8-Bit 電圧出力多重 DAC データシート MDAC8 V 2.2

Copyright © 2001-2011 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	フラッシュ	RAM	
CY8C29/27/26/25/24/22xxx, CY8C23x33, CY8CLED04/08/16, CY8CLED0xD, CY8CLED0xG, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C28x43, CY8C28x52						
	0	0	2	255	0	1

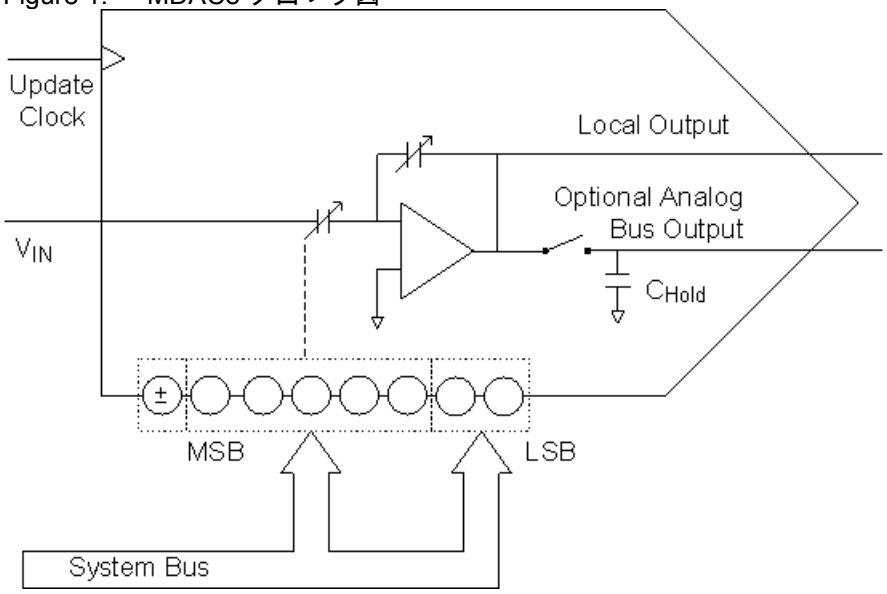
このユーザ モジュールを使用する機能例として完全に設定されたプロジェクトについては、以下を参照してください。www.cypress.com/psocexampleprojects

特性と概要

- 8-bit 解像度
- 電圧出力
- 4 象限乗算
- 2 の補数、オフセットバイナリ、および符号 / 絶対値表現の入力データフォーマット
- アナログバスと外部出力装置用のサンプルアンドホールド回路
- 最大 125 ksps のアップデートレート

MDAC8 は 8-bit で、入力電圧に比例したデジタルコードを出力する 4 象限多重 DAC です。MDAC8 は毎秒最大 250k サンプルのアップデートレートで、デジタルコードを出力電圧に換算します。アプリケーション•プログラミング•インタフェース (API) は、オフセット•バイナリ、2 の補数、および符号 / 絶対値表現をサポートします。オフセット補正により変換エラーを最小化します。

Figure 1. MDAC8 ブロック図



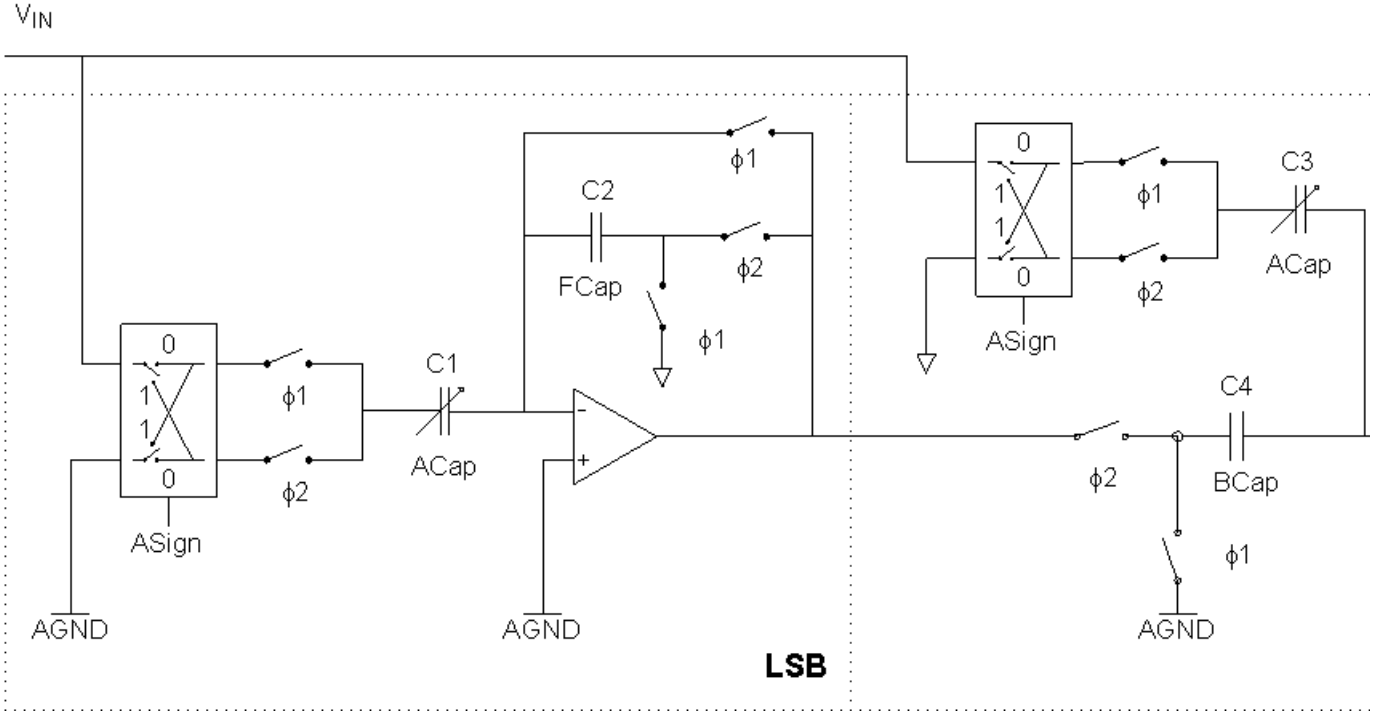
機能説明

MDAC8 ユーザーモジュールは、アナログ入力電圧とデジタルコードを乗算します。デジタルコードは、2 の補数、もしくは -127 ~ +127 の範囲の符号 / 絶対値表現として表示されます。もしくは、入力コードを 0 ~ 254 の範囲の、オフセットバイナリ形式の数値で表記することができます。入力及び出力電圧は、システムレベルのパラメータである RefMux とともに設定される、AGND を基準とします。SetOutputRange API 関数で選択された値によって、入力電圧は 1 倍か 2 倍になります。

