12-Bit Incremental ADC Datasheet ADCINC12 V 5.3

Copyright © 2002-2013 Cypress Semiconductor Corporation. All Rights Reserved.

Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	Flash	RAM	
CY8C29/27/24/22x13, CY8C23x33, CY8CLED04/08/16, CY8C28x45, CY8C28x43, CY8C28x52	2	0	1	224	6	1
CYWUSB6953	0	0	0	25	0	1 to 4

See AN2239, ADC Selection Guide for other converters.

Features and Overview

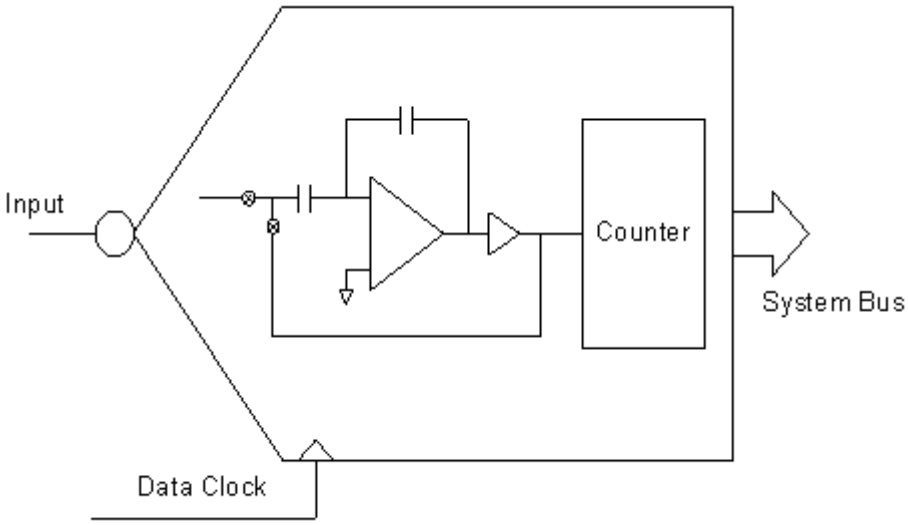
- 12-bit resolution, 2's complement
- Sample rate from 7.8 sps to 480 sps
- Input range AGND +/- V_{Ref}
- Provides normal mode rejection of high frequency harmonics
- Internal or external clock

The ADCINC12 User Module implements a 12-bit incremental A/D that generates a 12-bit, full-scale 2's complement output (+2047 to -2048 count range) with several input ranges to select from. Input voltage ranges, including rail-to-rail, may be measured by configuring the proper reference voltage and analog ground. It supports sample rates from 7.8 sps to 480 sps. The ADCINC12 programming interface allows you to select from 0 to 255 samples, where zero specifies continuous sampling.

The ADCINC12 is an integrating ADC that provides removal of higher frequencies. Optimum rejection of 50 Hz, 60 Hz, and any harmonics of these two frequencies (normal mode rejection) can be achieved by setting the sample window to 100 ms (sample rate to 9.84 sps). The CPU load varies with the input level.

For example, when V_{in} = +V_{ref}, there are 249 CPU cycles (maximum 13 bit). When V_{in} = AGND, there are 47 CPU cycles (average 13 bit). When V_{in} = -V_{ref}, there are 43 CPU cycles (minimum 7-13 bit).

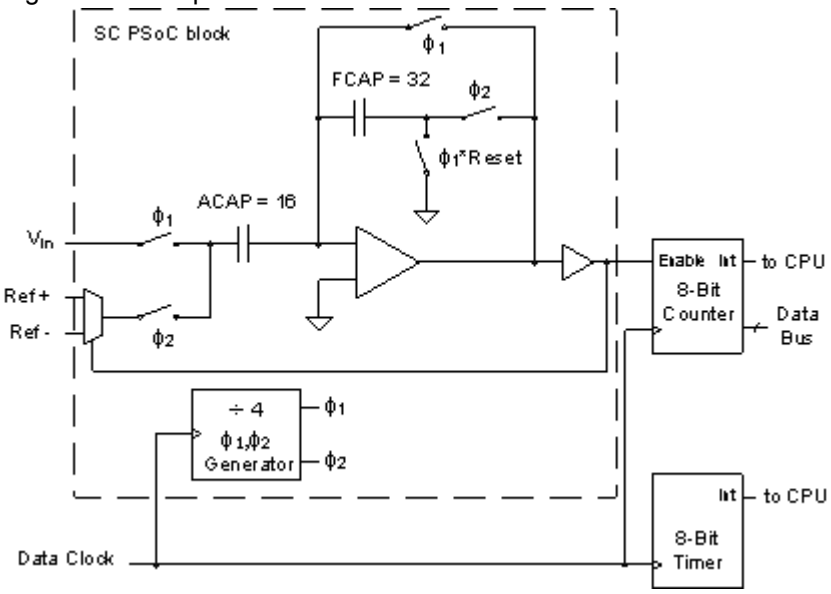
Figure 1. ADCINC12 Block Diagram



Functional Description

The ACDINC12 User Module is formed from a single analog switched capacitor PSoC block and two digital PSoC blocks, as shown in Figure 2.

