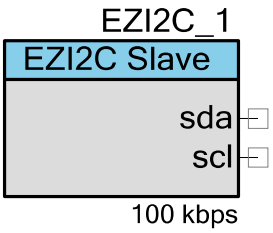
EZI2C スレーブ

1.61



特長

- 業界標準 NXP® I²C バス インタフェース
- 一般的な I²C EEPROM インタフェースをエミュレート
- 2 つのピン (SDA、SCL) で I²C バスへインタフェース
- 50/100/400/1000 kbps の標準データ レート
- ユーザプログラミングを最小限に抑える高級 API
- 1 つまたは 2 つのアドレスをサポートし、独立したメモリ バッファでデコードします
- メモリバッファは、構成可能な読み出し/書き込み領域および読み出し専用領域を提供します

概要

EZI2C スレーブコンポーネントは、I²C レジスタベースのスレーブデバイスを実装します。I²C バスは、フィリップス社® によって開発された、業界標準の 2 線式ハードウェアインタフェースです。マスタは、I²C バスの全通信を開始し、すべてのスレーブデバイスへクロックを供給します。EZI2C スレーブは、最大 1000kbps までの標準データ速度をサポートし、同じバスの複数のデバイスに互換性があります。

EZI2C スレーブは I²C スレーブ独自の実装であり、マスタとスレーブ間でのあらゆる通信は ISR (割り込みサービスルーチン) で取り扱われ、メインプログラムフローとのやりとりは必要ありません。インタフェースは、マスタとスレーブ間の共有メモリとして表示されます。EZI2C_Start() 関数が実行された後は、API とのやりとりはほとんど必要ありません。

EZI2C スレーブの用途

このコンポーネントは、I²C スレーブと I²C マスタの間で共有メモリ モデルを使用したい場合に使用します。I²C プロトコルを気にすることなく、EZI2C スレーブバッファを任意の変数、配列または構造体として定義できます。I²C マスタは、このバッファのあらゆる変数を表示することができ、EZI2C_SetBuffer1() または EZI2C_SetBuffer2() 関数で定義した変数を修正できます。

入出力の接続

このセクションでは、EZI2C スレーブのさまざまな入出力接続について説明します。

sda – 入力/出力

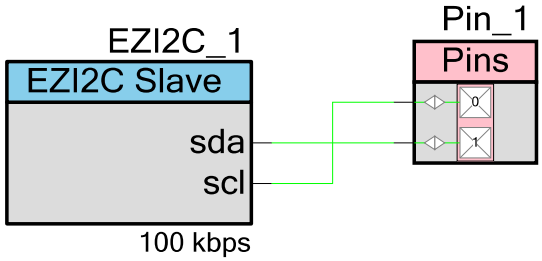
シリアル データ (SDA) は I^2C データ信号です。これは双方向のデータ信号で、すべてのバス データを送信または受信するのに使用します。

scl – 入力/出力

シリアルクロック (SCL) はマスタ生成による I^2C クロックです。スレーブはクロック信号を生成しませんが、信号をローレベルにに保ち、データの送信準備ができるまで、あるいは最新のデータまたはアドレスを NAK/ACK にするまで、バスをストールします。

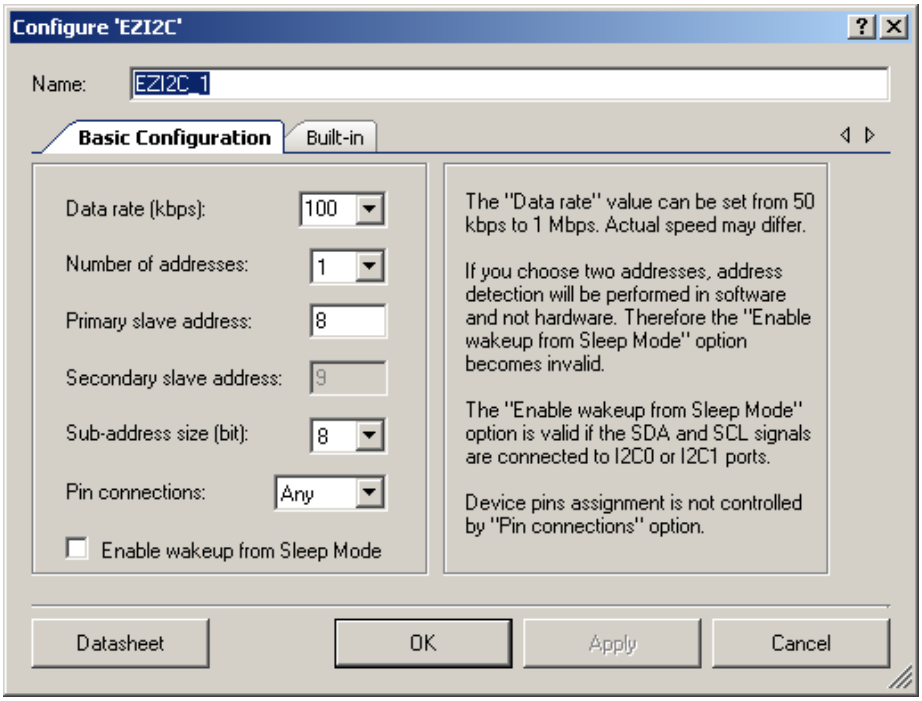
回路図マクロ情報

コンポーネント カタログ内の初期設定の EZI2C スレーブは、EZI2C スレーブの初期設定を使用した回路図マクロです。EZI2C スレーブ コンポーネントは、SIO ペアとして設定されているピン コンポーネントに接続されています。



コンポーネント パラメータ

EZI2C コンポーネントをデザイン上にドラッグし、ダブルクリックして **Configure** ダイアログを開きます。



EZI2C コンポーネントには次のパラメータがあります。

Data rate

このパラメータを使用して、I²C データレートを設定します。最大値は 1000kbps です。実際のレートは、使用できるクロック速度と分周器の範囲に基づき、異なる場合があります。標準データ レートは 50、100 (初期値)、400、1000kbps です。

