# 7 ~ 13-Bit 可変分解能インクリメンタル ADC Data Sheet ADCINCVR V 3.2

Copyright © 2001-2010 Cypress Semiconductor Corporation. All Rights Reserved.

| リソース | PSoC® ブロック | | | API メモリ（バイト） | | ピン（外部 I/O あたり） |
|---|---|---|---|---|---|---|
| | デジタル | アナログ CT | アナログ SC | Flash | RAM | |
| CY8C29/27/24/22xxx、CY8C23x33、CY8CLED04/08/16、CY8CLED0xD、CY8CLED0xG、CY8CTST120、CY8CTMG120、CY8CTMA120、CY8C28x45、CY8CPLC20、CY8CLED16P01、CY8C28x43、CY8C28x52 | | | | | | |
| | 3 | 0 | 1 | 325 | 5 | 1 |
| CY8C25/26xxx | 3 | 0 | 1 | 320 | 5 | 1 |

他の変換器については、AN2239, ADC Selection Guide を参照してください。

このユーザ モジュールを使用した機能プロジェクトの実現例については、以下を参照してください。www.cypress.com/psocexampleprojects。

## 特徴と概要

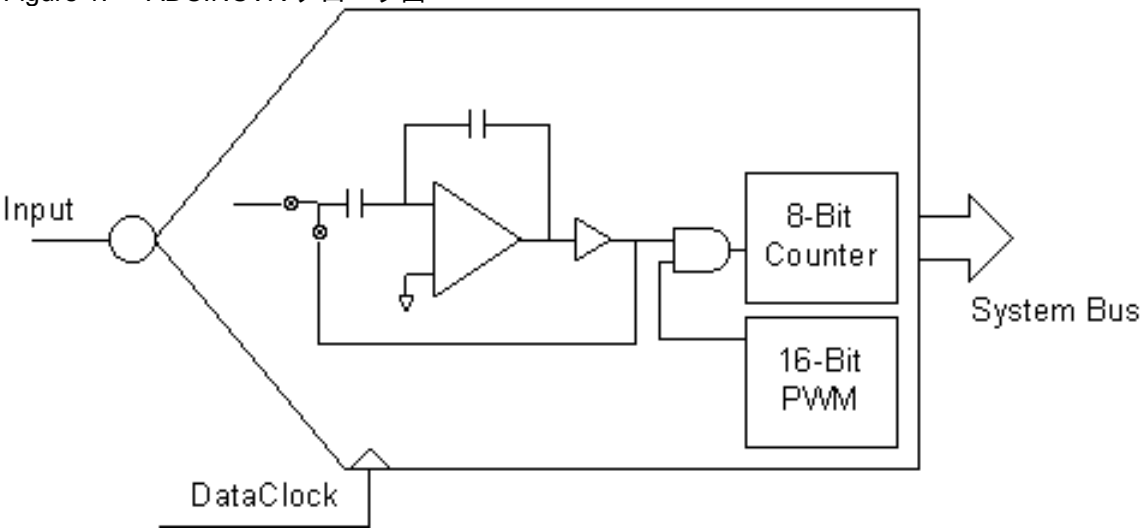
- 分解能 7 ~ 13-bit、2 の補数データ フォーマットに対応
- サンプリング速度 4 ~ 10,000 sps
- 入力範囲 Vss ~ Vdd
- 積分型変換器による優れたノーマルモード ノイズの低減
- 内部または外部クロック

ADCINCVR は、7 ~ 13 ビットの間で分解能を設定可能な積分型 ADC です。積分時間を最適化し、不要な高周波成分を取り除くように設定できます。レール ツー レール などの入力電圧範囲は、適切なリファレンス電圧とアナログ グラウンドを構成することで測定できます。出力のフォーマットは、AGND を中心とする $-V_{ref}$ と $+V_{ref}$ 間の入力電圧に基づいた 2 の補数です。

分解能、DataClock および CalcTime パラメータの選択内容によって、4 ~ 10,000 sps のサンプリング速度を実現できます。

プログラミング インタフェースにより、取得する連続サンプル数を指定したり、連続サンプリングを選択したりできます。CPU の負荷は、入力レベルによって異なります。たとえば、$V_{in} = +V_{ref}$ の場合、CPU サイクル数は 5076 です（最大 13 ビット）。$V_{in}$ = AGND の場合、CPU サイクル数は 2708 です（平均 13 ビット）。また、$V_{in} = -V_{ref}$ の場合、CPU サイクル数は 340 です（最小 7-13 ビット）。

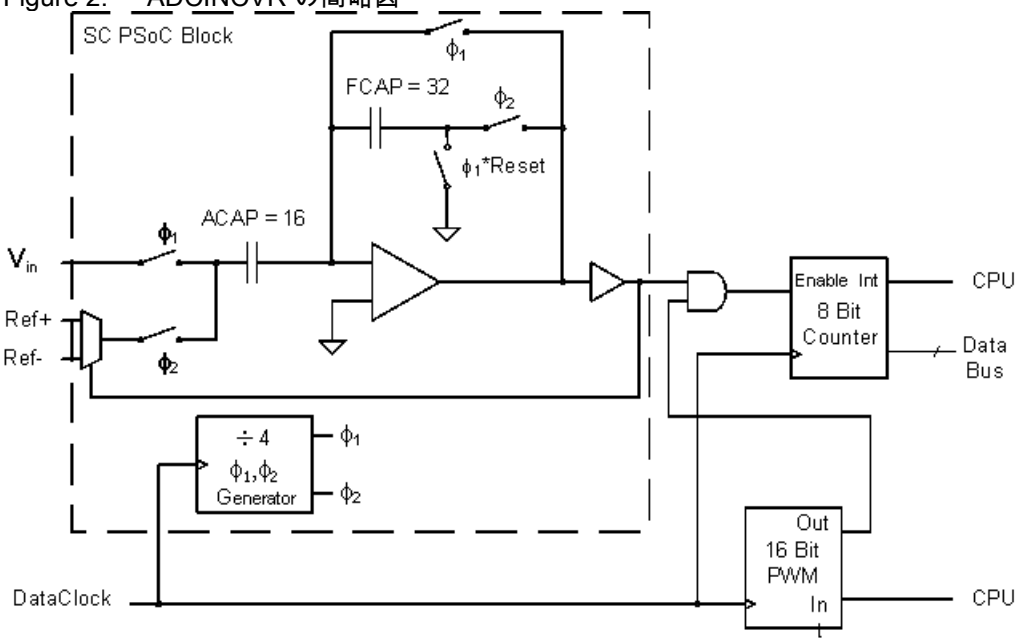
Figure 1. ADCINCVR ブロック図




## 機能説明

ADCINCVR は、図 2 に示すように、1 個のアナログ スイッチド キャパシタ PSoC ブロックと 3 つのデジタル PSoC ブロックで構成されています。

Figure 2. ADCINCVR の簡略図




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ています。出力の極性によって、リファレンス電圧が入力に加算または減算され、積分器に置かれるようにリファレンス制御が設定されます。リファレンス制御は、積分器の出力を AGND の電位に戻す動作もします。積分器が $2^{ビット}$回動作し、出力電圧コンパレータの出力が High であるときの回数が 「n」の場合、出力の残留電圧（$V_{resid}$）は、式 1 を使用して以下のように計算されます。



[+] 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 の範囲が $\pm V_{ref}$、分解能（LSB）が $V_{ref}/2^{ビット-1}$ で、計算の最後の出力の電圧が余りと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、$V_{resid}/2^{ビット}$ は LSB の半分未満となり、無視できます。この結果、以下の式 3 を導き出すことができます。

