10-Bit SAR ADC Datasheet SAR10 V 2.00

Copyright © 2009-2012 Cypress Semiconductor Corporation. All Rights Reserved.

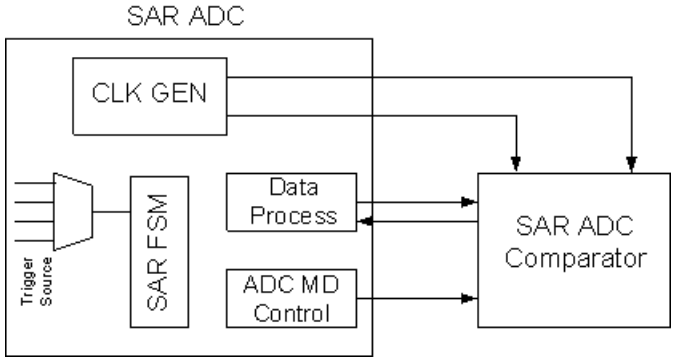
Resources	PSoC® Blocks		API Memory (Bytes)		Pins (per External I/O)
	Digital	SAR	Flash	RAM	
CY8C21x45, CY8C22x45, CY8C28x45, CY8C28x43, CY8C28x13, CY8C28x03, CY8C28x52					
SAR10	0	1	240	0	0

Features and Overview

- The best analog-to-digital conversion results on CY8C21x45, CY8C22x45, and CY8C28x45 devices.
- 10-bit resolution
- One-shot conversion
- Free running conversion
- Selectable conversion trigger
- Programmable clock divider
- Automatically enter low power mode right after conversion finishing

The SAR10 User Module is a 10-bit Successive Approximation Register (SAR) ADC converter that converts an input voltage to a digital code using the SAR block. It produces a 10-bit unsigned value for each sample. This user module supports three modes of analog-to-digital conversion: Software Trigger, Hardware Trigger, and Freerun.

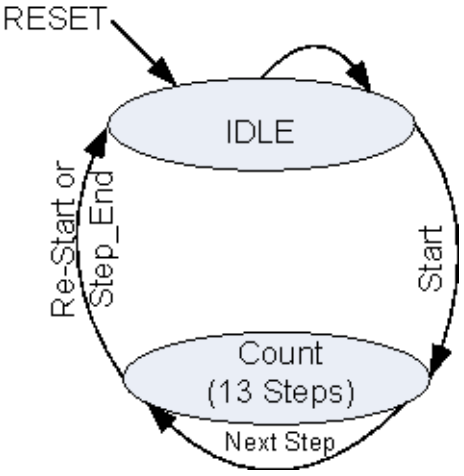
Figure 1. 10-Bit SAR ADC Block Diagram



Functional Description

The SAR10 User Module needs 12 ADC clock cycles for each conversion. The first two clock cycles are for sampling the analog input signal. Ten clock cycles are used for data conversion. On one cycle the chosen ADC clock cycle is stretched to twice its usual duration when compared to reference clocks during the conversion. Therefore, the conversion takes 13 clock cycles from the perspective of the ADC finite state machine. The finite state machine requires at least one extra SYSCLK tick for an IDLE state. Every conversion must start from an IDLE state and return to an IDLE state.

Figure 2. 10-bit SAR ADC State Machine



The SAR10 User Module has three analog-to-digital conversion modes:

- SW trigger mode: Whenever you write a one to the START bit in the SAR_CR0_REG register (if the SAR is enabled), it triggers a new conversion. Any incomplete conversion is interrupted and a new conversion is started immediately. The state machine returns to IDLE after the conversion completes.
- Free-run mode: The conversions run repeatedly until you disable the ADC. The SW trigger is still available and a new conversion is started if you write a a one to the START bit in the SAR_CR0_REG register.
- HW trigger mode: Also called autotrigger mode or autoalign mode. Select one of four hardware trigger sources and use that hardware source to trigger the START of a conversion. It behaves exactly like the SW trigger mode except that it uses a different trigger source.