Figure 2. Simplified Schematic of the ADCINC12



The analog block is configured as a integrator that can be reset. Depending on the output polarity, the reference control is configured so that the reference voltage is either added or subtracted from the input and placed in the integrator. This reference control attempts to pull the integrator output back towards AGND. If the integrator is operated 4096 (2^{12}) times and the output voltage comparator is positive “n" of those times, the residual voltage (V_{resid}) at the output is:

Equation 1

$$V_{resid} = 2^{12} \times V_{in} - n \times V_{Ref} + (2^{12} - n) \times V_{Ref}$$

Equation 2

$$V_{in} = \frac{n - 2^{12-1}}{2^{12-1}} V_{ref} + \frac{V_{resid}}{2^{12}}$$

The value calculated is an ideal value and may differ based on system noise and chips offsets.

This equation states that the range of this ADC is $\pm V_{Ref}$, the resolution (LSB) is $V_{Ref}/2048$, and the voltage on the output at the end of the computation is defined as the residue. Since V_{resid} is always less than V_{Ref}, $V_{resid}/4096$ is less than half a LSB and can be ignored. The resulting equation is Equation 3:

Equation 3

$$V_{in} = \frac{n - 2048}{2048} V_{ref}$$

Example 1

For a V_{ref} of 1.3V you can easily calculate the input voltage based on the value read from the incremental ADC at the time the data is ready. This calculation is done using Equation 4:

Equation 4

$$V_{in} = \frac{n - 2048}{2048} 1.3$$

The result of the calculation is referenced to AGND. For a ADC data value of 1000, you can calculate the Voltage measured to be -0.66V. This is shown in Equation 5:

Equation 5

$$V_{in} = \frac{1000 - 2048}{2048} 1.3 = -0.66\,V$$

The value calculated is an ideal value and may differ based on system noise and chips offsets.

To determine the expected code given a specific input voltage, you can rearrange the equation to give Equation 6:

Equation 6

$$n = \frac{2048 \cdot V_{in}}{V_{ref}} + 2048$$

Example 2

For a V_{ref} of 1.3V, you can easily calculate the expected ADC code based on the input Voltage. This calculation is done using Equation 7:

Equation 7

$$n = \frac{2048 \cdot V_{in}}{V_{ref}} + 2048$$

For an input voltage of 1V above AGND, the code from the ADC can be expected to be 3623.38. This calculation is based on Equation 8:

Equation 8

$$n = \frac{2048 \cdot 1}{1.3} + 2048 = 3623.38$$

The value calculated is an ideal value and may differ based on system noise and chips offsets.

Incremental ADC

The following digital resources are used to make the integrator function as an incremental ADC:

- A timer to count the proper number of integration cycles.
- A counter to accumulate the number of cycles that the output comparator is positive.

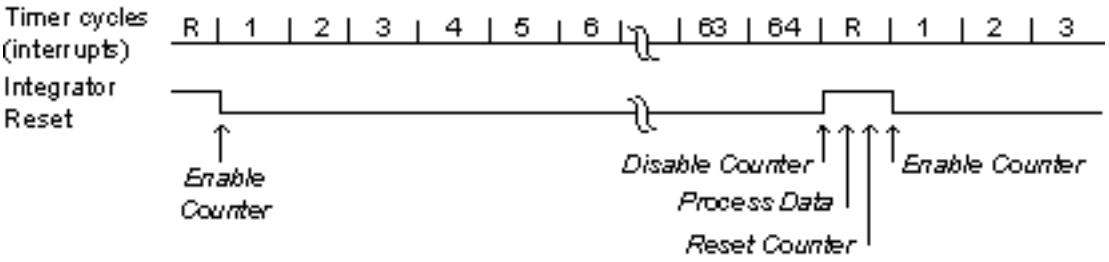
Typically, building a 12-bit incremental ADC converter requires a 12-bit timer and a 12-bit counter. Both the timer and counter would have to be clocked at one-fourth the frequency used to clock the column clock feeding the integrator. For this implementation, the timer and the counter are clocked with the same clock used by the column clock for the integrator block. The counter is incremented by four each integration cycle where the output comparator is positive, requiring that it be able to accumulate 14 bits worth of data. The timer must also be four times longer (14 bits).

Note **CAUTION:** It is imperative that when placing this module, it must be configured with the same clock for all three blocks. Failure to do so causes it to operate incorrectly.

To reduce the number of resources, the lower 8 bits of the timer and counter are each implemented with one digital block and the upper bits are implemented in software.

The timer is set up to generate an interrupt every 256 counts or 64 integration cycles. This defines one integrate cycle. The counter is set up to integrate 64 of these time cycles. At the end of the 64th cycle, the counter is put into reset for one cycle. The sample window is 64 time cycles and the sample time is 65 integrate cycles.

Figure 3. Timer Cycles for the ADCINC12



Immediately at the start of timer cycle number 1, the counter is enabled. It stays enabled until the start of the reset cycle. During this reset period, the ADC value is calculated from the software and hardware counter values. This value is shifted right twice to compensate for the four times over clocking. The flag variable is set signifying that new data is available.

Because the ACDINC12 control is interrupt based and the sample time is relatively long, it is unreasonable to expect the processor to wait while a sample is being processed. The primary communication between the ADC routine and the main program is a data-available flag that may be polled. APIs are available to check the data flag and retrieve data.

The data handler was designed to be poll based. If an interrupt-based data handler is desired, application-specific data handler code can be added to the interrupt routine ADCINC12_TMR_ISR, located in the assembly file *adcinc12INT.asm*. The point to insert code is clearly marked.

DC and AC Electrical Characteristics

The following values are indicative of expected performance and based on initial characterization data. Unless otherwise specified in the following table, TA = 25°C, Vdd = 5.0V, Power HIGH, Op-Amp bias LOW, output referenced to 2.5V external Analog Ground on P2[4] with 1.25 external Vref on P2[6], sample rates of 100 sps, and a data clock of 1.66 MHz, unless otherwise noted.

