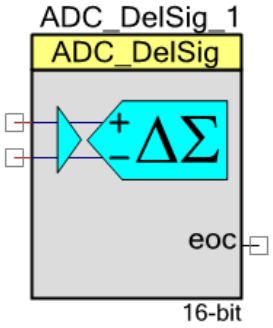
Delta Sigma Analog to Digital Converter (ADC_DelSig)

3.0



Features

- Selectable resolutions, 8 to 20 bits
- Eleven input ranges for each resolution
- Sample rate 8 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

General Description

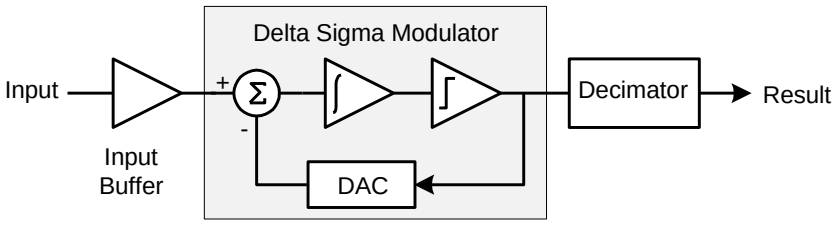
The Delta Sigma Analog to Digital Converter (ADC_DelSig) provides a low-power, low-noise front end for precision measurement applications. You can use it in a wide range of applications, depending on resolution, sample rate, and operating mode. It can produce 16-bit audio; high speed and low resolution 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) applications, and low-speed high-resolution (16 to 20 bits) applications. The sample rate can be adjusted between 10 and 384000 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.

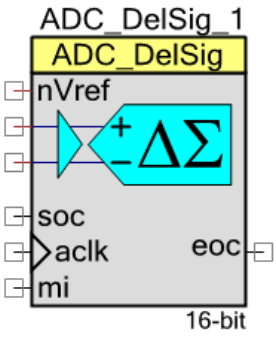
The ADC_DelSig is composed of three blocks: an input amplifier, a third-order delta-sigma modulator, and a decimator (see Figure 1). The input amplifier provides a high-impedance input and a user-selectable input gain. The decimator block contains a four-stage CIC decimation filter and a post-processing 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

Input 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** checkbox. This allows you to connect the ADC's reference Vssa to the analog global (AGL[6]). If the **Enable_Vref_Vssa** checkbox is not selected, this pin is not shown on the symbol. See the parameter Enable Vref/Vssa 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 is 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_StartConvert() function. This input is shown when the user selects the **Hardware SOC** parameter, which enables an external pin to start conversion. If **Hardware SOC** is not selected, the I/O pin on the component will be hidden. In Single Sample mode, a single conversion is executed and 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. Set this clock to the value displayed in the **Clock Frequency** parameter to achieve the selected sample rate. The duty cycle should be 50 percent. This clock determines the conversion rate as a function of conversion method and resolution. A clock that is external to the ADC should always be based on the system clock. If a more accurate and stable clock is required, the system clock should be based on an external (external to the PSoC) clock or oscillator.

