8-Bit 逐次逼近 ADC 数据表 SAR8 V 1.0

Copyright © 2005-2010 Cypress Semiconductor Corporation. All Rights Reserved.

资源	PSoC® 模块			API 存储器 （字节）		引脚 （每个外部 I/O 和时钟）
	数字	模拟 CT	模拟 SC	闪存	RAM	
CY8C24x33, CY8C23x33	0	0	0	147	0	1

请参见应用笔记 “Analog - ADC Selection” （模拟 - ADC 选择） AN2239 获取有关其他转换器的信息。

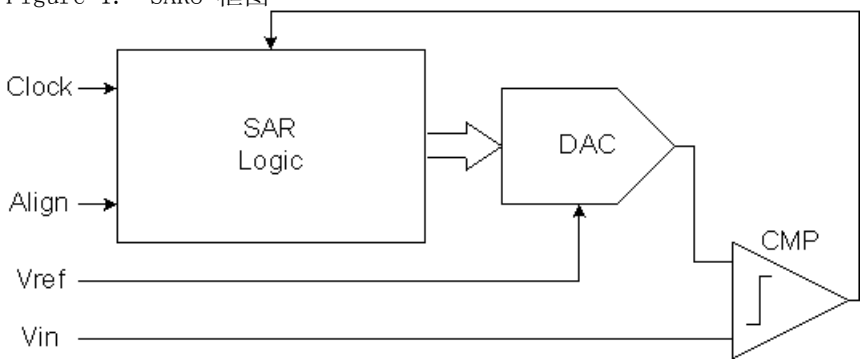
如需获取一个或多个使用此用户模块的完整配置功能性示例项目，请转到 www.cypress.com/psocexampleprojects.

特性与概述

- 在 CY8C24x33 和 CY8C23x33 器件上的最佳模数转换结果
- 快速转换时间 （典型值 2.7 µs）
- 逐次逼近功能支持
- 8-bit 分辨率
- 来自 CT 模块的八个主要输入模拟通道和两个内部模拟通道
- 单一转换
- 自由运行转换
- 可选转换触发器
- 可编程采样时间
- 可编程时钟分频器
- 取消 / 重启运行转换的功能
- 支持 PWM、定时器和计数器周期的任一点的自动对齐 / 触发器
- 在单一转换模式下的每个转换之后，自动进入低功耗模式

SAR8 用户模块是 8-bit 逐次逼近寄存器 （SAR）ADC 转换器，通过使用专用模拟 PSoC 模块能够把输入电压转换为数字代码。 特性是 2.7 µs 的典型转换时间（受固件处理的限制）和对每个采样生成一个 8-bit 无符号值。 在电机控制应用中，要获取最佳性能，需要在 PWM 周期的特定时间触发 ADC 执行一次模数转换。

Figure 1. SAR8 框图



功能说明

逐次逼近寄存器是 PSoC SARADC_DL 寄存器。 该寄存器保存转换结果。 通过标量参考电压和与输入电压相比较，SAR 逻辑构成一系列的逼近值。 执行二进制搜索找到与输入电压最接近的 DAC 设置。

