PSoC 6 MCU 割込み

About this document

Scope and purpose

本アプリケーション ノート (AN217666) は、PSoC™ 6 MCU の割込みアーキテクチャおよび PSoC Creator™, ModusToolbox™と PSoC 6 Peripheral Driver Library (PDL) API を使用したその設定について説明します。本書は割込みを使用するプロジェクトを開発する際のガイドとなります。割込みレイテンシ、コード最適化、デバッグ手法などの高度な割込みのコンセプトも説明します。

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 ライブラリはここからご覧ください。

Associated Part Family

PSoC™ 6 MCU

Table of contents

About this document **1**
Table of contents **1**
1 はじめに **3**
1.1 本書の使用方法について 3
2 PSoC 6 MCU 割込みアーキテクチャ **4**
2.1 CY8C61x6/7, CY8C62x6/7, および CY8C63xx 割込みアーキテクチャ 5
2.2 CY8C62x4, CY8C62x5, および CY8C62x8/A 割込みアーキテクチャ 6
2.3 割込みのタイプ 6
2.3.1 レベルとパルスの割込み 7
2.4 割込みと電力モード 7
2.5 CPU のスリープとウェイクアップ 8
3 割込みの設定 **10**
3.1 ModusToolbox を使用した割込みの設定 10
3.1.1 HAL の使用 10
3.1.2 デバイスコンフィギュレーターと PDL の使用 11
3.2 PDL を使用した割込みの設定 11
3.3 PSoC Creator を使用した割込みの設定 14
3.3.1 回路図 (TopDesign) の使用 14
3.3.1.1 Deep Sleep Capable (ディープ スリープ対応) 16
3.3.1.2 Interrupt Type (割込みタイプ) 16
3.3.2 Design-Wide Resource ウィンドウ (CyDWR) の使用 16
3.3.3 PSoC Creator で生成されたコードと PDL の使用 17
4 デバッグ ヒント **19**
5 高度な割込みトピック **20**
5.1 例外 20
5.2 割込みレイテンシ 21

Table of contents

5.3 ネストされた割込み 22
5.4 コード最適化 23
6 関連リソース 24
付録 A. PSoC 6 MCU の割込みソース 26
改訂履歴 29

1 はじめに

割込みは、プログラム実行を通常フローから代替の命令セットに移行させるハードウェア信号またはイベントです。割込みは、CPU が特定のイベントに対する連続的なポーリングを無くし、イベントが発生した時にのみ CPU に通知してイベントを処理させます。代替のプログラム フローは、割込みサービス ルーチン (ISR) と呼ばれます。また、ISR は割込みハンドラとも呼ばれます。割込みが処理された後、プログラム フローは中断されたフローに戻ります。PSoC などのようなシステムオンチップ (SoC) アーキテクチャでは、割込みはオンチップ ペリフェラル状態を CPU に知らせるために頻繁に使用されます。

割込みはタイマーやシリアル通信ブロック、ポート ピン信号などの CPU 外部のペリフェラルによって生成されるイベントですが、例外はメモリ アクセス フォールトや内部システム タイマー イベントなどの CPU によって生成されるイベントです。PSoC 6 MCU は、Arm® Cortex®-M4 (CM4) CPU と Arm Cortex-M0+ (CM0+) CPU の両方で割込みと例外に対応します。

1.1 本書の使用方法について

本書は PSoC 6 MCU アーキテクチャおよびサイプレス PSoC Creator™統合開発環境 (IDE) と Peripheral Driver Library (PDL) を使用した PSoC デバイスのアプリケーション開発に精通されていることを前提とします。PSoC 6 MCU の概要は **AN210781 - Getting Started with PSoC 6 MCU with Bluetooth Low Energy (Bluetooth LE) Connectivity** を参照してください。PSoC Creator, ModusToolbox または PDL を初めてお使いの方は**関連リソース**節に記載する利用可能なリソースへのリンクをご覧ください。

*Note: PSoC 6 MCU ベースの設計には、**PSoC Creator バージョン 4.2** 以降を使用してください。*

本書は、PSoC 6 MCU の割込みアーキテクチャについて簡単に説明します。PSoC 6 MCU の詳細は **PSoC 6 MCU: PSoC 63 with Bluetooth LE Architecture Technical Reference Manual (TRM)** を参照してください。割込みを使用するファームウェア作成の概要にスキップするには、**PDL を使用した割込みの設定**, **PSoC Creator を使用した割込みの設定**または **ModusToolbox を利用した割込みの設定**を参してください。さまざまなペリフェラル割込みの使用方法を示すサンプル コードは、**関連リソース**節にリストされています。

デバッグ ヒント節は、割込みを使用して遭遇した一般的な問題を見つけて解決するためのヒントを提供します。より複雑なトピックは**高度な割込みトピック**で説明しています。

2 PSoC 6 MCU 割込みアーキテクチャ

PSoC 6 MCU は、CM4 と CM0+の 2 つの CPU が含まれます。各 CPU への割込み信号は、それぞれのネスト型ベクタ割込みコントローラー (NVIC) によって処理されます。NVIC はユーザー設定に基づいて割込みを有効／無効にします。また、同時に複数の要求が起きた場合、割込みの優先順位を解決し、ネストされた割込みをサポートし、優先順位の低い ISR よりも優先順位の高い割込みを処理できるようにします。
PSoC 6 MCU は、ウェイクアップ割込みコントローラー (WIC) および複数の同期ブロックもサポートします。WIC ブロックにより、割込みを使用してスリープまたはディープスリープの低電力モードから CPU を起動できます。NVIC、プロセッサコア、およびその他のデバイス周辺機器がシャットダウンしている間、WIC ブロックはアクティブのままです。割込みがトリガされると、WIC は電源管理システムを起動し、NVIC とプロセッサコアを他の周辺機器とともに復元します。各 CPU には独立した WIC 設定があります。
ネイティブでは、CM4 は最大 240 の割込みをサポートし、CM0 +は 32 の割込みをサポートします。ユーザーが使用できる CPU 割込みの数は、デバイスによって異なります。**Table 1** を参照してください。
CM4 は、0～7 の設定可能な割込み優先順位に対応します。CM0+は 0～3 の優先順位に対応します。
PSoC 6 MCU デバイスには、最大 175 個の割込みソース (システム割込みとも呼ばれます) があります。システム割込みは、いずれかまたは両方の CPU をトリガできます。

WIC ブロックは、CPU を Deep Sleep 電力モードからウェイクアップできます。**Table 2** に、CPU をディープスリープからウェイクアップできる割込みソースを示します。
CPU マスク不可割り込み (NMI) のソースとして 1 つ以上のシステム割込みを選択できます。**Table 1** を参照してください。

Table 1 PSoC 6 MCU の割込み機能

パラメーター	CY8C61x6/7, CY8C62x6/7, CY8C63xx	CY8C62x8/A	CY8C62x5	CY8C62x4
システム割込みの数 (“N”)	147	168	174	175
ディープスリープ対応システムの割込みソースの数 (“W”)	41	39	39	45
利用可能な CM0+割込みベクタの数	32 (8 ディープスリープ対応)	8 ハードウェア (ディープスリープ対応) 8 ソフトウェアトリガ	8 ハードウェア (ディープスリープ対応) 8 ソフトウェアトリガ	8 ハードウェア (ディープスリープ対応) 8 ソフトウェアトリガ
CM0+マルチプレクサ/ベクタに接続できるシステム割込みの数	1	すべて(168)	すべて(174)	すべて(175)
利用可能な CM4 割込みベクタの数	240	240	240	240
CM4 マルチプレクサ/ベクタに接続できるシステム割込みの数	1 (1:1 マッピング)	1 (1:1 マッピング)	1 (1:1 マッピング)	1 (1:1 マッピング)
CM0+/CM4 NMI 割込みに接続できるシステム割込みの数	1	4	4	4

2.1 CY8C61x6/7, CY8C62x6/7, および CY8C63xx 割込みアーキテクチャ

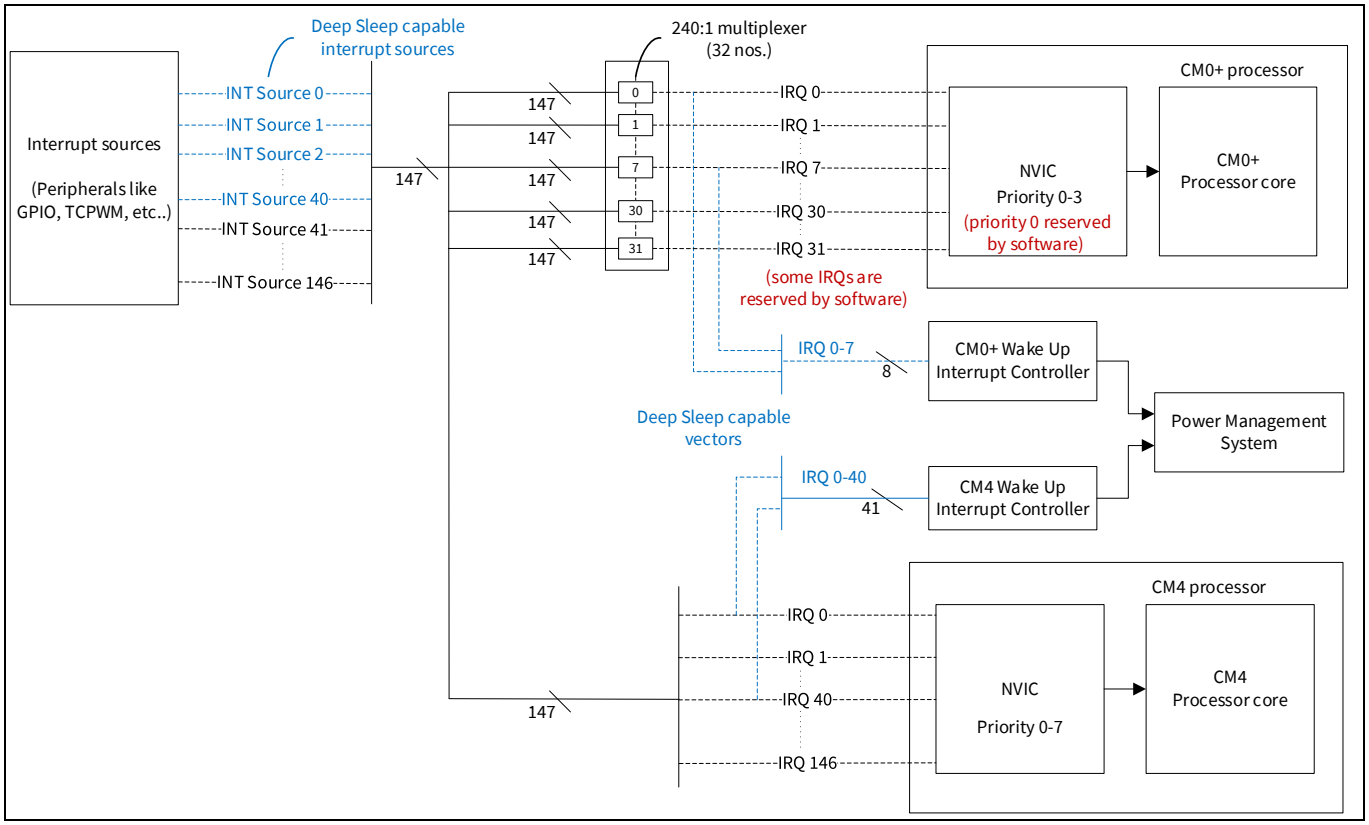


Figure 1 CY8C61x6/7, CY8C62x6/7, および CY8C63xx 割込みアーキテクチャ

CY8C61x6/7、CY8C62x6/7、および CY8C63xx デバイスは、最大 147 のシステムおよびペリフェラルの割込みソースをサポートします。

CM4 では、147 の割込みソースが最初の 147 の IRQ ラインに直接マッピングされます。つまり、INT ソース *n* は IRQ *n* に接続されます (ここで n は 0～146)。CM0+では、240:1 マルチプレクサが 32 の各 IRQ の前にあり、147 の割込みのいずれかを CM0+ IRQ ラインの 1 つにリダイレクトします。これにより、任意の割込みソースが任意の CM0+ IRQ トリガを可能にします。