eoc – Output

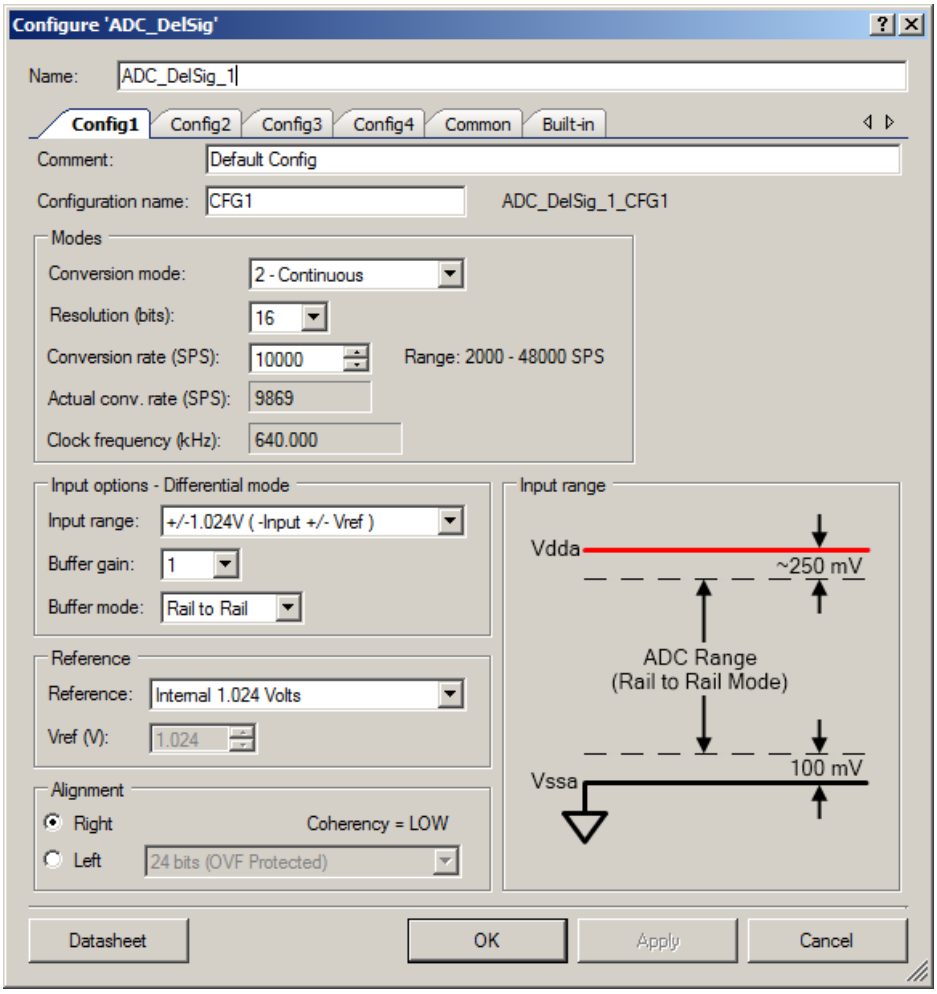
A rising edge on the End of Conversion (eoc) signals that a conversion is complete. The pin goes high for one ADC clock period. The eoc is typically connected to an interrupt or DMA request. The DMA request is typically used to transfer the conversion output to system RAM, DFB, or another component. There is already an interrupt, internal to the component, which is connected to this signal.

mi – Input *

Modulator Input is used to dynamically control the polarity of the signal in the modulator. Polarity is inverted when “mi” input is high. This input signal allows the user to modulate the input signal with an independent clock to act as a signal mixer.

Component Parameters

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.



Modes

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*). Single Sample conversion mode allows queuing of one conversion. If a sample is requested either by calling the StartConvert() function or by using the “soc” input before the current conversion is complete, another conversion will automatically be started after the current conversion is complete.
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** Take care 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 (ADC_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. Do not use this mode 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 resolutions of 8 to 16 bits. For resolutions of 17 to 20 bits, the performance of this mode is about four times faster than the Multi-Sample mode. **Note** Take care 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 (ADC_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.

All four ADC modes fully flush the decimator when the ADC initially starts conversions. This ensures that the first reading from the ADC is valid as long as the input voltage is stable before starting conversions with either the ADC_StartConvert() API or when triggered by the “soc” input. Although all modes reset the decimator when starting the ADC, only the continuous mode does not reset the decimator between readings. Because of this, the first reading in continuous mode takes four times longer than the subsequent readings. When using an analog mux to scan between multiple inputs, make sure that the ADC is not running while the input switches are changing. To switch input between samples when using modes other than continuous, use the analog hardware mux.

When changing the **Conversion mode** parameter, the clock frequency changes to maintain the selected sample rate. If the ADC clock frequency exceeds the minimum or maximum, an error message is displayed.

Note When operating in either the Multi-Sample, Continuous, or Multi-Sample (Turbo) modes, exercise caution to read the last ADC reading prior to the next ADC result completion. If reading the last result at the exact same time as the next result is complete, an incoherent reading may occur. When using DMA triggered by EOC or code in the ADC's ISR, this issue is easily avoided. Since you have a complete conversion time to read the last result, this should not be a problem in most designs. If the firmware cannot keep up with the ADC's conversion rate, the rate should be reduced to a manageable speed. This condition only occurs in the PSoC 3 family of products.

