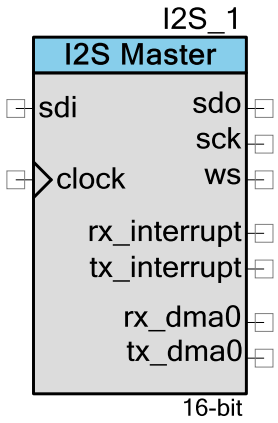
IC 間サウンド バス (I2S)

2.20

特徴



- マスタのみ
- サンプル当たり 8～32 データビット
- 16、32、48、64 ビットワード選択期間
- データレート最大 96 kHz まで (64 ビットワード選択期間) 6.144 MHz
- Tx および Rx FIFO 割り込み
- DMA サポート
- 独立型左・右チャネル FIFO またはインターリーブ型ステレオ FIFO
- Rx と Tx を独立イネーブル

一般的な説明

統合型 IC 間サウンド バス (I2S) は、複数のデジタルオーディオデバイスを接続するために使用されるシリアルバス インターフェース標準です。仕様は、Philips® セミコンダクター製です (I2S バス仕様; 1986 年 2 月、1996 年 6 月 5 日改訂)。

I2S コンポーネントはマスタ モードでのみ動作します。これはトランスミッタ (Tx) およびレシーバ (Rx) として二方向でも動作します。Tx と Rx のデータは独立したバイトストリームを構成します。バイトストリームには最初の最も重要なバイトと、最初のワードのビット 7 の最も重要なビットがパックされています。各サンプル (左または右チャンネルのサンプル) には、最小のバイト数が使用されます。

I2S の用途

コンポーネントはステレオオーディオデータのシリアルバスインターフェースを提供します。このインターフェースはオーディオ ADC および DAC コンポーネントで最も一般に使用されています。

入出力接続

このセクションでは、I2S 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。I/O リストのアスタリスク (*) は、I/O が、その I/O の説明でリストされている条件において、シンボルに隠れている可能性があることを示します。

sdi – 入力 *

シリアル データ入力 **Direction** パラメータに Rx オプションを選択した場合に表示します。

この信号が入力ピンに接続されている場合、このピンの [Input Synchronized] の選択はディスエーブルにされている必要があります。信号に遅延があると、入力ピン シンクロナイザによって、次のクロックサイクルにシフトできるよう、この信号は既に SCK に同期化されている必要があります。

clock – 入力

提供されるクロック レートは、出力シリアルクロック (SCK) の希望するクロック レートの 2 倍である必要があります。64 ビットワード選択期間を使用して、48-kHz のオーディオを生成するには、クロック周波数は以下のようになります:

2 × 48 kHz × 64 = 6.144 MHz

sdo – 出力 *

シリアル データ出力 **Direction** パラメータに TX オプションを選択すると、表示されます。

sck – 出力

シリアルクロックを出力します。

ws – 出力

ワードセレクト出力は送信されるチャンネルを示します。

rx_interrupt – 出力 *

Rx 方向の割り込み **Direction** パラメータに Rx オプションを選択した場合に表示します。

tx_interrupt – 出力 *

Tx 方向の割り込み **Direction** パラメータに TX オプションを選択すると、表示されます。

rx_DMA0 – 出力 *

FIFO 0 の Rx 方向 DMA 要求 (左またはインターリーブ) **DMA Request** パラメータに **Rx DMA** を選択すると、表示されます。

rx_DMA1 – 出力 *

FIFO 1 の Rx 方向 DMA 要求 (右)。**DMA Request** パラメータに **Rx DMA** を選択した場合、および **Data Interleaving** パラメータに **Separated L/R** を選択した場合、表示されます。

tx_DMA0– 出力 *

FIFO 0 の Tx 方向 DMA 要求 (左またはインターリーブ)。**DMA Request** パラメータに **Tx DMA** を選択した場合、表示されます。

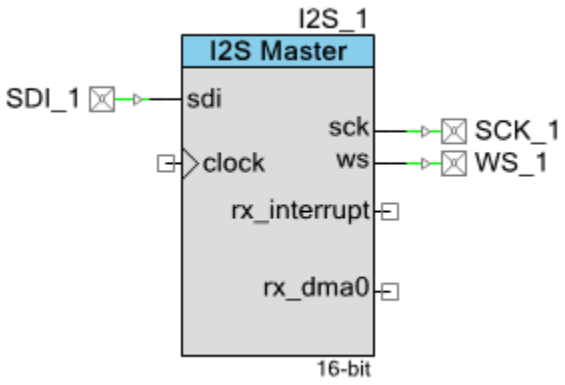
tx_DMA1 – 出力 *

FIFO 1 の Tx 方向 DMA 要求 (右)。**DMA Request** パラメータに **Tx DMA** を選択した場合、および Tx の **Data Interleaving** パラメータに **Separated L/R** を選択した場合、表示されます。

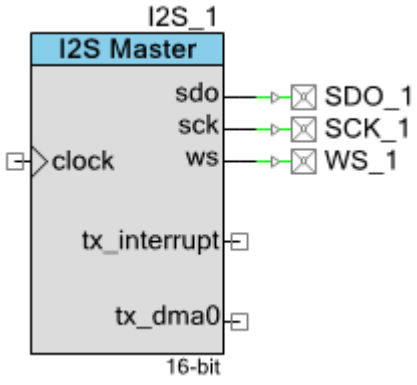
回路図マクロ情報

デフォルトで、PSoC Creator コンポーネント カタログは、I2S コンポーネントの 3 つの回路図マクロ実装を含みます。これらのマクロには、すでにデジタルピン コンポーネントに接続されている I2S が含まれています。SDI ピンの [Input Synchronized] オプションのチェックを解除し、全てのピンに対する API の生成を無効にします。回路図マクロは、以下のダイアグラムで示すように、Rx のみ、Tx のみ、および Rx と Tx の両方向用に構成された I2S コンポーネントを使用します。