Number of addresses

このオプションでは、独立した I²C スレーブ アドレスが **1** つ (初期値) または **2** つ認識されるかどうかを決定します。2 つのアドレスが認識される場合、アドレス検出はハードウェアではなくソフトウェア内で実行されます。そのため、**Enable wakeup from Sleep Mode** オプションは利用できません。

Primary slave address

これは、I²C スレーブの 第一アドレスです (初期値は **8**)。この値は 10 進数または 16 進数形式で入力できます。16 進数の場合、数の前に「0x」と入力します。このアドレスは、7 ビットの右揃えスレーブ アドレスで、R/W ビットを含みません。

Secondary slave address

これは、I²C スレーブの 第二アドレスです (初期値は 9)。この値は 10 進数または 16 進数形式で入力できます。16 進数の場合、数の前に「0x」と入力します。この 第二アドレスは、**Number of addresses** パラメータが **2** に設定されている場合のみ有効です。スレーブの 第一アドレスと 第二アドレスは、別のものである必要があります。このアドレスは、7 ビットの右揃えスレーブ アドレスで、R/W ビットを含みません。

Sub-address size

このオプションで、アクセス可能なデータ範囲を決定します。**8** ビット (初期値) または **16** ビットのサブ アドレスを選択できます。8 ビットのアドレス サイズを使用すると、マスタは 0 ～ 255 のデータ オフセットにのみアクセスできます。16 ビットのサブ アドレス サイズも選択できます。これにより I²C マスタは、各スレーブ アドレスで最大 65,535 バイトのデータ アレイにアクセスできます。

Pin connections

このパラメータで、SDA および SCL 信号接続で使用するピンのタイプを決定します。このオプションは、**Enable wakeup from Sleep mode** オプションを補完し、単一の I²C アドレスが **Number of addresses** オプションで選択されている場合にのみ使用できます。以下の 3 つの値があります。**Any**、**I2C0** および **I2C1**。初期設定は **Any** です。

Any は汎用 I/O (GPIO) を示します。**Enable wakeup from Sleep Mode** が必要ではない場合、SDA および SDL に **Any** を使用します。**Enable wakeup from Sleep Mode** が必要な場合、I2C0 (P12[4]、P12[5]) または I2C1 (P12[0]、P12[1]) を使用する必要があります。これを使用して、I²C アドレスの一致時にデバイスが復帰するように設定できます。

Enable wakeup from Sleep Mode

このパラメータによって、スレーブアドレスの一致時にデバイスをスリープモードから復帰できます。このオプションは、初期設定では無効になっています。アドレスの一致時に復帰するオプションは、単一の I²C アドレスが選択されていて、SDA および SCL 信号が SIO ポートに接続されている場合に有効です (ピンのペア I2C0 または I2C1)。**Enable wakeup from Sleep mode** は PSoC 3 製品のみサポートされています。

クロックの選択

クロックはシステムバスクロックに結合されており、ユーザが変更することはできません。

リソース

固定機能 I²C ブロックがこのコンポーネントで使用されます。

モード	デジタル ブロック					API メモリ (バイト)		ピン
	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	
1アドレスデコード (初期設定)	該当せず	該当せず	該当せず	該当せず	該当せず	895	18	2
2アドレスデコード (16 ビット Sub-address size)	該当せず	該当せず	該当せず	該当せず	該当せず	1620	37	2

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"EZI2C_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、定数名の接頭辞になります。便宜上、次の表では"EZI2C"というインスタンス名を使用します。

関数	機能
EZI2C_Start()	I²C トラフィックへの応答を開始します。I²C 割り込みを有効にします。
EZI2C_Stop()	I²C トラフィックへの応答を停止します。I²C 割り込みを無効にします。
EZI2C_EnableInt()	I²C 割り込みを有効にします。これは大半の I²C 処理で必要です。
EZI2C_DisableInt()	I²C 割り込みを無効にします。EZI2C_Stop() API はこれを自動で実行します。
EZI2C_SetAddress1()	第一 I²C アドレスを設定します。
EZI2C_GetAddress1()	第一 I²C アドレスを返します。
EZI2C_SetBuffer1();	第一 I²C のバッファ ポインタを設定します。
EZI2C_GetActivity(void)	コンポーネントの動作状況を確認します。
EZI2C_Sleep()	I²C 処理を停止し、I²C 設定を保存します。I²C 割り込みを無効にします。
EZI2C_Wakeup()	I²C 設定を復元し、I²C 処理を開始します。I²C 割り込みを有効にします。

関数	機能
EZI2C_Init()	Configureダイアログで設定した値を用いて I²C レジスタを初期化します。
EZI2C_Enable()	ハードウェアを起動し、コンポーネントの処理を開始します。
EZI2C_SaveConfig()	EZI2C コンポーネントの現在のユーザ設定を保存します。
EZI2C_RestoreConfig()	非保持 I²C レジスタを復元します。

第二アドレス API

これらのコマンドは、I²C アドレスが 2 つ有効になっている場合にのみ存在します。

関数	機能
EZI2C_SetAddress2()	第二 I²C アドレスを設定します。
EZI2C_GetAddress2()	第二 I²C アドレスを返します。
EZI2C_SetBuffer2();	第二 I²C のバッファ ポインタを設定します。

スリープ/ ウェイク モード

これらの関数は、単一のアドレスが使用され、SCL と SDA 信号が SIO ピンに配線され、**Enable wakeup from Sleep Mode** が選択されている場合にのみ使用できます。

関数	機能
EZI2C_SlaveSetSleepMode()	EZI2C スリープアドレスデコードを有効にし、I²C 設定を保存します。割り込みを無効にします。
EZI2C_SlaveSetWakeMode()	EZI2C スリープアドレスデコードを無効にし、I²C 設定を復元し、I²C 処理を開始します。割り込みを有効にします。