**Equation 3**

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

**例 1**

$V_{ref}$ が 1.3V で分解能が 8-bits の場合は、データの使用準備が整った時点でインクリメンタル ADC から読み取られた値を基に、入力電圧を簡単に計算できます。この計算は以下の式 4 を使用して行います。

**Equation 4**

$$V_{in} = \frac{n - 128}{128} 1.3$$

計算結果は AGND を基準とします。ADC データの値が 200 の場合、測定電圧は式 5 を使用して以下のように 0.73V と計算できます。

**Equation 5**

$$V_{in} = \frac{200 - 128}{128} 1.3 = 0.73\,V$$

計算値は理想値であり、システム雑音とチップのオフセットによって異なる場合があります。

特定の入力電圧を前提として予想されるコードを決定するために式を再整理すると、以下の式 6 のようになります。

**Equation 6**

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

### 例 2

$V_{ref}$ が 1.3V で分解能が 8-bits の場合は、入力電圧を基に、予想される ADC を簡単に計算できます。この計算は以下の式 7 を使用して行います。

**Equation 7**

$$n = \frac{128 \cdot V_{in}}{1.3} + 128$$

入力電圧が AGND より -1V 低い場合、ADC からの変換結果は 29.53 になると予想できます。この計算は式 8 に基づいて行います。

**Equation 8**

$$n = \frac{128 \cdot (-1)}{1.3} + 128 = 29.53$$

計算値は理想値であり、システム雑音とチップのオフセットによって異なる場合があります。

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。

- 出力が正のサイクル数を累積する 8-bit カウンタ。
- 積分時間を計り、8-bit カウンタへのクロックをゲート制御する 16-bit PWM。

1 つの DataClock が、8-bit カウンタ、16-bit PWM、およびアナログ SC PSoC ブロックに接続するアナログ コラムのクロックに接続されます。アナログ コラムのクロックは、実際には DataClock から生成される $\varphi_1$ と $\varphi_2$ の 2 つのクロックです。これらの 2 つの追加クロックは、厳密に DataClock の周波数の 1/4 です。つまり、PWM とカウンタは必要な速度の 4 倍で動作するため、N+2 ビット相当のデータを累積する必要があります（N は分解能のビット数と等しい）。

**Note** **CAUTION:** このモジュールを配置する場合は、3 つのブロックすべてに対して同じクロックを設定する必要があります。そうしないと、間違った動作の原因となります。

カウンタは、LSB 用の 8-bit のデジタル ブロックおよび MSB 用のソフトウェア カウンタを使用して実装します。ハードウェア カウンタがオーバフローするたびに割り込みが生成され、カウンタの上位 MSB が増分されます。これにより、4 個ではなく 3 個のデジタル ブロックだけで ADCINCVR モジュールを実装できます。

サンプリング速度は、DataClock を積分時間で割り、結果の計算にかかる時間（CalcTime）を足した値です。積分時間は、入力信号を ADCINCVR がサンプリングしている時間です。

**Equation 9**

$$SampleRate = \frac{DataClock}{2^{Bits+2} + CalcTime}$$

結果の計算にかかる時間、CalcTime は、CPU クロックに反比例して変化します。CalcTime には、結果の計算に必要な時間よりも大きな値を設定する必要があります。最小 CalcTime は 180 CPU サイクルと等しく、DataClock に換算して表現する必要があります。また、CalcTime は、サンプリング速度を最適化するために、最小値より増加させることもできます。

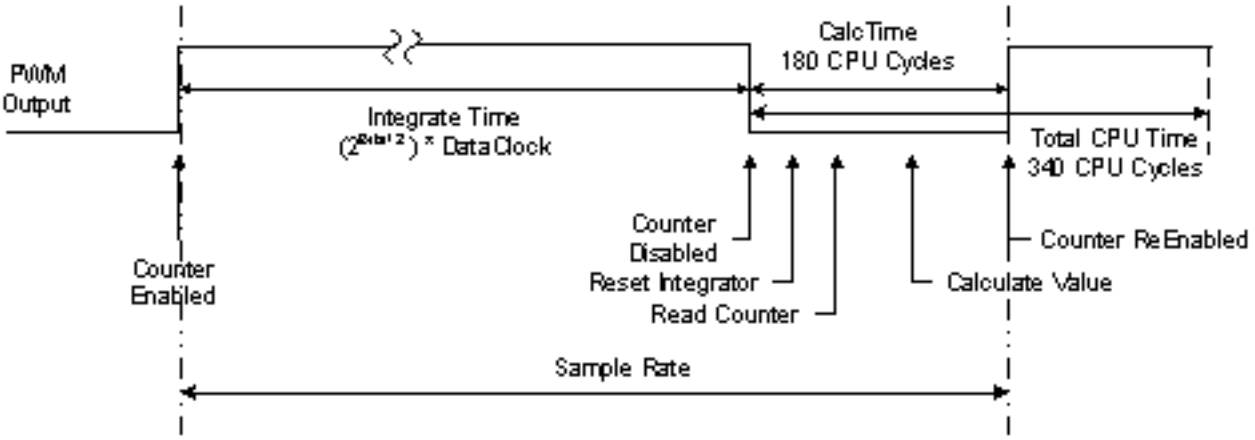
**Note** $2^{ビット+2}$ と CalcTime の合計は、$2^{16}$-1 つまり 65,535 より大きくなってはなりません。

**Equation 10**

$$CalcTime \geq \frac{180 \cdot DataClock}{CPU_Clock}$$

16-bit PWM は、$2^{ビット+2}$ と DataClock の積である HIGH 信号を出力するようにプログラムします。たとえば、分解能を 10 ビットに設定すると、PWM 出力は 4096 （$2^{10+2}$）DataClock の間、HIGH のままとなります。PWM の出力は、最小結果を計算して積分器をリセットするためにかかる時間の間、LOW になります。この LOW 時間も、DataClock と組み合わせてより正確なサンプリング速度を提供する目的で調整できます。PWM の合計時間は、積分時間と CalcTime の和です。

Figure 3. PWM 出力に関する ADCINCVR のタイマ サイクル




最初の読み取りが開始されると、PWM の構成が計算され、積分器がリセットされて、カウンタが FFh にリセットされます。最初の遅延は、常に計算時間の遅延以上となります。PWM は、最初の読み取りを行う前にのみ初期化されます。比較および時間レジスタをいったん設定した後は、分解能または計算時間を変更しない限り、再初期化は必要ありません。PWM カウントが積分値以下の場合、出力は HIGH になるため 8-bit カウンタがカウントダウンを実行できます。PWM の出力は、カウンタがゼロになるまで HIGH のままです。この時点で、8-bit カウンタへのクロックが無効化され、PWM 割り込みが生成されます。

この 8-bit ソフトウェア カウンタの初期値は、最も小さい負の数の $2^{ビット}$/64 倍に設定されます。8-bit カウンタがオーバフローするたびに、8-bit カウンタが割り込まれ、ソフトウェア カウンタが 1 増分されます。

ADC への入力が最も大きい正の数以上の場合は、DataClock が正になるたびに 8-bit カウンタが増分されます。ADC への入力が最も小さい負の入力値以下の場合は、8-bit カウンタが減分されることはなく、したがって割り込みが生成されることもありません。理想的な条件下でアナログ グラウンドに近い入力を行うことで、カウンタは半分の時間で増分できます。入力電圧レベルによって、8-bit カウンタからの割り込みの回数が 0 ~ $(2^{ビット+2})/256$ の範囲で変化することは容易に理解できます。たとえば、分解能を 10 ビットに設定すると、PWM 比較値には $2^{10+2}$ （4096）が設定されます。つまり、積分時間中、最大 4096/256、つまり 16 回プロセッサへの割り込みが発生する可能性があります。



