Analog Switched Capacitor PSoC Block Datasheet SCBLOCK V 2.4

Copyright © 2002-2015 Cypress Semiconductor Corporation. All Rights Reserved.

Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	Flash	RAM	
CY8C29/27/24/23/22x13, CY8CLED04/08/16, CY8CLED0xD, CY8CLED0xG, CY8CTST120, CY8CTMG120, CY8CTMA120, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28x43, CY8C28x52						
	0	0	1	20	0	

For one or more fully configured, functional example projects that use this user module go to www.cypress.com/psocexampleprojects.

Features and Overview

- Fully parameterized for custom development
- Custom block for prototypes
- Selectable power settings

The SCBLOCK User Module is an analog switched capacitor (SC) PSoC block that is fully parameterized. This allows for the creation of custom switched capacitor functions. Application Programming Interfaces (APIs) are included for SCBLOCK power management.

The SCBLOCK is formed with either a “C" or “D" type SC PSoC block.

These types are shown here:

Figure 1. SCBLOCK Type A (ASA) Block Diagram

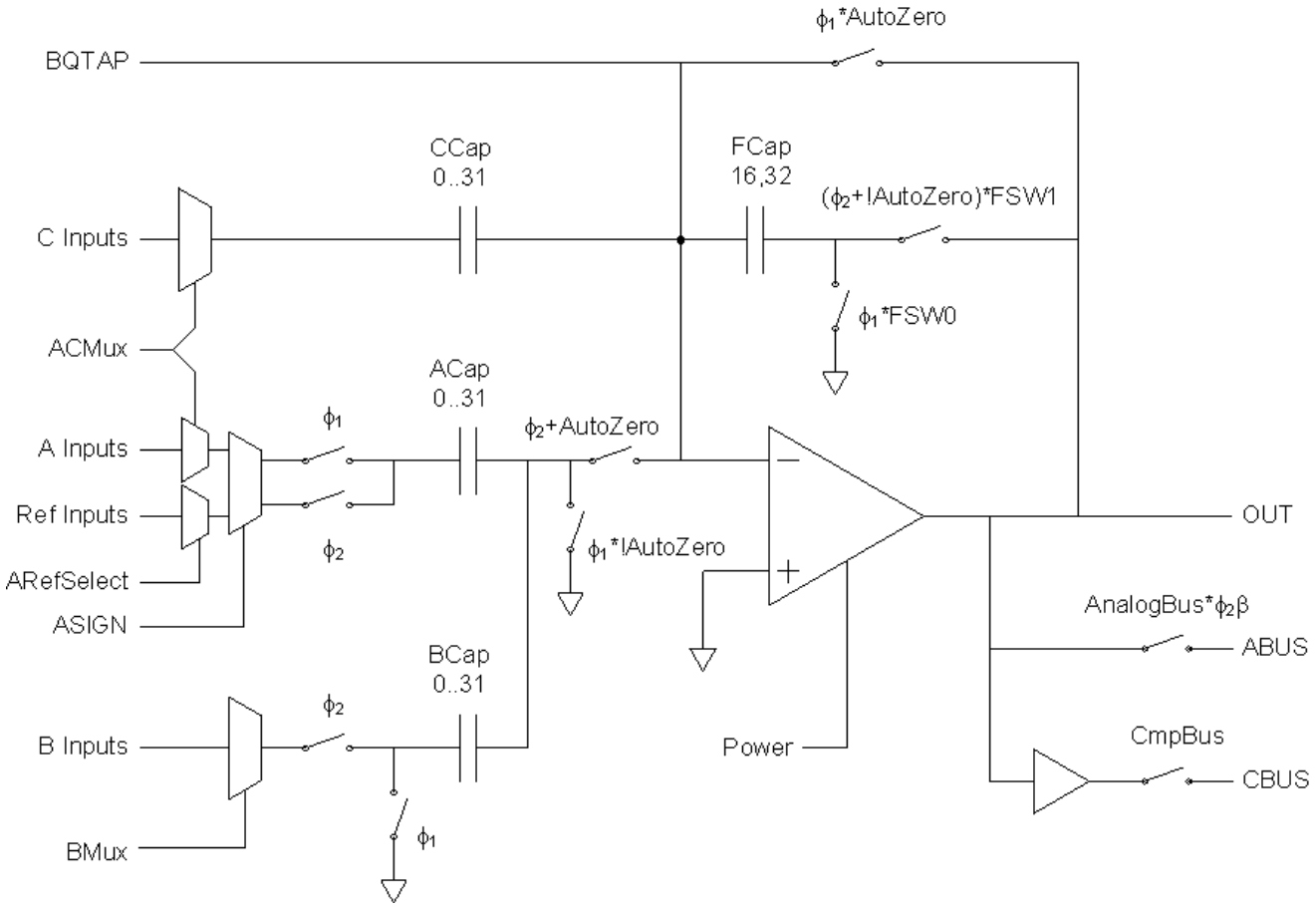


Figure 2. SCBLOCK Type B (ASB) Block Diagram

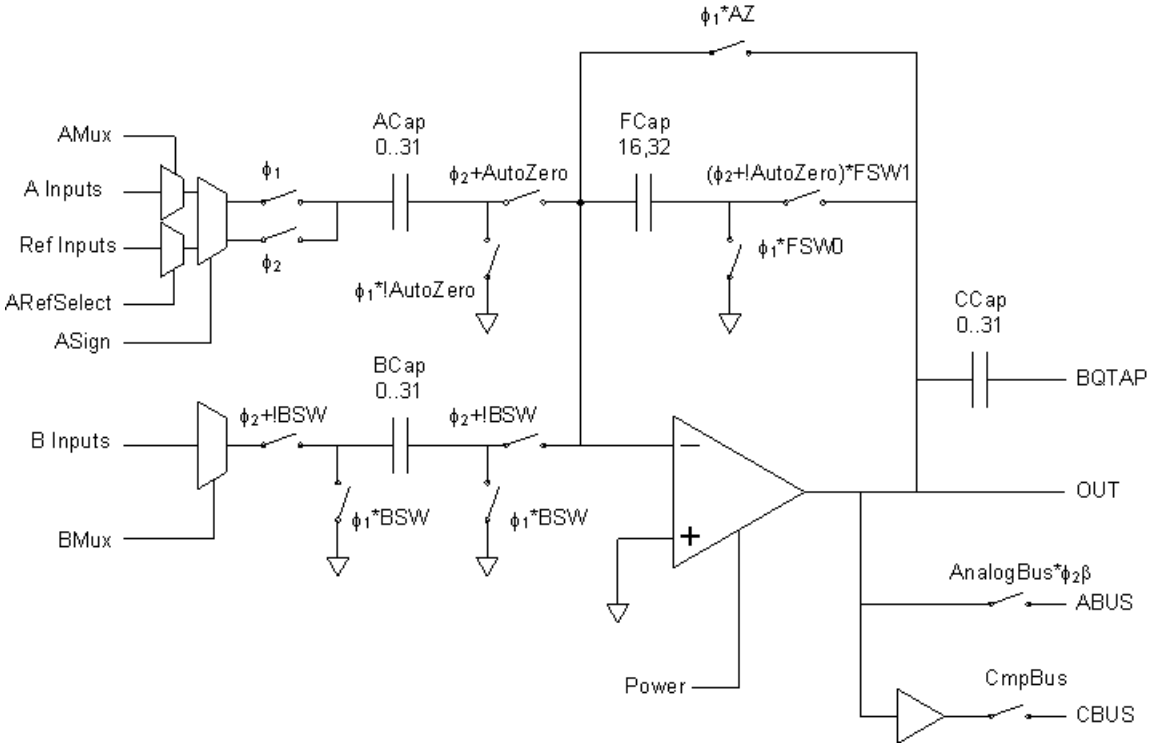


Figure 3. SCBLOCK Type C (ASC) Block Diagram

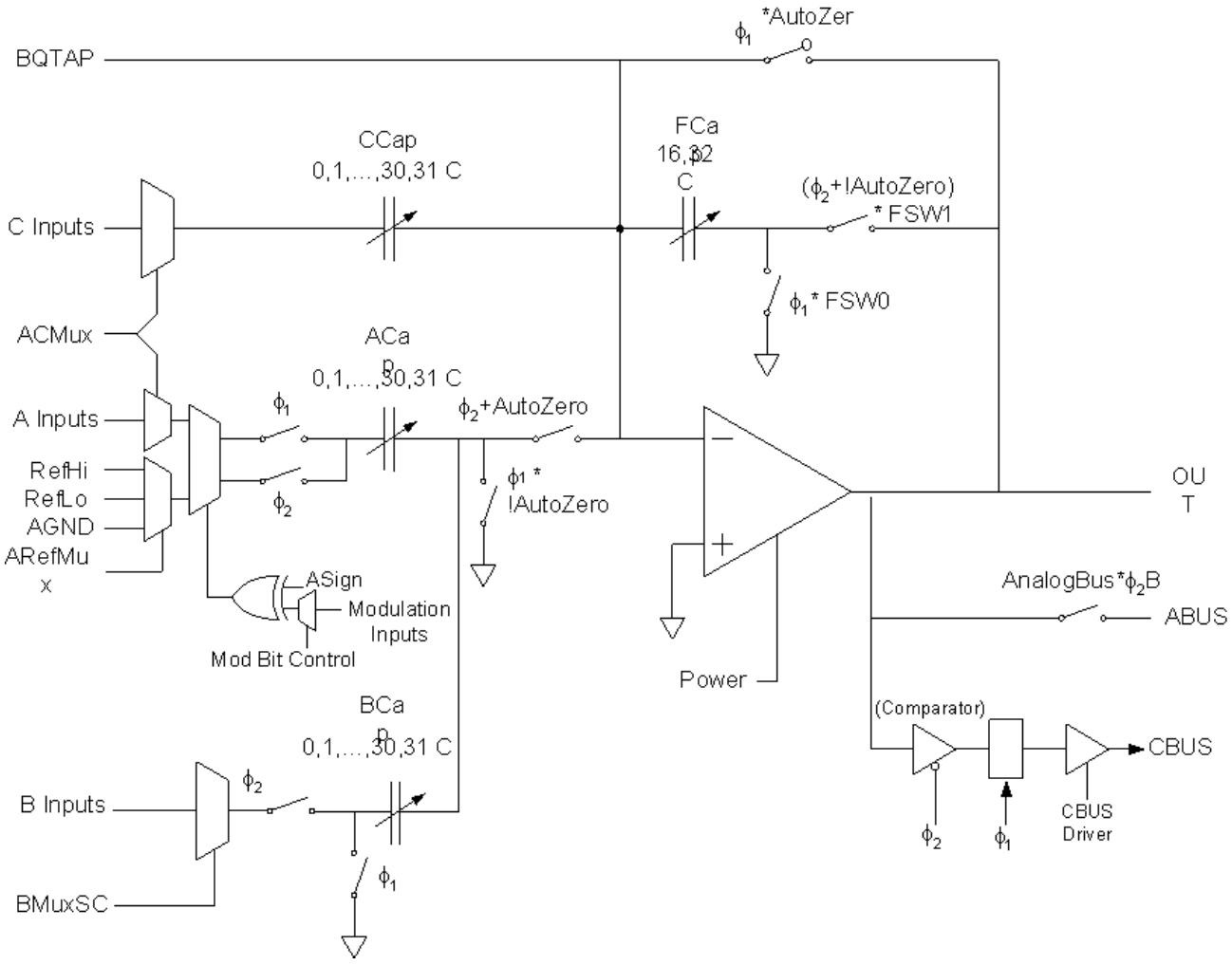
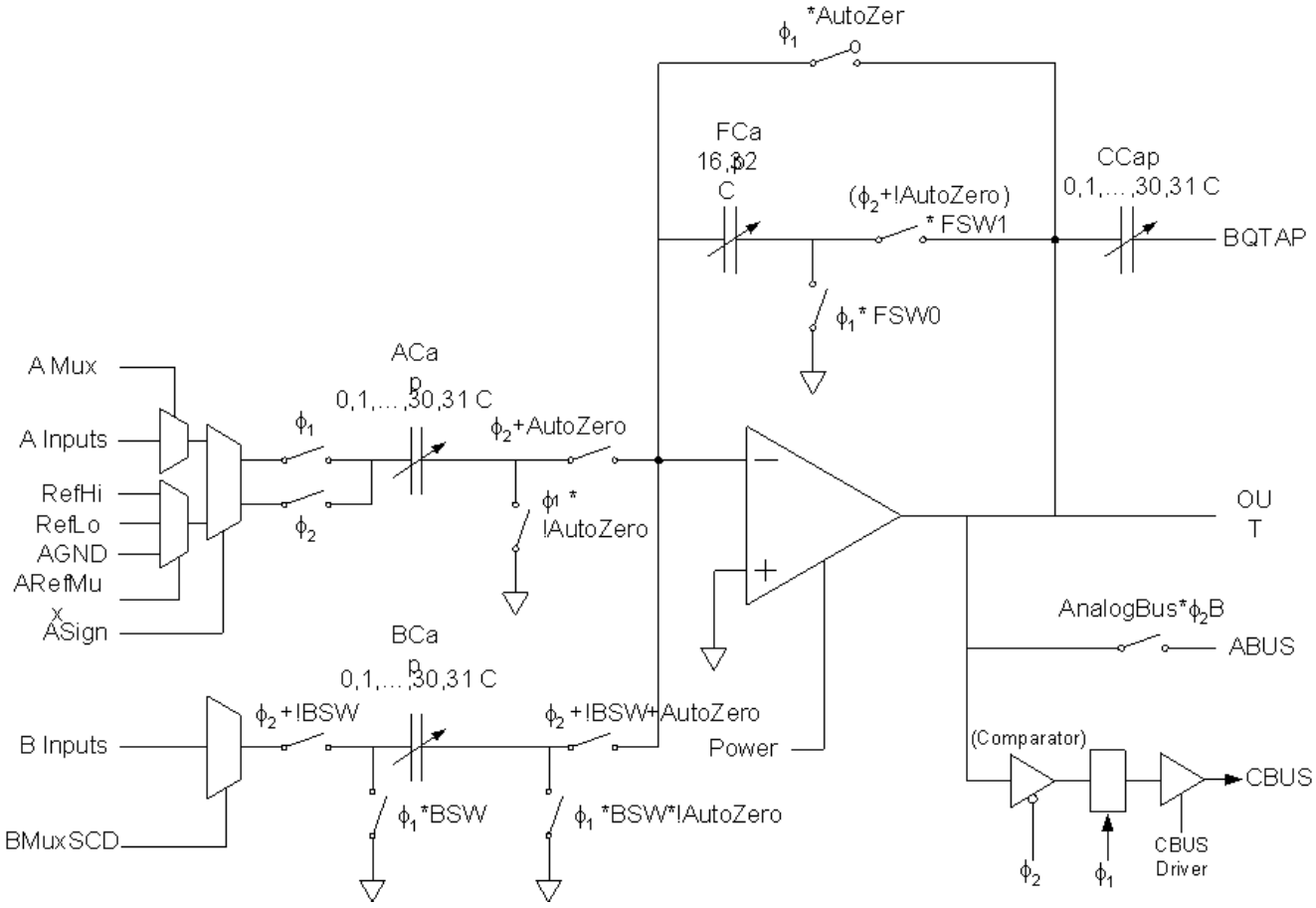