Resolution

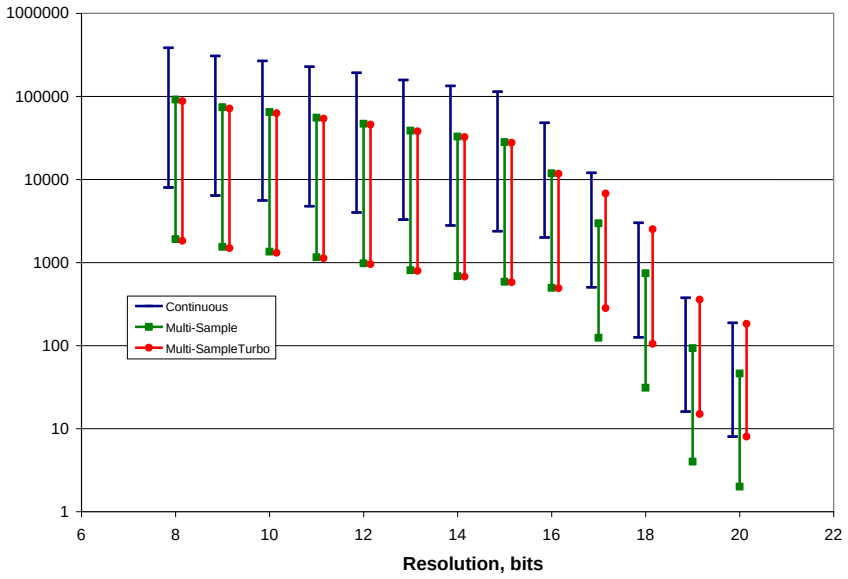
The resolution of the ADC_DelSig is entered as an integer value, limited to 8 to 20 bits. Higher resolution results in lower sample rate. Default resolution is 16 bits. When you change the **Resolution** parameter, the clock frequency changes to maintain the selected sample rate. If the ADC clock frequency exceeds the minimum or maximum, conversion rate also changes to fit into range.

Delta-sigma ADCs have inherent instability, resulting in nonlinearity at the positive and negative limits of the operating range. To correct for this, the input has been attenuated by 10 percent 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 percent. For example, if you select the input range ±1.024 V, the actual range of the ADC is approximately ±1.126 V. The usable input range remains ±1.024 V, but the ADC will not saturate until the input exceeds ±1.126 V.

The digital output from the ADC also over-ranges by 10 percent. 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 percent 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, make sure that the numerical value does not wrap around from its most positive or negative value to a negative or positive value, respectively. To make sure that 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, use the ADC_GetResult32() function instead of ADC_GetResult16(). The proper 16-bit value is 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, use the ADC_GetResult16() function. The proper 8-bit value is 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 Table 1.

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)

Resolution	Single Sample		Multi-Sample		Continuous		Multi-Sample Turbo	
	Min	Max	Min	Max	Min	Max	Min	Max
8	1561	74926	1911	91701	8000	384000	1581	75851
9	1293	62060	1543	74042	6400	307200	1307	62693
10	1143	54857	1348	64673	5566	267130	1154	55351
11	993	47627	1154	55351	4741	227555	1000	48000
12	854	40960	978	46900	4000	192000	860	41234
13	716	34324	806	38641	3283	157538	720	34516
14	616	29538	685	32855	2783	133565	619	29681

Resolution	Single Sample		Multi-Sample		Continuous		Multi-Sample Turbo	
	Min	Max	Min	Max	Min	Max	Min	Max
15	534	25600	585	28054	2371	113777	536	25707
16	458	10971	495	11861	2000	48000	459	11010
17	271	6494	124	2965	500	12000	272	6508
18	104	2475	31	741	125	3000	104	2477
19	15	357	4	93	16	375	15	357
20	8	182	2	46	8	187	8	182

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 also affect the maximum sample rate.

When changing the **Conversion rate** parameter, the clock frequency changes to maintain the selected sample rate. If the ADC clock frequency exceeds the minimum or maximum, an error indication displays next to the parameter. See Invalid settings.

Range

This field is a read-only (always unavailable) area that displays the minimum and maximum available conversion rate for the current settings.

Actual conversion rate

This read-only field displays an actual recalculated conversion rate based on the Master Clock frequency taken from the Design-Wide Resources (DWR) Clock Editor and calculated integer divider. The **Actual conversion rate** may differ from **Conversion rate** because of an integer divider of the source clock.

Clock frequency

This text box is a read-only (always unavailable) area that displays 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 change. The clock frequency is displayed with a resolution of 1 Hz.

