インクリメンタル ADC データシート ADCINC V 1.1

Copyright © 2002-2010 Cypress Semiconductor Corporation. All Rights Reserved.

リソース	PSoC® ブロック				API メモリ （バイト数）			ピン （外部入出力ごと）
	デジタル	アナログ CT	アナログ SC		フラッシュ		RAM	
変調器 (1st または 2nd 次)		1st および 2nd	1st	2nd	1st	2nd		
CY8C29xxx, CY8C24x94, CY8C23x33, CY7C64215, CY8CLED04 / 16, CY8CLED0xD, CY8CLED0xG, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C28x45, CY8C28x43, CY8C28x52, CY8CPLC20, CY8CLED16P01								
	1	0	1	2	273	322	8	1
CY8C27 / 24 / 22xxx, CY8CLED08	1	0	1	2	226	275	8	1

その他のコンバータ向け ADC 選択ガイドは、AN2239 をご覧ください。当ユーザ モジュールを使用した、構成済みの機能的なプロジェクト例は下記のウェブサイトでご覧ください。
www.cypress.com/psocexampleprojects.

特性および概要

- 6 ~ 14-bit 分解能
- シンクロナス 8-bit PWM 出力が可能
- ディファレンシャル入力を選択可能
- 符号付きまたは符号なしの データ フォーマット
- 最大 15.6ksps までのサンプリング速度 (6-bit 分解能)
- 入力範囲は内部及び外部リファレンス オプションによって選択可能
- 内部または外部クロック

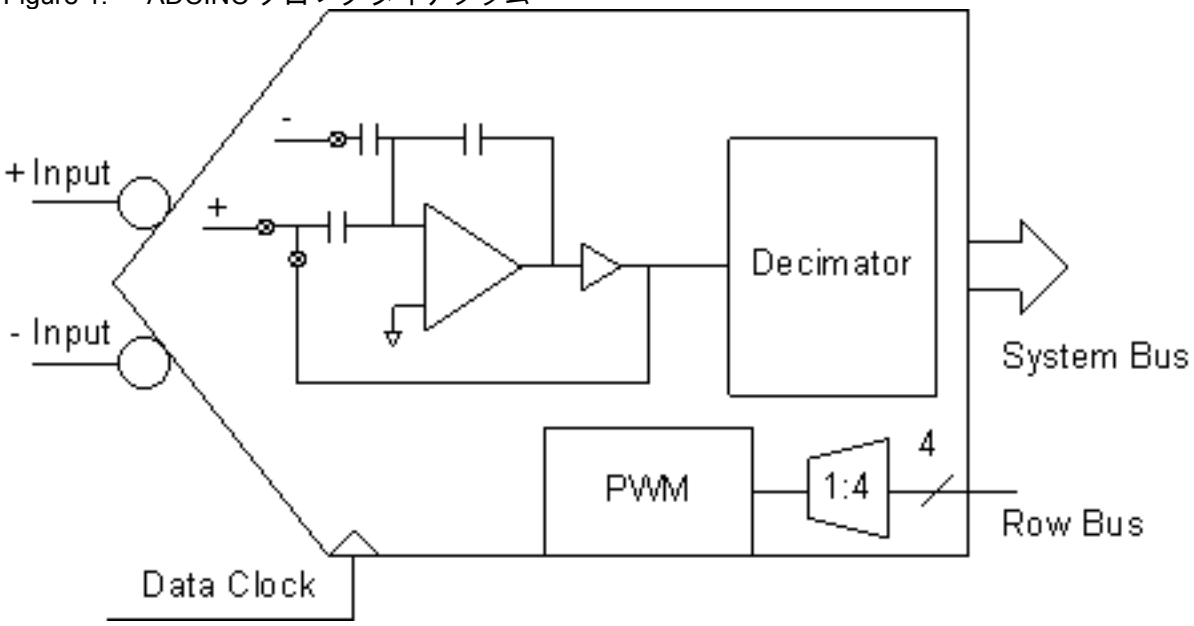
Note このユーザ モジュールは、29k ファミリで使用すると、さらに 6mA を消費します。代わりに、ADCINCVR ユーザ モジュールを使用してください。

ADCINC は差動または単一入力 ADC で、6 ~ 14 ビットになります。データ クロックの最大周波数は 8MHz ですが、より高い直線性を確保するためには、最大周波数は 2MH z にしてください。この ADC は、デジタル ブロックではなくハードウェア デシメータを使用して実装しているため、1 個のみ設置できます。これは最もリソース効率の高い ADC です。2 次変調器はスイッチド キャパシタ ブロックを追加することで実装することが可能であり、8MHz のデータ クロックでもより高い直線性を達成します。

タイミングは 8 ビット PWM で実装され、入力サンプルと同期する変調したパルス幅が得られます。

ADCINC は n ビットの分解能で出力を生成するために、2^n–1 の積分サイクルを必要とします。

Figure 1. ADCINC ブロック ダイアグラム



クイック スタート

事前設定されたプロジェクト例は下記のサイトからダウンロードしていただけます。
www.cypress.com/psocexampleprojects. ADCINC プロジェクトを作成するには

1. PSoC Designer から **New Project（新規プロジェクト）を選択してください。**
2. 新規プロジェクトのダイアログ ボックスで、**Designer Only Project を選択します。**. プロジェクトの名前と保存先を選択し、**OK をクリックします。**.
3. そして **New Base Part ダイアログ ボックスで、**該当するユーザ モジュールをサポートするデバイスをひとつ選択します。このデータシートの始めにあるリソース表をご覧ください。
4. そして **User Module Catalog （ユーザ モジュール カタログ）で** ADC リストの展開をクリックします。**ADCs**ADCINC フォルダーを展開します。**ADCINC** folder. User Module Catalog が表示されていない場合は、**View（表示）をクリックし、User Module Catalog を選択してください。** .
5. ADCINC には一段変調器または二段変調器を備えたものがあります。どちらのバージョンを使ったらいいか不明な場合は、どちらかの上で右クリックして **データシートを選択し、**決定する前にもう一度シートを見直してください。
6. ユーザ モジュール を右クリックし、**Place を選択します。**.