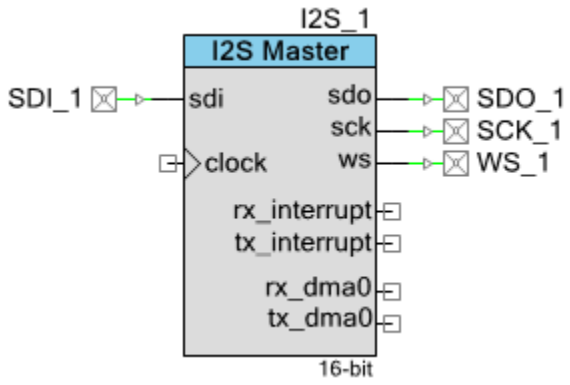
I2S Rx のみ



I2S Tx のみ



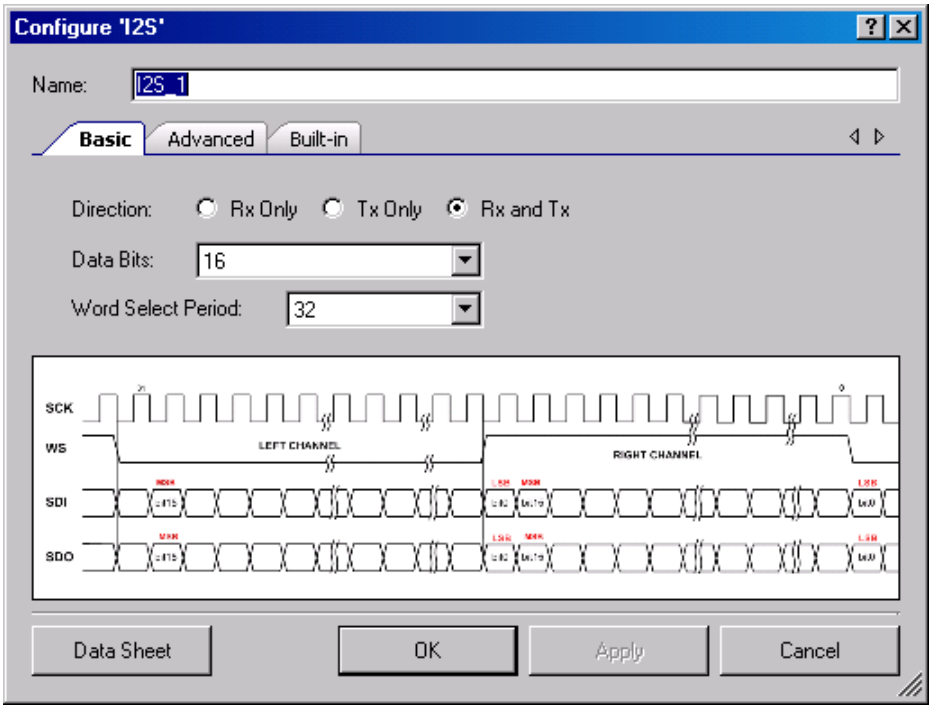
I2S Rx および Tx



コンポーネント パラメータ

I2S コンポーネントを設計上にドラッグし、ダブルクリックして **Configure (設定)** ダイアログを開きます。このダイアログには I2S コンポーネントのセットアップをガイドする 2 つのタブがあります。

基本タブ



方向

コンポーネントの動作方向を決定します。この値は **Rx Only**、**Tx Only**、**Rx and Tx** (デフォルト) に設定することができます。

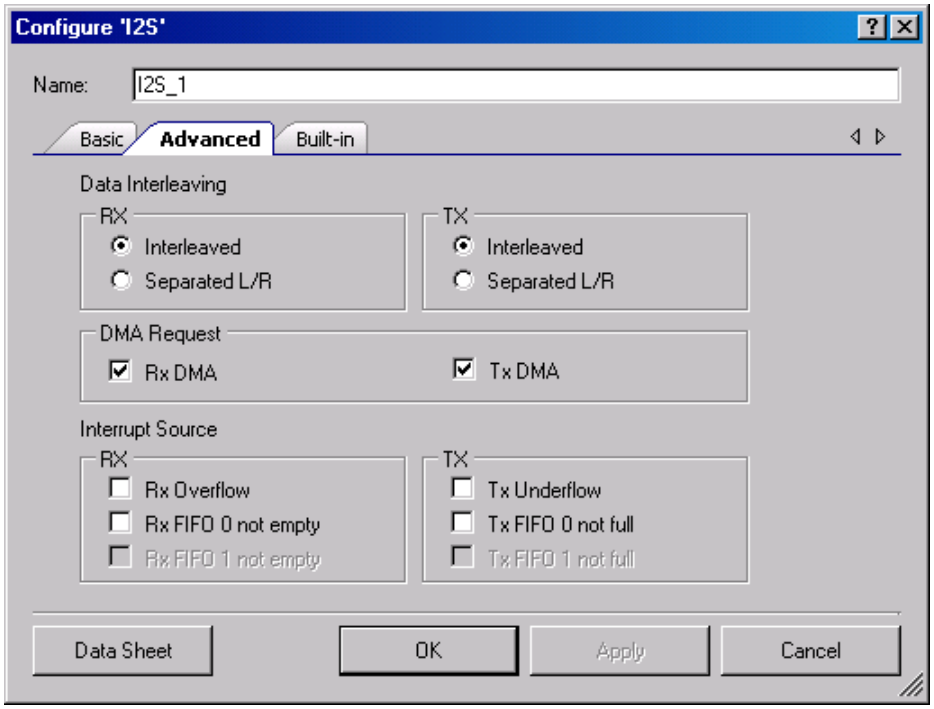
データビット

各サンプルに対して構成されるデータビットの数を決定します (コンパイルされたハードウェア)。この値は 8～32 の範囲で設定できます。デフォルトの設定は、**16** です。

ワード選択期間

左と右両チャンネルの完全なサンプルの期間を定義します。値のセット:**16**、**32** (デフォルト)、**48**、**64** に設定できます。

Advanced (詳細) タブ



データのインターリーブ

データが **Interleaved** (デフォルト) と **Separate L/R** のいずれかを選択できます。Rx と Tx を別々に選択します。

DMA 要求

コンポーネントの DMA リクエスト信号をイネーブルおよびディスエーブルにします。Rx と Tx を別々に設定します。このオプションはデフォルトでイネーブルにされています。

割り込みソース

I2S 割り込みのソースを選択します。Rx と Tx の割り込みは分離しています。複数のソースは OR で結合されることがあります。設定には以下を含みます。

- Rx:
  - **Rx オーバーフロー**
  - **Rx FIFO 0 は空ではありません** (左またはインターリーブ)
  - **Rx FIFO 1 は空ではありません** (右) - インターリーブされていない場合の唯一のオプション
- Tx:
  - **Tx アンダーフロー**
  - **Tx FIFO 0 はフルではありません** (左またはインターリーブ)
  - **Tx FIFO 1 は空ではありません** (右) - インターリーブされていない場合の唯一のオプション