Figure 4. SCBLOCK Type D (ASD) Block Diagram



Functional Description

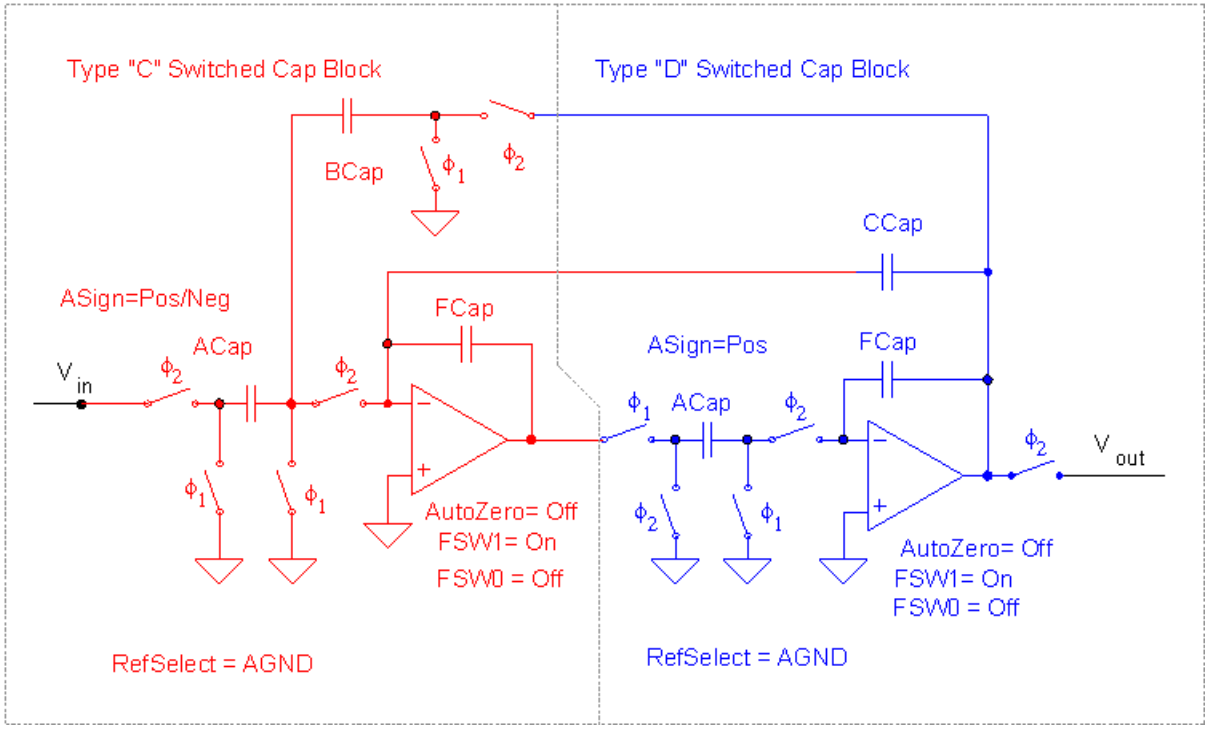
See the application note AN2041 - Understanding PSoC 1 Switched Capacitor Analog Blocks for a complete description of the PSoC switched capacitor blocks. The purpose of this user module is to enable the design of custom analog functions using switched capacitor analog PSoC blocks.

