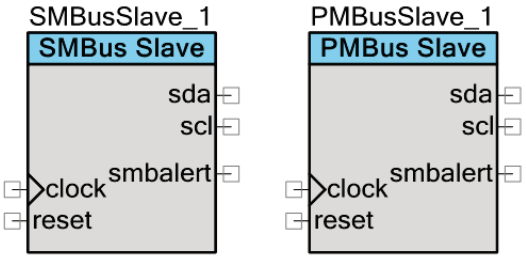
SMBus/ PMBus スレーブ

1.0



特長

- SMBus スレーブモード
- PMBus スレーブモード
- SMBALERT#ピンをサポート
- 25ms のタイムアウト
- 専用機能ブロック(Fixed-function: FF)実装と UDB 実装
- 構成可能な SM/PM Bus コマンド

概要

システム管理バス(SMBus)および電力管理バス(PMBus)スレーブコンポーネントは、SMBus または PMBus のプロトコルを実行する PSoC 3 または PSoC 5 の設計に、I^2C 物理レイヤインターフェースを簡単に追加する方法を提供します。

SMBus は、システムホストと通信できるさまざまなシステム管理チップとの 2 線式インターフェースです。物理レイヤとして、I^2C を使用しています。SMBus スレーブコンポーネントは SMBus スレーブデバイス仕様の大半を実装しており、スレーブデバイスのパラメータを設定できるようにしています。スレーブデバイスは、API を使用して SMBus マスターと通信することができます。

PMBus プロトコルは、一般的な SMBus プロトコルを特化させ実装したものです。PMBus では、コンポーネントはすべての使用可能な PMBus コマンドを提示し、アプリケーションに関連するコマンドを選択することができます。

SMBus/ PMBus スレーブの用途

このコンポーネントは、SMBus または PMBus スレーブ通信インターフェースを必要とする設計に使用できます。このコンポーネントが、ハードウェアの物理レイヤでの要件の大半を処理します。ファームウェアは、プロトコルやメモリバッファ管理を処理します。また、I^2C との間でのデータ転送も管理します。

出入力の接続

このセクションでは、SMBus スレーブの入出力接続について説明します。I/O 項目のアスタリスク(*)はその I/O が、説明に挙げられた条件において、回路シンボルに表示されない場合があることを示します。

clock – 入力

クロック入力は、I²C SCL/SDAスタッククローのタイムアウトタイマーのクロックソースとして使用します。ImplementationパラメータをUDBに設定する場合、16倍オーバーサンプリングできるクロックが必要です。

クロック入力は I²C **Implementation** パラメータが **UDB** に設定されている場合に使用できます。

データレート	クロック
10kbps	160kHz
50kbps	800kHz
100kbps	1.6MHz
400kbps	6.4MHz
1000kbps	16MHz

reset – 入力

UDB I²C 実装のハードウェアリセット。アクティブハイリセットピンがハイレベルに保持されている場合、I²C ブロックはリセットを継続し I²C の通信が停止します。SDA および SCL は強制的にハイレベルになります。これはハードウェアリセット専用です。ソフトウェアは Stop()および Start() API を使用して個別にリセットする必要があります。

sda (SMBDAT) – 入力/出力

シリアル データ(SDA)は、I²C データ信号です。これはバスデータを送受信するために使用される双方向データ信号です。sda に接続されたピンは Open-Drain-Drives-Low として設定する必要があります。カスタマイザーで"External I/O Option"を選択すると、sda 双方向信号が個別の入力および出力(sda_o および sda_i)に置換されます。これは、一部のアプリケーションで必要となる複数の I²C バスの多重化を有効にするために必要です。

scl (SMBCLK) – 入力/出力

シリアルクロック(SCL)はマスタで生成される I²C 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、クロックをローレベルに保ち、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が ACK/NAK になるまで、バスをストールすることがあります。scl に接続されたピンは Open-Drain-Drives-Low として設定する必要があります。カスタマイザーで"External I/O Option"を選択すると、scl 双方向信号が個別の入力および出力(scl_o および scl_i)に置換されます。これは、一部のアプリケーションで必要となる複数の I²C バスの多重化を有効にするために必要です。

smbalert (SMBALERT#) – 出力

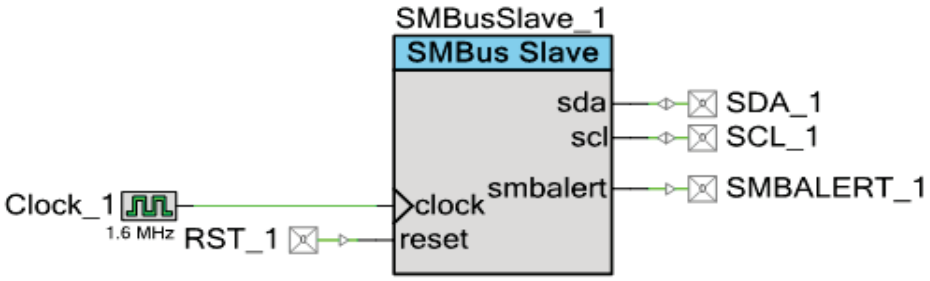
アラートは、SMBus で SMBALERT#と定義されるオプションピンです。この信号は選択して有効にできます。アラートピンは、API 経由でアサート、デアサートできます。smbalert に接続されたピンは Open-Drain-Drives-Low として設定する必要があります。

回路図マクロ情報

初期設定では、PSoC Creator コンポーネントカタログに SM/PM Bus スレーブコンポーネント用の 2 つの回路図マクロ実装があります。これらのマクロには、既に接続済みで設定済みのピン、データレートを定義する内部的に設定されたクロック値が含まれます。回路マクロは、FF の I^2C ブロックやハードウェアアドレスデコードを使用するように設定された I^2C スレーブコンポーネントを使用します。

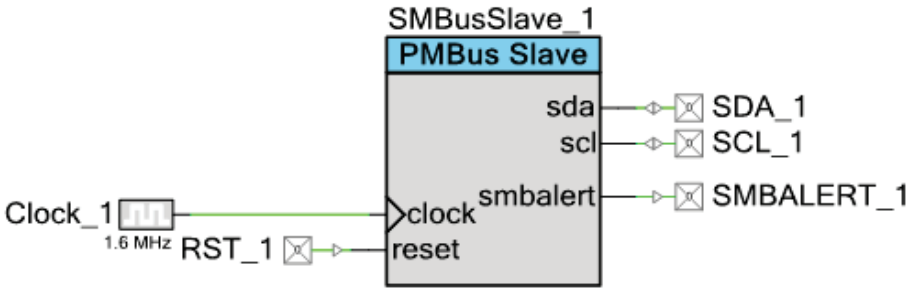
SMBus スレーブマクロ

このマクロは、SMBus モードで SM/PM Bus スレーブコンポーネントを正しく設定します。SCL および SDA 端子に接続されたピンは、ドライブモードを Open Drain Drives Low に設定した双方向として設定します。このコンポーネントは、データレートを 100kHz と定義します。



PMBus スレーブマクロ

このマクロは、PMBus モードで SM/PM Bus スレーブコンポーネントを正しく設定します。SCL および SDA 端子に接続されたピンは、ドライブモードを Open Drain Drives Low に設定した双方向として設定します。このコンポーネントは、データレートを 400kHz と定義します。



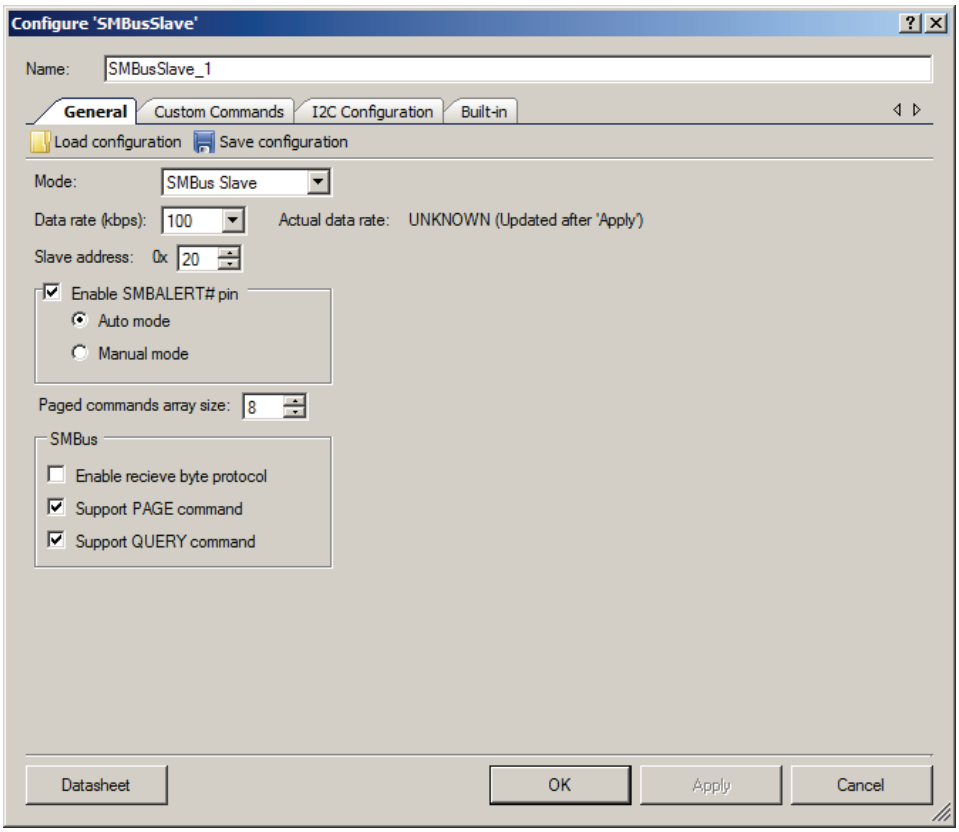
コンポーネントパラメータ

SM/PM Bus スレーブコンポーネントを回路図の上にドラッグし、ダブルクリックして **Configure** ダイアログを開きます。初期設定では、**Configure** ダイアログに、General タブが図 1 のように表示されます。

