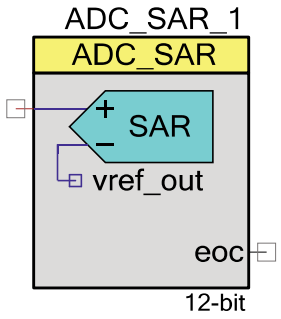
# ADC 逐次比較型レジスタ（ADC_SAR）

1.71

## 特長




- PSoC 5 デバイスをサポート
- 最大 700 ksps で 12 ビットの分解能
- 4 つの消費電力モード
- 選択可能な分解能およびサンプル レート
- シングルエンド入力または差動入力

## 概要説明

ADC 逐次比較型レジスタ（ADC_SAR）コンポーネントは中速（最大 700 ksps のサンプリング）、中分解能（最大 12 ビット）、アナログ-デジタル変換を提供します。

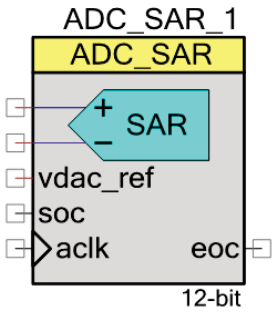
### ADC_SAR の用途

ADC_SAR コンポーネントの標準的なアプリケーションには次のようなものがあります。

- LED 照明管理
- モータ制御
- 磁気カードリーダ
- 高速データ収集
- 電力メータ
- パルスオキシメーター（心拍酸素濃度計）

## 入出力接続

ここのセクションでは、ADC_SAR の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、シンボルから隠されている可能性があることを示します。




## +入力 – アナログ

この入力は ADC_SAR への正のアナログ信号入力です。変換結果は+入力信号 – リファレンス電圧の関数となります。リファレンス電圧は–入力信号か $V_{SSA}$ のいずれかになります。

## –入力 – アナログ *

表示された場合、このオプション入力は ADC_SAR への負のアナログ信号(リファレンス)入力となります。変換結果は+入力 – –入力の関数となります。**入力範囲**パラメータを差動モードの 1 つに設定するとこのピンが現れます。

## vdac_ref – 入力 *

VDAC リファレンス (vdac_ref)はオプション ピンです。**Vssa から VDAC*2 (シングル エンド)**または **0.0 +/- VDAC (差動)**入力範囲を選択した場合、これが表示されます。その他の場合、この I/O は非表示になります。このピンのみが VDAC コンポーネントの出力に接続できます。他の信号には接続しないでください。

## soc – 入力 *

変換開始入力(soc)はオプション ピンです。**Triggered** サンプル モードを選択すると表示されます。この入力の立ち上がりエッジで ADC 変換を開始します。**Sample Mode** パラメータを **Free Running** に設定すると、この I/O は非表示になります。

## aclk – 入力 *

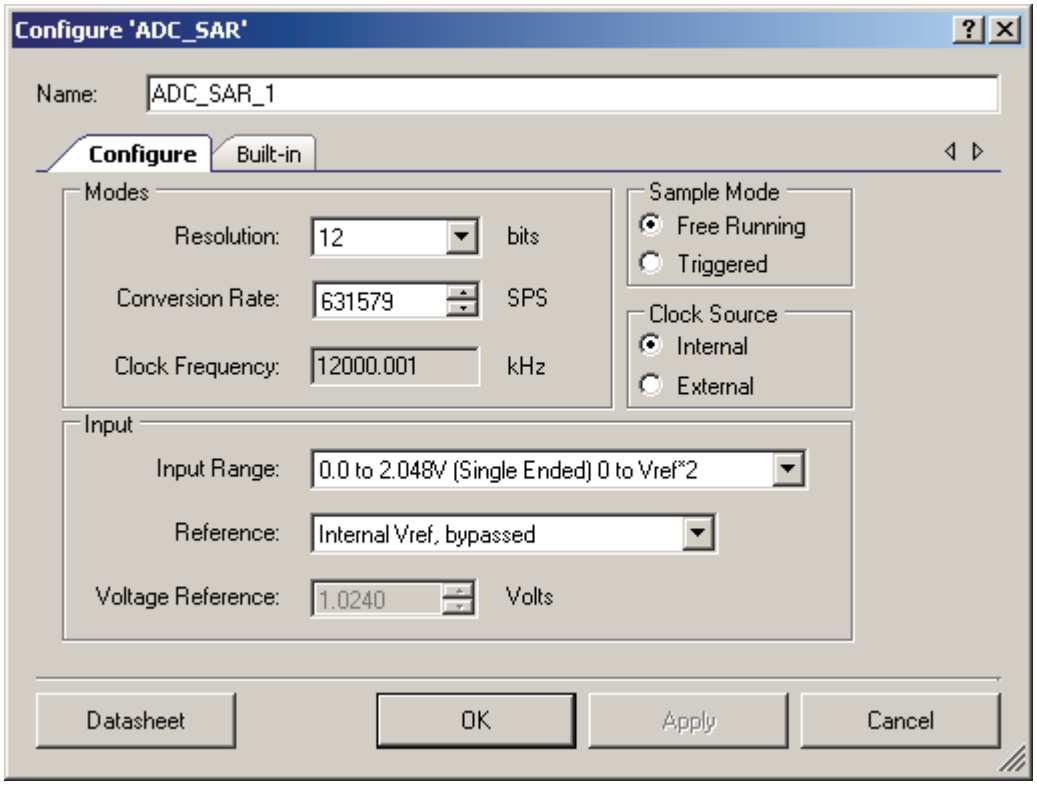
このオプション ピンは **Clock Source** パラメータを **External** に設定すると表示されます。その他の場合、このピンは非表示になります。このクロックは、変換の方法と解像度の関数として変換速度を決定します。

