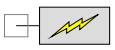
インタラプト
1.50

特長



- ハードウェアトリガによる割り込みを定義
- 割り込みを保留するソフトウェアメソッドを提供

概要

インタラプトコンポーネントは、ハードウェアトリガによる割り込みを定義します。これは Interrupt Design-Wide Resource system に不可欠な部分です(PSoC Creator Help の Design-Wide Resources セクションを参照してください)。

割り込みコントローラによって処理できるシステム割り込み波形は 3 種類あります。

- **Level** – IRQ ソースに動きが少なく、ファームウェアがリクエストソースをクリアする(読み出しによるクリアなど)までアクティブ状態を続けます。殆どのフィクスドファンクション(固定機能)ペリフェラルコンポーネントは、レベル検出割り込みを持っており、これには UDB FIFO とステータスレジスタも含まれます。
- **Pulse** – 理想的には、パルス IRQ は、保留中のアクションをログに記録し、ISR アクションが一回だけ実行されることを保証する、バスクロック 1 クロック分のパルスです。ペリフェラルコンポーネントに対してファームウェアは操作する必要はありません。
- **Edge** – 任意の同期波形がエッジ検出回路への入力になります。その波形の立ち上がりエッジが同期した 1 サイクルパルスになります (Pulse モード)。

注:上記の割り込み波形タイプは、**Configure** ダイアログにおける **InterruptType** パラメータの設定とは異なります。このパラメータはマルチプレクサ選択ラインのみを構成し、マルチプレクサの選択(Level、Edge)に基づいて、割り込みコントローラに送信される"IRQ"信号を処理します。

すなわち、**InterruptType** マルチプレクサ選択に関わらず、割り込みコントローラはレベル、エッジ、パルス波形を処理できることを意味します。詳細については、該当する TRM 資料を参照してください。

インタラプトコンポーネントの用途

インタラプトコンポーネントは、ハードウェアトリガによる割り込みが必要な場合に使用します。ポーリングと比較すると、割り込みは、ハードウェアを使用してレイテンシとイベント検出のオーバーヘッドを両方低減するため、必要不可欠です。