機能説明

ADCINC は以下の図で示すように、単一のアナログ スイッチド キャパシタ PSoC ブロック、ひとつのデジタル PSoC ブロック、およびデシメータで構成される 1 次変調器を提供します。

Figure 2. 1 次変調器を備えた ADCINC の回路図

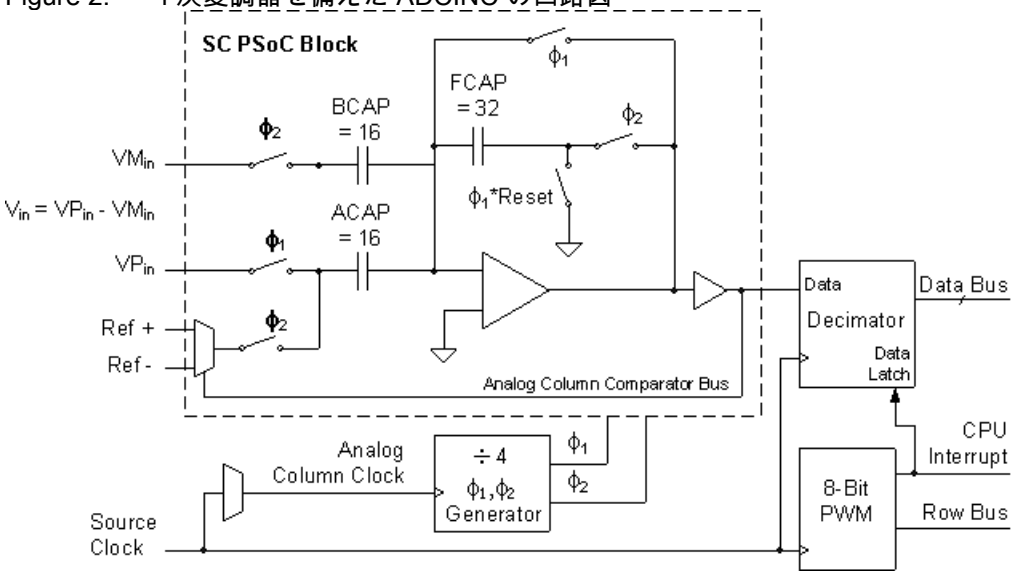
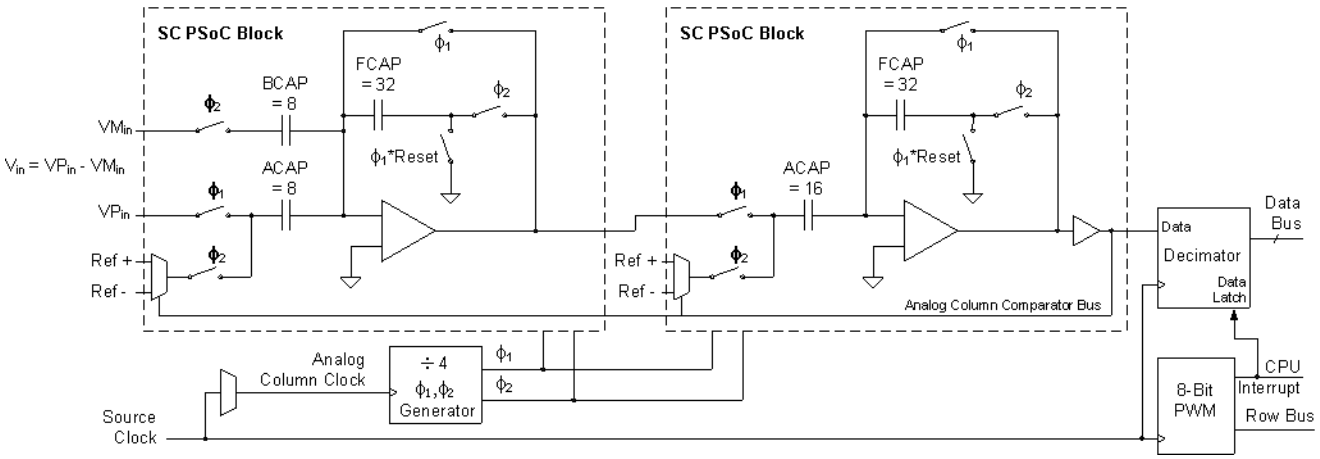


Figure 3. 2 次変調器を備えた ADCINC の回路図



ADCINC の範囲は $\pm V_{Ref}$ で設定されます。V_{Ref} は PSoC Designer の Global Resources ウィンドウで 設定できます。固定スケールの場合は、V_{Ref} には $V_{Bandgap}$, または 1.6 $V_{Bandgap}$ を設定してください。調整可能なスケールの場合は、V_{Ref} は Port 2[6] に設定してください。供給比メトリック スケールの場合、V_{Ref} は V_{DD} / 2 に設定してください。

アナログ ブロックは、リセット可能な積分器として構成されます。出力極性によって、リファレンス制御は、リファレンス電圧が入力に加算または入力から減算され、積分器に置かれるように構成されます。リファレンス制御は、積分器の出力を AGND に向かって引き戻そうとします。積分器が 2^{Bits} 回作動し、出力電圧コンパレータがこれらの回数の正の 「n」の場合、出力の残留電圧 (V_{resid}) は以下のようになります。

[+] Feedback

Equation 1

$$V_{resid} = 2^{Bits} \cdot V_{in} - n \cdot V_{ref} + (2^{Bits} - n) \cdot V_{ref}$$

Equation 2

$$V_{in} = \frac{n - 2^{Bits-1}}{2^{Bits-1}} V_{ref} + \frac{V_{resid}}{2^{Bits}}$$

この式は、この ADC の入力範囲が $\pm V_{Ref}$ であり、分解能 (LSB) が $V_{Ref} / 2^{Bits-1}$ であり、変換が終わったときの出力電圧が残留電圧として定義されることを示しています。V_{resid} は常に V_{Ref} 未満なので、LSB の半分未満の $V_{resid} / 2^{Bits}$ は無視できます。