## eoc – 出力

変換完了(eoc)出力の立ち上がりエッジは、変換動作が完了したことを表しています。DMA リクエスト入力にこのピンを接続して、変換出力をシステム RAM、DFB、または他のコンポーネントに転送できます。内部割込みをこの信号に接続したり、自分自身の割り込みを接続することもできます。

# コンポーネント パラメータ

ADC_SAR 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。




ADC_SAR には次のパラメータがあります。太字で表示されたオプションはデフォルトです。

## モード

### 分解能

ADC の分解能の設定。

| ADC_Resolution | 値 | 説明 |
|---|---|---|
| **12** | 12 | 分解能を 12 ビットに設定。 |
| 10 | 10 | 分解能を 10 ビットに設定。 |
| 8 | 8 | 分解能を 8 ビットに設定。 |

SAR は常に 12 ビット モードで動作します。8 および 10 ビットのオプションが残っていますが、ADC_GetResult16() および ADC_GetResult8() API のみに影響を与えます。

### Conversion Rate

このパラメータは ADC 変換を設定します。変換時間は変換レートと逆の関係になります。変換レートを 1 秒あたりのサンプル数で入力します。1 つのサンプルの変換には 19 SAR ADC クロック サイクルを要します。

### Clock Frequency

このテキスト ボックスは読み取り専用（常に灰色表示）で、その領域には選択された動作条件（解像度や変換速度）の必要クロック速度が表示されています。これらの条件のいずれか、またはその両方が変更された場合、更新されます。クロック周波数は 1 MHz から 14 MHz の間であれば動作可能です。デューティ比は 50 パーセントにしてください。最小パルス幅は 33 ns 以上にしてください。PSoC Creator は、クロックがそれらの制限範囲に入らない場合、ビルドプロセス中にエラーを生成します。その場合、Design-Wide Resources Clock Editor の Master Clock を変更してください。

## Sample Mode

このパラメータは ADC の動作方法を決定します。

| Start_of_Conversion | 説明 |
|---|---|
| **Free Running** | ADC は継続して動作します。 |
| Triggered | SOC ピンの立ち上がりエッジのパルスで1回の変換を開始します。 |

## Clock Source

このパラメータにより、ADC_SAR モジュールに対し内部にあるクロックまたは外部クロックのいずれかを選択できるようになります。

| ADC_Clock | 説明 |
|---|---|
| **Internal** | ADC_SAR の内部クロックを使用します。 |
| External | 外部クロックを使用します。クロック ソースはアナログかデジタル、またはその他のコンポーネントによって生成されます。 |

# Input

## Input Range

このパラメータは ADC を指定された入力範囲になるようコンフィギュレーションします。PSoC に接続されたアナログ信号は入力範囲設定に関係なく、$V_{SSA}$ と $V_{DDA}$ 間の値にしなければなりません。

| Input Range | 説明 |
|---|---|
| **0.0 から 2.048V まで（シングル エンド）**<br>**0 から Vref*2 まで** | 内部リファレンス（1.024 V）使用時、使用可能な入力範囲は 0.0 から 2.048 V です。ADC は、シングル エンド入力モードで動作するように構成されます。この入力モードでは Vrefhi_out に内部接続された-入力を利用します。外部リファレンス電圧を使用している場合、使用できる入力範囲は 0 から Vref*2 までです。 |
| Vssa から Vdda まで（シングル エンド） | このモードはリファレンス $V_{DDA}/2$ を使用します。使用可能入力範囲はすべてのアナログ供給電圧に対応しています。ADC は Vrefhi_out へ内部的に接続された –入力を用いて、シングルエンド入力モードに設定されます。 |
| Vssa から VDAC*2（シングル エンド） | このモードはVDACリファレンス を使用し、vdac_ref ピンに接続する必要があります。使用可能な入力範囲は Vssa から VDAC*2 ボルトです。ADC は Vrefhi_out へ内部的に接続された –入力を用いて、シングル エンド入力モードで動作するように設定されます。 |
| 0.0 ± 1.024V（差動）<br>–入力 ± Vref | このモードは差動入力用に設定されます。内部リファレンス（1.024 V）を使用すると、入力範囲は –入力 ± 1.024 V となります。<br>例えば、–入力が 2.048 V に接続されている場合、使用可能な入力範囲は 2.048 ± 1.024 V または 1.024 から 3.072 V です。シングルエンドと差動信号の両方がスキャンされるシステムの場合、シングルエンド入力のスキャン時に、–入力を Vssa へ接続します。<br>外部リファレンスを使用して、より広い動作範囲を提供することができます。同一の方程式、–入力 ± Vref を用いて使用可能な入力範囲を計算できます。 |
| 0.0 ± Vdda（差動）<br>–入力 ± Vdda | このモードは差動入力用に設定され、出力特性が供給電圧と比例します。入力範囲は–入力 ± Vdda となります。シングルエンドと差動信号の両方がスキャンされるシステムの場合、シングルエンド入力のスキャン時に、–入力を Vssa へ接続します。 |
| 0.0 ± Vdda/2（差動）<br>–入力 ± Vdda/2 | このモードは差動入力用に設定され、出力特性が供給電圧に比例します。入力範囲は–入力 ± Vdda/2 となります。シングルエンドと差動信号の両方がスキャンされるシステムの場合、シングルエンド入力のスキャン時に、–入力をVssa へ接続します |
| 0.0 ± VDAC（差動）<br>–入力 ± VDAC | このモードは差動入力用に設定され、VDACリファレンス を使用し、vdac_ref ピンに接続する必要があります。入力範囲は<br>–入力 ± VDAC となります。シングルエンドと差動信号の両方がスキャンされるシステムの場合、シングルエンド入力のスキャン時に、–入力をVssa へ接続します。 |

