10-Bit SAR ADC データシート SAR10 V 1.0

Copyright © 2009-2011 Cypress Semiconductor Corporation. All Rights Reserved.

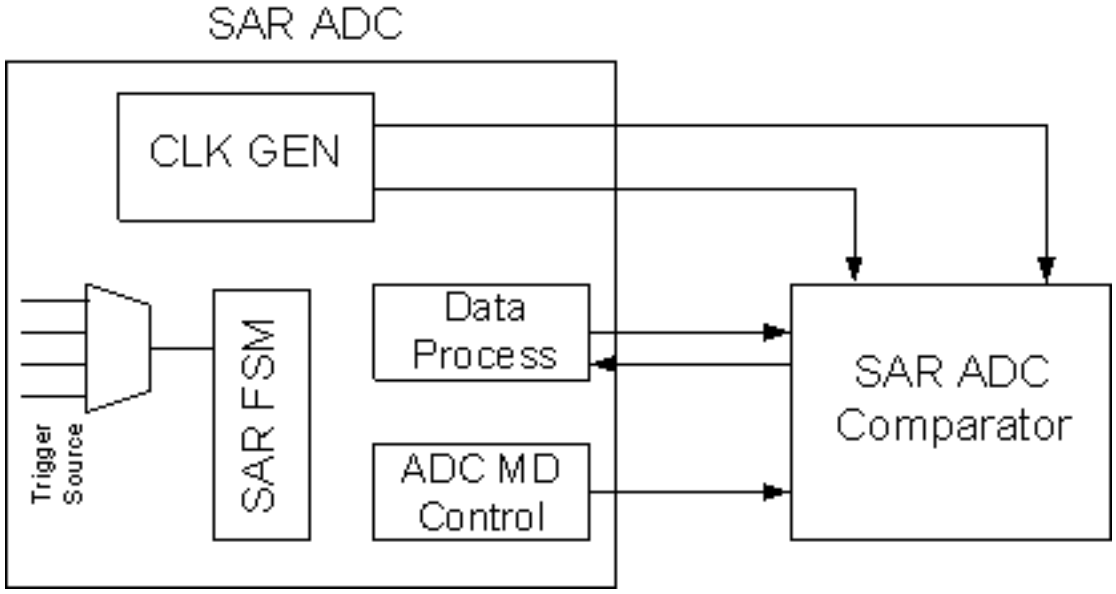
リソース	PSoC® ブロック		API メモリ (バイト数)		ピン (外部入出力ごと)
	デジタル	SAR	フラッシュ	RAM	
CY8C21x45、CY8C22x45、CY8C28x45、CY8C28x43、CY8C28x13、CY8C28x03、CY8C28x52					
SAR10	0	1	240	0	0

特性および概要

- 最高のアナログ - デジタル変換を CY8C21x45、CY8C22x45、CY8C28x45 デバイスで可能にします。
- 10-bit の分解能
- ワンショット変換
- フリーラン変換
- 選択可能な変換トリガ
- プログラマブル クロック分周器
- 変換終了後、自動で低消費電力モードに切り替え

SAR10 ユーザ モジュールは、SAR ブロックを使って入力電圧をデジタル コードに変換する、10-bit 逐次比較型 (SAR) ADC 変換器です。これは、各サンプルで 10-bit の符号なし値を生成します。このユーザ モジュールは、ソフトウェア トリガ、ハードウェア トリガ、フリーランという 3 モードのアナログ - デジタル変換をサポートします。

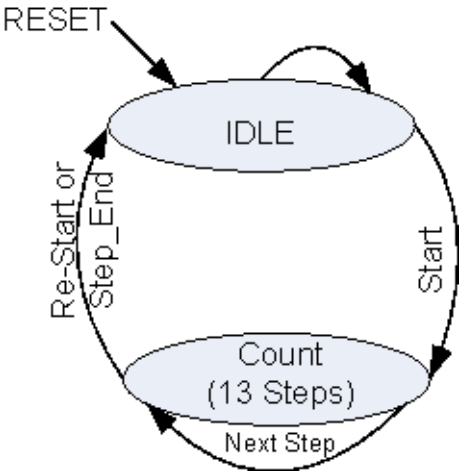
Figure 1. 図 1. 10-Bit SAR ADC のブロック ダイアグラム