The rate is displayed in the Design-Wide Resources Clock Editor, which always shows the clock frequency for Config 1. The ADC API sets the current clock frequency based on the configuration selected during run time when the **Clock source** parameter is set to **Internal**.

When you use a clock external to the ADC (either external to the chip or supplied from a user-selected internal clock), its value is displayed in this field. Conversion rate field is read-only in this mode and is calculated based on the clock frequency.

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 appears near **Conversion rate** field.

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

Input options

Input range

This parameter configures the ADC for a given input range. This configures the input to the ADC and is independent of the **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 is always dictated by the absolute maximum and minimum of the **Buffer mode**.

The options available for this parameter vary depending on the **Input Mode** selection; Table 2 and Table 3 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 **Rail to Rail**, **Level Shift**, or **Bypass Buffer** using the **Buffer mode** parameter. See the Buffer parameter description for more details.

You can use an external reference to provide a different operating range. The usable input range can be calculated with the applicable 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 is –Input ± 1.024 V. If the negative input is connected to 2.048 V 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 is –Input ± 2.048 V. If the negative input is connected to 2.028 V 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 is –Input ± 6.144 V, but not exceeding maximum electrical input range. You can use this mode to measure the supply voltages when connecting the negative input to Vssa. If you intend to measure the supply you must bypass the buffer.

Input Range Internal Ref (External Ref)	Description
±0.512 V (–Input ± Vref/2)	When using the internal reference (1.024 V), the input range is –Input ± 0.512 V. If –Input is connected to 1.0 V 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 is –Input ± 0.256 V. If –Input is connected to 1.0 V the usable input range is 1.0 ± 0.256 V or 0.744 to 1.256 V.
±0.128 V (–Input ± Vref/8)	When using the internal reference (1.024 V), the input range is –Input ± 0.128 V. If –Input is connected to 1.0 V the usable input range is 1.0 ± 0.128 V or 0.872 to 1.128 V.
±0.064 V (–Input ± Vref/16)	When using the internal reference (1.024 V), the input range is –Input ± 0.064V. If –Input is connected to 1.0 V the usable input range is 1.0 ± 0.064 V or 0.936 to 1.064 V.

The options listed in the previous table depend on the Vref setting. The following table shows examples with the Vref equal to 1.024 V and 1.200 V. The leading numeric value depends on the **Vref** setting in the **Reference** section.

With Vref = 1.024 V	With Vref = 1.2 V
±1.024 V (–Input ± Vref)	±1.200 V (–Input ±Vref)
±2.048 V (–Input ± Vref*2)	± 2.400 V (–Input ± Vref*2)
±6.144 V (–Input ± 6*Vref)	±7.200 V (–Input ± 6*Vref)
±0.512 V (–Input ± Vref/2)	±0.600 V (–Input ± Vref/2)
±0.256 V (–Input ± Vref/4)	±0.300 V (–Input ± Vref/4)
±0.128 V (–Input ± Vref/8)	±0.150 V (–Input ± Vref/8)
±0.064 V (–Input ± Vref/16)	±0.075 V (–Input ±Vref/16)

The following options are available when **Input Mode** is set to **Single**. To simulate single-ended operation, connect the negative input to an internal reference value (Vssa or Vref).

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

You can use an external reference 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 V.
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 V. 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. This is because the reference voltage in this case is connected to the negative input directly. The gain of the input buffer will only amplify the positive input and not the negative, causing an unbalanced amplification.
Vssa to Vdda	This mode is ratiometric of the supply voltage. The input range is Vssa to Vdda. Do not use an external reference 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 is 0.0 to 6.144 V, but not exceeding maximum electrical input range. You can use this mode to measure the supply voltage. If you intend to measure the supply you must bypass the buffer.

The following table shows the example with Vref = 1.024 V and 1.2 V. The leading numeric value depends on the **Vref** setting in the **Reference** section.

With Vref = 1.024 V	With Vref = 1.2 V
Vssa to 1.024 V (0.0 to Vref)	Vssa to 1.200 V (0.0 to Vref)
Vssa to 2.048 V (0.0 to Vref*2)	Vssa to 2.400 V (0.0 to Vref*2)
Vssa to Vdda	Vssa to Vdda
Vssa to 6.144 V (0.0 to 6*Vref)	Vssa to 7.200 V (0.0 to 6*Vref)

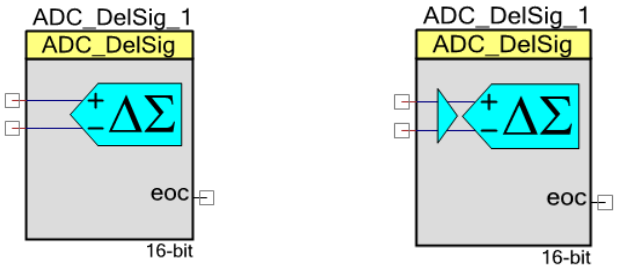
Note When selecting the Vssa to Vdda option, the customizer automatically selects either the Vdda/4 or Vdda/3 reference. This selection is based on the value of Vdda entered in the design-wide resources.