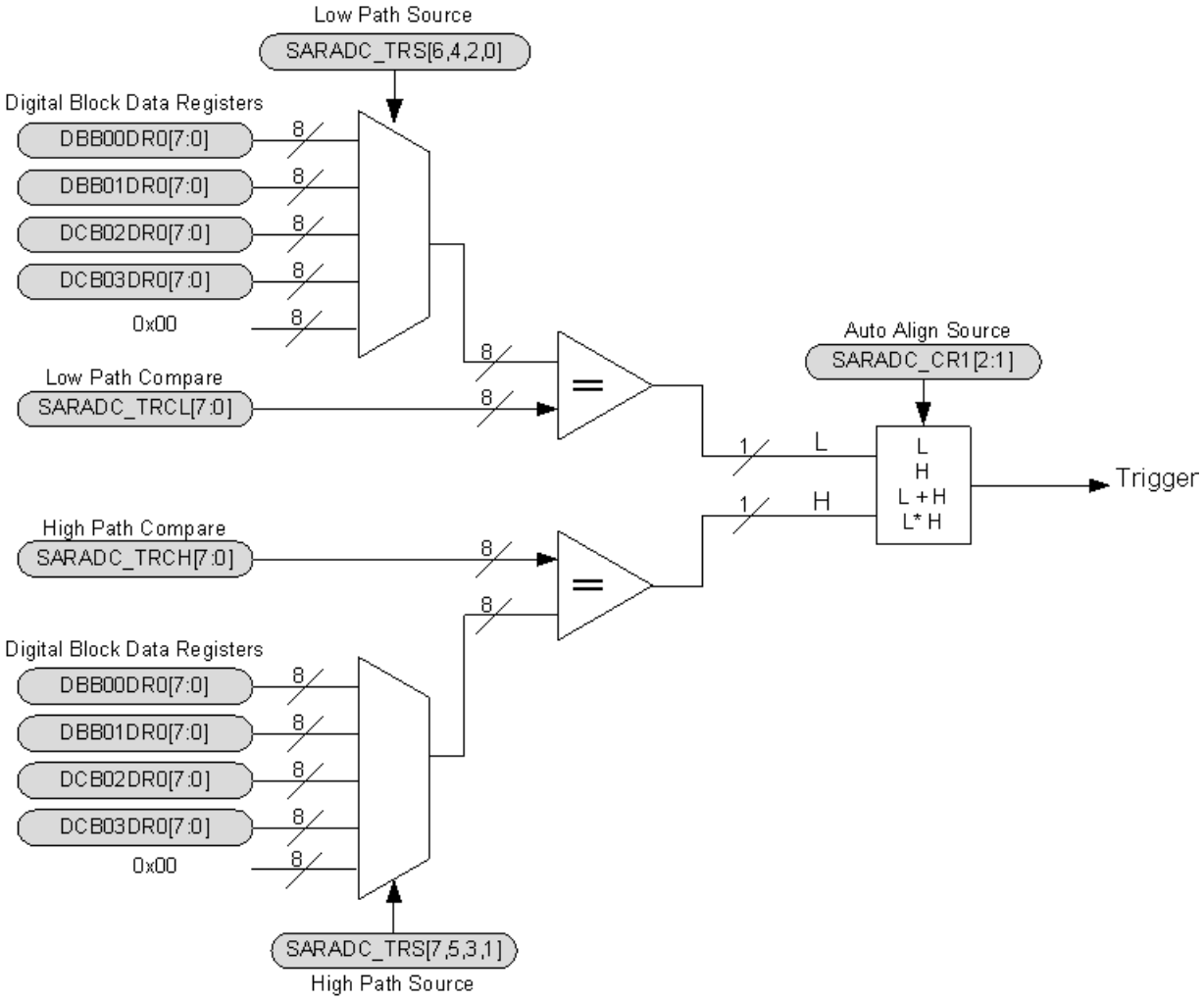
SAR8 的独特功能是允许转换和一两个数字模块同步的高级转换触发器逻辑。

SAR8 ADC 可以在单触发模式或者自由运行模式下运作。 在这两种情况下，甚至当 ADC 在执行一个转换时，API 功能可以让您开始新的测量。 在单触发模式下，经过转换之后 ADC 将自动进入低功耗模式。 在自由运行模式下，一旦停止，ADC 将进入低功耗模式。

自动对齐选项可以让您用两个 8-bit 数字电压比较器 （低和高）来触发转换请求。 这些电压比较器可由任一 PSoC 器件的数字模块驱动。 电压比较器将模块 DRO 寄存器的实际值与存储在 SAR8_COMPARE_LO_REG 和 SAR8_COMPARE_HI_REG 寄存器中的数据相比较。 每个电压比较器能够独立地触发 ADC 或者他们能够结合构成一个 16-bit 触发器源。

[+] Feedback

Figure 2. SAR8 自动对齐原理图



SAR8 ADC 的另一个功能是数据标量。 可以让您正确移位从结果寄存器读取出的数据。 比例因子在 1 到 2^{-N}, N=0..6 范围内可调。

[+] Feedback

直流和交流电气特性

下表列出了在以下电压和温度范围内许可的最大和最小规范： 4.75V 到 5.25V 和 -40° C。 TA. 85° C、或者 3.0V 到 3.6V 和 -40° C。 TA. 分别 85° C。 典型参数适用于 25° C 且电压为 5V 和 3.3V 的情况，仅供设计指导之用

