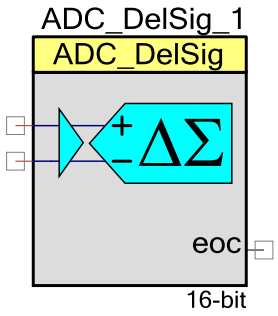
Delta Sigma Analog to Digital Converter (ADC_DelSig)

2.0

Features



- Selectable resolutions, 8 to 20 bits (device dependent)
- Eleven input ranges for each resolution
- Sample rate 10 sps to 384 ksps
- Operational modes:
  - Single sample
  - Multi-sample
  - Continuous mode
  - Multi-sample (Turbo)
- High input impedance input buffer
  - Selectable input buffer gain (1, 2, 4, 8) or input buffer bypass
- Multiple internal or external reference options
- Automatic power configuration
- Up to four run-time ADC configurations
- Supports PSoC 3 ES3 or later and PSoC 5 ES1 or later

General Description

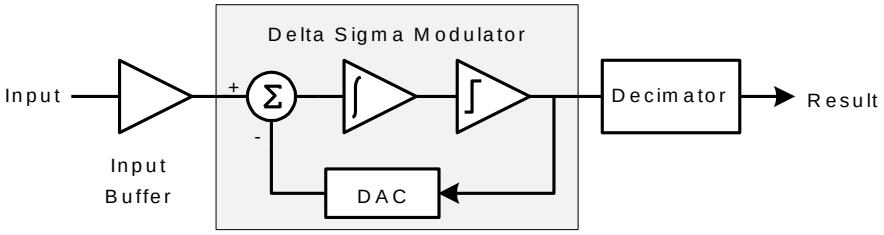
The Delta Sigma Analog to Digital Converter (ADC_DelSig) provides a low power, low noise front end for precision measurement applications. The ADC_DelSig is usable in a wide range of applications depending on resolution, sample rate and operating mode. It is capable 16-bit audio, high speed low resolution for communications processing, and high precision 20-bit low speed conversions for sensors such as strain gauges, thermocouples and other high precision sensors. When processing audio information, the ADC_DelSig is used in a continuous operation mode. When used for scanning multiple sensors, the ADC_DelSig is used in one of the multi-sample modes. When used for single point high resolution measurements, the ADC_DelSig is used in single sample mode.

Delta sigma converters use oversampling to spread the quantization noise across a wider frequency spectrum. This noise is shaped to move most of it outside the input signal's bandwidth. An internal low pass filter is used to filter out the noise outside the desired input signal bandwidth. This makes delta sigma converters good for both high speed medium

resolution (8 to 16 bits) and low speed high resolution (16 to 20 bits) applications. The sample rate can be adjusted between 10 and 375000 samples per second, depending on mode and resolution. Choices of conversion modes simplify interfacing to single streaming signals such as audio, or multiplexing between multiple signal sources.

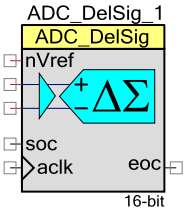
The ADC_DelSig is composed of three blocks: input amplifier, third order Delta-Sigma modulator, and a decimator. The input amplifier provides a high impedance input and a user-selectable input gain. The decimator block contains a 4 stage CIC decimation filter and a post-processing unit. The CIC filter operates on the data sample directly from the modulator. The post-processing unit optionally performs gain, offset, and simple filter functions on the output of the CIC decimator filter.

Figure 1: ADC_DelSig Block Diagram



Input/Output Connections

Inputs and output connections to the ADC_DelSig component are displayed as pins on the component symbol in the schematic view. An asterisk (*) in the list of I/Os indicates that the I/O may be hidden on the symbol under the conditions listed in the description of that I/O.



nVref – Input *

The nVref is an optional pin. It is shown if you have selected the **Enable_Vref_Vssa** check box. This will allow you to connect the ADC's reference Vssa to the analog global (AGL[6]). If the **Enable_Vref_Vssa** check box is not selected, this pin will not be shown on the symbol. Refer to **Enable_Vref_Vssa** parameter description for more information.

+Input – Analog

Positive analog signal input to the ADC_DelSig. The positive input signal is always present in both the single ended and differential input modes. The ADC converter output returns a value that represents the difference in voltage between positive input and the negative input signal.

–Input – Analog *

Negative analog signal input to the ADC_DelSig. The negative input pin is only displayed on the component when the ADC Input Mode is set to Differential. When ADC Input Mode is set to Single, the negative input will be connected to either Vssa or Vref depending on the input range selected.

soc – Input *

Start of Conversion (soc) starts hardware triggered ADC conversions when a rising edge is detected. A rising edge on this pin, has the same effect as calling the ADC_StartConversion() function. This input is shown when the user selects the "External soc" parameter. If "External soc" is not selected, the I/O pin on the component will be hidden. In Single Sample mode, a single conversion is executed, then the ADC halts. In Continuous and other modes, ADC conversions continue until either the ADC_StopConvert() or ADC_Stop() functions are executed.

aclk – Input *

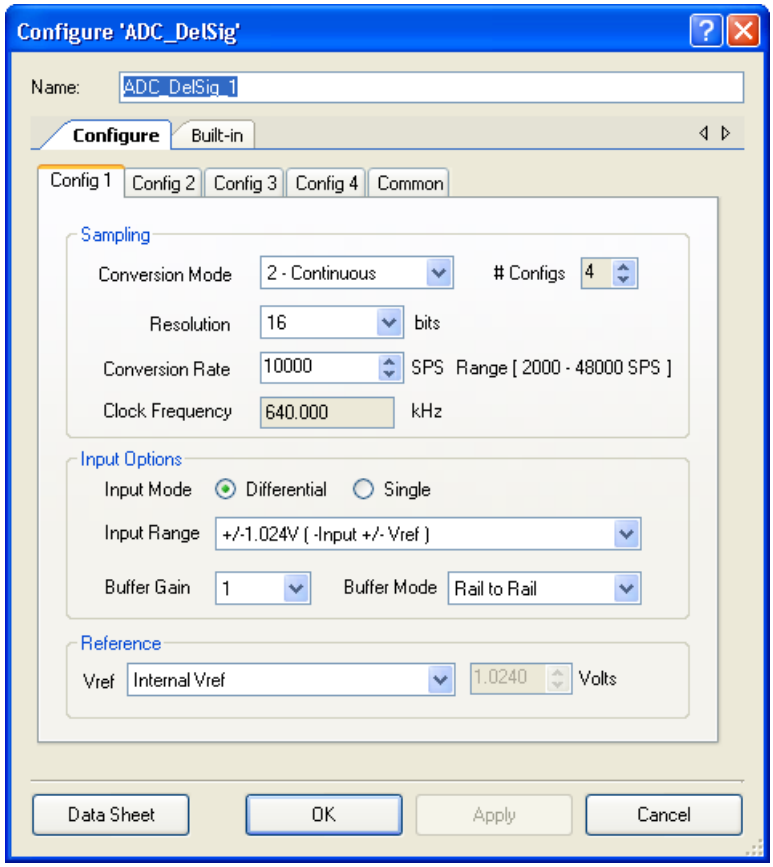
External clock source. This pin is present if the **Clock Source** parameter is set to "External." If the **Clock Source** parameter is set to "Internal," the clock is configured automatically within the component and the aclk pin is not shown. The aclk input is a clock that is generated outside the component. This clock signal may be derived internal to the chip or from a source external to the PSoC. This clock should be set to the value displayed in the Clock Frequency parameter to achieve the selected sample rate. The duty cycle should be 50%. This clock determines the conversion rate as a function of conversion method and resolution.

eoc – Output

A rising edge on the End of Conversion (eoc) signals that a conversion is complete. The pin goes high for one aclk period. The eoc is typically connected to an interrupt or DMA request. The DMA request is typically used to transfer the conversion output to system RAM, DFB, or other component.

Parameters and Setup

The Delta Sigma ADC is a highly configurable analog to digital converter. Drag an ADC_DelSig component onto your design and double-click it to open the Configure dialog.



Sampling

Conversion Mode

The ADC_DelSig operates in one of four modes:

Mode	Description
0 - Single Sample	The ADC produces one sample per startup conversion. The interrupt should be enabled for ADC conversion with Single Sample conversion mode when the resolution is above 16 bits. To do so, enable the Global Interrupt (by calling CYGlobalIntEnable) in the application (*main.c*).

Mode	Description
1 - Multi-Sample	Multi-sample mode captures single samples back to back, resetting itself and the modulator between each sample automatically. This mode is useful when the input is switched between multiple signals. The filters are flushed between each sample so previous samples do not affect the current conversion. **Note** Care should be taken when switching signals between ADC conversions. Either switch the input quickly between conversions with hardware control or Stop the ADC conversion (ADC_StopCOnvert()) while switching the input then restart the ADC conversion (StartConvert()) after the new signal has been connected to the ADC. Failure to do this may result in contamination between signals in the ADC results.
2 - Continuous	Continuous sample mode operates as a normal Delta-Sigma converter. Use this mode when measuring a single input signal. There is a latency of three conversion times before the first conversion result is available. This is the time required to prime the internal filter. After the first result, a conversion will be available at the selected sample rate. This mode should not be used when multiple signals are multiplexed and measured with a single ADC.
3 - Multi-Sample (Turbo)	The Multi-Sample (Turbo) mode operates identically to the Multi-Sample mode for resolution of 8 to 16 bits. For resolutions of 17 to 20 bits, the performance of this mode is about 4 times faster than the Multi-Sample mode. **Note** Care should be taken when switching signals between ADC conversions. Either switch the input quickly between conversions with hardware control or stop the ADC conversion (ADC_StopConvert()) while switching the input. Then restart the ADC conversion (StartConvert()) after the new signal has been connected to the ADC. Failure to do this may result in contamination between signals in the ADC results.

When changing the **Conversion Mode** parameter, the clock frequency will change to maintain the selected sample rate. If the ADC clock frequency exceeds the minimum or maximum an error condition will be displayed.

Configs

You may define up to four different configurations using the **# Configs** parameter. For example, the system may require switching between continuous mode 16 bit, 48 ksps for audio, single sample mode 20 bit 60 sps for low level analog sensors, and multi-sample mode for 12 bit general purpose multi-channel data logging. All configurations must use the same Input Mode, all single-ended or all differential.

By default, the ADC will be set to the first configuration (Config1) unless the ADC_SelectConfiguration() function sets the configuration to a different value. When selecting between two and four configurations, additional tabs will appear in the Configure dialog. These multiple configurations allow you to change modes during run time. Each configuration is contained in its own tab.