グローバル変数

通常の処理では、これらの変数の知識は必要ありません。

関数	機能
EZI2C_initVar	EZI2C の初期化が済んでいるかを示します。変数は、0 に初期化され、最初に EZI2C_Start() が呼び出されると 1 にセットされます。これで、EZI2C_Start() ルーチンを最初に呼び出した後で、再初期化を行うことなく、コンポーネントを再起動できます。 コンポーネントの再初期化が必要な場合は、EZI2C_Start() ルーチンを呼び出す前に変数を 0 にクリアする必要があります。あるいは、EZI2C_Init() および EZI2C_Enable() 関数を呼び出すことで EZI2C を再初期化することができます。
EZI2C_dataPtrS1	第一スレーブアドレスでI²C マスタに見せられるデータへのポインタを保存します。

関数	機能
EZI2C_rwOffsetS1	読み出しおよび書き込み処理用にオフセットを保存します。第一スレーブアドレスのそれぞれの書き込みシーケンスに設定されます。
EZI2C_rwIndexS1	第一スレーブアドレスで読み出しまたは書き込まれる次の値へのポインタを保存します。
EZI2C_wrProtectS1	第一スレーブアドレスで、読み出し専用のデータへのオフセットを保存します。
EZI2C_bufSizeS1	第一スレーブアドレスでI^2C マスタに見せられるデータ配列のサイズを保存します。
EZI2C_dataPtrS2	第二スレーブアドレスでI^2C マスタに見せられるデータへのポインタを保存します。
EZI2C_rwOffsetS2	読み出しおよび書き込み処理用にオフセットを保存します。第二スレーブデバイスのそれぞれの書き込みシーケンスに設定されます。
EZI2C_rwIndexS2	第二スレーブアドレスで、読み出しまたは書き込まれる次の値へのポインタを保存します。
EZI2C_wrProtectS2	第二スレーブアドレスで、読み出し専用のデータへのオフセットを保存します。
EZI2C_bufSizeS2	第二スレーブ アドレスでI^2C マスタに見せられるデータ配列のサイズを保存します。
EZI2C_curState	I^2C ステート マシンの現在の状態を保存します。
EZI2C_curStatus	コンポーネントの現在の状態を保存します。

void EZI2C_Start(void)

機能: これは、コンポーネントの動作を開始する際に推奨される方法です。EZI2C_Start() は initVar 変数を設定し、EZI2C_Init() 関数を呼び出してから、EZI2C_Enable() 関数を呼び出します。これは、I^2C バス処理の前に実行する必要があります。

EZI2C_Start() が EZI2C_Enable() を呼び出した後、EZI2C_Enable() が EZI2C_EnableInt() を呼び出し、I^2C 割り込みが有効になります。

パラメータ: なし

返り値: なし

注意事項: なし

void EZI2C_Stop(void)

機能: この関数は I^2C ハードウェアを無効にし、I^2C 割り込みを無効にします。必要に応じて、アクティブモードパワーテンプレートビットまたはクロックゲーティングを無効にします。

PSoC 3: I^2C バスがデバイスによりロックされていた場合、解放してアイドル状態にします。

パラメータ: なし

返り値: なし

注意事項: なし

void EZI2C_EnableInt(void)

機能: この関数は I²C 割り込みを有効にします。割り込みは、ほとんどの処理で必要になります。EZI2C_Start() API を呼び出したときに、呼び出されます。

パラメータ: なし

返り値: なし

注意事項: なし

void EZI2C_DisableInt(void)

機能: この関数は I²C 割り込みを無効にします。EZI2C_Stop() 関数により割り込みが無効になるため、この関数は通常必要ありません。

パラメータ: なし

返り値: なし

注意事項: I²C の実行中にI²C 割り込みを無効にする場合、I²C バスがロックされることがあります。

void EZI2C_SetAddress1(uint8 address)

機能: この関数は第一メモリバッファの I²Cスレーブアドレスを設定します。この値は 0 ～ 127 の任意の値にすることができます。

パラメータ: address：7 ビットのスレーブのアドレスは、0 ～ 127 です。このアドレスは右揃えで、R/W ビットを含みません。

返り値: なし

注意事項: なし

uint8 EZI2C_GetAddress1(void)

機能: この関数は第一メモリバッファの I²C スレーブ アドレスを返します。

パラメータ: なし

返り値: SetAddress1 で設定したアドレス、または初期値の I²C アドレスと同じ I²C スレーブ アドレスです。

注意事項: なし

void EZI2C_SetBuffer1(uint16 bufSize, uint16 rwBoundry, void * dataPtr)

機能: この関数は、スレーブデータのバッファポインタ、サイズ、読み出し/書き込み領域を設定します。これは I²C マスタに見せられるデータです。

パラメータ: bufSize: バッファサイズ (バイト)

rwBoundry: バッファの開始時に書き込み可能なバイト数を設定します。この値はバッファ サイズ以下でなければなりません。rwBoundry オフセットした位置にあるデータとそこを超えるデータは読み出し専用です。

dataPtr: データバッファへのポインタ

返り値: なし

注意事項: なし

uint8 EZI2C_GetActivity(void)

機能: この関数は、最後にこの関数が呼び出されてから、I²C の読み出しまたは書き込みが発生した場合に、零ではない値を返します。この関数呼び出しの最後で、動作フラグが 0 にリセットされます。

読み出し/書き込みビジー フラグは、読み出し後にクリアされますが、"BUSY"フラグは I²C Stopによってのみクリアされます。

パラメータ: 動作が検出されると、零ではない値が返されます。

返り値: I²C 動作状態

定数	説明
EZI2C_STATUS_READ1	第一アドレスで、読み出しシーケンスが検出された場合にセットされます。ステータスが読み込まれるとクリアされます。
EZI2C_STATUS_WRITE1	第一アドレスで、書き込みシーケンスが検出された場合にセットされます。ステータスが読み込まれるとクリアされます。
EZI2C_STATUS_READ2	第二アドレス (有効の場合) で、読み出しシーケンスが検出された場合にセットされます。ステータスが読み込まれるとクリアされます。
EZI2C_STATUS_WRITE2	第二アドレス (有効の場合) で、書き込みシーケンスが検出された場合にセットされます。ステータスが読み込まれるとクリアされます。
EZI2C_STATUS_BUSY	Startが検出された場合にセットされます。Stopが検出された場合にクリアされます。
EZI2C_STATUS_ERR	I²C ハードウェア エラーが検出された場合にセットされます。ステータスが読み込まれるとクリアされます。