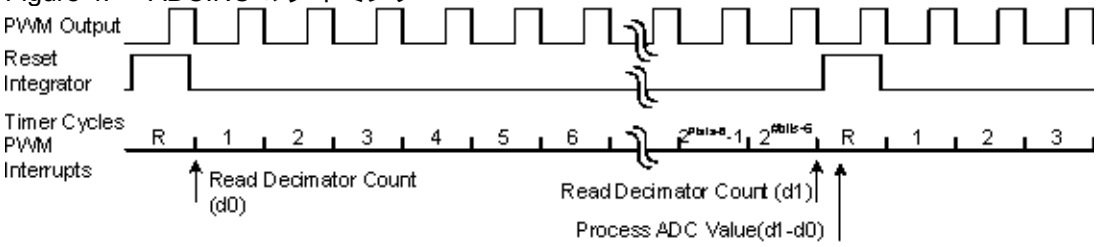
積分器をインクリメンタル ADC として機能させるには、以下のデジタル リソースを使用します。

- 正確な積分サイクル数をカウントする PWM。
- 出力コンパレータが正の期間のサイクル数を累積するためにインクリメンタルモードで構成されるデシメータ。

Note **CAUTION:** このモジュールを配置する場合、アナログ ブロックとデジタル ブロックの両方で同じソース クロックを構成する必要があります。そうしないと、正常に動作しません。

PWM は、2 5 6 カウントごとに割り込みを生成するように設定されています。これにより、入力は 1 積分サイクルと等しい 6 4 回サンプリングされます。デシメータ カウンタは、これらの積分サイクルの $2^{Bits} / 64$ を累積するように設定されています。累積値は、積分時間の最初と最後にサンプリングされます。積分をリセットし、結果を処理するために、1 サイクルを追加します。

Figure 4. ADCINC のタイミング



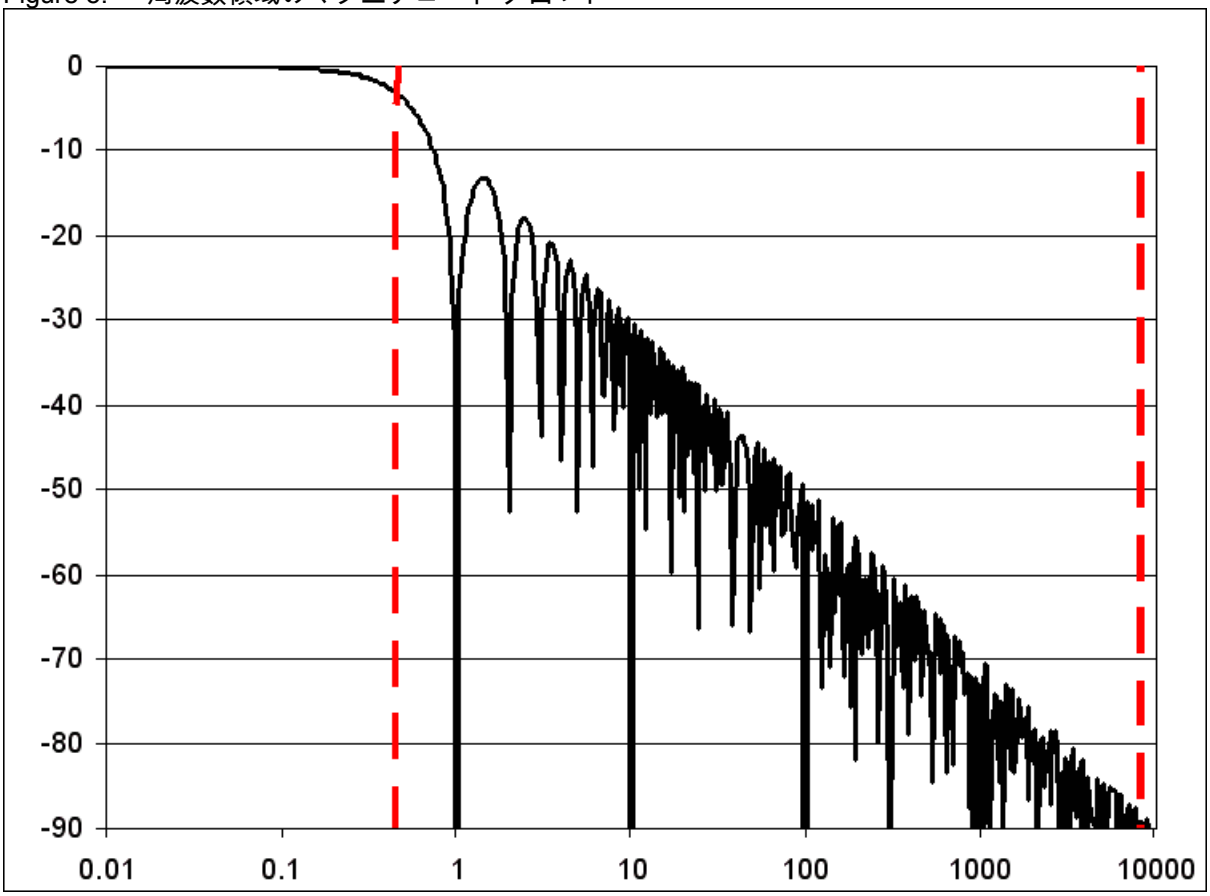
ACDINC 制御は割り込みベースで、サンプル時間は比較的長いので、サンプルの処理中にプロセッサを待機させるのは合理的ではありません。ADC ルーチンとメインプログラム間の主な通信は、ポーリング可能なデータ使用可能を示すフラグです。API は、データフラグを確認し、データを取得するために使用できます。

データ ハンドラは、ポーリング ベースとして設計されています。割り込みベースのデータ ハンドラが必要な場合は、アセンブラ ファイル ANDINCINT.asm 内にある割り込みルーチン _ADCINC_ADConversion_ISR にアプリケーション固有のデータ ハンドラ コードを追加できます。コードを挿入する点は、はっきりマークされています。

[+] Feedback

下の周波数領域のマグニチュード プロットでは、サンプル レート、14-bit F_{nom} = 1.0 となるように周波数を標準化しています。-3 dB 変化するポイントは .443 ∞ F_{nom} であり、マグニチュードは F_S の整数倍のところでゼロになります。ADCINCPWM は 14 ビットの分解能に対して設定されており、実際には、公称の出力速度より 16385 倍速くサンプリングを行うので、ナイキスト限界は 8,192 高く、F_{nom} の 13 オクターブ上になり、これによりアンチエイリアス フィルタの要件が大幅に低減されます。ナイキスト限界は 13 ビットの分解能では 12 オクターブ、12 ビットの分解能では 11 オクターブなどとなります。

