_block_config_t` 構造体へのポインタと NULL 終端を含む構造体を作成します。

```
typedef struct
{
    const cy_stc_smif_block_config_t * smifCfg; /* Pointer to SMIF top-level
configuration */
    const uint32_t null_t; /* NULL termination */
} stc_smif_ipblocks_arr_t;
```

2. この構造体のインスタンスを作成し、既知の場所に配置します。

```
CY_SECTION(".cy_sflash_user_data") __attribute__( (used) )
const stc_smif_ipblocks_arr_t smifIpBlocksArr = {&smifBlockConfig, 0x00000000};
```

3. 次の方法で、構造体を TOC2 に配置します。この構造体は、フラッシュローダが認識するフラッシュ内の所定の固定位置にあります。

```
CY_SECTION(".cy_toc_part2") __attribute__( (used) )
const uint32_t cyToc[128] =
{
    0x200-4,                /* Offset=0x0000: Object Size, bytes */
    0x01211220,             /* Offset=0x0004: Magic Number (TOC Part 2, ID) */
    0,                      /* Offset=0x0008: Key Storage Address */
(int)&smifIpBlocksArr,  /* Offset=0x000C: This points to a null terminated array of SMIF
structures */
    0x10000000u,            /* Offset=0x0010: App image start address */
                            /* Offset=0x0014-0x01F7: Reserved */
    [126] =  0x000002C2,    /* Offset=0x01F8: Bits[1:0] CLOCK_CONFIG(0=8MHz, 1=25MHz,
2=50MHz, 3=100MHz)
                                                Bits[4:2]
LISTEN_WINDOW(0=20ms,1=10ms,2=1ms,3=0ms,4=100ms)
                                                Bits[6:5] SWJ_PINS_CTL (0/1/3=Disable SWJ,
2=Enable SWJ)
                                                Bits[8:7] APP_AUTHENTICATION (0/2/3=Enable,
1=Disable)
                                                Bits[10:9] FB_BOOTLOADER_CTL: UNUSED */
    [127] =  0x3BB30000     /* Offset=0x01FC: CRC16-CCITT (the upper 2 bytes contain the
CRC and the lower 2 bytes are 0) */
};
```

TOC2 の使用法, およびその内容の詳細については、デバイス アーキテクチャ TRM を参照してください。

また、QSPI フラッシュ ローダは SFDP 検出を行いません。つまり、デバイスの設定値はフラッシュメモリに保存されている必要があります。QSPI Configurator ツールを使用してこれを行うには、**Memory Part Number** ドロップダウン メニューからメモリ パーツを選択し、**Config Data in Flash** オプションが選択されていることを確認します。

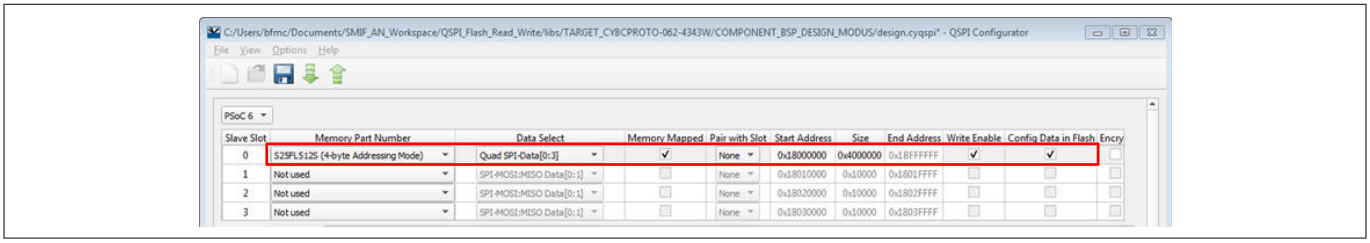

図 11　　外部メモリ プログラミングの設定例

ファームウェアでは、外部メモリのアドレス指定範囲内の場所に関数または変数を配置し始めることができます。これを行うには、変数または関数の宣言の前に属性 `CY_SECTION(".cy_xip") __attribute((used))`を使用して、値を外部メモリ領域に配置できます。

8 QSPI によるセキュリティ

QSPI を使用して機密データや独自のライブラリを保存している場合は、QSPI システムを保護する必要があります。メモリ デバイス信号インターフェースで説明されている暗号化を使用すると、データ自体を保護するのに役立ちますが、暗号化されたデータを読み出したり、XIP セクションに他のコードを挿入したりすることを防ぐことはできません。