入出力の接続

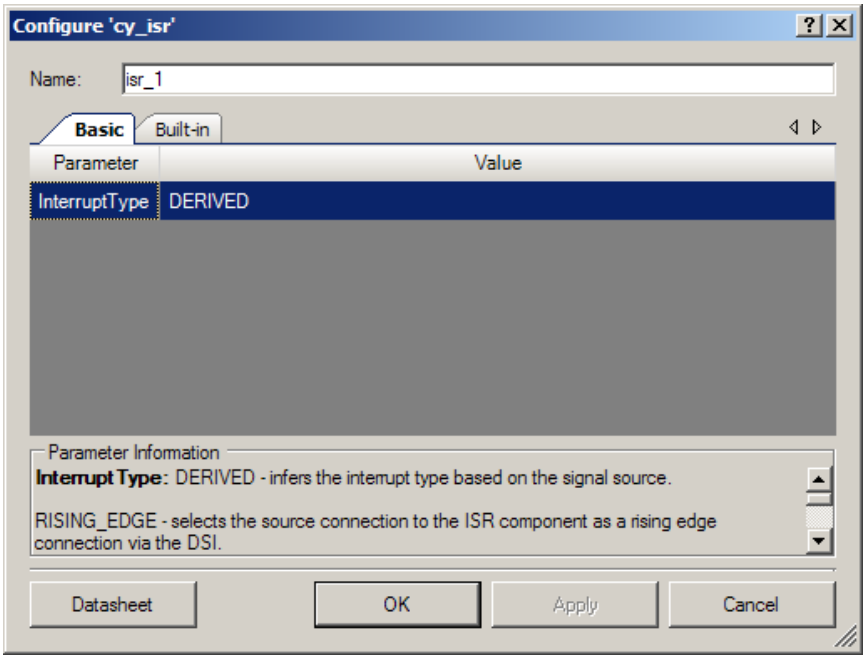
ここでは、インタラプトコンポーネントのさまざまな入出力接続について説明します。

int_signal – 入力

割り込みを生成する信号を接続します。信号がハイレベルになると、割り込みがトリガされます。

コンポーネント パラメータ

インタラプトコンポーネントを回路図にドラッグし、ダブルクリックして **Configure** ダイアログを開きます。



インタラプトコンポーネントには次のパラメータがあります。

InterruptType

このパラメータは、割り込みをトリガするために処理する波形のタイプを設定します。このパラメータには 3 つの値があります。

- **RISING_EDGE** – ソース信号の立ち上がりエッジで割り込みをトリガします。このオプションを選択すると、"int_signal"入力の立ち上がりエッジが"bus_clk"のパルスに変換され、割り込みコントローラに送信されます。
- **LEVEL** –DSI を介したレベル検出接続として、割り込みに接続されているソースを選択します。このオプションを選択すると、"int_signal"入力が割り込みコントローラに直接渡されます。詳細については、概要セクションを参照してください。

- **DERIVED** – これは初期設定です。"int_signal"の信号源を調べ、フィクスドファンクションブロック(I²C、USB、CAN など)に接続される場合、その接続先の割り込みタイプを引き継ぎます。自動割り当ては、デバイスのデータシート記載の情報に基づいて行われます。

フィクスドファンクション割り込み出力に接続する場合、タイプは DERIVED になります。他の割り込みソースの場合、通常は、イベント(例えば周期的なクロック)のキャプチャでは"RISING_EDGE"を、状態(例えば FIFO 書き込みレベル)では"LEVEL"を選択します。DMA NRQ シグナルの場合、どの設定によっても、それぞれの NRQ イベントに対するシングル割り込みと同じ結果が得られます。

配置

各インタラプトコンポーネントは、デバイスの割り込みベクタメモリ中の 1 つのエントリを消費します。

リソース

インタラプトコンポーネントは、次のデバイスリソースを使用します。

- 割込みサービス ルーチン(ISR)コード用のフラッシュスペース
- 割り込みベクタメモリ中の 1 エントリ

	デジタルブロック					API メモリ(バイト)		
アナログブロック	データパス	マクロセル	ステータスレジスタ	コントロールレジスタ	Counter7	フラッシュ	RAM	ピン (外部入出力ごと)
該当せず	該当せず	該当せず	該当せず	該当せず	該当せず	189	0	該当せず

アプリケーションプログラミングインタフェース

アプリケーションプログラミングインターフェース (API) ルーチンにより、ソフトウェアを使用してコンポーネントを設定できます。次の表は、各関数へのインターフェースとその説明を示しています。その次のセクションでは、各関数について詳しく説明します。

初期設定では、PSoC Creator は、ユーザの回路図に最初に配置されたコンポーネントのインスタンス名として"ISR_1"を割り当てます。コンポーネントの名称は、識別子の文法ルールに従って固有の名前に変更できます。インスタンス名は、すべてのグローバル関数名、変数名、定数名のプリフィックスになります。読みやすいように、次の表では"ISR"というインスタンス名を使用します。

関数	機能
ISR_Start()	割り込みが作動します。
ISR_StartEx()	割り込みが作動し、アドレスをISRベクタに設定します。

関数	機能
ISR_Stop()	割り込みを無効にし削除します。
ISR_Interrupt()	ISR 用のデフォルトの割り込みハンドラ。
ISR_SetVector()	アドレスを新規 ISR ベクタに設定します。
ISR_GetVector()	現在の ISR ベクタのアドレスを取得します。
ISR_SetPriority()	割り込みの優先順位を設定します。
ISR_GetPriority()	割り込みの優先順位を取得します。
ISR_Enable()	割り込みコントローラに対して割り込みを有効にします。
ISR_GetState()	割り込みの状態(有効、無効)を取得します。
ISR_Disable()	割り込みを無効にします。
ISR_SetPending()	割り込みを保留状態にします。これはソフトウェアによる割り込みを生成する手段です。
ISR_ClearPending()	保留中の割り込みをクリアします。

void ISR_Start(void)

機能: 割り込みを設定し、有効にします。この関数は割り込みを無効にし、デフォルトの割り込みベクタを設定し、Design Wide Resources Interrupt エディタの値から優先順位を設定し、割り込みコントローラに対して割り込みを有効にします。

パラメータ: void

返り値: void

注意事項: なし

void ISR_StartEx(cyisraddress address)

