8-Bit 逐次比較型　ADC データシート SAR8 V 1.0

Copyright © 2005-2011　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（各外部 I/O およびクロックにつき）
	デジタル	アナログ CT	アナログ SC	フラッシュ	RAM	
CY8C24x33, CY8C23x33	0	0	0	147	0	1

その他のコンバータについては、アプリケーション ノート「アナログ - ADC の選択」AN2239 を参照してください。

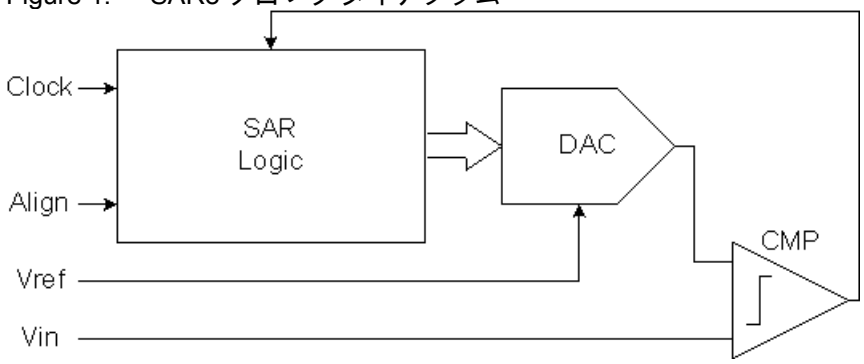
このユーザ モジュールを使用する単一あるいはすべてを設定するサンプルプログラムについては、www.cypress.com/psocexampleprojects を参照してください。

特性および概要

- CY8C24x33 および CY8C23x33 デバイスで、アナログ→デジタル変換の最高性能を実現
- 変換時間が短い
- 逐次比較機能をサポート
- 8-bit 分解能
- 8 つの外部信号と 2 つの CT ブロックからの内部信号を接続可能
- 1 回で変換
- フリーラン変換
- 選択可能な変換トリガ
- プログラム可能なサンプル時間
- プログラマブル クロック分周器
- 変換のキャンセル / リスタート機能
- PWM、タイマ、カウンタ周期の任意のタイミングにおけるオートライン / トリガをサポート
- 単一変換モードで変換完了後は、自動的に低消費電力モードへ移行

SAR8 ユーザ モジュールは、専用アナログ PSoC ブロックを使用して、入力電圧をデジタルコードに変換する 8-bit 逐次比較型（SAR）ADC コンバータです。変換時間は標準的な 2.7 µs（ファームウェア処理による制限付き）で、各サンプルで 8-bit の未割り当て値を作ります。モータ制御アプリケーションでは、最高の性能を得るために、PWM 期間中の一定のタイミングにアナログ→デジタル変換を実行しなければならない場合もあります。