MDAC8 ユーザーモジュールは、任意の二つのアナログ PSoC ブロックに対しマッピングします。これらのブロックは、LSB と MSB と呼ばれます。LSB ステージとも呼ばれる LSB ブロックは、“BCap" キャパシタ C_4 を通して MSB ステージと接続しています。内部では、この操作は符号 / 絶対値表現で行われいます。絶対値が大きい方から 5 つのビットが、以下の模式図におけるバイナリ加重のコンデンサ配列の C_3 値に設定されます。絶対値が小さい方から 2 つのビットが、C_1 値に設定されます。キャパシタの単位で、C_3 は 0 から 31 までの値、C_1 は {0, 8, 16, 24} のいずれかの値をとるものとします。各ステージの (ASign ビットの値によっては反転された) リファレンス電圧は、それぞれマグニチュードキャパシタ C_1 と C_3 の比、およびフィードバックキャパシタ C_2 と C_5 の比でスケール調整されます。それぞれ、32 もしくは 16 ユニットの値をとります (入力電圧ゲインは 32 ユニットの場合 1、16 ユニットの場合 2)。LSB ステージの出力は、さらにカップリングキャパシタ C_4 とフィードバックキャパシタ C_5 の比でスケール調整されます。SetOutputRange API 関数は、C_2 と C_5 の値を両方変更します。

[+] Feedback

Figure 2. MDAC8 の模式図



ハードウェアは、各アップデートサイクルでオフセット補正を行います。φ_1 及び φ_2 によって制御されるスイッチが、φ_1 の間、ユニティゲインフォロアとしてオペアンプを構成します。この構成では、オフセット電圧がサミングノードにかかり、ACaps、BCaps、FCaps を充電します。φ_2 で再構成されたように、回路はこれらのコンデンサのオフセット電荷を反転し、オフセット電圧を実質的にキャンセルします。

各アップデート周期で、オフセット補正の直接的な効果として、V_{out} は オペアンプオフセット電圧 (φ_1 の間) と目標電圧 (φ_2 の間に安定する) の間をスルーします。精度を高めるためのこの代償を軽減する一つの方法は、出力バスに関連するサンプルアンドホールド回路を使用することです。V_{out} は、ロードおよびホールドコンデンサ (MDAC8 ブロックダイアグラムの C_{Hold}) を、φ_2 の後半に充電します。C_{Hold} は、このサイクルが終了するときにオペアンプ出力から切り離されます。各アナログ出力バスには、十分高い入力インピーダンスのアナログ出力バッファが付随します。

スケールされた入力を組み合わせた、出力は以下のようになります。

Equation 1

$$V_{Out} = (V_{IN}–AGND)\frac{C_1C_4}{C_2C_5} + (V_{IN}–AGND)\frac{C_3}{C_5} + AGND$$

デバイスエディターで、グロバルパラメータの RefMux が (2BandGap)±BandGap に設定された場合、AGND は 2.6 ボルトです。$C_F = C_2 = C_1$, $C_4 = 1$ 及び $C_F = 2^5$ と設定します。対応する出力は以下になります。

Equation 2

$$V_{Out} = (V_{IN} - 2.6)\frac{C_1}{2^5 2^5} + (V_{IN} - 2.6)\frac{C_3}{2^3} + 2.6$$

[+] Feedback

Equation 3

$$V_{Out} = V_{IN}\left(\frac{C_1}{2^{10}} + \frac{C_3}{2^5}\right) - 2.6\left(\frac{C_1}{2^{10}} + \frac{C_3}{2^5}\right) + 2.6$$

C_1 の値は、絶対値が小さい方から 2 つのビットに 8 を掛けた (左に 3 桁シフトした) スケーリング値に限られます。C_1 とその分母のスケール係数を打ち消しすことで、次のような結果が得られます。

Equation 4

$$V_{Out} = 2.6Volts + V_{IN}\left(\frac{C_1}{2^8} + \frac{C_3}{2^5}\right) - 2.6\left(\frac{C_1}{2^8} + \frac{C_3}{2^5}\right), C_1 \in 0, 1, 2, 3$$

例

外部ソースによる 1V の入力電圧が与えられた場合、式 3 は以下のようになります。

Equation 5

$$V_{Out} = 2.6Volts \pm 1.6Volts\left(\frac{C_1}{2^8} + \frac{C_3}{2^5}\right), C_1 \in 0, 1, 2, 3$$

MDAC への入力コードは、入力コードの MSB の符号をとります。ビット 2 - 7 に含まれている C_3 の値。C_1 の値は、入力コードの LSB の 2 ビットによってあらわされます。よって、値が -105 の場合、出力電圧は 1.3V になります。

Equation 6

$$V_{Out} = 2.6Volts - 1.6Volts\left(\frac{1}{256} + \frac{26}{32}\right) = 1.3V$$