機能: 割り込みを設定し、有効にします。この関数は割り込みを無効にし、渡されたアドレスに基づいて割り込みベクタを設定し、Design Wide Resources Interrupt エディタの値から優先順位を設定し、割り込みコントローラに対して割り込みを有効にします。

パラメータ: address: 割り込みベクタテーブルに設定されている ISR のアドレス。

返り値: void

注意事項: なし

void ISR_Stop(void)

機能: 割り込みを無効にし、削除します。

パラメータ: void

返り値: void

注意事項: なし

void ISR_Interrupt(void)

機能: コンポーネント用のデフォルト ISR。対応する C ファイルでこの関数を見つけ、START コメントと END コメントの間にコードを挿入します。

パラメータ: void

返り値: void

注意事項: なし

void ISR_SetVector(cyisraddress address)

機能: ISR ベクタを変更します。この関数は、異なる割込みサービス ルーチンのアドレスに ISR ベクタを変更するために使用します。注:ISR_Start() を呼び出すと、このメソッドがそれまで保持していた効果が上書きされます。コンポーネント開始前にこのベクタを設定するには、ISR_StartEx() を使用します。

パラメータ: address: 割り込みベクタテーブルに設定されている ISR のアドレス。

返り値: void

注意事項: この関数を呼び出す前、そしてその後これを再び有効にする前に、この割り込みを無効にします。

cyisraddress ISR_GetVector(void)

機能: 現在の ISR ベクタのアドレスを取得します。

パラメータ: void

返り値: cyisraddress: 現在の ISR のアドレス。

注意事項: なし

void ISR_SetPriority(uint8 priority)

機能: 割り込みの優先順位を設定します。

注 ISR_Start() または ISR_StartEx() を呼び出すと、このメソッドがそれまで保持していた効果が上書きされます。このメソッドは、ISR_Start() または ISR_StartEx() が呼び出された後でのみ、呼び出すことができます。コンポーネントに初期の優先順位を設定するには、Design Wide Resources Interrupt エディタを使用します。

パラメータ: priority: 割り込みの優先順位。0～7。0が最高。

返り値: void

注意事項: なし

uint8 ISR_GetPriority(void)

機能: 割り込みの優先順位を取得します。

パラメータ: void

返り値: 割り込みの優先順位。0～7。0が最高。

注意事項: なし

void ISR_Enable(void)

機能: 割り込みコントローラに対して割り込みを有効にします。ISR_Start() が呼び出された場合、または ISR_Start() 関数のベクタと優先順位を設定する機能が呼び出された場合以外は、この関数を呼び出さないでください。

パラメータ: void

返り値: void

注意事項: なし

uint8 ISR_SetState(void)

機能: 割り込みの状態(有効、無効)を取得します。

パラメータ: void

返り値: 1が有効、0が無効。

注意事項: なし

void ISR_Disable(void)

機能: 割り込みコントローラに対して割り込みを無効にします。

パラメータ: void

返り値: void

注意事項: なし

void ISR_SetPending(void)

機能: 割り込みを保留状態にします。これはソフトウェアによる割り込みを生成する手段です。

パラメータ: void

返り値: void

注意事項: 割り込みが有効で、適切に設定されている場合、ISR に入ります。(この割り込みとその他保留中の割り込みの優先順位によります)。

void ISR_ClearPending(void)

機能: 割り込みコントローラに対して保留中の割り込みをクリアします。

パラメータ: void

返り値: void

注意事項: 一部の割り込みソースは、適切なブロック API (GPIO、UART等) を使用してクリアしなければなりません。そうしないと、割り込みが再び保留されます。ISR の実行により保留ビットがクリアされる割り込みソースもあります。