Figure 1. SAR8 ブロック ダイアグラム



機能説明

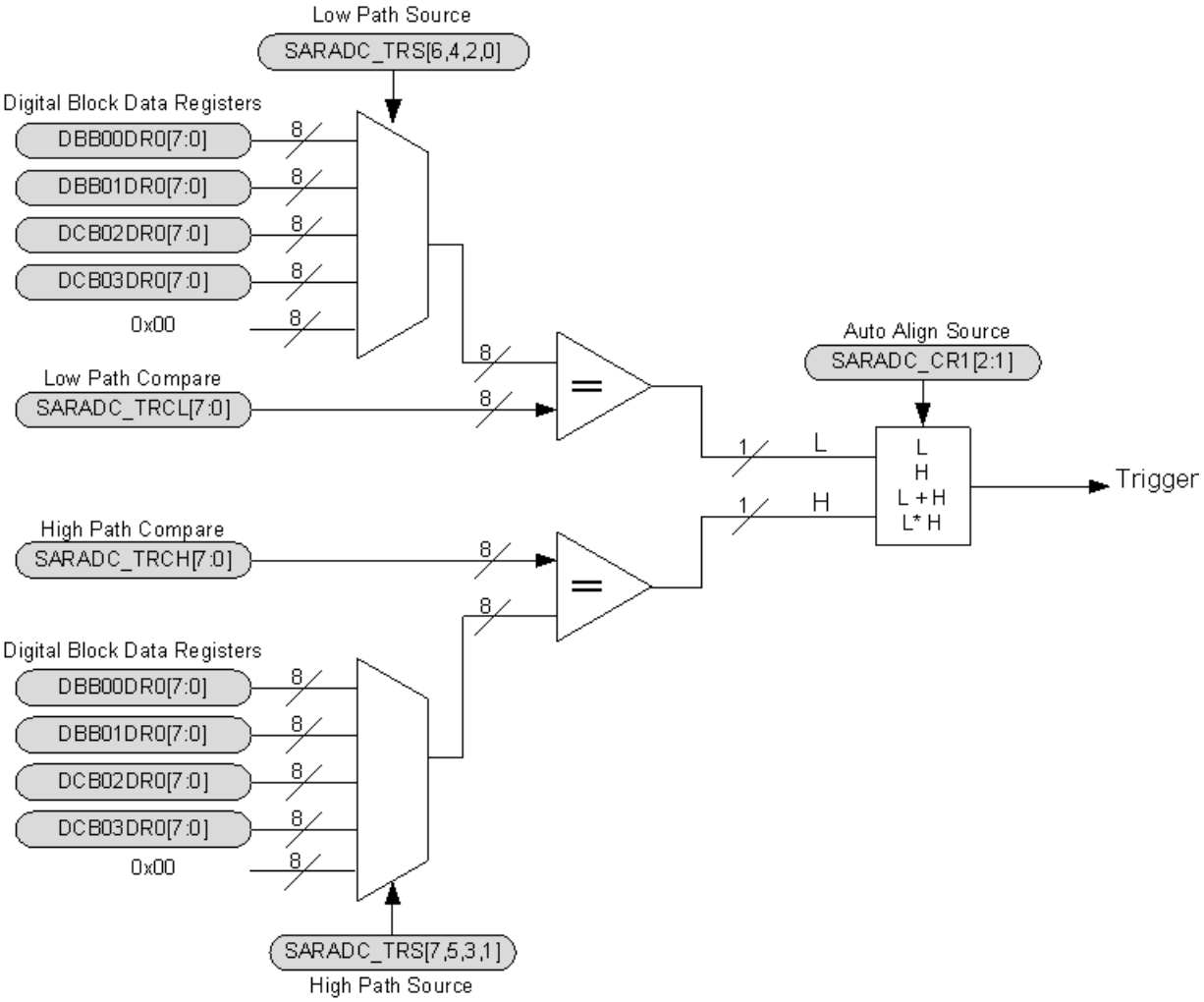
逐次比較レジスタは PSoC SARADC_DL レジスタです。このレジスタは、変換の結果を保存します。SAR 回路は、リファレンス電圧をスケーリングし、入力電圧と比較することにより、一連の比較を実行します。二分探索を行い、入力電圧にもっとも近い DAC 設定を見つけます。

SAR8 の独自機能には、1 つまたは 2 つのデジタルブロックと同期した変換を可能にする高度な変換トリガ回路があります。

SAR8 ADC は、ワンショットモードまたはフリーランモードで作動します。いずれの場合も、API 関数では、ADC が変換の実行中でも、新しい測定が開始できます。ワンショットモードでは、ADC は変換後自動的に低消費電力モードへ移行します。フリーランモードでは、ADC は動作が止まったときに低消費電力モードへ移行します。

オートアラインオプションを利用すると、2 つの 8-bit デジタルコンパレータ（LOW と HIGH）を用いて変換要求をトリガできます。これらのコンパレータは、任意の PSoC デバイスのデジタルブロックで制御できます。コンパレータは、ブロックの DR0 レジスタの実際の値を、SAR8_COMPARE_LO_REG と SAR8_COMPARE_HI_REG レジスタに保存されているデータと比較します。各コンパレータは、独立して ADC をトリガすることもでき、結合して 16-bit トリガソースを構成することもできます。

Figure 2. SAR8 オートアラインの回路図



SAR8 ADC のもう 1 つの機能には、データケーリングがあります。この機能を使用すると、右シフト済みの値をレジスタから読み取ることが出来るようになります。スケールファクタは 1 ~ 2^{-N} の間で調節可能です（N=0..6）。

[+] Feedback

DC 電気的特性と AC 電気的特性

以下の表に、電圧範囲および温度範囲で保証されている最大値と最小値の仕様を示します。4.75V ~ 5.25V および -40°C、TA、85°C、または 3.0V ~ 3.6V および -40°C、TA、85°C (それぞれ)。標準のパラメータは、温度 25°C、電圧 5V および 3.3V の場合の値です。これはデザインの参考データにすぎません。