Table 1. 5.0V ADCINC12 DC and AC Electrical Characteristics

Parameter	Typical	Limit	Units	Conditions and Notes
Input				
Input Voltage Range	---	Vss to Vdd		Ref Mux = Vdd/2 ± Vdd/2
Input Capacitance[1]	3	--	pF	
Input Impedance	1/(C*Clk)	--	Ω	
Resolution	--	12	Bits	
Sample Rate	--	7.8 to 480	sps	
SNR	71	--	dB	
DC Accuracy				
INL	1.0	--	LSB	
DNL	0.5	--	LSB	

Parameter	Typical	Limit	Units	Conditions and Notes
Offset Error	9	--	mV	
Gain Error				
Including Reference Gain Error	1.5	--	% FSR	
Excluding Reference Gain Error[2]	0.1	--	% FSR	
Operating Current				
Low Power	250	--	µA	
Med Power	640	--	µA	
High Power	2000	--	µA	
Data Clock	--	0.125 to 8.0	MHz	Input to digital blocks and analog column clock

The following values are indicative of expected performance and based on initial characterization data. Unless otherwise specified in the following table, TA = 25°C, Vdd = 3.3V, Power HIGH, OpAmp bias LOW, output referenced to 1.64V external Analog Ground on P2[4] with 1.25 external Vref on P2[6], sample rates of 100 sps, and a data clock of 1.66 MHz, unless otherwise noted.

Table 2. 3.3V ADCINC12 DC and AC Electrical Characteristics for PSoC Devices

Parameter	Typical	Limit	Units	Conditions and Notes
Input				
Input Voltage Range	---	Vss to Vdd		Ref Mux = Vdd/2 ± Vdd/2
Input Capacitance[1]	3	--	pF	
Input Impedance	1/(C*Clk)	--	Ω	
Resolution	--	12	Bits	
Sample Rate	--	7.8 to 480	sps	
SNR	72	--	dB	
DC Accuracy				
INL	1.0	--	LSB	
DNL	0.5	--	LSB	
Offset Error	7	--	mV	
Gain Error				
Including Ref Gain Error	1.5	--	% FSR	
Excluding Reference Gain Error[2]	0.4	--	% FSR	
Operating Current				

Parameter	Typical	Limit	Units	Conditions and Notes
Low Power	140	--	µA	
Med Power	490	--	µA	
High Power	1830	--	µA	
Data Clock	--	0.125 to 8.0	MHz	Input to digital blocks and analog column clock

Electrical Characteristics Notes

1. Includes I/O Pin.
2. Reference Gain Error measured by comparing the external reference to $V_{RefHigh}$ and V_{RefLow} routed through the test mux and back out to a pin.

Unless otherwise specified in the following table, all limits guaranteed for T_A = -40°C to +85°C, V_{dd} = 4.75V to 5.5V, Power HIGH, Op-Amp bias LOW, output referenced to 2.5V external Analog Ground on P2[4] with 1.25 external Vref on P2[6], sample rates of 100 sps, and a data clock of 1.66 MHz, unless otherwise noted.

Table 3. 5.0V ADCINC12 DC and AC Electrical Characteristics

Parameter	Typical[1]	Limit	Units	Conditions and Notes
INPUT				
Input Voltage Range[2]	---	Vss to Vdd		Ref Mux = Vdd/2 ± Vdd/2
Input Capacitance[3]	0.8	--	pF	
Input Impedance[4,5]	1/(C*Clk)	--	W	
Resolution	--	12	Bits	2’s Complement
Sample Rate	--	7.8 to 100	sps	samples per second
SNR[6]	68		dB	at 100 sps
DC ACCURACY				
INL	0.5	1	LSB	
DNL	0.25	0.5	LSB	
Offset Error	12	49	mV	Using external AGND
Gain Error	0.5	1.5	% FSR	Relative to reference input
OPERATING CURRENT				
Low Power	125	--	µA	
Med Power	240	--	µA	
High Power	640	1060	µA	
Data Clock[7]	--	0.125 to 1.66	MHz	Input to digital blocks and analog column clock

Unless otherwise specified in the following table, all limits guaranteed for T_A = -40°C to +85°C, V_{dd} = 4.75V to 5.5V, Power HIGH, Op-Amp bias LOW, output referenced to 2.5V external Analog Ground on P2[4] with 1.25 external Vref on P2[6], sample rates of 100 sps, and a data clock of 1.66 MHz, unless otherwise noted.

Table 4. 5.0V ADCINC12 DC and AC Electrical Characteristics

Parameter	Typical[1]	Limit	Units	Conditions and Notes
INPUT				
Input Voltage Range[2]	---	Vss to Vdd		Ref Mux = Vdd/2 ± Vdd/2
Input Capacitance[3]	0.8	--	pF	
Input Impedance[4,5]	1/(C*Clk)	--	W	
Resolution	--	12	Bits	2's Complement
Sample Rate	--	7.8 to 100	sps	samples per second
SNR[6]	68		dB	at 100 sps
DC ACCURACY				
INL	0.5	1	LSB	
DNL	0.25	0.5	LSB	
Offset Error	12	49	mV	Using external AGND
Gain Error	0.5	1.5	% FSR	Relative to reference input
OPERATING CURRENT				
Low Power	125	--	µA	
Med Power	240	--	µA	
High Power	640	1060	µA	
Data Clock[7]	--	0.125 to 1.66	MHz	Input to digital blocks and analog column clock

Unless otherwise specified in the following table, all limits guaranteed for T_A = -40°C to +85°C, V_{dd} = 3.3V to 0.3V, Power HIGH, OpAmp bias LOW, output referenced to 1.64V external Analog Ground on P2[4] with 1.25 external Vref on P2[6], sample rates of 100 sps, and a data clock of 1.66 MHz, unless otherwise noted.