計算された値は理想的な値であり、システムのノイズやチップのオフセットによって異なってきます。

4 象限乗算は、入力電圧と入力コードの両方が、出力電圧の正負を決定することを意味します。次の 3 つの図を見てください。

Figure 3.　入力電圧対時間

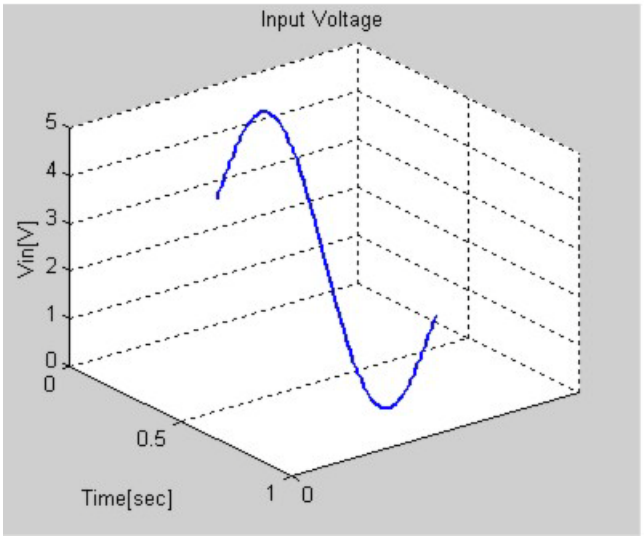


Figure 4.　出力電圧対入力コードと時間、FCap=32

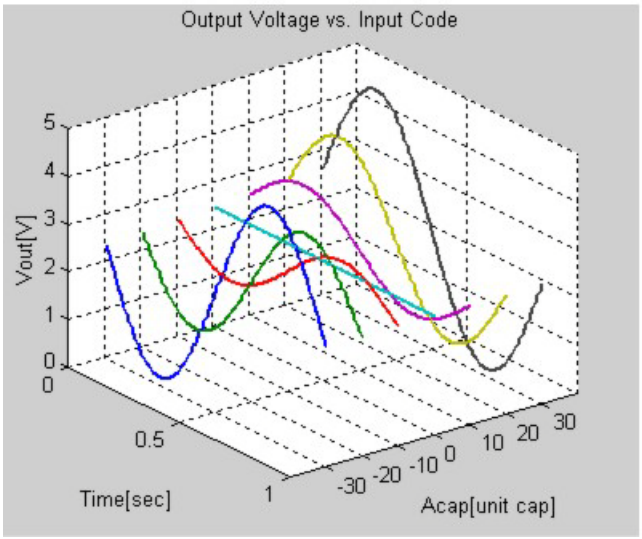
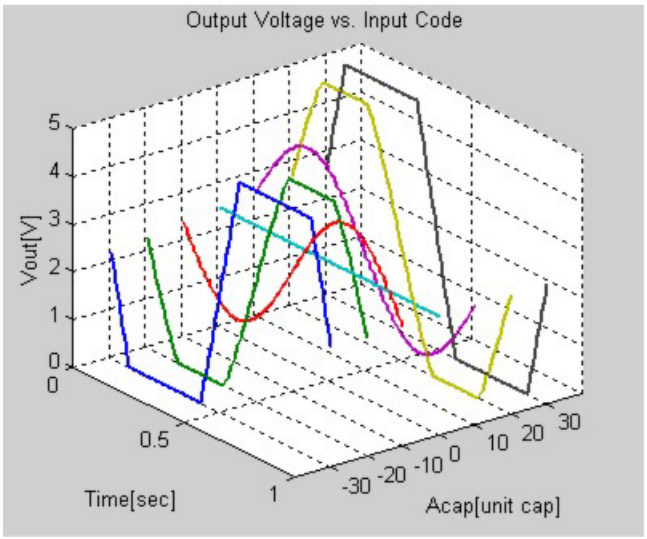


Figure 5.　出力電圧対入力コードと時間、FCap=16



上記のように、出力信号のクリッピングを防ぐためには、入力電圧を下げる必要があります。

DC 電気的特性と AC 電気的特性

以下の値は、初期の特性データを元に予測される性能を示しています。下の表に別途記されてない場合、T_A = 25°C 及び Vdd = 5V です。別途記されていない場合、f_{clock} = 125 kHz、外部 AGND 2.50V、外部 V_{Ref} 1.23V, REFPWR = HIGH, SCPOWER = ON, PSoC ブロック出力 HIGH です。

Table 1.　5.0V MDAC8 DC 及び AC の電気特性

パラメータ	標準値	リミット	単位	条件と注記
解像度	--	8	ビット	
線形性				
DNL	0.5	--	LSB	
INL	0.3	--	LSB	
単調な	可能	--		
ゲイン誤差				
リファレンス ゲイン誤差を含む	3.5	--	%FSR	
リファレンス ゲイン誤差を除く [3]	0.5	--	%FSR	
V_{OS}, オフセット電圧	±2.1	--	mV	

[+] Feedback

パラメータ	標準値	リミット	単位	条件と注記
出力ノイズ	4.5	--	mV rms	0 ~ 300 kHz
f_{clock}, 内部アップデートレート [1]				
低出力	2 ~ 125	--	kHz	
中出力	1 ~ 500	--	kHz	
高出力	1 ~ 800	--	kHz	
動作電流 [2]				
低出力	305	--	µA	
中出力	1130	--	µA	
高出力	4315	--	µA	

以下の値は、初期の特性データに基づく予測される性能を示しています。下の表に別途記されてない場合、T_A = 25°C 及び Vdd = 3.3V です。. 別途記されていない場合、f_{clock} = 125 kHz、外部 AGND 1.50V、外部 V_{Ref} 0.8V、REFPWR = HIGH, SCPOWER = ON, PSoC ブロック出力 HIGH です。

Table 2. 3.3V DAC8 DC 及び AC の電気特性

パラメータ	標準値	リミット	単位	条件と注記
解像度	--	8	ビット	
線形性				
DNL	0.5	--	LSB	
INL	0.4	--	LSB	
単調な	可能	--		
ゲイン誤差				
リファレンス ゲイン誤差を含む	2.6	--	%FSR	
リファレンス ゲイン誤差を除く [3]	0.3	--	%FSR	

パラメータ	標準値	リミット	単位	条件と注記
V_{OS}, オフセット電圧	±3.5	--	mV	
出力ノイズ	2.5	--	mV rms	0 ~ 300 kHz
f_{clock}, 内部アップデートレート [1]				
低出力	2 ~ 125	--	kHz	
中出力	1 ~ 500	--	kHz	
高出力	1 ~ 800	--	kHz	
動作電流 [2]				
低出力	270	--	µA	
中出力	1020	--	µA	
高出力	3900	--	µA	

電気特性に関する注記

1. φ_1, φ_2 の限界値：ブロードバンドノイズの 3dB が増加するため
2. すべてのアナログブロックに共通の、基準ブロックの出力は含まれていません。(PSoC ファミリのデータシートを参照)。