2.2 CY8C62x4, CY8C62x5, および CY8C62x8/A 割込みアーキテクチャ

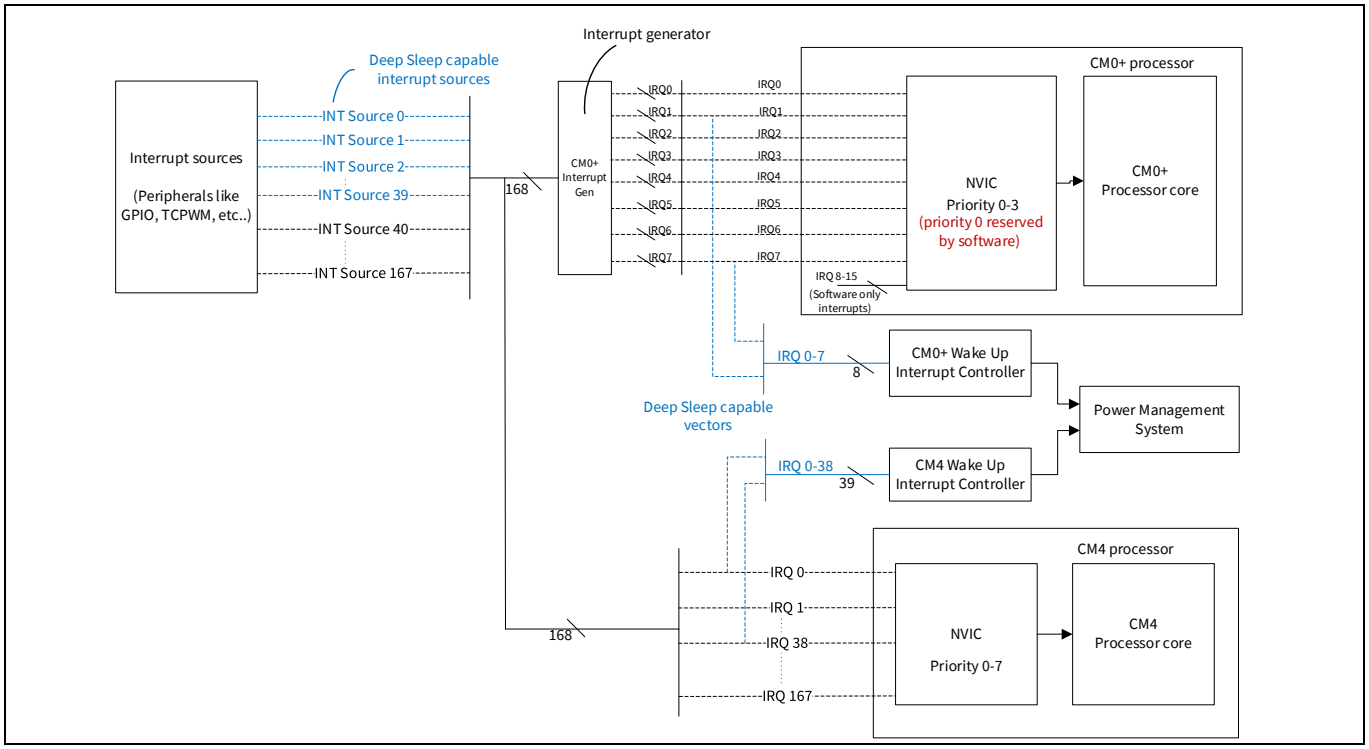


Figure 2 CY8C62x4, CY8C62x5, および CY8C62x8/A 割込みアーキテクチャ

CY8C62x4, CY8C62x5, および CY8C62x8/A デバイスでは、'N'の割込みソースが CM4 の最初の'N'の IRQ ラインに直接マッピングされます。CM0+では、16 の割込みに対応していて、そのうち最初の 8 つの割込み (IRQ 0〜7) は周辺割込みソースからトリガできます。他の 8 つはソフトウェアによってトリガできます。1 つまたは複数のシステム割込み ('N'まで) を、IRQ 0〜7 の各ラインの割込みソースとして割り当てられます。これにより、複数の割込みソースを同じ CPU 割込みに同時に接続できます。

WIC ブロックは、CPU をディープスリープ電力モードからウェイクアップできる'W'までの割込みに対応します。 **Table 1** の「ディープ スリープ対応システム割込みソースの数」を参照してください。

Table 2 に、CPU をディープスリープからウェイクアップできる割込みソースを示します。

*Note: サイプレスのソフトウェア (PDL または PSoC Creator) を使用する場合、ユーザーが利用できる CPU 割込みの数と割込みの優先度には、特定のソフトウェアの制限が適用されます。詳細は **PDL を使用した割込みの設定** 節を参照してください。*

2.3 割込みのタイプ

PSoC 6 MCU では、2 種類の割込みソースがあります。

- 固定機能割込みソース
  これらは GPIO、TCPWM、SCB、Bluetooth LE ラジオなどのオンチップ ペリフェラルからの定義済み割込みソースです。固定機能ソースからの割込みは、設定可能なイベントから生成されます。例えば、入力ピン (GPIO) の立ち上りエッジ信号の割込み、またはカウンター オーバーフロー (TCPWM) の割込みです。
- 汎用デジタル ブロック (UDB) 割込みソース
  UDB はプログラマブル ロジック デバイス (PLD)、データパスおよび柔軟なルーティングで構成されて

おり、タイマー、PWM、UART、SPI などのさまざまなデジタル機能を合成するために使用できます。固定機能の割込みソースとは対照的に、UDB で生成されたいかなるデジタル信号も割込みをトリガできます。この信号は、デジタル システム相互接続 (DSI) と呼ばれるルーティング ファブリックを経由し、割込みコントローラーにルーティングされます。UDB ソースは、CY8C61x6/7, CY8C62x6/7, および CY8C63xx デバイスでのみ使用できます。

PSoC 6 MCU の割込みソース一覧は、**付録 A** を参照してください。

2.3.1 レベルとパルスの割込み

CM0+および CM4 の NVIC は、IRQ ライン上のレベルとパルスの信号に対応します。レベルまたはパルスとしての割込み分類は割込みソースに依存します。固定機能割込みは、レベル依存として扱われます。UDB を含む DSI ソースの場合、割込みは立ち上りエッジ トリガまたはレベル トリガのいずれかに設定できます。この設定は PSoC Creator でのみ利用可能です。割込みタイプ選択の詳細は、PSoC Creator コンポーネントのデータシートまたは割込みソースの PDL API リファレンスを参照してください。

レベル割込みの場合、ISR の完了後も割込み信号がまだ HIGH であれば、割込みは保留され続け、ISR が再び実行されます。**Figure 3** にレベル トリガ割込みのタイミング図を示します。ここでは、割込み信号が HIGH の場合は ISR が実行されます。

パルス割込みの場合、ISR が CPU によって実行されている間、割込み信号の 1 つ以上の立ち上りエッジが単一の保留中要求として記録されます。保留中の割込みは、現在の ISR 実行の完了後に再び処理されます。**Figure 4** にパルス割込みのタイミング図を示します。

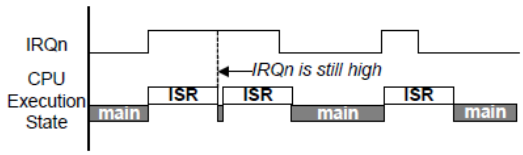


Figure 3 レベル割込み

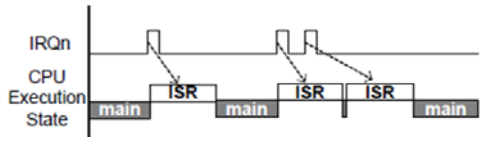


Figure 4 パルス割込み

Note: *GPIO 割込みロジックは、立ち上りエッジ、立ち下りエッジおよび両エッジの割込みに対応するための追加の回路を持ちます。詳細は **PSoC 6 MCU Architecture TRM** の I/O System 章を参照してください。*

2.4 割込みと電力モード

PSoC 6 MCU には、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のシステム電力モードがあります。Arm CPU 電力モードは、アクティブ、スリープ、およびディープスリープです。これらは、システム LP および ULP 電力モードで使用できます。

CPU アクティブモードでは、CPU はコードを実行し、すべてのメモリ ブロックとペリフェラルが使用できます。

他のすべての電力モード (スリープ、ディープスリープ、ハイバネート) では、CPU クロックがオフになり、コードの実行が停止します。

アクティブモードで使用可能なすべてのペリフェラルは、スリープ、ディープスリープ、ハイバネートでも使用できます。CPU にマスクされているペリフェラル割込みは、CPU をアクティブモードにウェイクアップします。

ディープ スリープ モードでは、ペリフェラルのサブセットのみが動作します。これらのペリフェラルからの割込みは、CPU をアクティブ モードに復帰させます。**Table 2** にそれらのペリフェラルの一覧を示します。各 CPU には、CPU をディープ スリープ モードから復帰させるウェイクアップ割込みコントローラー (WIC) があります。ディープ スリープ ウェイクアップ機能は、CM0+の最初の 8 つの IRQ (0〜7) と、CM4 の最初の'W'の IRQ でのみサポートされます。**Table 1** を参照してください。

ハイバネート モード中は、すべてのペリフェラルとクロックがオフになり、低消費電力コンパレータ、RTC、専用 WAKEUP ピンや XRES イベントなどの特定のソースのみがデバイスをウェイクアップできます。ウェイクアップ動作は、CPU への割込みではなくデバイス リセットです。

デバイスの電力モードの CPU スリープと割込みによるウェイクアップ動作の詳細は、**AN219528 - PSoC 6 MCU Low-Power Modes and Power Reduction Techniques** または **PSoC 6 MCU Architecture TRM** を参照してください。

Table 2　　ディープ スリープ復帰可能割込み一覧

割込みソース	割込みソース番号			
	CY8C63xx	CY8C62x6/7 CY8C61x6/7	CY8C62x8/A CY8C62x5	CY8C62x4
GPIO ポート割込み	0–14	0–14	0–14	0–14
全ポートの GPIO	15	15	15	15
GPIO 電源検出割込み	16	16	16	16
低消費電力コンパレータ割込み	17	17	17	17
シリアル通信ブロック割込み	18	18	18	18
マルチカウンターウォッチドッグタイマー	19, 20	19, 20	19, 20	19, 20
バックアップ ドメイン割込み	21	21	21	21
SRSS 用のその他の組み合わされた割込み	22	22	22	22
組合せ連続時間ブロック (CTBm) 割込み	23	23	–	–
Bluetooth 無線割込み	24	–	–	–
プロセッサ間通信割込み	25–40	25–40	23-38	23-38
SAR ADC 割込み	–	–	–	39, 40
個別連続時間ブロック (CTBm) 割込み	–	–	–	41
PASS タイマー割込み	–	–	–	42
PASS FIFO 割込み	–	–	–	43, 44