注意事項: なし

void EZI2C_SetAddress2(uint8 address)

機能: この関数は第二メモリバッファの I²C スレーブアドレスを設定します。この値は 0 ～ 127 の任意の値にすることができます。この関数は、ユーザ パラメータで 2 つの I²C アドレスが選択されている場合のみ提供されます。

パラメータ: address：7 ビットのスレーブのアドレスは、0 ～ 127 です。このアドレスは右揃えで、R/W ビットを含みません。

返り値: なし

注意事項: なし

uint8 EZI2C_GetAddress2(void)

機能: この関数は第二メモリバッファの I²C スレーブ アドレスを返します。この関数は、ユーザ パラメータで 2 つの I²C アドレスが選択されている場合のみ提供されます。

パラメータ: なし

返り値: SetAddress2 で設定したアドレス、または初期値の I2C アドレスと同じ I²C スレーブ アドレスです。

注意事項: なし

void EZI2C_SetBuffer2(uint16 bufSize, uint16 rwBoundry, void * dataPtr)

機能: この関数は、第二スレーブ データのバッファポインタ、サイズ、読み出し/書き込み領域を設定します。これは第二 I²C アドレスで I²C マスタに見せられるデータです。この関数は、ユーザ パラメータで 2 つの I²C アドレスが選択されている場合のみ提供されます。

パラメータ: bufSize: I²C マスタに見せられるバッファのサイズです。

rwBoundry: I²C マスタによる読み出しと書き込みが可能なバイト数を設定します。この値はバッファ サイズ以下でなければなりません。rwBoundry オフセットした位置にあるデータとそこを超えるデータは読み出し専用です。

dataPtr: I²C データ バッファで使用されるデータ配列または構造体へのポインタです。

返り値: なし

注意事項: なし

void EZI2C_SlaveSetSleepMode(void)

機能: これは、**Enable wakeup from Sleep Mode** が選択されている場合に、コンポーネントのスリープ準備をするために推奨されるAPIです。この API は I^2C スリープアドレスデコードを有効にします。すべての I^2C トラフィックが完了前に停止するまで待機します。後続の I^2C トラフィックは、デバイスがスリープになるまで NAK にされます。

パラメータ: なし

返り値: なし

注意事項: Wake up from Sleep mode オプションが有効になっている場合、I^2C 割り込みは無効になります (PSoC 3 製品のシリコンの場合のみ)。

void EZI2C_SlaveSetWakeMode(void)

機能: これは、コンポーネントを EZI2C_SlaveSetSleepMode() が呼び出された時の状態に復元する場合に推奨される API です。スリープ EZI2Cスレーブを無効にし、実行時の EZI2C を再び有効にします。スリープから復帰した直後に呼び出す必要があります。この機能は、単一の I^2C アドレスが使用されている場合にのみ提供されます。

パラメータ: なし

返り値: なし

注意事項: Wake up from Sleep mode オプションが有効になっている場合、I^2C 割り込みは有効になります (PSoC 3 製品のシリコンの場合のみ)。

void EZI2C_Sleep(void)

機能: これは、**Enable wakeup from Sleep Mode** が選択されていない場合に、コンポーネントのスリープ準備に推奨される API です。EZI2C_Sleep() API は、現在のコンポーネントの状態を保存します。次に、EZI2C_Stop() 関数と EZI2C_SaveConfig() を呼び出して、ハードウェアの構成を保存します。

CyPmSleep() および CyPmHibernate() 関数を呼び出す前に、EZI2C_Sleep() 関数を呼び出してください。電源管理関数については、『PSoC Creator *System Reference Guide*』を参照してください。

パラメータ: なし

返り値: なし

注意事項: なし

void EZI2C_Wakeup(void)

機能: これは、コンポーネントを EZI2C_Sleep() が呼び出されたときの状態に復元するための推奨 API です。EZI2C_Wakeup() 関数は EZI2C_RestoreConfig() 機能を呼び出し、ハードウェア設定を復元します。EZI2C_Sleep() 関数が呼び出される前にコンポーネントが有効になっている場合、EZI2C_Wakeup() 関数もコンポーネントを再び有効にします。

パラメータ: なし

返り値: なし

注意事項: EZI2C_SaveConfig() または EZI2C_Sleep() の前にこの関数を呼び出すことで、予期しない動作を引き起こす場合があります。

void EZI2C_Init(void)

機能: この関数はConfigureダイアログの設定に従って、コンポーネントを初期化または復元します。EZI2C_Init() を呼び出す必要はありません。EZI2C_Start() API が、このコンポーネント処理を開始するための推奨手段であるこの関数を呼び出すからです。

パラメータ: なし

返り値: なし

注意事項: 全レジスタは、Configureダイアログの設定に従って、値が設定されます。

void EZI2C_Enable(void)

機能: この関数はハードウェアの使用を開始し、コンポーネントの処理を開始します。EZI2C_Enable() を呼び出す必要はありません。EZI2C_Start() API が、このコンポーネント処理を開始するための推奨手段であるこの関数を呼び出すからです。この関数はまた、EZI2C_EnableInt() を呼び出し、I^2C 割り込みを有効にします。

パラメータ: なし

返り値: なし

注意事項: なし

void EZI2C_SaveConfig(void)

機能: この関数は、コンポーネントの設定と保持されないレジスタを保存します。この関数は、Configureダイアログで定義されている、または該当する API で変更される、現在のコンポーネント パラメータ値も保存します。この関数は、EZI2C_Sleep() 関数に呼び出されます。

パラメータ: なし

返り値: なし

注意事項: なし

void EZI2C_RestoreConfig(void)

機能: この関数は、コンポーネントの設定と非保持レジスタを復元します。また、この関数はコンポーネントのパラメータ値を EZI2C_Sleep() 関数を呼び出す前の状態に復元します。