下の表に別途記されていない場合、全ての限界値は TA = 25C 及び Vdd = 5V で保証されてします。別途記されていない場合、fclock = 125 kHz、外部 AGND 2.50V、外部 VRef 1.23V, REFPWR = HIGH, SCPOWER = ON, PSoC ブロック出力 HIGH です。

Table 3. 5.0V MDAC8 の DC 及び AC 電気特性

パラメータ	一般的な [1]	限界値の [2]	単位	条件および注記
解像度	—	8	ビット	
線形性				
DNL	0.10	0.25	LSB	
INL	0.15	0.40	LSB	

[+] Feedback

パラメータ	一般的な [1]	限界値の [2]	単位	条件および注記
単調な	—	½	Bit（ビット）	
ゲイン誤差	1.0	2.5	%FSR	
V_{OS}, オフセット電圧 [3]	8	43	mV	
出力ノイズ				
帯域限界	0.3	1	mV rms	0 ~ 10 kHz
広帯域	7	10	mV rms	0 ~ 300 kHz
f_{clock}, 内部アップデートレート [4]	—	32 ~ 333	kHz	
V_{in}（帯域幅）	40	—	kHz	
動作電流 [5]				
低出力	250	—	µA	
中出力	560	—	µA	
高出力	1560	2000	µA	

下の表に別途記されていない場合、全ての限界値は T_A = 25°C 及び Vdd = 3.3V で保証されています。別途記されていない場合、f_{clock} = 125 kHz、外部 AGND 1.50V、外部 V_{Ref} 0.80V、REFPWR = HIGH, SCPOWER = ON, PSoC ブロック出力 HIGH です。

Table 4. 3.3V MDAC8 DC 及び AC 電気特性

パラメータ	一般的 [1]	限界 [2]	単位	条件および注記
解像度	—	8	ビット	
線形性				
DNL	0.10	0.20	LSB	
INL	0.20	0.45	LSB	
単調な	—	½	Bit（ビット）	
ゲイン誤差	1.0	2.5	%FSR	
V_{OS}, オフセット電圧 [3]	7	31	mV	
出力ノイズ				
帯域限界	0.3	1	mV rms	0 ~ 10 kHz
広帯域	7	10	mV rms	0 ~ 300 kHz
f_{clock}, 内部アップデートレート [4]	—	32 ~ 333	kHz	
V_{in}（帯域幅）	40	—	kHz	
動作電流 [5]				
低出力	200	—	µA	
中出力	500	—	µA	
高出力	1280	1800	µA	

電気特性に関する注意

1. 一般値は、平均値 +1σ です。
2. 限界値は、試験または統計分析により保証されています。

[+] Feedback

3. 2 の補数の 0 は、外部 AGND に対しオフセットしています。また、アナログ出力バッファのオフセットエラーを含みません。
4. φ_1, φ_2 の限界：ブロードバンドノイズにおける 3dB の増加のため
5. すべてのアナログブロックに共通の、基準ブロックの出力は含まれていません。(PSoC ファミリのデータシートを参照)。

タイミング

アンダーラップ位相のクロック φ_1 及び φ_2 の、タイミング信号と適切なデューティ比を生成するため、アナログコラムクロック回路は二つの divide-by-2 回路を使用します。このため、MDAC8 に用いられるクロックソースは、求められる最大アップデートレートの少なくとも 4 倍速い速度で駆動する必要があります。出力のグリッチを無くすため、内部でサンプルアンドホールド信号が用いられるうえ、"Ready" 信号が生成されます。"Ready" は、コントロールレジスタがセットアップ時間の要求条件に違反せずに 0 を書きこむことができることを意味します。下のアップデート•タイミングダイアグラムを参照してください。

同じアップデートサイクルで、二つの書き込みに対するセットアップ時間の要求条件が満たされず、両方のレジスタへの書き込みに失敗した場合、そのサイクルでは不適切な出力値が生成されます。例えば、φ_2 のオンセットの前に二つの書き込みが行われた場合、出力は前回の出力電圧と望ましい出力電圧の間の値になります。φ_2 が高い状態で 1 つまたは 2 つの書き込みが行われた場合、出力は前回の出力電圧と望ましい出力電圧の間にない値を取ることがあります。割り込みのような非決定性のイベントにより引き起こされる可能性のある、このような逸脱を最小化するために、最初に MSB レジスタに書き込んでください。

多くのアプリケーションでは、上記のような瞬間的な (一つのアップデートサイクルに限る) 逸脱は問題ありません。それ以外のアプリケーションでは、より厳しい要求条件が求められます。一つの例は、低ひずみ波形発生器です。このような逸脱は、例えばレジスタをアップデートするタイミングの調節方法の一つであるハードウェア同期化によって防ぐことができます。ハードウェア同期化は、API では "Stall" で終わるエントリーポイントでサポートされます。

ハードウェーア同期化は、出力値のレジスタへの最速のアクセスを保証します。ASY_CR レジスタに書き込むことで、ハードウェーア同期化がトリガされます。ACLKi が有効な場合、IO スペースへの次の書き込みがすぐ処理されます。無効な場合は、ACLKi が確認されるまで、IO 書き込みの最中に CPU クロックがストールします。ストールにより失われる CPU サイクル最小化させる一つの方法は、求められるアップデータレートを大きく上回ってもいいので、アップデートクロックを高い周波数 (ただし f_{MAX} 以下) で駆動させることです。高速アップデートされるクロックによる強制同期化は、レベル "A" のセットアップ時間の損失、すなわち CPU クロックにおける最悪のケースのストールを引き起こします。