There are several differences between the ASC and ASD type SCBLOCK User Modules:

- The CCap branch is used as an input in the ASC type SCBLOCK, and as an output in the ASD type SCBLOCK.
- ASC type SCBLOCKs permit normal autozero operation for the BCap.
- ASC type SCBLOCKs have three selectable inputs, whereas ASD type SCBLOCKs have only two selectable inputs.

The ASC and ASD SCBLOCKs are designed so that they can be combined to create a single biquad filter. An ASC block is used for the input stage of the biquad filter and the ASD block is used for the output stage. The following figure shows how two user modules, an ASC and an ASD SCBLOCK User Module, can be connected to form a low pass biquad filter. The ASC and ASD type SCBLOCKs are improved versions of the ASA and ASB type SCBLOCKs. The ASC and ASD type SCBLOCKs have more input selection options and the opamps have improved electrical characteristics.

Figure 5. Low Pass Biquad Filter Using ASC and ASD Blocks



DC and AC Electrical Characteristics

Unless otherwise specified in the following table, all limits guaranteed for T_A = -40°C to, +85°C, V_{dd} = 5.0V +/- 5%.

Table 1. SCBLOCK DC and AC Electrical Characteristics

Parameter	Conditions and Notes	Typical	Limit	Units
Capacitor Unit Value		70		fF
Capacitor Matching		0.1		%
F_{max}	Clock at the SCBLOCK[1]	2[1]		MHz
SUPPLY CURRENT				
Bias = Low		125		uA
Bias = Medium		280		uA
Bias = High		760		uA

Electrical Characteristics Notes

1. The clock signal is passed through a 1:4 divider, between the analog clock mux and the SCBLOCK. To provide a 2 MHz clock to the SCBLOCK, an 8 MHz clock must be connected to the analog clock mux.

Placement

The SCBLOCK block can be placed in any of the switched capacitor PSoC blocks. Differentiation between ASC and ASD type blocks is dependent upon placement.

Parameters and Resources

This section discusses the parameters for the SCBLOCK. Note that some parameters are specific to either ASC or ASD type SCBLOCK User Modules.

FCap

- **16** - Cap set to 16 units
- **32** - Cap set to 32 units

ClockPhase

- **Norm** - Normal phasing
- **Swap** - φ1 and φ2 clocks are swapped

ASign

- **Pos** - Cap connected to the A Input during φ1 and the Ref Input during φ2
- **Neg** - Cap connected to the Ref Input during φ1 and the A Input during φ2

ACap

Sets the value of ACap between 0 and 31 units.

ACMux (ASC)

Selects which A Input is connected to the ACap and which C Input is connected to CCap. PSoC Designer presents the available connection options when the SCBLOCK is placed in a particular block.

Note This parameter is only applicable when the SCBLOCK component is placed in an ASC block.

AMux (ASD)

Selects which A Input is connected to ACap. PSoC Designer presents the available connection options when the SCBLOCK is placed in a particular block.

Note This parameter is only applicable when the SCBLOCK component is placed in an ASD block.