パラメータ: なし

返り値: なし

注意事項: EZI2C_Sleep() または EZI2C_SaveConfig() の前にこの関数を呼び出すことで、予期しない動作を引き起こす場合があります。

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

このコンポーネントは、I^2C アドレスを 1 つまたは 2 つ持つ I^2C スレーブ デバイスをサポートしています。いずれかのアドレスで RAM、EEPROM、またはフラッシュデータ空間のメモリバッファにアクセスできます。EEPROM とフラッシュメモリバッファは読み出し専用ですが、RAM バッファは読み出し/書き込みが可能です。アドレスは右揃えです。

このコンポーネントを使用する場合、I^2C ハードウェアが割り込み駆動であるため、グローバル割り込みを有効にする必要があります。このユーザモジュールは割り込みを必要としますが、ISR (割り込みサービスルーチン) にコードを追加する必要はありません。モジュールは、ユーザコードから独立してすべての割り込み(データ転送)を担います。このインタフェースに割り当てられたメモリバッファは、アプリケーションと I^2C マスタ間のシンプルなデュアルポートメモリのように見えます。

必要に応じ、データ構造体にセマフォとコマンドを定義して、マスタとスレーブ間のより高級インタフェースを作成できます。

メモリ インタフェース

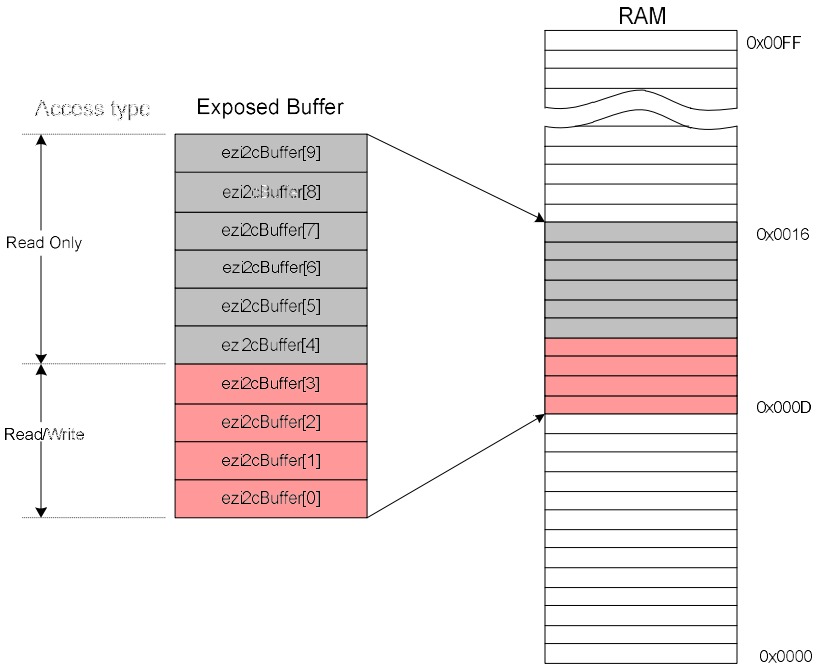
I^2C マスタへのインタフェースは、一般的な I^2C EEPROM のインタフェースとよく似ています。EZI2C インタフェースは単純な変数、配列または構造体として設定することが可能です。ある意味でこれは、ユーザプログラムと I^2C マスタ間を I^2C バスで結ぶ 1 つまたは 2 つの共有メモリインタフェースのように動作します。コンポーネントは、I^2C マスタがメモリの特定の領域にアクセスすることだけを許可し、その領域外での読み出しや書き込みを防止し

ます。例えば、スレーブの 第一アドレスのバッファが以下のコード例のように設定されている場合、メモリマップに示されるバッファは図 1 のように表現されます。

```
#define BUFFER_SIZE (0x0Au)
#define BUFFER_RW_AREA_SIZE (0x04u)

EZI2C_SetBuffer1(BUFFER_SIZE, BUFFER_RW_AREA_SIZE, (void *) ezi2cBuffer);
```

図 1. I²C マスタに見せられる EZI2C バッファのメモリマップ



インタフェース (I²C マスタ) は、バイト配列としての構造体しか見ないので、定義された領域外のメモリにはアクセスできません。上の構造体を例に取る場合、提供される API はデータ構造体を I²C インタフェースに晒します。

```
char ezi2cBuffer2[15u];
EZI2C_SetBuffer2(15u, 8u, (void *) ezi2cBuffer2);
```

データは、異なるアーキテクチャに対して、異なるエンディアンで送信されます。エンディアンは、16、32 または 64 ビットワードの中のバイトの順番を指しています。そのため、送信データの型が 1 バイトより大きい場合は、特定のエンディアンで送信するために、さらにコードが必要になります。例えば、CY_GET_REGXX()/ CY_SET_REGXX() マクロ (XX は 16/24/32) は、デバイスのアーキテクチャにかかわらず、リトルエンディアンのワードに使用できます。エンディアンについての詳細は、*System Reference Guide* の"Register Access"のセクションを参照してください。

次の単純な例では、1 つの整数 (2 バイト) のみが見せられています。両バイトとも、I²C マスタによる読み出しと書き込みが可能です。

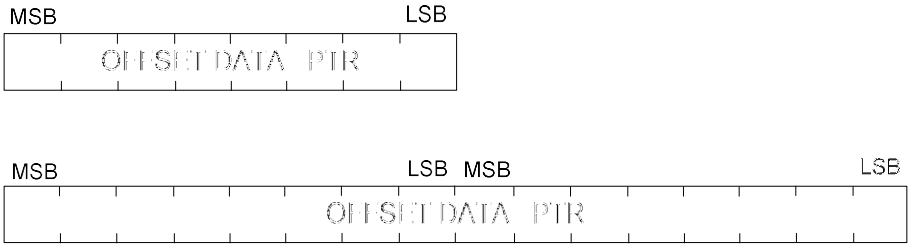
```
uint16 ezi2cVariable1;
```

```
CY_SET_REG16(&ezi2cVariable1, 0xABCD);
EZI2C_SetBuffer1(2u, 2u, (void *) (&ezi2cVariable1));
```