Figure 6. アップデートタイミング

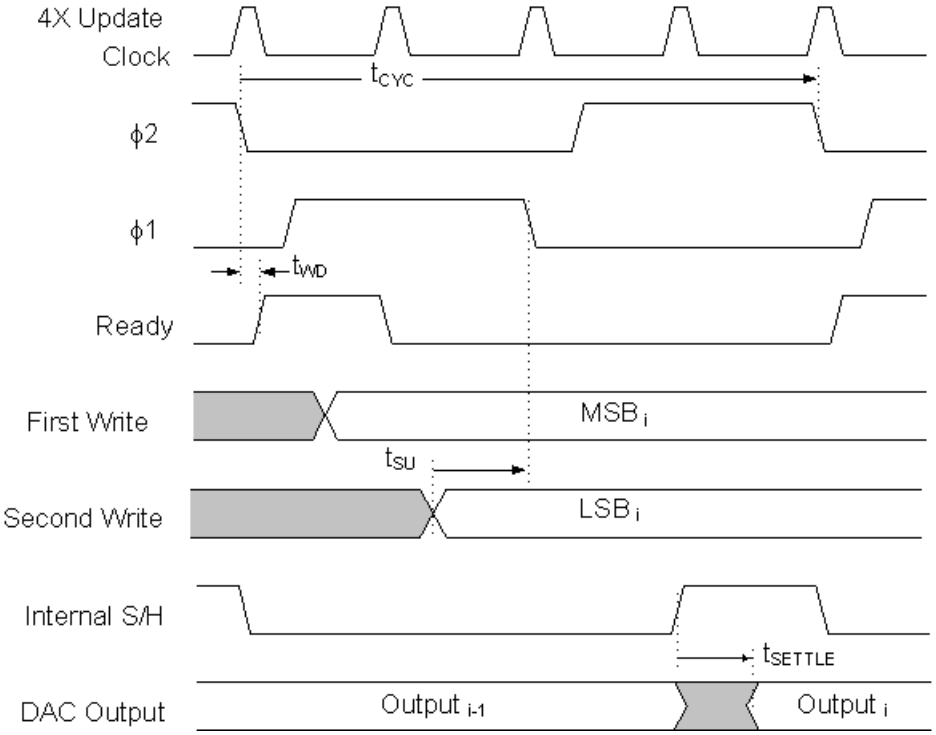
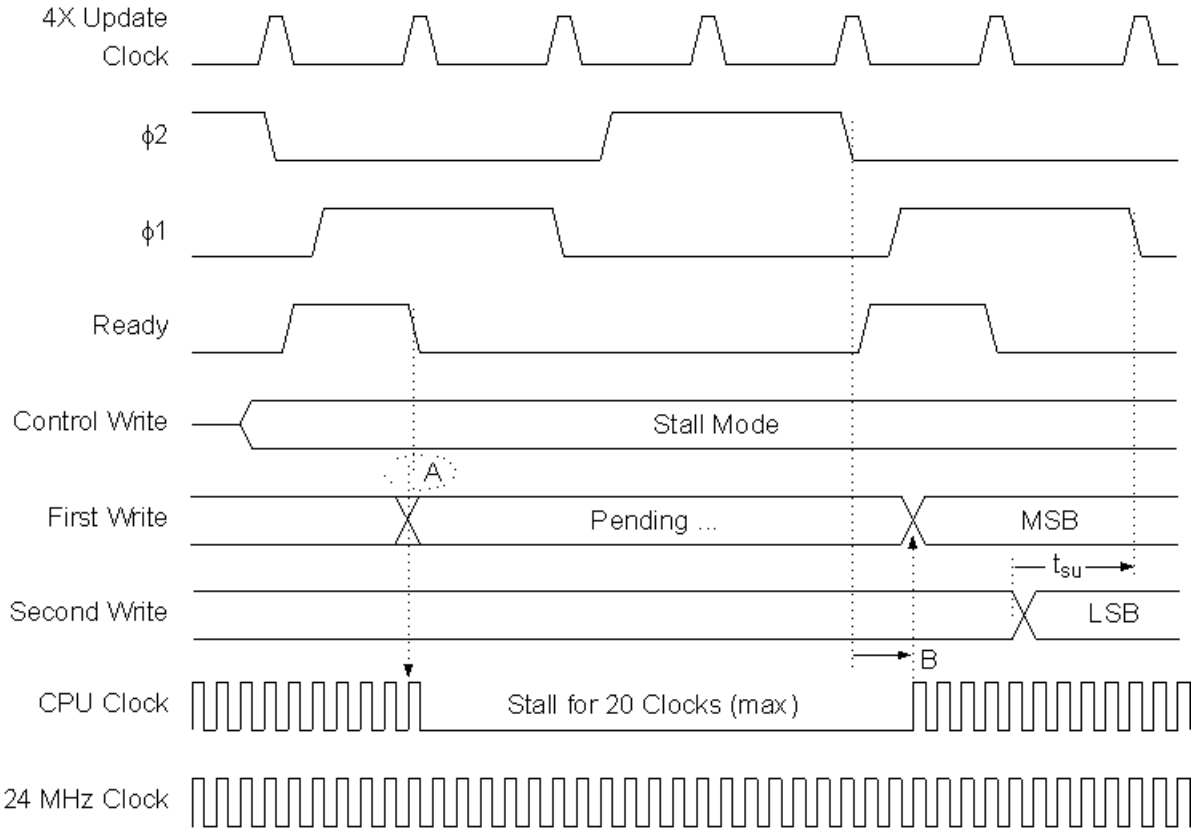


Figure 7. 高速アップデートされるクロックによる強制同期化



CPU ストールの間、すべてのアナログ及びデジタル PSoC ブロックは正常に機能します。DAC CR0 レジスタを書き込むという、最初の MOV インストラクションは単に延期され、この間は全ての割り込みはペンディングになる、もしくはペンディング状態のまま保持されます。ストールから解放され、ストールされた書き込みが終了すると、有効ならばペンディングの状態の割り込み処理されます。割り込みレイテンシの合計値の最大値と クロック φ_1 と φ_2 の周期によって、問題が起きるか、起きないかが決まります。

配置

MDAC8 ユーザ モジュールは、LSB と MSB という 2 つの PSoC ブロックにマッピングされます。LSB ブロックの出力は MSB ブロックの入力になります。そのため、これらのブロックは隣り合わせに配置されます。CY8C26/25xxx デバイスファミリでは、MSB ブロックは “ タイプ A" スイッチ キャパシタの PSoC ブロックにのみマッピングされます。CY8C27/24/22xxx デバイスファミリでは、MSB ブロックは “ タイプ C" スイッチ キャパシタの PSoC ブロックにのみマッピングされます。これにより線形誤差が軽減されます。その理由は、これらのタイプのブロックが、LSB ブロックと MSB ブロックを接続する “BCap" キャパシタ (上図の C_4) の、オフセット誤差を打ち消す自動ゼロ化プロセスを可能にするからです。