General タブ

General タブには、SM/PM Bus の全般的な設定項目があります。以下のパラメータが使用可能です。

図 1. General タブ



Export/Import Configuration

Export/Import Configuration では、カスタマイザー設定を外部ファイルに保存および復元することができます。この機能により、プリセットプロファイルを簡単に読み込み、カスタム設定を保持することができます。

Mode

Mode パラメータは、コンポーネントのモードを、SMBus スレーブモードまたは PMBus スレーブモードのどちらかに設定します。SMBus スレーブを選択した場合、PMBus Commands タブは無効になります。また、**Mode** によって使用できるデータレートも決まります。PMBus スレーブモードには、100 および 400kHz の 2 つの設定があ

ります。100 および 400kHz。SMBus スレーブモードでは、さらに 10 および 50kHz の設定があります。このパラメータの初期設定は SMBus スレーブモードです。

Data Rate

Data Rate パラメータは、I^2C データレートを選択するために使用します。選択可能な設定は、SM/PM Bus**Mode** の選択によって異なります。PMBus スレーブモードでは、**Data Rate** の値には 100kHz および 400kHz が設定できます。SMBus スレーブモードでは、10kHz、50kHz、100kHz、400kHz の設定があります。**Data Rate** パラメータの初期値は 100kHz です。

Slave address

Slave address パラメータは、デバイスの I^2C アドレス(7 ビット形式)を決定します。値は 10 進または 16 進数("0x"を前置)で入力します。カスタマイザーは、アドレスが SMBus スレーブ予約リストにあるアドレスと競合していないことを確認します。**Slave address** の初期値は 0x20 です。

SMBALERT

SMBALERT ボックスでは、ホストに通知する SMBALERT# 出力を構成できます。**Enable SMBALERT# pin** を選択すると、ピンがコンポーネントシンボルに出力として表示されます。Auto/Manual ボタンは、ホストがアラート応答アドレス(SetSmbAlertMode() API の GUI での表現)でデバイスを照会したときに、**SMBALERT# ピン**が自動的にデアサートされるかを決めます。このパラメータの初期設定は EnabledAuto です。

Paged Commands

このパラメータボックスは、PMBus の PAGE コマンドを設定します。**Maximum page** パラメータは、ページコマンドの配列サイズを決定します。すべてのページコマンドが、この配列サイズを共有します。初期値は 8 ページです。**有効な範囲は、1～32** です。

SMBus ボックス

SMBus ボックスは、SMBus の選択機能を設定します。SMBus スレーブモードでのみ表示されます。

- **Enable receives the byte protocol** チェックボックスは、SMBus バイト受信プロトコルのサポートを有効にします。選択を解除すると、バイト受信のトランザクションはバスエラーとして処理されます。選択すると、コンポーネントは SMBus_GetReceiveByteResponse() API を呼び出して、バイト受信要求に対する応答バイトを決定します。
- **Support PAGE Command** チェックボックスでは、SMBus スレーブモードで PAGE コマンドにアクセスできます。
- **Support QUERY Command** チェックボックスでは、SMBus スレーブモードで QUERY コマンドにアクセスできます。

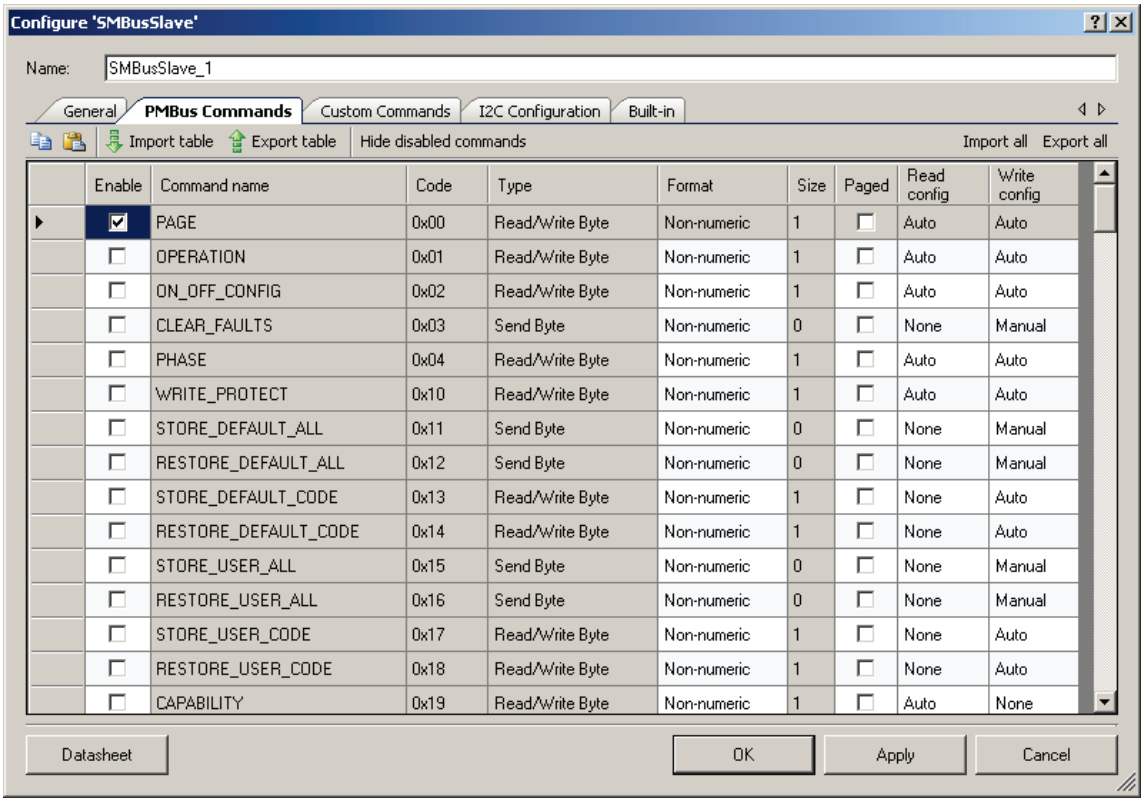
PAGE または QUERY コマンドのどちらかが有効であれば、これらのコマンドが **Custom Commands** タブに追加されます。これらのコマンドのプロパティは、PMBus の仕様に基づいていますが、コマンドコードを完全にカスタマイズすることも可能です。

このボックスの初期設定：**Enable receives byte protocol** は無効、**Support PAGE Command** は有効、**Command Code**=0x00、**Support QUERY command** は有効、**Command Code**=0x1A。

PM Bus Commands タブ

PM Bus Commands タブは、コンポーネントの **Mode** が PMBus スレーブに設定されている場合に使用可能です。タブには、PMBus 仕様に定義されたコマンドのリスト全体が表示されます。記入済みの情報は、コマンド名、そのコマンドコードの数値、およびコマンドのタイプです。コンポーネントのインスタンスが処理するコマンドを有効化/無効化することができます。Name、Code、Type は読み取り専用フィールドです。このタブで使用可能なパラメータは以下の通りです。

図 2. PMBus Commands タブ



Enable	Command name	Code	Type	Format	Size	Paged	Read config	Write config
☑	PAGE	0x00	Read/Write Byte	Non-numeric	1	☐	Auto	Auto
☐	OPERATION	0x01	Read/Write Byte	Non-numeric	1	☐	Auto	Auto
☐	ON_OFF_CONFIG	0x02	Read/Write Byte	Non-numeric	1	☐	Auto	Auto
☐	CLEAR_FAULTS	0x03	Send Byte	Non-numeric	0	☐	None	Manual
☐	PHASE	0x04	Read/Write Byte	Non-numeric	1	☐	Auto	Auto
☐	WRITE_PROTECT	0x10	Read/Write Byte	Non-numeric	1	☐	Auto	Auto
☐	STORE_DEFAULT_ALL	0x11	Send Byte	Non-numeric	0	☐	None	Manual
☐	RESTORE_DEFAULT_ALL	0x12	Send Byte	Non-numeric	0	☐	None	Manual
☐	STORE_DEFAULT_CODE	0x13	Read/Write Byte	Non-numeric	1	☐	None	Auto
☐	RESTORE_DEFAULT_CODE	0x14	Read/Write Byte	Non-numeric	1	☐	None	Auto
☐	STORE_USER_ALL	0x15	Send Byte	Non-numeric	0	☐	None	Manual
☐	RESTORE_USER_ALL	0x16	Send Byte	Non-numeric	0	☐	None	Manual
☐	STORE_USER_CODE	0x17	Read/Write Byte	Non-numeric	1	☐	None	Auto
☐	RESTORE_USER_CODE	0x18	Read/Write Byte	Non-numeric	1	☐	None	Auto
☐	CAPABILITY	0x19	Read/Write Byte	Non-numeric	1	☐	Auto	None

Format

Format パラメータは、このコマンドの数値フォーマットを指定します。このフォーマットは、QUERY コマンドへの応答で記述されるコンポーネントで使用されます。QUERY コマンドで使用できるフォーマットの値は、Linear(直

線)、Signed(符号付)、Direct(直接)、Unsigned(符号なし)、VID モードです。コンポーネントは実際に数値変換を行うわけではないので、このフィールドは QUERY コマンドの目的でのみ使用されます。

Size

すべてのブロックおよびプロセスコールのタイプのコマンドでは、**Size** フィールドを編集してデータ要素のサイズを指定できます。このサイズには、SMBus プロトコルがブロック転送の先頭に追加するサイズ/カウントバイトは含まれません。このフィールドが編集できるのは、ブロックまたはプロセスコールのコマンドだけです。すべての他のタイプでは、**Size** フィールドは PMBus の仕様で決まります。初期値は 16 です。