Table 5. 3.3V ADCINC12 DC and AC Electrical Characteristics

Parameter	Typical[1]	Limit	Units	Conditions and Notes
INPUT				
Input Voltage Range[2]	---	Vss to Vdd		Ref Mux = Vdd/2 ± Vdd/2
Input Capacitance[3]	0.8	--	pF	
Input Impedance[4,5]	1/(C*Clk)	--	W	
Resolution	--	12	Bits	2's Complement
Sample Rate	--	7.8 to 100	sps	samples per second
SNR[6]	68		dB	at 100 sps
DC ACCURACY				
INL	0.5	1	LSB	
DNL	0.25	0.5	LSB	
Offset Error	12	49	mV	
Gain Error	0.5	1.5	% FSR	Relative to reference input
OPERATING CURRENT				
Low Power	70	--	µA	
Med Power	170	--	µA	
High Power	540	970	µA	
Data Clock[7]	--	0.125 to 1.66	MHz	Input to digital blocks and analog column clock

Electrical Characteristics Notes

1. Typical values represent parametric norm at +25°C.
2. Input voltages above the maximum generate a maximum positive reading. Input voltages below the minimum generate a maximum negative reading.
3. User module only, not including I/O pin.
4. The input capacitance or impedance is only applicable when input to analog block is directly to a pin.
5. C = input capacitance, clk = Data Clock (Analog Column Clock).
6. SNR = Ratio of power of full-scale, single tone divided by total noise integrated to $F_{sample}/2$.
7. Minimum data clock is 125 kHz when the CPU is operating at 12 MHz or lower. If the CPU is operating at 24 MHz, the minimum data clock is 250 kHz.

CY8C29xxx Typical Performance

The graphs shown in this section are restricted to a subset of the input range that best displays the dominant error. Operation with the user module set to LOWPOWER is not recommended.

Figure 4. Typical DNL, CY8C29xxx, 5.0V, 25°C, Power Level = Medium

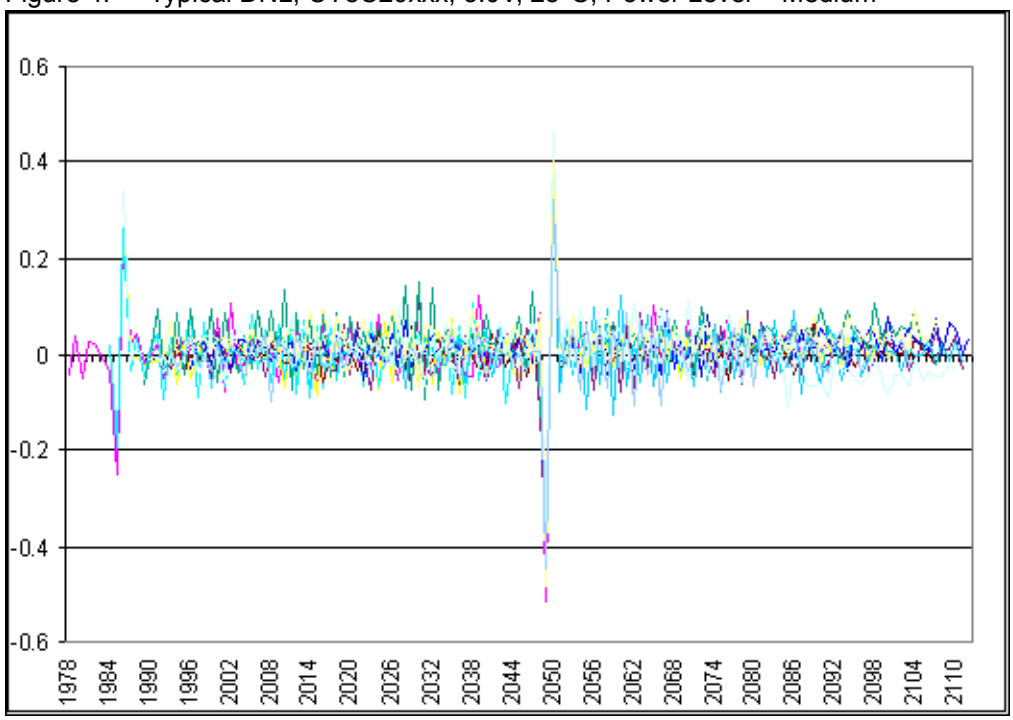
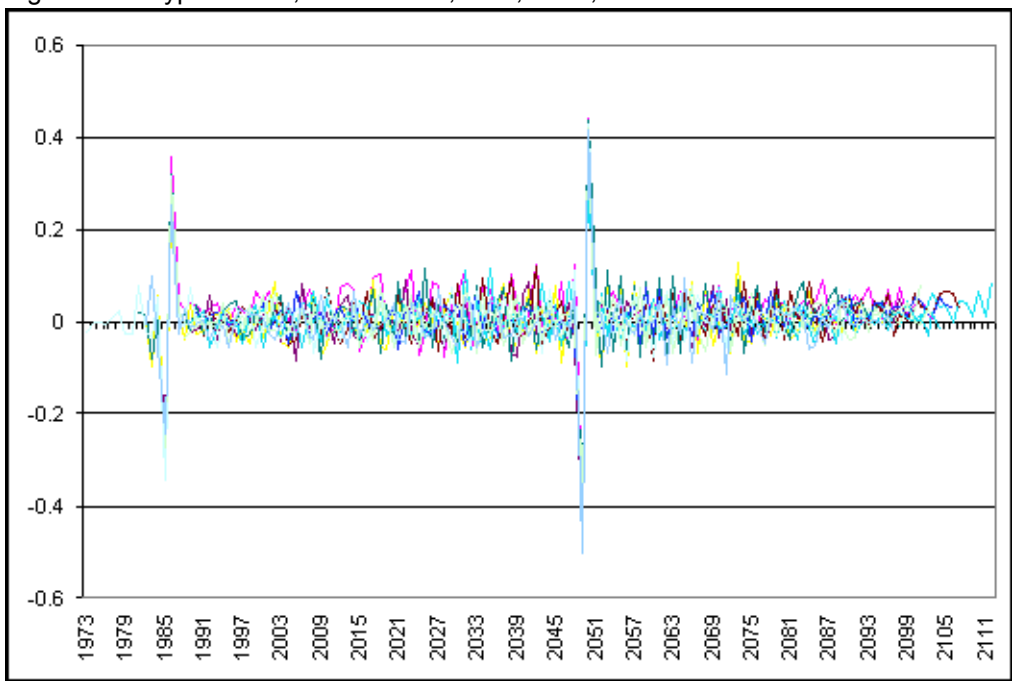