## リファレンス

このパラメータは ADC_SAR のリファレンス設定用スイッチを選択します。

| ADC_Reference | 説明 |
|---|---|
| 内部 Vref | 内部リファレンスの使用 このオプションで許容される最大サンプリング レートは、100,000 sps |

| | |
|---|---|
| | です。より高い変換速度の場合は**Internal Vref, bypassed** オプションを使用してください。 |
| **内部 Vref、バイパス** | 内部リファレンスの使用。SAR1 は P0[2]* ピンに、または SAR0 は P0[4]* ピンにバイパス コンデンサを配置する必要があります。 |
| 外部 Vref | SAR1 は P0[2]* ピンに、または SAR0 は P0[4]* ピンで外部リファレンスを使用します。 |

* デジタル スイッチングに起因する内部ノイズがアプリケーションのアナログ性能要件を超える場合、外部バイパス コンデンサの使用を推奨します。このオプションを使用するには、アナログ HI-Z ピンなどの P0[2] または P0[4] ポート ピンのいずれかを構成し、0.01 µF から 10 µF の値を持つ外部コンデンサを接続してください。

### リファレンス電圧

リファレンス電圧アプリケーション プログラミング インタフェースのセクションで説明されている、カウント数から電圧への変換関数で使用します。このパラメータは内部リファレンス使用する場合、読み取り専用です。外部リファレンス使用する場合、この値を編集して外部リファレンス電圧に一致させることができます。

- 入力範囲として **Vssa から Vdda**、**-入力 +/- Vdda**、または **-入力 +/- Vdda/2** を選択する場合、$V_{DDA}$ 供給電圧を入力してください。
- 入力範囲を **Vssa から VDAC*2** または **–入力 +/- VDAC** を選択する場合、VDAC 供給電圧値を入力してください。

**注記** 入力範囲およびリファレンス電圧は $V_{DDA}$ 電圧によって制限されます。

## 配置

ADC_SAR コンポーネントは 2 つの利用可能な SAR ブロックの 1 つに配置されます。配置情報は、*cyfitter.h* ファイルを通して API に提供されます。デフォルトの配置を変更する必要がある場合、Design-Wide Resources – Directives Editor（プロジェクトの .cydwr ファイル内）を使用してパラメータを編集します。

## リソース

ADC_SAR はシリコンおよびクロック ソースの固定ブロック SAR を使用します。

| リソース | リソースのタイプ | | | | API メモリ (バイト) | | ピン (外部入出力ごと) |
|---|---|---|---|---|---|---|---|
| | クロック分周器 | マクロセル | 割り込み | SAR 固定ブロック | フラッシュ | RAM | |
| 8 ～ 12 ビット | 1 | 1 | 1 | 1 | 1106 | 7 | 1 |

# 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。以後のセクションで、各関数について詳しく説明します。

デフォルトでは、PSoC Creator は、インスタンス名「ADC_SAR_1」を設計上のコンポーネントの最初のインスタンスに割り当てます。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、定数名の接頭辞になります。読みやすいように、下表では「ADC」というインスタンス名を使用しています。

| 関数 | 説明 |
|---|---|
| ADC_Start() | ADC を起動してすべての状態をリセットします |
| ADC_Stop() | ADC 変換を停止して電力を最小に低減します |
| ADC_SetPower() | 消費電力モードを設定します |
| ADC_SetResolution() | ADC の分解能を設定します |
| ADC_StartConvert() | 変換を開始します |
| ADC_StopConvert() | 変換を停止します |
| ADC_IRQ_Enable() | 内部 IRQ が eoc に接続されます。この API が内部 ISR を有効にします。 |
| ADC_IRQ_Disable() | 内部 IRQ が eoc に接続されます。この API が内部 ISR を無効にします。 |
| ADC_IsEndConversion() | 変換が完了した場合、ゼロ以外の値を返します |
| ADC_GetResult8() | 符号付き 8 ビット変換結果 を返します |
| ADC_GetResult16() | 符号付き 16 ビット変換結果 を返します |
| ADC_SetOffset() | ADC のオフセットを設定します |
| ADC_SetGain() | 1 ボルトあたりのカウントで ADC ゲインを設定します |
| ADC_CountsTo_Volts() | ADC カウントを浮動小数点電圧に変換します |
| ADC_CountsTo_mVolts() | ADC カウントをミリボルトに変換します |
| ADC_CountsTo_uVolts() | ADC カウントをマイクロボルトに変換します |
| ADC_Sleep() | ADC 処理を停止し、ユーザ設定を保存します |
| ADC_Wakeup() | ユーザ設定を復元し、有効にします |
| ADC_Init() | カスタマイザで提供されたデフォルト設定を初期化します |
| ADC_Enable() | ADC のクロックおよび電源を有効にします |
| ADC_SaveConfig() | 現在のユーザー設定を保存します |