QSPI システムを保護するには、いくつかの重要な手順が必要です。最初のステップは、QSPI が提供する暗号化またはその他の堅牢な方法を使用して、データを暗号化していることを確認することです。これにより、データ自体が保護されるため、ハッカーが機密情報を読み出すのが難しくなります。

次のセキュリティ層は、QSPI レジスタ空間を読み出しおよび書き込みアクセスから保護することです。これにより、ハッカーが QSPI ブロック モードを XIP モードからコマンド モードに切り替えて、暗号化されたデータを実行中に読み出すのを防ぐことができます。これにより、暗号化キーの変更も防げます。これを行うには、暗号化キーが正しいこと、および外部メモリの実行可能コードが暗号化されて外部メモリデバイスにプログラムされていることを確認します。次に、ペリフェラル保護ユニット (PPU) を使用することで QSPI ブロックのアクセス制限を設定し、QSPI ブロック内のレジスタの読み出し/書き込み制限を設定できます。

XIP モードを使用するアプリケーションの場合、外部メモリ内のコードも保護する必要があります。このメモリは複数のマスタ (DMA や CPU など) で共有およびアクセスできるため、共有メモリ保護ユニット (SMPU) を使用する必要があります。SMPU を使用すると、領域ベースアドレス, 保護するメモリのサイズ, およびメモリ領域のアクセス制限を設定できます。デフォルトでは、外部メモリ領域は 0x18000000 ですが、これは QSPI Configurator ツールで変更できます。

最後に、外部メモリ領域に格納されているすべてのコードが、アプリケーションイメージの残りの部分とともに検証されていることを確認します。PSOC™ 6 MCU を使用したコード署名と安全なシステムアーキテクチャの詳細については、AN221111 を参照してください。

9 パフォーマンス

シリアルメモリデバイスは、フレームバッファの外部メモリデバイスとして、または EEPROM のように頻繁にアクセスされるメモリとしてよく使用されます。このような場合、高スループットで低遅延の転送を行うことが重要です。QSPI ブロックは、コマンド モード転送、またはキャッシュ ミスがある XIP モードでの転送において SPI プロトコルに従います。このような場合、転送の遅延は下式で求められます。

$$\text{Interface clock cycles} = \left[\frac{\text{opcode size}}{\text{opcode width}}\right] + \left[\frac{\text{address size}}{\text{address width}}\right] + \left[\frac{\text{mode size}}{\text{mode width}}\right] + \text{dummy cycles} + \left[\frac{\text{data size}}{\text{data width}}\right] \quad \text{Number}$$

コマンドモードでデータを暗号化する場合、暗号化プロセスには約 13 *clk_hf*サイクルかかるため、計算された転送時間が 13 サイクルを超えている限り、遅延は発生しません。

キャッシュが有効になっている XIP モードの場合、キャッシュ内でヒットしたアクセスはすべてキャッシュによって処理されます。キャッシュは 16 B で、プリフェッチ バッファはキャッシュの直後に続く連続した 16 B であるため、連続した 32 B のアクセスごとに、次の 32 B に対して 1 回のリフィルが発生することに注意してください。このリフィルは、コマンドモード転送と同様に動作し、上記の式を使用して計算できます。これにより、XIP モード アクセスは短時間の読み出しまたは実行アクセスに適していますが、大規模な読み出しアクセスでは、繰り返しのリフィルによる余分なサイクルにより、望ましくない遅延が追加されます。したがって、大規模なデータ転送には Command モードが推奨されます。

XIP モードでの暗号化はオンザフライで行われ、遅延は発生しません。

10 まとめ

このアプリケーションノートでは、PSOC™ 6 MCU QSPI ブロックを使用して外部メモリデバイスにアクセスする方法について説明しました。簡単な参照フローを提供し、ブロックの機能を説明し、セキュリティ設計要件とブロックのパフォーマンスについて説明しました。PSOC™ 6 MCU 内の QSPI ブロックのアーキテクチャに関するローレベルの詳細については、アーキテクチャ TRM を参照してください。さらに、QSPI で外部メモリを使用する方法を示すサンプルコードが多数あります。これらの例については、関連資料を参照してください。