Figure 5. Typical DNL, CY8C29xxx, 3.3V, 25°C, Power Level = Medium



Placement

The ADC block can be placed in any of the switched capacitor PSoC blocks. However, it requires the comparator bus for the particular column to which it is connected. Other user modules that use the column comparator cannot be placed in the same column.

Both the timer and counter can be placed in any digital PSoC block. Both digital blocks have an interrupt service routine (ISR). It is desirable, but not required, that the Timer block have a higher interrupt priority than the Counter block. It is recommended that the Timer block be placed in a lower digital block position than the Counter block.

Parameters and Resources

These input parameters are user controlled:

- Input (Signal Multiplexer)
- Clock Phase
- TMR Clock
- CNT Clock

Input

The selection of the input is done after the analog PSoC block has been placed. Each switched capacitor block has different input selections. Each can be connected to most of its neighbors, while some can be directly connected to external input pins. Placement of the analog block must be done with some consideration of input signal routing.

Clock Phase (ClockPhase)

The selection of the Clock Phase is used to synchronize the output of one analog PSoC block to the input of another. The switched capacitor analog PSoC blocks use a two-phase clock (φ1, φ2) to acquire and transfer signal. Normally, the input to the ADCINC12 is sampled on φ1. A problem arises in that many of the user modules auto-zero their output during φ1 and only provide a valid output during φ2. If such a module's output is fed to the ADCINC12's input, the ADCINC12 acquires an auto-zeroed output instead of a valid signal. The Clock Phase selection allows the phases to be swapped so that the input signal is now acquired during φ2.

TMR Clock and CNT Clock (Data Clock)

The source for these two clocks should always be identical and is referred to as the "Data Clock." The Data Clock determines the sample rate and the signal sample window. This clock is routed to the clock input of the counter block "CNT," the clock input of the timer block "TMR," and the analog column clock for the column containing the integrator "ADC."

Note It is imperative that the same clock be used for all three blocks or this user module will not function correctly.

The Data Clock should not be set to less than 250 kHz when the CPU is running at 24 MHz. Otherwise, it may be set as low as 125 kHz. The timer is set to provide an interrupt every 256 counts of the Data Clock. The counter integrates the signal for 64 of these cycles. An additional cycle is required to reset the integrator and process the data. Equation 9 defines the sample rate:

Equation 9

$$SampleRate = \frac{DataClock}{65 \times 256}$$

The sample window determines the normal mode frequencies the ADC rejects. Equation 10 defines the sample window:

Equation 10

$$SampleWindow = \frac{64 \times 256}{DataClock}$$

To reject a higher frequency and its harmonics, select the sample window such that it is an even multiple of the frequency-to-reject.

Example 1

A sample window of 100 milliseconds allows 6 cycles of 60 Hz or 5 cycles of 50 Hz to be integrated, effectively removing it or any of its harmonics. For a 100-millisecond sample window, the Data Clock must be:

Equation 11

$$DataClock = \frac{64 \times 256}{100ms} = 163.84kHz$$

The sample rate is:

Equation 12

$$SampleRate = \frac{163.84kHz}{65 \times 256} = 9.85sps$$

The timer function uses interrupts to mark time. It is possible that the servicing of some other interrupt could delay servicing of the timer. This delay erroneously increases sample time, allowing more counts to be accumulated. Care must be taken in the design of the total system interrupt scheme. Each CPU cycle delay in servicing the timer allows that much more time to erroneously collect samples. Each Data Clock cycle is ¼th of a count. The maximum error from one CPU cycle of interrupt latency is:

Equation 13

$$LatencyError = \frac{delay_{CPUcycles}}{CPUClock} \times \frac{DataClock}{4}$$
$$= delay_{cycles} \times 0.25 \times \frac{DataClock}{CPUClock}$$

Example 2

If the worst case interrupt latency for a system is 120 CPU cycles, the CPU clock is 24 MHz, and the data clock is 512 kHz, then the error would be 0.62 counts.