Buffer gain

This parameter 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. Make sure the **Buffer gain** and **Input range** settings 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.

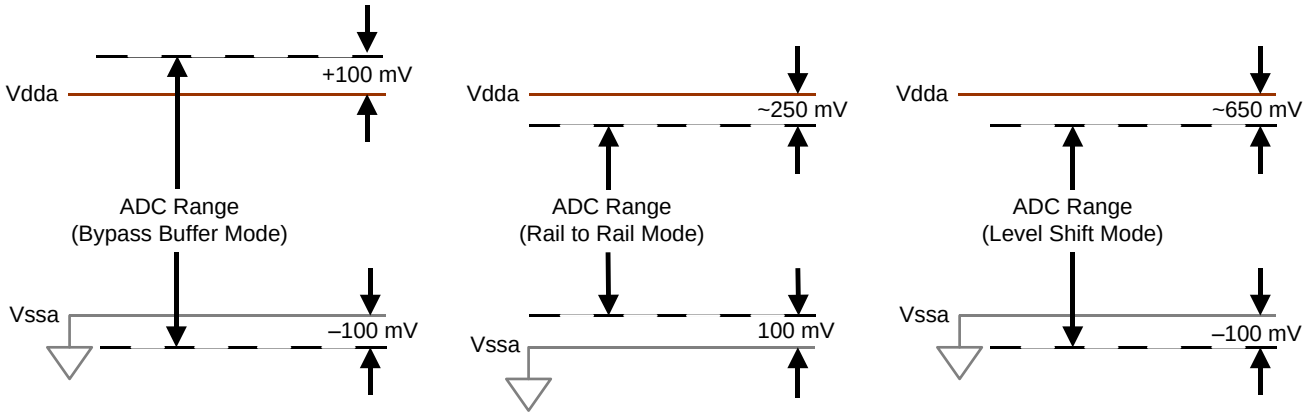
Buffer mode

This parameter 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 is disabled to reduce the overall power consumption. The **Buffer Gain** parameter does not have any effect if this mode is selected. If this mode is selected then input impedance is reduced to less than 500 kΩ. See Figure 3 for specifics on range.
Rail to Rail	Sets the input buffer mode to rail to rail. See Figure 3 for specifics on range.
Level Shift	Sets the input buffer mode to Level Shift. Both positive and negative input buffers are used. Level Shift mode allows you to go below the V_{SSA} but not all the way to V_{DDA}. See Figure 3 for specifics on range.

Figure 3 shows the ADC range for all Buffer Modes.

Figure 3. ADC Range



Reference

Vref

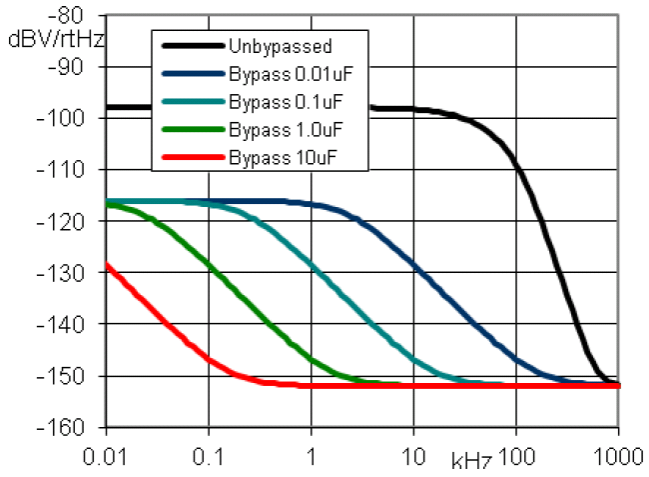
This parameter selects the ADC_DelSig reference voltage and configuration. The reference voltage sets the range of the ADC.