外部マスタから見たインタフェース

The EZI2C スレーブのコンポーネントは、読み出し/ 書き込み領域に対する基本的な読み出し・書き込み処理、および読み出し専用領域の読み出し専用処理をサポートします。2 つの I^2C アドレスインタフェースは、個別のオフセットデータポインタでアドレス指定される個別のデータバッファを含みます。オフセットデータポインタは、書き込み処理の最初の 1 つまたは 2 つのデータバイトとしてマスタによって書き込まれます。1 バイトか 2 バイトかは **Sub-address size** パラメータに依存します。以下の議論では、8 ビットの Sub-address size に専念することにします。

図 2. 8 ビットおよび 16 ビットのサブアドレス サイズ

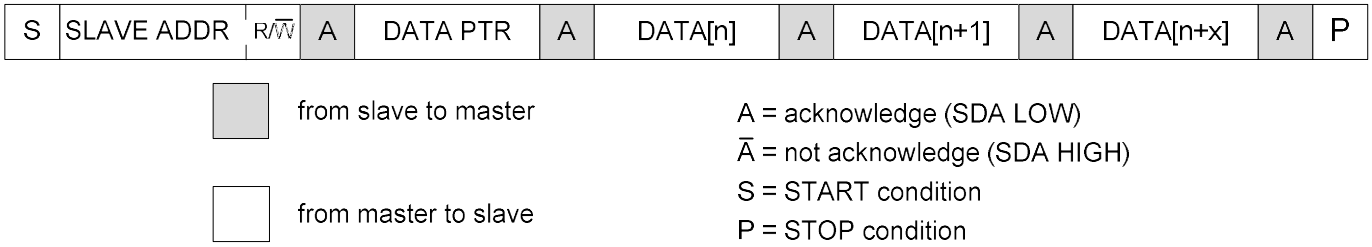


書き込み処理の場合、最初のデータ バイトは常にオフセットデータポインタです。(**サブアドレス サイズ** = 16 の場合 2 バイト) オフセットデータポインタに続くバイトが、オフセットデータポインタで示される位置に書き込まれます。次のデータバイトは、オフセットデータポインタ +1 に書き込まれるというように、書き込みが完了するまで続きます。書き込み処理の長さは、最大バッファ読み出し/ 書き込み領域サイズによってのみ制限されます。書き込み処理の場合、オフセットデータポインタは提供される必要があります。

読み出し処理は、必ず最新の書き込み処理で提供されたオフセットデータポインタから始まります。オフセットデータポインタは、各バイトを読み出すごとに +1 されます。これは書き込み処理と同様です。新しい読み出し処理は、前回読み出し処理が停止した位置から継続して処理されません。新しい読み出し処理は、必ず前回の書き込み処理のオフセットデータポインタで示された位置からデータ読み出しが始まります。読み出し処理の長さはデータバッファの最大サイズによってのみ制限されます。

典型的には、読み出しは、リスタート(あるいはストップ/ スタート)が後続する、オフセットデータポインターだけの書き込み処理とそれに続く読み出し処理が含まれます。繰り返し同じデータを読み出す場合のように、オフセットデータポインタが更新を要求しない場合、書き込み処理は最初の処理に続いて要求されることはありません。これによって、直接データを連続させることができ、大幅に読み出し処理が速くなります。

図 3. I²C スレーブへの (x+1) バイトの書き込み



例えば、オフセットデータポインタが 4 に設定されている場合、読み出し処理は 4 の位置からデータの読み出しを開始し、データの終端か、ホストが読み出し処理を完了するまで連続して読み出しを継続します。これは、単一または複数のいずれの読み出し処理が実行される場合でも同じです。オフセットデータポインタは、新しい書き込み処理が開始されるまで変更されません。

I²C マスタが、EZI2C_SetBuffer1() または EZI2C_SetBuffer2() 関数で指定されている領域を超えるデータの書き込みを試みると、データが破棄されるため、RAM 内部または指定された RAM 領域外には影響を与えません。データは許可されている範囲外から読み出すことはできません。マスタから、許可されている範囲外からの読み出しが要求されると、無効の返り値が返されます。

図 4 は 8 ビットのオフセットデータポインタのデータポインタ書き込みを示します。

図 4. スレーブデータポインタの設定

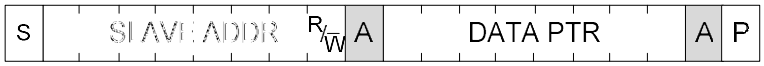
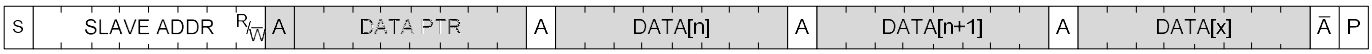


図 5 は 8 ビットのオフセットデータポインタの読み出し処理を示します。データ書き込み処理は、オフセット データポインタを常に書き換えるので注意が必要です。

図 5. I²C スレーブから (x+1) バイトの読み出し



リセットまたは電源投入時に、EZI2C スレーブコンポーネントが構成され、API が供給されますが、リソースは EZI2C_Start() 関数を使って、明示的にオンにする必要があります。

I²C バスの詳細な説明と実装は、I²C specification として NXP® のウェブサイトに掲載されています。またデバイスデータシートにも記載されています。

データの一貫性

データバッファは 1 バイトを超えるデータ構造体を含みますが、マスタの読み出しまたは書き込み処理は複数の 1 バイト処理で構成されています。これはデータの一貫性の問題を引き起こす可能性があります。なぜならマルチバイトの読み出しまたは書き込みでは両方のインターフェイス(マスタとスレーブ)の同期を保証するメカニズムが存在しないからです。例えば、1 つの 2 バイト整数を含むバッファを考えてみてください。マスタは 2 バイト整数を 1

バイトづつ読み出します。マスタが整数の最初のバイト (LSB) を読み出し、次のバイト (MSB) を読み出そうとしている時に、スレーブが整数全体を更新してしまっている場合があります。LSB が本来のデータから読み出され、MSB が更新された値から読み出されたので、マスターによって読み出されたデータは無効かもしれません。

