Easy ADC Datasheet EzADC V 1.00

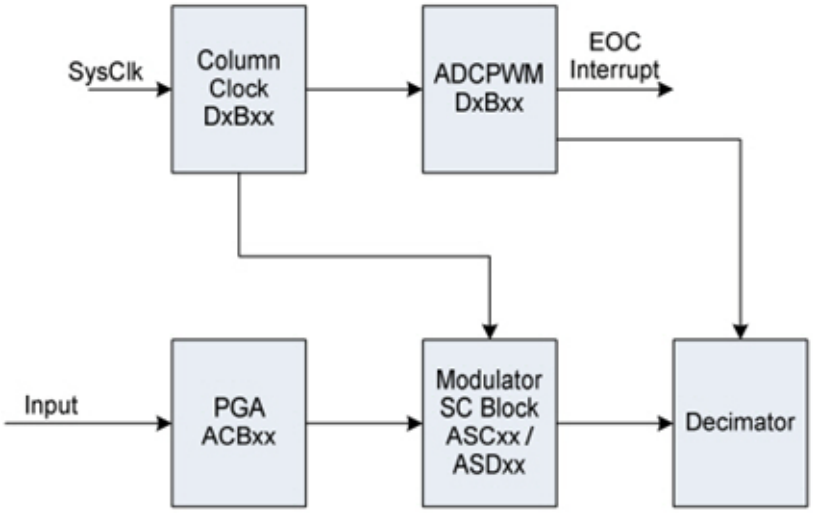
Copyright © 2012-2013 Cypress Semiconductor Corporation. All Rights Reserved.

MUM	PSoC® Blocks				API Memory (Bytes)				Pins
	Digital	Analog CT	Analog SC		Flash		RAM		
Modulators (1st or 2nd Order)		1st and 2nd	1st	2nd	1st	2nd	1st	2nd	
Supported Devices: CY8C24x23A, CY8C27x43, CY8C24x94, CY8C29x66, CY8C28x23,CY8C28x33, CY8C28x43, CY8C28x45, CY8C28x52									
Configuration	2	1	1	2	365	413	11	11	1

Features and Overview

- Incremental ADC supporting first order and second order modulator
- Selectable sample rate 0.007–15.625 ksps
- Selectable resolution 6–14 bits
- Selectable gain 1x–48x
- Selectable reference
- Automatic clock calculation
- Signed and unsigned output data format
- Enables AGND output to AnalogBus
- Offset error compensation
- Configurable offset error compensation frequency