関連資料

PSOC™ 6 MCU リソースの包括的なリストについては、コミュニティの KBA223067 を参照してください。PSOC™ 3, PSOC™ 4, および PSOC™ 5LP リソースの包括的なリストについては、コミュニティの KBA86521 を参照してください。

アプリケーションノート	
AN210781 – Getting Started with PSoC™ 6 MCU with Bluetooth Low Energy (BLE) Connectivity	Bluetooth® Low Energy 接続デバイスを備えた PSOC™ 6 MCU と、最初の PSOC™ Creator プロジェクトの構築方法について説明します。
AN221774 – PSOC™ Creator での PSoC™ 6 MCU 入門	このアプリケーションノートは、PSOC™ 6 MCU アーキテクチャと開発ツールの探索に役立ち、ModusToolbox™と PSOC™ Creator を使用して最初のプロジェクトを作成する方法を示します。
AN215656 – PSoC™ 6 MCU デュアルコア システム設計	PSOC™ 6 MCU のデュアル CPU アーキテクチャについて説明し、シンプルなデュアル CPU デザインを構築する方法を示します
AN219434 – PSOC™ 6 MCU - 生成コードの IDE へのインポート (PSoC™ Creator コード)	PSOC™ Creator で生成されたコードをお好みの IDE にインポートする方法について説明します
サンプルコード	
CE220823 – PSoC™ 6 MCU SMIF Memory Write and Read Operation	この例では、PSOC™ 6 MCU のシリアルメモリインターフェース (SMIF) への書き込みおよび読み出し操作を示します
CE222460 – SPI F-RAM Access Using PSoC™ MCU SMIF	CE222460 は、SMIF コンポーネントを使用して PSOC™ 6 MCU に SPI ホストコントローラを実装し、SPI F-RAM のさまざまな機能にアクセスする方法を示すサンプルコードを提供します。
CE228954 – PSoC™ 6 MCU QSPI flash read and write using SFDP	この例では、PSoC™ 6 MCU のシリアルメモリインターフェース (SMIF) ブロックを使用して、Quad-SPI モードで外部 NOR フラッシュメモリとインターフェースする方法を示します。 この例では、Serial Flash Discoverable Parameters (SFDP) 標準を使用して、フラッシュ パラメータと読み出し, プログラム, および消去操作のコマンドを自動検出します。
デバイス資料	
PSOC™ 6 MCU データシート	
PSOC™ 6 MCU テクニカル リファレンス マニュアル	
PSOC™ 6 MCU プログラミング仕様	
開発キットドキュメント	
CY8CKIT-062-BLE, PSoC™ 6 BLE Pioneer Kit	
CY8CKIT-062-WIFI-BT, PSoC™ 6 WiFi-BT Pioneer Kit	
CY8CPROTO-062-4343W, PSoC™ 6 WiFi-BT Prototyping Kit	
CY8CPROTO-063-BLE, PSoC™ 6 BLE Prototyping Kit	
ツールドキュメント	
ModusToolbox™ IDE	`<ModusToolbox install folder>/doc` を参照

改訂履歴

版数	発行日	変更内容
英語版**	-	本版は英語版のみの発行です。英語版の改訂内容: New application note.
英語版*A	-	本版は英語版のみの発行です。英語版の改訂内容: Updated to Infineon template.
英語版*B	-	本版は英語版のみの発行です。英語版の改訂内容: Updated 図 2.
**	2025-08-06	これは英語版 002-28740 Rev. *C を翻訳した日本語版 002-41854 Rev. ** です。英語版の改訂内容: Updated hyperlinks.

Trademarks

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

Edition 2025-08-06

Published by

Infineon Technologies AG

81726 Munich, Germany

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

Email: erratum@infineon.com

Document reference
IFX-ssr1649326406540

重要事項

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

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

警告事項

技術的要件に伴い、製品には危険物質が含まれる可能性があります。当該種別の詳細については、インフィニオンの最寄りの営業所までお問い合わせください。

インフィニオンの正式代表者が署名した書面を通じ、インフィニオンによる明示の承認が存在する場合を除き、インフィニオンの製品は、当該製品の障害またはその使用に関する一切の結果が、合理的に人的傷害を招く恐れのある一切の用途に使用することはできないこと予めご了承ください。