Table 1. SAR8 DC および AC の電気的特性

パラメータ	最小値	典型値	最大値	単位	条件および注記
$V_{ADCVREF}$	3.0[a]	-	Vdd	V	ADC リファレンス電圧として構成された場合のピン P3[0] におけるリファレンス電圧。 $V_{ADCVREF} \leq$ Vdd.
$I_{ADCVREF}$	-	-	3	mA	P3[0] が ADC V_{REF} として構成されているときに SAR8 ADC に供給される電流
C_{IN}	-	-	60	pF	ADC 入力の入力容量
R_{IN}	-	-	1	kΩ	ADC 入力の入力抵抗
INL[b]	-1.2	-	+1.2	LSB	R-2R 積分非直線性。 最大 LSB はフルスケール範囲の 1/16 を超えないサブレンジ上。
DNL[b]	-1	-	+1	LSB	R-2R 微分非直線性。出力は単調増加
周波数	0.375	-	24	MHz	入力クロック周波数

a. 低い $V_{ADCVREF}$ 電圧は、精度が低下しますが、使用することができます。

b. 周波数が 1.5 MHz 以下の ADC クロックに適用。

配置

SAR8 をサポートする専用リソースが 1 つあり、これが配置に関する唯一の選択肢です。

パラメータおよびリソース

ADCInput

ADC の信号ソースには、任意のポート 0 ピンを選択することができます。アナログブロックの ACB00 と ACB01 も、信号ソースとして使用できます。

ADCClock

クロック速度は以下から選択できます。

- SYSCLK
- SYSCLK/2
- SYSCLK/4
- SYSCLK/8
- SYSCLK/16
- SYSCLK/32
- SYSCLK/64

ADC サンプリング速度は、ADC クロック ÷ 8 で、変換時間は ADC クロック周期 x 8 です。

ADC は変換後自動的に低消費電力モードに入るため、ワンショットモードでは、低速の ADC クロックを使用すると、高速 ADC クロックより電力を多く消費します。フリーランモードでは、低速の ADC クロックを使用すると、高速 ADC クロックより少ない電力を消費します。

Scale

変換の結果は、自動的に、読み出したときのスケールファクタで割った整数となります。スケールになり得る数値は次のとおりです。1、2、4、8、16、32、64。

Run

このパラメータには、ワンショットまたはフリーラン変換モードを選択します。

ADCPower

SAR8 ブロックは、内部 VPWR (Vdd) または P3[0] から電力を得ることができます。

VPWRADC コンパレータブロックは、内部 VPWR (Vdd) から電源を得ます。**P3[0]**ADC コンパレータブロックは、P3[0] から電源を得ます。P3[0] は VPWR (Vdd) 以下でなければなりません。

R2RP Power

2 つの選択肢があります。

VPWRADC DAC リファレンス生成ブロックは、内部 VPWR (Vdd) から電源を得ます。ADC DAC 出力 (R2RPPower) は、ADC コンパレータブロックの出力 (ADCPower) 以下でなければなりません。**P3[0]**ADC R2R リファレンス生成ブロックは、P3[0] から電源を得ます。P3[0] は VPWR 以下でなければなりません。

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、全てのユーザー モジュールの API では、A と X レジスタの値は、API 関数を呼び出すことによって変更することができます。API をコールした後でも A と X の値を保持したいときは、API をコールするファンクションで A と X の値を保持する必要があります。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。一部のユーザーモジュール 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。

SAR8 ユーザ モジュールを初期化し、変換を実行して読み出し、SAR8 機能を無効にするためのエントリポイントが用意されます。

SAR8_Start

説明 :

ADC 動作を有効にします。SAR8 は、変換終了後すぐ自動的に低消費電力モードに入ります。Run パラメータがフリーランに設定されている場合、SAR8_Start が呼び出されたあと ADC はデータの収集を開始します。Run パラメータがワンショットに設定されている場合、変換を開始するためには SAR8_Trigger 関数がコールされるか、オートアライン回路からのトリガ信号が発生する必要があります。

C プロトタイプ :

```
void  SAR8_Start(void)
```

アセンブラ :

```
lcall  SAR8_Start
```

パラメータ :

なし

戻り値 :

なし

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

SAR8_Stop

説明 :