2.5　　CPU のスリープとウェイクアップ

CPU がスリープ モードに入るためには、「Wait-for-Interrupt」 [__WFI()] と「Wait-for-Event」 [__WFE()] の 2 つ命令があります。WFI 命令が実行されたとき、CPU はスリープまたはディープ スリープ (SCR レジスタの SLEEPDEEP ビットに依存) に入り、割込み要求 (現在の優先順位より高い優先順位) またはデバッグ要求でウェイクアップします。WFE 命令は WFI に似ていますが、次の割込み *または* イベント送信 (SEV 命令)、外部イベントやデバッグ信号などのイベントでウェイクアップします。スリープおよびウェイクアップ命令の詳細は **AN219528** を参照してください。

PSoC 6 MCU 割込みアーキテクチャ

通常、ISR の実行が終わると、CPU の実行はその ISR の前のところに戻ります。PSoC 6 MCU は、ISR 実行が完了すると直ちに CPU がスリープまたはディープ スリープ (WFI に類似した状態) に入るか、復帰する「Sleep-on-Exit」(スリープオンイグジット) 機能をサポートします。**Figure 5** に示すように、この機能を有効にした後、スリープ モードに入るためには 1 つの WFI 命令だけが必要です。CPU は各 ISR 実行後 main に戻る代わりにスリープに戻ります。Sleep-on-Exit 機能は、CPU のアクティブ サイクルを減少させ、割込み間の処理のスタッキング (スタックへプッシュ) およびアンスタッキング (スタックからポップ) によって消費されるエネルギーを削減します。**ネストされた割込み**は、Sleep-on-Exit が有効な場合にもサポートされます。

Sleep-on-Exit 機能は、SCR レジスタの SLEEPONEXIT ビットを設定することにより有効になります。利用可能な PDL 機能 (`Cy_SysPm_SleepOnExit`) もあります。詳細は **PDL を使用した割込みの設定**を参照してください。

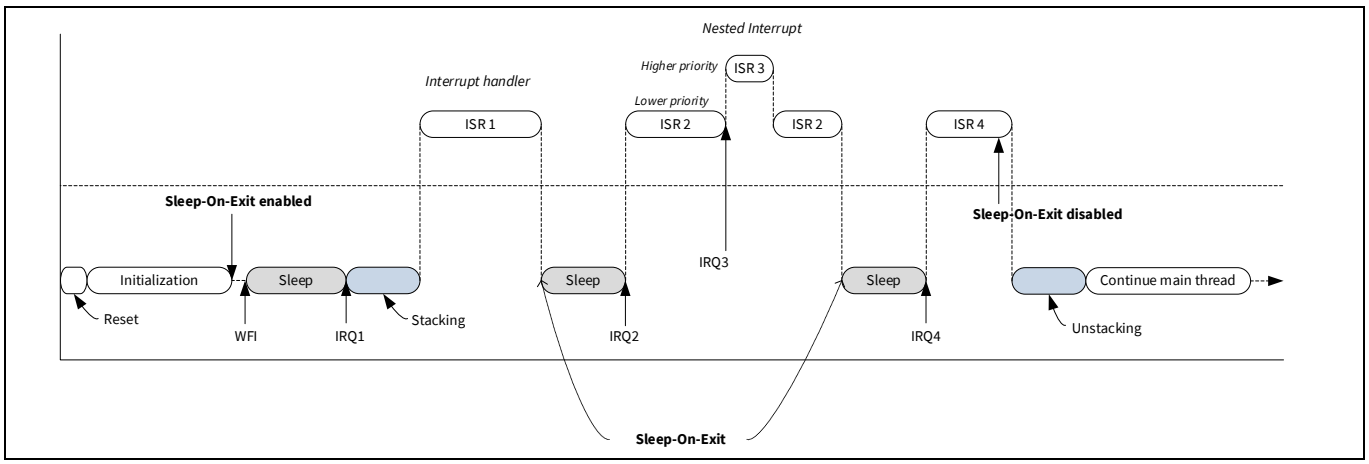


Figure 5　　Sleep-on-Exit 機能

3 割込みの設定

ここでは、PSoC 6 MCU デバイスに割込みを設定するために必要なステップを示します。なお、この説明では使用されるソフトウェアの詳細については触れません。これらのステップは、特に明記されない限り、CM0+と CM4 の両方に共通であり、また CPU ごとに個別に実行する必要があります。

- デバイスをリセットすると、すべての割込みは無効になり、割込み優先順位はゼロに設定されます。
- NVIC で必要な IRQ の優先レベルを設定してください。
- 割込みパスを設定してください。
- CPU での必要な IRQ に接続する割込みソースを選択してください。CM0+では、CPU に接続する適切なペリフェラル割込みを選択するために割込みマルチプレクサの 1 つを設定します。CM4 の場合、これは設定不可です。割込みソース *n* は常に *IRQn* に接続されます。
- 割込みソース (ペリフェラル) を設定し、その割込みを有効にしてください。
- ISR (ベクタ) のアドレスをベクタ テーブルに設定してください。ベクタ テーブルには各例外ハンドラのエントリ アドレスを格納します。詳細は **PSoC 6 MCU Architecture TRM** の Interrupts 章の Exception Vector Table を参照してください。
- 任意で、NVIC の保留中の割込みステータスをクリアします。
- 以前に無効化された割込みを有効にする場合、その割込みを有効にする前に NVIC の保留状態をクリアすることを推奨します。これは、保留状態にした以前の割込みによって引き起こされた誤ったトリガを抑止します。
- NVIC での割込みを有効にしてください。
- グローバル割込みを有効にしてください。以上で割込み設定が完了です。

有効になった割込みは、割込みソースからのハードウェア信号がアクティブであり、実行中の優先順位の高い割込みがない時にトリガされます。これが起きると、CPU の実行はトリガされた割込みに対応するベクタ テーブルに格納される位置にジャンプします。この位置には、その割込みに関連する ISR のアドレスが格納されます。

ISR は割込みを処理するために必要なタスクを実行します。通常、ISR が最初にすることは、その ISR の再入を避けるために割込みソースをクリアすることです。ISR が完了すると、CPU は中断される前に実行していたアドレスに戻ります。次の節では、これらのステップを実行するために利用できるソフトウェア ツールについて説明します。

3.1 ModusToolbox を使用した割込みの設定

ModusToolbox アプリケーションは、PSoC 6 **Hardware Abstraction Layer** (HAL) および **Peripheral Driver Library** (PDL) ライブラリの両方をサポートします。

3.1.1 HAL の使用

HAL は、サイプレス MCU のさまざまなブロックを設定して使用するための抽象化されたインターフェースを提供します。HAL には、割込み用の個別のブロックはありません。さまざまなブロックの割込みは、それらのブロックに固有の HAL API を使用して設定されます。

例えば、GPIO 割込みの場合、GPIO ブロックから発生する割込みは、GPIO HAL API を使用して設定されます。この例の GPIO HAL ブロックを設定する手順は次のとおりです。

- GPIO ピンの初期化: GPIO ピン、方向、ドライブ モード、および初期値が `cyhal_gpio_init` HAL API 関数に渡されます。
- 割込みコールバック関数の登録: 割込みのコールバック関数は、`cyhal_gpio_register_callback` を使用して GPIO ピンに登録されます。

割込みの設定

- 割込みの設定: GPIO イベントや割込み優先度などの割込み設定は、`cyhal_gpio_enable_event` を使用して設定されます。
- *GPIO HAL API Reference Guide* の **Interrupts on GPIO event code snippet** を参照してください。

HAL API の詳細については、**PSoC 6 HAL on GitHub** を参照してください。

3.1.2 デバイスコンフィギュレーターと PDL の使用

割込みは、MCU ペリフェラルとその割込みパラメーターを有効にして設定するために使用される GUI ベースのツールである **デバイスコンフィギュレーター**を使用して ModusToolbox で設定できます。

Figure 6 に、立ち下りエッジで割込みを生成する GPIO ピンの設定を示します。

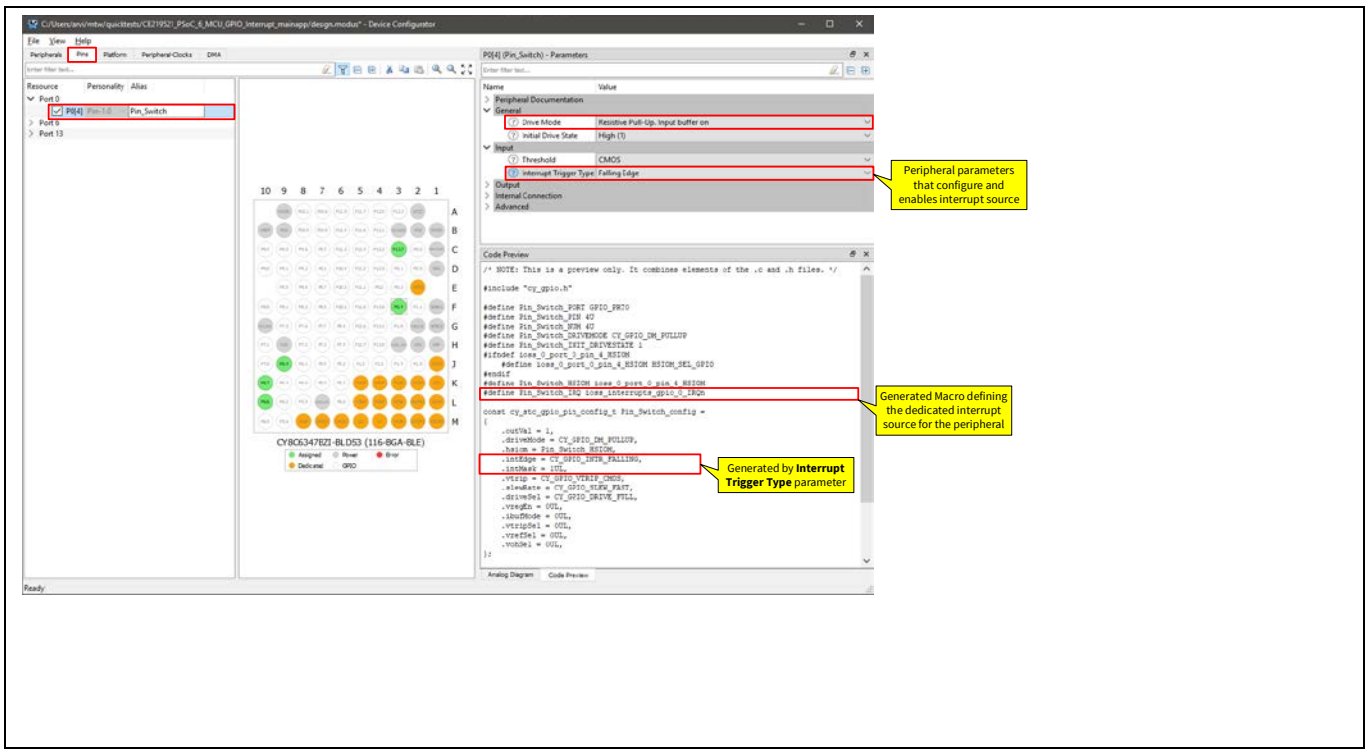


Figure 6 ModusToolbox ペリフェラル設定

設定に基づいて、ModusToolbox は必要な設定を実現するための'C'コードを生成します。生成されたコードは、Code Preview ペインで表示できます。ModusToolbox アプリケーションの *TARGET_<BSP name>/COMPONENT_BSP_DESIGN_MODUS/GeneratedSource* フォルダ内に関連する *cycfg_xxx.c/h* ファイルが追加されます。生成されたコードには、割込みソース番号を定義するマクロと、割込みソースを設定して有効にするために必要な周辺機器構成が含まれます。これにより、デバイス ヘッダ ファイルで専用の割込み番号を検索するプロセスが簡素化されます。ユーザー アプリケーションは、**PDL を使用した割込みの設定**で説明されるように、CPU で割込みベクトルを有効にし、割込みハンドラ関数を割り当てるだけです。