Read/Write Config

それぞれのコマンドについて、そのコマンドが読み取り可能、書き込み可能であるかを **Read Config** および **Write Config** パラメータで選択できます。それぞれについて、None(なし)、Auto(自動)、Manual(手動)のいずれかを選択します。None はアクションが無効(即ち読み取り専用コマンドに対する書き込み設定を None に設定)であることを意味します。"Auto"モードコマンドは、完全にコンポーネントによって処理されます。レジスタストアと I^2C 転送バッファの間で、ユーザーファームウェアの介入、通知なくで転送されます。"Manual"コマンドは、トランザクションキューに追加され、必ずユーザーファームウェアで処理される必要があります。これらのパラメータの初期値は"Manual"です。

注: 書き込みおよび読み取りの非対称性や、プロセスコールプロトコルの複雑性から、このプロトコルを使用するコマンドでは Auto モードは選択できません。

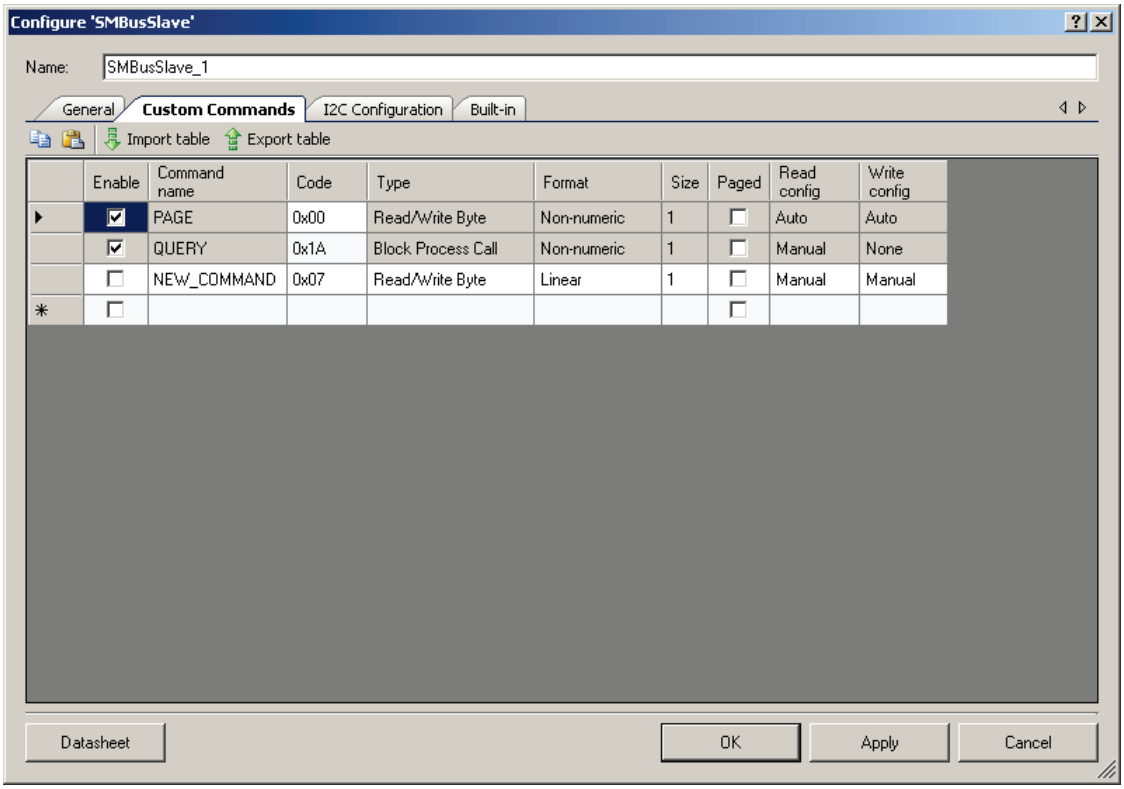
Paged

Paged チェックボックスは、そのコマンドがページ化(例、インデックス化)されるかどうかを示します。"Paged"と指定されるコマンドに対して、コンポーネントはそのコマンド用の配列をレジスタストア内に自動的に生成します。配列のサイズは、**Maximum Page** パラメータによって決まります。自動読み取りおよび書き込みでは、コンポーネントは現在の PMBus ページ(最後のページコマンドで選択される)に基づいて、ページパラメータの正しい配列に自動的にインデックス付けされます。このパラメータは初期設定では無効になっています。

Custom Commands タブ

このタブでは、コマンド名、コード、タイプフィールドの変更やカスタマイズが可能です。

図 3. Custom Commands タブ



Command Name

これは、ユーザーが指定するコマンドの名前です。使用可能な文字は、A-Z (すべての caps)、0-9、および下線 "_"です。最長で 24 文字です。最初の文字は数字であってはいけません。コマンド名は重複できません(標準の PMBus コマンド名と重複するカスタムコマンド名を含めて)。初期設定では、**Command Name** は空白です。

Command Code

これはそのコマンドの数値コードです。16 進数で、2 文字(0-9、ssA-F)に限定されます。コマンドコードは重複できません。これには、有効な PMBus コマンドと競合するコマンドコードも含まれます。初期設定では、**Command Code** は空白です。

Type

Type は、このコマンドで使用する SMBus 転送プロトコル/データサイズを指定します。設定可能な値は、送信バイト、読み取り/書き込みバイト、読み取り/書き込みワード、読み取り/書き込みブロック、プロセスコール、およびブロックプロセスコールです。初期設定では、読み取り/書き込みバイトに設定されます。

Format

Format パラメータは、このコマンドの数値フォーマットを指定します。このフォーマットは、QUERY コマンドへの応答で記述されるコンポーネントで使用されます。QUERY コマンドで使用できるフォーマットの値は、Linear(直線)、Signed(符号付)、Direct(直接)、Unsigned(符号なし)、VID モードです。コンポーネントは実際に数値変換を行うわけではないので、このフィールドは QUERY コマンドの目的でのみ使用されます。

Read/Write Config

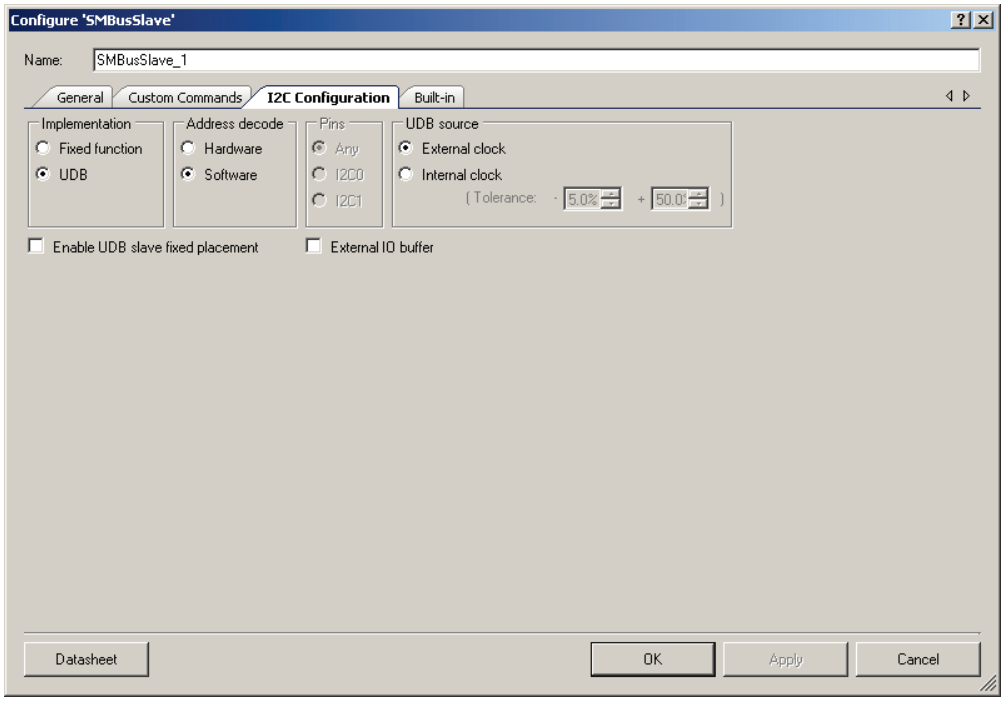
それぞれのコマンドについて、そのコマンドが読み取り可能および/または書き込み可能であるかを **Read Config** および **Write Config** パラメータで選択できます。それぞれについて、None(なし)、Auto(自動)、Manual(手動)のいずれかを選択します。"None" はアクションが無効(即ち読み取り専用コマンドに対する書き込み設定を None に設定)であることを意味します。"Auto"モードコマンドは、完全にコンポーネントによって処理されます。レジスタストアと I²C 転送バッファの間で、ユーザーファームウェアの介入、通知なく転送されます。"Manual"コマンドは、トランザクションキューに追加され、必ずユーザーファームウェアで処理される必要があります。これらのパラメータの初期値は "Manual"です。

注: 書き込みおよび読み取りの非対称性や、プロセスコールプロトコルの複雑性から、このプロトコルを使用するコマンドでは自動モードは選択できません。

I²C Configuration タブ

このタブでは、I²C ハードウェアを設定できます。

図 4. I²C Configuration タブ



Implementation

このパラメータは、I²CハードウェアをFF[1](Fixed-function、専用機能ブロック)またはUDBを使用して実装するかどうかを決定します。初期設定はUDBです。