配置場所の選定でもう一つ重要なこととして、MSB クロックと LSB クロックを同じソースから生成されなければなりません。両クロックがアナログアレイの同じコラムに配置されていれば、これは自動的に満たされます。別々のコラムに配置されている場合は、両コラムのマルチプレクサを同じソースに設定しなければなりません。いくつかの MDAC8 配置では、MSB 及び LSB ブロックの両方が同一ソースを選択できません。

Note MSB 及び LSB ブロックについて、同じ入力電圧ソースを選択してください。

パラメータとリソース

作動するデジタル - アナログコンバーターを生成するためには、デバイスエディタのユーザー選択モジュールで、MDAC8 ユーザーモジュールのインスタンスを生成してください。次に、MSB 及び LSB ブロックを、アナログアレイ上のスイッチ コンデンサの PSoC ブロックにマッピングしてください。さらに、φ_1 及び φ_2 を駆動するのに適切なアップデートクロックリソースの構成、データフォーマットの指定、GAIN 範囲の選択、入力電圧ソースの選択及び PSoC ブロックと関連した出力バスの割り込みなどを行ってください。

アップデートクロックリソースの構成には、3 ステップあります。最初に、アナログコラムクロックジェネレータのクロックソースを構成します。コラムクロックジェネレータは、入力を 4 つに分けて φ_1 と φ_2 を生成します。このため、ソースは求められるアナログ出力アップデート率の 4 倍早く駆動する必要があります。タイミングの節に、アップデートクロック周波数の選択に関連している項目が記載されています。クロックソースの選択肢には、V1 及び V2 ディバイダー、およびデジタル PSoC ブロック全てが含まれます。外部ソースを使用する、もしくは V1 及び V2 を他の目的に使用しなければならない場合は、レート生成にタイマー、カウンタ及びパルス幅変調 (PWM) ユーザーモジュールを使用することができます。φ_1 と φ_2 のアクティブ周期のデッドバンド時間は、24 MHz システムクロックと同期して発生します。従って、システムクロックと同期していない外部クロックソースの使用は、予測できない結果を起こす可能性があります。

次に、デバイスエディタで CLKmux を設定し、クロックソースをコラムクロックジェネレータに接続してください。デジタル PSoC ブロック出力は、ACLK0 あるいは ACLK1 マルチプレクサを通して接続しなければなりません。追加の情報については *PSoC Designer を参照してください。統合開発環境ユーザーガイド及び PSoC データシート*。

最後に、Normal(デフォルト値) あるいは Swapped を選択して、MDAC8 ユーザーモジュールのパラメータである ClockPhase の値を選択してください。これは MDAC8 出力を、他の PSoC ブロック入力に

同期化させることができます。スイッチコンデンサのアナログ PSoC ブロックは、φ_1 と φ_2 を通して信号を取得し伝送します。MSB ブロックの出力は φ_2 の間のみが有効なので、φ_1 の間に入力サンプリングする他のユーザーモジュールに接続した場合に、問題が発生します。ClockPhase パラメータを Swapped に設定する場合、MSB 及び LSB ブロック内部の φ_1 と φ_2 の役割が交換されますし、それによって出力はダウンストリームユーザーモジュールが入力信号をサンプリングする時に有効になります。(なお、Normal モードでは、入力電圧は φ_1 の間にサンプリングされます。)

Note MSB 及び LSB には、同じ入力電圧ソースを選択してください。

InputMSB

MSB ブロック電圧ソースを入力してください。入力電圧ソースとして REFHI を選択する場合、MDAC8 は DAC8 と同じように作動します。他の入力電圧を選択する場合、選択されたブロック内で適切なユーザーモジュールを構成する必要があります。いくつかの MDAC8 配置では、MSB 及び LSB ブロックの両方で同一ソースを選択することができません。

Note MSB 及び LSB について、同じ入力電圧ソースを選択してください。

InputLSB

LSB ブロック電圧ソースを入力してください。入力電圧ソースとして REFHI を選択する場合、MDAC8 は DAC8 と同じように作動します。他の入力電圧を選択する場合、選択されたブロック内で適切なユーザーモジュールを構成する必要があります。いくつかの MDAC8 配置では、MSB 及び LSB ブロックの両方で同一ソースを選択することができません。

Note MSB 及び LSB について、同じ入力電圧ソースを選択してください。

アナログバス

MDAC8 ブロックの出力は、ローカルに相互接続されたアナログ PSoC ブロック アレイのネットワークやアナログ出力バスを通すことができます。MDAC8 ユーザ モジュールの アナログバス パラメータをデフォルト値の Disable (無効) に設定すると、ローカル ネットワークへの接続数が限定されます。Enable(有効) を選択する場合、ピンを駆動する、もしくはローカルネットワークだけでは不可能な、いくつかの追加入力マルティプレクサへ接続できる関連アナログ出力バッファに接続できる経路の数が増えます。

各スイッチコンデンサ PSoC ブロックは、φ_2 のバス有効信号をサンプリングする回路を含みます。これにより、オートゼロ作動中に発生する電圧スウィングが除去できます。

Note アナログバスを Enabled に設定し、クロック位相を Swapped に設定すると、サンプルアンドホルド関数が無効になります。この場合、バス出力はローカル PSoC ブロック出力と同じになり、φ_1 の間の AGND(+ オフセット電圧) と φ_2 の間の求められる値を交互に出力します。

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 キャパシタのアナログ PSoC ブロックは、2 相クロック (φ_1、φ_2) を使用して信号を取得し伝送します。クロック位相 パラメータを Swapped に設定する場合、MSB 及び LSB ブロック内部の φ_1 と φ_2 の役割が交換されるため、ダウンストリームユーザーモジュールが入力信号をサンプリングする時に出力が有効になります。(なお、Normal モードで、入力電圧は φ_1 の間にサンプリングされます。)

GainRange

提供された入力コード、入力電圧及び AGND はゲイン範囲を低いところから高いところまで変更する場合、出力電圧が適切に増加します。なお、高いゲイン範囲における入力電圧範囲は、低いゲイン範囲での入力電圧範囲の約半分になります。

DataFormat（データフォーマット）

MDAC8 ユーザーモジュールの API は、オフセットバイナリ、2 の補数、そしてと符号 / 絶対値の 3 種類の異なるデータ形式が使えます。このユーザーモジュールの API セクションの WriteBlind エントリーポイントは、これらの慣行や関連する値の範囲が説明されています。