3.2 PDL を使用した割込みの設定

ペリフェラル ドライバー ライブラリ (PDL) は、PSoC 6 MCU アーキテクチャのソフトウェア開発を簡素化します。PDL を使用すると、レジスタの使用法とビット構造を理解する必要がなくなり、利用可能なペリフェラルの広範なセットのソフトウェア開発が容易になります。

割込みの設定

Note: *PDL の ModusToolbox ソフトウェア バージョンは、**サイプレスの GitHub** サイトで入手できます。PSoC Creator と互換性がありません。PDL の ModusToolbox バージョンには、新しい PSoC 6 MCU デバイスとドライバーのサポートが含まれます。また、Windows だけでなく、macOS および Linux ホストもサポートします。開発者は、プロジェクトとスケジュールが許す限り、ModusToolbox パッケージに移行する必要があります。**PDL v3.1** は PSoC Creator 向けに設計されており、PSoC Creator と連携して動作します。PDL v3.1 は、最終的な PSoC Creator 互換リリースになる予定です。PDL v3.0.x は、**PSoC Creator 4.2** 開発ツールとともにインストールされます。*

PDL API 関数呼び出しは、ペリフェラルドライバーの設定、初期化、有効化、および使用に使用されます。そのようなドライバーの 1 つに、システム割込み (SysInt) があります。SysInt は、割込み機能を設定して有効にするための構造と関数を提供します。PDL は、割込み設定に使用される **NVIC 関数**を含む **CMSIS-Core** ライブラリもサポートします。

以下の手順では、PDL および NVIC API を使用し、ペリフェラルからの信号でトリガする割込みを設定します。

- 割込みが生成されるようにペリフェラルを設定します。例えば、GPIO の場合、ドライブ モード (プルアップまたはプルダウン)、立ち下りエッジまたは立ち上りエッジでの割込み信号生成、および割込みのマスク解除を設定します。この情報については、使用されるペリフェラルの PDL API リファレンス文書を参照してください。
- SysInt API によって提供される構造体を使用して割込みを設定します。

この構造体は、PDL SysInt ドライバー ファイル *cy_sysint.h* で定義されます。

```
* Initialization configuration structure for a single interrupt channel */
typedef struct {
    IRQn_Type       intrSrc;    /**< Interrupt source */
#if (CY_CPU_CORTEX_M0P)
    cy_en_intr_t    cm0pSrc;    /**< (CM0+ only) Maps cm0pSrc device
interrupts to intrSrc */
#endif
    uint32_t        intrPriority;   /**< Interrupt priority number (Refer to
__NVIC_PRIO_BITS) */
} cy_stc_sysint_t;
```

この構造体は、以下を設定するために使用されます (簡単な要約は、**Figure 7** を参照してください)。

- 割込みソース (intrSrc)
  - これらは、デバイス ヘッダ ファイル (例: *cy8c6247bzi_d44.h*) で定義される専用の割込み番号です。
  - この選択は、割込みを割り当てる CPU に依存します。
    - CM4 の場合、この番号はソースの割込み番号と CPU IRQ 番号の両方を表します。CPU にルーティングするペリフェラル割込みの割込み番号を選択します。例えば、ポート 0 GPIO 割込みをルーティングするためには、ioss_interrupts_gpio_0_IRQn (=0) の値を割り当てます。
    - CM0+の場合、この番号は割込みを CM0+にルーティングするために使用できる 32 のマルチプレクサの 1 つを表します。各マルチプレクサは専用の CM0+ IRQ ラインに接続されているため、これを使用して対象の CM0+ IRQ 番号を選択します。例えば、マルチプレクサ#4 (CM0+ IRQ#4) を使用するためには、「NvicMux4_IRQn」 (=4) を使用します。

割込みの設定

- CM0+割込み番号 (cm0pSrc)
  - このパラメーターは CM0+にのみ適用可能です。
  - これは intrSrc パラメーターを使用して選択されたマルチプレクサ/CM0+割込み生成ロジックにルーティングされるソースの割込み番号を表します。CPU にルーティングしたいペリフェラル割込みの割込み番号を選択します。例えば、ポート 0 の GPIO 割込みをルーティングするためには、「ioss_interrupts_gpio_0_IRQn」 (=0) の値を割り当てます。
- 割込み優先順位 (intrPriority)
  - 割込みの優先順位を設定します。CM4 の場合、サポートされる優先順位は 0～7 です。CM0+の場合は、0～3 です。

Note:

1. *CM0+ では、一部の IRQ はソフトウェアで使用するために予約され、ユーザーは使用できません。予約された IRQ の一覧は、PDL API リファレンス文書の SysInt ドライバーにある「Configuration Considerations」を参照してください。*
2. *CM0+ では、割込み優先順位 0 はシステム コール用に予約されています。*

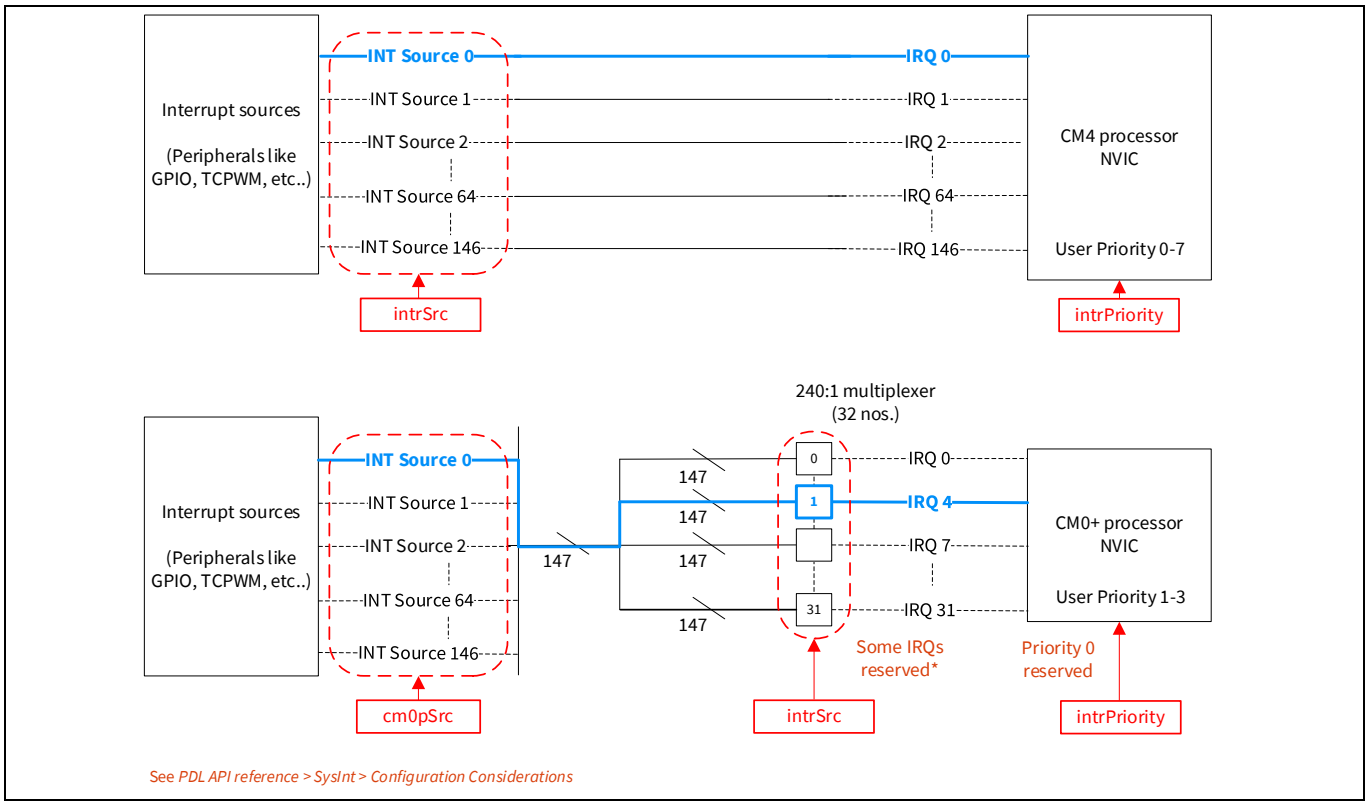
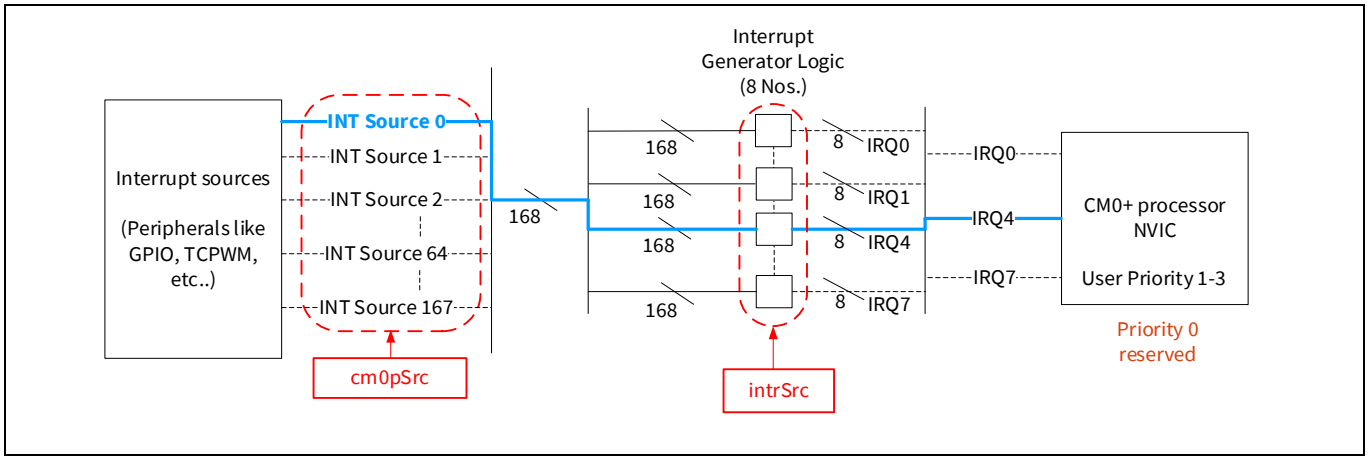