There are some considerations when using multiple ADC configurations:

- All configurations must use the same **Input Mode**, all single-ended or all differential.
- The **Common** tab contains the **Hardware SOC**, **Clock Source**, **Low Power C-Pump**, **ADC Mode**, and **Enable_Vref_Vssa** parameters, which are common to all modes. These parameters are described under Common Settings.

- The **Ref Mode** parameter will also have some restrictions. If the options on Config 1 set an external reference or bypass mode, the other configurations may select the same mode or use the internal reference.
- Each configuration will have a separate Interrupt Service Routine function. When the ADC_SetConfiguration() function is called, the interrupt vector will be changed to the corresponding interrupt vector routine.

When a clock external to the ADC (either external to the chip or supplied from user selected internal clock) is utilized, the required clock rate is displayed in this field. It is your responsibility to provide the appropriate clock for each configuration.

Resolution

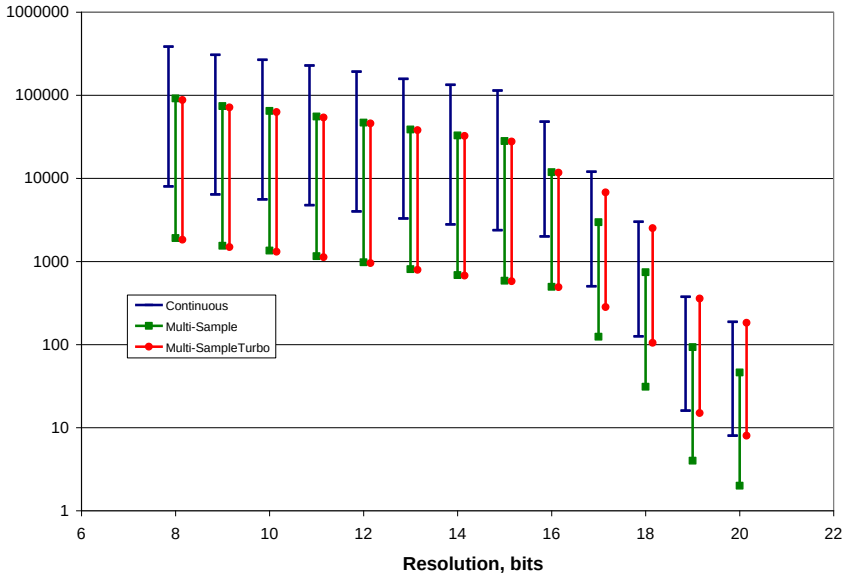
The resolution of the ADC_DelSig is entered as an integer value, limited to 8 to 20 bits. Higher resolution results in lower sample rate. Default resolution is 16 bits. When changing the **Resolution** parameter, the clock frequency will change to maintain the selected sample rate. If the ADC clock frequency exceeds the minimum or maximum an error condition will be displayed.

Delta-Sigma ADCs have inherent instability resulting in non-linearity at the positive and negative limits of the operating range. To correct for this phenomenon, the input has been attenuated by 10% at the front end of the modulator. The post processor then compensates for this attenuation with a gain of about 1.11. The end result expands the input range by 10%. For example if the input range ± 1.024 volts is selected, the actual range of the ADC is approximately ± 1.126 volts. The usable input range remains ± 1.024 volts, but the ADC will not saturate until the input exceeds ± 1.126 volts.

The digital output from the ADC will also over range by 10%. If the ADC is configured for 10-bit operation, normally a 10-bit differential ADC's output will range from -512 to 511, for an input of -1.024 to +1.022 respectively. Because of this additional 10% of range, the digital output will not saturate until about ± 563 counts, instead of -512 to 511.

This is not normally a concern unless a resolution of 8 or 16 bits is selected. When the resolution is set to either 8 or 16 bits, care must be taken to ensure that the numerical value does not wrap around from its most positive or negative value to a negative or positive value respectively. To ensure this does not occur, it is good practice to use the API function that returns a word larger than the set resolution. For example, if the resolution is set to 16 bits and there is a possibility that the most positive value may be larger than 32767 or less than -32768, the ADC_GetResult32() function should be used instead of ADC_GetResult16(). The proper 16 bit value will be returned without over-ranging. When the resolution is set to 8 bits and the ADC output values may be less than -128 or greater than 127, the ADC_GetResult16() function should be used. The proper 8 bit value will be returned without over-ranging.

Figure 2: Sample Rate Limits for ADC_DelSig



Conversion Rate

ADC conversion rate is entered as an integer decimal value in samples per second (SPS). The maximum sample rate is a function of resolution, sample mode and maximum operating clock frequency; the higher the resolution, the lower the sample rate. The minimum clock for all resolutions is 128 kHz. The maximum clock for resolutions between 8 and 15 bits is 6.144 MHz. The maximum clock for resolutions between 16 and 20 bits is 3.027 MHz.

See Figure 2 for valid conversion rates for each resolution and conversion mode combination; the same information is presented in tabular form in the following table.

The following data applies to ADC range = +/-1.024 V with Buffer Gain = 1.0.

Table 1: Sample Rate Limits for ADC_DelSig (Buffer Gain = 1)

	Single-Sample		Multi-Sample		Continuous		Multi-Sample Turbo	
Resolution	Min	Max	Min	Max	Min	Max	Min	Max
8	1911	91701	1911	91701	8000	384000	1829	87771
9	1543	75042	1543	75042	6400	307200	1489	71441
10	1348	64673	1348	64673	5566	267130	1307	62693
11	1154	55351	1154	55351	4741	227555	1123	53894
12	978	46900	978	46900	4000	192000	956	45894
13	806	38641	806	38641	3283	157538	791	37925
14	685	32855	685	32855	2783	133565	674	32337

Resolution	Single-Sample		Multi-Sample		Continuous		Multi-Sample Turbo	
	Min	Max	Min	Max	Min	Max	Min	Max
15	585	28054	585	28054	2371	113777	577	27675
16	495	11861	495	11861	2000	48000	489	11725
17	124	2965	124	2965	500	12000	282	6766
18	31	741	31	741	125	3000	105	2513
19	4	93	4	93	16	375	15	357
20	2	46	2	46	8	187	8	183

The ADC buffer has a finite gain bandwidth which affects settling time. Increasing the buffer gain reduces the available maximum sample rate. The maximum sample rate is the sample rate in Table 1 divided by the buffer gain. Other ranges and buffer gains will affect the maximum sample rate.

When changing the **Conversion Rate** parameter, the clock frequency will change to maintain the selected sample rate. If the ADC clock frequency exceeds the minimum or maximum an error condition will be displayed.

Range [__ SPS]

This field is a non-editable (always grayed out) area used to display the minimum and maximum available conversion rate for the current settings. If the conversion rate is outside the minimum and maximum limits, the background turns red.

Clock Frequency

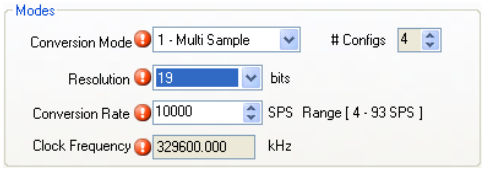
This text box is a non-editable (always grayed out) area used to display the required clock rate for the selected operating conditions: conversion mode, resolution, conversion rate, input range and buffer gain. It is updated when any of these conditions are changed. The clock frequency is displayed with a resolution of 1 Hz. If the required clock frequency for the selected operating conditions is outside of the minimum and maximum limits, the background turns red.

The clock frequency is calculated based on the Resolution, Conversion Mode, and Conversion Rate. The rate will be displayed in the Design-Wide Resources Clock Editor, which will always show the clock frequency for Config 1. The ADC API will set the current clock frequency based on the configuration selected during run time when the **Clock Source** parameter is set to "Internal."

When a clock external to the ADC (either external to the chip or supplied from user selected internal clock) is utilized, the required clock rate is displayed in this field. It is your responsibility to provide the appropriate clock for each configuration.

Invalid Settings

The parameters **Conversion Mode**, **Resolution**, and **Conversion Rate** all affect the ADC clock frequency. Changing any of these parameters may cause the ADC clock frequency to exceed the maximum or minimum rate. The maximum ADC frequency is a function of **Resolution**, **Buffer Gain** and **Input Range**. If an invalid setting for these parameters occurs, a red circle with an exclamation point will appear, as follows:



If you hover the cursor over one of the error symbols, it will display an error message. Change the parameters as needed to comply with the ADC specifications.

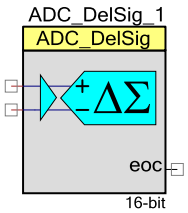
Input Options

Input Mode

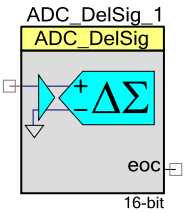
The ADC is inherently differential; however, you may use this parameter to simplify single-ended use.

This parameter configures the ADC for a Differential or Single Ended input. The default selection is Differential. In this mode, both negative and positive inputs are shown on the symbol. When Single Ended mode is selected, the negative input to the ADC is connected to Vssa.

Differential



Single Ended



The **Input Mode** can only be set in Config 1. If more than one configuration is used, all configurations must use the same Input Mode set in Config 1. If both Single Ended and Differential modes are required, the Differential mode must be selected and you can use an analog mux to connect Vssa to the negative input of the ADC to use as a single-ended ADC..

This parameter controls the options available in the **Input Range** parameter.

Input Range

This parameter configures the ADC for a given input range. This configures the input to the ADC and is independent of the **Input Buffer Gain** setting. The analog signals connected to the IC must be between Vssa and Vdda no matter what input range settings are used.

The absolute maximum of the ADC Input Range will always be dictated by the absolute maximum and minimum of the **Buffer Mode**. See Buffer Mode section for more details.

The options available for this parameter vary depending on the **Input Mode** selection; the following tables describe the options.

The following options are available when Input Mode is set to Differential: For systems where both single ended and differential signals are scanned, connect the negative input to Vssa when scanning a single ended input. Depending on the application, you can select either Rail to Rail, Offset, or Bypass using the **Buffer Mode** parameter. See Buffer Mode parameter description for more details.