ファームウェアソースコードのサンプル

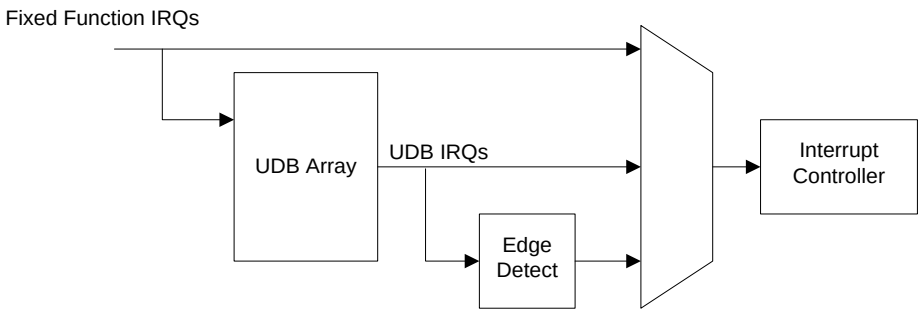
PSoC Creator は、Find Example Project ダイアログに、回路図およびサンプルコードを含む多くのサンプルプロジェクトを提供しています。コンポーネント特有のサンプルを見るには、Component Catalog または回路図に置いたコンポーネントインスタンスからダイアログを開きます。一般的なサンプルについては、Start Page または **File** メニューからダイアログを開きます。必要に応じてダイアログにある **Filter Options** を使用し、選択できるプロジェクトのリストを絞り込みます。

詳しくは、PSoC Creator ヘルプの Find Example Project を参照してください。

機能の詳細

割り込みルーティングは、PSoC 3 アーキテクチャ上では柔軟に設計されています。フィクスドファンクションペリフェラルデバイスに加えて、UDB アレイ配線内の任意のデータ信号も割り込みの生成に使用することができます。図 1 は、割り込みマルチプレクサ (IDMUX)周辺のブロック図です。IDMUX は、割り込み要求の利用可能なソースから選択します。

図 1. IDMUX 周辺ブロック図



Design-Wide Resources

インタラプトコンポーネントを設計で使用すると、Design-Wide Resources エディタにエントリが生成されます。**Interrupts** タブには次のパラメータがあります。

Instance Name	Priority	ES2 Patch	Vector
isr_1	Default <7>	☐	1
isr_2	Default <7>	☐	3

- **Instance Name** – 設計中のコンポーネントインスタンス名を示します。
- **Priority** – インスタンスの優先順位を示し、変更することもできます。
- **ES2 Patch** –PSoC 3 割り込みの問題を修正する一時的な回避手段です。低優先順位の割り込みから戻る (RETI) と同時に、高優先順位の割り込みが到着するとその高優先順位の割り込みは失われます。アプリケーションで複数の割り込みが使用される場合、このオプションにチェックマークを付けます。注　このオプションはシリコンが修正されるまでの一時的な対策です。
- **Vector** – 割り込みベクタを指します。

コンポーネントの変更

ここでは、前のバージョンからコンポーネントに加えられた主な変更を示します。

バージョン	変更の説明	変更の理由 / 影響
1.50.c	データシートの [Derived (派生)] オプションの詳しい説明	
1.50.b	データシートの修正	
1.50.a	データシートのマイナーな編集と更新	
1.50	InterruptType パラメータを追加。	旧機能 (新バージョンでの「DERIVED」選択に相当) では、すべての状況で望ましい割り込みタイプを特定することができなかったので、手動による指定機能を追加しました。
	すでに存在している場合は、CYINT_VECTORSと CYINT_IRQ_BASE を再定義しません。	これらのマクロはすでに*CyLib.h*で定義されています。再定義によって、cy_boot の一部バージョンで警告が発生しました。この変更で影響を受けるのは PSoC 5 だけです。
	CY_ISR を伴った ISR を宣言。	これにより、コンパイラが PSoC 5 上の正しいスタックアライメントを保証するコードを生成する。
	cydevice.h の代わりに *cydevice_trm.h* を使用します。	*cydevice.h* は旧式であり、古いコンポーネントとファームウェアとの整合性の目的にのみ使用します。割り込み API 関数が *cydevice.h* を必要とする場合は、*cydevice.h* を「Place your includes, defines, and code here(添付、定義、コードをここに配置する)」セクションに含めます。
	Added ISR_StartEx	割り込み開始前に割り込みベクタテーブルを設定するために、ISR のアドレスの設定を許可。これにより、ISR_Interrupt の代わりにこれをデフォルトとして使用します。
	「=ReentrantKeil($INSTANCE_NAME . "_...")」を次の関数に追加します。 void ISR_Stop() void ISR_SetVector() cyisraddress ISR_GetVector() void ISR_SetPriority() uint8 ISR_GetPriority() void ISR_Enable() uint8 ISR_GetState() void ISR_Disable() void ISR_SetPending() void ISR_ClearPending()	必要な場合に、これらの API を再入可能できるようにするためです。
1.20	ES2 ISR パッチ。	