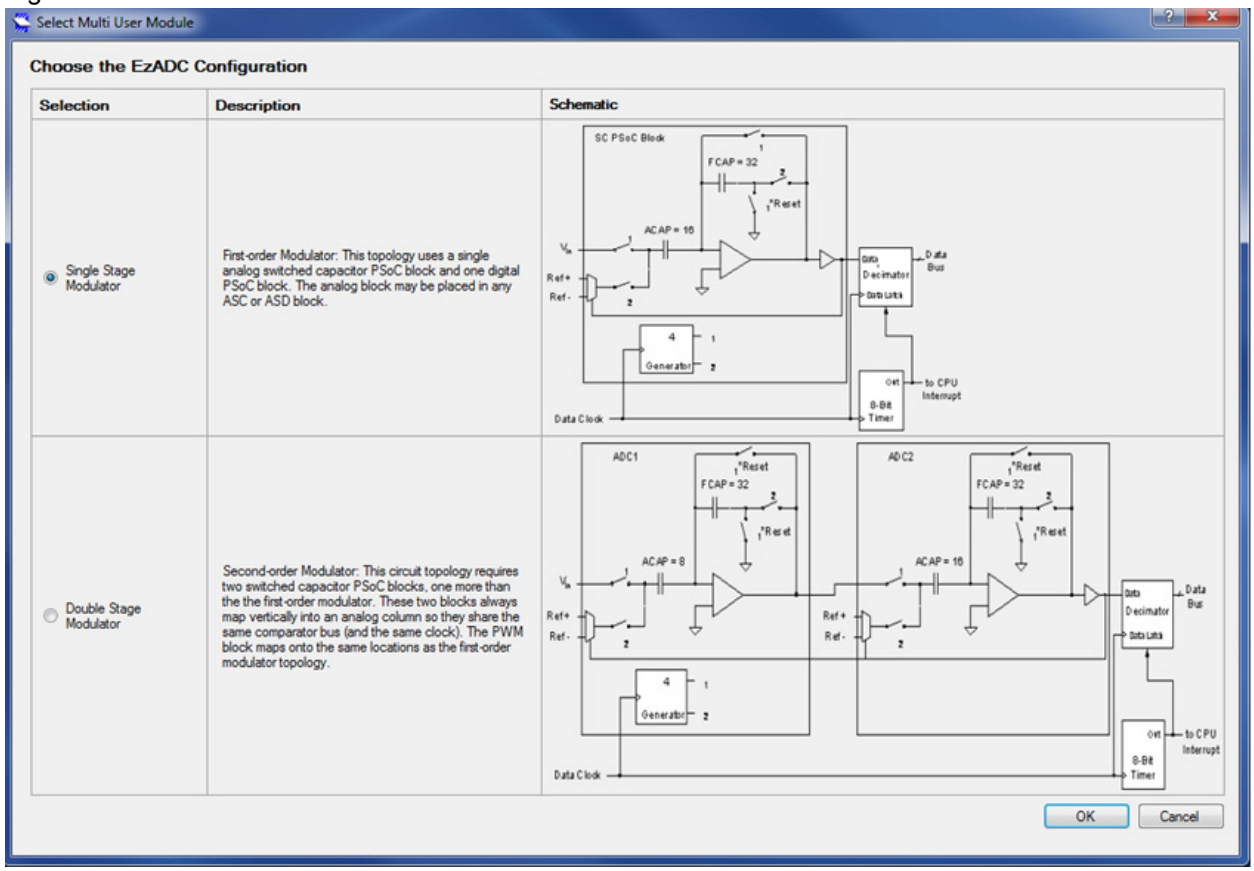
Figure 1. EzADC Block Diagram



Functional Description

The EzADC User Module has two configurations: first order modulator and second order modulator.

Figure 2. EzADC MUM Wizard



As shown in Figure 2, the first order modulator uses a single analog switched-capacitor PSoC block. The analog block may be placed in any analog switched capacitor (ASC) or an ASC Type D (ASD) block.

The second order modulator requires two switched-capacitor PSoC blocks. Because these two blocks always map vertically into an analog column, they share the same comparator bus (and the same clock). The EzADC User Module comprises of a programmable gain amplifier (PGA), a clock divider (Counter8), a decimator (Integrator), an ADC modulator (depending on the selected order, it can consume one or two SC blocks), a pulse-width modulator (PWM8), and AnalogClock_0_Select MUX.

The clock divider is clocked by the system clock (SysClk) and is used to provide the clock to the PWM and SC blocks. The output of the clock divider is routed to the analog column and the PWM. The user module consumes two consecutive digital blocks.

The ADC is formed by the modulator (analog SC blocks), decimator, and the PWM. The PWM is used to set the integration time based on the selected resolution. At the end of conversion, the PWM latches the decimator output and also generates an interrupt, signaling the end of conversion.

The positive input of the ADC SC modulator block is connected to the output of the CT block, which is located in the same column as that of the SC blocks. The CT block is configured as a PGA. The CT block's reference and input can be configured in the User Module Parameters window. The EzADC User Module has only one configurable input in the Interconnect View – Input. The possible ADC input values are defined by possible PMux inputs that depend on the occupied ACB block.

All analog blocks are always located in the same analog column. The AnalogColumn_Clock MUX is always connected to the AnalogClock_0_Select MUX. The EzADC User Module with first order modulator can be placed at the top and middle blocks of any column. The user module with second order modulator consumes three analog blocks in an analog column.