[+] Feedback

ADCINCVR の制御が割り込みベースであることと、高い分解能の結果を得るための時間の長さにより、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、API 関数、ADCINCVR_IsDataAvailable(). を使用してポーリングできるフラグです。値が返された場合、API ADCINCVR_iGetData() を呼び出してデータを取得できます。

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 ファイル内の割り込みルーチン ADCINCVR_PWM16_ISR に独自のデータ ハンドラ コードを挿入できます *adcincvrINT.asm*。コードを挿入するのに最適な点は、はっきりマークされています。

## CPU 使用率

ADCINCVR では、結果を計算し、ハードウェア カウンタがオーバフローするたびにソフトウェア カウンタを増分するため CPU 時間が必要です。CPU のオーバヘッドは、CPU クロック、DataClock、および入力電圧の 3 つの変数に左右されます。入力電圧は ADC の CPU のオーバヘッドに影響することになります。$-V_{ref}$ に近いかそれ未満の入力電圧では、CPU のオーバヘッドはほとんど必要ありません。$+V_{ref}$ に近いかそれより大きい入力電圧では、CPU のオーバヘッドが必要となります。特定の入力に必要な CPU サイクルを計算する手順は、以下のとおりです。

**Equation 11**

$$CPU_Cycles = PWM_IRQ_CPU_Cycles + \left(\frac{2^{Bits}}{64}\left(\frac{V_{ref}+V_{in}}{2 \cdot V_{ref}}\right)Counter_IRQ_CPU_Cycles\right)$$

**Equation 12**

$$CPUcycles = 340 + \left(\frac{2^{Bits}}{64} * \left(\frac{Vref + Vin}{2 * Vref}\right) * 37\right)$$

10-bits の分解能での最大 CPU サイクルを計算するには、$V_{in}$ に $V_{ref}$ を設定します。

**Equation 13**

$$CPUcycles = 340 + \left(\frac{2^{10}}{64} * \left(\frac{Vref + Vref}{2 * Vref}\right) * 37\right) = 340 + (16 * 1 * 37) = 932$$

ADCINCVR の CPU 使用率を計算する場合は、以下の式 14 を使用できます。

**Equation 14**

$$Percent_CPU_Utilization = \frac{Sample_Rate * CPUcycles}{CPU_frequency} * 100$$

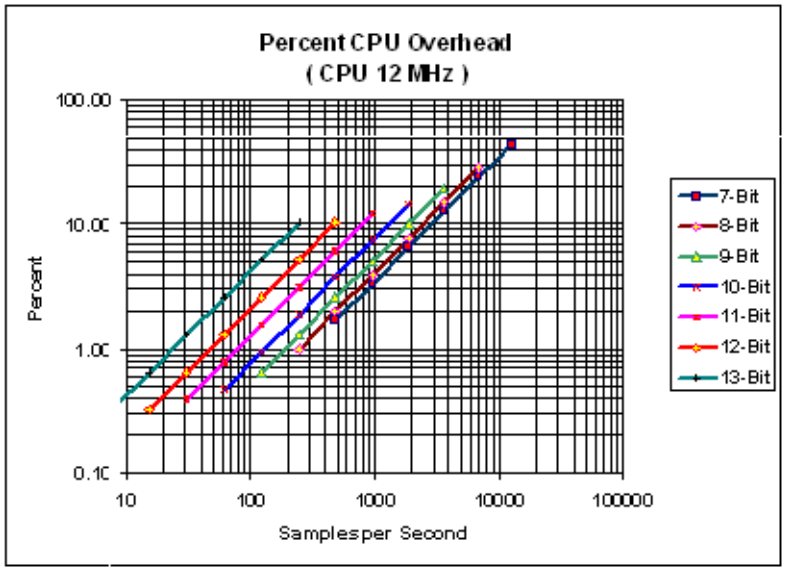
以下の式で分解能を（上の例のように）10 ビットに、サンプリング速度を 1000 sps に、CPU クロックを 12 MHz に設定した場合、使用される CPU は 8%未満です。

**Equation 15**

$$Percent_CPU_Utilization = \frac{1000 * 932}{12MHz} * 100 = 7.8\%$$

図 4 は、サポートされているサンプリング速度と分解能に対する CPU 使用率を示したものです。デフォルトの CPU 速度は 12 MHz に設定されています。

Figure 4. CPU 使用率




## 周波数遮断

ノイズ源は、適切な積分時間を選択することによって、ある程度遮断できます。ノイズ源とその高調波を遮断するには、ノイズ信号の積分サイクルと等しい積分時間を選択します。複数の信号を遮断する場合は、両方の信号の積分サイクルと等しい積分時間を選択します。

たとえば、50 Hz および 60 Hz の信号によって引き起こされるノイズを遮断するには、50 Hz および 60 Hz の両方の信号の整数を含む時間を選択します。

IntegrateTime を 100 ms にすると、50 Hz と 60 Hz の両方およびこれらの信号の高調波が遮断されます。次に、適切な IntegrateTime （積分時間）を生成するために必要な DataClock を計算します。

サンプリング速度に影響するにもかかわらず、この計算では CalcTime は使用されていないことに注意してください。IntegrateTime は、ADCINCVR が入力電圧を実際にサンプリングする時間です。サンプリング速度は IntegrateTime および結果の計算にかかる時間に基づきます。

**例**

あるアプリケーションでは、100 ms の IntegrateTime と 13 ビットの A/D 分解能が必要です。IntegrateTime を 100 ms にするには、以下のようなデータ クロックが必要です。

**Equation 16**

$$DataClock = \frac{2^{Bits+2}}{IntegrateTime} = \frac{2^{13+2}}{100ms} = 327.7kHz$$

データ クロックに換算した CalcTime は、DataClock と CPU クロックから計算します。CPU クロックが 12 MHz の場合、最小計算時間は以下のようになります。

**Equation 17**

$$CalcTime = \frac{DataClock * 180}{CPUClock} = \frac{327.7kHz * 180}{12{,}000\,kHz} = 4.9\,DataClocks$$



[+] Feedback

この CalcTime は、最も近い整数（この例では「5」）に切り上げます。次に、式 18 を使用してサンプリング速度を決定します。

**Equation 18**

$$SampleRate = \frac{DataClock}{2^{13+2} + CalcTime} = \frac{327.7kHz}{32768 + 5} = 9.99Samples/Second$$

より長いサンプリング速度が必要な場合は、CalcTime + $2^{13+2}$ が $2^{16}$-1（65535）以下になるまで CalcTime を増加できます。

## DC および AC の電気的特徴

The following values are indicative of expected performance and based on initial characterization data. 以下の表で別途指定されている場合を除き、$T_A$ = 25°C、Vdd= 5.0V、パワー設定 HIGH、オペアンプ バイアス LOW で、出力は P2[6] で 1.25 の外部 Vref を持つ P2[4] での 2.5V の外部アナログ グラウンドを基準とし、分解能は 13 ビットに設定されています。

Table 1. 5.0V ADCINCVR の DC および AC 電気的特徴、PSoC デバイスの Arch27Name; ファミリ

| パラメータ | 典型 | 制限 | 単位 | 条件および注意 |
|---|---|---|---|---|
| 入力 | | | | |
| 入力電圧範囲 | --- | Vss ~ Vdd | | Ref Mux = Vdd/2 ± Vdd/2 |
| 入力静電容量 | 3 | --- | pF | |
| 入力インピーダンス | 1/(C*clk) | --- | W | |
| 分解能 | --- | 7 ~ 13 | ビット | |
| サンプリング速度 | --- | 4 ~ 10,000 | SPS | |
| SNR | 77 | --- | dB | |
| DC 精度 | | | | |
| DNL | 0.4 | --- | LSB | コラムのクロック 2 MHz |
| INL | 1.0 | --- | LSB | |
| オフセット誤差 | 9 | --- | mV | |
| ゲイン誤差 | | | | |
| リファレンス ゲイン誤差を含む | 2.0 | -- | % FSR | |
| リファレンス ゲイン誤差を除く | 0.1 | -- | % FSR | |
| 動作電流 | | | | |