Clock Select（クロック選択）

このコンポーネントには、内部クロックはありません。クロックソースを必ずつけてください。提供されるクロック レートは、出力シリアルクロック (SCK) の希望するクロック レートの 2 倍である必要があります。

配置

I2S コンポーネントは UDB アレイ全体に配置されます。すべての配置情報は *cyfitter.h* ファイルを介して API に提供されます。

リソース

リソース	リソース タイプ				API メモリ(バイト)		ピン(外部入出力当たり)
	データパス セル	PLD[1]	ステータス セル	Control/ Count7 セル	フラッシュ	RAM	
Rx 方向	1	2(3)	1	2	220	3	3
Tx 方向	1	2(3)	1	2	220	3	3
Rx および Tx	2	4(6)	2	2	326	3	4

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。続くセクションでは、各関数について詳しく説明します。

デフォルトでは、PSoC Creator が特定の設計のコンポーネントの最初のインスタンスにインスタンス名 “I2S_1” を割り当てます。インスタンス名は、識別子の構文ルールに従った固有の値に変更できます。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、下表では “I2S” というインスタンス名を使用しています。

関数	説明
I2S_Start()	I2S インタフェースを起動します。
I2S_Stop()	I2S インタフェースをディスエーブルにします。
I2S_EnableTx()	I2S インタフェースの Tx 方向をイネーブルにします。
I2S_DisableTx()	I2S インタフェースの Tx 方向をディスエーブルにします。
I2S_EnableRx()	I2S インタフェースの Rx 方向をイネーブルにします。
I2S_DisableRx()	I2S インタフェースの Rx 方向をディスエーブルにします。
I2S_SetRxInterruptMode()	I2S Rx 方向割り込みの割り込みソースを設定します。
I2S_SetTxInterruptMode()	I2S Tx 方向割り込みの割り込みソースを設定します。
I2S_ReadRxStatus()	I2S Rx ステータス レジスタの状態を返します。
I2S_ReadTxStatus()	I2S Tx ステータス レジスタの状態を返します。
I2S_ReadByte()	Rx FIFO から 1 バイトを返します。

[1] PLD の数は、特定の構成に使用する典型的および最大リソースの数を示します。

関数	説明
I2S_WriteByte()	Tx FIFO に 1 バイト書き込みます。
I2S_ClearRxFIFO()	Rx FIFO をクリアします。
I2S_ClearTxFIFO()	Tx FIFO をクリアします。
I2S_Sleep()	設定を保存し、I2S インタフェースをディスエーブルにします。
I2S_WakeUp()	設定を復元し、I2S インタフェースをイネーブルにします。
I2S_Init()	I2S インタフェースをイネーブルにします。
I2S_Enable()	I2S 設定を初期化またはデフォルト設定を復元します。
I2S_SaveConfig()	I2S インタフェースの設定を保存します。
I2S_RestoreConfig()	I2S インタフェースの設定を復元します。

グローバル変数

変数	説明
I2S_initVar	I2S が初期化されているかどうかを示します。この変数は、0 に初期化され、I2S_Start() が呼び出されたときに 1 に設定されます。これにより、I2S_Start() ルーチンへの最初の呼び出し後、再初期化せずに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、I2S_Start() を呼び出す前に I2S_Init() を呼び出します。または、I2S_Init() および I2S_Enable() 関数を呼び出すことで I2S を再初期化できます。

void I2S_Start(void)

説明: I2S インタフェースを起動します。アクティブ モードのパワー テンプレート ビットまたは該当する場合はクロック ゲーティングを有効にします。SCK および WS 出力の生成を開始します。Tx および Rx 方向はディスエーブルにされたままになります。

パラメータ: なし

戻り値: なし

副作用: なし

void I2S_Stop(void)

説明: I2S インタフェースをディスエーブルにします。アクティブ モードのパワー テンプレート ビットまたは該当する場合はクロック ゲーティングをディスエーブルにします。SCK および WS 出力は 0 に設定されます。Tx および Rx 方向はディスエーブルにされ、その FIFO はクリアされます。

パラメータ: なし

戻り値: なし

副作用: なし

void I2S_EnableTx(void)

説明: I2S インタフェースの Tx 方向をイネーブルにします。送信は、次のワード選択立ち下がりエッジで開始されます。

パラメータ: なし

戻り値: なし

副作用: なし

void I2S_DisableTx(void)

説明: I2S インタフェースの Tx 方向をディスエーブルにします。次のワードセレクトの立ち下がりエッジでデータ送信が停止され、定数 0 値が送信されます。

パラメータ: なし

戻り値: なし

副作用: なし

void I2S_EnableRx(void)

説明: I2S インタフェースの Rx 方向をイネーブルにします。データ受信は、次のワードセレクトの立ち下がりエッジで開始されます。

パラメータ: なし

戻り値: なし

副作用: なし

void I2S_DisableRx(void)

説明: I2S インタフェースの Rx 方向をディスエーブルにします。次のワードセレクトの立ち下がりエッジでは、データ受信は受信 FIFO に送信されません。

パラメータ: なし

戻り値: なし

副作用: なし

void I2S_SetRxInterruptMode(uint8 interruptSource)

説明: I2S Rx 方向割り込みの割り込みソースを設定します。複数のソースが OR 結合されます。

パラメータ: uint8: 選択された割り込みソースの定数を含むバイト

I2S Rx 割り込みソース	値
RX_FIFO_OVERFLOW	0x01
RX_FIFO_0_NOT_EMPTY	0x02
RX_FIFO_1_NOT_EMPTY	0x04

戻り値: なし

副作用: なし

void I2S_SetTxInterruptMode(uint8 interruptSource)

説明: I2S Tx 方向割り込みの割り込みソースを設定します。複数のソースが OR 結合されます。

パラメータ: uint8: 選択された割り込みソースの定数を含むバイト

I2S Tx 割り込みソース	値
TX_FIFO_UNDERFLOW	0x01
TX_FIFO_0_NOT_FULL	0x02
TX_FIFO_1_NOT_FULL	0x04