| 関数 | 説明 |
|---|---|
| ADC_RestoreConfig() | ユーザー設定を復元します |

## グローバル変数

| 変数 | 説明 |
|---|---|
| ADC_initVar | この変数は ADC が初期化されたかどうかを示します。変数は、0 に初期化され、ADC_Start() が初めて呼び出されたときに 1 に設定されます。これにより、コンポーネントは ADC_Start() ルーチンへの最初の呼び出し後、再初期化せずに再起動できます。<br>コンポーネントの再初期化が必要な場合、関数 ADC_Start() や ADC_Enable() の前に、関数 ADC_Init() が呼び出されます。 |
| ADC_offset | この変数はオフセットを補正します。これは 1 に設定されており、最初に ADC_Start() が呼び出されて、ADC_SetOffset() を使用して修正できます。変数は、特定のオフセットを減算することで ADC_CountsTo_Volts()、ADC_CountsTo_mVolts()、および ADC_CountsTo_uVolts() 関数に影響を与えます。 |
| ADC_countsPerVolt | この変数はゲインを補正するのに使用されます。これが計算されて最初に ADC_Start() が呼び出され、そのたびに ADC_SetResolution() が呼び出されます。この値は分解能、入力範囲、およびリファレンス電圧に依存します。それは ADC_SetGain() を使用して変更できます。<br>この変数は、ADC カウントと印加入力電圧との間で適切な変換を提供し、ADC_CountsTo_Volts()、ADC_CountsTo_mVolts()、および ADC_CountsTo_uVolts() 関数に影響を与えます。 |
| ADC_shift | 別の入力モードでは、SAR ADC はデジタル処理で変換されたデータをバイナリのオフセット スキームで出力します。この変数は、ADC カウントを 2 の補数形式に変換するのに使用されます。<br>この変数は、最初に ADC_Start() が呼び出されたとき、及び ADC_SetResolution() が呼び出されるたびに計算されます。計算値は分解能と入力モードに依存します。<br>この変数は、適切なシフト値を減算することで ADC_GetResult8() および ADC_GetResult16() 関数に影響を与えます。 |

## void ADC_Start(void)

**説明:** これは、コンポーネントの動作を開始する方法として、推奨されています。ADC_Start() は initVar 変数を設定し、ADC_Init() 関数を呼び出して、ADC_Enable() 関数を呼び出します。

**パラメータ:** なし

**戻り値:** なし

**注意事項:** initVar 変数がすでに設定されている場合は、この関数は ADC_Enable() 関数を呼び出すだけです。

## void ADC_Stop(void)

**説明:** ADC 変換を停止して電力を最小に低減します。

**注記** この API は ADC の電源を遮断しませんが、その電力を最小に低減します。このデバイスには、デバイスに電源が供給されていない場合、複数のアナログ リソースに接続すると信頼性の低下を引き起こすという不具合があります。信頼性の低下は、そのリソースを使用しているコンポーネントが停止した際に、サイレントな欠陥 (例:アナログ コンポーネントの予期しない不良発生) という形で現れます。

**パラメータ:** なし

**戻り値:** なし

**注意事項:** なし

## void ADC_SetPower(uint8 power)

**説明:** ADC の動作電力を設定します。高速なクロック速度の時ほど、高い電力設定を使用する必要があります。

**パラメータ:** uint8 power: 電源設定

| パラメータ名 | 値 | 説明 | クロック速度 |
|---|---|---|---|
| ADC__HIGHPOWER | 0 | 標準パワー | 14 MHz |
| ADC__MEDPOWER | 1 | 1/2 パワー | 7 MHz |
| ADC__LOWPOWER | 2 | 1/3 パワー | 4.6 MHz |
| ADC__MINPOWER | 3 | 1/4 パワー | 3.5 MHz |

**戻り値:** なし

**注意事項:** 電源設定は変換精度に影響を与える場合があります。

## void ADC_SetResolution(uint8 resolution)

**説明:** GetResult16() および GetResult8() API の分解能を設定します。この関数は実際の変換には影響しません。

**パラメータ:** uint8 resolution: 分解能設定

| パラメータ名 | 値 | 説明 |
|---|---|---|
| ADC__BITS_12 | 12 | 分解能を 12 ビットに設定。 |
| ADC__BITS_10 | 10 | 分解能を 10 ビットに設定。 |
| ADC__BITS_8 | 8 | 分解能を 8 ビットに設定。 |

**戻り値:** なし

**注意事項:** ADC の分解能は変換サイクル中は変更することができません。推奨される最も良い実行方法は、ADC_StopConvert() を用いて変換を停止し、分解能を変更してから、ADC_StartConvert() を用いてリスタートすることです。

この API を呼び出す前に変換を停止したくない場合には、ADC_IsEndConversion() を使用して変換が完了するまで待ってから分解能を変更します。

変換中に ADC_SetResolution() を呼び出す場合、分解能は進行中の変換が完了するまで変更されません。6 +「新規分解能（ビット単位）」クロック サイクルの間、新しい分解能のデータは有効になりません。データが再び有効になる前に ADC_SetResolution() を呼び出した後で、このクロック サイクル数の遅延を追加しなければならない場合があります。

ADC カウントと印加入力電圧との間で適切な変換を計算することで ADC_CountsTo_Volts()、ADC_CountsTo_mVolts()、および ADC_CountsTo_uVolts() に影響を与えます。計算は解像度と入力範囲、およびリファレンス電圧に依存します。