ADC 動作を無効にします。Stop が呼び出されたあと、ADC は低消費電力モードに入ります。

C プロトタイプ :

```
void  SAR8_Stop(void)
```

アセンブラ :

```
lcall  SAR8_Stop
```

パラメータ :

なし

戻り値 :

なし

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

SAR8_EnableInt

説明 :

割り込みを有効にします。M8C_EnableGInt マクロを使用してすべての割り込みが有効にならなければ、SAR8 割り込みは使用できません。

C プロトタイプ :

```
void  SAR8_EnableInt(void)
```

アセンブラ :

```
lcall  SAR8_EnableInt
```

パラメータ :

なし

戻り値 :

なし

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

SAR8_DisableInt

説明 :

割り込みを無効にします。

C プロトタイプ :

```
void SAR8_DisableInt(void)
```

アセンブラ :

```
lcall SAR8_DisableInt
```

パラメータ :

なし

戻り値 :

なし

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

SAR8_Trigger

説明 :

ADC をトリガして、次のシステムクロック周期からサンプルと変換を実行します。ADC がすでにサンプルの変換中である場合、SAR8_Trigger は強制的に ADC が進行中の変換をキャンセルさせ、次のシステムクロック周期で新しい変換をリスタートします。ADC がフリーランモードで稼動している場合、この関数は ADC サンプリングに影響を与えません。

C プロトタイプ :

```
void  SAR8_Trigger(void)
```

アセンブラ :

```
lcall  SAR8_Trigger
```

パラメータ :

なし

戻り値 :

なし

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

SAR8_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  SAR8_fIsDataAvailable(void)
```

アセンブラ :

```
lcall  SAR8_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、ゼロ以外の値を返します。

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

SAR8_bGetData

説明 :

変換したデータを符号なしバイト値として返します。データ サンプルが取得できるかどうか確認するには、SAR8_fIsDataAvailable() を呼び出してください。

C プロトタイプ :

```
BYTE  SAR8_bGetData(void)
```

アセンブラ :

```
lcall  SAR8_bGetData
```

パラメータ :

なし

戻り値 :

変換したデータを符号なしバイト値として返します。

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

SAR8_SelectADCChannel

説明 :

10 個の入力チャネルから ADC 入力を選択します。

C プロトタイプ :

```
void  SAR8_SelectADCChannel(BYTE bChannel)
```

アセンブラ :

```
mov  A, bChannel
lcall  SAR8_SelectADCChannel
```

パラメータ :

bChannel: ADC 入力チャネルを指定する 1 バイト。C およびアセンブリで用意されたシンボル名、およびそれらに関連付けられた値は、以下の表で示されます。

記号名	値	チャネル
SAR8_P0_0	0x00	P0[0]
SAR8_P0_1	0x08	P0[1]
SAR8_P0_2	0x10	P0[2]
SAR8_P0_3	0x18	P0[3]
SAR8_P0_4	0x20	P0[4]
SAR8_P0_5	0x28	P0[5]
SAR8_P0_6	0x30	P0[6]
SAR8_P0_7	0x38	P0[7]
SAR8_ACB00	0x40	ACB00
SAR8_ACB01	0x48	ACB01

戻り値 :

なし

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

SAR8_AutoAlign

説明 :

選択されたデジタルブロックを使用した ADC サンプリングを有効・無効にします。

C プロトタイプ :

```
void  SAR8_AutoAlign(BYTE bAlignMode)
```

アセンブラ :

```
mov  A, bAlignMode
lcall  SAR8_AutoAlign
```

パラメータ :

bAlignMode: アラインモードを指定する 1 バイト。

記号名	値
SAR8_NOAUTOALIGN	0
SAR8_AUTOALIGN	1

戻り値 :

なし

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

SAR8_SetAlignPath

説明 :

ADC オートアライントリガ回路を選択します。

C プロトタイプ :

```
void  SAR8_SetAlignPath(BYTE bAlignPath)
```

アセンブラ :

```
mov  A, bAlignPath
lcall  SAR8_SetAlignPath
```

パラメータ :

bAlignPath: ADC オートアライントリガ回路を指定する 1 バイト。C およびアセンブリで用意されたシンボル名、およびそれらに関連付けられた値は、以下の表で示されます。