ADC_Reference	Description
Internal Vref 1.024 Volts	Use the internal 1.024-V reference (default)
Internal Vref, Bypassed on P0[3] [1]	Use the internal 1.024-V reference and allow an external bypass capacitor to be connected on pin P0[3].
Internal Vref, Bypassed on P3[2] [1]	Use the internal 1.024-V reference and allow an external bypass capacitor to be connected on pin P3[2].
External Vref on P0[3]	Use an external reference on pin P0[3]. See Delta-Sigma ADC DC Specifications for the allowable range.
External Vref on P3[2]	Use an external reference on pin P3[2].
Internal Vdda/4	Use the internal Vdda/4 reference.
Internal Vdda/4, Bypassed on P0[3] [1]	Use the internal Vdda/4 reference and allow an external bypass capacitor to be connected on pin P0[3].
Internal Vdda/4, Bypassed on P3[2] [1]	Use the internal Vdda/4 reference and allow an external bypass capacitor to be connected on pin P3[2].

[1] 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. The reference noise is frequency dependent, as shown in the measurements of Figure 4.

ADC_Reference	Description
Internal Vdda/3	Use the internal Vdda/3 reference.
Internal Vdda/3, Bypassed on P0[3] [1]	Use the internal Vdda/3 reference and allow an external bypass capacitor to be connected on pin P0[3].
Internal Vdda/3, Bypassed on P3[2] [1]	Use the internal Vdda/3 reference and allow an external bypass capacitor to be connected on pin P3[2].

Figure 4. Reference Noise



When the integrated reference noise is less than the integrated quantization noise over the band up to $f_{SAMPLE}/2$, the reference does not limit noise performance. You can select a suitable larger capacitor value according to resolution using the following table.

Resolution (bits)	Bypass Capacitor Value (µF)
10	0.01
12	0.01
14	0.1 to 1.0
16	0.1 to 1.0
18	1.0 to 10.0

Note The same internal reference is used for ADC_SAR and for ADC_DelSig components. If both types of the ADC have to work with internal reference simultaneously, use the Internal Vref, bypassed option for the best performance.

Vref value

This parameter sets or displays the reference voltage used by the ADC. If the **Internal 1.024 Volts** reference is used, the value 1.0240 is displayed. If the **Internal Vdda/3** or **Internal Vdda/4** reference option is selected, the value is derived from the Vdda setting in the design-wide resource page. If an external reference is selected, you can enter the reference used to ensure the counts-to-volts APIs work properly. The minimum and maximum values that can be entered are 0.9 to 1.3V, respectively. The default is 1.024V. When the selected reference is outside the range of 0.9 to 1.3V, an error symbol at the end of the Vref value numerical drop down menu alerts you that an error condition has occurred. Also, compilation fails if the Vref value exceeds the expected range of 0.9 to 1.3V.

The table below shows the Vref value displayed when different reference options are selected.

Reference Selected	Vref Value Displayed	Comment
Internal 1.024 Volts	1.0240	Value not editable.
Internal Bypassed on P0.3	1.0240	Value not editable.
Internal Bypassed on P3.2	1.0240	Value not editable.
External Vref on P0.3	1.0240	Value is editable with a default value of 1.0240.
External Vref on P3.2	1.0240	Value is editable with a default value of 1.0240.
Internal Vdda/4	1.2500*	This value is derived from the Vdda setting in the design-wide resource. Value is not editable. *The example shown is for a Vdda of 5 V.
Internal Vdda/4 Bypassed on P0.3	1.2500*	This value is derived from the Vdda setting in the design-wide resource. Value is not editable. *The example shown is for a Vdda of 5 V.
Internal Vdda/4 Bypassed on P3.2	1.2500*	This value is derived from the Vdda setting in the design-wide resource. Value is not editable. *The example shown is for a Vdda of 5 V.
Internal Vdda/3	1.100**	This value is derived from the Vdda setting in the design-wide resource. Value is not editable. **The example shown is for a Vdda of 3.3 V
Internal Vdda/3 Bypassed on P0.3	1.100**	This value is derived from the Vdda setting in the design-wide resource. Value is not editable. **The example shown is for a Vdda of 3.3 V.
Internal Vdda/3 Bypassed on P3.2	1.100**	This value is derived from the Vdda setting in the design-wide resource. Value is not editable. **The example shown is for a Vdda of 3.3 V.