## void ADC_StartConvert(void)

**説明:** ADCに変換を開始させます。フリーラン モードでは、ADC は連続して実行されます。トリガ モードでは、この関数が SOC のソフトウェア バージョンとして動作し、変換を行うたびに ADC_StartConvert() でトリガされる必要があります。

**パラメータ:** なし

**戻り値:** なし

**注意事項:** ADC_StartConvert() を呼び出すと外部 SOC ピンが無効になります。

## void ADC_StopConvert(void)

**説明:** 強制的に ADC による変換を停止します。変換が実行中の場合、変換は完了しますがこれ以上の変換は行われなくなります。

**パラメータ:** なし

**戻り値:** なし

**注意事項:** トリガ モードでは、この関数が SOCのソフトウェア バージョンを'L'レベルに設定し、SOC ソースをハードウェア SOC 入力に切り替えます。

## void ADC_IRQ_Enable(void)

**説明:** 割り込みを変換終了時に発生できるようにします。グローバル割り込みも ADC 割り込みが発生するように、イネーブルする必要があります。グローバル割り込みを有効化するには、すべての割り込みを有効にする前にmain.cファイルにある、グローバル割り込み有効化マクロ「`CYGlobalIntEnable;`」を 呼び出します。

**パラメータ:** なし

**戻り値:** なし

**注意事項:** 割り込みの発生をイネーブルします。結果を読み込むと割り込みはクリアされます。

## void ADC_IRQ_Disable(void)

**説明:** 変換終了割り込みをディセーブルします。

**パラメータ:** なし

**戻り値:** なし

**注意事項:** なし

## uint8 ADC_IsEndConversion(uint8 retMode)

**説明:** すぐに変換のステータスを返すか、変換が完了するまで返す（ブロックします）かはretMode パラメータに依存します。

**パラメータ:** uint8 retMode: 変換リターン モードを検査します。オプションについては下表を参照してください。

| オプション | 説明 |
|---|---|
| ADC_RETURN_STATUS | すぐに変換のステータスを返します。値がゼロを返した場合、変換は完了していません。この関数はゼロ以外の結果が返されるまでリトライする必要があります。 |
| ADC_WAIT_FOR_RESULT | ADC 変換が完了するまで結果を返しません。 |

**戻り値:** uint8: ゼロ以外の値が返された場合、最後の変換が完了します。返された値がゼロの場合、ADCは引き続き最後の結果を計算し続けます。

**注意事項:** この関数は変換ステータスの終了を読み込むと、読み込み時点でクリアされます。

## int8 ADC_GetResult8(void)

**説明:** 8 ビット変換結果 を返します 分解能が 8 ビットより大きな値に設定されている場合、結果の LSB を返します。この関数は、分解能が 12 ビット未満に設定されているとシフト値を返します。
ADC_IsEndConversion() は、データ サンプルが準備されていることを確認するために呼び出す必要があります。

**パラメータ:** なし

**戻り値:** int8: 最後の ADC 変換の LSB。

**注意事項:** ADC カウントを 2 の補数形式に変換します。

## int16 ADC_GetResult16(void)

**説明:** 8 から 12 ビットの分解能での変換結果に対する 16 ビットの結果を返します。この関数は、分解能が 12 ビット未満に設定されているとシフト値を返します。ADC_IsEndConversion() は、データ サンプルが準備されていることを確認するために呼び出す必要があります。

**パラメータ:** なし

**戻り値:** int16: 最後の ADC 変換の 16 ビット変換結果

**注意事項:** ADC カウントを 2 の補数形式に変換します。

## void ADC_SetOffset(int16 offset)

**説明:** ADC_CountsTo_Volts()、ADC_CountsTo_mVolts()、および ADC_CountsTo_uVolts() で使用されるADC オフセットを設定し、、電圧変換の前にオフセットを特定の読み込み値から減算します。

**パラメータ:** int16 offset: この値は入力が短絡したり、同じ入力電圧に接続された場合に測定されます。

**戻り値:** なし

**注意事項:** 特定の読み込み値から減算することで、ADC_CountsTo_Volts()、ADC_CountsTo_mVolts()、および ADC_CountsTo_uVolts() に影響を与えます。

## void ADC_SetGain(int16 adcGain)

**説明:** 1 ボルトあたりのカウントで電圧変換関数に ADC ゲインを設定します。この値はデフォルトではリファレンスと入力範囲設定により設定されます。既存の入力を用いて更に ADC を補正する場合か、ADC が外部リファレンスを使用している場合に限り使用してください。

**パラメータ:** int16 adcGain: 1 ボルトあたりのカウントでの ADC ゲイン

**戻り値:** なし

**注意事項:** ADC カウントと印加入力電圧との間で適切な変換を提供することで ADC_CountsTo_Volts()、ADC_CountsTo_mVolts() および ADC_CountsTo_uVolts() に影響を与えます。

## float ADC_CountsTo_Volts(int16 adcCounts)

**説明:** ADC 出力を浮動小数点数として電圧に変換します。例えば、ADC が 0.534 ボルトを実測した場合、戻り値は 0.534 となります。

**パラメータ:** int16 adcCounts: ADC 変換からの結果

**戻り値:** Float: ボルトでの結果

**注意事項:** なし

## int16 ADC_CountsTo_mVolts(int16 adcCounts)

**説明:** ADC 出力を 16 ビットの整数としてミリボルトに変換します。例えば、ADC が 0.534 ボルトを実測した場合、戻り値は 534 となります。

**パラメータ:** int16 adcCounts: ADC 変換からの結果

**戻り値:** int16: mV での結果

**注意事項:** なし

## int32 ADC_CountsTo_uVolts(int16 adcCounts)

**説明:** ADC 出力を 32 ビットの整数としてマイクロボルトに変換します。例えば、ADC が 0.534 ボルトを実測した場合、戻り値は 534000 となります。

**パラメータ:** int16 adcCounts: ADC 変換からの結果

**戻り値:** int32: µV での結果

**注意事項:** なし

## void ADC_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望ましいルーチンです。ADC_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、ADC_Stop() 関数を呼び出し、ADC_SaveConfig() を呼び出してハードウェア 設定を保存します。