Address decode

このパラメータはアドレスデコードをソフトウェアによるかハードウェアによるかを選択します。付属 API が十分であって、スレーブアドレスが 1 つだけ必要な、大半のアプリケーションでは、ハードウェアによるアドレスデコードを推奨します。ソースコードを修正して複数のスレーブアドレスまたは 10 ビットアドレスの検出を選択するアプリケーションでは、ソフトウェアアドレス検出を使用する必要があります。このパラメータの初期設定は、ハードウェアです。ハードウェアアドレスのデコードが有効である場合、ブロックは自動的に CPU による介入なく、自身のアドレスでないものを NAK します。正しいアドレス受信時に自動的に CPU に割り込み、CPU の介入まで SCL ラインをローレベルに保持します。

Pins

このパラメータは SDA および SCL 信号接続に使用するピンのタイプを決定します。以下の 3 つの値が使用可能です。Any、I2C0、および I2C1。初期値は"Any"です。"Any" は汎用 I/O(GPIO または SIO)を意味します。

UDB clock source

このパラメータは、データレート生成を内部クロックによるか外部クロックによるかを選択します。Internal Clock に設定すると、PSoC Creator は 16 倍オーバーサンプリングを考慮して、**Data Rate** パラメータに基づいた必要クロック周波数を計算し、設定します。External Clock モードでは、コンポーネントはデータレートを制御できませんが、ユーザーが接続したクロックソースに基づいた実際のデータレートは表示できます。このパラメータを内部クロックに設定すると、クロック入力はシンボルに表示されなくなります。希望する内部クロック許容偏差を入力できます。クロック許容偏差はパーセントで指定します。初期設定の範囲は-5%～+50%です。

Enable UDB slave fixed placement

Enable UDB slave fixed placement パラメータにより、制約なし配置の場合よりも、コンポーネントの位置を固定することでコンポーネント性能を向上させることができます。このパラメータが設定されていると、コンポーネントリソースのすべてがデバイスの右上隅に固定されます。このパラメータはコンポーネントに接続されたピン割り当てを管理します。ピン割り当ての選択はコンポーネント性能の決定因子ではありません。この項目は、UDB が **Implementation** に設定されている場合にだけ利用できます。初期設定は無効です。コンポーネントを固定配置することにより、配線の融通が失われます。非固定配置の設計がかなり空の設計で動作するのと同じ程度に、固定配置も動作を継続できます。

[1] コンポーネントを PSoC5 で使用する場合は、FF は選択できません。PSoC5 I2C の FF は、バスリセットのため、バスストックからリカバリできません。

External IO Buffer

このパラメータによって、内部 I²C バスを多重化できます。内部 OE バッファが削除され、双方向 scl および sda 端子は個別の入力(sda_i および scl_i)および出力(sda_o および scl_o)によって置換されます。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。以後のセクションでは、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"SMBusSlave_1"を割り当てます。インスタンスの名称は、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"SMBusSlave"というインスタンス名を使用します。

注: コンポーネントの一部の API 関数は、コンポーネント ISR で使用されるため、Keil コンパイラで構築する場合、コンパイラの警告表示がでることがあります。これを回避するために、js ではこれらの関数に".cyre"ファイルが含まれています。

関数	機能
SMBusSlave_Start()	SMBusコンポーネントを初期化して有効にします。I²C割り込みを有効にして、コンポーネントがSMBusトラフィックに応答できるようにします。
SMBusSlave_Stop()	SMBusトラフィックへの応答を停止します。また、割り込みも無効にします。
SMBusSlave_EnableInt()	I²C割り込みを有効にします。
SMBusSlave_DisableInt()	I²C 割り込みを無効にします。
SMBusSlave_Init()	カスタマイザーから提供された初期値を用いて I²C レジスタを初期化します。
SMBusSlave_Enable()	I²Cハードウェアを有効化します。通常の動作に必要なI²C割り込みは有効化しません。
SMBusSlave_SetAddress()	I²Cスレーブのアドレスを設定します。
SMBusSlave_SetAlertResponseAddress()	アラート応答アドレスモードの時にデバイスが応答するI²Cスレーブアドレスを設定します。
SMBusSlave_GetNextTransaction()	トランザクションキューにある次のトランザクションレコードを指すポインタを返します。キューが空である場合は、関数はNULLを返します。
SMBusSlave_GetTransactionCount()	トランザクションキューにあるトランザクションレコードの数を返します。
SMBusSlave_CompleteTransaction()	コンポーネントに、キューの先頭にある現在保留中のトランザクションを完了させます。
SMBusSlave_SetSmbAlert()	SMBALERT# smbalertピンをアサートまたはデアサートします。
SMBusSlave_SetSmbAlertMode()	アラート応答アドレスで、コンポーネントがSMBusマスター読み取りに対してどのように応答するかを決定します。

関数	機能
SMBusSlave_HandleSmbAlertResponse()	ホストがアラート応答アドレスに応答する時にコンポーネントから呼び出され、SMBALERTモードをFIRMWARE_MODEに設定します。
SMBusSlave_GetReceiveByteResponse()	"Receive Byte"プロトコル要求を検出した時に、I^2C ISRによって呼び出されて、応答バイトを決定します。
SMBusSlave_HandleBusError()	バスプロトコルエラーが発生する度に、コンポーネントから呼び出されます。
SMBusSlave_StoreUserAll()	フラッシュで、RAMレジスタストアをユーザーレジスタストアに保存します。
SMBusSlave_RestoreUserAll()	ユーザーレジスタストアのCRCフィールドを確認してから、ユーザーレジスタストアの内容をRAMレジスタストアにコピーします。
SMBusSlave_RestoreDefaultAll()	デフォルトレジスタストアの署名フィールドを確認してから、デフォルトレジスタストアの内容をRAMレジスタストアにコピーします。
SMBusSlave_StoreComponentAll()	この関数を呼び出して、システムの他のコンポーネントのパラメータを現在のPMBusの設定で更新します。
SMBusSlave_RestoreComponentAll()	この関数を呼び出して、PMBusオペレーティングレジスタストアをシステムの他のコンポーネントの現在の設定パラメータで更新します。
SMBusSlave_Lin11ToFloat()	引数"linear11"を浮動小数点数に変換して返します。
SMBusSlave_FloatToLin11()	引数"floatvar"(浮動小数点数)を取り、16ビットのLINEAR11値(11ビットの仮数 + 5ビットの指数)に変換して返します。
SMBusSlave_Lin16ToFloat()	引数"linear16"を浮動小数点数に変換して返します。
SMBusSlave_FloatToLin16()	引数"floatvar"(浮動小数点数)を受け取り、16ビットのLINEAR16値(16ビットの仮数)に変換して返します。

グローバル変数

関数	機能
SMBusSlave_initVar (static)	initVar変数は、このコンポーネントの設定を示すために使用します。この変数には、コンポーネントの名前(この場合は、SMBusSlave)が付けられています。SMBusSlave_initVar変数は0に初期化されており、初めてSMBusSlaveStart()が呼び出されたときに1に設定されます。これにより、SMBusSlave Start()ルーチンの以後のすべての呼び出しで再初期化を行うことなく、コンポーネントを初期化できます。 デバイスがスリープサイクルになると、コンポーネントを再初期化する必要があります。したがって、スリープ状態、SMBusSlave Sleep()になるとこの変数は 0 に設定され、SMBusSlave Wakeup()で再初期化の実行間に設定されます。

void SMBusSlave_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。SMBusSlave_Start()はSMBusSlave_Init()関数を呼び出し、次にSMBusSlave_Enable()関数を呼び出します。SMBusSlave_Start()はI²Cバス処理の前に呼び出す必要があります。このAPIがI²C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 SMBusSlave_Stop(void)

機能: この関数はI²Cハードウェアおよび割り込みを無効にします。デバイスによってバスがロックアップされている場合、I²Cバスを開放し、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 SMBusSlave_EnableInt(void)

機能: この関数はI²C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 SMBusSlave_DisableInt(void)

機能: この関数はI²C割り込みを無効にします。I2C_Stop() 関数が割り込みを無効にしているため、この関数は通常不要です。

パラメータ: なし

返り値: なし

注意事項: I²Cの実行中にI²C割り込みが無効であると、I²Cバスがロックアップします。

void SMBusSlave_Init(void)

機能: この関数はカスタマイザーの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。SMBusSlave_Start() APIがこの関数を呼び出すので、SMBusSlave_Init()を呼び出す必要はありません。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。

パラメータ: なし

返り値: なし

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

void SMBusSlave_Enable(void)

機能: この関数はハードウェアを起動し、コンポーネントの動作を開始します。SMBusSlave_Start() APIがこの関数を呼び出すので、SMBusSlave_Enable()を呼び出す必要はありません。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。このAPIを呼び出す前に、まずSMBusSlave _Start()またはSMBusSlave _Init()を先に呼び出す必要があります。

パラメータ: なし

返り値: なし

注意事項: なし

void SMBusSlave_SetAddress(uint8 address)

機能: この関数はI^2Cスレーブアドレスを設定します。

パラメータ: uint8アドレス: プライマリデバイスのI^2Cスレーブアドレス。この値は0から127(0x00 ～ 0x7F)の任意の値に設定できます。このアドレスは7ビットの右揃えスレーブアドレスであり、R/Wビットは含んでいません。

返り値: なし

注意事項: なし

voidSMBusSlave_SetAlertResponseAddress(uint8 address)

機能: この関数は、アラート応答アドレスモードの時にデバイスが応答するI^2Cスレーブアドレスを設定します。

パラメータ: uint8アドレス: アラート応答モードのI^2Cスレーブアドレスです。この値は0から127(0x00 ～ 0x7F)の任意の値に設定できます。このアドレスは7ビットの右揃えスレーブアドレスであり、R/Wビットは含んでいません。

返り値: なし

注意事項: なし

TRANSACTION_STRUCT* SMBusSlave_GetNextTransaction(void)