Figure 7　　割込み設定に使用される SysInt PDL 構造体パラメーター (赤でハイライト) (設定されたパスの例は青でハイライト)

CY8C62x4、CY8C62x5 および CY8C62x8/A では、以下のとおり。



- Cy_SysInt_Init(&SysInt_SW_cfg_1, ISR_1_handler) を呼び出します。
- ここで、`SysInt_SW_cfg_1` は構成された構造体の名前です。`ISR_1_handler` は、割込みがトリガされた時に実行される割込みハンドラの名前です。この関数は割込みのルーティングと優先順位の設定をしますが、割込みを有効にしません。
- 保留中の割込みをクリアするために、NVIC_ClearPendingIRQ(SysInt_SW_cfg_1.intrSrc) を呼び出します。
- 割込みを有効にするために、NVIC_EnableIRQ(SysInt_SW_cfg_1.intrSrc) を呼び出します。
- グローバル割込みを有効にするために、`__enable_irq()` 関数を呼び出します。これは、それぞれの CPU 割込みがまだ有効になっていないため、最初のステップとして実行することが安全です。これは後で実行することもできますが、これが呼び出されない限り、スタートアップ時に割込みは無効になります。

PDL SysInt ドライバーに加えて、システム電力モード (SysPm) ドライバーAPI はスリープオンイグジット機能を有効にします。割込みを使用するアプリケーションで、スリープまたはディープ スリープ モードが使用される場合、この機能により、ファームウェアはシステムをほぼ常にスリープ モードに保ち、ウェイクアップして割込みを実行してから直ちに同じスリープ モードに戻ります。スリープオンイグジット機能が再び無効にされない限り、プログラムは main 関数に戻らず、割込みハンドラまたは同じスリープ状態のいずれかになります。

```
Cy_SysPm_SleepOnExit(true);
```

3.3 PSoC Creator を使用した割込みの設定

PSoC Creator は、ペリフェラルからの信号を CPU IRQ ラインにルーティングするためのグラフィカル インターフェースを提供します。PSoC Creator は、割込み (SysInt) コンポーネントを提供します。このコンポーネントは、前節で説明した SysInt PDL ドライバーのトップにある UI 要素です。コンポーネントの設定に基づき、PSoC Creator はペリフェラルの初期化、割込みのルーティングおよび割込みコンフィギュレーション構造体を構成するためのコードを生成します。これにより、割込みを設定する時に書かなければならないコードの量が減ります。

次の節では、PSoC Creator を使用して割込みを設定するステップを示します。サンプル コードは、**関連リソース**を参照してください。

3.3.1 回路図 (TopDesign) の使用

コンポーネントを Component Catalog から TopDesign にドラッグアンドドロップします。TopDesign を使用し、割込みソースを提供するペリフェラルを配置して設定します。特定のペリフェラルの割込み設

割込みの設定

定には、コンポーネントのデータシートを参照してください。TCPWM などのいくつかのペリフェラルは、割込み端子を提供します。SysInt コンポーネントのインスタンスを配置し、ペリフェラルの割込み端子に接続します。

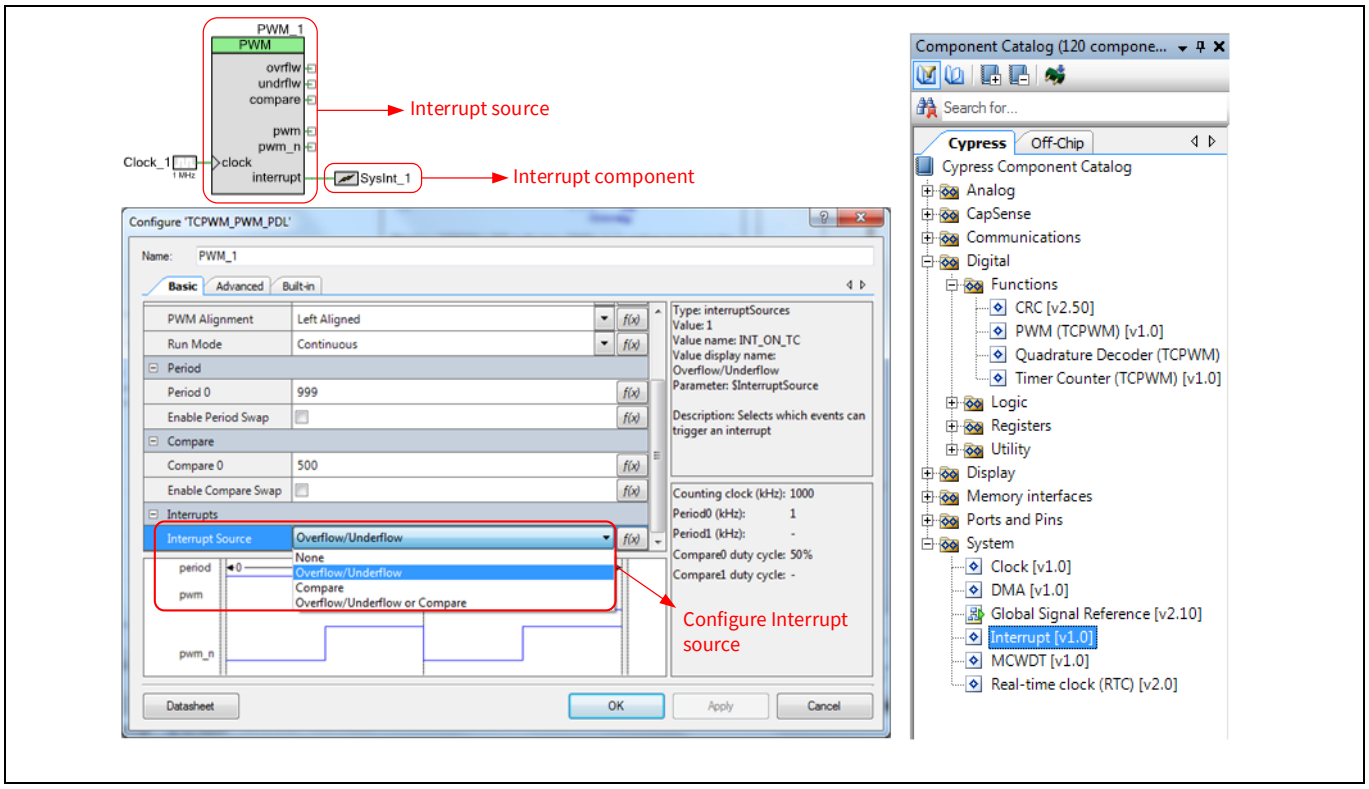


Figure 8　　割込みコンポーネントの TopDesign

一部のペリフェラルは、外部割込み端子がないもの (例えば、SCB に割込みが組み込まれている) や、それを公開するオプション (例えば、UART) などがあります。

Figure 9 に示すように、割込みコンポーネントには 2 つの設定可能なオプションがあります。

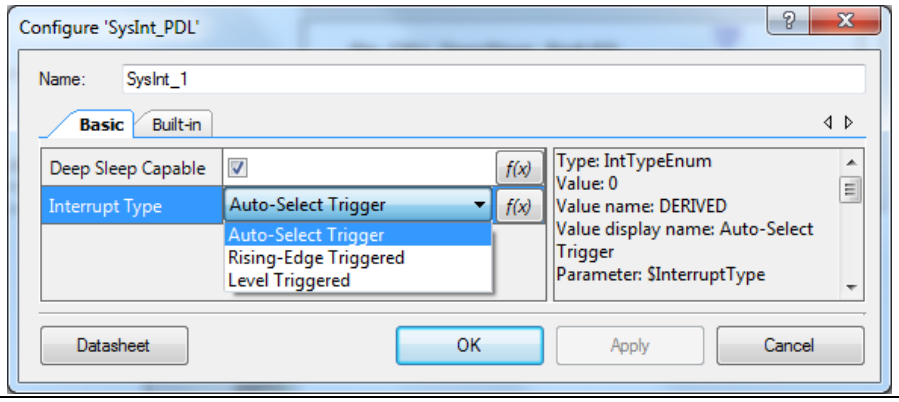


Figure 9　　SysInt (割込み) コンフィギュレーション

割込みの設定

3.3.1.1 Deep Sleep Capable (ディープ スリープ対応)

ディープ スリープ対応の CPU IRQ ラインに割込みを割り当てる場合は、このチェックボックスを有効にします。割込みソースもアクティブで、ディープ スリープ中に割込み信号を提供できることを確認する必要があります。失敗すると、プロジェクトのビルド時に PSoC Creator がエラーを表示します。このオプションは、8 つ (IRQ 0～7) のディープ スリープ スロットを持つ CM0+に割込みが割り当てられている場合にのみ有効であることに注意してください。このチェックボックスは、自動的に割込み IRQ を割り当てるためのガイダンスとしてのみ提供され、**CyDWR ウィンドウ**からの手動割当てによって無効にできます。CM4 では、割込みソースがディープ スリープ対応 (IRQ 0～40) の場合、チェックボックスを無効にしても、割込みのディープ スリープ機能には影響しません。

3.3.1.2 Interrupt Type (割込みタイプ)

割込みコンポーネントの設定では、Auto-Select Trigger (自動選択トリガ)、Rising-Edge Triggered (立ち上りエッジ トリガ)、Level Triggered (レベル トリガ) の 3 つの割込みタイプ オプションがあります。特定のオプション選択は、割込みソース (固定機能または UDB/DSI) とアプリケーション要件に依存します。ほとんどのケースでオプションを自動選択にしておくと、PSoC Creator が割込みソースの特性から割込みタイプを派生できます。

固定機能割込みソースに対しては、レベル トリガのみを選択します。UDB ソースに対しては、レベル トリガまたは立ち上りエッジを選択します。

3.3.2 Design-Wide Resource ウィンドウ (CyDWR) の使用

PSoC Creator プロジェクトの Design-Wide Resource ウィンドウ (.cydwr ファイル) には、「Interrupts」タブがあります。このタブには、TopDesign 回路図で使用されるすべての割込みのインスタンス名およびそれらの割込み番号がリストされます。

各割込みは、「Arm CMx Enable」チェックボックスを使用し、CM0+または CM4 または両方の CPU に割り当てられます。特に指定しない限り、すべての割込みはデフォールトで CM4 に割り当てられます。**アプリケーションに必要にならない限り、両方の CPU に割込みを割り当てることは推奨しません。両方の CPU が同じ割込み処理をする場合、インスタンス名のコラムに警告アイコンが表示されます。**アイコンの上にマウス ポインターを置くと、警告のツール ヒント説明が表示されます。

CM0+では、「Arm CM0+ Vector」コラムを使用し、CPU IRQ ラインを割り当てます。**一部の CM0+ IRQ は予約されている**ことに注意してください。PSoC Creator はこれらの IRQ への割当てを許可しないため、警告が表示されます。CM4 のベクタを選択するオプションは、対応する割込み番号に直接マッピングされるため、ありません。

CPU に割り当てた後、対応する優先順位フィールドを使用して優先順位を割り当てます。CM0+の優先順位は 1～3 です (**優先順位 0 はシステム コール用に予約済み**)。CM4 の優先順位は 0～7 です。両方の CPU で優先順位 0 が最高の優先順位に対応し、大きい数値は優先順位が低いことを示します。