An external reference may be used to provide a different operating range. The usable input range can be calculated with the equation,

Table 2 Differential Input Range Options

Input Range Internal Ref (External Ref)	Description
+/- 1.024 V (-Input +/- Vref)	When using the internal reference (1.024 V), the input range will be –Input +/- 1.024 V. If the negative input is connected to 2.048 volts the usable input range is 2.048 +/- 1.024 V or 1.024 to 3.072 V.
+/- 2.048 V (-Input +/- 2*Vref)	When using the internal reference (1.024 V), the input range will be –Input +/- 2.048 V. If the negative input is connected to 2.028 volts the usable input range is 2.048 +/- 2.048 V or 0.0 to 4.096 V.
+/- 6.144 V (-Input +/- 6*Vref)	When using the internal reference (1.024 V), the input range will be –Input +/- 6.144 V., but not exceeding maximum electrical input range. This mode can be used to measure the supply voltages when connecting the negative input to Vssa. If you intend to measure the supply you must bypass the buffer..
+/- 0.512 V (-Input +/- Vref/2)	When using the internal reference (1.024 V), the input range will be –Input +/- 0.512 V. If –Input is connected to 1.0 volt the usable input range is 1.0 +/- 0.512 V or 0.488 to 1.512 V.
+/- 0.256 V (-Input +/- Vref/4)	When using the internal reference (1.024 V), the input range will be –Input +/- 0.256 V. If –Input is connected to 1.0 volt the usable input range is 1.0 +/- 0.256 V or 0.744 to 1.256 V.
+/- 0.125 V (-Input +/- Vref/8)	When using the internal reference (1.024 V), the input range will be –Input +/- 0.125 V. If –Input is connected to 1.0 volt the usable input range is 1.0 +/- 0.125 V or 0.875 to 1.125 V.
+/- 0.0625 V (-Input +/- Vref/16)	When using the internal reference (1.024 V), the input range will be –Input +/- 0.0625V. If –Input is connected to 1.0 volt the usable input range is 1.0 +/- 0.0625 V or 0.9375 to 1.0625 V.

The following options are available when Input Mode is set to Single-Ended: To simulate single ended operation the neg input is connected to an internal reference value (Vssa or Vref)

Depending on the application, you can select either Rail to Rail, Offset, or Bypass using the **Buffer Mode** parameter. See Buffer Mode parameter description for more details.

An external reference may be used to provide a different operating range. The usable input range can be calculated with the applicable equation,

Table 3 Single-Ended Input Range Options

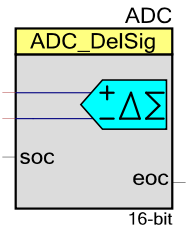
Input Range Internal Ref (External Ref)	Description
Vssa to 1.024 V (0 to Vref)	When using the internal reference (1.024 V), the usable input voltage to the ADC is 0.0 to 1.024 Volts.
Vssa to 2.048 V (0.0 to 2*Vref)	When using the internal reference (1.024 V), the usable input voltage to the ADC is 0.0 to 2.048 Volts. This range requires that the input buffer gain be equal to 1. If a gain other than 1 is selected, the ADC will not operate properly
Vssa to Vdda	This mode is ratiometric of the supply voltage. The input range is Vssa to Vdda. An external reference should not be used for this setting. This range requires that the input buffer gain be equal to 1. If a gain other than 1 is selected, the ADC will not operate properly.
Vssa to 6.144 V (Vssa to 6*Vref)	When using the internal reference (1.024 V), the input range will be 0.0 to 6.144 V., but not exceeding maximum electrical input range. This mode can be used to measure the supply voltage. If you intend to measure the supply you must bypass the buffer.

Buffer Gain

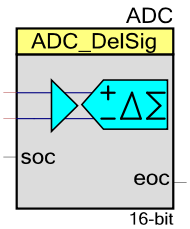
Selects the ADC input buffer gain. The ADC buffer has a finite gain bandwidth which affects settling time. Increasing the buffer gain reduces the available maximum sample rate. The maximum sample rate is the sample rate in Table 1 divided by the buffer gain.

To achieve the highest signal to noise ratio, it is important to use the full range of the ADC. The input buffer can be used to amplify the input signal to make use of the full range of the ADC. Care should be taken to make sure the Buffer_Gain and ADC_Input_Range setting are compatible.

Unbuffered



Buffered



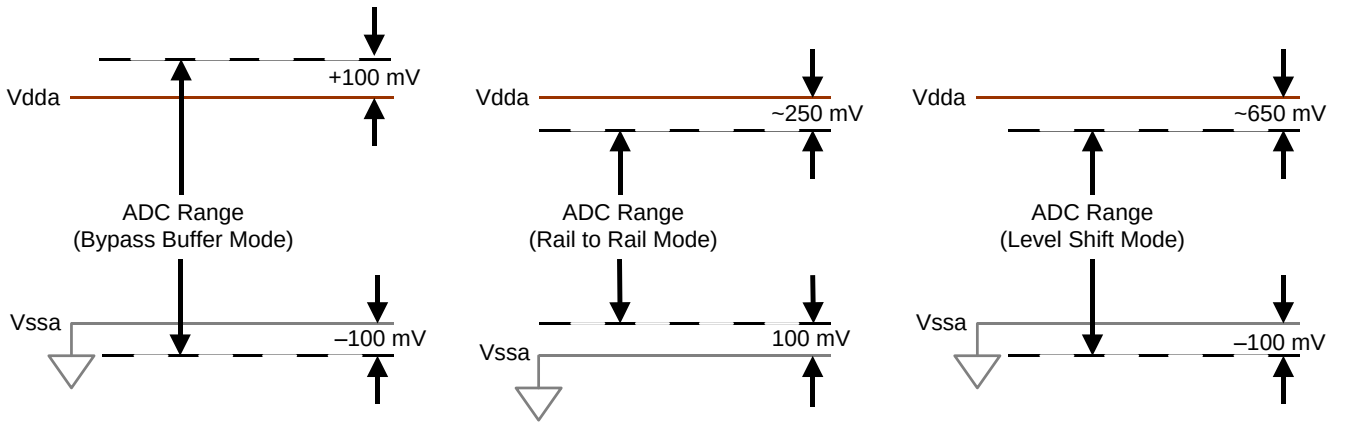
Buffer Gain	Description
1	Sets input buffer gain at 1.
2	Sets input buffer gain at 2.
4	Sets input buffer gain at 4.
8	Sets input buffer gain at 8.

Buffer Mode

Selects the ADC input buffer mode. The ADC has maximum sample rate when the buffer is used. The unbuffered modes have slightly reduced bandwidth.

Buffer Mode	Description
Bypass Buffer	Disables the input buffer gain. If selected, the buffer will be disabled to reduce the overall power consumption. The **Buffer Gain** parameter will not have any effect if this mode is selected. If this mode is selected then input impedance is reduced to less than 500 K Ohms. See the following diagram for specifics on range.
Rail to Rail	Sets the input buffer mode to rail to rail. See the following diagram for specifics on range.
Level-Shift	Sets the input buffer mode to Level-Shift. Both positive and negative input buffers will be used. Level-Shift mode allows you to go below the Vssa but not all the way to Vdda. See the following diagram for specifics on range.

The following figures show the ADC range for all Buffer Modes.



Reference

Vref

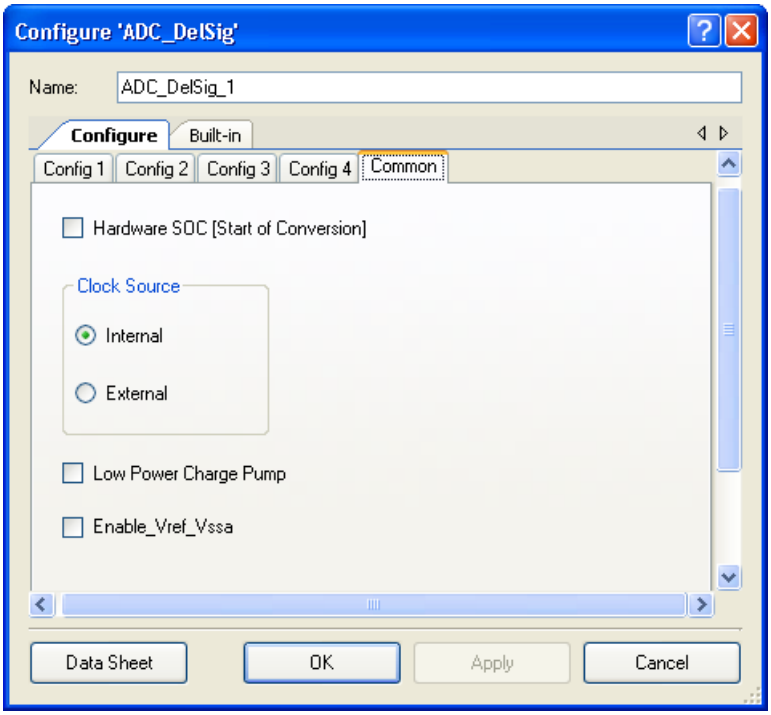
This parameter selects the ADC_DelSig reference voltage and configuration. The reference voltage sets the range of the ADC.

ADC_Reference	Description
Internal Vref	Use the internal 1.024 V reference (default)
Internal Vref, bypassed on P0[3] *	Use internal 1.024 V reference, and allow an external bypass capacitor to be connected on pin P0[3].
Internal Vref, bypassed on P3[2] *	Use internal 1.024 V reference, and allow an external bypass capacitor to be connected on pin P3[2].
External Vref on P0[3]	Use external reference on pin P0[3]. See DC Characteristics section for allowable range.
External Vref on P3[2]	Use external reference on pin P3[2].

* The accuracy and signal to noise ratio are highly dependent on the quality of the reference. The reference supplied to the ADC can be bypassed on either port P0[3] or port P3[2]. The use of an external bypass capacitor is recommended for resolutions of 14 bits and greater. Your chosen reference pin will be automatically configured to use this option. Refer to the following graphic/table for recommendations on bypass cap values.

TBD: Graph of cap value vs. noise. Noise vs cap value vs resolution. Bandwidth

Common Settings



Hardware SOC (Start of Conversion)

The ADC may be started by firmware with the ADC_StartConvert() function or by triggering with a hardware signal. Checking the Hardware SOC parameter enables an external pin to start conversion. When Hardware SOC is enabled the pin is displayed on the component; when not enabled, no pin is displayed. The conversion starts on the rising edge of the signal on the pin. Conversions continue until ADC_StopConvert() is called. By default, Hardware SOC is disabled. If a conversion is already in process, a Hardware SOC trigger is ignored.