機能: この関数は、トランザクションキューにある次のトランザクションレコードを指すポインタを返します。キューが空である場合は、関数はNULLを返します。コンポーネントがキューにあるすべての自動トランザクションを処理するので、この関数は手動での読み取りおよび書き込みだけを返します。書き込みの場合、ユーザーファームウェアの機能を使って"ペイロード"をレジスタストアへコピーする必要があります。読み取りの場合、ユーザーファームウェアの機能を使ってレジスタストアでこのコマンドの変数の内容を更新する必要があります。どちらの場合も、SMBusSlave_CompleteTransaction()を呼び出して、トランザクションレコードを解放します。

読み取りトランザクションの場合、ほぼすべてのトランザクションでは、長さとペイロードのフィールドは使用されないことに注意してください。例外はプロセスコールで、書き込みフェーズからのワードがペイロードフィールドに保存されます。

パラメータ: なし

返り値: 次のトランザクションレコードを指すポインタ

注意事項: なし

uint8 SMBusSlave_GetTransactionCount(void)

機能: トランザクションキューにあるトランザクションレコードの数を返します。

パラメータ: なし

返り値: uint8: トランザクションキューのレコードの数

注意事項: なし

void SMBusSlave_CompleteTransaction(void)

機能: キューの先頭にある現在保留中のトランザクションがコンポーネントによって完了されます。ユーザーファームウェアのトランザクションハンドラーが、トランザクション処理後にこの関数を呼び出します。これは、コンポーネントコードを変更し、転送を完了させるために、保留中の読み取りトランザクションに関連するレジスタ変数をI^2C転送バッファにコピーします。また、キューも先へ進めます。読み取りと書き込みに対して呼び出す必要があります。

パラメータ: なし

返り値: なし

注意事項: なし

void SMBusSlave_SetSmbAlert(uint8 assert)

機能: SMBALERT# smbalertピンをアサートまたはデアサートします。SMBALERT#がアサートされている限り、コンポーネントはアラート応答アドレスに対するマスターREADに応答します。応答は、デバイスのプライマリスレーブアドレスになります。モードの設定によって、コンポーネントは自動的にSMBALERT#をデアサートするか、SMBusSlave_HandleSmbAlertResponse()を呼び出すか、何もしません。

パラメータ: uint8: アサート

SPDIFステータスマスク	タイプ
SMBusSlave_SMBALERT_DEASSERT	smbalertピンのデアサート
SMBusSlave_SMBALERT_ASSERT	smbalertピンのアサート

返り値: なし

注意事項: なし

void SMBusSlave_SetSmbAlertMode(uint8 alertMode)

機能: この関数は、アラート応答アドレスで、コンポーネントがSMBusマスターReadに対してどのように応答するかを決定します。SMBALERT#がアサートされると、SMBusマスターはReadをグローバルアラート応答アドレスへブロードキャストします。これによりバスのどのSMBusデバイスがSMBALERT#をアサートしているか確認します。

自動モードで、バスマスターが正常にアラート応答アドレスをREADすると、SMBALERT#は自動的にデアサートされます。

手動モードでは、コンポーネントがAPIのSMBusSlave_HandleSmbAlertResponse()を呼び出し、ユーザーコード(マージセクションの)がSMBALERT#をデアサートします。

DO_NOTHINGモードでは、コンポーネントは動作をしません。

パラメータ: uint8: alertMode、SMBALERTピンモードを定義するバイトです。

SPDIFステータスマスク	タイプ
SMBusSlave_DO_NOTHING	SMBALERT#ピンには何もしない
SMBusSlave_AUTO_MODE	SMBALERT#ピンを自動的にデアサートする
SMBusSlave_FIRMWARE_MODE	ユーザーが、SMBALERT#ピンをデアサートする必要があります

返り値: なし

注意事項: なし

void SMBusSlave_HandleSmbAlertResponse(void)

機能: このAPIは、ホストがアラート応答アドレスに応答する時にコンポーネントから呼び出され、SMBALERTモードをFIRMWARE_MODEに設定します。この関数には、マスターが応答した後に、実行するコードをユーザーによる挿入するマージコードセクションがあります。たとえば、ユーザーはステータスレジスタを更新し、SMBALERT#ピンをデアサートします。

パラメータ: なし

返り値: なし

注意事項: なし

uint8 SMBusSlave_GetReceiveByteResponse(void)

機能: この関数は、"Receive Byte"プロトコル要求を検出した時に、I^2C ISRによって呼び出されて、応答バイトを決定します。この関数には、返り値の初期値(0xFF)を上書きするコードをユーザーが挿入するマージコードセクションがあります。この関数は、ISRのコンテキストで呼び出されます。したがって、ユーザーのマージコードは、高速で、ブロックしない、リエントラントな関数だけを呼び出すものである必要があります。

パラメータ: なし

返り値: uint8: ユーザー指定のステータスバイト

SPDIFステータスマスク	タイプ
SMBusSlave_RET_UNDEFINED	初期設定の戻りステータス

注意事項: なし

void SMBusSlave_HandleBusError(uint8 errorCode)

機能: このAPIは、バスプロトコルエラーが発生する度に、コンポーネントから呼び出されます。バスエラーの例：無効なコマンド、データのアンダーフロー、クロックストレッチ違反。コンポーネントが既に決定的な方法でエラーを処理しているので、この関数はエラーの余波だけを処理します。この関数の主な目的は、エラーが発生したことをユーザーファームウェアに通知することです。たとえば、PMBusデバイスで、これはユーザーファームウェアにSTATUS_CMLレジスタで適切なエラービットを設定する機会を与えます。

パラメータ: uint8 errorCode:

SPDIFステータスマスク	タイプ
SMBusSlave_ERR_READ_FLAG	Readフラグが間違ってセットされた
SMBusSlave_ERR_RD_TO_MANY_BYTES	ホストが多すぎるバイト数を読み取ろうとした
SMBusSlave_ERR_WR_TO_MANY_BYTES	ホストが多すぎるバイト数を書き込もうとした
SMBusSlave_ERR_UNSUPORTED_CMD	受信したコマンドはサポートされていません
SMBusSlave_ERR_INVALID_DATA	受信したデータが無効です
SMBusSlave_ERR_TIMEOUT	バスリセットタイムアウトが発生しました
SMBusSlave_ERR_WR_TO_FEW_BYTES	ホストが少なすぎるバイト数を書き込もうとした

返り値: なし

注意事項: なし

uint8 SMBusSlave_StoreUserAll(char * flashRegs)

機能: この関数は、RAMレジスタストアをフラッシュにあるユーザーレジスタストアに保存します。保存する前に、レジスタストアデータ構造のCRCフィールドが再計算、更新されます。この関数は、初期設定では、フラッシュになにも保存しません。代わりに、ユーザーがオペレーティングメモリをフラッシュに保存するアルゴリズムを実装できるマージ領域があります。

パラメータ: flashRegs:
オペレーティングメモリ(RAM)を保存するフラッシュの場所を指すポインタ。

返り値: uint8: 以下の標準リターンステータスのいずれか

SPDIFステータスマスク	タイプ
CYRET_SUCCESS	アクションは正しく完了しました
CYRET_MEMORY	メモリーの問題が発生しました

注意事項: なし

uint8 SMBusSlave_RestoreUserAll(char * flashRegs)

機能: この関数は、ユーザーレジスタストアのCRCフィールドを確認してから、ユーザーレジスタストアの内容をRAMレジスタストアにコピーします。この関数は、初期設定では、フラッシュからのレジスタストアの復元は実行しません。代わりに、ユーザーがオペレーティングメモリ(RAM)にデータを復元するアルゴリズムを実装できるマージ領域があります。

パラメータ: flashRegs:
オペレーティングメモリ(RAM)が保存されるフラッシュの場所を指すポインタ 。

返り値: uint8: 以下の標準リターンステータスのいずれか。

SPDIFステータスマスク	タイプ
CYRET_SUCCESS	アクションは正しく完了しました
CYRET_BAD_DATA	データが不良です。CRCが一致しません

注意事項: なし

uint8 SMBusSlave_RestoreDefaultAll(void)

機能: この関数は、デフォルトレジスタストアの署名フィールドを確認してから、デフォルトレジスタストアの内容をRAMレジスタストアにコピーします。

パラメータ: なし

返り値: uint8: 以下の標準リターンステータスのいずれか。

SPDIFステータスマスク	タイプ
CYRET_SUCCESS	アクションは正しく完了しました
CYRET_BAD_DATA	データが不良です。CRCが一致しませ

注意事項: なし

uint8 SMBusSlave_StoreComponentAll(void)

機能: ユーザーはこの関数を呼び出して、システムの他のコンポーネントのパラメータを現在のPMBusの設定で更新します。このアクションは、アプリケーション固有性が強いため、この関数はほぼ全体がマージセクションで構成されます。コンポーネントが供給するファームウェアは、CYRET_SUCCESSに初期化され、関数の最後に返される返り値変数(retval)だけです。関数の残りの部分は、ユーザーが提供する必要があります。

パラメータ: なし

返り値: uint8: 以下の標準リターンステータスのいずれか。

SPDIFステータスマスク	タイプ
CYRET_SUCCESS	アクションは正しく完了しました

あるいは、ユーザーが規定するSUCCESS以外のステータスです。

注意事項: なし

uint8 SMBusSlave_RestoreComponentAll(void)

機能: この関数を呼び出して、PMBusオペレーティングレジスタストアをシステムの他のコンポーネントの現在の構成パラメータで更新します。このアクションは、アプリケーション固有性が強いため、この関数はほぼ全体がマージセクションで構成されます。コンポーネントが供給するファームウェアは、CYRET_SUCCESSに初期化され、関数の最後に返される返り値変数(retval)だけです。関数の残りの部分は、ユーザーが提供する必要があります。