[+] Feedback

| パラメータ | 典型 | 制限 | 単位 | 条件および注意 |
|---|---|---|---|---|
| Low Power | 250 | --- | µA | |
| Med Power | 640 | --- | µA | |
| High Power | 2000 | --- | µA | |
| データ クロック | | 0.125 to 2.67 | 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 |

The following values are indicative of expected performance and based on initial characterization data. 以下の表で別途指定されている場合を除き、$T_A$ = 25°C、Vdd= 3.3V、パワー設定 HIGH、オペアンプ バイアス LOW で、出力は P2[6] で 1.25 の外部 Vref を持つ P2[4] での 1.64V の外部アナログ グラウンドを基準とし、分解能は 13 ビットに設定されています。

Table 2. 3.3V ADCINCVR の DC および AC 電気的特徴、PSoC デバイスの Arch27Name; ファミリ

| パラメータ | 標準値 | 制限 | 単位 | 条件および注意 |
|---|---|---|---|---|
| 入力 | | | | |
| 入力電圧範囲 | --- | Vss ~ Vdd | | Ref Mux = Vdd/2 ± Vdd/2 |
| 入力静電容量[a] | 3 | --- | pF | |
| 入力インピーダンス | 1/(C*clk) | --- | W | |
| 分解能 | --- | 7 ~ 13 | ビット | |
| サンプリング速度 | --- | 4 ~ 10,000 | SPS | |
| SNR | 77 | --- | dB | |
| DC 精度 | | | | |
| DNL | 0.4 | --- | LSB | コラムのクロック 2 MHz |
| INL | 1.0 | --- | LSB | |
| オフセット誤差 | 4 | --- | mV | |
| ゲイン誤差 | | | | |
| リファレンス ゲイン誤差を含む | 2.0 | -- | % FSR | |
| リファレンス ゲイン誤差を除く[b] | 0.4 | -- | % FSR | |
| 動作電流 | | | | |
| 低出力 | 140 | --- | µA | |
| 中出力 | 490 | --- | µA | |
| 高出力 | 1830 | --- | µA | |
| データ クロック | | 0.125 ~ 2.67 | 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 |

a. 入出力ピンを 含みます。

b. リファレンス ゲイン誤差は、テストマルチプレクサを使用し、信号経路を指定してピンに戻される $V_{RefHigh}$ および $V_{RefLow}$ と外部リファレンスを比較して測定します。

以下の表で別途指定されている場合を除き、$T_A$ = -40°C ~ +85°C、Vdd= 5.0V ±10%、パワー設定 HIGH、オペアンプ バイアス LOW ですべての制限が保証され、出力は P2[6] で 1.25 の外部 Vref を持つ P2[4] での 2.5V の外部アナログ グラウンドを基準とし、分解能は 12 ビットに設定されています。

Table 3. 5.0V ADCINCVR の DC および AC 電気的特徴、PSoC デバイスの CY8C25/26xxx ファミリ

| パラメータ | 標準値 [1] | 制限 | 単位 | 条件および注意 |
|---|---|---|---|---|
| 入力 | | | | |
| 入力電圧範囲 [2] | --- | Vss ~ Vdd | | Ref Mux = Vdd/2 ± Vdd/2 |
| 入力静電容量 [3] | 0.8 | --- | pF | |
| 入力インピーダンス [4,5] | 1/(C*clk) | --- | W | |
| 分解能 | --- | 7 ~ 13 | ビット | 2 の補数 |
| サンプリング速度 | --- | 4 ~ 10,000[6] | SPS | 1 秒あたりのサンプル数 |
| SNR[7] | 68 | --- | dB | 100 sps |
| DC 精度 | | | | |
| INL | 0.5 | 1 | LSB | |
| DNL | 0.25 | 0.5 | LSB | |
| オフセット誤差 | 12 | 49 | mV | 外部 AGND を使用 |
| ゲイン誤差 | 0.5 | 1.5 | % FSR | リファレンス ゲイン誤差を除く |
| 動作電流 | | | | |
| 低出力 | 125 | --- | µA | |
| 中出力 | 240 | --- | µA | |
| 高出力 | 640 | 1100 | µA | |
| データ クロック | --- | 0.125 ~ 8[6] | 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

以下の表で別途指定されている場合を除き、$T_A$ = -40°C ~ +85°C、Vdd= 5.0V ±10%、パワー設定 HIGH、オペアンプ バイアス LOW ですべての制限が保証され、出力は P2[6] で 1.25 の外部 Vref を持つ P2[4] での 2.5V の外部アナログ グラウンドを基準とし、分解能は 12 ビットに設定されています。

Table 4. 5.0V ADCINCVR の DC および AC 電気的特徴、PSoC デバイスの CY8C25/26xxx ファミリ

| パラメータ | 典型 [1] | 制限 | 単位 | 条件および注意 |
|---|---|---|---|---|
| 入力 | | | | |
| 入力電圧範囲 [2] | --- | Vss ~ Vdd | | Ref Mux = Vdd/2 ± Vdd/2 |
| 入力静電容量 [3] | 0.8 | --- | pF | |
| 入力インピーダンス [4,5] | 1/(C*clk) | --- | W | |
| 分解能 | --- | 7 ~ 13 | ビット | 2 の補数 |
| サンプリング速度 | --- | 4 ~ 10,000[6] | SPS | 1 秒あたりのサンプル数 |
| SNR[7] | 68 | --- | dB | 100 sps |
| DC 精度 | | | | |
| INL | 0.5 | 1 | LSB | |
| DNL | 0.25 | 0.5 | LSB | |
| オフセット誤差 | 12 | 49 | mV | 外部 AGND を使用 |
| ゲイン誤差 | 0.5 | 1.5 | % FSR | リファレンス誤差を除く |
| 動作電流 | | | | |
| 低出力 | 125 | --- | µA | |
| 中出力 | 240 | --- | µA | |
| 高出力 | 640 | 1100 | µA | |
| データ クロック | --- | 0.125 ~ 8[6] | 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 |

**Note** 電気的特徴

1. 標準値はパラメータ基準の +25°C で測定された値です。
2. 最大値を超えた入力電圧は最大の正の測定値を示します。最大値未満の入力電圧は最小の負の測定値を示します。
3. ユーザ モジュールのみで、入出力ピンは含みません。
4. 入力静電容量または入力インピーダンスは、アナログ ブロックへの入力が直接ピンに対するものである場合のみに該当します。
5. C = 入力静電容量、clk = データ クロック ( アナログ コラムのクロック )。
6. 仕様は、別途注記のない限り、サンプリング速度 100 sps、データ クロック 8 MHz の場合です。サンプリング速度はデータ クロックと分解能の両方に依存します。
7. SNR = 単一のトーンのフルスケールの出力を $F_{sample}/2$ に積分した全ノイズで除算した割合です。



[+] Feedback

## 配置

ADC 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。ADC ブロックは、配置される特定のコラムのコンパレータ バスを独占して使用できる必要があります。

カウンタ ブロックは、使用可能なデジタルブロックであればどれにでも配置できますが、PWM16 は特定の場所にしか配置できません。有効な配置については、以下の表を参照してください。

Table 5. 有効な配置

| 部品ファミリ | 有効な PWM16 配置の LSB/MSB |
|---|---|
| CY8C27xxx | DBB00/DBB01、DBB01/DCB02、DBB10/DBB11、DBB11/DCB12 |
| CY8C24xxx/CY8C22xxx | DBB00/DBB01、DBB01/DCB02 |