ディープ スリープ対応の割込みソースまたは IRQ はアイコンで示されます。情報アイコンは、ディープ スリープ非対応の割込みがディープ スリープ対応の IRQ ラインに割り当てられている場合に表示されます。割込み番号とアイコンを更新するためには、ビルドが必要です。

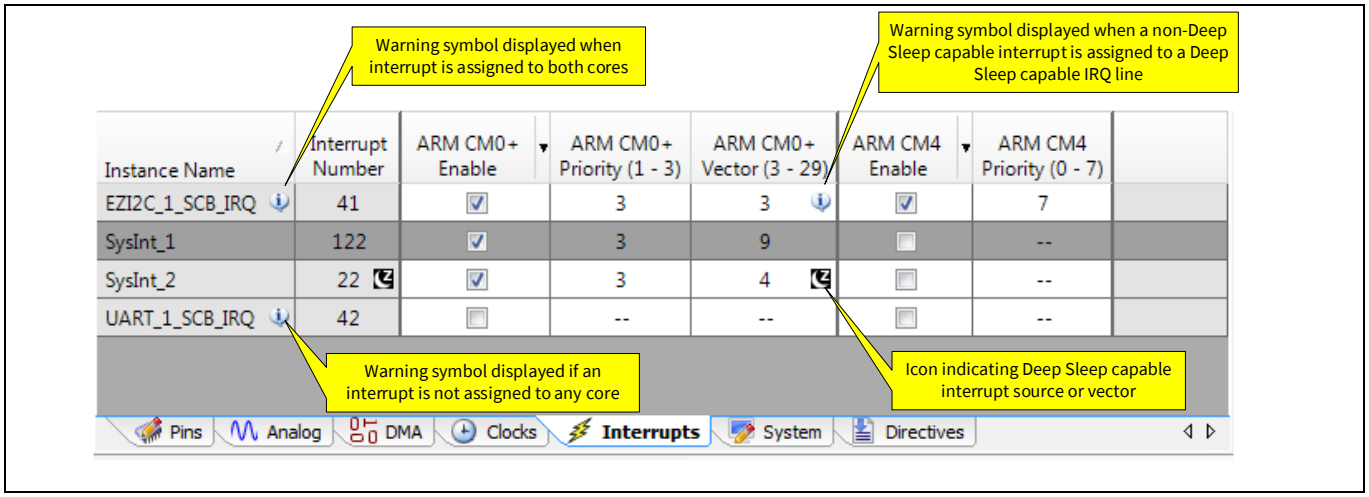


Figure 10　　CyDWR での割込み割当て

3.3.3　PSoC Creator で生成されたコードと PDL の使用

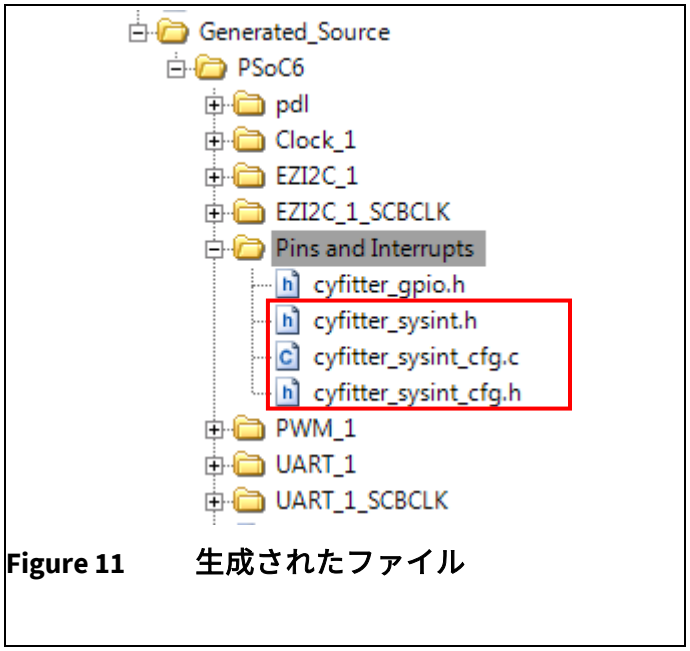


Figure 11　　生成されたファイル

プロジェクトをビルドすると、アプリケーションで使用するコードが生成されます。**Pins and Interrupts** フォルダには、CyDWR の「Interrupts」タブで入力された情報を使用し生成されたコードが含まれます。

Cyfitter_sysint.h には、割込み番号、CPU 割当ておよび優先順位に関する情報を含むマクロが含まれます。

Cyfitter_sysint_cfg.c/h は、CyDWR 情報を使用し SysInt PDL コンフィギュレーション構造体のインスタンスを宣言して事前に構成します。

各割込みのコンフィギュレーション構造体は、CPU 割当てに基づき、条件付きで定義されます。

ファームウェアで割込みを有効にする手順は、**PDL の節**にあるものと似ていますが、数は少ないです。

- グローバル割込みを有効にするために、`__enable_irq()` API を呼び出してください。
- `Cy_SysInt_Init(&SysInt_1_cfg, ISR_1_handler)` を呼び出してください。
  - ここで、`SysInt_1_cfg` は *cyfitter_sysint_cfg.c* ファイルから自動生成された構造体の名前です。`ISR_1_handler` は、割込みがトリガされた時に実行される割込みハンドラの名前です。ハンドラ関数は、割込みが割り当てられているそれぞれの CPU の *main.c* に配置できます。ハンドラが *main.c* の外部にある場合、そのファイルをコンパイルし、ISR を処理する CPU の実行ファイルにリンクする必要があります。
  - このステップでは、割込みの設定 (ルーティング、優先順位、割込みハンドラの割当て) をしますが、有効にはしません。
- 保留中の割込みをクリアするために、NVIC_ClearPendingIRQ(SysInt_1_cfg.intrSrc) を呼び出してください。

割込みの設定

- 割込みを有効にするために NVIC_EnableIRQ(SysInt_1_cfg.intrSrc) を呼び出してください。

PSoC Creator でコードを生成し、そのコードを好む IDE にインポートできます。その方法は **AN219434 – Importing PSoC Creator Code into an IDE for a PSoC 6 MCU Project** に説明されています。PSoC Creator を使用し、PSoC 6 MCU の割込みを設定および構成し、そのプロジェクトを好む IDE にエクスポートして、その IDE でファームウェア コードを開発し続けることを推奨します。

4 デバッグ ヒント

ここでは、トラブルシューティングと割込みのデバッグに関するヒントを提供します。高い頻度で遭遇するケースについて、いくつかを以下に示します。

- 割込みがトリガされない
  - 割込みソースとグローバル割込みが有効になっていることを確認します。
  - 割込みベクタが正しい ISR で初期化されていることを確認します。
  - 他の割込みソースが繰り返しトリガされていないか確認します。この場合、CPU 帯域幅全体が消費されます。
- 割込みが繰り返しトリガされる
  これは、複数のケースで起きることがあります。繰り返し実行されると予想されるプログラムの ISR および他の場所 (例えば、main 関数のスーパーループ) にブレークポイントを挿入します。プログラムが main 関数に入らない場合、割込みは繰り返しトリガされます。
  - 固定機能ソースからの割込みラインは、レベル タイプに設定された SysInt コンポーネントに接続されます。
    解決策: この問題を解決するために、割込みソースをクリアします。
  - コンポーネント (割込みラインではない) からのデジタル出力は、PSoC Creator のレベル タイプに設定された SysInt コンポーネントに接続されます。
    解決策: 割込みコンポーネントを立ち上りエッジに設定し、立ち上りエッジごとに 1 つの割込みを取得します。
- ISR 実行に予想以上の時間がかかる
  これは、ISR の実行中に他の優先順位の高い割込みがトリガされた場合に発生します。
  解決策: 他の割込みソースと比較し、割込みの優先順位を上げます。

PSoC 6 Bluetooth LE Pioneer Kit は、KitProg2 オンボード プログラマ／デバッガが備わっています。

CY8CPROTO-062-4343W PSoC 6 Wi-Fi BT Prototyping Kit には、KitProg3 オンボードプログラマー/デバッガが含まれます。このキットは ModusToolbox でのみサポートされます。

PSoC Creator は、一度に 1 つの CPU (CM0+または CM4) のデバッグに対応します。 ModusToolbox IDE は、両方の CPU のデバッグを同時にサポートします。

デバッグ モードは下記の割込みをチェックするのに便利です。

- 割込みが実行されているかどうかを確認するために、ISR のいずれかの命令にブレークポイントを追加します。
- Breakpoint Hit Count (ブレークポイント ヒット カウント) またはブレークポイント条件を使用し、割込みがトリガされた回数を検出します。これは、割込み信号が何度もトリガを引き起こすグリッチがあるかどうかを確認するために特に便利です。ブレークポイント ヒット カウントを表示するためには、ブレークポイントを右クリックし、Hit Count を選択して現在のヒット カウントを観察します。
- デバッガのコール スタック ウィンドウを使用し、特定の ISR が実行される時点が分かるためにプログラム フローを確認します。また、これは優先順位の低い ISR の実行中に優先順位の高い割込みが発生したかどうかを確認するためにも使用できます。
- デバッガの代わりに、ピンを使用して以下ができます。
- CPU が ISR に入っているかどうか確認します。
- ISR の実行時間を測定します。例えば、これは、ISR の先頭でピンをアサートし、ISR から戻る前にピンをデアサートすることによって行えます。オシロスコープを使用してピンが HIGH である時間を測定することで、ISR 実行時間が分かります。

5 高度な割込みトピック

5.1 例外

例外は、プロセッサに現在実行中のコードを中断させ、ハンドラに分岐させるイベントです。割込みは例外のサブセットです。割込みの他に、オペレーティング システム アプリケーションおよびフォールト処理による例外があります。

例外	例外番号	例外優先順位	例外をサポートする CPU	説明
リセット	1	-3	CM0+および CM4	この例外は、パワーオンリセット (POR)、XRES ピンによる外部リセット信号またはウォッチドッグ リセットなど複数の原因で発生。 Cortex-M4 の実行は、CM0+が M4 のリセットをデアサートした後にのみ開始される。 デバイスがリセットから解除された時点でフラッシュ ベクタ テーブルが選択されるため、SRAM ベクタ テーブルのリセット例外アドレスが使用されることはない。リセットがアサート解除された後、SRAM ベクタ テーブルを選択するためのレジスタ コンフィギュレーションはフラッシュのスタートアップ コードの一部として実行される。
マスク不可割込み (NMI)	2	-2	CM0+および CM4	両方のコアはそれぞれ独自の NMI 例外がある。NMI は以下によりトリガされる。いずれかの割込みソース、NMIPENDSET ビットのセット、またはシステム コールの使用。 PSoC 6 Bluetooth LE では、NMI のソースとしてシステム割込みを 1 つだけ、そのルーティングをサポートします。 CY8C62x8/A は、NMI の 4 つのシステム割込みソースをサポートします。 選択した 4 つの割込みソースは、単一の CPU NMI 入力に論理和されます。 NMI 例外ハンドラ アドレスは、ブート コードによって SROM (0x0000000D) にあるシステム コール API に自動的に初期化される。この値はベクタ テーブルの再配置中にユーザーが保持する必要がある。そうしない場合システム コールは実行されない。
HardFault 例外	3	-1	CM0+および CM4	HardFault 例外は、未定義命令の実行時または無効なメモリ アドレスへのアクセス時に発生