戻り値: なし

副作用: なし

uint8 I2S_ReadRxStatus(void)

説明: I2S Rx ステータス レジスタの状態を返します。

パラメータ: なし

戻り値: uint8: I2S Rx ステータス レジスタの状態

I2S Rx ステータス マスク	値	種類
RX_FIFO_OVERFLOW	0x01	Clear-on-Read
RX_FIFO_0_NOT_EMPTY	0x02	Transparent
RX_FIFO_1_NOT_EMPTY	0x04	Transparent

副作用: clear-on-read タイプの I2S Rx ステータス レジスタのビットをクリアします。

uint8 I2S_ReadTxStatus(void)

説明: I2S Tx ステータス レジスタの状態を返します。

パラメータ: なし

戻り値: uint8: I2S Tx ステータス レジスタの状態

I2S Tx ステータス マスク	値	種類
TX_FIFO_UNDERFLOW	0x01	Clear-on-Read
TX_FIFO_0_NOT_FULL	0x02	Transparent
TX_FIFO_1_NOT_FULL	0x04	Transparent

副作用: clear-on-read タイプの I2S Rx ステータス レジスタのビットをクリアします。

uint8 I2S_ReadByte(uint8 wordSelect)

説明: Rx FIFO から 1 バイトを返します。Rx FIFO が空でないことを確認するために、この呼び出しの前に Rx のステータスをチェックします。

パラメータ: uint8: 左 (0) または右 (1) チャンネルから読み出すことを示します。インターリーブ モードではこのパラメータは無視されます。

戻り値: uint8: 受信したデータを含むバイト

副作用: なし

void I2S_WriteByte(uint8 wrData, uint8 wordSelect)

説明: Tx FIFO に 1 バイトを書き込みます。Tx FIFO がフルでないことを確認するために、この呼び出しの前に Tx のステータスをチェックします。

パラメータ: uint8 wrData: 送信するデータを含むバイト

uint8 wordSelect: 左 (0) または右 (1) チャンネルから書き込むことを示します。インターリーブ モードでは、このパラメータは無視されます

戻り値: なし

副作用: なし

void I2S_ClearRxFIFO(void)

説明: Rx FIFO をクリアします。FIFO にあるデータは失われます。Rx 方向がディスエーブルになっている場合のみこの関数を呼び出します。

パラメータ: なし

戻り値: なし

副作用: なし

void I2S_ClearTxFIFO(void)

説明: Tx FIFO をクリアします。FIFO にあるデータは失われます。Tx 方向がディスエーブルになっている場合のみこの関数を呼び出します。

パラメータ: なし

戻り値: なし

副作用: なし

void I2S_Sleep(void)

説明: これは、コンポーネントのスリープを準備するのに推奨されるルーチンです。I2S_Sleep() ルーチンは現在のコンポーネントの状態を保存します。次に I2S_Stop() 関数を呼び出し、I2S_SaveConfig() を呼び出してハードウェア構成を保存します。アクティブ モードのパワー テンプレート ビットまたは該当する場合はクロック ゲーティングをディスエーブルにします。SCK および WS 出力は 0 に設定されます。Tx および Rx 方向はディスエーブルにされます。

CyPmSleep() または CyPmHibernate() 関数を呼び出す前に I2S_Sleep() 関数を呼び出します。パワーマネジメント関数については、PSoC Creator *System Reference Guide (システム リファレンス ガイド)* を参照してください。

パラメータ: なし

戻り値: なし

副作用: なし

void I2S_WakeUp(void)

説明: I2S 設定および非保持レジスタ値を復元します。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 ゲーティングをイネーブルにします。SCK および WS 出力の生成を開始します。スリープする前の状態に従って Rx/Tx 方向をイネーブルにします。

パラメータ: なし

戻り値: なし

副作用: まず I2S_Sleep() または I2S_SaveConfig() 関数を呼び出さずに I2S_Wakeup() 関数を呼び出すと予想外の動作が発生する場合があります。

void I2S_Init(void)

説明: I2S 設定を初期化するか、コンポーネントの割り込みソースを定義する、カスタマイザで提供されたデフォルトの設定を復元します。

パラメータ: なし

戻り値: なし

副作用: 割り込み生成のためのマスク レジスタのみを復元します。FIFO からのデータはクリアしませんし、コンポーネント ハードウェア状態マシンはリセットしません。

void I2S_Enable(void)

説明: ハードウェアの使用を開始し、コンポーネントの動作を開始します。I2S_Start() ルーチンがこの関数を呼び出し、それがコンポーネント操作を開始する優先方法であるため、I2S_Enable() を呼び出す必要はありません。

パラメータ: なし

戻り値: なし

副作用: なし

void I2S_SaveConfig(void)

説明: 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、I2S_Sleep() 関数で呼び出されます。

パラメータ: なし

戻り値: なし

副作用: なし

void I2S_RestoreConfig(void)

説明: 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、I2S_Sleep() 関数を呼び出す前のコンポーネント パラメータ値を復元します。このルーチンは I2S_Wakeup() によって呼び出され、スリープ状態を終了したときにコンポーネントを復元します。

パラメータ: なし

戻り値: なし

副作用: 呼び出す前に、まず I2S_SaveConfig() ルーチンを呼び出します。そうしないと、コンポーネント設定が初期設定によって上書きされます。

ファームウェア ソースコードのサンプル

PSoC Creator は、[Find Example Project] ダイアログに数多くのサンプルプロジェクトを提供しており、そこには回路図およびコード例が含まれています。コンポーネント固有の例を見るには、[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。一般例については、[Start Page] または **[File（ファイル）]** 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（サンプルプロジェクトを検索）」を参照してください。

機能の説明

左/右および Rx/Tx 設定

Rx および Tx 方向、ビット数、ワード選択期間などの左および右チャンネルの設定は同一です。アプリケーションで Rx および Tx の設定が異なる場合は、2 つの双方向コンポーネント インスタンスを使用する必要があります。

データ ストリーム フォーマット

Tx および Rx のデータは独立したバイト ストリームです。バイトストリームには最初の最も重要なバイトと、最初のワードのビット 7 の最も重要なビットがパックされています。各サンプルで使用されるバイト数 (左または右チャンネル) は、サンプルを格納するための最小バイト数です。未使用のビットは、Tx では無視され、Rx では 0 になります。