機能説明

SAR10 ユーザ モジュールは、各変換で 12 ADC クロック サイクルを必要とします。最初の 2 つのクロック サイクルは、アナログ入力信号をサンプリングするためのものです。残りの 10 クロック サイクルがデータ変換に使用されます。変換中に基準クロックと比較される際、1 サイクルで、選択された ADC クロック サイクルは通常の 2 倍に延長されます。このため、ADC 有限ステートマシンから見ると、変換には 13 クロック サイクルかかります。有限ステートマシンは、アイドル状態で少なくともう 1 つ余分の SYSCLK ティックを必要とします。すべての変換は、アイドル状態から開始され、アイドル状態に戻る必要があります。

Figure 2.　図 2. 10-bit SAR ADC ステートマシン



SAR10 ユーザ モジュールは、以下の 3 つのアナログ - デジタル変換モードを持ちます。

- ソフトウェア トリガ モード: SAR_CR0_REG レジスタで START ビットに 1 ビット書き込むたびに (SAR が有効な場合)、新しい変換がトリガされます。終わっていない変換は割り込まれ、新しい変換が直ちに開始されます。変換が完了すると、ステートマシンが IDLE に戻ります。
- フリーラン モード : 変換は、ADC を無効にするまで繰り返し実行されます。ソフトウェア トリガは利用可能です。SAR_CR0_REG レジスタで START ビットに 1 ビット書き込むと、新しい変換が開始されます。
- ハードウェア トリガ モード : 自動トリガ モードまたは自動調整モードとも呼ばれます。4 つのハードウェア トリガ ソースのいずれかを選択し、そのハードウェア ソースを使用して変換の開始 (START) をトリガします。異なるトリガ ソースを使用する以外は、ソフトウェア トリガとまったく同じように動作します。

[+] Feedback

自動トリガ モードが有効になっている場合は、ADC が自動トリガ モードで実行されます。自動トリガ モードが無効になっていて、フリーラン ビットが設定されている場合は、ADC は連続して実行されます。自動トリガとフリーランの両方が無効になっている場合は、ADC がワンショット モードで実行されます。ワンショット モードでは、SAR_CR0_REG レジスタの START ビットに 1 ビットが書き込まれるたびに、ADC が実行されます。

SAR10 サンプリング速度を下の表に示します。

Table 1. ADC サンプリング速度とクロックの選択

SYSCLK (IMO)	最高速		最低速	
	クロック設定	実際の SPS	クロック設定	実際の SPS
24 MHz	SYSCLK/12	152.8 KSPS	SYSCLK/64	28.8 KSPS

DC 電気的特性と AC 電気的特性

その他の電気的特性については、ご使用になる PSoC デバイスのデータシートを参照してください。

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。それぞれ、4.75V～5.25V および T_A = –40ºC～+85ºC と、3.0V～3.6V および T_A = –40ºC～+85ºC です。典型的なパラメータは、温度 25°C、電圧 5V および 3.3V の場合の値で、設計の指針としてのみ示します。

Table 2. SAR10 ADC の AC 仕様

記号	説明	最小値	標準値	最大値	単位	注
$F_{INSAR10}$	SAR10 ADC の入力クロック周波数	–	–	2.0	MHz	
F_{SSAR10}	SAR10 ADC のサンプリング速度 SAR10 ADC の分解能 = 10 ビット	–	–	152.8	ksps	サンプル期間には、13 ADC 入力クロックと、少なくともも う 1 つ余分の SysClk ティックが含まれています。

DC 仕様

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。それぞれ、4.75V～5.25V および T_A = –40ºC～+85ºC と、3.0V～3.6V および T_A = –40ºC～+85ºC です。典型的なパラメータは、温度 25°C、電圧 5V および 3.3V の場合の値で、設計の指針としてのみ示します。