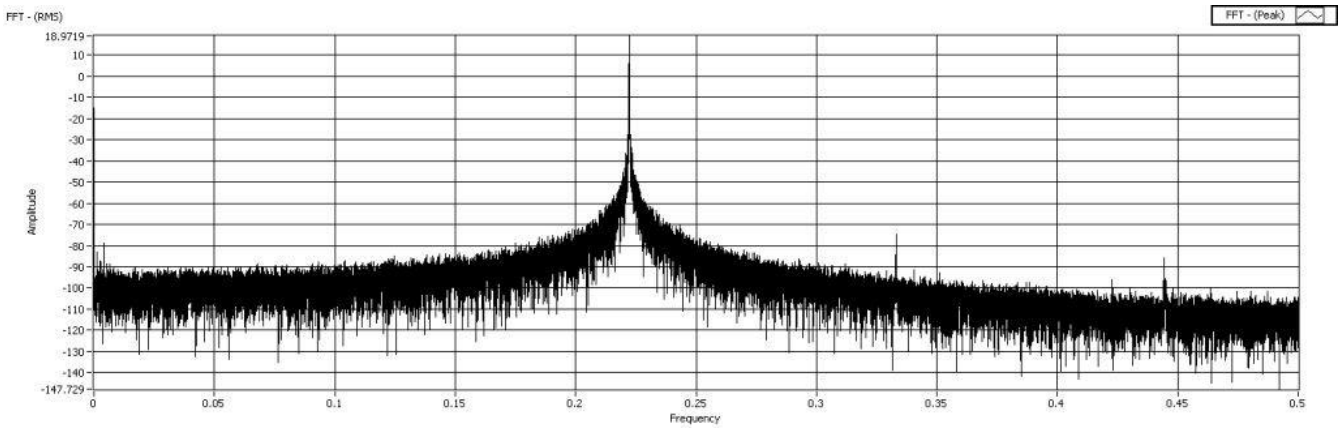
Clock Source

The ADC can be clocked by a source internal to the ADC component, a source external to the component but internal to the chip using a standard clock component or UDB, or by a source external to the chip. The internal or external clock selection is made via a radio button. When external clock is enabled, a clock input pin is displayed on the ADC schematic symbol. External clocks must have 50% duty cycle, the internal clock is guaranteed by design to have the correct duty cycle.

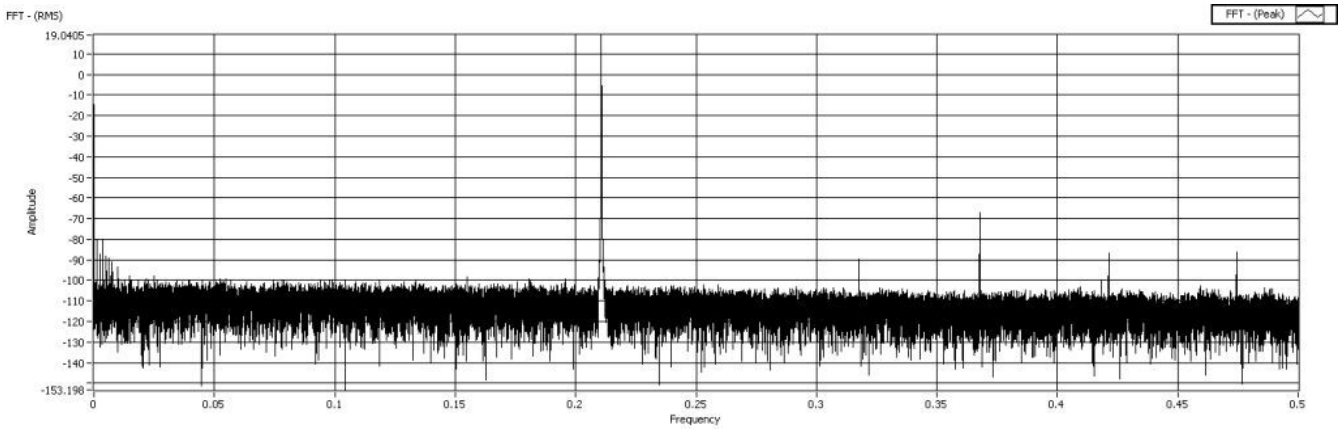
Clock stability is important for achieving low noise operation. One of the effects of jitter is substantial spreading of the signal. These are clearly shown in the following FFTs. The signal to noise ratio (SNR) of the ADC can be significantly improved with the use of an external clock.

Figure 3: Noise vs Clock

16 bit 65536 Point FFT - Internal Clock, Fin=10 kHz



16 bit 65536 Point FFT - External Clock, Fin=10 kHz



Low Power Charge-Pump

Portions of the ADC_DelSig are powered by an internal charge pump which operates on a high frequency clock. When this option is selected, a lower frequency clock is routed to the internal charge pump, which can reduce power This may save 100 to 300 uA, but will consume an additional clock resource. If selected, the ADC_Start and ADC_Stop functions will control this clock. By default, the Low Power Charge Pump is disabled.

Enable_Vref_Vssa

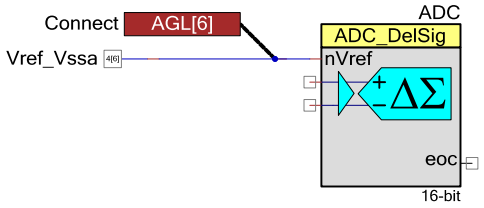
This parameter will allow you to connect the negative input of the ADC's reference Vssa to the analog global AGL[6]. For high accurate systems the Vref_Vssa can be connected to the external Vssa to eliminate any small difference between the on-chip Vssa and the off-chip Vssa. This small difference may cause a gain error in the ADC.

Vref_Vssa is an advanced feature that is useful when using an external reference supplied to the ADC. The Vref_Vssa connection can be routed through the analog routing fabric and brought out

to a pin. This enables you to connect to an external reference and eliminate any offset in the reference supplied to the ADC due to I*R drops in the Vssa pin and bonding wire.

The Vref_Vssa makes a direct connection to Analog Global Left 6 (AGL[6]) [See the analog routing diagram in the device data sheet for more information]. AGL[6] makes direct connections to pins P4[6], P4[2], P0[6] and P0[2]. For the best possible performance, you should ensure that Vref_Vssa is connected to one of these pins. Placing Vref_Vssa on another pin will cause extra routing resources to be consumed and extra resistance will be added in series with the connection.

The manual analog routing system (MARS) components allow you to a add rule check that ensures only the specified pins can be used. By placing an Analog Resource Constraint on the Vref_Vssa net, only resources that make direct connections to that net can be used. If you place a pin on that net that is not directly connected to AGL[6], the tool will generate an error during the build process. The error will indicate that you have connected a resource to that net that has no direct connection to the pin and therefore cannot route. The following is an example:



Placement

Not applicable

Resources

The ADC_DelSig uses a decimator, Delta-Sigma modulator, and a clock source. If an external reference or reference bypass is selected, P0[3] or P3[2] can be used for the external reference or bypass capacitor.

Resolution	Digital Blocks					API Memory (Bytes)		Pins (per External I/O)
	Datapaths	Macro cells	Status Registers	Control Registers	Counter7	Flash	RAM	
8-20 Bits	0	0	0	0	0	TBD	TBD	-

Application Programming Interface

Application Programming Interface (API) routines allow you to configure the component using software. The following table lists and describes the interface to each function. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name "ADC_DelSig_1" to the first instance of a component in a given design. You can rename the instance to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol. For readability, the instance name used in the following table is "ADC".

Function	Description
void ADC_Init(void)	Initializes/restores default configuration provided with the customizer.
void ADC_Enable(void)	Enables the ADC.
void ADC_Start(void)	Power up ADC and reset all states.
void ADC_Stop(void)	Stop ADC conversions and power down.
void ADC_SetBufferGain(uint8 gain)	Select input buffer gain (1,2,4,8)
void ADC_StartConvert(void)	Start conversion.
void ADC_StopConvert(void)	Stop Conversions
void ADC_IRQ_Enable(void)	Enables interrupts at end of conversion.
void ADC_IRQ_Disable(void)	Disables interrupts.
uint8 ADC_IsEndConversion(uint8 retMode)	Returns a non-zero value if conversion is complete.
int8 ADC_GetResult8(void)	Returns an 8-bit conversion result, right justified.
int16 ADC_GetResult16(void)	Returns a 16-bit conversion result, right justified.
int32 ADC_GetResult32(void)	Returns a 32-bit conversion result, right justified.
void ADC_SetOffset(int32 offset)	Sets the offset used by the ADC_CountsTo_mVolts, ADC_CountsTo_uVolts, and ADC_CountsTo_Volts functions.
void ADC_SelectConfiguration(uint8 config uint8 restart)	Sets one of up to four ADC configurations.
void ADC_SetGain(int32 adcGain)	Sets the gain used by the ADC_CountsTo_mVolts, ADC_CountsTo_uVolts, and ADC_CountsTo_Volts functions.
int32 ADC_CountsTo_mVolts(int32 adcCounts)	Convert ADC counts to mVolts.
Int32 ADC_CountsTo_uVolts(int32 adcCounts)	Convert ADC counts to uVolts.
float ADC_CountsTo_Volts(int32 adcCounts)	Convert ADC counts to floating point Volts.

Function	Description
void ADC_Sleep(void)	Stop ADC operation and saves the user configuration.
void ADC_Wakeup(void)	Restores and enables the user configuration.
void ADC_SaveConfig(void)	Save the current configuration.
void ADC_RestoreConfig(void)	Restores the configuration.

Global Variables

Variable	Description
ADC_initVar	Indicates whether the ADC has been initialized. The variable is initialized to 0 and set to 1 the first time ADC_Start() is called. This allows the component to restart without reinitialization after the first call to the ADC_Start() routine. If reinitialization of the component is required, then the ADC_Init() function can be called before the ADC_Start() or ADC_Enable() functions.
ADC_offset	The ADC_offset variable is used for offset calibration purpose. Initially this variable is set to zero. Application can modify it using ADC_SetOffset function. Affects only the ADC_CountsTo_Volts, ADC_CountsTo_mVolts, ADC_CountsTo_uVolts functions by subtracting the given offset.
ADC_CountsPerVolt	The ADC_countsPerVolt variable is used for gain calibration purpose. Initially this variable is calculated for default ADC configuration. The calculated value depends on resolution, input range and voltage reference. Application can modify it using ADC_SetGain() function. Affects only the ADC_CountsTo_Volts, ADC_CountsTo_mVolts, ADC_CountsTo_uVolts functions by supplying the correct conversion between ADC counts and the applied input voltage.
ADC_convDone	The ADC_conDone variable is used as the software flag for checking the ADC conversion in case of single sample conversion mode for resolutions above 16 bit.

void ADC_Init(void)

Description: Initializes component's parameters and variables to the parameters set in the customizer of the component placed onto the schematic. You are not required to call this function if ADC_Start is called.