Figure 5. 周波数領域のマグニチュード プロット



DC 電気的特性と AC 電気的特性

以下の値は、拡張された性能と最初の特性データを表わしています。特に指定されている場合を除き、T_A = 25°C、V_{dd} = 5.0V、Power HIGH、オペアンプ バイアス LOW 、基準電圧は P2[4] で外部 2.5V と接続され、範囲は 1.25 を P2[6] 外部 V_{ref} に接続した状態を基準とします。

Table 1. 5.0V 2nd 次変調器の DC 電気的特性と AC 電気的特性

パラメータ	標準値	制限	単位	条件および注記
入力				
入力電圧範囲	---	Vss ~ Vdd	V	Ref Mux = Vdd / 2 ± Vdd / 2
入力容量 [1]	3	---	pF	
入力インピーダンス	1 / (C*clk)	---	Ω	
分解能	---	8	ビット	
サンプリング速度	---	.125 ~ 31.25	ksps	
SNR	46	---	dB	
DC 精度				
DNL	0.1	---	LSB	アナログ カラム クロック 2 MHz
INL	0.5	---	LSB	
オフセット誤差	10	---	mV	
ゲイン誤差				
リファレンス ゲイン誤差を含む	3.0	--	% FSR	
リファレンス ゲイン誤差を除く [2]	0.1	--	% FSR	
動作電流				
Low Power 時	180	---	uA	
Med Power 時	840	---	uA	
High Power 時	3450	---	uA	
データ クロック	---	0.032 ~ 8.0	MHz	デジタル ブロックへの入力およびアナログ コラムのクロック

Table 2. 5.0V 1st 次変調器の DC 電気的特性と AC 5.0V 電気的特性

パラメータ	標準値	制限	単位	条件および注記
入力				
入力電圧範囲	---	Vss ~ Vdd		Ref Mux = Vdd / 2 ± Vdd / 2
入力容量 [1]	3	---	pF	
入力インピーダンス	1 / (C*clk)	---	Ω	
分解能	---	8	ビット	
サンプリング速度	---	.125 ~ 31.25	ksps	
SNR	44	---	dB	
DC 精度				
DNL	0.6	---	LSB	アナログ カラム クロック 2MHz
INL	0.7	---	LSB	
オフセット誤差	5	---	mV	
ゲイン誤差				
リファレンス ゲイン誤差を含む	3.0	--	% FSR	
リファレンス ゲイン誤差を除く [2]	0.1	--	% FSR	
動作電流				
Low Power 時	50	---	uA	
Med Power 時	500	---	uA	
High Power 時	1900	---	uA	
データ クロック	---	0.032 ~ 8.0	MHz	デジタル ブロックへの入力およびアナログ コラムのクロック

以下の値は、拡張された性能と最初の特性データを表わしています。特に指定されている場合を除き、T_A = 25°C、V_{dd} = 5.0V、Power HIGH、オペアンプ バイアス LOW、基準電圧は P2[4] で外部 2.5V と接続され、範囲は 1.25 を P2[6] 外部 V_{ref} に接続した状態を基準とします。

Table 3. 3.3V 2nd 次変調器の DC 電気的特性と AC 電気的特性

パラメータ	標準値	制限	単位	条件および注記
入力				
入力電圧範囲	---	Vss ~ Vdd	V	Ref Mux = Vdd / 2 ± Vdd / 2
入力容量 [1]	3	---	pF	
入力インピーダンス	1 / (C*clk)	---	Ω	
分解能	---	8	ビット	
サンプリング速度	---	.125 ~ 31.25	ksps	
SNR	46	---	dB	
DC 精度				
DNL	0.1	---	LSB	コラムのクロック 2 MHz
INL	0.5	---	LSB	
オフセット誤差	10	---	mV	
ゲイン誤差				
リファレンス ゲイン誤差を含む	3.0	--	% FSR	
リファレンス ゲイン誤差を除く [2]	0.3	--	% FSR	
動作電流				
Low Power	130	---	uA	
Med Power	840	---	uA	
High Power	3370	---	uA	
データクロック	---	0.032 ~ 8.0	MHz	デジタル ブロックへの入力およびアナログ コラムのクロック

[+] Feedback

Table 4. 3.3V 1st 次変調器の DC 電気的特性と AC 電気的特性

パラメータ	標準値	制限	単位	条件および注記
入力				
入力電圧範囲	---	Vss ~ Vdd	V	Ref Mux = Vdd / 2 ± Vdd / 2
入力容量 [1]	3	---	pF	
入力インピーダンス	1 / (C*clk)	---	Ω	
分解能	---	8	ビット	
サンプリング速度	---	.125 ~ 31.25	ksps	
SNR	44	---	dB	
DC 精度				
DNL	0.6	---	LSB	コラム クロック 2 MHz
INL	0.8	---	LSB	
オフセット誤差	6	---	mV	
ゲイン誤差				
リファレンス ゲイン誤差を含む	3.0	--	% FSR	
リファレンス ゲイン誤差を除く [2]	0.3	--	% FSR	
動作電流				
Low Power	50	---	uA	
Med Power	500	---	uA	
High Power	1900	---	uA	
データ クロック	---	0.032 ~ 8.0	MHz	デジタル ブロックへの入力およびアナログ コラムのクロック

電気的特性に関する注意事項

1. 入出力ピンを含みます。
2. リファレンス ゲイン誤差は、テスト マルチ プレクサによって経路指定され、ピンに戻される VRefHigh および VRefLow と外部リファレンスを比較して測定します。

配置

1 次変調器を設計するには、2 つの PSoC ブロック、1 つのデジタルブロック、および 1 つのアナログ ブロックが必要です。アナログ ブロックは、どのスイッチド キャパシタ PSoC ブロックにも配置できます。考慮する必要があるのは、入力とクロックを使用できるかどうかだけです。ただしデジタル ブロックは、ハードウェア デシメータに入力できなければなりません。CY8C27xxx ファミリで認証されているデジタルブロックは、DBB01、DBB02、DBB11、および DCB12 です。その他のデバイス ファミリにおいては、どのデジタル ブロックでも使用できます。デジタル ブロックにもアナログ ブロックにもに、同じクロックを選択する必要があります。クロックが同じでなければこのユーザ モジュールは正確に機能しません。