Table 3. SAR10 ADC の DC 仕様

記号	説明	最小値	標準値	最大値	単位	注
INL_{SAR10}	積分非直線性	-2.5	–	2.5	LSB	10-bit の分解能
DNL_{SAR10}	微分非直線性	-1.5	–	1.5	LSB	10-bit の分解能
$I_{VREFSAR10}$	SAR10 ADC の VREF 入力として構成する場合の、P2[5] への入力電流。	-	–	0.5	mA	内部電圧リファレンス バッファは、この構成では無効になっています。
$V_{VREFSAR10}$	SAR10 ADC の外部電圧リファレンスとして構成する場合の、P2[5] の入力基準電圧。	3.0	–	4.95	V	VREF が SAR10 ADC 内部にバッファリングされる場合は、P2[5] における電圧レベル (外部基準電圧として構成されている場合) を、Vdd ピンのチップ電源電圧レベルより 300 mV 以上低い電圧に、常に維持する必要があります。($V_{VREFSAR10}$ < (Vdd - 300 mV))。

配置

SAR10 ユーザ モジュールは、SAR10 ブロックを占有します。1 つの構成に、複数の SAR10 ユーザ モジュールを配置することはできません。

パラメータおよびリソース

ADC Clock (ADC クロック)

ADC クロック パラメータは、SAR モジュールの現在のクロック源を設定します。このパラメータには、次のオプションが含まれています。

ADC Clock (ADC クロック)	説明
DivideBy2	ADC クロックは、システムクロックを 2 で除算して取得されます。
DivideBy4	ADC クロックは、システムクロックを 4 で除算して取得されます。
DivideBy6	ADC クロックは、システムクロックを 6 で除算して取得されます。
DivideBy8	ADC クロックは、システムクロックを 8 で除算して取得されます。

ADC Clock (ADC クロック)	説明
DivideBy12	ADC クロックは、システムクロックを 12 で除算して取得されます。
DivideBy16	ADC クロックは、システムクロックを 16 で除算して取得されます。
DivideBy32	ADC クロックは、システムクロックを 32 で除算して取得されます。
DivideBy64	ADC クロックは、システムクロックを 64 で除算して取得されます。

Run Mode (動作モード)

動作モード パラメータは、ユーザ モジュールがフリーランまたはワンショット モードのどちらで実行されるかを設定します。このパラメータには、以下のオプションがあります。

Run Mode (動作モード)	説明
One-shot (ワンショット)	1 つのアナログ - デジタル変換のみが実行されます。
Free run (フリーラン)	アナログ - デジタル変換が繰り返し実行されます。

Input (入力)

ADC 入力チャネル選択パラメータは、現在の入力ソースを設定します。このパラメータには、次のオプションがあります。

ADC Input Channel Selection (ADC 入力チャネル選択)	説明
Port_0_0	ポート 0 のピン 0 が、SAR10 モジュールの入力ソースとして使用されます。
Port_0_1	ポート 0 のピン 1 が、SAR10 モジュールの入力ソースとして使用されます。
Port_0_2	ポート 0 のピン 2 が、SAR10 モジュールの入力ソースとして使用されます。
Port_0_3	ポート 0 のピン 3 が、SAR10 モジュールの入力ソースとして使用されます。
Port_0_4	ポート 0 のピン 4 が、SAR10 モジュールの入力ソースとして使用されます。
Port_0_5	ポート 0 のピン 5 が、SAR10 モジュールの入力ソースとして使用されます。

ADC Input Channel Selection (ADC 入力チャネル選択)	説明
Port_0_6	ポート 0 のピン 6 が、SAR10 モジュールの入力ソースとして使用されます。
Port_0_7	ポート 0 のピン 7 が、SAR10 モジュールの入力ソースとして使用されます。
AnalogMuxBus_0	AnalogMuxBus_0 が、SAR10 モジュールの入力ソースとして使用されます。
AnalogMuxBus_1	AnalogMuxBus_1 が、SAR10 モジュールの入力ソースとして使用されます。