Parameters: None

Return Value: None

Side Effects: All registers will be reset to their initial values. This will re-initialize the component.

void ADC_Enable(void)

Description: Enables the clock and power for ADC.

Parameters: None

Return Value: None

Side Effects: None

void ADC_Start(void)

Description: Configures and powers up the ADC, but does not start conversions. By default the ADC is configured for Config1 unless the ADCSelectCOnfig() function selects an alternate configuration.

Parameters: None

Return Value: None

Side Effects: None

void ADC_Stop(void)

Description: Disables and powers down the ADC.

Parameters: None

Return Value: None

Side Effects: None

void ADC_SetBufferGain(uint8 gain)

Description: Sets the input buffer gain.

Parameters: (uint8) gain: Input gain setting. See table below for valid gain constants.

Gain Options	Description
ADC_BUF_GAIN_1X	Set input buffer gain to 1.
ADC_BUF_GAIN_2X	Set input buffer gain to 2.
ADC_BUF_GAIN_4X	Set input buffer gain to 4.
ADC_BUF_GAIN_8X	Set input buffer gain to 8.

Return Value: None

Side Effects: None

void ADC_StartConvert(void)

Description: Forces the ADC to initiate a conversion. If in Single Sample mode, one conversion will be performed then the ADC will halt. If in one of the other three conversion modes, the ADC will run continuously.
If the StartConvert function is called while the conversion is in progress, the next conversion start is queued and a new conversion will start after finishing the current conversion. If you want to start a new conversion without waiting for the current conversion to finish, then stop the current conversion by calling StopConvert. After stopping the conversion, restart the conversion by calling StartConvert.

Parameters: None

Return Value: None

Side Effects: None

void ADC_StopConvert(void)

Description: Forces the ADC to stop all conversions. If the ADC is in the middle of the current conversion, the ADC will be reset and not provide a result for that partial conversion.

Parameters: None

Return Value: None

Side Effects: None

void ADC_IRQ_Enable(void)

Description: Enables interrupts to occur at the end of a conversion. Global interrupts must also be enabled for the ADC interrupts to occur. To enable global interrupts, use the enable global interrupt macro "`CYGlobalIntEnable;`" in *main.c*, prior to when interrupts should occur. See Interrupt Service Routine section.

Parameters: None

Return Value: None

Side Effects: Enables interrupts to occur. Reading the result will clear the interrupt.

void ADC_IRQ_Disable(void)

Description: Disables interrupts at the end of a conversion.

Parameters: None

Return Value: None

Side Effects: None

uint8 ADC_IsEndConversion(uint8 retMode)

Description: Check for ADC end of conversion. This function provides the programmer with two options. In one mode this function immediately returns with the conversion status. In the other mode, the function does not return (blocking) until the conversion has completed.

Parameters: (uint8) retMode: Check conversion return mode. See table below for options.

Options	Description
ADC_RETURN_STATUS	Immediately returns conversion result status.
ADC_WAIT_FOR_RESULT	Does not return until ADC conversion is complete.

Return Value: (uint8) If a non-zero value is returned, the last conversion has completed. If the returned value is zero, the ADC is still calculating the last result.

Side Effects: None

int8 ADC_GetResult8(void)

Description: This function will return the result of an 8-bit conversion. If the resolution is set greater than 8-bits, the LSB of the result will be returned. When the ADC is configured for 8-bit single ended mode, the ADC_GetResult16() function should be used instead. This function returns only signed 8-bit values. The maximum positive signed 8-bit value is 127, but in singled ended 8-bit mode, the maximum positive value is 255.

Parameters: None

Return Value: (int8) The LSB of the last ADC conversion.

Side Effects: None

int16 ADC_GetResult16(void)

Description: Returns a 16-bit result for a conversion with a result that has a resolution of 8 to 16 bits. If the resolution is set greater than 16-bits, it will return the 16 least significant bits of the result. When the ADC is configured for 16-bit single ended mode, the ADC_GetResult32() function should be used instead. This function returns only signed 16-bit result, which allows a maximum positive value of 32767, not 65535.

Parameters: None

Return Value: (int16) The 16-bit result of the last ADC conversion.

Side Effects: None

int32 ADC_GetResult32(void)

Description: Returns a 32-bit result for a conversion with a result that has a resolution of 8 to 20 bits.

Parameters: None

Return Value: (int32) Result of the last ADC conversion.

Side Effects: None

void ADC_SetOffset(int32 offset)

Description: Sets the ADC offset which is used by the functions ADC_CountsTo_uVolts, ADC_CountsTo_mVolts, and ADC_CountsTo_Volts to subtract the offset from the given reading before calculating the voltage conversion.

Parameters: (int32) offset: This value is a measured value when the inputs are shorted or connected to the same input voltage.

Return Value: None.

Side Effects: Affects the ADC_CountsTo_uVolts, ADC_CountsTo_mVolts, and ADC_CountsTo_Volts functions by subtracting the given offset.

void ADC_SetGain(int32 adcGain)

Description: Sets the ADC gain in counts per volt for the voltage conversion functions below. This value is set by default by the reference and input range settings. It should only be used to further calibrate the ADC with a known input or if an external reference is used.

Parameters: (int32) adcGain: ADC gain in counts per volt.

Return Value: None.

Side Effects: Affects only the ADC_CountsTo_uVolts, ADC_CountsTo_mVolts, and ADC_CountsTo_Volts functions by supplying the correct conversion between ADC counts and voltage.

void ADC_SelectConfiguration(uint8 config uint8 restart)

Description: Sets one of up to four ADC configurations. Before setting the new configuration, the ADC is stopped and powered down. After setting the new configuration, the ADC can be powered and conversion can be restarted depending up on the value of second parameter restart. If the value of this parameter is 1, then ADC will be restarted. If this value is zero, then user must call ADC_Start and ADC_StartConvert() to restart the conversion.

Parameters: (uint8) config: Configuration option between 1 and 4.

(uint8) restart: Restart option. 1 means start the ADC and restart the conversion. 0 means do not start the ADC and conversion.

Return Value: None.

Side Effects: None

int32 ADC_CountsTo_mVolts(int32 adcCounts)

Description: Converts the ADC output to mVolts as a 16-bit integer. For example, if the ADC measured 0.534 volts, the return value would be 534 mVolts.

Parameters: (int32) adcCounts: Result from the ADC conversion.

Return Value: (int32) Result in mVolts.

Side Effects: None

int32 ADC_CountsTo_uVolts(int32 adcCounts)

Description: Converts the ADC output to uVolts as a 32-bit integer. For example, if the ADC measured –0.02345 Volts, the return value would be –23450 uVolts.

Parameters: (int32) adcCounts: Result from the ADC conversion.

Return Value: (int32) Result in uVolts.

Side Effects: None

float ADC_CountsTo_Volts(int32 adcCounts)

Description: Converts the ADC output to Volts as a floating point number. For example, if the ADC measure a voltage of 1.2345 Volts, the returned result would be +1.2345 Volts.

Parameters: (int32) adcCounts: Result from the ADC conversion.

Return Value: (float) Result of the last ADC conversion.

Side Effects: None

void ADC_Sleep(void)

Description: Stops the ADC operation and saves the configuration registers and component enable state. Should be called just prior to entering sleep.

Parameters: None

Return Value: None

Side Effects: None

Note If you put the ADC hardware to sleep using the CyPmSleep() API, then after coming out of sleep ADC_Start() and ADC_StartConvert() need to be executed to restart conversions.

void ADC_Wakeup(void)

Description: Restores the component enable state and configuration registers. Should be called just after awaking from sleep mode.

Parameters: None

Return Value: None

Side Effects: Calling this function without previously calling ADC_Sleep() may lead to unpredictable results.

void ADC_SaveConfig(void)

Description: Save the current configuration of ADC non-retention registers.

Parameters: None

Return Value: None

Side Effects: None.

void ADC_RestoreConfig(void)

Description: Restores the configuration of ADC non-retention registers.

Parameters: None

Return Value: None

Side Effects: Calling this function without previously calling ADC_SaveConfig() or ADC_Sleep() may produce unexpected behavior.

DMA Information

The DMA component can be used to transfer converted results from the ADC_DelSig register to RAM or another component, such as the Digital Filter Block (DFB). The DMA data request signal (DRQ) should be connected to EOC pin from ADC. The DMA Wizard can be used to configure DMA operation as follows:

Name of DMA source / destination in DMA Wizard	Direction	DMA Req Signal	DMA Req Type	Description
ADC_DelSig_DEC_SAMP_PTR	source	EOC	Edge	Receives 1 byte conversion result for input analog value that has a resolution of 8 bits.
ADC_DelSig_DEC_SAMPM_PTR	source	EOC	Edge	Receives 2 byte conversion result for input analog value that has a resolution of 9-16 bits.
ADC_DelSig_DEC_SAMPH_PTR	source	EOC	Edge	Receives 3 byte conversion result for input analog value that has a resolution of 17-20 bits.

Sample Firmware Source Code

The following is a C language example demonstrating the basic functionality of the ADC_DelSig component. This example assumes the component has been placed in a design with the default name "ADC_DelSig_1."

Note If you rename your component you must also edit the example code as appropriate to match the component name you specify.

```
#include <device.h>

void main()
{
    int16  result;
    ADC_DelSig_1_Start();
    ADC_DelSig_1_StartConvert();
    ADC_DelSig_1_IsEndConversion(ADC_DelSig_1_WAIT_FOR_RESULT);
    result = ADC_DelSig_1_GetResult16();
}
```

Example for Multiplexing Analog Input at Runtime

```
#include <device.h>

void main()
{
    int16 result;
    ADC_DelSig_1_Start();
    AMux_1_Start( ); /* Reset all channels */
    AMux_1_Select(0); /* Connect channel 1 */

    ADC_DelSig_1_StartConvert();
    ADC_DelSig_1_IsEndConversion(ADC_DelSig_1_WAIT_FOR_RESULT);
    ADC_DelSig_1_StopConvert();
    result = ADC_DelSig_1_GetResult16();/* Get the result */

    AMux_1_Select(1); /* Connect channel 2 */
    ADC_DelSig_1_StartConvert();
    ADC_DelSig_1_IsEndConversion(ADC_DelSig_1_WAIT_FOR_RESULT);
    result = ADC_DelSig_1_GetResult16();
}
```

Interrupt Service Routine

The ADC_DelSig contains a blank interrupt service routine in the *ADC_DelSig_1_INT.c* file, where "ADC_DelSig_1" is the instance name. You may place custom code in the designated areas to perform whatever function is required at the end of a conversion. A copy of the blank interrupt service routine is shown below.