例外	例外番号	例外優先順位	例外をサポートする CPU	説明
SVCall 例外	11	設定可能	CM0+および CM4	スーパバイザ コール (SVCall) は CPU が SVC 命令をアプリケーションコードの一部として実行することにより発生する常に有効な例外。SVC 命令はアプリケーションがシステムへの特権アクセスを要するスーパバイザコールを発行することを可能にする。
PendSV	14	設定可能	CM0+および CM4	PendSV 例外は、通常はソフトウェアで生成される。PendSV は、SVCal と同様のスーパバイザ コール関連の例外。
SysTick 例外	15	設定可能	CM0+および CM4	SysTick は、定期的な割込みを生成する 24 ビット デクリメント カウンター。
メモリ管理フォールト例外	4	設定可能	CM4 のみ	メモリ管理フォールトは、メモリ保護関連のフォールトのために発生する例外。
バスフォールト例外	5	設定可能	CM4 のみ	バス フォールトは、命令またはデータ メモリ トランザクションのメモリ関連フォールトのために発生する例外。
使用法フォールト例外	6	設定可能	CM4 のみ	使用法フォールトは、命令の実行に関連するフォールトのために発生する例外。

Note:

1. *設定可能な例外優先順位は、CM0+の場合は 0～3、CM4 の場合は 0～7 です。*
2. *割込みも例外の一部です。割込みベクタ番号 0 (すなわち、IRQ 0) は例外番号 16 に対応し、それ以外も同じです。*

5.2 割込みレイテンシ

割込みレイテンシは、割込みのアサートと ISR の最初の命令実行との間の時間遅延として定義されます。最悪の場合として、CM0+のレイテンシは 15 クロック サイクル、CM4 のレイテンシは 12 クロック サイクルです。一部のペリフェラルは、ペリフェラルと CPU との間の同期回路のため追加のサイクルが発生します。**Table 3** に、PSoC 6 MCU のさまざまなペリフェラルの遅延の CPU クロック サイクル数を示します。

Table 3 さまざまなペリフェラルの同期遅延

割込みソース	同期遅延
TCPWM、DMA、USB、I2S、PDM – PCM、CDS	0 クロック サイクル
SCB、GPIO、LPComp、RTC、WDT、SMIF、Bluetooth LE	2 クロック サイクル

両方の CPU がスリープ／ディープ スリープ電力モードにある場合、同期化に追加の 2 クロック サイクルが必要です。

コンテキスト切替えはレイテンシに影響し、次の手順が必要です。

- 現在の命令実行が完了します。

- プロセッサは、現在のプログラム カウンター (PC)、リンク レジスタ (LR)、プログラム ステータス レジスタ (PSR) およびいくつかの汎用レジスタ (プログラムおよびステータス レジスタ (PSR)、リターン アドレス、リンク レジスタ (LR または R14)、R12、R3、R2、R1 および R0) をスタックにプッシュします。
- プロセッサは NVIC からベクタ アドレスを読み出し、それを PC に更新します。
- プロセッサは NVIC レジスタを更新します。

したがって、現在実行されている命令によってレイテンシは異なります。プロセスを効率的にするために、CM0+と CM4 プロセッサは両方とも次の 2 つの方式を採用します。

テール チェーン: プロセッサが他の割込みハンドラを実行中に、割込みが保留状態になっている場合、最初の割込みの実行が終了すると、アンスタッキングはスキップされ、保留中の割込みハンドラが直ちに実行されます。これにより、レジスタをスタックから復元してそれを再びスタックにプッシュする時間を節約できます。これは次の節で説明するように、ネストされた割込みと低優先順位割込みのレイテンシ削減に役立ちます。

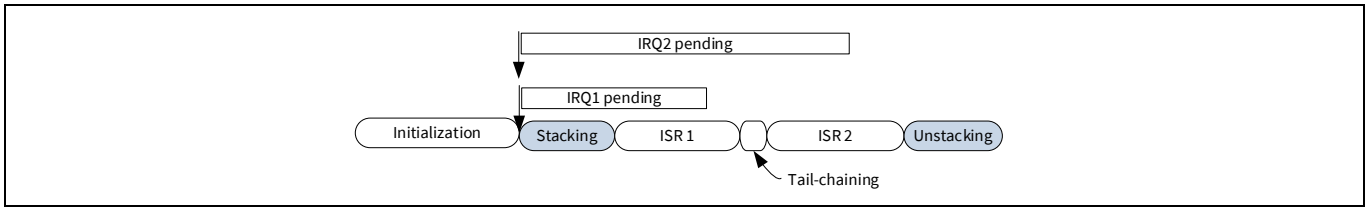


Figure 12　　テール チェーン

レイト アライバル: 優先順位の低い割込みのスタッキング処理中に優先順位の高い割込みが発生した場合、プロセッサは優先順位の低い割込みハンドラの代わりに優先順位の高い割込みハンドラにジャンプします。プロセッサは、スタッキング プロセスの終了時に優先順位の高い割込みのベクタ アドレスを読み出します。優先順位の高い割込みハンドラの実行が完了すると、優先順位の低い割込みハンドラのベクタ アドレスがフェッチされて実行されます。これにより、優先順位の低い ISR に入ってレジスタ値をスタックにプッシュすることにより、優先順位の高い割込みのレイテンシが削減されます。

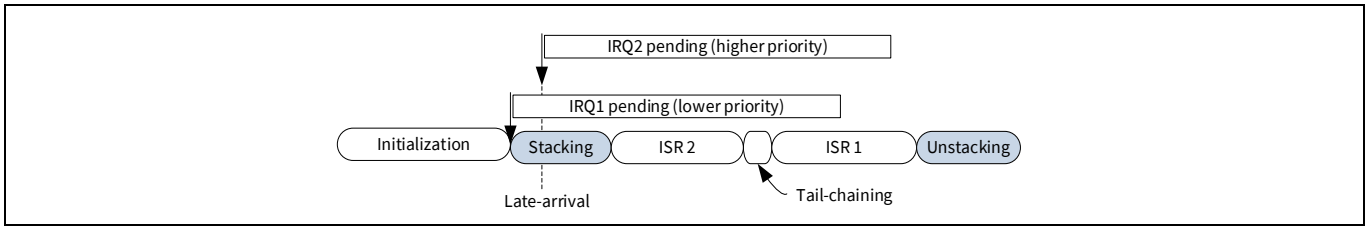


Figure 13　　レイト アライバル

5.3　　ネストされた割込み

NVIC はソフトウェア オーバーヘッドがなく、ネストされた割込みを自動的に処理します。優先順位の低い割込みハンドラ実行中に優先順位の高い割込みがアサートされると、汎用レジスタのいくつかがスタックにプッシュされ、CPU は NVIC からベクタ アドレスを読み出し、優先順位の高い割込みハンドラにジャンプします。この実行が完了すると、プロセッサはレジスタ値を復元し、優先順位の低い割込みの処理は再開されます。

5.4 コード最適化

割込みベースのアプリケーションで重要なパフォーマンス要件は、ISR コードの実行時間です。一部のアプリケーションでは、割込み要求を受け取った特定の時間内に ISR の重要なコードを実行しなければなりません。また割込み実行は、余りに多くの時間を要し、main コードの実行あるいは他の割込みを停止してはいけません。これらの要件を満たすためには、以下のガイドラインに従ってください。

- ISR から実行時間が長い関数の呼び出しを避けます。キャラクタ LCD 表示ルーチンや、長い文字列を UART 端末に出力するなどの機能は、実行時間が掛かり、他の低優先順位割込みの実行をブロックすることになります。推奨される手法は、重要ではない関数呼び出しを main コードに移し、ISR ではフラグ変数をセットすることです。main コードは定期的にフラグをチェックし、フラグがセットされれば、それをクリアして関数を呼び出します。
- 割込み処理に適切な優先順位を割り当てます。複数の割込みを持つアプリケーションでは、よりタイム クリティカルな割込みを優先します。

AN89610 – PSoC 4 and PSoC 5LP Arm Cortex Code Optimization は、異なる CPU アーキテクチャを対象としますが、コンパイラの一般的なトピックの参考に役立ちます。

6 関連リソース

PSoC 6 MCU サンプルコードの全リストについては、**Code Examples** のページを利用してください。
PSoC 6 MCU 関連のドキュメントについては、**PSoC 6 MCU 製品**のページを利用してください。

Table 4 PSoC 6 MCU 機能に関連するドキュメント

ドキュメント	資料名
アプリケーションノート	
AN228571	Getting Started with PSoC 6 MCU on ModusToolbox
AN221774	Getting Started with PSoC 6 MCU on PSoC Creator
AN210781	Getting Started with PSoC 6 MCU with Bluetooth Low Energy (Bluetooth LE) Connectivity
AN215656	PSoC 6 MCU Dual-CPU System Design
AN219434	Importing PSoC Creator Code into an IDE for a PSoC 6 MCU Project
プログラマブルデジタル	
Code Examples (ModusToolbox) on GitHub	
mtb-example-psoc6-gpio-interrupt	PSoC 6 MCU GPIO Interrupt
mtb-example-psoc6-hello-world	PSoC 6 MCU Hello World Example
mtb-example-psoc6-uart-transmit-receive-dma	PSoC 6 MCU: SCB UART Transmit and Receive with DMA
mtb-example-psoc6-spi-master-dma	PSoC 6 MCU: SCB SPI Master with DMA
mtb-example-psoc6-wdt	PSoC 6 MCU Watchdog Timer
mtb-example-psoc6-capsense-buttons-slider	PSoC 6 MCU: CapSense Buttons and Slider
mtb-example-psoc6-csdadc	PSoC 6 MCU: CSD Analog-to-Digital Converter (ADC)
mtb-example-psoc6-capsense-custom-scan	PSoC 6 MCU: CapSense Custom Scan
mtb-example-psoc6-capsense-buttons-slider-freertos	PSoC 6 MCU: CapSense Buttons and Slider (FreeRTOS)
mtb-example-psoc6-ble-findme	PSoC 6 MCU with Bluetooth Low Energy (Bluetooth LE) Connectivity: Find Me Application
mtb-example-psoc6-ble-battery-level-freertos	PSoC 6 MCU with Bluetooth LE Connectivity: Battery Level (FreeRTOS)
mtb-example-psoc6-ble-throughput-freertos	PSoC 6 MCU: Bluetooth LE Throughput Measurement (FreeRTOS)
mtb-example-psoc6-usb-audio-device-freertos	PSoC 6 MCU: USB Audio Device with FreeRTOS
mtb-example-psoc6-usb-cdc-echo	PSoC 6 MCU: USB CDC Echo Application
mtb-example-psoc6-usb-hid-mouse	PSoC 6 MCU: USB HID Mouse Application
mtb-example-psoc6-usb-hid-generic	PSoC 6 MCU: USB HID Generic Application
mtb-example-psoc6-usb-msc-file-system	PSoC 6 MCU: USB Mass Storage File System
mtb-example-psoc6-usb-msc-logger	PSoC 6 MCU: USB Mass Storage Logger
Code Examples (PSoC Creator)	
CE219521	PSoC 6 MCU - GPIO Interrupt