Table 6. CY8C26/25xxx の有効な配置

| 部品ファミリ | 有効な PWM16 配置の LSB/MSB |
|---|---|
| CY8C26xxx/CY8C25xxx | DBA01/DBA02、および DCA05/DCA06 |

両方のデジタル ブロックには割り込みサービス ルーチンが含まれています。絶対必要な条件ではありませんが、カウンタ ブロックが PWM16 ブロックよりも高い割り込み優先順位を持つことが望ましいです。したがって、PWM16 ブロックが配置されるブロック番号よりも低いデジタル ブロック番号の位置にカウンタ ブロックを配置することを推奨します。

**Note** ADCINCVR の配置には DEC_CR0[7:4] および DEC_CR1[5:3] を使用します。この ADC では、デシメータは使用されませんが、ゲート制御を目的としてデシメータ レジスタが使用されるため、この ADC の複数のインスタンスを配置する機能が制限されます。

## パラメータおよびリソース

Input（入力）

入力は、アナログ PSoC ブロックの配置後に選択します。8 個のスイッチド キャパシタ ブロックはそれぞれ異なる入力選択ができます。各ブロックは隣接するブロックのほとんどに接続でき、一部は外部入力ピンに直接接続できます。アナログ ブロックは、そのブロックに入力信号を伝える伝える信号経路を考慮しながら配置する必要があります。一部の配置では、パッケージ ピンから直接入力することができます。このような直接の接続では、電源レールの 40 mV 以内の入力を正確に測定できます。信号は、コラムのマルチプレクサ、または CT ブロックのテスト マルチプレクサを経由して、ADCINCVR が電源レール付近で信号を測定することができるアナログ コラムに乗せることも可能です。

ClockPhase（クロック位相）

クロック位相の選択は、あるスイッチド キャパシタのアナログ PSoC ブロックの出力を別のスイッチド キャパシタのアナログ 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 キャパシタのアナログ PSoC ブロックは、2 相クロック（$\varphi_1$、$\varphi_2$）を使用して信号を取得および転送します。一般的に、ADCINCVR への入力は、通常の設定、$\varphi_1$ でサンプリングされます。ユーザ モジュールの多くは、$\varphi_1$ 中に出力を自動的にゼロに設定し、$\varphi_2$ 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 ADCINCVR 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と、切り替え設定、$\varphi_2$ 中に入力信号を取得するように、位相を交換できます。

ADCResolution（ADC 分解能）

このパラメータを選択すると、デバイス エディタで ADCINCVR の分解能を設定できます。分解能を設定または変更するための API ルーチンもありますが、デバイス エディタで設定を行えば、必要ありません。分解能は、API 呼び出しによってもいつでも変更できますが、ADC の動作が停止するため、再起動する必要があります。有効な分解能設定は、7 以上 13 以下です。

CalcTime（計算時間）

CalcTime は、次の積分サイクルを開始する前に中間の積分結果を CPU が計算するためにかかる時間です。結果の計算にかかる時間、「CalcTime」は、CPU クロックに反比例して変化します。この値は、DataClock に換算する必要があります。最小 CPU 計算時間は、180 CPU クロックです。また、CalcTime は、サンプリング速度を最適化するために増加させることもできます。

**Note** CalcTime と $2^{ビット+2}$ の和が $2^{16}$-1、つまり 65,535 を超えないように注意する必要があります。

設定すべき CalcTime を決定するには、以下の式 19 を使用できます。

**Equation 19**

$$CalcTime \geq \frac{180 \cdot DataClock}{CPU_Clock}$$

たとえば、DataClock に 1.5 MHz が設定されており、CPU が 6 MHz で動作している場合は、CalcTime を 45 以上に設定する必要があります。

Clock and Integrator Column Clock（クロックと積分器コラムのクロック）

データ クロックにより、サンプリング速度と信号サンプル時間枠が決まります。このクロックは、カウンタ ブロックのクロック入力、16 ビットの PWM ブロック、および積分器を含むコラムのコラム クロックに送る必要があります。

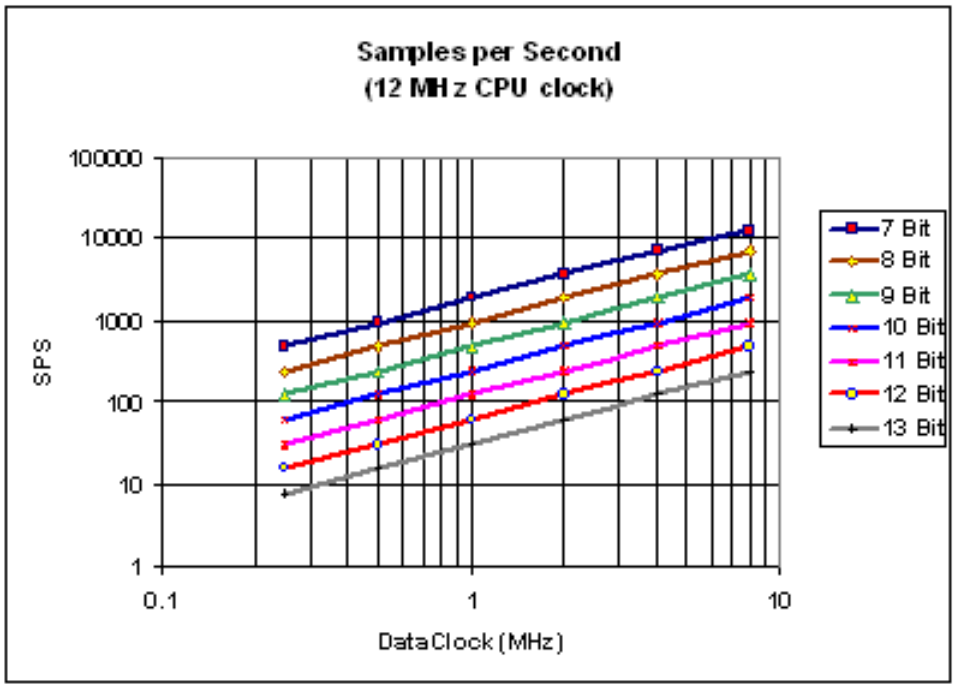
**Note** 積分器スイッチド キャパシタ ブロックのコラム クロックには、同じクロックを手動で設定する必要があります。3 つのブロックすべてで同じクロックを使用する必要があります。同じクロックを使用しないと、このユーザ モジュールは正しく機能しません。

このパラメータ設定では、カウンタ ブロックおよび PWM ブロックだけにクロックが設定されます。このクロックは、125 kHz ~ 8 MHz の間のクロック レートを持つ任意のクロック ソースです。

$$SampleRate = \frac{DataClock}{2^{Bits+2} + CalcTime}$$

図 5 は、ADCINCVR の分解能オプションごとに使用可能なサンプリング速度を示したものです。

Figure 5. ADCINCVR のサンプリング速度




DataFormat（データフォーマット）

このパラメータを選択すると、結果が返されるフォーマットが決まります。「Signed（符号付き）」を選択し、選択された分解能が「N」であるとすると、結果の範囲は $2^{N-1}$ ~ $2^{N-1}$ –1 となります。「Unsigned（符号なし）」を選択すると、結果の範囲は 0 ~ $2^{N}$-1 となります。データ フォーマットと分解能ごとの結果範囲については、以下の表を参照してください。