Coherency

The output sample value is three bytes wide. It is protected on reads so that the underlying hardware doesn't update it when partially read by the system software or DMA. Depending on the configuration of the block, not all bits of the output sample register are of interest. The coherency methodology allows for any size of the output field and handles it properly.

By default the coherency is set to LSB byte for Right alignment mode (LOW) and could be changed to medium (MID) or MSB byte (HIGH) when Left alignment mode selected.

Alignment

This parameter determines how the result is aligned in the 24 bit result word. Normally all results are right justified, but in cases where the ADC result will be transferred to the DFB, it makes more sense to have the data left justified.

The following tables show all the details.

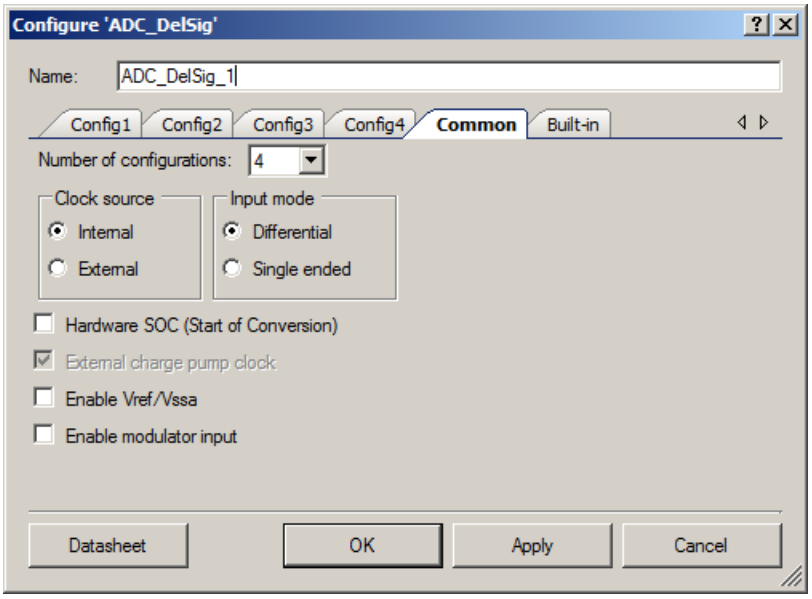
Resolution	Alignment Option	Coherency	Comment
8-20	Right	LOW	The full range results will be right aligned to bit 0.

Resolution	Left Alignment Options	Coherency	Comment
8 to 15	16 bits (OVF Protected)	MID	The full range results will be aligned to bit 14.
	16 bits (Not OVF Protected)	MID	The full range results will be aligned to bit 15. Care should be taken that most negative values do not exceed full range limit.
16	24 bits (OVF Protected)	HIGH	The result will be 24 bits and the full range value will be aligned at bit 22.
	16 bits (Not OVF Protected)	MID	The full range results will be aligned to bit 15. Care should be taken that most negative values do not exceed full range limit.
17-20	Bit-23 (OVF Protected)	HIGH	24-bit values aligned with bit 22 but sign extended to 32 bits.