1 方向のデータ ストリームは 1 バイトのストリーム、または 2 バイト ストリームも可能です。1 バイト ストリームの場合、左および右チャンネルは、まず左チャンネル用のサンプルを使ってインターリーブされ、次に右チャンネルが処理されます。2 ストリームの場合、左および右チャンネル バイト ストリームは個別の FIFO を使用します。

DMA

I2S インタフェースは、データのストリームが中断されてはならない連続インタフェースです。多くのアプリケーションの場合、Tx 方向のアンダーフローまたは Rx 方向のオーバーフローを防止するため、DMA 転送を使用する必要があります。

I2S は各方向に最高 2 つの DMA コンポーネントを駆動できます。DMA Wizard を使って次のように DMA 処理を設定できます。

DMA ソース名/ DMA Wizard の送信先	方向	DMA 要求信号	DMA 要求タイプ	説明
I2S_RX_FIFO_0_PTR	Source (ソース)	rx_dma0	レベル	左またはインターリーブ チャンネルの FIFO を受信
I2S_RX_FIFO_1_PTR	Source (ソース)	rx_dma1	レベル	右チャンネルの FIFO を受信
I2S_TX_FIFO_0_PTR	Destination (転送先)	tx_dma0	レベル	左またはインターリーブ チャンネルの FIFO を送信
I2S_TX_FIFO_1_PTR	Destination (転送先)	tx_dma1	レベル	右チャンネルの FIFO を送信

すべての場合で、DMA 要求信号の"HIGH"信号は、さらに 1 バイト送信される可能性があることを示します。

イネーブル化

Rx と Tx 方向は個別にイネーブルにされます。イネーブルにされない場合、Tx 方向はすべて 0 値を送信し、Rx 方向はすべての受信データを無視します。イネーブル状態への、またはイネーブル状態からの変化は、左/右サンプル ペアが常に送信または受信されるようワード選択境界で発生します。

エラー処理

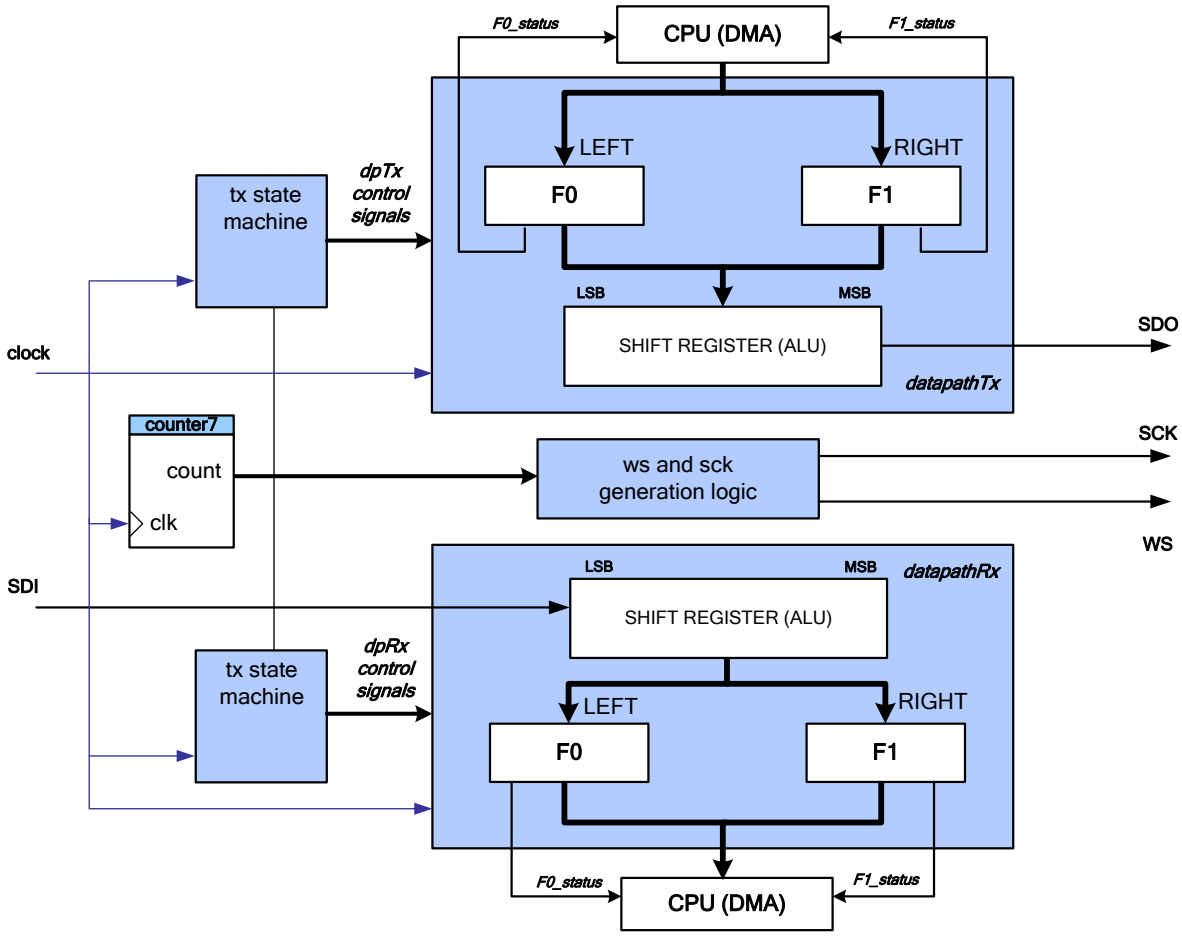
コンポーネントの 2 つのエラー条件は、送信 FIFO が空で次の読み出しが発生した (送信アンダーフロー)、または受信 FIFO が満杯で次の書き込みが発生した (受信オーバーフロー) 場合に発生します。

送信 FIFO が空になり、送信するデータがない場合(送信アンダーフロー)に送信がイネーブルにされると、コンポーネントによって 0 の継続送信が強制されます。送信を再開する前に、送信をディスエーブルにする必要があり、FIFO をクリアし、送信するデータをバッファに格納してから送信を再有効化します。CPU は、送信ステータス ビット I2S_TX_FIFO_UNDERFLOW を使用してこのアンダーフロー状態を監視できます。このエラー条件用に割り込みを設定できます。