If autotrigger mode is enabled the ADC runs under autotrigger mode. If autotrigger mode is disabled and the free-running bit is set, the ADC runs continuously. If both autotrigger and free-running are disabled, the ADC runs in one-shot mode. In one shot mode, the ADC runs once every time a one is written to the Start bit of the SAR_CR0_REG register.

The SAR10 sample rates are based on the following table:

Table 1. ADC Sample Rate and Clock Selection

SYSCLK (IMO)	Fastest		Slowest	
	Clock Setting	Actual SPS	Clock Setting	Actual SPS
24 MHz	SYSCLK/12	152.8 KSPS	SYSCLK/64	28.8 KSPS

DC and AC Electrical Characteristics

See the device datasheet for your PSoC device for additional electrical characteristics.

AC Electrical Characteristics

The following table lists guaranteed maximum and minimum specifications for the voltage and temperature ranges: 4.75V to 5.25V and T_A = –40ºC to +85 ºC, or 3.0V to 3.6V and T_A = –40 ºC to +85 ºC, respectively. Typical parameters apply to 5V and 3.3V at 25 °C and are for design guidance only.

Table 2. AC SAR10 ADC Specifications

Symbol	Description	Min	Typ	Max	Units	Notes
$F_{INSAR10}$	Input clock frequency for SAR10 ADC	–	–	2.0	MHz	
F_{SSAR10}	Sample rate for SAR10 ADC SAR10 ADC Resolution = 10 bits	–	–	152.8	ksps	The sample period contains 13 ADC input clocks and at least one extra SysClk tick.

DC Specifications

The following table lists guaranteed maximum and minimum specifications for the voltage and temperature ranges: 4.75V to 5.25V and T_A = –40 ºC to +85 ºC, or 3.0V to 3.6V and T_A = –40 ºC to +85 ºC, respectively. Typical parameters apply to 5V and 3.3V at 25 °C and are for design guidance only.

Table 3. DC SAR10 ADC Specifications

Symbol	Description	Min	Typ	Max	Units	Notes
INL_{SAR10}	Integral nonlinearity	-2.5	–	2.5	LSB	10-bit resolution
DNL_{SAR10}	Differential nonlinearity	-1.5	–	1.5	LSB	10-bit resolution
$I_{VREFSAR10}$	Input current into P2[5] when configured as the SAR10 ADC's VREF input.	-	–	0.5	mA	The internal voltage reference buffer is disabled in this configuration.
$V_{VREFSAR10}$	Input reference voltage at P2[5] when configured as the SAR10 ADC’s external voltage reference.	3.0	–	4.95	V	When VREF is buffered inside the SAR10 ADC, the voltage level at P2[5] (when configured as the external reference voltage) must always be at least 300 mV less than the chip supply voltage level on the VDD pin. ($V_{VREFSAR10}$ < (VDD - 300 mV)).

Placement

The SAR10 User Module occupies the SAR10 Block. You cannot place multiple SAR10 User Modules in a single configuration.

Parameters and Resources

ADC Clock

The ADC Clock parameter sets the current clock source for SAR Module. For the 24-MHz system clock, this parameter contains the following selections:

ADC Clock	Description
DivideBy12	ADC Clock is derived from division of system clock by 12.
DivideBy16	ADC Clock is derived from division of system clock by 16.
DivideBy32	ADC Clock is derived from division of system clock by 32.
DivideBy64	ADC Clock is derived from division of system clock by 64.

For the 6-MHz system clock, this parameter contains the following selections:

ADC Clock	Description
DivideBy4	ADC Clock is derived from division of system clock by 4.
DivideBy6	ADC Clock is derived from division of system clock by 6.
DivideBy8	ADC Clock is derived from division of system clock by 8.
DivideBy12	ADC Clock is derived from division of system clock by 12.
DivideBy16	ADC Clock is derived from division of system clock by 16.
DivideBy32	ADC Clock is derived from division of system clock by 32.
DivideBy64	ADC Clock is derived from division of system clock by 64.