Auto Trigger Global Enable (自動トリガ グローバル イネーブル)

自動トリガ グローバル イネーブル パラメータは、ハードウェア トリガ モードを有効にします。このパラメータには、以下のオプションがあります。

Auto Trigger Global Enable (自動トリガ グローバル イネーブル)	説明
Disable (無効)	アナログ - デジタル変換は、ソフトウェア トリガによって駆動されます。
Enable (有効)	アナログ - デジタル変換は、外部ブロック トリガ信号によって駆動されます。

Select Auto Trigger Source (自動トリガ ソースの選択)

自動トリガ ソースの選択は、現在の自動トリガ ソースを設定します。このパラメータには、以下のオプションがあります。

Select Auto Trigger Source (自動トリガ ソースの選択)	説明
TGL	下位 8 ビットのデジタル パスが、自動トリガ ソースとして使用されます。
TGH	上位 8 ビットのデジタル パスが、自動トリガ ソースとして使用されます。
TG16Bit	上位 / 下位 8 ビットのデジタル パスの組み合わせが、自動トリガ ソースとして使用されます。
TGINCMP	GIE または内部コンパレータが、自動トリガ ソースとして使用されます。

Resolution (分解能)

分解能は、現在の分解能モードを設定します。このパラメータには、以下のオプションがあります。

Resolution (分解能)	説明
8 bits (8 ビット)	左寄せモードが設定されます。8-bits の結果が返されます。
10 bits (10 ビット)	右寄せモードが設定されます。10-bits の結果が返されます。

Note * このパラメータは、CY8C28x45 デバイスでのみ利用できます。

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) 関数は、高レベルでモジュールを扱うことができるように、ユーザ モジュールの一部として提供されています。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。

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 デザイナが、そのプロジェクトのユーザ モジュールの第 1 インスタンスに SAR10_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。指定されたサンプルの名前は、すべてのグローバル機能名、変数、定数記号の接頭辞になります。次の説明では、簡単にするために、インスタンス名は省略されて単に 「SAR10」となっています。

Note ** ここで、全ユーザ 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できます。これらの値が呼び出し後に必要となる場合は、呼び出す関数を使って、呼び出し前に A および X の値を維持してください。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。

SAR10_Start

説明

SAR10 ブロックを有効にし、アナログ - デジタル変換を開始します。

C プロトタイプ :

```
void  SAR10_Start(void);
```

アセンブリ :

```
lcall  SAR10_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。

SAR10_Stop

説明

SAR10 ブロックを無効にし、アナログ - デジタル変換を停止します。

C プロトタイプ :

```
void  SAR10_Stop(void);
```

アセンブリ :

```
lcall  SAR10_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。

SAR10_EnableInt

説明

SAR10 ブロックの割り込みを有効にします。

C プロトタイプ :

```
void  SAR10_EnableInt(void);
```

アセンブリ :

```
lcall  SAR10_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。

SAR10_DisableInt

説明

SAR10 ブロックの割り込みを無効にします。

C プロトタイプ :

```
void  SAR10_DisableInt(void);
```

アセンブリ :

```
lcall  SAR10_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。

SAR10_Trigger

説明

SAR10 をトリガして、サンプル 1 つを変換します。SAR10 ユーザ モジュールが自動トリガ モードで実行されている場合、この関数は、ADC サンプリング動作には影響を与えません。

C プロトタイプ :

```
void  SAR10_Trigger(void);
```

アセンブリ :

```
lcall  SAR10_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。

SAR10_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  SAR10_fIsDataAvailable(void);
```

アセンブリ :

```
lcall  SAR10_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。

SAR10_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に、データを取得する前に SAR10_fIsDataAvailable() を呼び出してください。

C プロトタイプ :

```
INT  SAR10_iGetData(void);
```

アセンブリ :