Table 7. ADCINCVR データ フォーマットと分解能の結果範囲

| 分解能設定 | 符号付きデータ フォーマット | 符号なしデータ フォーマット |
|---|---|---|
| 7 | -64 ~ 63 | 0 ~ 127 |
| 8 | -128 ~ 127 | 0 ~ 255 |
| 9 | -256 ~ 255 | 0 ~ 511 |
| 10 | -512 ~ 511 | 0 ~ 1023 |
| 11 | -1024 ~ 1023 | 0 ~ 2047 |
| 12 | -2048 ~ 2047 | 0 ~ 4095 |
| 13 | -4096 ~ 4095 | 0 ~ 8191 |

### 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。これらは、**Project（プロジェクト）> Settings（設定）> Chip Editor（チップ エディタ）**で使用できます。複数のオーバーレイが使用され、そのオーバーレイ全体の複数のユーザ モジュールにより割り込みが共用されている場合は、割り込み生成制御が重要です。

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ーレイ内に存在する複数のユーザ モジュールで共用されている割り込みについて、割り込み要求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ー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、オーバーレイが最初にロードされたときだけ共用割り込み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、RAM のバイトを消費します。

## グローバル リソース

使用可能な入力電圧は、デバイス エディタの「Global Resource（グローバル リソース）」セクションの「Ref Mux（リファレンス マルチプレクサ）」の選択内容によって決まります。Ref Mux の選択内容によって、アナログ グラウンドとアナログ グラウンドの入力電圧の使用可能範囲が決まります。たとえば、「Vdd/2 +/-BandGap」を選択し、Vdd = 5V の場合、使用可能な入力範囲は 2.5 ± 1.3V（1.2 ~ 3.8V）です。「Vdd/2 ± Vdd/2」を選択すると、使用可能な入力電圧はレール間振幅の電源範囲全体となります。以下の表は、Vdd が 5V または 3.3V の場合の有効範囲を一覧にしたものです。

Table 8. RefMux 設定に対する CY8C25/26xxx の入力電圧範囲

| RefMux の設定 | Vdd = 5V | Vdd = 3.3V |
|---|---|---|
| (Vdd/2) ± BandGap | $1.2 < V_{in} < 3.8$ | $0.35 < V_{in} < 2.95$ |
| (Vdd/2) ± (Vdd/2) | $0 < V_{in} < 5$ | $0 < V_{in} < 3.3$ |
| (2*BandGap) ± BandGap | $1.3 < V_{in} < 3.9$ | 適用外 |
| (2*BandGap) ± P2[6] | $(2.6 - V_{P2[6]}) < V_{in} < (2.6 + V_{P2[6]})$ | 適用外 |
| P2[4] ± BandGap | $(V_{P2[4]} - 1.3) < V_{in} < (V_{P2[4]} + 1.3)$ | $(V_{P2[4]} - 1.3) < V_{in} < (V_{P2[4]} + 1.3)$ |
| P2[4] ± P2[6] | $(V_{P2[4]}-V_{P2[6]}) < V_{in} < (V_{P2[4]}+V_{P2[6]})$ | $(V_{P2[4]}-V_{P2[6]}) < V_{in} < (V_{P2[4]}+V_{P2[6]})$ |

Table 9. リファレンス マルチプレクサ設定ごとの CY8C29/27/24/22xxx の入力電圧範囲

| RefMux の設定 | Vdd = 5V | Vdd = 3.3V |
|---|---|---|
| (Vdd/2) ± BandGap | $1.2 < V_{in} < 3.8$ | $0.35 < V_{in} < 2.95$ |
| (Vdd/2) ± (Vdd/2) | $0 < V_{in} < 5$ | $0 < V_{in} < 3.3$ |
| BandGap ± BandGap | $0 < V_{in} < 2.6$ | $0 < V_{in} < 2.6$ |
| (1.6*BandGap) ± (1.6*BandGap) | $0 < V_{in} < 4.16$ | 適用外 |
| (2*BandGap) ± BandGap | $1.3 < V_{in} < 3.9$ | 適用外 |
| (2*BandGap) ± P2[6] | $(2.6 - V_{P2[6]}) < V_{in} < (2.6 + V_{P2[6]})$ | 適用外 |
| P2[4] ± BandGap | $(V_{P2[4]} - 1.3) < V_{in} < (V_{P2[4]} + 1.3)$ | $(V_{P2[4]} - 1.3) < V_{in} < (V_{P2[4]} + 1.3)$ |
| P2[4] ± P2[6] | $(V_{P2[4]}-V_{P2[6]}) < V_{in} < (V_{P2[4]}+V_{P2[6]})$ | $(V_{P2[4]}-V_{P2[6]}) < V_{in} < (V_{P2[4]}+V_{P2[6]})$ |

## 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、設計者が高級言語でモジュールを操作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、各機能に対するインタフェースを「include」ファイルによって提供される関連定数とともに示します。

**Note** すべてのユーザ モジュール API の場合と同じように、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。A と X の値が呼び出し後に必要な場合は、呼び出しの前に呼び出し元関数で A と X の値を保存してください。PSoC Designer のバージョン 1.0 以降では、効率を高めるために、この「registers are volatile（レジスタの揮発性）」ポリシーが選択され、実施されています。C コンパイラは、自動的にこの条件で処理されています。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する必要があります。一部のユーザ モジュール API 関数では、A と X は変更されないこともありますが、将来も変更されないという保証はありません。

エントリ ポイントは、ADC を初期化し、サンプリングを開始し、ADC を停止するために提供されています。モジュールの「インスタンス名」は、必ず以下のエントリ ポイントに示されている「ADCINCVR」のプリフィックスと置き換えます。間違ったインスタンス名の使用は、構文エラーの一般的な原因です。

### ADCINCVR_Start

説明：

このユーザ モジュールで必要なすべ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。

C プロトタイプ：

```
void  ADCINCVR_Start(BYTE bPower)
```

アセンブラ：

```
mov   A, ADCINCVR_HIGHPOWER
lcall  ADCINCVR_Start
```

パラメータ：

Power：出力レベルを指定する 1 バイト。リセットと構成の後、ADCINCVR 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 |
|---|---|
| ADCINCVR_OFF | 0 |
| ADCINCVR_LOWPOWER | 1 |
| ADCINCVR_MEDPOWER | 2 |
| ADCINCVR_HIGHPOWER | 3 |

出力レベルは、アナログ性能に影響します。正しい出力設定はデータ クロックのサンプル レートの影響を受けやすく、アプリケーションごとに決める必要があります。開発開始時にはフル出力を選択することを推奨します。その後、テストを実行して、出力設定をどれだけ低く設定するかを決めることができます。

戻り値：

なし

注意事項：

A および X レジスタは、この関数のこの実装または将来の実装で変更される場合があります。大容量メモリ モデル（CY8C29xxx）のすべての 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 レジスタだけが変更されています。

## ADCINCVR_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  ADCINCVR_SetPower(BYTE bPower)
```

アセンブラ：

```
mov   A, [bPower]
lcall  ACDINCVR_SetPower
```

パラメータ：

Power：上記の 「Start」API ルーチンで使用した bPower パラメータと同じです。これにより、ADC を動作させながら、出力レベルを変更できます。

戻り値：

なし

注意事項：

A および X レジスタは、この関数のこの実装または将来の実装で変更される場合があります。大容量メモリ モデル（CY8C29xxx）のすべての 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 レジスタだけが変更されています。

## ADCINCVR_SetResolution

説明：

A/D 変換器の分解能を設定します。

C プロトタイプ：