2 次変調器の設計には、2 つ目のスイッチド キャパシタ PSoC ブロックが必要です。両方のアナログブロックは、コラム コンパレータバスを共有できるように、同じコラムに配置しなければなりません。デジタル ブロックには、1 次および 2 次変調器の両方に対するのと同じ制限がかかります。

アナログおよびデジタル ブロックでは多数の配置が可能ですが、ADCINC は、PSoC デバイス唯一のハードウェア デシメータも使用します。あるコンフィグレーションに指定できる ADCINC インスタンスは 1 つだけです。ダイナミック リコンフィグレーションを使用すると、ブロックが重ならない限り、複数のコンフィグレーションをロードできます。どちらのインスタンスも機能しているように見えますが、一番最近ロードされたインスタンスの出力のみが有効です。

パラメータおよびリソース

ADCINC PWM インスタンスを配置したら、正しい動作を実現するために、パラメータ、分解能、データ フォーマット、データ クロック、PosInput 信号マルチ プレクサの選択、NegInput マルチ プレクサの選択、NegInput ゲイン、クロック位相、パルス幅、および PWM 出力を必ず設定してください。

データ フォーマット

この選択により、返される結果のデータ フォーマットが決まります。符号付きの結果は、選択した分解能の 2 の補数です。

分解能

この選択により、返される結果のデータ フォーマットが決まります。分解能の有効な選択肢は、6 ~ 14 ビットです。

データ クロック

データ クロックによりサンプリング速度が決まります。このクロックは、1 次変調器設計の PSoC ブロックと、2 次設計の 3 つの PSoC ブロックの両方に反映されます。

Note **IMPORTANT:** デジタル ブロックとアナログ コラム クロックの両方に、必ず同じクロックを選択しなければなりません。クロックが同じでなければ、このユーザ モジュールは正確に機能しません。

CPU が 24MH z で動作している場合は、データ クロックを 250kHz 未満で設定しないでください。CPU が 24MH z で動作している場合でなければ、125kHz まで値を下げられます。データ クロックは CPU クロックを超えてはなりません。常に CPU クロックと同じか、それより低く設定してください。PWM はデータクロックの 256 カウントごとに割り込みを発生させるように設定されています。カウンタは、これらのサイクルの 2^{Bits-6} 間の信号を積分します。積分器をリセットし、

[+] Feedback

データを処理するために、追加サイクルが必要です。サンプリング速度は以下の式 3 で定義されます。

Equation 3

$$SampleRate = \frac{DataClock}{256 \cdot (2^{Bits-6} + 1)}$$

データ クロックの使用可能な最大値は、8MH z です。これはスイッチド キャパシタ ブロックの制限によります。各種ビット レートの最大サンプリング速度 は、8MHz のクロック レートを使用して計算できます。サンプリング速度は下の表に示されています。

分解能	最大サンプリング速度
6-bit	15.6 ksps
7-bit	10.4 ksps
8-bit	6.25 ksps
9-bit	3.4 ksps
10-bit	1.8 ksps
11-bit	976 sps
12-bit	480 sps
13-bit	242 sps
14-bit	121 sps

サンプル ウィンドウは、ADC が遮断する通常モードの周波数を決定します。これは以下のように定義されます。

Equation 4

$$SampleWindow = \frac{2^{Bits}}{DataClock}$$

より高い周波数とその高調波を遮断するには、遮断周波数の偶数倍のサンプル ウィンドウを選択します。

クロック位相

クロック位相の選択は、あるアナログ PSoC ブロックの出力を別のアナログ PSoC ブロックの入力と同期させるために使用します。スイッチド キャパシタ アナログ PSoC ブロックは、2 相クロック (phi1、phi2) を使用して、信号を取得および転送します。通常、ADCINC への入力は、phi1 でサンプリングされます。ユーザ モジュールの多くは、phi1 中に出力を自動的にゼロに設定し、phi2 中しか有効な出力を供給しないため、問題が発生します。このようなモジュールの出力が ADCINC の入力に入力されると、有効な信号の代わりに自動的にゼロに設定された出力が取得されます。クロック位相を選択することで、入力信号を Phi2 中に取得するように、位相を交換できます。

PosInput

ADC のメイン入力。PSoC Designer では、ユーザが有効な入力を選択できます。

NegInput

ADC の差動入力を作成できるようにします。この入力は、NegInput ゲイン パラメータを使用して重み付けできます。差動入力に対比するような単一入力が必要な場合は、NegInput ゲイン パラメータの値に 「Disconnected (切断)」を設定します。PSoC Designer では、ユーザが NegInput パラメータに対して適正な入力を選択できます。

NegInput ゲイン

負の入力ゲインを選択します。シングル エンドの入力が必要な場合は、この値に 「Disconnected (切断)」を設定します。

パルス幅

PWM パルス幅は、1 ~ 255 カウントの値に設定できます。値が設定されていない場合は、GetSamples 関数が呼び出されると、ユーザ モジュールは PWM パルス幅を自動的に 1 に設定します。

PWM 出力

出力パラメータは、無効にすることや 4 つのグローバル出力信号のひとつに接続することができます。

割り込み生成制御

以下のパラメータにアクセスできるのは、PSoC Designer の Enable Interrupt Generation Control （割り込み生成制御をイネーブルにする) チェック ボックスがチェックされている場合だけです。これは **Project（プロジェクト）メニューの > Setting...（設定）から > Device Editor（デバイス エディタ）を選択すると使用できます。**

IntDispatchMode

IntDispatchMode パラメータを使用して、同一ブロック内の異なるオーバレイ内に存在する複数のユーザ モジュールで共用している割り込みについて、割り込みをどのように処理するかを指定します。 「ActiveStatus」を選択すると、ファームウェアは共用されている割り込み要求を処理する前に、どのオーバレイがアクティブかをテストします。このテストは、共用割り込みが要求されるたびに行われます。このためにレイテンシが付加され、共用割り込み要求を処理する非決定性のプロセスも生じますが、RAM は不要です。 「OffsetPreCalc」を選択すると、ファームウェアはオーバレイが最初にロードされたときだけ共用割り込みの要求のソースを計算します。この計算によって割り込みレイテンシは減少し、共用割り込み要求を処理する決定性のプロセスが生じますが、これは RAM のバイトを消費します。