```
lcall  SAR10_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、LSB は X に、MSB はアキュムレータに返されます。

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。

この関数は、右寄せモードの場合にのみ、CY8C28x45 デバイス上の正しい結果を返します。

SAR10_bGetData

説明

最後に変換したデータの最上位 8 ビットを返します。データが有効なことを確認するために、データを取得する前に SAR10_fIsDataAvailable() を呼び出してください。

C プロトタイプ :

```
BYTE  SAR10_bGetData(void);
```

アセンブリ :

```
lcall  SAR10_bGetData
```

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、結果はアキュムレータに返されます。

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。

SAR10_SetADCChannel

説明

SAR10 入力ソースを選択します。

C プロトタイプ :

```
void  SAR10_SetADCChannel(BYTE bChannel);
```

アセンブリ :

```
mov A,bChannel
lcall  SAR10_SetADCChannel
```

パラメータ :

bChannel は入力ソースです。シンボル名は、C 言語およびアセンブリ言語で記載されています。これらに関連する値は、以下の表に記載されています。

シンボル名	値	説明
SAR10_CHS_P00	0x00	Port_0_0 は、SAR10 モジュールの入力ソースです。
SAR10_CHS_P01	0x08	Port_0_1 は、SAR10 モジュールの入力ソースです。
SAR10_CHS_P02	0x10	Port_0_2 は、SAR10 モジュールの入力ソースです。
SAR10_CHS_P03	0x18	Port_0_3 は、SAR10 モジュールの入力ソースです。
SAR10_CHS_P04	0x20	Port_0_4 は、SAR10 モジュールの入力ソースです。
SAR10_CHS_P05	0x28	Port_0_5 は、SAR10 モジュールの入力ソースです。
SAR10_CHS_P06	0x30	Port_0_6 は、SAR10 モジュールの入力ソースです。
SAR10_CHS_P07	0x38	Port_0_7 は、SAR10 モジュールの入力ソースです。
SAR10_CHS_AMUX0	0x60	AnalogMuxBus_0 は、SAR10 モジュールの入力ソースです。
SAR10_CHS_AMUX1	0x68	AnalogMuxBus_1 は、SAR10 モジュールの入力ソースです。

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。

SAR10_SetTriggerSrc

説明

SAR10 自動トリガ ソースを選択します。

C プロトタイプ :

```
void  SAR10_SetTriggerSrc(BYTE bSrc);
```

アセンブリ :

```
mov A,bSrc
lcall  SAR10_SetTriggerSrc
```

パラメータ :

bSrc: トリガ ソースです。シンボル名は、C 言語およびアセンブリ言語で記載されています。これらに関連する値は、以下の表に記載されています。

シンボル名	値	説明
SAR10_SRC_TGRL	0x00	下位 8 ビットのデジタル パスが自動トリガ ソースです。
SAR10_SRC_TGRH	0x10	上位 8 ビットのデジタル パスが自動トリガ ソースです。
SAR10_SRC_TGR16	0x20	上位 / 下位 8 ビットのデジタル パスの組み合わせが、自動トリガ ソースです。
SAR10_SRC_TGRINCMP	0x30	GIE または内部コンパレータが、自動トリガ ソースです。

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。

SAR10_EnableAutoTrigger

説明

SAR10 自動トリガ関数のグローバル イネーブル制御。

C プロトタイプ :

```
void  SAR10_EnableAutoTrigger(BYTE bMode);
```

アセンブリ :

```
mov A,bMode
lcall  SAR10_EnableAutoTrigger
```

パラメータ :

bMode は、自動トリガ モードを有効または無効にします。シンボル名は、C 言語およびアセンブリ言語で記載されています。これらに関連する値は、以下の表に記載されています。

シンボル名	値	説明
SAR10_AUTOTGR_ENABLE	0x01	自動トリガ モードを有効にします。
SAR10_AUTOTGR_DISABLE	0x00	自動トリガ モードを無効にします。