DC and AC Electrical Characteristics

Table 1. EzADC Electrical Characteristics

Parameter	Min	Typ	Max	Units	Conditions and Notes
V_{IN}	–	–	V_{SS} to V_{DD}	V	
Resolution	–	–	–	Bits	
Sample rate	–	–	–	ksps	
G_{48}	–	3.0	–	%	Buffer gain = 48; offset comp enabled
G_{24}	–	2.2	–	%	Buffer gain = 24; offset comp enabled
G_{16}	–	1.5	–	%	Buffer gain = 16; offset comp enabled
G_4	–	0.7	–	%	Buffer gain = 4; offset comp enabled
G_1	–	0.5	–	%	Buffer gain = 1; offset comp enabled
V_{OS}	–	4.5	–	mV	Buffer gain = 1; offset comp disabled
$V_{OS\ COMP}$	–	0.5	–	mV	Buffer gain = 1; offset comp enabled
$I_{leakage}$	–	1	–	nA	
C_{IN}	–	3	–	pF	
PSRR	–	73	–	dB	
I_{oper}	–	272 1380 5452	–	µA	Low Power Medium Power High Power
DNL	–	0.1	–	LSB	Column clock = 2 MHz
INL	–	0.5	–	LSB	Column clock = 2 MHz
SNR	–	46	–	dB	

Placement

The EzADC with the first order modulaotor requires one CT block, one SC block, and two digital blocks. The analog blocks may be placed in any analog column occupying the top and middle rows.The two digital blocks occupy consecutive blocks in any digital row.The second order modulator requires one CT block and two SC blocks and, therefore, occupies a complete analog column.

Parameters and Resources

ADC Resolution

This parameter sets the resolution of the ADC. The range is 6 to 14 bits.

ADC Sample Rate

This parameter sets the sample rate of the ADC.

Resolution	Sample Rate Choices (ksps)
6	3.906, 1.953, 0.976
7	2.604, 1.302, 0.651
8	1.562, 0.781, 0.390
9	0.868, 0.434, 0.217
10	0.459, 0.229, 0.114
11	0.236, 0.118, 0.059
12	0.120, 0.060, 0.030
13	0.060, 0.030, 0.015
14	0.030, 0.015, 0.007

Note All the above sample rates have been calculated based on data clocks of 2 MHz, 1 MHz, and 500 KHz using the following formula:

$$SampleRate = \frac{DataClock}{256 \cdot (2^{Bits-6} + 1)}$$

Buffer Gain

This parameter sets the PGA Gain. The available gain options are 48.000, 24.000, 16.000, 8.000, 5.333, 4.00, 3.200, 2.667, 2.286, 2.000, 1.777, 1.600, 1.455, 1.333, 1.231, 1.143, 1.062, and 1.000.

Reference

This parameter sets the reference for the PGA. VSS and AGND are the reference options

Data Format

This parameter sets the data format. The data can be set to signed or unsigned formats.

AGND Output

This parameter enables or disables the AGND output to the analog bus. This option can be used to bring out the internal AGND to connect bipolar signals to the ADC.

Offset Compensation

This parameter enables or disables correlated double sampling. For more details on correlated double sampling, refer to AN2226. To perform offset compensation, set PGA reference to AGND, connect the PGA input to AGND, and perform a single conversion. Store the output of the ADC in a RAM variable. During all other measurements, this offset value is subtracted from the ADC result.

Offset Compensation Frequency

This parameter sets the number of ADC samples after which offset is corrected. The best results can be obtained by setting this parameter to one in two samples but this reduces the throughput of the ADC by half.

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.

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.

EzADC_Start

Description:

Initializes and starts the EzADC User Module resources.

C Prototype:

```
void  EzADC_Start (BYTE bPowerSetting)
```

Assembly:

```
mov   A, bPowerSetting
lcall  EzADC_Start
```

Parameters:

bPowerSetting: sets ADC and PGA operating power.

Symbolic Name	Value
EzADC_LOWPOWER	1
EzADC_MEDPOWER	2
EzADC_HIGHPOWER	3

Return Value:

None

EzADC_Stop

Description:

Stops the EzADC User Module resources.