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) ルーチンは設計者がより高度なレベルでモジュールを処理できるようにユーザ モジュールの一部として提供されます。このセクションでは、「include」ファイルによって提供される各機能に対するインタフェースおよび定数を示します。

Note ここでは、全てのユーザ モジュール API と同じように、API 関数を呼び出すことで A と X レジスタの値が変更されることがあります。関数を呼び出す場合、呼出し後に A と X の値が必要になるならば、必ず呼び出し前に A と X の値を保存してください。PSoc Designer のバージョン 1.0 以降、効率性の観点から、この「registers are volatile (レジスタの揮発性)」ポリシーが採用されています。C コンパイラは自動的にこの条件を処理します。アセンブラ言語のプログラマは、コードがこのポリシーを遵守していることも確認しなければなりません。ユーザ モジュール API 関数の中には、A と X を変更しないものもありますが、今後も変更されないという保証はありません。

ユーザ モジュールを配置するたびに、インスタンス名が割り当てられます。デフォルトでは、PSoC Designer プロジェクトで、このユーザ モジュールの最初のインスタンスに ADCINC_1 を割り当てます。これは識別子の構文ルールに従った一意の値に変更できます。割り当てたインスタンス名が、全てのグローバル関数名、変数、および定数記号の接頭語になります。以下の説明では、簡潔にするために、インスタンス名を ADCINC に短縮しています。

ADCINC_Start

説明

このユーザ モジュールで必要な全ての初期化を実行し、スイッチド キャパシタ PSoC ブロックの出力レベルを設定します。PWM が起動します。

C プロトタイプ

```
void  ADCINC_Start (BYTE bPowerSetting)
```

アセンブラ

```
mov   A, bPowerSetting
lcall  ADCINC_Start
```

パラメータ

bPowerSetting: 出力レベルを指定する 1 バイト。リセットとコンフィグレーションの後、ADCINC に割り当てられたアナログ PSoC ブロックの電力が遮断されます。C およびアセンブラで指定されている記号名と関連する値を以下の表に示します。

記号名	値
ADCINC_OFF	0
ADCINC_LOWPOWER	1
ADCINC_MEDPOWER	2
ADCINC_HIGHPOWER	3

戻り値

なし

副作用

A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル (CY8C29xxx) の全ての RAM ページ ポインタ レジスタにも同じことが言えます。必要に応じて、fastcall16 関数の呼び出しでこれらの値を保存してください。

ADCINC_SetPower

説明

スイッチド キャパシタ PSoC ブロックの出力レベルを設定します。

C プロトタイプ

```
void  ADCINC_SetPower (BYTE bPowerSetting)
```

アセンブラ

```
mov   A, bPowerSetting
lcall  ADCINC_SetPower
```

パラメータ

bPowerSetting: Start エントリ ポイントで使用した bPowerSetting パラメータと同じ。

戻り値

なし

副作用

A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル (CY8C29xxx) の全ての RAM ページ ポインタ レジスタにも同じことが言えます。必要に応じて、fastcall16 関数の呼び出しでこれらの値を保存してください。

ADCINC_Stop

説明

スイッチド キャパシタ PSoC ブロックの出力レベルをオフに設定します。

C プロトタイプ

```
void  ADCINC_Stop (void)
```

アセンブラ

```
lcall  ADCINC_Stop
```

パラメータ

なし

戻り値

なし

副作用

A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル (CY8C29xxx) の全ての RAM ページ ポインタ レジスタにも同じことが言えます。必要に応じて、fastcall16 関数の呼び出しでこれらの値を保存してください。

ADCINC_GetSamples

説明

指定したサンプル数 ADC を実行します。

C プロトタイプ

```
void  ADCINC_GetSamples (BYTE bNumSamples)
```

アセンブラ

```
mov   A, bNumSamples
lcall  ADCINC_GetSamples
```

パラメータ

bNumSamples: 変換するサンプル数を設定する 8-bit の値。値が 「0」の場合、ADC は継続的に実行されます。

戻り値

なし

副作用

A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル (CY8C29xxx) の全ての RAM ページ ポインタ レジスタにも同じことが言えます。必要に応じて、fastcall16 関数の呼び出しでこれらの値を保存してください。現時点では , CUR_PP ページ ポインタ レジスタのみが変更されています。

ADCINC_StopADC

説明

ADC を即座に停止します。PWM は実行を続けます。

C プロトタイプ

```
void  ADCINC_StopADC (void)
```

アセンブラ

```
lcall  ADCINC_StopADC
```

パラメータ

なし

戻り値

なし

副作用

A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル (CY8C29xxx) の全ての RAM ページ ポインタ レジスタにも同じことが言えます。必要に応じて、fastcall16 関数の呼び出しでこれらの値を保存してください。

ADCINC_fIsDataAvailable

説明

サンプリングしたデータが使用できるかどうかチェックします。

C プロトタイプ

```
BYTE  ADCINC_fIsDataAvailable(void)
```

アセンブラ

```
lcall  ADCINC_fIsDataAvailable
```

パラメータ

なし

戻り値

データが変換され、読み取れる状態の場合は、非ゼロ値を返します。

副作用

A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル (CY8C29xxx) の全ての RAM ページ ポインタ レジスタにも同じことが言えます。必要に応じて、fastcall16 関数の呼び出しでこれらの値を保存してください。現時点では , CUR_PP ページポインタレジスタのみが変更されています。

ADCINC_iGetData

説明

変換したデータを符号付き整数として返します。データサンプルがすぐ使用できるかどうか確認するためには、ADCINC_bfIsDataAvailable() を呼び出してください。

C プロトタイプ

```
INT   ADCINC_iGetData(void)
```

アセンブラ

```
lcall  ADCINC_iGetData           ; Data will be in A and X upon return
```

パラメータ

なし

戻り値

変換したデータ サンプルを 16-bit の 2 の補数の形で返します。

副作用

A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル (CY8C29xxx) の全ての RAM ページ ポインタ レジスタにも同じことが言えます。必要に応じて、fastcall16 関数の呼び出しでこれらの値を保存してください。現時点では , CUR_PP ページ ポインタ レジスタのみが変更されています。

ADCINC_wGetData

説明

変化したデータを符号なし整数として返します。データ サンプルがすぐ使用できるかどうか確認するためには、ADCINC_bfIsDataAvailable() を呼び出してください。