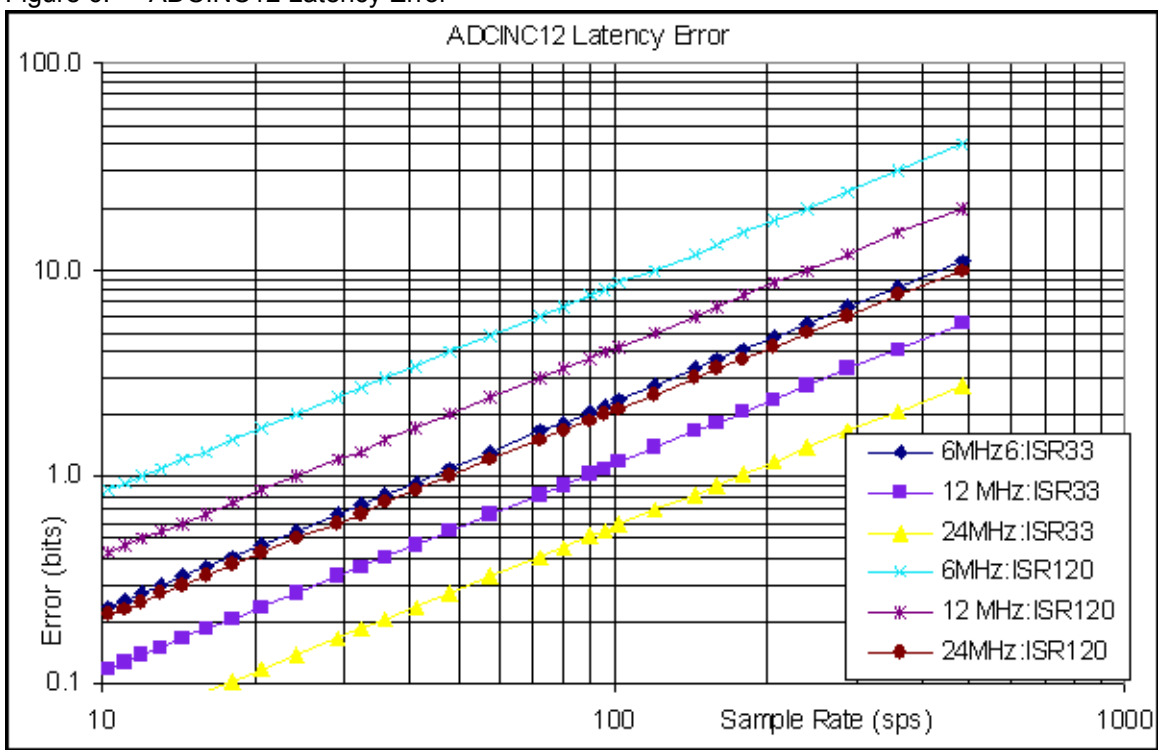
It is possible for the counter routine to interfere and delay the timer. The counter routine requires 33 CPU cycles to be serviced. Each Data Clock cycle is ¼th of a count, so the maximum error from counter interrupt latency can be as much as:

Equation 14

$$CounterError = \frac{33}{CPUClock} \times \frac{DataClock}{4} = 8.25 \times \frac{DataClock}{CPUClock}$$

The effect of interrupt latency on ADC error is shown in Figure 6 for two typical ISR routines (33 and 120 CPU cycles), as a function CPU clock and sample rate.

Figure 6. ADCINC12 Latency Error



The routine to calculate the ADC value can take as many as 234 CPU cycles and must be calculated during the reset period. This reset period is 256 data clock cycles. The CPU clock must always be faster than the Data Clock.

To summarize, the CPU clock must be faster than the Data Clock. It is recommended that it be at least eight times faster. A system's interrupt scheme must be designed carefully.

Ref Mux Global Resource

The most important global resource when dealing with analog to digital converters (ADC) is the Ref Mux. The setting of the Ref Mux determines the usable input voltage range of the ADC. The following table shows the valid ranges for a V_{dd} of 5V and 3.3V:

Table 6. CY8C26/25xxx Input Voltage Ranges with Respect to the RefMux Setting

RefMux Setting	Vdd = 5 Volts	Vdd = 3.3 Volts
(Vdd/2) ± BandGap	$1.2 < V_{in} < 3.8$	$0.35 < V_{in} < 2.95$
(Vdd/2) ± (Vdd/2)	$0 < V_{in} < 5$	$0 < V_{in} < 3.3$
(2*BandGap) ± BandGap	$1.3 < V_{in} < 3.9$	NA
(2*BandGap) ± P2[6]	$(2.6 - V_{P2[6]}) < V_{in} < (2.6 + V_{P2[6]})$	NA
P2[4] ± BandGap	$(V_{P2[4]} - 1.3) < V_{in} < (V_{P2[4]} + 1.3)$	$(V_{P2[4]} - 1.3) < V_{in} < (V_{P2[4]} + 1.3)$
P2[4] ± P2[6]	$(V_{P2[4]}-V_{P2[6]}) < V_{in} < (V_{P2[4]}+V_{P2[6]})$	$(V_{P2[4]}-V_{P2[6]}) < V_{in} < (V_{P2[4]}+V_{P2[6]})$