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、ユーザー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関連する定数を示します。

Note ここでは、全てのユーザーモジュール API と同じように、API 機能を呼び出すことによって、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であれば、呼び出し元関数にて 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。アセンブリ言語のプログラマは、コードがこのポリシーを守っていることを確認しなければなりません。一部のユーザ ーモジュール API 機能では 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。

エントリーポイントは、MDAC8 ユーザーモジュールを初期化し、数値を更新し、ユーザーモジュールを無効にするために提供されています。

MDAC8_Start

説明：

このユーザー モジュールにおける必要な全ての初期化を行い、スイッチ 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  MDAC8_Start(BYTE bPowerSetting)
```

アセンブラ：

```
mov   A, bPowerSetting
lcall  MDAC8_Start
```

パラメータ：

bPowerSetting: 出力レベルを指定する 1 バイト。再設定や構成後、MSB や LSB ブロックに指定された PSoC ブロックの電源が切られます。C 及びアセンブリで提供された、シンボリックネームと関連する値が、次の表に記載されています。

シンボリックネーム	値
MDAC8_OFF	0
MDAC8_LOWPOWER	1
MDAC8_MEDPOWER	2
MDAC8_FULLPOWER	3

戻り値：

なし

副作用：

MDAC 出力デバイスを駆動します。デフォルトの初期値は AGND です。電源を入れた時に他の出力値が必要な場合に、" スタート " を起動する前に、いずれかの書き込みルーチンを呼び出します。A 及び X レジスタは、この機能によって変更される可能性があります。

MDAC8_SetPower

説明：

DAC スイッチコンデンサ ブロックの PSoC ブロックの電源レベルを設定します。ブロックを無効化したり有効化したりするのに使用します。

C プロトタイプ：

```
void  MDAC8_SetPower(BYTE bPowerSetting)
```

アセンブラ：

```
mov   A, bPowerSetting
lcall  MDAC8_SetPower
```

パラメータ：

bPowerSetting: スタート エントリーポイントで使用する、PowerSetting パラメータと同じです。

戻り値：

なし

副作用：

MDAC 出力デバイスを駆動します。デフォルトの初期値は AGND です。電源を入れた時に他の出力値が必要な場合に、" スタート " を起動する前に、いずれかの書き込みルーチンを呼び出します。A 及び X レジスタが、この機能によって変更される可能性があります。

MDAC8_SetOutputRange

説明：

FCap を 32(低範囲 : ゲイン =1) あるいは 16(高範囲 : ゲイン =2) に設定してコンデンサ PSoC ブロックに変換された 2 つ MDAC で 1 つを選択します。

C プロトタイプ：

```
void  MDAC8_SetOutputRange(BYTE bRangeSetting)
```

アセンブラ：

```
mov   A, bRangeSetting
lcall  MDAC8_SetOutputRange
```

パラメータ：

RangeSetting: 範囲を設定する 1 バイト。

シンボリック名	値
MDAC8_LOWRANGE	0
MDAC8_HIGHRANGE	1

戻り値：

なし

副作用：

A 及び X レジスタが、この機能によって変更される可能性があります。

MDAC8_SetPhase

説明：

内部 φ1 及び φ2 クロックを、スワップあるいは通常 (デフォルト) のいずれかに設定します。

C プロトタイプ：

```
void  MDAC8_SetPhase(BYTE bPhaseSetting)
```

アセンブラ：

```
mov   A, bPhaseSetting
lcall  MDAC8_SetPhase
```

パラメータ：

bPhaseSetting: 通常と、スワップ位相のいずれかを設定する、1 バイト。

シンボリック名	値
MDAC8_NORMALPHASE	0
MDAC8_SWAPPEDPHASE	1

戻り値：

なし

副作用：

A 及び X レジスタが、この機能によって変更される可能性があります。

MDAC8_WriteBlind

説明：

出力電圧を、指示された値に直ちに更新します。

C プロトタイプ：

```
// For OffsetBinary: (BYTE = unsigned char)
void  MDAC8_WriteBlind(BYTE bOutputValue)
// For TwosComplement: (CHAR = signed char)
void  MDAC8_WriteBlind(CHAR cOutputValue)
// For TwoByteSignAndMagnitude: (BYTE of bit flags)
void  MDAC8_WriteBlind2B(BYTE bLSB, BYTE bMSB)
```

アセンブラ：

```
; for OffsetBinary
mov   A, bOutputValue
lcall  MDAC8_WriteBlind
; for TwosComplement
mov   A, cOutputValue
lcall  MDAC8_WriteBlind
; for TwoByteSignAndMagnitude format:
mov   A, bLSB
mov   X, bMSB
lcall  MDAC8_WriteBlind2B
```

パラメータ：

b/cOutputValue: 出力電圧を指定する 1 バイト。次の表に示すように、許可される値の範囲は、選択された DataFormat の値と同じです。

データフォーマット	最小値	最大値
オフセットバイナリ	0	254
2 の補数	-127	127
2 バイト符号 / 絶対値	3F18h	1F38h

オフセットバイナリ値は、最小の出力電圧を意味する 0 から最大値 254 までの正の数です。2 の補数は、M8C プロセッサの、ネイティブな符号付きの形式です。2 バイト符号 / 絶対値形式では、高いバイトは $00smmmmm_2$、低いバイトは $00tmm000_2$ の形式です。's' は符号、't' は反転させた符号、'm' は絶対値ビットです。たとえば、正の数の場合、s=0 及び t=1 です。

戻り値：

なし

副作用：

このユーザーモジュールのタイミングの節で説明している理由により、出力デバイスに問題が発生する可能性があります。A 及び X レジスタが、この機能によって変更される可能性があります。

MDAC8_WriteStall

説明：

Phi1 が起動されるまでマイクロプロセッサーをストールし、その後出力電圧を指定された値にアップデートします。なお、API は、割り込みが無効、もしくは最大の割り込みレイテンシが ACLKi 以下ということを想定しています。(図 「高速アップデートクロックを用いた強制同期化」を参照するとのこと)。

C プロトタイプ：