```
void  ADCINCVR_SetResolution(BYTE bResolution)
```

アセンブラ：

```
mov   A, [bResolution]
lcall  ADCINCVR_SetResolution
```

パラメータ：

Resolution：A/D 変換器の分解能は、デバイス エディタまたはユーザ ファームウェアのいずれかで設定できます。ファームウェアで設定されていない場合、ADC は、デバイス エディタで設定されている分解能を使用します。分解能の値には、7 ~ 13 ビットを設定できます。

戻り値：

ADCINCVR が入力をサンプリングしている場合、この関数が呼び出されると、サンプリングが停止されます。

注意事項：

A および X レジスタは、この関数のこの実装または将来の実装で変更される場合があります。大容量メモリ モデル（CY8C29xxx）のすべての 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。

## ADCINCVR_Stop

説明：

スイッチド キャパシタ積分器ブロックの出力レベルを 0ff に設定します。この設定は、ADCINCVR が使用されておらず、節電したいときに行います。このルーチンは、アナログ スイッチ キャパシタ ブロックの電源を遮断し、デジタル ブロックを無効化します。最も低い消費電力を実現するためには、クロックをデジタル ブロックから取り除く必要もあります。

C プロトタイプ：

```
void  ADCINCVR_Stop(void)
```

アセンブラ：

```
lcall  ACDINCVR_Stop
```

パラメータ：

なし

戻り値：

なし

注意事項：

A および X レジスタは、この関数のこの実装または将来の実装で変更される場合があります。大容量メモリ モデル（CY8C29xxx）のすべての 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。

## ADCINCVR_GetSamples

説明：

ADC アルゴリズムを初期化して開始し、指定された数のサンプルを収集します。必ず M8C.inc または M8C.h で M8C_EnableGInt マクロを呼び出して、グローバル割り込みを有効化する *M8C.inc* こと *M8C.h*。

C プロトタイプ：

```
void  ADCINCVR_GetSamples(BYTE bNumSamples)
```

アセンブラ：

```
mov   A, [bNumSamples]
lcall  ADCINCVR_GetSamples
```

パラメータ：

NumSamples：検索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

注意事項：

A および X レジスタは、この関数のこの実装または将来の実装で変更される場合があります。大容量メモリ モデル（CY8C29xxx）のすべての 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 レジスタだけが変更されています。

## ADCINCVR_StopAD

説明：

ADC を即座に停止します。

C プロトタイプ：

```
void  ADCINCVR_StopAD(void)
```

アセンブラ：

```
lcall  ADCINCVR_StopAD
```

パラメータ：

なし

戻り値：

なし

注意事項：

A および X レジスタは、この関数のこの実装または将来の実装で変更される場合があります。大容量メモリ モデル（CY8C29xxx）のすべての 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。

## ADCINCVR_fIsData、ADCINCVR_fIsDataAvailable

説明：

データ変換が完了し、データを読み取る準備が整うと、非ゼロ値を返します。

C プロトタイプ：

```
CHAR  ADCINCVR_fIsDataAvailable(void)
CHAR  ADCINCVR_fIsData(void)
```

アセンブラ：

```
lcall  ADCINCVR_fIsDataAvailable
```

パラメータ：

なし

戻り値：

データを使用できる場合は、非ゼロ値を返します。

注意事項：

A および X レジスタは、この関数のこの実装または将来の実装で変更される場合があります。大容量メモリ モデル（CY8C29xxx）のすべての 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 レジスタだけが変更されています。

## ADCINCVR_iGetData

説明：

最後に変換したデータを返します。データが有効なことを確認するため、データを取得する前に ADCINCVR_fIsDataAvailable() を呼び出してください。データは、次の変換サイクルが完了する前に取得する必要があります。取得しなかった場合、データは上書きされます。積分時間の最後にこの関数を正確に呼び出さないと、返されるデータが壊れる可能性があります。したがって、サンプリング レートより高い周波数でデータを検索することを強く推奨します。また、これを保証できない場合は、この関数を呼び出す前に割り込みをオフにすることを推奨します。

C プロトタイプ：

```
INT   ADCINCVR_iGetData(void)
```

アセンブラ：

```
lcall  ADCINCVR_iGetData
```

パラメータ：

なし

戻り値：

変換結果が返されます。アセンブラでは、X で MSB、累算器で LSB が返されます。

注意事項：

A および X レジスタは、この関数のこの実装または将来の実装で変更される場合があります。大容量メモリ モデル（CY8C29xxx）のすべての 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 レジスタだけが変更されています。

## ADCINCVR_ClearFlag

説明：

データ使用可能フラグを解除します。この関数は、データを読み取った後で呼び出すべきです。

C プロトタイプ：

```
void  ADCINCVR_ClearFlag(void)
```

アセンブラ：

```
lcall  ADCINCVR_ClearFlag
```

パラメータ：

なし

戻り値：

なし

注意事項：

A および X レジスタは、この関数のこの実装または将来の実装で変更される場合があります。大容量メモリ モデル（CY8C29xxx）のすべての 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 レジスタだけが変更されています。

## ADCINCVR_iGetDataClearFlag

説明：

最後に変換したデータを返し、データ使用可能フラグを解除します。データが有効なことを確認するため、データを取得する前に ADCINCVR_fIsDataAvailable() を呼び出してください。データは、次の変換サイクルが完了する前に取得する必要があります。取得しなかった場合、データは上書きされます。積分時間の最後にこの関数を正確に呼び出さないと、返されるデータが壊れる可能性があります。したがって、サンプリング レートより高い周波数でデータを取得することを強く推奨します。また、これを保証できない場合は、この関数を呼び出す前に割り込みをオフにすることを推奨します。

C プロトタイプ：

```
INT   ADCINCVR_iGetDataClearFlag(void)
```

アセンブラ：

```
lcall  ADCINCVR_iGetDataClearFlag
```

パラメータ：

なし

戻り値：

変換結果が返されます。アセンブラでは、X で MSB、累算器で LSB が返されます。

注意事項：

A および X レジスタは、この関数のこの実装または将来の実装で変更される場合があります。大容量メモリ モデル（CY8C29xxxc）のすべての 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 レジスタだけが変更されています。

## ファームウェア ソースコードの例

**このサンプル コードでは、連続的な変換を開始し、データ使用可能フラグをポーリングし、変換されたバイトをユーザ関数に送信します。**

```
;;;  Sample Code for the ADCINCVR
;;;  Continuously sample and call a user routine with the converted
;;;  data sample.
;;;
;;;  NOTE: The User Routine must complete operation within one
;;;        conversion cycle, in order to retrieve the next converted
;;;        sample data.
;;;

include "m8c.inc"                ; part specific constants and macros
include "PSoCAPI.inc"            ; PSoC API definitions for all User Modules

export _main

_main:
    M8C_EnableGInt                    ;Enable interrupts
    mov   a, 10                       ;Set resolution to 10 Bits
    call  ADCINCVR_SetResolution

    mov   a, ADCINCVR_HIGHPOWER       ;Set Power and Enable A/D
    call  ADCINCVR_Start

    mov   a, 00h                      ;Start A/D in continuous sampling mode
    call  ADCINCVR_GetSamples