受信 FIFO が満杯になり、さらにデータを受信した場合に受信がイネーブルにされる (受信オーバーフロー) と、コンポーネントはデータの取得を停止します。受信を再開する前に、受信をディスエーブルにする必要があり、FIFO をクリアしてから受信を再有効化します。CPU は、受信ステータス ビット I2S_RX_FIFO_OVERFLOW を使用してこのオーバーフロー状態を監視できます。このエラー条件用に割り込みを設定できます。

ブロックダイアグラムと設定

I2S は、一連の設定済み UDB として実装されています。実装は以下のブロック ダイアグラムに示されています。



レジスタ

I2S_CONTROL_REG

ビット	7	6	5	4	3	2	1	0
値	予約済み					enable	rxenable	txenable

- enable: I2S コンポーネントをイネーブル/ディスエーブルにします
- rxenable, txenable: Rx および Tx 方向をそれぞれイネーブル/ディスエーブルにします

I2S_TX_STATUS_REG

ビット	7	6	5	4	3	2	1	0
値	予約済み					F1_not_full	F0_not_full	アンダーフロー

- F1_not_full: 設定されている場合は、x FIFO 1 は満杯ではありません。
- F0_not_full: 設定されている場合は、Tx FIFO 0 が満杯ではありません
- underflow: 設定されている場合は、Tx FIFO アンダーフロー イベントが発生しています

レジスタ値は I2S_ReadTxStatus() API 関数を使用して読み出せます。

I2S_RX_STATUS_REG

ビット	7	6	5	4	3	2	1	0
値	予約済み					F1_not_empty	F0_not_empty	オーバーフロー

- F1_not_empty: 設定されている場合は、Rx FIFO 1 は空ではありません
- F0_not_empty: 設定されている場合は、Rx FIFO 0 は空ではありません
- overflow: 設定されている場合、Rx FIFO オーバーフロー イベントが発生しています

レジスタ値は I2S_ReadRxStatus() API 関数を使用して読み出せます。

DC 電気的特性と AC 電気的特性

以下の値は、期待される性能を示しており、初期特性データを基にしています。

「公称配線での最大」タイミング特性

パラメータ	説明	Min	Typ	Max [1]	単位
f_S	サンプリング周波数[2]	–	–	96	kHz
t_{WS}	ワードセレクト期間	16	–	64	ビット
f_{SCK}	シリアル クロック (出力) 周波数	–	$f_S \times t_{WS}$	6.144	MHz

[1]最大コンポーネント クロック周波数は、SCK 出力と SDI 入力の配線パス遅延と組み合わせて、t_{CLK_SCK} から派生します。これらの「公称」数は、公称配線条件でのコンポーネントの最大安全動作周波数を規定します。コンポーネントは、より高いクロック周波数で実行できますが、その場合、STA 結果を参照してタイミング要件を確認する必要があります。

[2] 最大ワード選択期間のサンプリング周波数が規定されています。

パラメータ	説明	Min	Typ	Max [1]	単位
f_{CLOCK}	コンポーネント クロック周波数	–	–	2 × f_{SCK}	MHz
t_{SCKH}	SCK 高レベル時間	–	0.5	–	1/f_{SCLK}
t_{SCKL}	SCK 低レベル時間	–	0.5	–	1/f_{SCLK}
t_{SCK_WS}	遅延時間、WS varidへの SCK 立下りエッジ	–20	–	20	ns
t_{SCK_SDO}	遅延時間、SDO varidへの SCK 立ち下りエッジ	–20	–	20	ns
t_{WS_SDO}	遅延時間、SDO varidへの WS エッジ	–20	–	20	ns
t_{S_SDI}	SDI セットアップ時間	25	–	–	ns