Place custom code between the " /* `#START MAIN_ADC_ISR` */ " and "/* `#END` */ " comments. This ensures that the code will be preserved when a project is regenerated.

```
CY_ISR( ADC_DelSig_1_ISR )
{
    /* Place user ADC ISR code here. This can be a good place */
    /* to place code that is used to switch the input to the  */
    /* ADC. It may be good practice to first stop the ADC     */
    /* before switching the input then restart the ADC.       */

    /* `#START MAIN_ADC_ISR` */
    /* Place user code here. */
    /* `#END`  */
}
```

A second designated area is made available to place variable definitions and constant definitions.

```
/* System variables */

/* `#START ADC_SYS_VAR`  */
    /* Place user code here. */
/* `#END`  */
```

The following is example code using an interrupt to capture data. The main is similar to the previous example except that the interrupt must be enabled.

```
#include <device.h>

int16  result = 0;
uint8 dataReady = 0;
void main()
{
    int16 newReading = 0;
    CYGlobalIntEnable;           /* Enable Global interrupts */
    ADC_DelSig_1_Start();        /* Initialize ADC */
    ADC_DelSig_1_IRQ_Enable();   /* Enable ADC interrupts */
    ADC_DelSig_1_StartConvert(); /* Start ADC conversions */
    for(;;)
    {
        if (dataReady != 0)
        {
           dataReady = 0;
           newReading = result;
           /* More user code */
        }
    }
```

```
}
```

Interrupt code segments in the file *ADC_DelSig_1_INT.c*.

```
    /*********************************
    *      System variables
    *********************************/
    /* `#START ADC_SYS_VAR`  */
     extern int16 result;
     extern uint8 dataReady;
     /* `#END`  */

CY_ISR(ADC_DelSig_1_ISR )
{
    /*******************************************/
    /* Place user ADC ISR code here.           */
    /* This can be a good place to place code   */
    /* that is used to switch the input to the  */
    /* ADC. It may be good practice to first    */
    /* Stop the ADC before switching the input  */
    /* then restart the ADC.                   */
    /*******************************************/
    /* `#START MAIN_ADC_ISR`  */
        result = ADC_DelSig_1_GetResult16();
        dataReady = 1;
    /* `#END`  */
}
```

Functional Description

The Delta Sigma Channel is made up the following blocks:

- A high input impedance, front-end buffer (with programmable gain) that can be bypassed (and powered down) when not required.
- A fully differential programmable third-order switched capacitor modulator.
- A downstream digital filtering option consisting of: A fourth-order Cascaded Integrator-Comb (CIC) filter (also called the decimator). The post processing engine (cicdec4_pproc) optionally performs gain, offset and simple FIR filtering functions on the data as it leaves the CIC filter.
- ANAIF - Analog interface Logic block consists of the register control for the input buffer and the modulator. The ANAIF also converts the 8-bit wide thermometer output from the modulator into 2's complement format, which is 4 bits wide. This 4-bit wide 2's complement code is sent to the decimator.

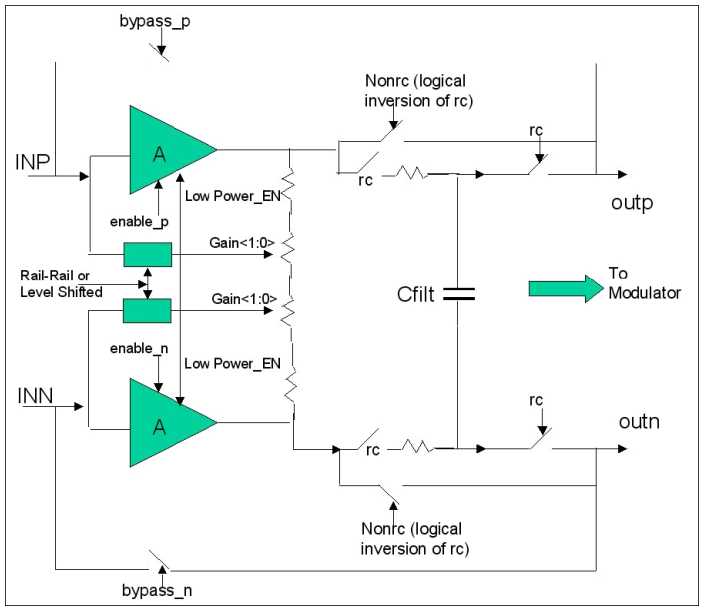
Without an input buffer, a switched capacitor input stage would consume current to charge the capacitor during each cycle. In that case, the equivalent input resistance would be of the order of 1/fs*C, or 1/(3MHz)(5pF) = 66 k ohms. Many sensor applications require a much higher

impedance to achieve an accurate reading. Therefore, an input buffer is made a part of the Delta Sigma Channel.

The input buffer must also deal with signals closer to ground in some applications and must work closer to the supply rail in others. Input buffer architecture comprises two single-ended buffers used to create a differential channel. Either buffer can be selected for the channel. When the channel operates in a single-ended mode, one of the inputs is connected to Ground Rail, and the corresponding buffer is bypassed. The buffers can be also be individually powered down. There are two main modes of operation for the buffer:

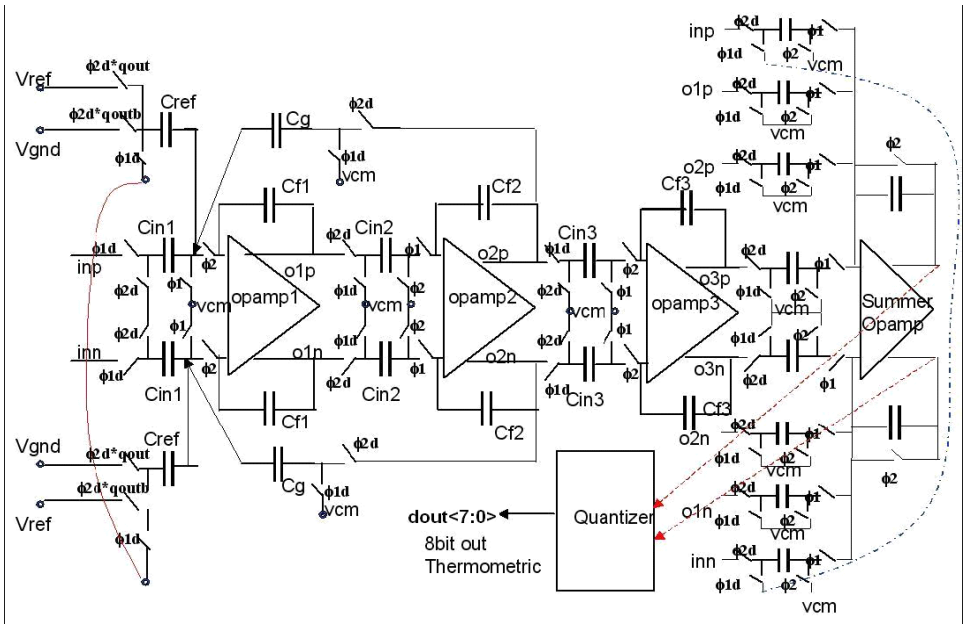
- Level-Shifted Mode: Buffer output can be level shifted up from the input when the input is close to 0V input common mode voltage range.
- Rail - Rail Mode: This mode is used when input is rail-to rail.

Figure 4: ADC buffer structure



The switched capacitor implementation is shown in the Figure. A dynamic element matching (DEM) block shapes the errors due to mismatch in the switched capacitor DAC feedback of the modulator, when operating under 9 level quantization. When the buffer and modulator are correctly configured for a given application, the variable-level programmable quantizer (level 2, 3, or 9) in the modulator produces “the-quantized” bit-stream. This quantized bit-stream is 8 bits wide and in the thermometric format. The conversion of thermometer quantizer code into 2’s complement format, for use in decimator (Sinc4 and Sinc1), is performed in the ANAIF block.

Figure 5: Switched Capacitor Delta Sigma Modulator Structure



Registers

Sample Registers

The ADC results may be between 8 and 24 bits of resolution. The output is divided into three 8-bit registers. The CPU or DMA may access these register to read the ADC result.

ADC_DEC_SAMP (ADC Output Data Sample Low Register)

Bits	7	6	5	4	3	2	1	0
Value	Data[7:0]							

ADC_DEC_SAMPM (ADC Output Data Sample Middle Register)

Bits	7	6	5	4	3	2	1	0
Value	Data[15:8]							

ADC_DEC_SAMPH (ADC Output Data Sample High Register)

Bits	7	6	5	4	3	2	1	0
Value	Data[23:16]							

DC and AC Electrical Characteristics

The following values are indicative of expected performance and based on initial characterization data. Unless otherwise specified operating conditions are:

- fclk = 3.072 MHz for resolution = 16 to 20 bits
- fclk = 6.144 MHz for resolution = 8 to 15 bits
- Vdda = 3.3 V
- Reference = 1.024 internal reference bypassed on P3[2] or P0[3]Operating Temperature = –40° to +85° C for min/max limits
- Operating Temperature = 25° C for typical values

DC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
	Resolution		8	-	20	bits
	Number of channels single ended		na	na	# GPIO	
	Number of channels differential	Differential pair is formed using a pair of GPIOs.	na	na	# GPIO/2	
	Monotonic		Yes	na	na	
	ADC gain error, independent of buffer gain	Differential range = +/- 2.048 V (Vref * 2)	na	tbc	tbc	%
		Differential range = +/- 1.024 V (Vref)	na	tbc	tbc	%
		Differential range = +/- 0.512 V (Vref / 2)	na	tbc	tbc	%
		Differential range = +/- 0.256 V (Vref / 4)	na	tbc	tbc	%
		Differential range = +/- 0.125 V (Vref / 8)	na	tbc	tbc	%
		Differential range = +/- 0.0625V (Vref / 16)	na	tbc	tbc	%
		Single-ended Vssa to 1.024 V (Vref)	na	tbc	tbc	%
		Single-ended Vssa to 2.048 V (Vref * 2)	na	tbc	tbc	%
		Single-ended Vssa to Vdda (ratiometric)	na	tbc	tbc	%
		Single-ended Vssa to Vref * 6 Note 1	na	tbc	tbc	%
	Buffer gain error (Note 2)	Gain = 1	na	tbc	tbc	%