Run Mode

The Run Mode parameter sets whether the user module is in free run or one-shot mode.This parameter contains the following selections:

Run Mode	Description
One-shot	Only one analog-to-digital conversion is executed.
Free run	Analog-to-digital conversion is executed repeatedly.

Input

The ADC Input Channel Selection parameter sets the current input source. The input connection depends on the chip that is selected. This parameter contains the following selections for the CY8C28000 chip:

Value	Description
P0[0]	Pin 0 of Port 0 is used as input source for SAR10 Module.
P0[1]	Pin 1 of Port 0 is used as input source for SAR10 Module.
P0[2]	Pin 2 of Port 0 is used as input source for SAR10 Module.
P0[3]	Pin 3 of Port 0 is used as input source for SAR10 Module.
P0[4]	Pin 4 of Port 0 is used as input source for SAR10 Module.
P0[5]	Pin 5 of Port 0 is used as input source for SAR10 Module.
P0[6]	Pin 6 of Port 0 is used as input source for SAR10 Module.
P0[7]	Pin 7 of Port 0 is used as input source for SAR10 Module.
ACC00	ACC00 block is used as input source for SAR10 Module.
ACC01	ACC01 block is used as input source for SAR10 Module.
ACC02	ACC02 block is used as input source for SAR10 Module.
ACC03	ACC03 block is used as input source for SAR10 Module.
AnalogMUXBus_0	AnalogMuxBus 0 is used as input source for SAR10 Module.
AnalogMuxBus_1	AnalogMuxBus_1 is used as input source for SAR10 Module.
VBG	VGB signal is used as input source for SAR10 Module.

This parameter contains the following selections for the CY8C22x45 chip:

Value	Description
P0[0]	Pin 0 of Port 0 is used as input source for SAR10 Module.
P0[1]	Pin 1 of Port 0 is used as input source for SAR10 Module.
P0[2]	Pin 2 of Port 0 is used as input source for SAR10 Module.
P0[3]	Pin 3 of Port 0 is used as input source for SAR10 Module.
P0[4]	Pin 4 of Port 0 is used as input source for SAR10 Module.
P0[5]	Pin 5 of Port 0 is used as input source for SAR10 Module.
P0[6]	Pin 6 of Port 0 is used as input source for SAR10 Module.
P0[7]	Pin 7 of Port 0 is used as input source for SAR10 Module.
AnalogMUXBus_0	AnalogMuxBus 0 is used as input source for SAR10 Module.
AnalogMuxBus_1	AnalogMuxBus_1 is used as input source for SAR10 Module.
VBG	VGB signal is used as input source for SAR10 Module.

Auto Trigger Global Enable

The Auto Trigger Global Enable parameter enables the Hardware Trigger mode.This parameter contains the following selections:

Auto Trigger Global Enable	Description
Disable	Analog-to-digital conversion is driven by software trigger.
Enable	Analog-to-digital conversion is driven by outside-block trigger signal.

Select Auto Trigger Source

The Select Auto Trigger Source sets the current autotrigger source.This parameter contains the following selections:

Select Auto Trigger Source	Description
TGL	The low 8 bit digital path is used as autotrigger source.
TGH	The high 8 bit digital path is used as autotrigger source.
TG16Bit	The combine high/low 8 bit digital path is used as autotrigger source.
TGINCMP	The GIE or internal comparators is used as autotrigger source

Resolution

The Resolution sets the current resolution mode.This parameter contains the following selections:

Resolution	Description
8 bits	Left-justify mode is set. 8-bits result is returned.
10 bits	Right-justify mode is set. 10-bits result is returned.

Note *This parameter is available only for CY8C28x45 device.