Table 1. SAR8 直流和交流电气特性

参数	最小值	典型值	最大值	单位	条件和注释
$V_{ADCVREF}$	3.0[a]	–	Vdd	V	在配置为 ADC 参考电压时，引脚 P3[0] 处的参考电压。 $V_{ADCVREF} \leq Vdd$.
$I_{ADCVREF}$	–	–	3	mA	当 P3[0] 配置为 ADC V_{REF} 时，提供到 SAR8 ADC 的电流
C_{IN}	–	–	60	pF	ADC 输入上的输入电容
R_{IN}	–	–	1	kΩ	ADC 输入上的输入电阻
积分非线性误差[b]	-1.2	–	+1.2	LSB	R-2R 积分非线性度。 最低有效位最大值在子范围之上，不超过全量程的 1/16。
DNL[b]	-1	–	+1	LSB	R-2R 微分非线性度。 输出是单调的。
频率	0.375	–	24	MHz	输入时钟频率

a. 较低 $V_{ADCVREF}$ 电压可以和精度上的减少一起使用。

b. 适用于 1.5 Mhz 或更低的 ADC 时钟频率。

放置

有一个支持 SAR8 的专用资源。 它是唯一的放置选择。

[+] Feedback

参数和资源

ADCInput

可以选择任一端口 0 引脚作为 ADC 的信号源。 也可以使用模拟模块 ACB00 和 ACB01 作为信号源。

ADCClock

时钟频率有以下几种选择：

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

ADC 采样率等于 ADC 时钟除以 8，转换时间等于 ADC 时钟周期乘以 8.

转换之后 ADC 自动进入到低功耗模式，所以对单触发模式而言，使用较低频率 ADC 时钟比较高频率 ADC 时钟消耗更多功耗。 在自由运行模式下，使用较低频率 ADC 时钟比较高频率 ADC 时钟消耗更少功耗。

标度

转换结果当读取出来时，自动用整数除以比例因子。 可能的标度值有： 1、2、4、8、16、32、64。

运行

该参数选择单触发模式或者自由运行转换模式。

ADCPower

SAR8 模块能够从内部 VPWR (Vdd) 或者从 P3[0] 中获取功耗。

VPWRADC 电压比较器模块从内部 VPWR (Vdd) 获取电源。**P3[0]**ADC 电压比较器模块从 P3[0] 获取电源。P3[0] 必须小于或等于 VPWR (Vdd)。

R2RPower

有两种可能的选择项。

VPWRADC DAC 参考生成模块从内部 VPWR (Vdd) 获取电源。 ADC DAC 功耗 (R2RPower) 必须小于或等于 ADC 电压比较器模块 (ADCPower) 的功耗。 **P3[0]**ADC R2R 参考生成模块从 P3[0] 获取电源。P3[0] 必须小于或等于 VPWR。

[+] Feedback

应用程序编程接口

应用程序编程接口 (API) 子程序作为用户模块的一部分提供，使设计人员能够在较高的层级处理模块。本部分具体指明了每个函数的接口以及由 “ 包含 ” 文件所提供的常数。

Note ** 在这里，如同所有用户模块 API 中的一样，A 和 X 寄存器的值可能会通过调用 API 函数而发生更改。 发起调用 API 的函数有责任在调用之前保留 A 和 X 的值 （如果调用后需要再次用到它们）。 选择这种 “ 寄存器易变 ” 策略是为了提高效率，并且自从 PsoC Designer 的 1.0 版本起使用。 C 语言编译器自动遵循此要求。 汇编语言程序员也必须确保其代码遵守这一策略。 虽然有些用户模块的 API 函数可能保留 A 和 X 不变，但并不保证在未来也将如此。

提供入口点以初始化 SAR8 用户模块、执行和读取转换以及禁用 SAR8 函数。

SAR8_Start

说明：

启用 ADC 运算。 在完成转换之后，SAR8 自动进入低功耗模式。 如果 “ 运行 ”(Run) 参数设为 “ 自由运行 ”(Free-Run)，ADC 就会在调用 SAR8_Start 后开始收集数据。 如果 “ 运行 ”(Run) 参数设为 “ 单触发 ” (One-Shot)，那么在采集到样本之前必须调用 SAR8_Trigger 函数或者必须出现来自自动对齐电路的触发信号。