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。

SAR10_SetClk

説明

ADC サンプリング速度とクロック選択を設定します。システムクロックが 24 MHz に設定されている場合は、SAR10_SYSCLK_2 設定により、動作が安定しないほど速いサンプリング速度が設定されます。24 MHz システムクロックでは、SAR10_SYSCLK_2 を使用しないでください。

C プロトタイプ :

```
void  SAR10_SetClk(BYTE bClkMode);
```

アセンブリ :

```
mov A,bClkMode
lcall  SAR10_SetClk
```

パラメータ :

bClkMode はクロック源です。シンボル名は、C 言語およびアセンブリ言語で記載されています。これらに関連する値は、以下の表に記載されています。

シンボル名	値	説明
SAR10_SYSCLK_2	0x00	システムクロックは 2 で除算されます。
SAR10_SYSCLK_4	0x02	システムクロックは 4 で除算されます。
SAR10_SYSCLK_6	0x04	システムクロックは 6 で除算されます。
SAR10_SYSCLK_8	0x06	システムクロックは 8 で除算されます。
SAR10_SYSCLK_12	0x08	システムクロックは 12 で除算されます。
SAR10_SYSCLK_16	0x0A	システムクロックは 16 で除算されます。
SAR10_SYSCLK_32	0x0C	システムクロックは 32 で除算されます。
SAR10_SYSCLK_64	0x0E	システムクロックは 64 で除算されます。

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。

SAR10_SetRunMode

説明

ADC をフリーラン モードまたはワンショット モードに設定します。

C プロトタイプ :

```
void  SAR10_SetRunMode(BYTE bRunMode);
```

アセンブリ :

```
mov A,bRunMode
lcall  SAR10_SetRunMode
```

パラメータ :

bRunMode は動作モードです。シンボル名は、C 言語およびアセンブリ言語で記載されています。これらに関連する値は、以下の表に記載されています。

シンボル名	値	説明
SAR10_ONESHOT	0x00	One-shot (ワンショット)
SAR10_FREERUN	0x08	Free run (フリーラン)

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

ここに記載されている C コードは、SAR10 ユーザ モジュールの使用方法を示しています。

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

int iResult;
void main(void)
{
SAR10_SetClk(SAR10_SYSCLK_64); // Set clock source - system clock/64
SAR10_SetRunMode(SAR10_ONESHOT); // Set running method - one-shot
SAR10_SetADCChannel(SAR10_CHS_P05); // Set Port_0_5 as input
SAR10_EnableInt(); // Enable SAR10 interrupt
SAR10_Start(); // Start conversion

M8C_EnableGInt; // Enable global interrupt

while(1)
{
SAR10_Trigger(); //Trigger new sample
 while(SAR10_fIsDataAvailable()==0);//Wait while data is not ready
 iResult = SAR10_iGetData(); // Read result
}
}
```

アセンブリ言語での同じコードは次のようになります。

```
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
area bss(RAM, REL)
_iResult:
iResult: BLK   2
area text (ROM, REL)
_main:
mov A, SAR10_SYSCLK_64
call SAR10_SetClk ;Set clock source - system clock/64
mov A, SAR10_ONESHOT
call SAR10_SetRunMode ;Set running method - one-shot
mov A, SAR10_CHS_P05
call SAR10_SetADCChannel ;Set Port_0_5 as input
call SAR10_EnableInt ;Enable SAR10 interrupt
call SAR10_Start ;Start conversion
M8C_EnableGInt ;Enable global interrupt
.ReadADCData:
call SAR10_Trigger ;Trigger new sample
.Wait:
call   SAR10_fIsDataAvailable
cmp    A, 0
jz .Wait ;Wait while data is not ready
call SAR10_iGetData ;Read result
mov [_iResult+1], X ;Get MSB of result
```