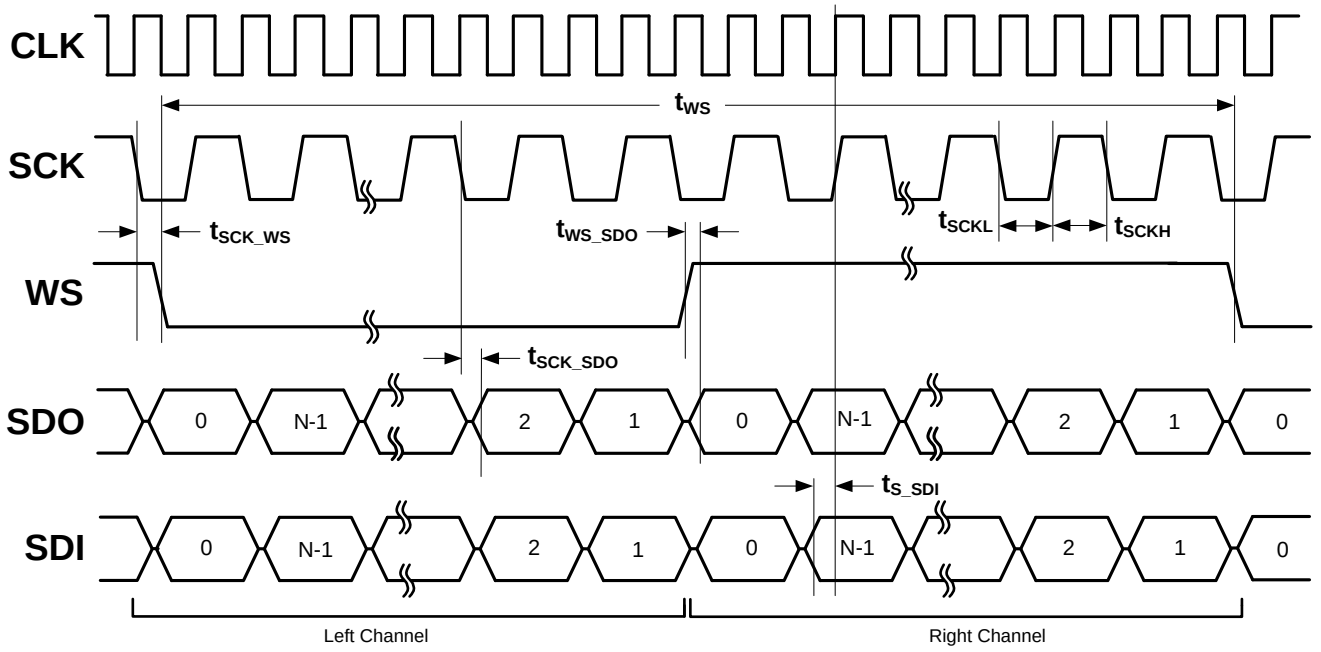
「すべての配線での最大」タイミング特性

パラメータ	説明	Min	Typ	Max [1]	単位
f_S	サンプリング周波数 [2]	–	–	96	kHz
t_{WS}	ワード選択期間	16	–	64	ビット
f_{SCK}	シリアル クロック (出力) 周波数	–	f_S × t_{WS}	6.144	MHz
f_{CLOCK}	コンポーネント クロック周波数	–	–	2 × f_{SCK}	MHz
t_{SCKH}	SCK "HIGH"レベル時間	–	0.5	–	1/f_{SCLK}
t_{SCKL}	SCK "LOW"レベル時間	–	0.5	–	1/f_{SCLK}
t_{SCK_WS}	SCK 立下りエッジからWS validへの遅延時間	–20	–	20	ns
t_{SCK_SDO}	SCK 立ち下りエッジからSDO validへの遅延時間	–20	–	20	ns
t_{WS_SDO}	WS エッジからSDO validへの遅延時間	–20	–	20	ns
t_{S_SDI}	SDI セットアップ時間	25	–	–	ns

[1] クロック周波数はこのコンポーネントの制限要素ではないため、「すべての配線」の最大値は「公称」と同じです。主な制限要素は、スレーブへ、およびスレーブの SDO 出力のパス遅延、およびマスターの SDI 入力に戻る、マスターのピンの SCK の立ち下がりエッジからのラウンド トリップ パス遅延です (本書の後記で説明)。

[2] 最大ワード選択期間のサンプリング周波数が規定されています。

図 1. データ トランザクションのタイミング ダイヤグラム



注 コンポーネント内部ロジックはすべて入力 2X クロックから動作します。これにより、t_{S_SDI} および t_{CLK_SCK} などの、この内部クロックに関するタイミング制約を参照する必要があります (本書の後記で説明)。

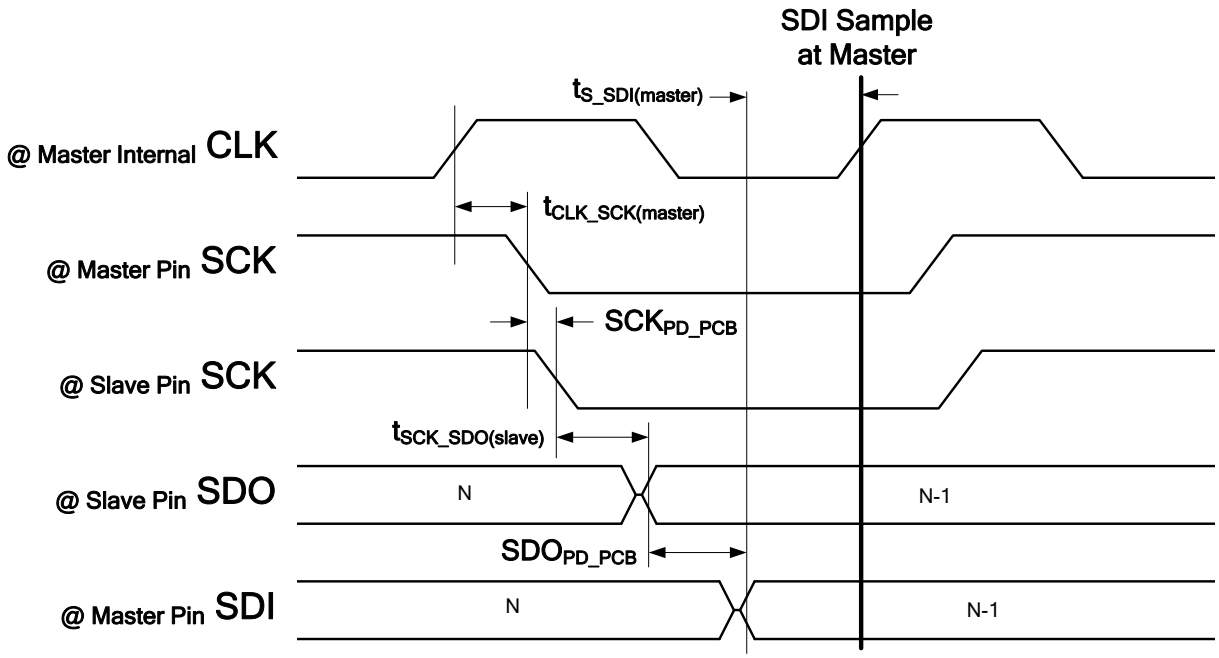
一般的な音声サンプリング周波数のためのコンポーネント クロック速度

サンプリング周波数	コンポーネント クロック周波数(f_{CLK}) MHz			
	t_{WS} = 16 ビット	t_{WS} = 32 ビット	t_{WS} = 48 ビット	t_{WS} = 64 ビット
8 kHz	0.2560	0.5120	0.7680	1.0240
16 kHz	0.5120	1.0240	1.5360	2.0480
32 kHz	1.0240	2.0480	3.0720	4.0960
44.1 kHz	1.4112	2.8224	4.2336	5.6448
48 kHz	1.5360	3.0720	4.6080	6.1440
88.2 kHz	2.8224	5.6448	8.4672	11.2896
96 kHz	3.0720	6.1440	9.2160	12.2880
192 kHz	6.1440	12.2880	該当なし	該当なし

特性データ用の STA 結果の使用方法

f_{SCK} SCK (または最大ビット レート) の最大周波数は STA では直接提供されません。ただし、STA 結果で提供されているデータは、内部ロジック タイミング制限の一部を示します。最大ビット レートを計算する際には、考慮すべき要因がいくつかあります。最大値を完全に理解するには、ボードのレイアウトとスレーブ通信デバイス仕様が必要です。このパラメータの主な制限要因は、マスターのピンでの SCK の立ち下がりエッジから、スレーブおよびスレーブまでの SDO 出力のパス遅延、マスターの SDI 入力へ戻るラウンドトリップ パス遅延です。この場合、コンポーネントは以下の計算式によるマスターでの SDI の設定時間に適合する必要があります。

図 2 最大 f_{SCK} 周波数の計算



この場合、f_{SCK} 周波数を以下の式を使用して計算する必要があります:

$$f_{SCK} < 1 \div [2 \times [t_{RT_PD} + t_{CLK_SCK(マスター)} + t_{S_SDI(マスター)}]]$$