関連リソース

ドキュメント	資料名
CE216795	PSoC 6 MCU Dual-Core Basics
CE218129	PSoC 6 MCU Wake up from hibernate Using Low-Power Comparator
CE218542	PSoC 6 Custom Tick Timer Using RTC Alarm Interrupt
CE219339	PSoC 6 MCU MCWDT and RTC Interrupts (Dual Core)
CE220061	PSoC 6 MCU Multi-Counter Watchdog Interrupts
CE220607	PSoC 6 MCU Watchdog Timer in interrupt mode
CE220169	PSoC 6 MCU Periodic Interrupt Using TCPWM
CE212736	PSoC 6 MCU with Bluetooth Low Energy (Bluetooth LE) Connectivity - Find Me
ソフトウェア/IDE	
PSoC Creator	PSoC Creator User Guide
ModusToolbox	ModusToolbox User Guide
Peripheral Driver Library (PSoC Creator)	PDL API Reference available on installation
Peripheral Driver Library (ModusToolbox)	PDL API Reference

付録 A. PSoC 6 MCU の割込みソース

各割込みの IRQ 番号と該当する電力モードについては、各デバイスのテクニカル リファレンス マニュアルの「割込み」章を参照してください。

割込みソース	詳細
GPIOs	各ポートは最大 8 ピンで構成される。各ピンは割込みを生成でき、そのベクタ アドレスはポート内のすべてのピンに共通。ファームウェアは、その割込みの原因となったピンを特定する必要がある。 PSoC 6 MCU は、GPIO 信号の立ち上りエッジ、立ち下りエッジまたは両方のエッジで割込みトリガを有効にする。この割込みは、デバイスをスリープ、ディープ スリープ モードから復帰させることが可能。
	すべてのポート割込みを 1 つのベクタに結合する「全ポートの GPIO」割込みがある。ファームウェアは、その割込みの原因となったポートを特定する必要がある。
	電源ランプ アップまたはランプ ダウンを検出するために使用できる「GPIO 電源検出」割込みがある
LPComp	GPIO と同様に、コンパレータの出力信号の立ち上りエッジ、立ち下りエッジまたは両方のエッジで割込みをトリガすることが可能。 LPComp は、デバイスをスリープ、ディープ スリープ モードおよびハイバネート 電力モードから復帰させることが可能。
SCB (ディープスリープ)	CPU/システムをディープスリープからウェイクアップできる SCB 割込み
マルチカウンター ウォッチドッグタイマー (MCWDT) 割込み	MCWDT は、定期的な割込みを生成するように 2 つの 16 ビット カウンターと 1 つの 32 ビット カウンターを構成することが可能。 MCWDT は、ディープ スリープ電力モードから CPU を復帰させることが可能。
バックアップドメイン割込み	バックアップ ドメイン割込みは、RTC ALARM1、RTC ALARM2 および RTC センチュリーオーバーフロー割込みが含まれる。この使用で、スリープ、ディープ スリープおよびハイバネート電力モードから CPU を復帰させることが可能。
SRSS 用のその他の組み合わされた割込み	この割込みは、WDT 割込み、低電圧検出 (LVD) 割込みおよびクロック校正割込みによって生成される。WDT 割込みは、ウォッチドッグ カウンター値がプリセットカウンター マッチ値と一致すると発生。2 つの割込みが起こらない場合、ウォッチドッグ リセットが発生。 デバイスの電源電圧が閾値を下回ると、低電圧検出 (LVD) 割込みが発生。 クロック校正が完了すると、クロック校正割込みがトリガされる。 これらは、ディープ スリープから CPU を復帰させることが可能。
CTBm 割込み (すべての CTBms)	このブロックは、連続時間アナログ機能を提供。コンパレータ トリガなどのイベントで割込みを生成。
Bluetooth 無線割込み	Bluetooth サブシステム割込み
IPC 割込み	IPC リリースまたは通知イベントが発生すると、IPC 割込みがトリガされる可能性がある。
SAR ADC	

付録 A. PSoC 6 MCU の割込みソース

割込みソース	詳細
CTBm 割込み (個別 CTBm)	このブロックは、連続時間アナログ機能を提供。 コンパレータ トリガなどのイベントで割り込みを生成。
PASS FIFO 割込み	
SCB	PSoC 6 MCU は、SPI、I2C または UART に設定可能な 9 個の SCB に対応。8 個の SCB のうち、ディープスリープ対応の割込みが 1 つある。以下のイベントは、SCB で割込みを生成。 • TX FIFO のエントリ数が指定されたエントリ数より少ない。TX FIFO は満杯でない／満杯／オーバーフロー／アンダーフロー。RX FIFO のエントリ数が指定された値より多く、RX FIFO が満杯／空でない。 • SPI: EZSPI 転送が発生した後の SPI マスター転送完了の時、SPI バス エラーの時、SPI スレーブ選択解除の時に SPI 割込みが発生。 • I2C: I2C マスターのアービトレーション負け、NACK 受信、ACK 受信、STOP 送信、I2C バス エラー、I2C スレーブのアービトレーション負け、NACK 受信、ACK 受信、STOP 受信、START 受信、アドレス一致。 • UART 割込み: TX の SmartCard モードでの NACK 受信、TX 完了、アービトレーション負け、受信データ フレームのフレーム エラー、受信データ フレームのパリティ エラー、LIN ボーレート検出完了、LIN ブレーク検出成功。
CSD (CapSense™) 割込み	タッチ アプリケーションに使用される CSD は、センサー スキャンが完了すると割込みを生成。
CPUSS DMAC, チャネル#0～3	DMA コントローラー (DMAC) は、転送完了、バスエラー、アドレスの不整合、現在のポインターが NULL、アクティブ チャネルが無効、ディスクリプタ エラーなどの DMA イベントで割り込みます。
DMA 割込み	データ転送が完了すると DMA 割込みが発生することがある。
CPUSS フォールト構造体 割込み#0	この割込みは、保護ユニットのアクセス違反がある場合に発生。
暗号化アクセラレータ割込み	暗号化割込みは、次の場合に生成される: FIFO イベントがアクティブになった時、FIFO がオーバーフローした時、真の乱数発生器が初期化された時、真の乱数発生器が指定されたビット サイズのデータ値を生成した時、擬似乱数生成器がデータ値を生成した時、命令デコーダが未定義のオペコードを有する命令に遭遇した時、命令デコーダが未定義の条件コードを有する命令に遭遇した時、AHB-Lite バス エラーが発生した時、真の乱数発生器モニター適応比例テストで特定のビット値の繰り返しを検出した時、真の乱数発生器モニター適応比例テストで特定のビット値の不均衡が発生した時。
フラッシュマクロ割込み	フラッシュ コントローラーには、割込みを生成するタイマーを備える。
浮動小数点割込み	浮動小数点演算フォールト
CM0+ CTI #0	CTI トリガは、デバッグ コンポーネント間でイベント通信のために使用。
CM4 CTI #0	CTI トリガは、デバッグ コンポーネント間でイベント通信のために使用。

付録 A. PSoC 6 MCU の割込みソース

割込みソース	詳細
TCPWM	TCPWM ブロックは、16 ビットまたは 32 ビットのタイマー、カウンターまたは PWM として動作するように設定可能。これは、ターミナル カウント、入力キャプチャ信号または比較の真のイベントで割込みを生成することが可能。
UDB 割込み#0	UDB で生成されたデジタル信号は、割込みをトリガすることが可能。信号は、デジタル システム相互接続 (DSI) と呼ばれるルーティング ファブリックを経由し、割込みコントローラーにルーティングされる。
I2S オーディオ割込み	次の場合に割込みが生成される: TX FIFO のエントリ数が指定された値より少なくなった時、TX FIFO が満杯ではない時、TX FIFO が空である時、満杯の TX FIFO への書き込みを試みた時、空の TX FIFO からの読み出しを試みた時、Tx ウォッチドッグ イベントが発生した時、RX FIFO のエントリ数が指定された値より多くなった時、RX FIFO が空ではない時、RX FIFO が満杯である時、満杯の RX FIFO への書き込みを試みた時、空の RX FIFO からの読み出しを試みた時、Rx ウォッチドッグ イベントが発した時。
PDM/PCM オーディオ割込み	RX FIFO のエントリ数が指定された値より多くなった時、RX FIFO が空ではない時、満杯の RX FIFO への書き込みを試みた時、空の RX FIFO からの読み出しを試みた時。
エネルギープロファイラー割込み	この割込みは、プロファイリング カウンター オーバーフローで発生。
シリアルメモリ インターフェース割込み	この割込みは、TX データ FIFIO アクティブ、RX データ FIFO アクティブ、アライメント エラー、FIFIO オーバーフローが発生した時にアクティブになる。
USB 割込み	USB ブロックは、3 つの割込みのいずれか 1 つにマッピングできる 13 個の割込みトリガ イベントの事前定義されたセットがある。USB フレーム開始 (SOF)、USB バス リセット、データ エンドポイント イベント、制御エンドポイント イベント、アービター割込みイベント、リンク電源管理 (LPM) イベントなどのイベントによって割込みが生成される
CAN 割込み	CAN 統合または個別チャネル割込み
全 DAC 用の統合された割込み	DAC バッファが空の時、割込みを生成することが可能。この割込みは、CPU が次の値を DAC に転送するために使用できる
SDHC の SDIO ウェイクアップ割込み	SDIO ウェイクアップ割込みは、カードの挿入、取り外し、SDIO カード割込みなどのイベントでトリガされます。システムはディープスリープから復帰しません。
SDHC の統合割込み	SDHC に関連する他のすべての通常/エラーイベントの統合された割込み
mxsdhc の EEMC ウェイクアップ割込み (未使用)	SDHC ブロックの EEMC ウェイクアップ割込み (予約済み)
SDHC の統合割込み	統合された割込み (予約済み)

改訂履歴

Document version	Date of release	Description of changes
**	2018-07-19	これは英語版 002-17666 Rev. **を翻訳した日本語版 Rev. **です。
*A	2019-09-26	これは英語版 002-17666 Rev. *B を翻訳した日本語版 Rev. *A です。
*B	2020-01-28	これは英語版 002-17666 Rev. *C を翻訳した日本語版 Rev. *B です。
*C	2021-06-16	これは英語版 002-17666 Rev. *E を翻訳した日本語版 Rev. *C です。

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

Edition 2021-06-16
Published by
Infineon Technologies AG
81726 Munich, Germany

© 2021 Infineon Technologies AG. All Rights Reserved.

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

Document reference
002-23482 Rev. *C

重要事項
本文書に記載された情報は、いかなる場合も、条件または特性の保証とみなされるものではありません（「品質の保証」）。本文に記された一切の事例、手引き、もしくは一般的価値、および／または本製品の用途に関する一切の情報に関し、インフィニオンテクノロジーズ（以下、「インフィニオン」）はここに、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。

さらに、本文書に記載された一切の情報は、お客様の用途におけるお客様の製品およびインフィニオン製品の一切の使用に関し、本文書に記載された義務ならびに一切の関連する法的要件、規範、および基準をお客様が遵守することを条件としています。

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。

本製品、技術、納品条件、および価格についての詳しい情報は、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（**www.infineon.com**）。

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。