```
mov [_iResult], A ;Get LSB of result
jmp .ReadADCData
```

コンフィグレーション レジスタ

Table 4. SAR_CR0_REG

ビット	7	6	5	4	3	2	1	0
値	0	InputSelect				Ready (準備完了)	Start (開始)	Enable (有効)

このレジスタは、SAR10 ADC を調整するために使用されます。これによって、入力の設定、ADC サンプリングの開始、SAR10 ブロックの有効化を実行できます。SAR_CR0_REG は、変換されたデータの状態も表します。

Enable ビットは、SAR10_Start または SAR10_Stop API ルーチンを呼び出して変更することができます。

Start ビットは、SAR10_Trigger API ルーチンを呼び出して変更することができ、ADC 変換 1 つを開始します。

Ready ビットは、SAR10_fIsDataAvailable API ルーチンを呼び出して読み取ることができ、変換されたデータの状態を判断します。

InputSelect ビットは、入力ソースを決定します。これらのビットの値は、デバイス エディタの 「ADC Input Channel Selection (ADC 入力チャネルの選択)」パラメータの設定によって決定されます。値は、SAR10_SetADCChannel API を呼び出すことによってランタイムで変更可能です。

Table 5. SAR_CR1_REG

ビット	7	6	5	4	3	2	1	0
値	0	0	TriggerSource		ClockSource			AutoTrig

このレジスタは、SAR10 ADC を調整するために使用されます。これを使って、クロックおよびトリガ ソースを設定し、自動トリガ モードを設定します。

TriggerSource ビットは、外部トリガ信号のソースを決定します。これらのビットの値は、デバイス エディタの 「Select Auto Trigger Source (自動トリガ ソースの選択)」パラメータの設定によって決定されます。値は、SAR10_SetTriggerSrc API を呼び出すことによってランタイムでも変更可能です。

ClockSource ビットは、クロック源を決定します。これらのビットの値は、デバイス エディタの「ADC Input Channel Selection (ADC 入力チャネルの選択)」パラメータの設定によって決定されます。値は、SAR10_SetClk API を呼び出すことによってランタイムでも変更可能です。

AutoTrig ビットは、自動トリガの読み取りモードを決定します。このビットの値は、デバイス エディタの 「Auto Trigger Global Enable (自動トリガ グローバル イネーブル)」パラメータの設定によって決定されます。値は、SAR10_EnableAutoTrigger API を呼び出すことによってランタイムでも変更可能です。

Table 6. SAR_CR2_REG

ビット	7	6	5	4	3	2	1	0
値	0	0	0	0	Freerun	0	0	0

このレジスタは、SAR10 ADC を調整するために使用されます。特に、繰り返して実行するようモードを設定することができます。

Freerun ビットは、フリーラン読み取りモードを決定します。このビットの値は、デバイス エディタにあるユーザ モジュール パラメータの 「Run Mode (動作モード)」パラメータの設定によって決定されます。値は、SAR10_SetRunMode API を呼び出すことによってランタイムでも変更可能です。

Table 7. SAR_DH_REG

ビット	7	6	5	4	3	2	1	0
値	HighData							

このレジスタには、CY8C21x45、CY8C22x45、CY8C28x45 デバイスの左寄せモードで、ADC 変換されたデータの最上位 8 ビットが含まれます。

このレジスタには、CY8C28x45 デバイスの右寄せモードで、ADC 変換されたデータの最上位 2 ビットが含まれます。

HighData ビットは、SAR10_iGetData. を呼び出すことによって読み取ることができます。

Table 8. SAR_DL_REG

ビット	7	6	5	4	3	2	1	0
値							LowData	

このレジスタには、CY8C21x45、CY8C22x45、CY8C28x45 デバイスの左寄せモードで、ADC 変換されたデータの最下位 2 ビットが含まれます。

このレジスタには、CY8C28x45 デバイスの右寄せモードで、ADC 変換されたデータの最上位 8 ビットが含まれます。

LowData ビットは、SAR10_iGetData. を呼び出すことによって読み取ることができます。

Table 9. SAR_CR3_REG

ビット	7	6	5	4	3	2	1	0
値	LALIGN							

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 © 2009-2011 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.