相手側がデータの読み取りまたは書き込みを行っている間、マスタまたはスレーブからの更新が発生しないように保証する仕組みをマスタ、スレーブ、またはその両方に提供する必要があります。EZI2C_GetActivity() 関数を使用してアプリケーション固有の仕組みを開発することができます。

スリープモードからの復帰

デバイスクロックの設定(バスクロック周波数)はスリープモード移行手順の一部として CyPmSaveClocks() 関数により変更できます。I^2C トランザクションがアクティブモードで継続できるようにするには、事前に CyPmRestoreClocks() 関数で復元しておく必要があります。

これらの要件を満たすには、I^2C 割り込みを EZI2C_SlaveSetSleepMode() で無効にし、EZI2C_SlaveSetWakeMode() で有効にします。結果的にハードウェア アドレスの一致イベントが発生すると、トランザクションは SCL ラインを LOW (クロック期間延長手順)に保持することで一時停止します。

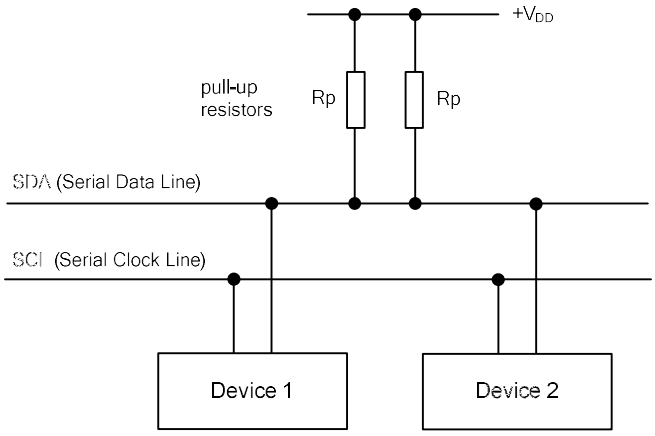
以下に正しいスリープモードへの移行手順を示します。これは、**Enable wake up from Sleep mode** が有効である場合です。

```
/* EZI2C がスリープモードから復帰するように準備 */
        EZI2C_SlaveSetSleepMode();
/* スリープモードに切り替え */
        CyPmSaveClocks();
        CyPmSleep(PM_SLEEP_TIME_NONE, PM_SLEEP_SRC_I2C);
        CyPmRestoreClocks();
/* EZI2C がアクティブモードで動作するように準備 */
        EZI2C_SlaveSetWakeMode();
```

外部電気接続

図 6 に示すように、I^2C バスには外部プルアップ抵抗が必要です。プルアップ抵抗(R_P)は、電源電圧、クロック速度およびバスの静電容量に応じて決定されます。任意のデバイス(マスタまたはスレーブ)の最小のシンク電流は、出力電圧 V_{OLmax} = 0.4V で 3mA 以上必要です。これは、5V システムの最小プルアップ抵抗値をおよそ 1.5kΩ に制限します。R_P の最大値は、バスの静電容量とクロック速度によって異なります。バスの静電容量が 150pF である 5V システムの場合、プルアップ抵抗が 6kΩ 以内である必要があります。詳細については NXP® ウェブサイト www.nxp.com の *The I^2C-Bus Specification を参照ください。*

図 6. I²C バスへの接続



注 I²C コンポーネントをサイプレスまたはサブライセンスを持つ関連業者から購入すると、Philips I²C の特許権の下でライセンスが付与されます。このライセンスにより、システムが Philips の定義する I²C の標準仕様に従う限り、I²C システムでこれらのコンポーネントを使用できます。2006年 10月 1日現在、Philips Semiconductors 社は新社名 NXP Semiconductors を使用しています。

割り込みサービスルーチン

割り込みサービスルーチンはコンポーネントコードによって使用されます。変更しないでください。

DC/ AC 電気的特性

EZI2C DC 仕様

記号	項目	条件	Min	Typ	Max	単位
	ブロック消費電流	有効、100kbpsに設定	–	–	250	µA
		有効、400kbpsに設定	–	–	260	µA
		スリープモードから復帰	–	–	30	µA

EZI2C AC 仕様

記号	項目	条件	Min	Typ	Max	単位
	ビットレート		–	–	1	Mbps

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
1.61	カスタマイザ内に構成された拡張検査オプションと関連する**スリープ モードからのウェークアップ有効化**オプション。	コンポーネントがサポートしていないモードで構成されるのを回避します。
	PSoC 3 デバイスの EZI2C_Stop() 実装を更新しました。	ロック時、EZI2C_Stop() のバスをリリースします。
	初期値の I2C アドレスを 8 および 9 に更新し、I2C バス仕様要件に準拠しました。	以前に使用されたアドレスを I²C バス仕様に従って予約します。
	コンポーネントのデバッガ ツール ウィンドウのサポートを更新しました。	デバッガ ツール ウィンドウのサポートを拡張しました。
	.cyre ファイルに関数名を追加することで、PSoC 3 が再入可能となるような各関数に宣言機能を追加しました。	すべての API が真に再入可能とは限りません。コンポーネント API ソースファイルのコメントは、どの関数が本当に再入可能にできないのかを示しています。 この変更には、安全な(フラグまたはクリティカル セクションによる並行コールから保護された)方法で、使用される再入可能ではない関数のコンパイラの警告を制限する必要があります。
1.60.b	データ シートの訂正	
1.60.a	スリープ モードからウェークアップ有効化とアドレス オプション番号との依存関係情報を用いてピン接続を更新しました。	オプションがスリープ モードからウェークアップ有効化モードを補足し、かつ 1 つの I²C アドレスがアドレス オプション番号で選択される場合に限り使用できることも説明しました。
	図 2、3 および 5 はビット フィールドを示すように更新されました。	表示機能の拡張、
	PSoC のデバイス アーキテクチャーにかかわらず、移植可能コードの書き込み方法を明確化しました。	文書の拡充
1.60	スレーブ有効化ビットで動作する方法を次のように変更しました。EZI2C_Stop() はもはやこのビットをクリアせず、このビットは EZI2C_Enable() から EZI2C_Init() に移動したものと設定されます。I²C の構成レジスタがこれで EZI2C_RestoreConfig() 関数に復元されます。	EZI2C_Start() - EZI2C_Stop() - EZI2C_Start() と EZI2C_Sleep() - EZI2C_Wakeup() シーケンスの正しい結果を得るために。機能的な影響は予想されません。
	ラベル I2C バス速度(カスタマイザ内)をデータ転送速度で置換しました。スリープ モードからウェークアップ セクションを機能説明に追加しました。	I²C バス仕様の表記法と I²C/EZI2C コンポーネント間の一貫性。
	カスタマイザの「I2C pins connected to」ラベルを「Pin Connections」に置換しました。	要件を用いて一貫性に関するテキストを修正しました。