```
// For OffsetBinary: (BYTE = unsigned char)
void  MDAC8_WriteStall(BYTE bOutputValue)
// For TwosComplement: (CHAR = signed char)
void  MDAC8_WriteStall(CHAR cOutputValue)
// For TwoByteSignAndMagnitude: (BYTE of bit flags)
void  MDAC8_WriteStall2B(BYTE bLSB, BYTE bMSB)
```

アセンブラ：

```
; for OffsetBinary
mov   A, bOutputValue
lcall  MDAC8_WriteStall
; for TwosComplement
mov   A, cOutputValue
lcall  MDAC8_WriteStall
; for TwoByteSignAndMagnitude format:
mov   A, bLSB
mov   X, bMSB
lcall  MDAC8_WriteStall2B
```

パラメータ：

b/cOutputValue: 形式と値の範囲は WriteBlind エントリーポイントのパラメータと同一。bMSB 及び bLSB: 形式と値の範囲は WriteBlind エントリーポイントのパラメータと同一。

戻り値：

なし

副作用：

ACLKi が無効の場合 ('i' は、アナログ PSoC ブロックがマップされるコラム)、φ_2 が無効になるまで、すなわち、おそらくアップデートサイクルの 3/4 の間 (+ 2 CPU クロック)、マイクロプロセッサーの CPU クロックは無効になります。ストール中は、割り込みは認識されません。A 及び X レジスタは、この機能によって変更される可能性があります。

MDAC8_Stop

説明：

ユーザーモジュールの電源を遮断します。

C プロトタイプ：

```
void  MDAC8_Stop(void)
```

アセンブラ：

```
lcall  MDAC8_Stop
```

パラメータ：

なし

戻り値：

なし

副作用：

出力が駆動されません。A 及び X レジスタは、この機能によって変更される可能性があります。

ファームウェア ソースコードの例

サンプルコードは、周期的で、ゆっくり減少するのこぎり波を発生させます。

```
;;-------------------------------------------------------------------
;; Sample Code for the MDAC8
;; Generate a falling sawtooth wave
;;-------------------------------------------------------------------

export _main
include "m8c.inc"
include "MDAC8.inc"

      area bss (RAM)
bVal: blk  1                ; RAM for loop iteration variable
bMAXVAL: equ  255           ; Top of ramp plus 1
      area text (ROM, REL)

_main:                      ; (contains infinite loop; never returns)
      mov  A, MDAC8_LOWPOWER ; specify DAC's amplify power
      call MDAC8_Start      ; and turn it on.
Init:
      mov  [bVal], bMAXVAL  ; Start ramp from the top
RampDown:
      mov  A, [bVal]        ;
      dec  A
      call MDAC8_WriteStall
      dec  [bVal]           ; Bottom of ramp?
      jnz  RampDown         ; No, not yet.
      jmp  Init             ; Yes, re-initialize ramp and loop
                            ; forever


//-------------------------------------------------------------------
// C main line
//-------------------------------------------------------------------

#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules

    BYTE cVal;
    #define cMax 255


void main(void)
{
    // Insert your main routine code here.
    MDAC8_Start(MDAC8_LOWPOWER);
    while(1) //infinite loop
    {
        cVal = cMax;
        while(cVal > 0)
        {
            MDAC8_WriteStall(cVal--);
        }
    }
```

```
}
```

設定レジスタ

API は、MDAC8 ユーザーモジュールの完全なインターフェースを提供します。設定レジスタに直接書き込むことで、出力をアップデートすることができます。いずれにせよ、出力デバイスの問題を防ぐため、タイミングについての配慮が必要です。次のレジスタが、MDAC8 変換されたコンデンサ LSB 及び MSB ブロックで使用されます。

Table 5. ブロック LSB：レジスタ CR0

Bit（ビット）	7	6	5	4	3	2	1	0
値	1	0	符号	絶対値		0	0	0

符号は、正の値に 「1」(AGND ~ RefHi)、負の値に 「0」(RefLow ~ AGND) を使用します。デフォルトは 「1」です。これは MSB ブロックで使用されている定義の逆です。API の書き込み関数を使用して、符号を変更します。絶対値のデフォルトは 「0」です。API の書き込み関数を使用して、値を変更します。

Table 6. ブロック LSB：レジスタ CR1

スイッチド キャパシタ タイプ A								
Bit（ビット）	7	6	5	4	3	2	1	0
値	0	1	0	0	0	0	0	0
スイッチド キャパシタ タイプ B								
Bit（ビット）	7	6	5	4	3	2	1	0
値	1	0	0	0	0	0	0	0

Table 7. ブロック LSB：レジスタ CR2

Bit（ビット）	7	6	5	4	3	2	1	0
値	アナログバス	0	1	0	0	0	0	0

アナログバスが無効になります。

[+] Feedback

Table 8.　ブロック LSB：レジスタ CR3

スイッチド キャパシタ タイプ A								
Bit（ビット）	7	6	5	4	3	2	1	0
値	0	0	1	1	0	0	Power (出力)	
スイッチド キャパシタ タイプ B								
Bit（ビット）	7	6	5	4	3	2	1	0
値	0	0	1	1	1	0	Power (出力)	

Power (出力) デフォルトは Off です。この値を設定するには、API の 「Start call」を使ってください。

Table 9.　ブロック MSB：レジスタ CR0

Bit（ビット）	7	6	5	4	3	2	1	0
値	1	0	符号	絶対値				

符号は API 書き込みルーチンによって設定されます。デフォルトは ’0’ です。. 絶対値は、API で書き込み関数のいずれかを使用することによって変更できます。デフォルトは ’0’ です。.

Table 10.　ブロック MSB：レジスタ CR1

Bit（ビット）	7	6	5	4	3	2	1	0
値	0	1	0	0	0	0	0	1

Table 11.　ブロック MSB：レジスタ CR2

Bit（ビット）	7	6	5	4	3	2	1	0
値	アナログバス	0	1	0	0	0	0	0

アナログバスは、デバイス エディターの構成時間によって有効化または無効化されます。

Table 12.　ブロック MSB：レジスタ CR3

Bit（ビット）	7	6	5	4	3	2	1	0
値	0	0	1	1	BMux		Power (出力)	

BMux を構成し、LSB PSoC ブロックから接続を選択します。Power (出力) 0=OFF(デフォルト)、1= 低、2= 中、3= 全力です。この値を設定するには、API の 「Start」コールを使ってください。

Copyright © 2001-2011 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.