記号名	値	説明 :
SAR8_LowHighIndependent	0x00	Low チャネルと High チャネルは完全に独立しています。いずれも ADC をトリガできます。
SAR8_LowOnly	0x02	Low チャネルだけが ADC をトリガします。
SAR8_HighOnly	0x04	High チャネルだけが ADC をトリガします。
SAR8_LowHighCombined	0x06	High チャネルと Low チャネルの組み合わせで、16-bit トリガソースを構成します。

戻り値 :

なし

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

SAR8_SetHighAlignSrc

説明 :

オートアライントリガ回路の High Path 用のアライメントソースとして、デジタルブロックの 1 つを選択します。

C プロトタイプ :

```
void  SAR8_SetHighAlignSrc(BYTE bAlignSrc)
```

アセンブラ :

```
mov  A, bAlignSrc
lcall  SAR8_SetHighAlignSrc
```

パラメータ :

bAlignSrc: High Path 用のアライメントソースとして、デジタルブロックの 1 つを指定する 1 バイト。C およびアセンブリで用意されたシンボル名、およびそれらに関連付けられた値は、以下の表で示されます。

記号名	値	デジタルブロック
SAR8_None	0x00	接続されていない
SAR8_DBB00	0x01	DBB00
SAR8_DBB01	0x02	DBB01
SAR8_DCB02	0x03	DCB02
SAR8_DCB03	0x04	DCB03

戻り値 :

なし

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

SAR8_SetLowAlignSrc

説明 :

オートアライントリガ回路の Low Path 用のアライメントソースとして、デジタルブロック 1 つを選択することを可能にします。

C プロトタイプ :

```
void  SAR8_SetLowAlignSrc(BYTE bAlignSrc)
```

アセンブラ :

```
mov  A, bAlignSrc
lcall  SAR8_SetLowAlignSrc
```

パラメータ :

bAlignSrc: Low Path 用アライメントソースとしてデジタルブロックの 1 つを指定する 1 バイト。C およびアセンブリで用意されたシンボル名、およびそれらに関連付けられた値は、以下の表で示されます。

記号名	値	デジタルブロック
SAR8_None	0x00	接続されない
SAR8_DBB00	0x01	DBB00
SAR8_DBB01	0x02	DBB01
SAR8_DCB02	0x03	DCB02
SAR8_DCB03	0x04	DCB03

戻り値 :

なし

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

SAR8_SetCmpH

説明 :

オートアラインの High Path 用にコンパレータ値をセットします。

C プロトタイプ :

```
void SAR8_SetCmpH(BYTE bValue)
```

アセンブラ :

```
mov  A, bValue
lcall  SAR8_SetCmpH
```

パラメータ :

bValue: オートアラインの High Path 用のコンパレータデータを指定する 1 バイト。

戻り値 :

なし

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

SAR8_SetCmpL

説明 :

オートアラインの Low Path 用にコンパレータ値をセットします。

C プロトタイプ :

```
void SAR8_SetCmpL(BYTE bValue)
```

アセンブラ :

```
mov  A, bValue
lcall  SAR8_SetCmpL
```

パラメータ :

bValue: オートアラインの Low Path 用のコンパレータデータを指定する 1 バイト。

戻り値 :

なし

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

SAR8_SetClock

説明 :

SAR8 ADC クロックレートを設定します。クロック速度を変更する場合、まず ADC 変換を停止し、クロック速度を変更してから変換をリスタートします。

C プロトタイプ :

```
void SAR8_SetClock(BYTE bClock)
```

アセンブラ :

```
mov  A, bClock
lcall  SAR8_SetClock
```

パラメータ :

bClock: クロックレートを指定する 1 バイト。C およびアセンブリで用意されたシンボル名、およびそれらに関連付けられた値は、以下の表で示されます。

記号名	値	クロックレート
SAR8_SYSCLK	0x00	SysClk
SAR8_SYSCLK_2	0x01	SysClk/2
SAR8_SYSCLK_4	0x02	SysClk/4
SAR8_SYSCLK_8	0x03	SysClk/8
SAR8_SYSCLK_16	0x04	SysClk/16
SAR8_SYSCLK_32	0x05	SysClk/32
SAR8_SYSCLK_64	0x06	SysClk/64

戻り値 :

なし

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

SAR8_SetScale

説明 :