パラメータ: なし

返り値: uint8: 以下の標準リターンステータスのいずれか。

SPDIFステータスマスク	タイプ
CYRET_SUCCESS	アクションは正しく完了しました

あるいは、他のユーザーが既定するSUCCESS以外のステータスです。

注意事項: なし

float SMBusSlave_Lin11ToFloat (uint16 linear11)

機能: この関数は、引数"linear11"を浮動小数点数に変換して返します。

パラメータ: uint16 linear11: LINEAR11フォーマットの数字です。

返り値: Float: 浮動小数点数に変換されたlinear11パラメータ

注意事項: なし

uint16 SMBusSlave_FloatToLin11 (float floatvar)

機能: この関数は、引数"floatvar"(浮動小数点数)を取り、16ビットのLINEAR11値(11ビットの仮数 + 5ビットの指数)に変換して返します。

パラメータ: float floatvar: 浮動小数点数

返り値: uint16: LINEAR11に変換された浮動小数点数の変数

注意事項: なし

float SMBusSlave_Lin16ToFloat(uint16 linear16, int8 inExponent)

機能: この関数は、引数"linear16"を浮動小数点数に変換して返します。引数Linear16には仮数があります。引数inExponentは、変換で使用される5ビットの2の補数です。

パラメータ: uint16 linear16: LINEAR16数の16ビットの仮数。

int8inExponent: LINEAR16数の5ビットの仮数。下位5ビットに収納されます。2の補数。

返り値: Float: 浮動小数点数に変換されたパラメータ

注意事項: なし

uint16 SMBus_FloatToLin16(float floatvar, int8 outExponent)

機能: この関数は、引数"floatvar" (浮動小数点数)を取り、16ビットのLINEAR16値(16ビットの仮数)に変換して返します。引数outExponentは、変換で使用される5ビットの2の補数です。

パラメータ: floatfloatvar: LINEAR16に変換される浮動小数点数です。

int8outExponent: 変換で使用する5ビットの指数は、ユーザーが提供します。

返り値: uint16: LINEAR16に変換されるパラメータ。

注意事項: なし

ブートローダサポート

SMBus および PMBus スレーブコンポーネントはブートローダの通信コンポーネントとして使用できます。ブートローダの詳細については、*System Reference Guide* の"Bootloader System"セクションを参照してください。

SMBus および PMBus スレーブコンポーネントはブートローダ用 API 関数を提供します。

関数	機能
SMBusSlave_CyBtldrCommStart	SMBusおよびPMBusスレーブコンポーネントを開始し、その割り込みを有効にします。

関数	機能
SMBusSlave_CyBtldrCommStop	SMBusおよびPMBusスレーブコンポーネントを無効にし、その割り込みを無効にします。
SMBusSlave_CyBtldrCommReset	読み取りおよび書き込み用I^2Cバッファを初期状態に設定し、スレーブ状態をリセットします。
SMBusSlave_CyBtldrCommWrite	呼び出し元がデータをブートローダホストに書き込むことができるようになります。この関数はポーリングを管理して、データブロックをホストデバイスへ完全に送信できるようにします。
SMBusSlave_CyBtldrCommRead	呼び出し元がデータをブートローダホストから読み取ることができるようになります。この関数はポーリングを管理して、データブロックをホストデバイスから完全に受信できるようにします。

void SMBusSlave_CyBtldrCommStart(void)

機能: 通信コンポーネントを開始し、割り込みを有効にします。読み取りバッファの初期状態はフルであり、読み取りは必ず0xFFuになります。書き込みバッファはクリアされ、コマンド受信可能になります。

パラメータ: なし

返り値: なし

注意事項: この関数は、コンポーネントの割り込みを有効にします。割り込みを有効にせずI^2Cを有効にすると、バスがロックアップする可能性があります。

void SMBusSlave_CyBtldrCommStop(void)

機能: 通信コンポーネントを無効にし、割り込みを無効にします。

パラメータ: なし

返り値: なし

注意事項: なし

void SMBusSlave_CyBtldrCommReset(void)

機能: バッファを初期状態に設定し、ステータスをリセットします。読み取りバッファの初期状態はフルであり、読み取りは必ず0xFFuになります。書き込みバッファはクリアされ、コマンド受信可能になります。

パラメータ: なし

返り値: なし

注意事項: なし

cystatus I2C_CyBtldrCommRead(uint8 * Data, uint16 size, uint16 * count, uint8 timeOut)

機能: ホストからコマンドを受信します。すべてのバイトがI²C ISRによって受信され、内部I²Cバッファに保存されます。関数は、タイムアウトのステータスをチェックして、転送の終了を確認してから、データをブートローダのバッファにコピーします。この関数が終了すると、I²C ISRはさらにデータを受信できるようになります。

パラメータ: uint8 *Data: ブートローダホストから読み取るデータブロックの保管場所を指すポインタ

uint16 size: 読み込むバイト数

uint16 *count: 実際に読み取ったバイト数を書き込む変数を指すポインタ

uint8 timeOut: タイムアウトによって復帰するまでの待ち時間の10ms単位の数値

返り値: cystatus: 何も問題がなかった場合、CYRET_SUCCESS を返すか、または問題を最も適切に表現する値を返します。詳細については、*System Reference Guide*の"Return Codes"を参照してください。

注意事項: なし

cystatus SMBusSlave_CyBtldrCommWrite(uint8 * Data, uint16 size, uint16 * count, uint8 timeOut)

機能: 実行したコマンドのステータスをホストに転送します。関数は、I²C読み取りバッファを応答で更新し、ホストへ解放します。バッファが解放されるまでは、すべての読み取りに対して0xFFを返します。すべてのバイトが、I²C ISRによって転送されます。すべてのバイトが読み取られるまで、関数はタイムアウトで待ちます。この関数の終了後、読み取りに対して0xFFを返します。

パラメータ: uint8 *Data: ブートローダーホストへ書き込まれるデータブロックを指すポインタ

uint16 size: 書き込まれたバイト数

uint16 *count: 実際に書き込んだバイト数を書き込む変数を指すポインタ

uint8 timeOut: タイムアウトによって復帰するまでの待ち時間の10ms単位の数値

返り値: cystatus: 何も問題がなかった場合、CYRET_SUCCESS を返すか、または問題を最も適切に表現する値を返します。詳細については、*System Reference Guide*の"Return Codes" を参照してください。

注意事項: "ブートローダー書き込み"トランザクション中にクロックのストレッチの先頭で呼び出されたが、この関数から戻る前に割り込みが(手動コマンドの処理動作と一致させるため)無効であった場合、コンポーネントの割り込みを一時的に有効にします。"ブートローダー書き込み"を使用しないで呼び出されると、コンポーネントの割り込みは必ず有効のままになります。

マクロ

- SMBusSlave_FL_ADDR_TO_ROW(addr) – 指定したアドレスから、フラッシュの行番号を取得します。

- SMBusSlave_FL_ADDR_TO_ARRAYID(addr) - 指定したアドレスから、フラッシュの配列 ID を取得します。
- SMBusSlave_SIZE_TO_ROW(size) – *サイズ*によって定義されるデータの数を保存するために必要なフラッシュ行の数を計算して、返します。
- SMBusSlave_MAX_PAGES – ページコマンドが使用するページの最大数を指定します。
- SMBusSlave_NUM_COMMANDS – ページコマンドのページの数を定義します。

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。

割り込みサービスルーチン

SMBus および PMBus スレーブのコンポーネントは、2 つの ISR を使用します。一つ目の割り込みは、コマンドのデコードと SM/PM Bus へのデータ転送を処理します。二つの目割り込みは、スタックロー状態が発生した場合にバスをリセットするように設計された 25ms タイムアウト割り込みです。

機能の詳細

このコンポーネントの動作原理は、I^2C スレーブコンポーネントと非常に似ています。SMBus 仕様は、SMBus specification version 2.0 によります。このコンポーネントの重要な機能は、このセクションで説明されています。

I^2C 物理レイヤ

SM/PM Bus スレーブコンポーネントの物理レイヤは、I^2C プロトコルに基づいています。このコンポーネントに影響する 3 つの主要な相違は以下の通りです。

SMBus の仕様では、SCL 信号が 25ms の間スタックローであると検出された場合、コンポーネントは SCL および SDA のラインをリセットして、解放しなければならないと規定されています。この詳細については、AC/ DC 電気的特性のセクションで説明しています。またこのコンポーネントは、SDA ラインを監視し、特別な予防措置として 25ms 間スタックローであった場合は、SDA ラインをリセットします。

SMBus の仕様によって、いかなる転送であっても、コンポーネントはクロックを累積 25ms 以上に伸ばしてはならないことが規定されています。スレーブでは、いかなるトランザクションでも累積のストレッチ時間が 25ms を超えない限り、SCL ロー(クロックストレッチ)でビジーである場合、転送を遅らせることができます。

デバイスに異常があることをホストに通知するには、SMBALERT#ピンを追加します。

SMBus/ PMBus のアドレス指定

すべての SMBus/ PMBus スレーブデバイスには I^2C アドレスがあります。以下のアドレスは、SMBus の動作上予約されており、SMBus/ PMBus スレーブの通常のスレーブアドレスとして使用することはできません。

スレーブアドレス(ビット7:1)	R/W#ビット(ビット0)	コメント
0000 000	0	一般的な呼び出しアドレス
0000 000	1	STARTバイト
0000 001	X	CBUSアドレス
0000 010	X	異なるバスフォーマットのために予約済み
0000 011	X	将来に使用するために予約済み
0000 1XX	X	将来に使用するために予約済み
0101 000	X	ACCESSバスホストのために予約済み
0110 111	X	ACCESSバスの初期アドレスのために予約済み
1111 0XX	X	10ビットスレーブアドレス指定のために予約済み
1111 1XX	X	将来に使用するために予約済み
0001 000	X	SMBusホストのために予約済み
0001 100	X	SMBusアラート応答アドレス
1100 001	X	SMBusデバイスの初期アドレス