Parameter	Description	Conditions	Min	Typ	Max	Units
		Gain = 2	na	tbc	tbc	%
		Gain = 4	na	tbc	tbc	%
		Gain = 8	na	tbc	tbc	%
	ADC input offset voltage		na	tbc	tbc	mV
	Buffer input offset voltage (Note 3)		na	tbc	tbc	mV
TCVos_ADC	ADC TC input offset voltage		na	tbc	tbc	uV/C
TCVos_Buf	Buffer TC input offset voltage		na	tbc	tbc	uV/C
	Input voltage range - single ended		Vssa	--	Vdda	V
	Input voltage range - differential		Vssa	--	Vdda	V
	Input voltage range - differential buffered		Vssa	--	Vdda-1	V
Rin_ADC16	ADC input resistance	16 bit, Range=+/-1.024V	na	74	na	kohm
Rub_ADC12	ADC input resistance	12 bit, Range=+/-1.024V	na	148	na	kohm
	Buffer input resistance		10	na	na	Megohm
	External reference range		TBD	1.024	TBD	V
	External reference input impedance		na	TBD	na	kohm
External Vref	Allowable external Vref range		0.9	--	1.3	V
	Current consumption					
Idd_20bit	Current consumption, 20 bit	187 sps, unbuffered	na	tbc	tbc	mA
Idd_16bit	Current consumption, 16 bit	48 ksps, unbuffered	na	tbc	tbc	mA
Idd_12bit	Current consumption, 12 bit	192 ksps, unbuffered	na	tbc	tbc	mA
Idd_8bit	Current consumption, 8 bit	384 ksps, unbuffered	na	tbc	tbc	mA
Ibuff	Buffer current consumption		na	tbc	tbc	mA

AC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
	Startup time	Continuous mode	na	na	4	samples
PSRRb	Power supply rejection ratio, buffered	Buffer gain=1, 16 bit, Range = +/-1.024V	90	na	na	dB
PSRRub	Power supply rejection ratio, unbuffered	16 bit, Range=+/1.024V		na	na	
CMRRb	Common mode rejection ratio, buffered	Buffer gain=1, 16 bit, Range= +/-1.024	90	na	na	dB
CMRRub	Common mode rejection ratio, unbuffered	16 bit, Range=+/1.024V		na	na	

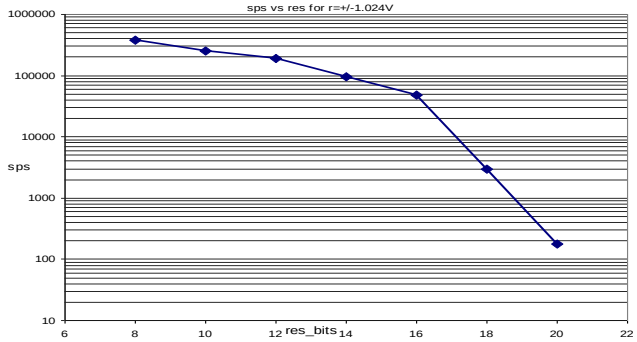
Parameter	Description	Conditions	Min	Typ	Max	Units
THD	Total harmonic distortion	Buffer gain=1, 16 bit, Range= +/-1.024V	na	tbc	0.0032	%
20 bit resolution mode						
SR20	Sample rate	Range=+/-1.024V, unbuffered	7.8	na	187	sps
BW20	Input bandwidth at max sample rate	Range=+/-1.024V, unbuffered	na	40	na	Hz
INL20	Integral non linearity	Range=+/-1.024V, unbuffered	na	tbc-	16	LSB
DNL20	Differential non linearity	Range=+/-1.024V, unbuffered	na	tbc	1	LSB
SNR20int	Signal to noise ratio, 20 bit, internal reference (Note 4)	Range=+/-1.024V, unbuffered	110	tbc	na	dB
SNR20ext	Signal to noise ratio, 20 bit, external reference	Range=+/-1.024V, unbuffered	tbc	tbc	na	dB
16-Bit Resolution Mode						
SR16	Sample rate	Range=+/-1.024V, unbuffered	2	na	48	ksps
BW16	Input bandwidth at max sample rate	Range=+/-1.024V, unbuffered	na	11	na	kHz
INL16	Integral non linearity	Range=+/-1.024V, unbuffered	na	tbc-	1	LSB
DNL16	Differential non linearity	Range=+/-1.024V, unbuffered	na	tbc	1	LSB
SNR16int	Signal to noise ratio, 16 bit, internal reference	Range=+/-1.024V, unbuffered	89	tbc	na	dB
SNR16ext	Signal to noise ratio, 16 bit, external reference	Range=+/-1.024V, unbuffered	tbc	tbc	na	dB
12-Bit Resolution Mode						
SR12	Sample rate, continuous - high power	Range=+/-1.024V, unbuffered	4		192	ksps
BW12	Input bandwidth at max sample rate	Range=+/-1.024V, unbuffered	na	44	na	kHz
INL12	Integral non linearity	Range=+/-1.024V, unbuffered	na	tbc	1	LSB
DNL12	Differential non linearity	Range=+/-1.024V, unbuffered	na	tbc	1	LSB
SNR12int	Signal to noise ratio, 12 bit, internal reference	Range=+/-1.024V, unbuffered	70	tbc	na	dB
8 Bit Resolution Mode						
SR8	Sample rate, continuous - high power	Range=+/-1.024V, unbuffered	8		384	ksps
BW8	Input bandwidth at max sample rate	Range=+/-1.024V, unbuffered	na	88	na	kHz
INL8	Integral non linearity	Range=+/-1.024V, unbuffered	na	tbc	1	LSB
DNL8	Differential non linearity	Range=+/-1.024V, unbuffered	na	tbc	1	LSB
SNR8int	Signal to noise ratio, 8 bit, internal reference	Range=+/-1.024V, unbuffered	70	tbc	na	dB

Notes

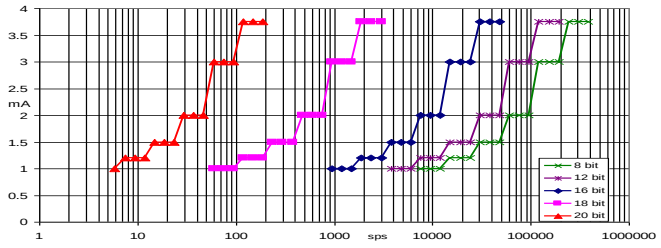
1. Vssa to 6*Vbg range is used for direct measurement of Vdda power supply. Actual scale is limited to Vdda.
2. Total gain error is sum of ADC error and buffer error.
3. Total offset voltage error is sum of buffer Vos and ADC Vos
4. SNR definition

Figures

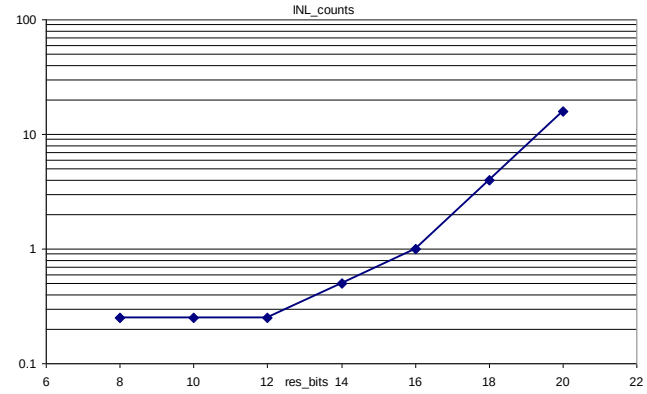
Maximum sample rate, Range=+/-1.024V
X axis: resolution, bits
Y axis: sps, log scale



Idd vs sps, Range=±1.024V
X axis: sps, log scale
Y axis: current, mA
1. Idd for 8 bits
2. Idd for 12 bits
3. Idd for 16 bits
4. Idd for 18 bits
5. Idd for 20 bits



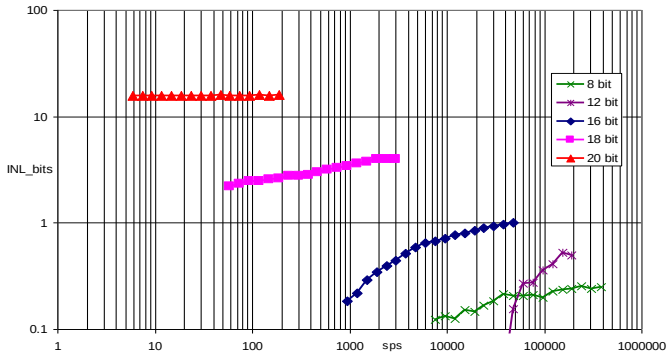
INL at maximum sample rate
X axis: sps, log scale
Y axis: INL, bits, log scale



Integral Non Linearity

Range	16 bit	12 bit
Single Ended		
0 to Vref	1	0.3
0 to Vref*2	1.2	0.4
Vss to Vdd	1.2	0.4
0 to Vref*6	1.3	0.5
Differential		
+/- Vref	1	0.3
+/- Vref*2	1.5	0.4
+/- Vref/2	1.2	0.5
+/- Vref/4	1.5	0.6
+/- Vref/8	1.8	1
+/- Vref/16	2	1.2

Integral Non Linearity, Range=+/-1.024V
X axis: samples per second
Y axis: INL, counts, log scale



Input Resistance, 16 bit, Unbuffered, vs sample rate
X axis: samples per second
Y axis: k ohms, log scale

Input Resistance, 12 bit, Unbuffered, vs sample rate
X axis: samples per second
Y axis: k ohms, log scale

Input Resistance, 8 bit, Unbuffered, vs sample rate
X axis: samples per second
Y axis: k ohms, log scale

Buffer Gain Error, Range=+/-1.024V vs Temp
X axis: Temp deg C, -40 to +95
Y axis: Error, %

(sinx/x)^4 Frequency Response, normalized to output sample rate = 48 kHz
X axis: freq, kHz, log scale
Y axis: dB