Application Programming Interface

The Application Programming Interface (API) functions are provided as part of the user module to allow you to deal with the module at a higher level. This section specifies the interface to each function together with related constants provided by the include files.

Each time a user module is placed, it is assigned an instance name. By default, PSoC Designer assigns SAR10_1 to the first instance of this user module in a given project. It can be changed to any unique value that follows the syntactic rules for identifiers. The assigned instance name becomes the prefix of every global function name, variable, and constant symbol. In the following descriptions, the instance name has been shortened to SAR10 for simplicity.

Note

For Large Memory Model devices, it is also the caller's responsibility to preserve any value in the CUR_PP, IDX_PP, MVR_PP, and MVW_PP registers. Even though some of these registers may not be modified now, there is no guarantee that will remain the case in future releases.

** In this, as in all user module APIs, the values of the A and X register may be altered by calling an API function. It is the responsibility of the calling function to preserve the values of A and X before the call if those values are required after the call. This "registers are volatile" policy was selected for efficiency reasons and has been in force since version 1.0 of PSoC Designer. The C compiler automatically takes care of this requirement. Assembly language programmers must also ensure their code observes the policy. Though some user module API functions may leave A and X unchanged, there is no guarantee they may do so in the future.

SAR10_Start

Description:

Enables the SAR10 Block and starts analog-to-digital conversion.

C Prototype:

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

Assembly:

```
lcall  SAR10_Start
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

SAR10_Stop

Description:

Disables the SAR10 Block and halts analog-to-digital conversion.

C Prototype:

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

Assembly:

```
lcall  SAR10_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

SAR10_EnableInt

Description:

Enables the SAR10 Block interrupt.

C Prototype:

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

Assembly:

```
lcall  SAR10_EnableInt
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

SAR10_DisableInt

Description:

Disables the SAR10 Block interrupt.

C Prototype:

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

Assembly:

```
lcall  SAR10_DisableInt
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

SAR10_Trigger

Description:

Triggers the SAR10 to convert one sample. If the SAR10 User Module is working under autotrigger mode this function does not influence the ADC sampling work.

C Prototype:

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

Assembly:

```
lcall  SAR10_Trigger
```

Parameters:

None

Return Value:

None

Side Effects:

See Note ** at the beginning of the API section.

SAR10_fIsDataAvailable

Description:

Checks the availability of sampled data.

C Prototype:

```
BYTE  SAR10_fIsDataAvailable(void);
```

Assembly:

```
lcall  SAR10_fIsDataAvailable
```

Parameters:

None

Return Value:

Returns a non-zero value if data has been converted and is ready to read.

Side Effects:

See Note ** at the beginning of the API section.

SAR10_iGetData

Description:

Return last converted data. SAR10_fIsDataAvailable() should be called before getting the data, to ensure that the data is valid.

C Prototype:

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

Assembly:

```
lcall  SAR10_iGetData
```

Parameters:

None

Return Value:

Conversion result is returned. In assembler, the LSB is returned in X and the MSB in the accumulator.

Side Effects:

See Note ** at the beginning of the API section.

This function returns the correct result on a CY8C28x45 device only in right-justify mode.

SAR10_bGetData

Description:

Returns the eight highest bits of last converted data. SAR10_fIsDataAvailable() should be called before getting the data to ensure that the data is valid.

C Prototype:

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

Assembly:

```
lcall  SAR10_bGetData
```

Parameters:

None

Return Value:

The conversion result is returned. In assembler, the result is in the accumulator.

Side Effects:

See Note ** at the beginning of the API section.

SAR10_SetADCChannel

Description:

Selects the SAR10 input source.