SMBALERT#ピンを有効にしていれば、コンポーネントはプライマリアドレスおよびアラート応答アドレスに応答します。

SMBus/ PMBus プロトコル

9 種類の異なるプロトコルが SMBus の仕様に定義されています。これらのプロトコルの概要とサポート状況を以下に示します。

プロトコル	対応状況	備考
クイックコマンド	非対応	SMBusのみ

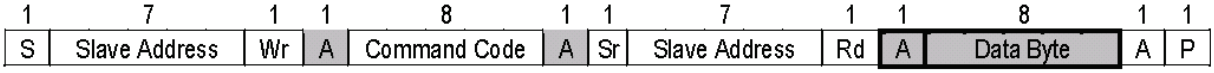
プロトコル	対応状況	備考
バイト送信	対応	両方
バイト受信	対応	SMBusのみ
バイト/ワード書き込み	対応	両方
バイト/ワード読み取り	対応	両方
プロセスコール	対応	両方
ブロック書き込み/読み取り	対応	両方
ブロック書き込み/読み取り、プロセスコール読み取り	対応	両方
SMBusホスト通知プロトコル	非対応	両方

SM/PM Bus コンポーネントを特徴付ける重要な概念は以下のとおりです。

SMBus、PMBus いずれもサブアドレスまたは I^2C レジスタマップの概念は使用しません。すべての転送はコマンドに基づきます。I^2C スレーブアドレスの直後に続くのはコマンドコードです。コマンドは、書き込み専用、読み取り専用、または読み取り/書き込みです。SMBus で、コマンドの定義およびそれらの読み取り/書き込みの制限の大部分は、ユーザーによって異なります。PMBus で、ほぼすべてのコマンドは事前に定義されますが、一部のコマンドコードは未割当であり、メーカー固有の実装として利用できます。

読み取り処理において、ホストは読み取りコマンドコードの書き込みの間に、リピートスタートコンディションを発行して必要なデータの読み取りに切り替えます。ストップコンディションは、処理全体が終了した時に一度だけ生成されます。下に例を示します。

Read Byte Protocol



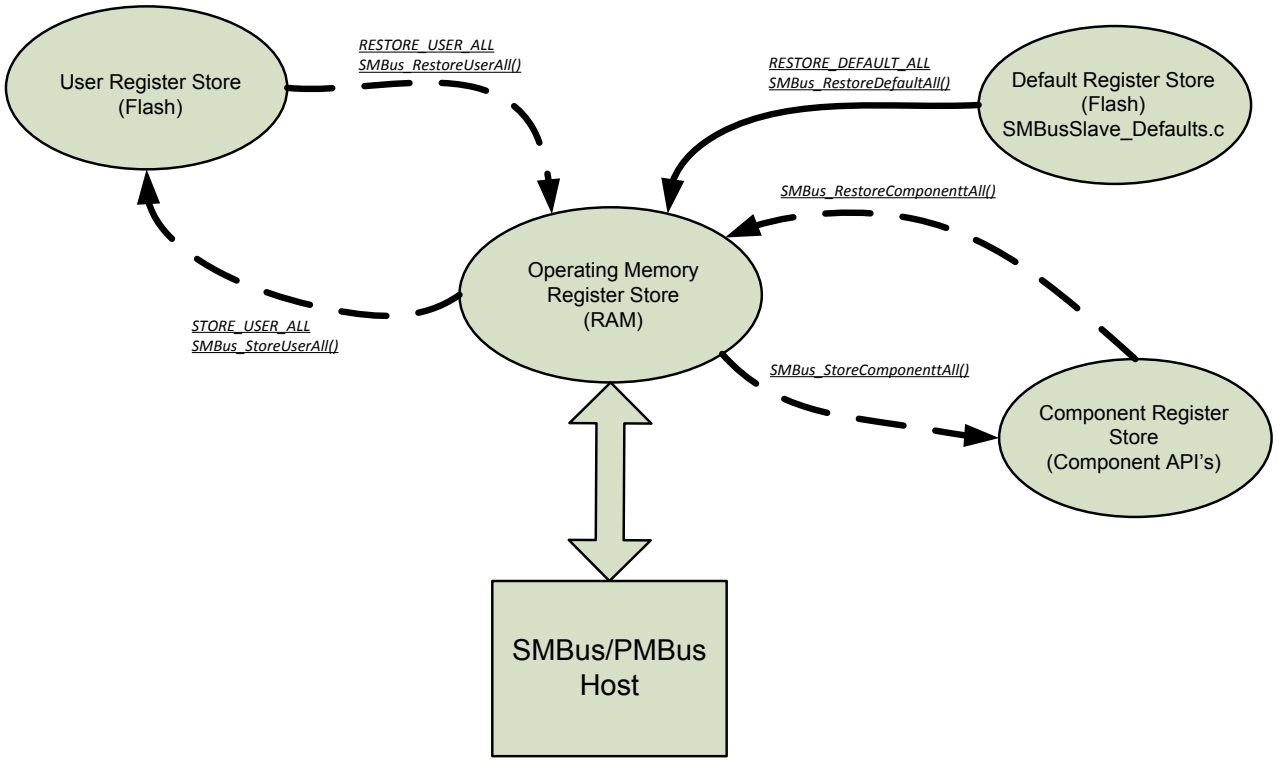
SMBus と PMBus はリトルエンディアンのプロトコルです(即ち複数バイトデータ型の最下位バイトが最初にバスに転送されます)。PSoC に関しては、すべてのレジスタが PSoC ネイティブのエンディアンフォーマットで保存される必要があります。コンポーネントは、PSoC と I^2C バッファの間でエンディアンを正しく変換する必要があります これが適用されるのは 16 ビットのワード転送だけです。

SMBus の仕様では、コマンドコードは一切定義されておらず、ユーザーが定義します。ただし、PMBus の仕様のパート II、付録 I は 255 の可能なコマンドコードをすべて定義して、その中の 46 はユーザーによって定義されます(メーカー固有のコマンドと呼ばれます)。

レジスタストアの概念

PMBus 仕様 II 5.4.2 によると「書き込み可能なパラメータは読み取り可能である必要があります。」一般に、書き込む値を受け取るすべてのコマンドは、読み取り時にも値を返す必要があります。この目的で、レジスタストアの概念を使用します。

図 5. レジスタストアの概念



オペレーティングメモリレジスタストア

これは、RAM バージョンのレジスタストアです。ランタイムの SMBus/ PMBus コマンドは、このバージョンのレジスタストアを変更します。このレジスタストアは RAM にあるので、その内容はリセット時には無効であると推測されます。

デフォルトレジスタストア

すべての SMBus/ PMBus パラメータの初期値を含んでいる、フラッシュバージョンのレジスタストア。起動時、デフォルトレジスタストアをオペレーティングメモリレジスタストアにコピーして、オペレーティングメモリストアを初期化できます。デフォルトレジスタストアのパラメータ値は、コンパイル/リンク時に決定されます。ユーザーは、レジスタストアの初期値を入力する必要があります。SMBusSlave_Defaults.c を開くには、コマンドブロックのパラメータをコピーし、下のマージ領域に貼り付ける必要があります。これによりパラメータが SMBusSlave_regsDefault に保

存され、毎回の起動時に、これらのパラメータで動作メモリーが初期化されます。初期値でオペレーティングメモリを初期化するもう一つの方法は、SMBus_RestoreDefaultAll()を呼び出すことです。この関数は、RESTORE_DEFAULT_ALL (0x12) PM バスコマンドのハンドラーとして使用できます。

ユーザーレジスタストア

これは、別のフラッシュバージョンのレジスタストアです。本質的に読み取り専用であるデフォルトレジスタストアとは異なり、ユーザーレジスタストアはオペレーティングレジスタストアの最新の内容に基づいて更新できます。フラッシュへのデータ保存は実装によって大きく異なるため、コンポーネントはフラッシュへの保存は一切行いません。代わりに、その目的で使用できる 2 つの API 関数がマージ領域で提供されています。SMBusSlave_StoreUserAll()は、リストアの時点でデータの正当性のチェックに必要な CRC を計算します。

データをフラッシュに保存する独自の方法を設計する際は、PSoC Creatorに付属のSystem Reference Guideを参照してください。System Reference Guideのセクション10に、基本情報とフラッシュで動作する関数の説明があります。フラッシュで作業をする際に便利なマクロのセットも、コンポーネントに提供されています。

コンポーネントレジスタストア

レジスタストアの主な目的は、PMBus が標準の PSoC Creator のコンポーネントからパラメータを抽出して設定できるようにすることです(コンポーネントレジスタストアの名前の通り)。Creator コンポーネントには、独自の設定パラメータがあり、通常はコンポーネントカスタマイザーを使って設定されます。これらのパラメータは、ランタイム時にコンポーネントに固有の SMBusSlave_StoreComponentAll()/SMBusSlave_RestoreComponentAll() の API でアクセスできます。API でアクセスできるコンポーネントのパラメータが、コンポーネントレジスタストアを構成します。起動時に、PMBus のユーザーファームウェアは以下を行います:

- ユーザーまたはデフォルトレジスタストアに保存されている PMBus パラメータに基づいて他のコンポーネントの設定を更新します。
- コンポーネントの設定に基づいて、PMBus レジスタストアを更新します。

特殊ケースのコマンド

PAGE コマンド

PMBus の PAGE (コード 0x00)コマンド(PMBus パート II – セクション 11.10)では、同じ PMBus I^2C アドレスで複数の PMBus 論理デバイスにアクセスできます。たとえば、複数の電源レールを制御する PMBus の電源監視は、それぞれ独自のページでそれぞれのレールのコマンド/パラメータにアクセスすることができます。ページとは、コマンド/レジスタの配列へのインデックスであると考えることができます。PAGE コマンドでページが設定されると、別の PAGE コマンドで再度ページが設定されるまで、ページの設定は保持されます。