バージョン	変更の説明	変更の理由 / 影響
	カスタマイザの「Enable wakeup from the Sleep mode」ラベルを「Enable wakeup from Sleep mode」に置換しました。	要件を用いて一貫性に関するテキストを修正しました。
	コンポーネント記号とカタログ配置名を更新:「EZ I2C」から「EZI2C」へ名前を変更しました。	要件を用いて一貫性に関するテキストを修正しました。
	記号と ISR の両方に使用されるグローバル変数がコンパイラで最適化される可能性がある場合の問題を修正しました。	予期せぬ結果をもたらす可能性がある最適化問題を防止します。
	データシートに特性データを追加	
	データシートのマイナーな編集と更新	
1.50.a	コンポーネントをコンポーネント カタログのサブフォルダに移動	
1.50	標準データ転送速度を更新し、最大 1 Mbps までをサポート	I^2C バス速度の設定を最大 1 Mbps まで実現します。
	Keil 社の再入可能のサポートを追加しました。	Keil 社のコンパイラを用いた PSoC 3 が複数のコントロールフローから呼び出される関数をサポートします。
	Sleep/Wakeup と Init/Enable API を追加しました。	ローパワー モードをサポートし、ほとんどのコンポーネントの初期化と有効化の制御を分離する共通インターフェースを提供するために。
	コンポーネントの XML 記述を追加しました。	これにより、PSoC Creator がこのコンポーネント用に新規デバッガ ツール ウィンドウの作成メカニズムを提供することが可能になります。
	PSoC 3 プロダクションデバイスのサポートを追加しました。	必要な変更は PSoC 3 ES2 とプロダクションデバイス間のハードウェア変更をサポートして適用済みです。
	初期設定の回路図テンプレートをコンポーネント カタログに追加しました。	各コンポーネントには回路図テンプレートが入っています。
	EZI2C のバス速度生成を修正しました。あらかじめ必要な値より大きい x4 に設定済みです。バス速度の計算を記述するソース コードにさらにコメントを追加しました。	正常な I^2C バス速度の計算および生成
	Microsoft Windows 7 用にフォームの高さを最適化しました。	Windows 7 では、カスタマイザ開始直後にスクロールバーが表示されます。
	「Use 0x prefix for hexadecimals」と表示したアドレス入力ボックスにツールチップを追加しました。	16 進を入力する可能性がある場合はユーザーに通知。
1.20.a	コンポーネントをコンポーネント カタログのサブフォルダに移動しました。	

バージョン	変更の説明	変更の理由 / 影響
	シリコン リビジョンとの互換性について知らせる情報をコンポーネントに追加しました。	このツールは、コンポーネントが不整合のシリコン上で使用された場合にエラー/警告を発します。エラーが表示されたら、対象デバイスをサポートするリビジョンにアップデートしてください。
1.20	[Configure] ダイアログが更新されました。	
	デジタル ポートを回路図のピン コンポーネントに変更しました。	

Copyright © 2005-2012 Cypress Semiconductor Corporation 本文書に記載される情報は、予告なく変更される場合があります。Cypress Semiconductor Corporationは、サイプレス製品に組み込まれた回路以外のいかなる回路を使用することに対しても一切の責任を負いません。特許又はその他の権限下で、ライセンスを譲渡又は暗示することもありません。サイプレス製品は、サイプレスとの書面による合意に基づくものでない限り、医療、生命維持、救命、重要な管理、又は安全の用途のために仕様することを保証するものではなく、また使用することを意図したものでもありません。さらにサイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことを合理的に予想される、生命維持システムの重要なコンポーネンツとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。

PSoC Designer™及びProgrammable System-on-Chip™は、Cypress Semiconductor Corp.の商標、PSoC®は同社の登録商標です。本文書で言及するその他全ての商標又は登録商標は各社の所有物です。

全てのソースコード(ソフトウェア及び/又はファームウェア)はCypress Semiconductor Corporation (以下「サイプレス」)が所有し、全世界(米国及びその他の国)の特許権保護、米国の著作権法並びに国際協定の条項により保護され、かつそれらに従います。サイプレスが本書面によるライセンシーに付与するライセンスは、個人的、非独占的かつ譲渡不能のライセンスであって、適用される契約で指定されたサイプレスの集積回路と併用されるライセンシーの製品のみをサポートするカスタムソフトウェア及び/又はカスタムファームウェアを作成する目的に限って、サイプレスのソースコードの派生著作物を複製、使用、変更、そして作成するためのライセンス、並びにサイプレスのソースコード及び派生著作物をコンパイルするためのライセンスです。上記で指定された場合を除き、サイプレスの書面による明示的な許可なくして本ソースコードを複製、変更、変換、コンパイル、又は表示することは全て禁止されます。

免責条項:サイプレスは、明示的又は黙示的を問わず、本資料に関するいかなる種類の保証も行いません。これには、商品性又は特定目的への適合性の黙示的な保証が含まれますが、これに限定されません。サイプレスは、本文書に記載される資料に対して今後予告なく変更を加える権利を留保します。サイプレスは、本文書に記載されるいかなる製品又は回路を適用又は使用したことによって生ずるいかなる責任も負いません。サイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことが合理的に予想される生命維持システムの重要なコンポーネンツとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。

ソフトウェアの使用は、適用されるサイプレスソフトウェアライセンス契約によって制限され、かつ制約される場合があります。