C Prototype:

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

Assembly:

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

Parameters:

bChannel is the input source. Symbolic names are provided in C and assembly. Their associated values are given in the following table:

Symbolic Name	Value	Description
SAR10_CHS_P00	0x00	Port_0_0 is input source for SAR10 module.
SAR10_CHS_P01	0x08	Port_0_1 is input source for SAR10 module.
SAR10_CHS_P02	0x10	Port_0_2 is input source for SAR10 module.
SAR10_CHS_P03	0x18	Port_0_3 is input source for SAR10 module.
SAR10_CHS_P04	0x20	Port_0_4 is input source for SAR10 module.
SAR10_CHS_P05	0x28	Port_0_5 is input source for SAR10 module.
SAR10_CHS_P06	0x30	Port_0_6 is input source for SAR10 module.
SAR10_CHS_P07	0x38	Port_0_7 is input source for SAR10 module.
SAR10_CHS_AMUX0	0x60	AnalogMuxBus_0 is input source for SAR10 module.
SAR10_CHS_AMUX1	0x68	AnalogMuxBus_1 is input source for SAR10 module.

Return Value:

None.

Side Effects:

See Note ** at the beginning of the API section.

SAR10_SetTriggerSrc

Description:

Select the SAR10 autotrigger source.

C Prototype:

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

Assembly:

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

Parameters:

bSrc: Is the trigger source. Symbolic names are provided in C and assembly. Their associated values are given in the following table:

Symbolic Name	Value	Description
SAR10_SRC_TGRL	0x00	Low 8 bit digital path is autotrigger source.
SAR10_SRC_TGRH	0x10	High 8 bit digital path is autotrigger source.
SAR10_SRC_TGR16	0x20	Combine high/ow 8 bit digital path is autotrigger source.
SAR10_SRC_TGRINCMP	0x30	GIE or internal comparators is autotrigger source.

Return Value:

None.

Side Effects:

See Note ** at the beginning of the API section.

SAR10_EnableAutoTrigger

Description:

Global enable control of the SAR10 autotrigger function.

C Prototype:

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

Assembly:

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

Parameters:

bMode enables or disables autotrigger mode. Symbolic names are provided in C and assembly. Their associated values are given in the following table:

Symbolic Name	Value	Description
SAR10_AUTOTGR_ENABLE	0x01	Enables autotrigger mode.
SAR10_AUTOTGR_DISABLE	0x00	Disables autotrigger mode.

Return Value:

None.

Side Effects:

See Note ** at the beginning of the API section.

SAR10_SetClk

Description:

Set ADC Sample Rate and Clock Selection. When the system clock is set to 24 MHz, the SAR10_SYSCLK_2 setting results in a sample rate that is too fast for reliable operation. Do not use SAR10_SYSCLK_2 with a 24 MHz system clock.

C Prototype:

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

Assembly:

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

Parameters:

bClkMode is the clock source. Symbolic names are provided in C and assembly. For the 24-MHz system clock, this API can be assigned the following values:

Symbolic Name	Value	Description
SAR10_SYSCLK_12	0x08	System clock is divided by 12.
SAR10_SYSCLK_16	0x0A	System clock is divided by 16.
SAR10_SYSCLK_32	0x0C	System clock is divided by 32.
SAR10_SYSCLK_64	0x0E	System clock is divided by 64.

For the 6-MHz system clock, this API can be assigned the following values:

Symbolic Name	Value	Description
SAR10_SYSCLK_4	0x02	System clock is divided by 4.
SAR10_SYSCLK_6	0x04	System clock is divided by 6.
SAR10_SYSCLK_8	0x06	System clock is divided by 8.
SAR10_SYSCLK_12	0x08	System clock is divided by 12.
SAR10_SYSCLK_16	0x0A	System clock is divided by 16.
SAR10_SYSCLK_32	0x0C	System clock is divided by 32.
SAR10_SYSCLK_64	0x0E	System clock is divided by 64.

Return Value:

None.

Side Effects:

See Note ** at the beginning of the API section.

SAR10_SetRunMode

Description:

Sets the ADC to free run or one shot mode.

C Prototype:

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

Assembly:

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

Parameters:

bRunMode is the run mode. Symbolic names are provided in C and assembly. Their associated values are given in the following table:

Symbolic Name	Value	Description
SAR10_ONESHOT	0x00	One-shot
SAR10_FREERUN	0x08	Free run

Return Value:

None.