Table 7. CY8C29/27/24/22xxx Input Voltage Ranges for Each Ref Mux Setting

RefMux Setting	Vdd = 5 Volts	Vdd = 3.3 Volts
(Vdd/2) ± BandGap	$1.2 < V_{in} < 3.8$	$0.35 < V_{in} < 2.95$
(Vdd/2) ± (Vdd/2)	$0 < V_{in} < 5$	$0 < V_{in} < 3.3$
BandGap ± BandGap	$0 < V_{in} < 2.6$	$0 < V_{in} < 2.6$
(1.6*BandGap) ± (1.6*BandGap)	$0 < V_{in} < 4.16$	NA
(2*BandGap) ± BandGap	$1.3 < V_{in} < 3.9$	NA
(2*BandGap) ± P2[6]	$(2.6 - V_{P2[6]}) < V_{in} < (2.6 + V_{P2[6]})$	NA
P2[4] ± BandGap	$(V_{P2[4]} - 1.3) < V_{in} < (V_{P2[4]} + 1.3)$	$(V_{P2[4]} - 1.3) < V_{in} < (V_{P2[4]} + 1.3)$
P2[4] ± P2[6]	$(V_{P2[4]}-V_{P2[6]}) < V_{in} < (V_{P2[4]}+V_{P2[6]})$	$(V_{P2[4]}-V_{P2[6]}) < V_{in} < (V_{P2[4]}+V_{P2[6]})$

Interrupt Generation Control

There is an additional parameter that becomes available when the **Enable interrupt generation control** check box in PSoC Designer is checked. This is available under **Project > Settings > Chip Editor**. Interrupt Generation Control is important when multiple overlays are used with interrupts shared by multiple user modules across overlays:

IntDispatchMode

The IntDispatchMode parameter is used to specify how an interrupt request is handled for interrupts shared by multiple user modules existing in the same block, but in different overlays. Selecting “ActiveStatus" causes firmware to test which overlay is active before servicing the shared interrupt request. This test occurs every time the shared interrupt is requested. This adds latency and also produces a nondeterministic procedure of servicing shared interrupt requests, but does not require any RAM. Selecting “OffsetPreCalc" causes firmware to calculate the source of a shared interrupt request only when an overlay is initially loaded. This calculation decreases interrupt latency and produces a deterministic procedure for servicing shared interrupt requests, but at the expense of a byte of RAM.

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.

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.

Entry Points are to initialize the ADC, start it sampling, and stop the ADC.

ADCINC12_Start

Description:

Performs all required initialization for this user module and sets the power level for the switched capacitor PSoC block.

C Prototype:

```
void  ADCINC12_Start (BYTE bPowerSetting)
```

Assembly:

```
mov   A, ADCINC12_HIGHPOWER    ;FULLPOWER = 3
lcall  ACDINC12_Start
```

Parameters:

PowerSetting: One byte that specifies the power level. Following reset and configuration, the analog PSoC block assigned to ADCINC12 is powered down. Symbolic names provided in C and assembly, and their associated values, are listed in the following table:

Symbolic Name	Value
ADCINC12_OFF	0
ADCINC12_LOWPOWER	1
ADCINC12_MEDPOWER	2
ADCINC12_HIGHPOWER	3

Power level has an effect on analog performance. The correct power setting is sensitive to the sample rate of the Data Clock and must be determined for each application. It is recommended that you start your development with full power selected. Testing can be done later to determine how low you can set the power setting.

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

ADCINC12_SetPower

Description:

Sets the power level for the switched capacitor PSoC block.

C Prototype:

```
void  ADCINC12_SetPower (BYTE bPowerSetting)
```

Assembly:

```
mov   A, bPowerSetting
lcall  ACDINC12_SetPower
```

Parameters:

PowerSetting: Same as the PowerSetting parameter used for the Start entry point.

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

ADCINC12_Stop

Description:

Sets the power level on the switched capacitor PSoC block to 0FF. This is done when the ADCINC12 in not being used and if you want to save power.

C Prototype:

```
void  ADCINC12_Stop (void)
```

Assembly:

```
lcall  ACDINC12_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

ADCINC12_GetSamples

Description:

Runs the ADC for specified number of samples.

C Prototype:

```
void  ADCINC12_GetSamples (BYTE bNumSamples)
```

Assembly:

```
mov   A, bNumSamples
lcall  ACDINC12_GetSamples
```

Parameters:

bNumSamples: 8-bit value that sets the number of samples to be converted. A value of ‘0’ causes the ADC to run continuously.

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP page pointer register is modified.

ADCINC12_StopAD

Description:

Immediately stops the ADC.

C Prototype:

```
void  ADCINC12_StopAD (void)
```

Assembly:

```
lcall  ADCINC12_StopAD
```

Parameters:

None

Return Value:

None

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP page pointer register is modified.

ADCINC12_fIsDataAvailable

Description:

Checks the status of the ADC.

C Prototype:

```
BYTE  ADCINC12_fIsDataAvailable(void)
BYTE  ADCINC12_fIsData(void)    (Deprecated)
```

Assembly:

```
lcall  ADCINC12_fIsDataAvailable
lcall  ADCINC12_fIsData
```

Parameters:

None

Return Value:

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

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP page pointer register is modified.

ADCINC12_iGetData

Description:

Returns converted data. ADCINC12_fIsDataAvailable() should be called to verify that the data sample is ready. There is a possibility that the returned data is corrupted if the call to this function is done exactly at the end of an integration period. It is therefore highly recommended that the data retrieval be done at a higher frequency than the sampling rate, or if that cannot be guaranteed that interrupts be turned off before calling this function.

C Prototype:

```
INT   ADCINC12_iGetData(void)
```

Assembly:

```
lcall  ADCINC12_iGetData
```

Parameters:

None

Return Value:

Returns the converted data sample in 16-bit, 2’s complement format.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP page pointer register is modified.

ADCINC12_ClearFlag

Description:

Resets the data-available flag.

C Prototype:

```
void  ADCINC12_ClearFlag(void)
```

Assembly:

```
lcall  ADCINC12_ClearFlag
```

Parameters:

None

Return Value:

None

Side Effect:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP page pointer register is modified.

Sample Firmware Source Code

The sample code polls the Flag register and sends the data to a routine that shifts the data out one of the I/O pins.