変換結果を読み出すときのスケールファクタを設定します。

C プロトタイプ :

```
void SAR8_SetScale(BYTE bScaleMode)
```

アセンブラ :

```
mov  A, bScaleMode
lcall  SAR8_SetScale
```

パラメータ :

bScaleMode: 右シフト数を指定する 1 バイト。C およびアセンブリで用意されたシンボル名、およびそれらに関連付けられた値は、以下の表で示されます。

記号名	値	スケールファクタ
SAR8_1_1	0x00	1
SAR8_1_2	0x08	1/2
SAR8_1_4	0x10	1/4
SAR8_1_8	0x18	1/8
SAR8_1_16	0x20	1/16
SAR8_1_32	0x28	1/32
SAR8_1_64	0x30	1/64

戻り値 :

なし

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

SAR8_SetRunMode

説明 :

ADC の実行モードを単一変換またはフリーランに設定します。このパラメータはユーザ モジュール Run パラメータでセットします。ランモードを変更する場合を除き、この関数を呼び出す必要はありません。

C プロトタイプ :

```
void SAR8_SetRunMode(BYTE bRunMode)
```

アセンブラ :

```
mov  A, bRunMode
lcall  SAR8_SetRunMode
```

パラメータ :

bRunMode: 変換モードを指定する 1 バイト。

記号名	値
SAR8_OneShot	0x00
SAR8_FreeRun	0x01

戻り値 :

なし

副作用 :

API セクションの冒頭にある注意事項 ** を参照してください。

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 Asm Code for the SAR8
;-------------------------------------------------------------------------

include "m8c.inc"       ; part specific constants and macros
include "memory.inc"    ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules

export _main

_main:
    M8C_EnableGInt
    mov    A, SAR8_P0_0
    call   SAR8_SelectADCChannel
    mov    A, SAR8_FreeRun
    call   SAR8_SetRunMode
    call   SAR8_Start
.loop:
    call   SAR8_fIsDataAvailable
    cmp    A, 0
    jz     .loop
    call   SAR8_bGetData
    mov    [_bResult], A
    ; call User function here
    jmp    .loop
```

C 言語での同じプログラムは次のようになります。

```
//-------------------------------------------------------------------------
// Sample C Code for the SAR8
//-------------------------------------------------------------------------

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

BYTE bResult;

void main(void)
{
  M8C_EnableGInt;

  SAR8_SelectADCChannel(SAR8_P0_0);
  SAR8_SetRunMode(SAR8_FreeRun);
  SAR8_Start();

  while(1) {
    while (0 == SAR8_fIsDataAvailable());  // wait for result
    bResult = SAR8_bGetData();
    // Add user code here
  }
```

}

コンフィグレーション レジスタ

次のレジスタは SAR8 ブロック用に使用されます。

Table 2. ブロック SAR8、レジスタ : SAR8_CONTROL_0_REG (SARADC_CR0; 0, 69h)

ビット	7	6	5	4	3	2	1	0
値	予約	ADC チャネル				データ準備完了	スタート / ビジー	ADCEN

Table 3. ブロック SAR8、レジスタ : SAR8_CONTROL_1_REG (SARADC_CR1; 0, 6Ah)

ビット	7	6	5	4	3	2	1	0
値	ADCPower	R2RPower	予約			Align Source（ソースをアライン）		AutoAlign

Table 4. ブロック SAR8、レジスタ : SAR8_CONTROL_2_REG (SARADC_CR2; 1, ABh)

ビット	7	6	5	4	3	2	1	0
値	0	FreeRun	スケール			クロック		

Table 5. ブロック SAR8、レジスタ : SAR8_TRIGGER_SRC_REG (SARADC_TRS; 1, A8h)

ビット	7	6	5	4	3	2	1	0
値	DCB03		DCB02		DBB01		DBB00	

Table 6. ブロック SAR8、レジスタ : SAR8_COMPARE_LO_REG (SARADC_TRCL; 1, A9h)

ビット	7	6	5	4	3	2	1	0
値	bValue							

Table 7. ブロック SAR8、レジスタ : SAR8_COMPARE_HI_REG (SARADC_TRCH; 1, AAh)

ビット	7	6	5	4	3	2	1	0
値	bValue							

Table 8. ブロック SAR8、レジスタ : SAR8_DATA_LO_REG (SARADC_DL; 0, 67h)

ビット	7	6	5	4	3	2	1	0
値	データ							

バージョン ヒストリー

バージョン	著者	説明 :
1.0	DHA	初期バージョン。

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 © 2005-2011 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.