;A/D conversion loop
loop1:

wait:                                 ;Poll until data is complete
    call  ADCINCVR_fIsDataAvailable
    jz    wait

    call  ADCINCVR_ClearFlag          ;Reset flag
    call  ADCINCVR_iGetData           ;Get Data - X=MSB A=LSB
    ; Place user code here

    jmp   loop1
```

C で記述した同じプロジェクト。

```
//--------------------------------------------------------------------
//  Sample C Code for the ADCINCVR
//  Continuously sample and call a user function with the data.
//
//--------------------------------------------------------------------
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules

void main(void)
{
    INT iData;
    M8C_EnableGInt;                     // Enable global interrupts
    ADCINCVR_Start(ADCINCVR_HIGHPOWER); // Turn on Analog section
    ADCINCVR_SetResolution(10);         // Set resolution to 10 Bits
    ADCINCVR_GetSamples(0);             // Start ADC to read continuously
    for(;;)
    {
        while(ADCINCVR_fIsDataAvailable() == 0); // Wait for data to
                                                 // be ready.
        iData = ADCINCVR_iGetData();    // Get Data
        ADCINCVR_ClearFlag();           // Clear data ready flag
       // Place user code here
    }
}
```

## 設定レジスタ

これらのレジスタは、初期化および API ライブラリによって構成されます。ユーザが、これらのレジスタを直接変更または読み取る必要はありません。このセクションは、参考用として示したものです。

ADC は、スイッチド キャパシタ PSoC ブロックです。ADC 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、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。

Table 10.　ブロック ADC：レジスタ CR0

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

Table 11.　ブロック ADC：レジスタ CR1

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | ACMux、AMux | | | 0 | 0 | 0 | 0 | 0 |

ACMux は、タイプ「A」のブロックにブロックを配置する場合に使用します。フィールドの値は、ユーザが入力を接続する方法によって異なります。AMux は、タイプ「B」のブロックにブロックを配置する場合に使用します。フィールドの値は、ユーザが入力を接続する方法によって異なります。

Table 12. ブロック ADC：レジスタ CR2

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

Table 13. ブロック ADC：レジスタ CR3

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | 1 | 1 | 1 | FSW0 | 0 | 0 | 0 | 0 |

FSW0 は、PWM16 割り込みハンドラおよび各種 API によって使用されます。値が 「0」の場合、ADC は無効化された積分器となります。値が 「1」の場合、ADC は有効化された積分器となります。

PWM16 は、ADC の積分時間を制御するために使用されるデジタル PsoC ブロックです。比較値には 2 ビット +2 を設定し、時間には CalcTime と比較値の和を設定します。

Table 14. ブロック PWM16_MSB：レジスタ関数

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | 0 | 0 | 1 | Compare Type | Interrupt Type | 0 | 0 | 1 |

Compare Type （比較タイプ）は、キャプチャ比較が 「以下」と 「未満」のどちらなのかを示すフラグです。Interrupt Type （割り込みタイプ）は、キャプチャ イベントと秒読み条件のどちらで割り込みをトリガするかを示すフラグです。どちらのパラメータも、デバイス エディタで設定します。

Table 15. ブロック PWM16_LSB：レジスタ関数

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | 0 | 0 | 0 | Compare Type | 0 | 0 | 0 | 1 |

Compare Type （比較値）は、比較関数が 「以下」と 「未満」のどちらに設定されているかを示すフラグです。このパラメータは、デバイス エディタで設定します。

Table 16. ブロック PWM16_MSB：レジスタ入力

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | 0 | 0 | 1 | 1 | クロック | | | |

クロックは、16 のクロック ソースからクロック入力を選択します。このパラメータは、デバイス エディタで設定します。

Table 17. ブロック PWM16_LSB：レジスタ入力

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | Enable | | | | クロック | | | |



[+] Feedback

Enable は 16 の入力ソースからデータ入力を 1 つ選択し、クロックは 16 のソースからクロック入力を 1 つ選択します。どちらのパラメータも、デバイス エディタで設定します。

Table 18. ブロック PWM16_MSB：レジスタ出力

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | 0 | 0 | 0 | 0 | 0 | Output Enable | Output Sel | |

Output Enable は、出力が有効なことを示すフラグです。Output Sel は、PWM16 の出力が送られる場所を示すフラグです。どちらのパラメータも、デバイス エディタで設定します。

Table 19. ブロック PWM16_LSB：レジスタ出力

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 20. ブロック PWM16_MSB：カウント レジスタ DR0

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | Count(MSB) | | | | | | | |

Count: PWM16 PWM の下の MSB。この値は、PWM16 API を使用して読み取れます。

Table 21. ブロック PWM16_LSB：カウント レジスタ DR0

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | Count(LSB) | | | | | | | |

Count: PWM16 PWM の下の LSB。この値は、PWM16 API を使用して読み取れます。

Table 22. ブロック PWM16_MSB：ピリオド レジスタ DR1

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | Period(MSB) | | | | | | | |

Period は、イネーブルまたは最終カウント条件によってカウンタ レジスタにロードされる時間値の MSB を保持します。この値は、デバイス エディタおよび PWM16 API で設定できます。

Table 23. ブロック PWM16_LSB：ピリオド レジスタ DR1

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | Period(LSB) | | | | | | | |

Period は、イネーブルまたは最終カウント条件によってカウンタ レジスタにロードされる時間値の LSB を保持します。この値は、デバイス エディタおよび PWM16 API で設定できます。

Table 24. ブロック PWM16_MSB：パルス幅レジスタ DR2

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | Pulse Width(MSB) | | | | | | | |

PulseWidth は、比較イベントの生成に使用されるパルス幅の値の MSB を保持します。この値は、デバイス エディタおよび PWM16 API で設定できます。

Table 25. ブロック PWM16_LSB：パルス幅レジスタ DR2

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | Pulse Width(LSB) | | | | | | | |

PulseWidth は、比較イベントの生成に使用されるパルス幅の値の LSB を保持します。この値は、デバイス エディタおよび PWM16 API で設定できます。

Table 26. ブロック PWM16_MSB：制御レジスタ CR0

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Start/ Stop(0) |

Start/Stop は、LSB 制御レジスタの値によって制御されます。ゼロに設定してください。

Table 27. ブロック PWM16_LSB：制御レジスタ CR0

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | | | | | | | | Start/ Stop |

Start/Stop が設定されている場合は、PWM16 がイネーブルです。この設定は、PWM16 API を使用して変更します。

CNT は、カウンタとして構成されるデジタル PSoC ブロックです。DR0 の値が最後までカウントダウンされると、上位桁のソフトウェア カウンタを減算するために割り込みが呼び出され、CNT が DR1 から再ロードされます。データは、DR2 を通して出力されます。

Table 28. ブロック CNT：レジスタ関数

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

Table 29. ブロック CNT：レジスタ入力

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | データ | | | | クロック | | | |

データは、ADC ブロックが配置されたコラム コンパレータを選択します。クロックは、16 のクロック ソースからクロック入力を 1 つ選択し、デバイス エディタで設定します。

Table 30. ブロック CNT：レジスタ出力

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 31. ブロック CNT：レジスタ DR0

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | カウント値 | | | | | | | |

Table 32. ブロック CNT：レジスタ DR1

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Table 33. ブロック CNT：レジスタ DR2

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | Data Out | | | | | | | |

Data Out は、カウンタ値を取得するために API によって使用されます。

Table 34. ブロック CNT：レジスタ CR0

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Enable |

Enable が設定されている場合は、CNT が有効です。この値は、ADCINCVR API によって変更および制御されます。

Table 35. レジスタ INT_MSK1

| ビット | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 値 | | | | | | | | |

ここでは、個々の割り込みを有効にするために、TMR ブロックと CNT ブロックに対応するマスク ビットを設定します。実際のマスク値は、各ブロックの配置位置によって決まります。

# バージョン履歴

| バージョン | 著者 | 説明 |
|---|---|---|
| 3.2 | DHA | 以下を確認するために DRC を追加。<br>1. デジタル リソースとアナログ リソース間でソース クロックが異なっている。<br>2. ADC クロックが CPU クロックより上位。<br><br>2 つの ADCINC14 または ADCINCVR ブロックがプロジェクトに追加された場合の DRC 警告を追加。 |

**Note** PSoC Designer 5.1 では、すべてのユーザ モジュール Data Sheet にバージョン履歴を導入しました。このセクションには、ユーザ モジュールの現行バージョンと旧バージョンとの相違に関する高度な説明が記載されています。



Copyright © 2001-2010 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.