CyPmSleep() または CyPmHibernate() 関数を呼び出す前に、ADC_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 *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

**注意事項:** なし

## void ADC_Wakeup(void)

**説明:** これは、コンポーネントを ADC_Sleep() が呼び出されたときの状態に復元するのに推奨されるルーチンです。ADC_Wakeup() 関数は、設定を復旧させるために ADC_RestoreConfig() 関数を呼び出します。このコンポーネントが ADC_Sleep() 関数を呼び出す前に有効化された場合、ADC_Wakeup() 関数はコンポーネントを再度有効化します。

**パラメータ:** なし

**戻り値:** なし

**注意事項:** 最初に ADC_Sleep() または ADC_SaveConfig() 関数を呼び出すことなく ADC_Wakeup() 関数を呼び出すと、予期されない振る舞いにつながる可能性があります。

## void ADC_Init(void)

**説明:** カスタマイザの [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。ADC_Start() ルーチンが ADC_Init() 関数を呼び出すので、この関数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

**戻り値:** なし

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

## void ADC_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。クロック速度に応じて、より高出力な電源が自動的に設定されます。ADC_SetPower() API の説明にはクロックレートに起因する電源の関係が含まれています。ADC_Start() ルーチンが ADC_Enable() 関数を呼び出すので、この関数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

**戻り値:** なし

**注意事項:** なし

## void ADC_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、ADC_Sleep() 関数で呼び出されます。

**パラメータ:** なし

**戻り値:** なし

**注意事項:** すべての ADC 構成レジスタは保持されます。この関数は実装されておらず、将来の使用のために保持されています。ここでは API がコンポーネント全体にわたって整合性を保つように提供されます。

### void ADC_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。この関数はまた、コンポーネントのパラメータ値を ADC_Sleep() 関数を呼び出す前の状態に復旧します。

**パラメータ:** なし

**戻り値:** なし

**注意事項:** 最初に ADC_Sleep() または ADC_SaveConfig() 関数を呼び出すことなく、この関数を呼び出すと、予期しない動作につながる可能性があります。この関数は実装されておらず、将来の使用のために保持されています。ここでは API がコンポーネント全体にわたって整合性を保つように提供されます。

## DMA

DMA コンポーネントを使用して ADC_SAR レジスタから RAM へ変換結果を転送することができます。DMA データ要求信号(DRQ)を ADC から EOC ピンに接続してください。DMA Wizard を使って次のように DMA 処理を設定できます。

| DMA ソースの名前 | Length (長さ) | 方向 | DMA 要求信号 | DMA 要求タイプ | 説明 |
|---|---|---|---|---|---|
| ADC_SAR_WRK0_PTR | 2 | Source (ソース) | EOF | 立ち上がりエッジ | 常に 12 ビットの分解能での変換結果に対する 2 バイトの結果を受信します。<br>このレジスタには符号拡張されていないことに注意してください。変換結果には常に符合無しです。0-V の差動入力はハーフ スケール コード(フルスケールの1/2の値を返します。負の下限レベルの時はコード 0 を返し、正の上限レベルの時はフル スケール コードを返します。 |

# ファームウェア ソースコードの例

PSoC Creator は、Find Example Project ダイアログに数多くのプロジェクト例を提供しており、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 (スタート ページ)」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 (サンプルプロジェクトを探す)」を参照してください。

# 割り込みサービスルーチン

ADC_SAR は *ADC_SAR_1_INT.c* ファイルにブランク割り込みサービス ルーチンを含み 、ここでは「ADC_SAR_1」がインスタンス名となります。カスタム コードを指定領域に配置して、変換終了時に必要とされる関数は何でも実行することができます。ブランク割り込みサービス ルーチンのコピーを下記に示します。カスタム コードを「/* `#START MAIN_ADC_ISR`」および「/* `#END` */"」コメントの間に配置します。これにより、プロジェクトが再生成されるとコードが維持されることを確認できます。

```
CY_ISR( ADC_SAR_1_ISR )
{
    /* Place user ADC ISR code here. This can be a good place */
    /* to place code that is used to switch the input to the  */
    /* ADC. It may be good practice to first stop the ADC     */
    /* before switching the input then restart the ADC.       */

    /* `#START MAIN_ADC_ISR` */
    /* Place user code here. */
    /* `#END` */
}
```

2 番目の指定領域には、変数定義および定数定義を配置することができます。

```
/* System variables */

/* `#START ADC_SYS_VAR` */
    /* Place user code here. */
/* `#END` */
```

割り込みを使用して以下のデータをキャプチャするコードの例

```
#include <device.h>

int16  result = 0;
uint8 dataReady = 0;
void main()
{
    int16 newReading = 0;
    CYGlobalIntEnable;           /* Enable Global interrupts */
    ADC_SAR_1_Start();           /* Initialize ADC */
    ADC_SAR_1_IRQ_Enable();      /* Enable ADC interrupts */
    ADC_SAR_1_StartConvert();    /* Start ADC conversions */
    for(;;)
    {
        if (dataReady != 0)
        {
           dataReady = 0;
           newReading = result;
           /* More user code */
        }
    }
}
```