C 语言原型：

```
void  SAR8_Start(void)
```

汇编程序：

```
lcall  SAR8_Start
```

参数：

无

返回：

无

副作用：

请参见 API 章节开始部分的注释 **。

SAR8_Stop

说明：

禁用 ADC 运算。 在调用停止后，ADC 会进入到低功耗模式。

C 语言原型：

```
void  SAR8_Stop(void)
```

汇编程序：

```
lcall  SAR8_Stop
```

参数：

无

返回：

无

[+] Feedback

副作用：

请参见 API 章节开始部分的注释 **。

SAR8_EnableInt

说明：

启用中断。 在使用 SAR8 中断之前，必须使用 M8C_EnableGInt 宏全局启用中断。

C 语言原型：

```
void  SAR8_EnableInt(void)
```

汇编语言：

```
lcall  SAR8_EnableInt
```

参数：

无

返回值：

无

副作用：

请参见 API 章节开始部分的注释 **。

SAR8_DisableInt

说明：

禁用中断。

C 语言原型：

```
void SAR8_DisableInt(void)
```

汇编程序：

```
lcall SAR8_DisableInt
```

参数：

无

返回值：

无

副作用：

请参见 API 章节开始部分的注释 **。

SAR8_Trigger

说明：

从下一个系统时钟周期触发 ADC 执行样本和转换。 如果 ADC 已经在转换样本中，SAR8_Trigger 会强制 ADC 取消正在进行的转换，并在下一个系统时钟周期重启新的转换。 如果 ADC 在自由运行模式下运作，则该函数将不会影响 ADC 采样。

[+] Feedback

C 原型：

```
void  SAR8_Trigger(void)
```

汇编程序：

```
lcall  SAR8_Trigger
```

参数：

无

返回：

无

副作用：

请参见 API 章节开始部分的注释 **。

SAR8_fIsDataAvailable

说明：

检查采样数据的可用性。

C 语言原型：

```
BYTE  SAR8_fIsDataAvailable(void)
```

汇编程序：

```
lcall  SAR8_fIsDataAvailable
```

参数：

无

返回：

如果已经转换好数据并已准备好读取，则返回一个非零数值。

副作用：

请参见 API 章节开始部分的注释 **。

SAR8_bGetData

说明：

以无符号字节值返回转换的数据。应该调用 SAR8_fIsDataAvailable() 以验证已准备好的数据样本。

C 语言原型：

```
BYTE  SAR8_bGetData(void)
```

汇编程序：

```
lcall  SAR8_bGetData
```

参数：

无

返回：

以无符号字节值返回转换的数据。

[+] Feedback

副作用：

请参见 API 章节开始部分的注释 **。

SAR8_SelectADCChannel

说明：

从 10 个可选输入通道中选择 ADC 输入。

C 语言原型：

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

汇编程序：

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

参数：

bChannel: 一个指定 ADC 输入通道的字节。 下表给出了在 C 语言和汇编语言中提供的符号名称及其相关数值。

符号名称	值	通道
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

返回：

无

副作用：

请参见 API 章节开始部分的注释 **。

SAR8_AutoAlign

说明：

启用 / 禁用带有选定数字模块的 ADC 采样。

C 语言原型：

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


[+] Feedback

汇编程序：

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

参数：

bAlignMode： 一个指定对齐模式的字节。

符号名称	值
SAR8_NOAUTOALIGN	0
SAR8_AUTOALIGN	1

返回：

无。

副作用：

请参见 API 章节开始部分的注释 **。

SAR8_SetAlignPath

说明：

选择 ADC 自动对齐触发器逻辑。

C 语言原型：

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

汇编程序：

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

参数：

bAlignPath： 一个指定 ADC 自动对齐触发器逻辑的字节。 下表给出了在 C 语言和汇编语言中提供的符号名称及其相关数值。

符号名称	值	说明
SAR8_LowHighIndependent	0x00	低和高通道完全独立。 两者都能触发 ADC。
SAR8_LowOnly	0x02	只有低通道能够触发 ADC。
SAR8_HighOnly	0x04	只有高通道能够触发 ADC。
SAR8_LowHighCombined	0x06	高和低通道结合构成一个 16-bit 触发源。

返回：

无。

[+] Feedback

副作用：

请参见 API 章节开始部分的注释 **。

SAR8_SetHighAlignSrc

说明：

选择一个数字模块作为自动对齐触发器逻辑高路径的对齐源。

C 语言原型：

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

汇编程序：

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

参数：

bAlignSrc：一个指定一个数字模块作为高路径的对齐源的字节。下表给出了在 C 语言和汇编语言中提供的符号名称及其相关数值。

符号名称	值	数字模块
SAR8_None	0x00	未连接
SAR8_DBB00	0x01	DBB00
SAR8_DBB01	0x02	DBB01
SAR8_DCB02	0x03	DCB02
SAR8_DCB03	0x04	DCB03

返回：

无。

副作用：

请参见 API 章节开始部分的注释 **。

SAR8_SetLowAlignSrc

说明：

允许您选择一个数字模块作为自动对齐触发器逻辑低路径的对齐源。

C 语言原型：

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

汇编程序：

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

参数：

bAlignSrc：一个指定任一数字模块作为低路径的对齐源的字节。下表给出了在 C 语言和汇编语言中提供的符号名称及其相关数值。

[+] Feedback

符号名称	值	数字模块
SAR8_None	0x00	未连接
SAR8_DBB00	0x01	DBB00
SAR8_DBB01	0x02	DBB01
SAR8_DCB02	0x03	DCB02
SAR8_DCB03	0x04	DCB03