C プロトタイプ

```
WORD  ADCINC_wGetData(void)
```

アセンブラ

```
lcall  ADCINC_wGetData           ; Data will be in A and X upon return
```

パラメータ

なし

戻り値

変換した 16-bit の符号なしデータ サンプルを返します。

副作用

A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル (CY8C29xxx) の全ての RAM ページ ポインタ レジスタにも同じことが言えます。必要に応じて、fastcall16 関数の呼び出しでこれらの値を保存してください。現時点では , CUR_PP ページ ポインタ レジスタのみが変更されています。

ADCINC_cGetData

説明

変換したデータを符号付き文字として返します。データサンプルがすぐ使用できるかどうか確認するためには、ADCINC_bfIsDataAvailable() を呼び出してください。

C プロトタイプ

```
CHAR  ADCINC_cGetData(void)
```

アセンブラ

```
lcall  ADCINC_cGetData           ; Data will be in A upon return
```

パラメータ

なし

戻り値

変換したデータ サンプルを 8-bit の 2 の補数の形で返します。

副作用

A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル (CY8C29xxx) の全ての RAM ページ ポインタ レジスタにも同じことが言えます。必要に応じて、fastcall16 関数の呼び出しでこれらの値を保存してください。現時点では , CUR_PP ページ ポインタ レジスタのみが変更されています。

ADCINC_bGetData

説明

変換したデータを符号なし文字として返します。データサンプルがすぐ使用できるかどうか確認するためには ADCINC_bfIsDataAvailable() を呼び出してください。

C プロトタイプ

```
BYTE  ADCINC_bGetData(void)
```

アセンブラ

```
lcall  ADCINC_bGetData           ; Data will be in A upon return
```

パラメータ

なし

戻り値

変換した 8-bit の符号なしデータサンプルを返します。

副作用

A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル (CY8C29xxx) の全ての RAM ページ ポインタ レジスタにも同じことが言えます。必要に応じて、fastcall16 関数の呼び出しでこれらの値を保存してください。現時点では , CUR_PP ページ ポインタ レジスタのみが変更されています。

ADCINC_iClearFlagGetData

説明

データ準備完了フラグを解除し、変換したデータを符号付き整数として取得します。そして、データ フラグがまだリセットされているかどうか確認します。もし、データフラグがリセットされていなかったら、それはデータが更新された場合です。これにより ADC 割り込みルーチンが、収集時に応答を更新していないことが確認されます。

C プロトタイプ

```
INT   ADCINC_iClearFlagGetData(void)
```

アセンブラ

```
lcall  ADCINC_iClearFlagGetData   ; Data will be in A and X upon return
```

パラメータ

なし

戻り値

変換したデータ サンプルを 16-bit の 2 の補数の形で返します。

副作用

グローバル変数 ADCINC_bfStatus をゼロに設定します。A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル (CY8C29xxx) の全ての RAM ページ ポインタ レジスタにも同じことが言えます。必要に応じて、fastcall16 関数の呼び出しでこれらの値を保存してください。現時点では , CUR_PP ページ ポインタ レジスタのみが変更されています。

ADCINC_wClearFlagGetData

説明

データ準備完了フラグを解除し、変換したデータを符号なし整数として取得します。そして、データフラグがまだリセットされているかどうか確認します。もし、データフラグがリセットされていなかったら、それはデータが更新された場合です。これにより ADC 割り込みルーチンが、収集時に応答を更新しないことが確認されます。

C プロトタイプ

```
WORD  ADCINC_wClearFlagGetData(void)
```

アセンブラ

```
lcall  ADCINC_wClearFlagGetData   ; Data will be in A and X upon return
```

パラメータ

なし

戻り値

変換した 16-bit の符号なしデータ サンプルを返します。

副作用

グローバル変数 ADCINC_bfStatus をゼロに設定します。A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル (CY8C29xxx) の全ての RAM ページ ポインタ レジスタにも同じことが言えます。必要に応じて、fastcall16 関数の呼び出しでこれらの値を保存してください。現時点では , CUR_PP ページ ポインタ レジスタのみが変更されています。

ADCINC_cClearFlagGetData

説明

データ準備完了フラグを解除し、変換したデータを符号付き文字として取得します。データ フラグがまだリセットされているかどうか確認します。もし、データフラグがリセットされていなかったら、それはデータが更新された場合です。これにより ADC 割り込みルーチンが、収集時に応答を更新していないことが確認されます。

C プロトタイプ

```
CHAR  ADCINC_cClearFlagGetData(void)
```

アセンブラ

```
lcall  ADCINC_cClearFlagGetData   ; Data will be in A upon return
```

パラメータ

なし

戻り値

変換したデータ サンプルを 8-bit の 2 の補数の形で返します。

副作用

グローバル変数 ADCINC_bfStatus をゼロに設定します。A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル (CY8C29xxx) の全ての RAM ページ ポインタ レジスタにも同じことが言えます。必要に応じて、fastcall16 関数の呼び出しでこれらの値を保存してください。現時点では , CUR_PP ページ ポインタ レジスタのみが変更されています。

ADCINC_bClearFlagGetData

説明

データ準備完了フラグを解除し、変換したデータを符号なし文字として取得します。データ フラグがまだリセットされているかどうか確認します。もし、データフラグがリセットされていなかったら、それはデータが更新された場合です。これにより ADC 割り込みルーチンが、収集時に応答を更新していないことが確認されます。

C プロトタイプ

```
BYTE  ADCINC_bClearFlagGetData(void)
```

アセンブラ

```
lcall  ADCINC_bClearFlagGetData   ; Data will be in A upon return
```

パラメータ

なし

戻り値

変換した 8-bit の符号なしデータ サンプルを返します。

副作用

グローバル変数 ADCINC_bfStatus をゼロに設定します。A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル (CY8C29xxx) の全ての RAM ページ ポインタ レジスタにも同じことが言えます。必要に応じて、fastcall16 関数の呼び出しでこれらの値を保存してください。現時点では , CUR_PP ページ ポインタ レジスタのみが変更されています。