*ADC_SAR_1_INT.c* ファイル内の割り込みコード セグメント。

```
    /**********************************
```

```
    *      System variables
    *********************************/
    /* `#START ADC_SYS_VAR`  */
     extern int16 result;
     extern uint8 dataReady;
     /* `#END`  */

CY_ISR(ADC_SAR_1_ISR )
{
    /*********************************************/
    /* Place user ADC ISR code here.             */
    /* This can be a good place to place code    */
    /* that is used to switch the input to the   */
    /* ADC. It may be good practice to first     */
    /* stop the ADC before switching the input   */
    /* then restart the ADC.                     */
    /*********************************************/
    /* `#START MAIN_ADC_ISR`  */
        result = ADC_SAR_1_GetResult16();
        dataReady = 1;
    /* `#END`  */
}
```

変換レートおよびマスタ クロック パラメータを正しく設定することが重要です。

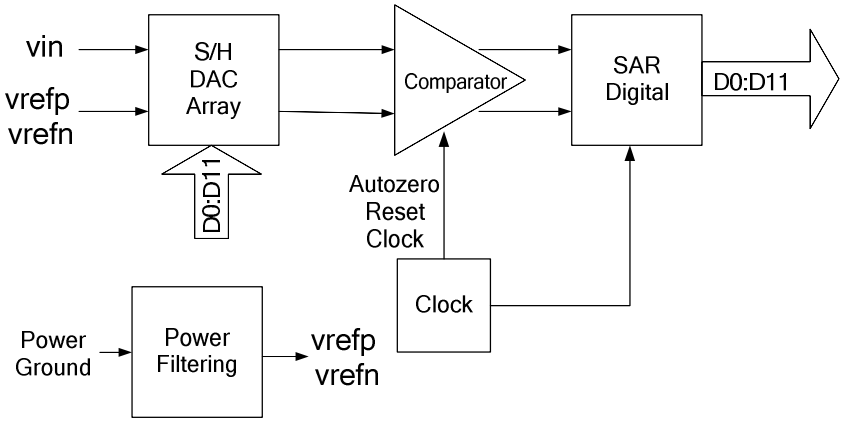
例えば、最大変換レート(12 ビットで 700 ksps)では、マスタ クロックを Design-Wide Resources Clock Editor で 53 MHz に設定し、ISR ルーチンを最適化します。そうしないと、プロセッサは十分な速度で ISR を処理できなくなります。低速マスタ クロックを選択すると、ISR の実行時間は ADC_SAR の変換時間よりも長くなります。

以下のサンプル レジスタを読み込むことで、ISR を最適化することができます。

```
CY_ISR(ADC_SAR_1_ISR )
{
    /*********************************************/
    /* Place user ADC ISR code here.             */
    /* This can be a good place to place code    */
    /* that is used to switch the input to the   */
    /* ADC. It may be good practice to first     */
    /* stop the ADC before switching the input   */
    /* then restart the ADC.                     */
    /*********************************************/
    /* `#START MAIN_ADC_ISR`  */
        result = CY_GET_REG16(ADC_SAR_1_SAR_WRK0_PTR);
        dataReady = 1;
    /* `#END`  */
}
```

# 機能説明

下図はブロック ダイアグラムを示します。入力アナログ信号は、サンプリングされ、二分探索アルゴリズムを使って DAC の出力と比較されて、MSB から LSB の方向へ順番に変換ビットが決まります。




# レジスタ

## サンプル レジスタ

ADC 結果は、分解能で 8 から 12 ビットの間となります。出力は 2 つの 8 ビット レジスタに分割されます。CPU または DMA はこれらのレジスタにアクセスして、ADC 結果を読み込むことができます。

### ADC_SAR_WRK0_REG (SAR ワーキング レジスタ 0)

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | Data[7:0] | | | | | | | |

### ADC_SAR_WRK1_REG (SAR ワーキング レジスタ 1)

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | overrun_det | 適用外 | | | Data[11:8] | | | |

- Data[11:0]: ADC 結果
- overrun_det: データ オーバラン検出フラグ。この関数はデフォルトでは無効です。

# DC 電気的特性と AC 電気的特性

次の値は予想されるパフォーマンスを示唆したもので、初期の特性データに基づいています。特記されていない場合の動作条件:

- 連続サンプル モードでの動作
- Fclk = 14 MHz
- 入力範囲 = ±$V_{REF}$
- 10 µF のバイパス コンデンサ

## SAR ADC の DC 仕様

| パラメータ | 説明 | 条件 | 最小値 | Typ | 最大値 | 単位 |
|---|---|---|---|---|---|---|
| | 分解能 | | 8 | – | 12 | ビット |
| | チャネルの数ーシングル エンド | | – | – | GPIO の数 | – |
| | チャネルの数ー差動 | 差動ペアは、一組の GPIO を使用して形成 | – | – | GPIO の数/2 | – |
| | 繰り返し[1] | | 有 | – | – | |
| Ge | ゲイン エラー | 外部リファレンス | – | – | ±0.2 | % |
| $V_{OS}$ | 入力オフセット電圧 | $V_{CM}$ = 0 V | – | – | ±2 | mV |
| | | $V_{CM}$ = $V_{DD}$/2 | – | – | ±6 | |
| $I_{DD}$ | 消費電流 | | – | – | 1 | mA |
| | 入力電圧範囲ーシングル エンド [1] | | $V_{SSA}$ | – | $V_{DDA}$ | V |
| | 入力電圧範囲ー差動[[1] | | $V_{SSA}$ | – | $V_{DDA}$ | V |
| | 外部リファレンス入力電圧範囲 | | 1.0 | – | $V_{DDA}$ | V |
| PSRR | 電源ノイズ除去比 [1] | | 70 | – | – | dB |
| CMRR | 同相除去比 | | 35 | – | – | dB |
| INL | 積分非直線性 [1] | $V_{BG}$ からの内部リファレンス | – | – | ±2 | LSB |
| DNL | 微分非直線性 [1] | $V_{BG}$ からの内部リファレンス | – | – | ±2 | LSB |

[1] デバイスの特性評価に基づく値 (生産試験されたものではない)

## SAR ADC の AC 仕様

| パラメータ | 説明 | 条件 | 最小値 | Typ | 最大値 | 単位 |
|---|---|---|---|---|---|---|
| | サンプリング速度 [2] | バイパス コンデンサ付き | – | – | 700 | ksps |
| | | バイパス コンデンサ無し | – | – | 100 | |
| | 起動時間 [2] | | – | – | 10 | µs |
| SINAD | 信号対ノイズ比 [2] | $V_{DDA} \le 3.6$ V、$V_{REF} \le 3.6$ V | 57 | – | – | dB |
| | | 3.6 V < $V_{DDA} \le 5.5$ V<br>$V_{REF} < 1.3$ V または $V_{REF} > 1.8$ V | 57 | – | – | |
| THD | 全高調波歪み [2] | $V_{DDA} \le 3.6$ V、$V_{REF} \le 3.6$ V | – | – | 0.1 | dB |
| | | 3.6 V < $V_{DDA} \le 5.5$ V<br>$V_{REF} < 1.3$ V または $V_{REF} > 1.8$ V | – | – | 0.1 | |

[2] デバイスの特性評価に基づく値 (生産試験されたものではない)

# 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 |
|---|---|---|
| 1.71 | ADC_GetResult8() および ADC_GetResult16() API を修正し、2 つの 8 ビット読み込みの代わりに 1 つの 16 ビット読み込み処理を実行します。 | バイトの 1 つが読み込まれた後で、SAR ADC が出力サンプリング レジスタを更新すると、結果のデータが破損することがあります。 |
| | ADC_IsEndConversion() API を修正して、EOF ステータス ビットが解放されるまで待ちます。 | この関数は、クイック連続コール後に予期されない変換完了ステータスを返すことができます。 |
| 1.70 | SampleRate エラー プロバイダ メッセージの修正最小値。 | |
| | リファレンスのドロップダウン リストから、VDAC が Input Range として選択される場合に、External Vref 項目を非表示にします。 | 外部リファレンスは、VDAC 範囲が選択されている場合、使用できません。 |
| | External Vref オプションを選択する場合、外部ピンを「ExtVref」に名前変更します。バイパス オプション付きの内部リファレンスを選択する場合、名前の「Bypass」は保持されます。 | ピン名を機能的に一致させるには。 |
| | データ シートの訂正 | |

| バージョン | 変更の説明 | 変更の理由 / 影響 |
|---|---|---|
| 1.60 | 「Power」パラメータをカスタマイザから除去します。 | クロック速度に応じて、よりハイパワーに自動的に設定されます。ADC_SetPower() API の説明にはクロックレートに起因する電源の関係が含まれています。 |
| | SAR は 12 ビット モードで動作します。8 および 10 ビット オプションは維持されますが、ADC_GetResult16() API に影響を与えるだけです。 | SAR ADC が 8 または 10 ビット モードの出力として ODD カウントを表示するだけです。 |
| | デフォルト SAR 変換レートは1 Msps から 631579 sps (12 MHz クロック)に変更されます。 | SAR はデフォルト設定で配置および構築できるようにする必要があります。 |
| | ADC_Stop() API はADC の電源を遮断しませんが、その電力を最小に低減します。 | PSoC 5 シリコンは、電源が供給されていない場合、複数のアナログ リソースに接続すると信頼性が低下するという不具合があります。 |
| | 変換時間を 18 から 19 サイクルに変更します。 | SAR の性能を向上するには。 |
| 1.50.a | クロック周波数の確認を追加します。 | この変更は仕様外のクロックでの SAR ADC 使用を回避する方法を提供します。<br>SAR ADC コンポーネントのバージョンを 1.10 から更新して動作範囲外でクロックを使用する場合、適切なクロック周波数を選択してください。 |
| | 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。 |
| | データシートのマイナーな編集と更新 | |
| 1.50 | 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。 |
| | ADC_CountsTo_Volts および ADC_CountsTo_uVolts API を追加します。 | 拡張機能。この API は Volts および uVolts の変換結果を返します。 |
| | DMA 機能ファイルにコンポーネントを追加します。 | このファイルにより、PSoC Creator の ADC_SAR が DMA ウィザード ツールでサポートされるようになります。 |
| | ADC カウントの 2 の補数形式への変換は ADC_GetResult8 および ADC_GetResult16 の API で実装されています。同様に ADC_CountsTo_mVolts 関数からも削除されます。 | この変更は ADC DelSig との一貫性を保つために実施済みです。 |