PMBus モードでは、コンポーネントに PAGE コマンドのサポートが組み込まれています。SMBus モードでは、ユーザーは PAGE コマンドを有効にするかどうかを選択でき、PAGE で使用するコマンドコードを指定できます。

PAGE コマンドが有効である場合、PAGE コマンドの値として有効な範囲は 1～32 であり、ユーザーが決定する**最大ページ**の設定内である必要があります。例外は、"All Pages"ワイルドカード設定で、値は 0xFF になります。"All Pages"ワイルドカード設定が有効であるのは、書き込み処理に対してだけであり、必ず"Manual"モードで処理する必要があります。PAGE が 0xFF に設定されている場合、以下の処理はエラーとして処理されます。

ページコマンドからの読み取り。

手動として設定されているページコマンドへの書き込み。

複数のページがある PMBus デバイスであっても、一部のコマンドは現在のページに依存せず、PAGE の設定に関わらず常に同じ方法で処理されます。たとえば、PMBUS_REVISION コマンドはデバイスがサポートしている PMBus のバージョンを返しますが、すべてのページで共通です。このように、ページコマンドと共通コマンドの概念があります。それぞれの SMBus/ PMBus コマンドについて、カスタマイザーで、コマンドがページであるか、共通であるかを指定することができます。コマンドがページとマークされると、コンポーネントはそのコマンドのレジスタストアで配列を定義します。

QUERY コマンド

PMBus の QUERY (Code 0x1A)コマンドは、PMBus デバイスで特定のコマンドがサポートされているか確認し、サポートされている場合は、そのコマンドでどんなデータ形式がサポートされているか確認します。PMBus モードでは、コンポーネントに QUERY コマンドのサポートが組み込まれています。SMBus モードでは、ユーザーは QUERY コマンドを有効にするかを選択でき、QUERY で使用するコマンドコードを指定できます。

このコマンドは、SMBus の仕様で定められているブロック書き込み-ブロック読み取りプロセスコールを使用します。プロセスコールの書き込み部分で、データバイト 1 バイトは符号なしのバイナリ整数であり、その値は対象となっているコマンドのコマンドコードと等しくなります。プロセスコールの読み取り部分では、データバイト 1 バイトは符号なしのバイナリ整数であり、その値は以下のとおりです:

ビット	値	意味
7	1	コマンドに対応
	0	コマンドに非対応
6	1	コマンドの書き込みに対応
	0	コマンドの書き込みに非対応
5	1	コマンドの読み取りに対応
	0	コマンドの読み取りに非対応
4:2	000	リニアデータ形式を使用
	001	16ビット符号付き数字
	010	予約済み
	011	ダイレクトモード形式を使用

ビット	値	意味
	100	8ビット符号なし数字
	101	VIDモード形式を使用
	110	メーカー固有形式を使用
	111	コマンドは数値データを返さない。これは、データブロックを返すコマンドでも使用されます。
1:0	XX	将来に使用するために予約済み

上の表の情報は、すべてカスタマイザーでのユーザー設定によって決まります。QUERY の対象のコマンドコードがサポートされていない場合は、"0x00"が返されます。

ブートローダコマンド

コンポーネントが"ブートローダ"プロジェクトに配置されると、**Custom Commands** タブの設定ダイアログで 2 つの追加コマンドが使えるようになります。それは、BOOTLOAD_READ と BOOTLOAD_WRITE であり、初期設定のコマンドコードはそれぞれ 0xFD と 0xFC です。これらのコマンドによって、コンポーネントはブートローダコンポーネントに対する通信コンポーネントとして機能します。

これら 2 つのコマンドは、回路図に配置されたブートローダコンポーネントと交信します。ブートローダコンポーネントを配置してから、ブートローダコンポーネントの設定ダイアログで"Custom interface"を選択します。

リソース

以下に示すのは、FF 実装の SM/PM Bus スレーブコンポーネントのリソース図です。データは、400kbps に設定されたデータレートで収集されました。

PSoC 3

構成	リソースタイプ					
	データパスセル	マクロセル	ステータスセル	コントロールセル	I^2C FF ブロック	割り込み
SMBus/ PMBus(UDB)	3	32	2	6	-	2
SMBus/ PMBus(FF)	2	9	1	4	1	2

PSoC 5

構成	リソースタイプ					
	データパス セル	マクロセル	ステータス セル	コントロール セル	I^2C FF ブロック	割り込み
SMBus/ PMBus(UDB)	3	32	2	6	-	2
SMBus/ PMBus(FF)	適用なし	適用なし	適用なし	適用なし	適用なし	適用なし

PSoC 5LP

構成	リソースタイプ					
	データパス セル	マクロセル	ステータス セル	コントロール セル	I^2C FF ブロック	割り込み
SMBus/ PMBus(UDB)	3	32	2	6	–	2
SMBus/ PMBus(FF)	1	5	2	2	1	2

API メモリー使用

コンポーネントのメモリー使用量は、コンパイラ、デバイス、使用する API の数、コンパイラの設定によって大きく変動します。以下の表は、特定のコンポーネント構成で使用可能なすべての API のメモリー使用量を示しています。

コンパイラをリリースモードに設定し、サイズを最適化して測定されました。特定の設計については、コンパイラが生成するマップファイルを分析して、メモリー使用量を決定できます。

構成	PSoC 3 (Keil_PK51)		PSoC 5 (GCC)		PSoC 5LP (GCC)	
	フラッシュ バイト	SRAM バイト	フラッシュ バイト	SRAM バイト	フラッシュ バイト	SRAM バイト
SMバススレーブ(FF)	5501	200	適用なし	適用なし	3540	168
SMBusスレーブ(UDB)	5587	196	3832	156	3824	156
PMBusスレーブ(FF)	8166	1788	適用なし	適用なし	4632	4068
PMBusスレーブ(UDB)	8080	1792	4924	4056	4916	4056

DC/ AC 電気的特性

特記なき限り、仕様は–40°C≤ T_A ≤ 85°CおよびTJ ≤ 100°Cの条件で有効です。特記なき限り、仕様は1.71V～5.5Vについて有効です。

SMBus/ PMBus スレーブ DC 仕様

記号	項目		Min	Typ[2]	Max	単位
I_{DD}	コンポーネントの消費電流					
	SMBus (UDB)	DataRate = 50kbps	–	380	–	µA
	SMBus (FF)		–	670	–	µA
	PMBus (UDB)	DataRate = 100kbps	–	440	–	µA
	PMBus (FF)		–	620	–	µA
	SMBus (UDB)	DataRate = 400kbps	–	720	–	µA
	SMBus (FF)		–	860	–	µA
	PMBus (UDB)	DataRate = 400kbps	–	750	–	µA
	PMBus (FF)		–	980	–	µA

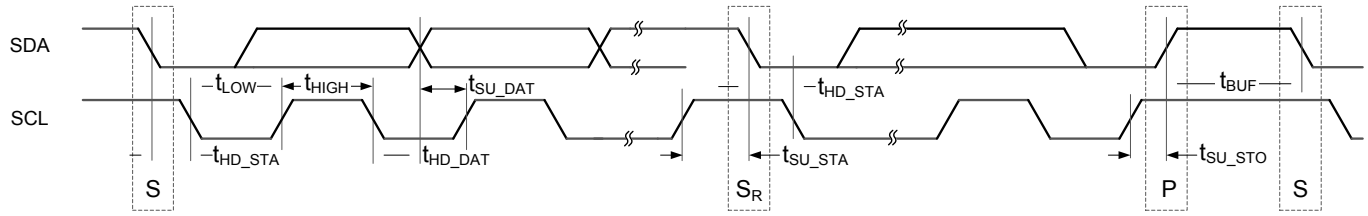
SMBus/ PMBus スレーブ AC 仕様

記号	項目	Min	Typ	Max	単位
f_{SCL}	SCL クロック周波数	– – –	– – –	100 400 1000	kHz
f_{CLOCK}	コンポーネント入力クロック周波数	–	16 × f_{SCL}	–	kHz
t_{RESET}	リセットパルス幅	–	2	–	t_{CY_clock}[3]
t_{LOW}	SCLクロックのLOW期間	4.7 1.3 0.5	– – –	– – –	µs
t_{HIGH}	SCLクロックのHIGH期間	4.0 0.6 0.26	– – –	– – –	µs
t_{HD_STA}	ホールド時間(リピート)スタートコンディション	4.0 0.6 0.26	– – –	– – –	µs
t_{SU_STA}	リピートスタートコンディションのセットアップ時間	4.7	–	–	µs

[2]. デバイス IO およびクロックの分布電流は含まれていません。値は 25°Cでの値です。データは BUS_CLK を 24MHz に設定して測定されています。
[3] $t_{CY_clock} = 1/f_{CLOCK}$. これは 1 クロック周期のサイクルタイムです

記号	項目	Min	Typ	Max	単位
		0.6 0.26	– –	– –	
t_{HD_DAT}	データホールド時間	5.0 – –	– – –	– – –	µs
t_{SU_DAT}	データセットアップ時間	250 100 50	– – –	– – –	ns
t_{SU_STO}	ストップコンディションのセットアップ時間	4.0 0.6 0.26	– – –	– – –	µs
t_{BUF}	ストップコンディションとスタートコンディションとの間のバス空き時間	4.7 1.3 0.5	– – –	– – –	µs
t_{FLASH_WRITE}		-	40.6[4]	-	ms

図. 6 データ遷移タイミング波形



変更履歴

ここでは、過去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
1.0	初版	

[4] PMBus モードに設定したレジスタストアを保存することにより値は得られます。BUS_CLK が 24MHz で、すべての PMBus コマンドが初期設定のままです。

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使用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提供することは、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提供することは、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。