ここで

$$t_{RT_PD} = [SCK_{PD_PCB} + t_{SCK_SDO(スレーブ)} + SDO_{PD_PCB}]$$

また:

SCK_{PD_PCB} は、マスター コンポーネントのピンからスレーブ デバイスのピンまでの SCK の PCB パス遅延です。

$t_{SCK_SDO(スレーブ)}$ は、スレーブ デバイス データシートから取得する必要があります

$t_{CLK_SCK(マスター)}$ は、マスター コンポーネントの内部 CLK から SCK ピン パスまでの遅延です。これは、以下のように、STA 結果の、クロックから出力までのセクションで説明されています:

- Clock To Output Section

- CLK

Source	Destination	Delay (ns)
\I2S:BitCounter\/count_0	SCK(0)_PAD	24.484
Net_5/q	SDO(0)_PAD	23.808
Net_4/q	WS(0)_PAD	22.958

$t_{S_SDI(マスター)}$ は、マスター コンポーネントの SDI ピンから内部ロジック パスまでの遅延です。これは以下のように、STA 結果の、入力からクロックまでのセクションで説明されています:

- Input To Clock Section

- CLK

Source	Destination	Delay (ns)
SDI(0)_PAD	\I2S:Rx:dpRx:u0\/route_si	19.977

SCK の最大周波数、つまり最大ビット レートを求める最終計算式は:

$$f_{SCK}\text{(最大)} = 1 \div [2 \times [t_{CLK_SCK(マスター)} + SCK_{PD_PCB} + t_{SCK_SDO(スレーブ)} + SDO_{PD_PCB} + t_{S_SDI(マスター)}]]$$

f_{CLOCK} 最大コンポーネント クロック周波数は、名前付き外部クロック (この場合、CLK) としてクロック サマリのタイミング結果に表示されます。STA レポートからの内部クロック限界の例は以下の通りです:

+ Clock Summary Section

Clock	Type	Nominal Frequency (MHz)	Required Frequency (MHz)	Maximum Frequency (MHz)	Violation
BUS_CLK	Sync	24.000	24.000	N/A	
CLK	Sync	6.000	6.000	65.398	
ClockBlock/clk_bus	Async	24.000	24.000	N/A	
ClockBlock/dclk_0	Async	6.000	6.000	N/A	
ILO	Async	0.001	0.001	N/A	
IMO	Async	3.000	3.000	N/A	
MASTER_CLK	Sync	24.000	24.000	N/A	
PLL_OUT	Async	24.000	24.000	N/A	

t_{SCKH} I2S コンポーネントは 50 パーセントのデューティ サイクル SCK を生成します

t_{SCKL} I2S コンポーネントは 50 パーセントのデューティ サイクル SCLK を生成します

t_{SCK_WS} SCK 立ち下りエッジと WS valid 間の遅延。この値は、WS および SCK ピンのクロックから出力時間の間の差として計算できます。データは、以下のように、STA レポートから抽出できます:

- Clock To Output Section

- CLK

Source	Destination	Delay (ns)
\I2S:BitCounter\/count_0	SCK(0)_PAD	24.484
Net_5/q	SDO(0)_PAD	23.808
Net_4/q	WS(0)_PAD	22.958

t_{SCK_SDO} SCK 立ち下りエッジと WS valid 間の遅延。この値は、SDO および SCK ピンのクロックから出力時間の間の差として計算できます。値は、以下のように、クロックから出力時間までの STA 結果に表示されます:

- Clock To Output Section

- CLK

Source	Destination	Delay (ns)
\I2S:BitCounter\/count_0	SCK(0)_PAD	24.484
Net_5/q	SDO(0)_PAD	23.808
Net_4/q	WS(0)_PAD	22.958

t_{WS_SDO} WS エッジと SDO valid 間の遅延。値は、以下のように、STA レポートのクロックから出力時間までのセクションを開くことで規定できます:

- Clock To Output Section

- CLK

Source	Destination	Delay (ns)
\I2S:BitCounter\/count_0	SCK(0)_PAD	24.484
Net_5/q	SDO(0)_PAD	23.808
Net_4/q	WS(0)_PAD	22.958

t_{S_SDI} SDI 設定時間とは、マスター コンポーネントの SDI ピンから内部ロジックパスまでの遅延です。これは以下のように、STA 結果の、入力からクロックまでのセクションで規定されています:

- Input To Clock Section

- CLK

Source	Destination	Delay (ns)
SDI(0)_PAD	\I2S:Rx:dpRx:u0\/route_si	19.977

コンポーネントの変更

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
2.20	コンポーネントによる内部 FIFO エラー検知を追加。	オーバーフロー/アンダーフロー エラーが発生した場合のコンポーネントによるエラー処理を改善。
2.10	DPクロックに対する再サンプルされた FIFOブロックステータス信号。	これですべての PSoC 3 および PSoC 5 シリコンの同じタイミング結果で、コンポーネントが機能します。
	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	

バージョン	変更の説明	変更の理由 / 影響
2.0	このコンポーネントのハードウェア実装を、クロック入力で 2X 周波数信号を必要とするよう変更。SCK 出力信号は、受信クロックを 2 で除算して生成されます。	SCK、WS、SDO および SDI 信号間のタイミング関係の制御を改善。
	I2S_Start() 関数を実装の変化に適合するよう更新。機能性は変更されていません。	実装の簡略化には初期化を変更する必要がありました。
	スリープ モード API が追加されています。	低電力モードをサポートするため。
	ステータス ビットtx_not_full および rx_not_emty は、clear-on-read から transparent モードに変更されました。	FIFO からの Full または Empty ステータスのステータス ビットが、FIFO の現在のライブ ステータスのみを表すには transparent である必要があります。
	コンポーネントに DMA 機能ファイルを追加。	このファイルにより、I2S を PSoC Creator の DMA ウィザード ツールでサポートすることが可能になります。
	I2S_Stop() API は、コンポーネントがディスエーブルにされた後、rx および tx FIFO をクリアするよう変更されました。	Tx および Rx FIFO ステータスを初期値にリセットします。コンポーネントの再有効化後の予想外の動作を防止します。
	Keil 関数再入可能性のサポートを追加。	顧客が個々に生成された関数を再入可能性として指定する能力を追加しました。

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。