C Prototype:

```
void  EzADC_Stop (void)
```

Assembly:

```
lcall  EzADC_Stop
```

Parameters:

None

Return Value:

None

EzADC_SetBufferGain

Description:

Sets the gain of the PGA.

Symbolic Name	Value
EzADC_G48_00	0Ch
EzADC_G24_00	1Ch
EzADC_G16_00	08h
EzADC_G8_00	18h
EzADC_G5_33	28h
EzADC_G4_00	38h
EzADC_G3_20	48h
EzADC_G2_67	58h
EzADC_G2_27	68h
EzADC_G2_00	78h
EzADC_G1_78	88h
EzADC_G1_60	98h
EzADC_G1_46	A8h
EzADC_G1_33	B8h

Symbolic Name	Value
EzADC_G1_23	C8h
EzADC_G1_14	D8h
EzADC_G1_06	E8h
EzADC_G1_00	F8h

C Prototype

```
void EzADC_SetBufferGain(BYTE bGainSetting)
```

Assembly

```
mov A, bGainSetting
lcall EzADC_SetBufferGain
```

Parameters

None

Return Value

None

EzADC_GetSamples

Description:

Starts ADC conversion.

C Prototype:

```
void EzADC_GetSamples (BYTE bNumSamples)
```

Assembly:

```
mov A, bNumSamples
lcall EzADC_GetSamples
```

Parameters:

None

Return Value:

None

EzADC_fIsDataAvailable

Description:

This function checks if the EzADC conversion is complete.

C Prototype:

```
BYTE EzADC_fIsDataAvailable(void)
```

Assembly:

```
lcall EzADC_fIsDataAvailable
```

Parameters:

None

Return Value:

Returns a zero if conversion is not complete and one if conversion is complete.

EzADC_iGetDataClearFlag

Description:

Reads ADC result and clears the ADC ready flag.

C Prototype:

```
INT   EzADC_iGetDataClearFlag(void)
```

Assembly:

```
lcall  EzADC_iGetDataClearFlag
```

Parameters:

None

Return Value:

Returns the ADC result in integer.

EzADC_SetReference

Description:

Sets the reference to the PGA.

C Prototype:

```
void EzADC_SetReference(BYTE bReferenceSetting)
```

Assembly:

```
mov A, bReferenceSetting
lcall EzADC_SetReference
```

Parameters:

bReference: Reference of the PGA

Symbolic Name	Value
EzADC_REF_VSS	0x10
EzADC_REF_AGND	0x01

Return Value

None

Sample Firmware Source Code

The following is the C sample code for this user module.

```
//-------------------------------------------------------------------------
// C main line
//-------------------------------------------------------------------------
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules
int ADCResult;
void main(void)
{
    M8C_EnableGInt;                             // Enable Global Interrupts
    EzADC_Start(EzADC_HIGHPOWER);               // Apply power to ADC
    EzADC_SetBufferGain(EzADC_G2_00);           // Set Gain
    EzADC_GetSamples(0);                        // Have ADC run continuously
    for(;;)
    {
        while (!(EzADC_fIsDataAvailable()));    // Loop until value ready
        ADCResult = EzADC_iGetDataClearFlag();  // Reset ready flag and get data
        // Add user code here to use or display result
    }
}
```

Here is equivalent code written on assembly:

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

export _main

area bss(RAM)
ADCResult    : blk 2

area text(ROM,REL)
_main:

    M8C_EnableGInt                  ; Enable Global Interrupts
    mov A, EzADC_HIGHPOWER
    lcall EzADC_Start               ; Apply power to ADC

    mov A, EzADC_G2_00
    lcall EzADC_SetBufferGain       ; Set Gain

    mov A, 0
    lcall EzADC_GetSamples       ; Have ADC run continuously
loop1:
wait:
    lcall EzADC_fIsDataAvailable ; Loop until value ready
    jz wait
```

```
    call EzADC_iGetDataClearFlag ; Reset ready flag and get data

    mov [ADCResult], X
    mov [ADCResult + 1], A
    ; Add user code here to use or display result

jmp loop1
```

Version History

Version	Originator	Description
1.0	KUK	Initial release

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 © 2012-2013 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.