BCap

Sets the value of BCap between 0 and 31 units.

AnalogBus

- **Disable** - The analog output is not connected to the analog output bus, freeing the output bus for other user modules.
- **Enable** - The analog output is connected to the analog output bus for its column and may be routed to the analog buffer.

Note Only one analog block output per column may connect to the analog bus.

CompBus

- **Disable** - The comparator output is not connected to the comparator output bus, freeing the comparator bus for other user modules placed in the same column.

- **Enable** - The comparator output is connected to the comparator output bus for the column and may be routed to the digital resources.

Note Only one analog block comparator output per column may connect to the comparator bus.

AutoZero

- **Off** - Autozero function is disabled.
- **On** - Autozero is enabled. The output is connected to the negative input during φ1, to measure the opamp's offset impedance. During φ2, the actual signal is compensated for this offset.

CCap

Sets the value of CCap between 0 and 31 units.

ARefMux

- **AGND** - ACap is connected to the AGND during φ2
- **REFHI** - ACap is connected to the REFHI during φ2
- **REFLO** - ACap is connected to the REFLO during φ2
- **ComparatorBus_x** - During φ2 the ACap is connected to REFHI when the comparator is high, or REFLO when the comparator is low.

FSW1

- **Off** - FCap is disconnected from the feedback loop
- **On** - FCap is connected in the feedback loop

FSW0

- **Off** - FCap is not discharged during φ1
- **On** - FCap is discharged during φ1

BSW (ASD)

- **Off** - BCap is not connected as a capacitor
- **On** - BCap is connected as a capacitor

BMux

Selects which B Input is connected to BCap.

Power

- **Off** - SCBLOCK is powered down
- **Low** - SCBLOCK set for lowest power
- **Med** - SCBLOCK set for medium power
- **High** - SCBLOCK set for full power

Application Programming Interface

The Application Programming Interface (API) routines are provided as part of the user module to allow the designer 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 SCBLOCK_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 is shortened to SCBLOCK for simplicity.

Note

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 ensure their code observes the policy, too. Though some user module API function may leave A and X unchanged, there is no guarantee they may do so in the future.

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.

SCBLOCK_Start

Description:

Sets the power level and performs all required initialization for this user module.

C Prototype:

```
void  SCBLOCK_Start (BYTE bPowerSetting)
```

Assembly:

```
mov   A, bPowerSetting
lcall  SCBLOCK_Start
```

Parameters:

bPowerSetting: One byte that specifies the power level. Following reset and configuration, the analog PSoC block assigned is powered down. Symbolic names provided in C and assembly, and their associated values, are given in the following table,

Symbolic Name	Value
SCBLOCK_OFF	0
SCBLOCK_LOWPOWER	1
SCBLOCK_MEDPOWER	2
SCBLOCK_HIGHPOWER	3

The power level has an effect on analog performance. The correct power setting has to be determined for each application.

Return Value:

None

Side Effects:

The A and X registers may be altered by this function.

SCBLOCK_SetPower

Description:

Sets the power level for the SCBLOCK User Module.

C Prototype:

```
void  SCBLOCK_SetPower (BYTE bPowerSetting)
```

Assembly:

```
mov   A, bPowerSetting
lcall  SCBLOCK_SetPower
```

Parameters:

bPowerSetting: Same as the bPowerSetting parameter used for the Start entry point. Allows the user to change the power level while operating the block.

Return Value:

None

Side Effects:

The A and X registers may be altered by this function.

SCBLOCK_Stop

Description:

Sets the power level to 0FF.

C Prototype:

```
void  SCBLOCK_Stop()
```

Assembly:

```
lcall  SCBLOCK_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be altered by this function.

Sample Firmware Source Code

Here is a sample program using SCBLOCK written in C:

```
//
//  This sample shows how to create a inverter of analog signal.
// // OVERVIEW:
////  In this example the SCBLOCK input is routed to P2[1] and the SCBLOCK output is
//  routed to P0[3].
//  The pin P0[3] has the inverting analog signal from pin P2[1].
//
//The following changes need to be made to the default settings in the Device Editor:
//
//   1. Select SCBLOCK user module.
//   2. The User Module will occupy the space in dedicated system resources.
//   3. Rename User Module's instance name to SCBLOCK.
//   4. Set SCBLOCK's ACMux Parameter to Port_2_1.
//   4. Set SCBLOCK's AnalogBus Parameter to AnalogOutBus_0.
//   5. Click on AnalogOutBuf_0 and connect AnalogOutBuf_0 to Port_0_3.
//
// CONFIGURATION DETAILS:
//
//  The UM's instance name must be shortened to SCBLOCK.
//
// PROJECT SETTINGS:
//
//
// USER MODULE PARAMETER SETTINGS:
//
//  ------------------------------------------------------------------------------
//  UM         Parameter         Value                 Comments
//  ------------------------------------------------------------------------------
//  SCBLOCK    Name              SCBLOCK               UM's instance name
//             FCap              16
//             ClockPhase        Norm
//             ASing         Neg
//             ACap   16
//             ACMux             Port_2_1
//             BCap         0
//             AnalogBus         Analogokutbus_0
//             CompBus           Disable
//             AutoZero          On
//             CCap          0
//             ARefMux         AGND
//             FSW1          On
//             FSW0              On
//             BMux          Port_2_3
//             Power          Low
//
//  ------------------------------------------------------------------------------
/* Code begins here */
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules
void main(void)
{
    SCBLOCK_Start(SCBLOCK_LOWPOWER);    // Turn on SCBlock power
```

```
    // User code
}
```

Here is the same code written in Assembly:

```
;
;  This sample shows how to create a inverter of analog signal.
;
; OVERVIEW:
;
;  In this example the SCBLOCK input is routed to P2[1] and the SCBLOCK output is routed
; to P0[3].
;  The pin P0[3] has the inverting analog signal from pin P2[1].
;
;The following changes need to be made to the default settings in the Device Editor:
;
;   1. Select SCBLOCK user module.
;   2. The User Module will occupy the space in dedicated system resources.
;   3. Rename User Module's instance name to SCBLOCK.
;   4. Set SCBLOCK's ACMux Parameter to Port_2_1.
;   4. Set SCBLOCK's AnalogBus Parameter to AnalogOutBus_0.
;   5. Click on AnalogOutBuf_0 and connect AnalogOutBuf_0 to Port_0_3.
;
; CONFIGURATION DETAILS:
;
;  The UM's instance name must be shortened to SCBLOCK.
;
; PROJECT SETTINGS:
;
;
; USER MODULE PARAMETER SETTINGS:
;
;  ------------------------------------------------------------------------------
;  UM         Parameter         Value                Comments
;  ------------------------------------------------------------------------------
;  SCBLOCK    Name              SCBLOCK              UM's instance name
;             FCap              16
;             ClockPhase        Norm
;             ASing         Neg
;             ACap  16
;             ACMux             Port_2_1
;             BCap        0
;             AnalogBus         Analogokutbus_0
;             CompBus           Disable
;             AutoZero          On
;             CCap          0
;             ARefMux        AGND
;             FSW1          On
;             FSW0              On
;             BMux          Port_2_3
;             Power         Low
;
;  ------------------------------------------------------------------------------
; Code begins here
```

```
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules

export _main

_main:

    mov A, SCBLOCK_LOWPOWER
    lcall SCBLOCK_Start    ; Turn on SCBlock power

.terminate:
    ; Insert your main assembly code here.
    jmp .terminate
```

Configuration Registers

Table 2. Block SCBLOCK: Register CR0

Bit	7	6	5	4	3	2	1	0
Value	FCap	ClockPhase	ASign	ACap				

Table 3. Block SCBLOCK: Register CR1

Bit	7	6	5	4	3	2	1	0
ASC	ACMux			BCap				
ASD	AMux			BCap				

ACMux is used when the block is placed in an ASC block. AMux is used when the block is placed in an ASD block. Both field values depend on how the user connects the input.

Table 4. Block SCBLOCK: Register CR2

Bit	7	6	5	4	3	2	1	0
Value	AnalogBus	CompBus	AutoZero	CCap				

Table 5. Block SCBLOCK: Register CR3

Bit	7	6	5	4	3	2	1	0
ASC	ARefMux		FSW1	FSW0	BMuxASC		Power	
ASD	ARefMux		FSW1	FSW0	BSW	BMuxASD	Power	

BMuxASC is used when the block is placed in an ASC block and depends on how the user connects the input. BMuxASD is used when the block is placed in an ASD block and depends on how the user connects the input. BSW is used when the block is placed in an ASD block. The value determines if BCap switches are active.

Version History

Version	Originator	Description
2.4	DHA	Added Version History

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 © 2002-2015 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.