Side Effects:

See Note ** at the beginning of the API section

Sample Firmware Source Code

The C code illustrated here shows you how to use the SAR10 User Module:

```
#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
}
}
```

The same code in assembly is:

```
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
```

Configuration Registers

Table 4. SAR_CR0_REG

Bit	7	6	5	4	3	2	1	0
Value	0	InputSelect				Ready	Start	Enable

This register is used to adjust the SAR10 ADC. It allows you to set the input, to start the ADC sampling, and enables the SAR10 block. The SAR_CR0_REG also represents the readiness of the converted data.

The Enable bit is modified by calling the SAR10_Start or SAR10_Stop API routine.

The Start bit is modified by calling the SAR10_Trigger API routine and starts a single ADC conversion.

The Ready bit is read by calling the SAR10_fIsDataAvailable API routine and determines the readiness of the converted data.

The InputSelect bits determine the input source. The value of these bits are determined by the choice made for the "ADC Input Channel Selection" parameter in the Device Editor. The value can be changed at runtime by calling the SAR10_SetADCChannel API.

Table 5. SAR_CR1_REG

Bit	7	6	5	4	3	2	1	0
Value	0	0	TriggerSource		ClockSource			AutoTrig

This register is used to adjust the SAR10 ADC. It allows you to set the clock and trigger source, to set autotrigger mode.

The TriggerSource bits determine the source of the outside trigger signal. The value of these bits is determined by the choice made for the "Select Auto Trigger Source" parameter in the Device Editor. The value can also be changed at runtime by calling the SAR10_SetTriggerSrc API.

The ClockSource bits determine the clock source. The value of these bits is determined by the choice made for the "ADC Input Channel Selection" parameter in the Device Editor. The value can also be changed at runtime by calling the SAR10_SetClk API.

The AutoTrig bit determines the autotrigger reading mode. The value of this bit is determined by the choice made for the "Auto Trigger Global Enable" parameter in the Device Editor. The value can also be changed at runtime by calling the SAR10_EnableAutoTrigger API.

Table 6. SAR_CR2_REG

Bit	7	6	5	4	3	2	1	0
Value	0	0	0	0	Freerun	0	0	0

This register is used to adjust SAR10 ADC, in particular allows to set the repeatedly running mode.

The Freerun bit determines the free run reading mode. The value of this bit is determined by the choice made for the "Run Mode" parameter under user module parameters in the Device Editor.The value can also be changed by calling the SAR10_SetRunMode API.

Table 7. SAR_DH_REG

Bit	7	6	5	4	3	2	1	0
Value	HighData							

This register contains the highest eight bits of ADC converted data for CY8C21x45,CY8C22x45 devices and CY8C28x45 device in left-justified mode.

This register contains the highest two bits of ADC converted data for CY8C28x45 device in right-justified mode.

HighData bits are read by calling the SAR10_iGetData.

Table 8. SAR_DL_REG

Bit	7	6	5	4	3	2	1	0
Value							LowData	

This register contains the least significant two bits of the ADC converted data for CY8C21x45, CY8C22x45, and CY8C28x45 devices in left-justified mode.

This register contains the most significant eight bits of ADC converted data for CY8C28x45 device in right-justified mode.

LowData bits are read by calling the SAR10_iGetData.

Table 9. SAR_CR3_REG

Bit	7	6	5	4	3	2	1	0
Value	LALIGN							

Version History

Version	Originator	Description
1.00	DHA	Initial version
2.00	DHA	VBG enabled as Input connection. Added DRC warning for max SAR10 input clock. Fixed sysclk dividers range.

Note PSoC Designer 5.1 introduces a Version History in all user module datasheets. This section documents high level descriptions of the differences between the current and previous user module versions.

Copyright © 2009-2012 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.