返回：

无。

副作用：

请参见 API 章节开始部分的注释 **。

SAR8_SetCmpH

说明：

对自动对齐的高路径设定电压比较器值。

C 语言原型：

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

汇编程序：

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

参数：

bValue： 一个指定自动对齐的高路径的电压比较器数据的字节。

返回：

无。

副作用：

请参见 API 章节开始部分的注释 **。

SAR8_SetCmpL

说明：

对自动对齐的低路径设定电压比较器值。

C 语言原型：

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

汇编程序：

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

参数：

bValue： 一个指定自动对齐的低路径的电压比较器数据的字节。

[+] Feedback

返回：

无。

副作用：

请参见 API 章节开始部分的注释 **。

SAR8_SetClock

说明：

设置 SAR8 ADC 时钟频率。 如果更改时钟的时钟频率，则必须先停止 ADC 转换，更改时钟频率，再重启转换。

C 语言原型：

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

汇编程序：

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

参数：

bClock： 一个指定时钟频率的字节。 下表给出了在 C 语言和汇编语言中提供的符号名称及其相关数值。

符号名称	值	时钟频率
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

返回：

无。

副作用：

请参见 API 章节开始部分的注释 **。

SAR8_SetScale

说明：

在读取转换结果数据时设置比例因子。

C 语言原型：

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


[+] Feedback

汇编程序：

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

参数：

bScaleMode： 一个指定合适大小标度的字节。 下表给出了在 C 语言和汇编语言中提供的符号名称及其相关数值。

符号名称	值	比例因子
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

返回：

无。

副作用：

请参见 API 章节开始部分的注释 **。

SAR8_SetRunMode

说明：

设置 ADC 运行模式为单一转换或自由运行。 该参数通过用户模块 “ 运行 ”(Run) 参数进行设置。 除非需要更改运行模式，否则没有必要调用该函数。

C 语言原型：

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

汇编程序：

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

参数：

bRunMode： 一个指定转换模式的字节

符号名称	值
SAR8_OneShot	0x00
SAR8_FreeRun	0x01

[+] Feedback

返回：

无。

副作用：

请参见 API 章节开始部分的注释 **。

固件源代码示例

此示例代码将启动连续转换、轮询 “ 数据可用 ”(Data Available) 标志以及将转换的字节发送至用户函数。

```
;-----------------------------------------------------------------------------
; 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 语言中的相同程序：

```
//-----------------------------------------------------------------------------
// 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);
```


[+] Feedback

```
  SAR8_Start();

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

配置寄存器

下面的寄存器用于 SAR8 模块。

Table 2. 模块 SAR8，寄存器： SAR8_CONTROL_0_REG (SARADC_CR0; 0, 69h)

位	7	6	5	4	3	2	1	0
值	保留	ADC 通道				数据就绪	开始 / 繁忙	ADCEN

Table 3. 模块 SAR8，寄存器： SAR8_CONTROL_1_REG (SARADC_CR1; 0, 6Ah)

位	7	6	5	4	3	2	1	0
值	ADCPower	R2RPower	保留			对齐源		自动对齐

Table 4. 模块 SAR8，寄存器： SAR8_CONTROL_2_REG (SARADC_CR2; 1, ABh)

位	7	6	5	4	3	2	1	0
值	0	FreeRun	标度			时钟		

Table 5. 模块 SAR8，寄存器： SAR8_TRIGGER_SRC_REG (SARADC_TRS; 1, A8h)

位	7	6	5	4	3	2	1	0
值	DCB03		DCB02		DBB01		DBB00	

Table 6. 模块 SAR8，寄存器： SAR8_COMPARE_LO_REG (SARADC_TRCL; 1, A9h)

位	7	6	5	4	3	2	1	0
值	bValue							

Table 7. 模块 SAR8，寄存器： SAR8_COMPARE_HI_REG (SARADC_TRCH; 1, AAh)

位	7	6	5	4	3	2	1	0
值	bValue							

[+] Feedback

Table 8. 模块 SAR8，寄存器： SAR8_DATA_LO_REG (SARADC_DL; 0, 67h)

位	7	6	5	4	3	2	1	0
值	数据							

版本历史记录

版本	创作者	说明
1.0	DHA	初始版本。

Note PSoC Designer 5.1 在所有用户模块数据表中提供版本历史记录。 本部分记录了当前和先前用户模块版本之间区别的高级描述。

Copyright © 2005-2010 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.