ADCINC_fClearFlag

説明

変換結果が使用可能かどうかを返し、フラグをリセットします。

C プロトタイプ

```
BYTE  ADCINC_fClearFlag(void)
```

アセンブラ

```
lcall  ADCINC_fClearFlag
```

パラメータ

なし

戻り値

状態レジスタの値を返します。

副作用

グローバル変数 ADCINC_bfStatus をゼロに設定します。A および X レジスタは、今回、または今後、この関数を実装することにより変更される可能性があります。大容量メモリ モデル (CY8C29xxx) の全ての RAM ページ ポインタ レジスタにも同じことが言えます。必要に応じて、fastcall16 関数の呼び出しでこれらの値を保存してください。現時点では , CUR_PP ページ ポインタ レジスタのみが変更されています。

ADCINC_WritePulseWidth

説明

PWM のパルス幅を変更します。

C プロトタイプ

```
void  ADCINC_WritePulseWidth(BYTE bPulseWidth)
```

アセンブラ

```
mov   A, bPulseWidth
lcall  ADCINC_WritePulseWidth
```

パラメータ

bPulseWidth: PWM のパルス幅を設定します。この値がゼロになっていなければ ADC は機能を停止します。

戻り値

なし

副作用

A および X レジスタは、今回、または今後、この関数を実装することによって変更される可能性があります。大容量メモリ モデル (CY8C29xxx) の全ての RAM ページ ポインタ レジスタにも同じことが言えます。必要に応じて、fastcall16 関数の呼び出しでこれらの値を保存してください。

ファームウェア ソースコードの例

以下のサンプル コードでは、フラグ レジスタをポーリングし、データを入出力ピンの 1 本にシフト アウトするルーチンにデータを送信します。

```
;;; Sample Code for the ADCINC
;;; Continuously Sample and Output Data to a pin.
;;;
;;; The user must provide the function to shift the data out.
;;;
include "m8c.inc"                ; part specific constants and macros
include "PSoCAPI.inc"            ; PSoC API definitions for all User Modules
export _main
_main:
   M8C_EnableGInt                  ; enable global interrupts
   mov  a,ADCINC_HIGHPOWER         ; set Power
   call ADCINC_Start
   mov  a,00h                      ; set ADC to continuous sampling
   call ADCINC_GetSamples
loop1:
wait:
   call ADCINC_fIsDataAvailable    ; poll flag
   jz wait
   call ADCINC_iClearFlagGetData   ; reset flag and retrieve data
   ;; call shift_it_out            ; (user provided) send data to output pin
   jmp loop1
```

C で記述した同じプロジェクトを以下に示します。

```
//-------------------------------------------------------------------------
//  Sample C Code for the ADCINC
//  Continuously Sample input voltage
//
//-------------------------------------------------------------------------
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules
INT  iData;
void main(void)
{
   M8C_EnableGInt;                        // Enable Global Interrupts
   ADCINC_Start(ADCINC_HIGHPOWER);        // Apply power to the SC Block
   ADCINC_GetSamples(0);                  // Have ADC run continuously
   for(;;){
      while(ADCINC_fIsDataAvailable() == 0);   // Loop until value ready
      ADCINC_iClearFlagGetData();              // Clear ADC flag and get data
      // Add user code here to use or display result
   }
 }
```

コンフィグレーション レジスタ

Table 5. 「ADC」アナログ スイッチド キャパシタ PSoC ブロックによって使用されるレジスタ

レジスタ	7	6	5	4	3	2	1	0
CR0	1	ClockPhase	0	ACap				
CR1	PosInputSource			NegInputGain				
CR2	0	1	AZ	0	0	0	0	0
CR3	1	1	1	FSW0	NegInputSource		0	0

ADC は、 1 つまたは 2 つのスイッチド キャパシタ PSoC ブロックを使用します。ブロックはアナログ変調器を作成するために構成されています。変調器を構築するために、ブロックは、入力値をデジタル パルス ストリームに変換するリファレンス フィードバックを持つ積分器として構成します。入力マルチプレクサは、デジタル化する信号を決定します。

ClockPhase は、スイッチのクロック位相だけでなく、スイッチド キャパシタ ブロック内のコンパレータのクロック位相を制御します。

ACap はキャパシタ ACap 用の 32 の可能なキャパシタ サイズのバイナリ エンコーディングを含みます。

InputSource フィールドは、コンバータによってデジタル化される入力信号を選択します。このパラメータはデバイス エディタで設定します。

AZ と FSW0 は PWM 割り込みハンドラ および各種 API によって、積分器をリセットするために使用されます。

Table 6. PWM デジタル PSoC ブロックが使用するレジスタ

レジスタ	7	6	5	4	3	2	1	0
関数	0	0	1	1	0	0	0	1
入力	0	0	0	1	クロック			
出力	0	0	0	0	0	0	0	0
DR0	タイマ停止カウント値（API がアクセスすることはありません）							
DR1	1	1	1	1	1	1	1	1
DR2	パルス幅							
CR0	0	0	1	0	0	0	0	イネーブル

PWM は 256 カウントの間、タイマ機能を持つように構成されたデジタル PSoC ブロックです。割り込みに際して、デシメータが読み取られ、ADC 値が計算されます。

クロックは 16 のクロック ソースの 1 つから入力クロックを選択します。このパラメータはデバイス エディタで設定します。

Note 選択したソースは、ADC ブロックが存在するコラムのアナログ クロックを制御するためにも使用する必要があります。

イネーブルが設定されると、PWM が有効化されます。これは ADCINC API によって変更および制御されます。

Table 7. デシメーション制御レジスタ

ビット	7	6	5	4	3	2	1	0
DEC_CR0	0	0	0	0	ICLKS0	0	DCol	DCLKS0
DEC_CR1	1	1	0	0	ICLKS1			DCLKS1
DEC_DH	デシメータの上位バイト出力							
DEC_DL	デシメータの下位バイト出力							

バージョン ヒストリー

バージョン	著者	説明
1.1	DHA	以下の項目を確認するために DRC を付加しました。 1. ソース クロックがデジタル リソースとアナログ リソースで違ったものであること 2. ADC クロックが CPU クロックより上位であること

Note PSoC Designer 5.1 は、すべてのユーザ モジュール データシートにおいてバージョン ヒストリーを導入しています。このセクションでは、ユーザ モジュールの過去のバージョンと現在のバージョンとの違いに関して高度な解説を掲載しています。

Copyright © 2002-2010 Cypress Semiconductor Corporation. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC Designer™ and Programmable System-on-Chip™ are trademarks and PSoC® is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.