CMRR vs Frequency 16 bit, 48 ksps, Range=+/-1.024V X axis: freq, kHz, log scale Y axis: CMRR dB	PSRR vs Frequency 16 bit, 48 ksps, Range=+/-1.024V X axis: freq, kHz, log scale Y axis: CMRR dB
Noise histogram, 1000 samples, 20 bit, 180 sps, ext Ref, Vin=Vref/2, Range=+/-1.024V X axis: ADC counts Y axis: % in integer bins	Noise histogram, 1000 samples, 16 bit, 48ksps, ext Ref, Vin=Vref/2, Range=+/-1.024V X axis: ADC counts Y axis: % in integer bins
Noise histogram, 1000 samples, 16 bit, 48ksps, int Ref, Vin=Vref/2, Range=+/-1.024V X axis: ADC counts Y axis: % in integer bins	Noise histogram, 1000 samples, 12 bit, 192 ksps, int Ref, Vin=Vref/2, Range=+/-1.024V X axis: ADC counts Y axis: % in integer bins

Maximum sample rate, Range=+/-1.024V

20 bit RMS noise max sps ext ref

Single

sps	0 to Vref	0 to Vref*2	Vss to Vdd	0 to Vref*6	+/- Vref	+/- Vref*2	+/- Vref/2	+/- Vref/4	+/- Vref/8	+/- Vref/1
2.8125	1.3	1.1	0.6	1.0	0.6	1.1	0.4	1.2	0.5	1.1
5.625	1.1	1.2	1.0	0.8	0.8	0.8	1.2	1.4	0.8	0.8
11.25	2.0	1.5	1.7	2.0	2.1	1.3	1.3	1.7	1.9	1.1
22.5	2.5	2.6	3.0	2.7	2.4	2.8	3.0	2.9	3.1	3.2
45	5.4	5.1	5.3	5.4	5.2	5.4	5.4	5.3	5.3	4.7
90	9.8	9.1	9.3	9.8	9.4	10.0	9.7	9.5	9.4	9.3
180	18.7	18.4	18.7	18.4	18.5	18.6	18.5	18.4	18.1	18.4

Maximum sample rate, Range=+/-1.024V

16 bit RMS noise VBvolts at max sps, int ref

Single

sps	0 to Vref	0 to Vref*2	Vss to Vdd	0 to Vref*6	+/- Vref	+/- Vref*2	+/- Vref/2	+/- Vref/4	+/- Vref/8	+/- Vref/1
750	1.3	0.9	1.0	1.7	0.9	1.0	1.4	0.9	1.4	0.8
1500	2.3	1.8	2.4	1.8	1.8	1.8	2.3	2.2	1.8	2.4
3000	3.3	3.1	3.6	3.2	3.8	3.2	3.1	3.5	3.3	3.9
6000	7.0	6.6	6.6	6.8	6.5	7.0	6.4	7.0	6.0	6.5
12000	12.2	12.4	12.5	12.3	12.3	12.4	12.9	12.8	12.7	12.7
24000	24.4	24.2	24.4	24.2	24.5	24.1	24.2	24.9	24.8	24.1
48000	48.2	48.0	48.9	48.7	48.1	48.7	48.3	48.7	49.0	48.9

ENOB vs resolution

20 bit RMS noise max sps ext ref

Single

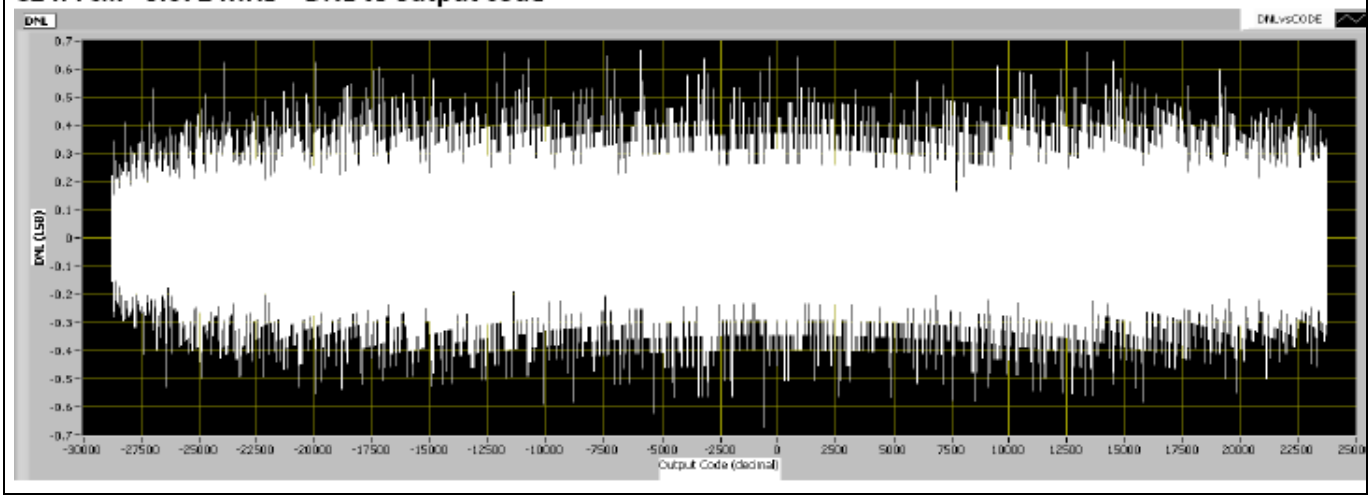
res	max sps	0 to Vref	0 to Vref*2	Vss to Vdd	0 to Vref*6	+/- Vref	+/- Vref*2	+/- Vref/2	+/- Vref/4	+/- Vref/8	+/- Vref/1
20	180	20.0	19.2	19.6	19.8	19.5	20.0	19.1	19.8	20.0	19.8
18	3000	17.5	18.0	17.6	17.1	17.7	17.5	17.8	17.5	17.4	17.0
16	48000	15.2	15.4	15.5	15.5	15.8	16.0	15.3	15.3	15.1	15.1
14	96000	13.4	13.1	13.3	13.5	13.3	13.8	13.2	13.9	13.9	13.9
12	192000	11.5	11.1	11.8	11.8	11.5	11.1	11.5	11.7	11.9	11.3
10	256000	9.9	9.8	9.9	9.2	9.9	9.0	9.4	9.4	9.6	9.7
8	384000	8.0	7.7	7.7	7.7	7.9	7.4	7.6	7.8	7.2	7.1

DNL vs Output code 16 bit, 48ksps

X axis: output code, -32768 to 32767

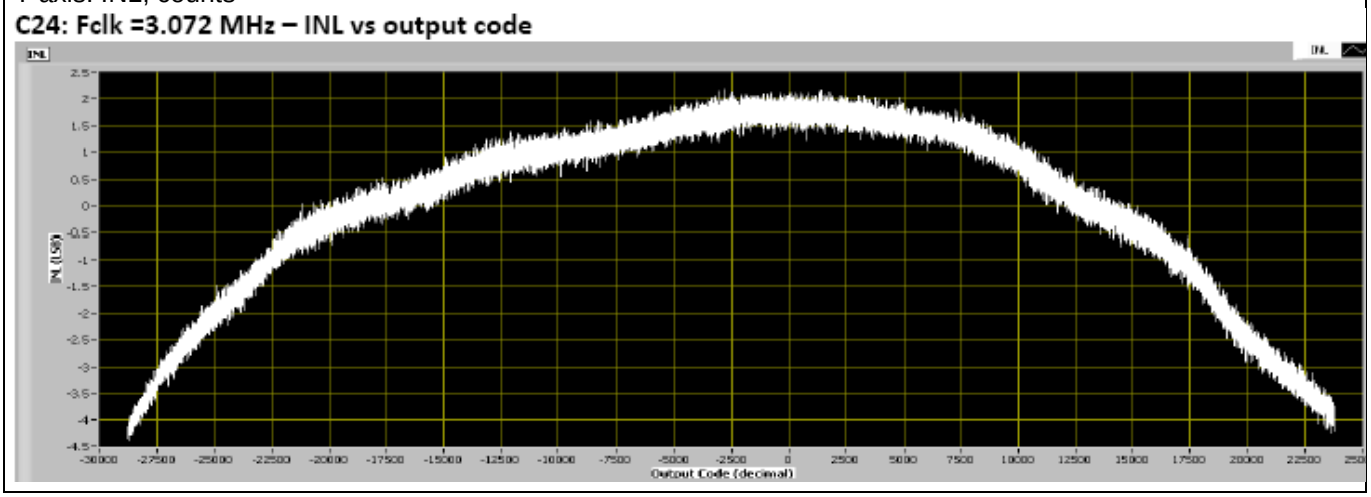
Y axis: DNL, counts

C24: Fclk =3.072 MHz – DNL vs output code



PRELIMINARY

INL vs Output code 16 bit, 48ksps
X axis: output code, -32768 to 32767
Y axis: INL, counts



Component Changes

This section lists the major changes in the component from the previous version.

Version	Description of Changes	Reason for Changes / Impact
2.0.b	Minor datasheet edit.	
2.0.a	Minor datasheet edit.	
2.0	Updated to support PSoC 3 ES3 or later and PSoC 5 ES1 or later.	This version supports PSoC 3 ES3 or later and PSoC 5 ES1 or later. Older versions of the component will display an error message when used with newer versions of the silicon.
	Several enumerated types in the component were changed from previous versions.	This change was made to improve functionality, and it will cause incompatibility issues in existing designs. If you update to the ADC_DelSig version 2.0, your design will likely not work unless you reconfigure various parameters.
	Added Sleep/Wakeup and Init/Enable APIs.	To support low power modes, as well as to provide common interfaces to separate control of initialization and enabling of most components.
	Added DMA capabilities file to the component.	This file allows the ADC_DelSig to be supported by the DMA Wizard tool in PSoC Creator.
	Added Keil function reentrancy support to the APIs.	Add the capability for customers to specify individual generated functions as reentrant.
	Edited the Configure Dialog.	Made Voltage Reference parameter editable. Added different configurations to support changing the configuration during run time. Dialog allows you to modify the voltage values when Vssa to Vdda input range is selected.
	Trim values are incorporated into the ADC implementation for the selected input ranges.	Trim values will be used to adjust the Decimator gain to improve the performance of ADC.
	Added Constants to the header file for easier use.	The ADC component now has Constants such as reference used, gain set, mode used, sample rate used, etc. so that you can use them in your applications.
	New optional connection has been added to the ADC DelSig component	This can be used to connect the -ve input of modulator to AGL6.
	Charge pump power setting has been enabled depending on the clock frequency.	There was a problem with 8-bit ADC range. The problem was due to not setting charge pump power setting bits with respect to ADC clock in the DSM_CR16 register. ADC code was modified to set these bits depending on the ADC clock frequency.
	Removed the SetPower API.	The SetPower API was a non-functioning API. It was removed intentionally because it did not offer any value. If you had this function in your code, you need to remove it.