```
;;; Sample Code for the ADCINC12
;;; Continuously Sample and Output Data to a pin.
;;;
;;; The user must provide the function to shift the data out.
;;;

include "m8c.inc"       ; part specific constants and macros
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules

export _main

_main:
   M8C_EnableGInt                  ; enable global interrupts
   mov  a,ADCINC12_HIGHPOWER       ; set Power
   call ADCINC12_Start

   mov a,00h                      ; set ADC to continuous sampling
   call ADCINC12_GetSamples
loop1:
wait:
   call ADCINC12_fIsDataAvailable  ; poll flag
   jz wait

   call ADCINC12_ClearFlag         ; reset flag
   call ADCINC12_iGetData          ; retrieve the data
   ;; call shift_it_out            ; (user provided) send data to output pin
```

```
   jmp loop1
```

The same project written in C is:

```
//-------------------------------------------------------------------------
//  Sample C Code for the ADCINC12
//  Continuously Sample input voltage
//
//-------------------------------------------------------------------------
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules
INT  iData;

void main(void)
{
   M8C_EnableGInt;                        // Enable Global Interrupts
   ADCINC12_Start(ADCINC12_HIGHPOWER);    // Apply power to the SC Block
   ADCINC12_GetSamples(0);                // Have ADC run continuously
   for(;;){
      while(ADCINC12_fIsDataAvailable() == 0);   // Loop until value ready
      ADCINC12_ClearFlag();                      // Clear ADC flag
      iData=ADCINC12_iGetData();                 // Get ADC result
      // Add user code here to use or display result
   }
 }
```

Configuration Registers

The ADC is a switched capacitor PSoC block. It is configured to make an analog modulator. To build the modulator, the block is configured to be an integrator with reference feedback that converts the input value into a digital pulse stream. The input multiplexer determines what signal is digitized.

Table 8. Block ADC: Register CR0

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

Table 9. Block ADC: Register CR1

Bit	7	6	5	4	3	2	1	0
Value	ACMux, AMux			0	0	0	0	0

ACMux is used when the block is placed in a type “A" block. AMux is used when the block is placed in a type “B" block. The field value of both depends on how you connect the input.

Table 10. Block ADC: Register CR2

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

AZ is used by the TMR interrupt handler and various APIs. A ‘1’ value causes ADC to be a disabled integrator. A ‘0’ value causes ADC to be an enabled integrator.

Table 11. Block ADC: Register CR3

Bit	7	6	5	4	3	2	1	0
Value	1	1	1	FSW0	0	0	0	0

FSW0 is used by the TMR interrupt handler and various APIs. A ‘1’ value causes ADC to be a disabled integrator. A ‘0’ value causes ADC to be an enabled integrator.

The TMR is a digital PSoC block configured to have a timer with a period of 256 counts. 64 of these timer periods are used to set the integration time and sample rate.

Table 12. Block TMR: Register Function

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

Table 13. Block TMR: Register Input

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

Clock selects the input clock from one of 16 sources. This parameter is set in the Device Editor.

Table 14. Block TMR: Register Output

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

Table 15. Block TMR: Register DR0

Bit	7	6	5	4	3	2	1	0
Value	Timer Value (Never Used by API)							

Table 16. Block TMR: Register DR1

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

Table 17. Block TMR: Register DR2

Bit	7	6	5	4	3	2	1	0
Value	Not Used							

Table 18. Block TMR: Register CR0

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

Enable enables TMR when set. It is modified and controlled by the ADCINC12 API.

The CNT is a digital PSoC block configured as a counter. When the value in DR0 counts down to terminal count, an interrupt is called to decrement a higher value software counter and the CNT reloads from DR1. The data is output through DR2.

Table 19. Block CNT: Register Function

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

Table 20. Block CNT: Register Input

Bit	7	6	5	4	3	2	1	0
Value	Data				Clock			

Data selects the column comparator where the ADC block has been placed. Clock selects the input clock from one of 16 sources and is set in the Device Editor.

Table 21. Block CNT: Register Output

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

Table 22. Block CNT: Register DR0

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

Table 23. Block CNT: Register DR1

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

Table 24. Block CNT: Register DR2

Bit	7	6	5	4	3	2	1	0
Value	Data Out							

Data Out is used by the API to get the counter value.

Table 25. Block CNT: Register CR0

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

Enable enables CNT when set. It is modified and controlled by the ADCINC12 API.

Table 26. Register INT_MSK1

Bit	7	6	5	4	3	2	1	0
Value								

The mask bits corresponding to the TMR block and CNT block are set here to enable their respective interrupts. The actual mask values are determined by the placement position of each block.

Version History

Version	Originator	Description
5.3	DHA	Added DRC to check if: 1. The source clock is different between digital and analog resources. 2. The ADC Clock is higher than CPU Clock.
5.3.b	HPHA	Added design rules check for the situation when the ADC clock is faster than 8 MHz.

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 © 2002-2013 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.