Alignment	Res.	Result register bits for differential mode ("0" – LSB, "-" – null)																							
		23	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Right	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	7	6	5	4	3	2	1	0
	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	8	7	6	5	4	3	2	1	0
	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	9	8	7	6	5	4	3	2	1	0
	11	11	11	11	11	11	11	11	11	11	11	11	11	11	10	9	8	7	6	5	4	3	2	1	0
	12	12	12	12	12	12	12	12	12	12	12	12	12	11	10	9	8	7	6	5	4	3	2	1	0
	13	13	13	13	13	13	13	13	13	13	13	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	14	14	14	14	14	14	14	14	14	14	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	15	15	15	15	15	15	15	15	15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	16	16	16	16	16	16	16	16	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	17	17	17	17	17	17	17	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Alignment	Res.	Result register bits for differential mode (“0” – LSB, “-“ – null)																							
		23	**22**	**21**	**20**	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	18	18	18	18	18	18	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	19	19	19	19	19	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	20	20	20	20	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Left	8	8	8	8	8	8	8	8	8	8	7	6	5	4	3	2	1	0	-	-	-	-	-	-	-
16 bits (OVF Protected)	9	9	9	9	9	9	9	9	9	9	8	7	6	5	4	3	2	1	0	-	-	-	-	-	-
	10	10	10	10	10	10	10	10	10	10	9	8	7	6	5	4	3	2	1	0	-	-	-	-	-
	11	11	11	11	11	11	11	11	11	11	10	9	8	7	6	5	4	3	2	1	0	-	-	-	-
	12	12	12	12	12	12	12	12	12	12	11	10	9	8	7	6	5	4	3	2	1	0	-	-	-
	13	13	13	13	13	13	13	13	13	13	12	11	10	9	8	7	6	5	4	3	2	1	0	-	-
	14	14	14	14	14	14	14	14	14	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	-
	15	15	15	15	15	15	15	15	15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Left	16	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	-	-	-	-	-	-	-
24 bits (OVF Protected)																									
Left	8	8	8	8	8	8	8	8	8	7	6	5	4	3	2	1	0	-	-	-	-	-	-	-	-
16 bits (Not OVF Protected)	9	9	9	9	9	9	9	9	9	8	7	6	5	4	3	2	1	0	-	-	-	-	-	-	-
	10	10	10	10	10	10	10	10	10	9	8	7	6	5	4	3	2	1	0	-	-	-	-	-	-
	11	11	11	11	11	11	11	11	11	10	9	8	7	6	5	4	3	2	1	0	-	-	-	-	-
	12	12	12	12	12	12	12	12	12	11	10	9	8	7	6	5	4	3	2	1	0	-	-	-	-
	13	13	13	13	13	13	13	13	13	12	11	10	9	8	7	6	5	4	3	2	1	0	-	-	-
	14	14	14	14	14	14	14	14	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	-	-
	15	15	15	15	15	15	15	15	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	-
	16	16	16	16	16	16	16	16	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Left	17	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	-	-	-	-	-	-
Bit-23 (OVF Protected)	18	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	-	-	-	-	-
	19	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	-	-	-	-
	20	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	-	-	-

Common Settings



Number of configurations

You may define up to four different configurations using the **Number of configurations** parameter. For example, the system may require switching between continuous mode 16 bits, 48 ksps for audio; single sample mode 20 bits, 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 is 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 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 **Vref** parameter also has 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 has a separate Interrupt Service Routine function. When the ADC_SelectConfiguration() function is called, the interrupt vector changes to the corresponding interrupt vector routine.

Clock source

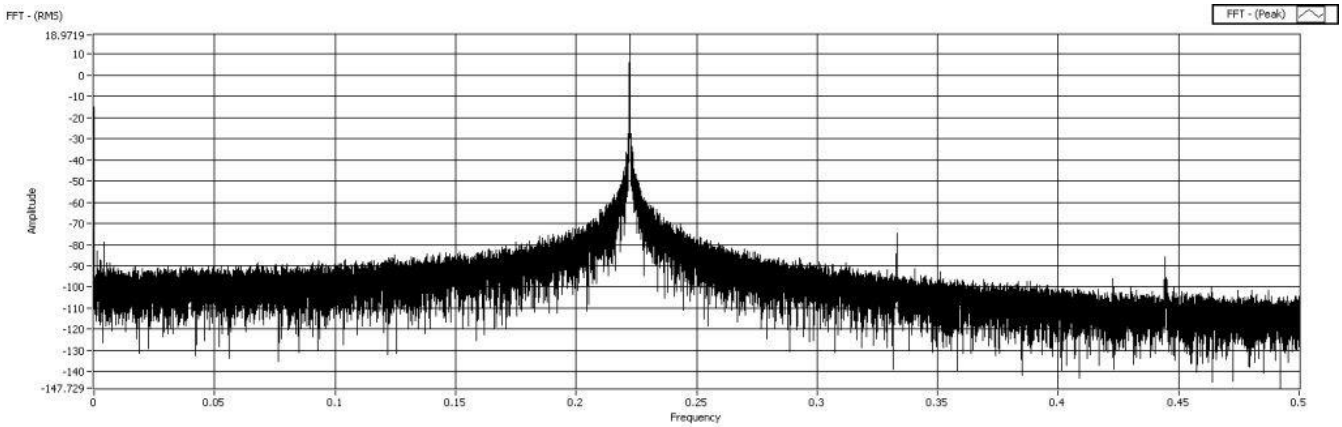
The ADC can be clocked by a source internal to the ADC component, a source external to the component but internal to the chip using a standard clock component or UDB, or by a source external to the chip. The internal or external clock selection is made using a radio button. When external clock is enabled,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

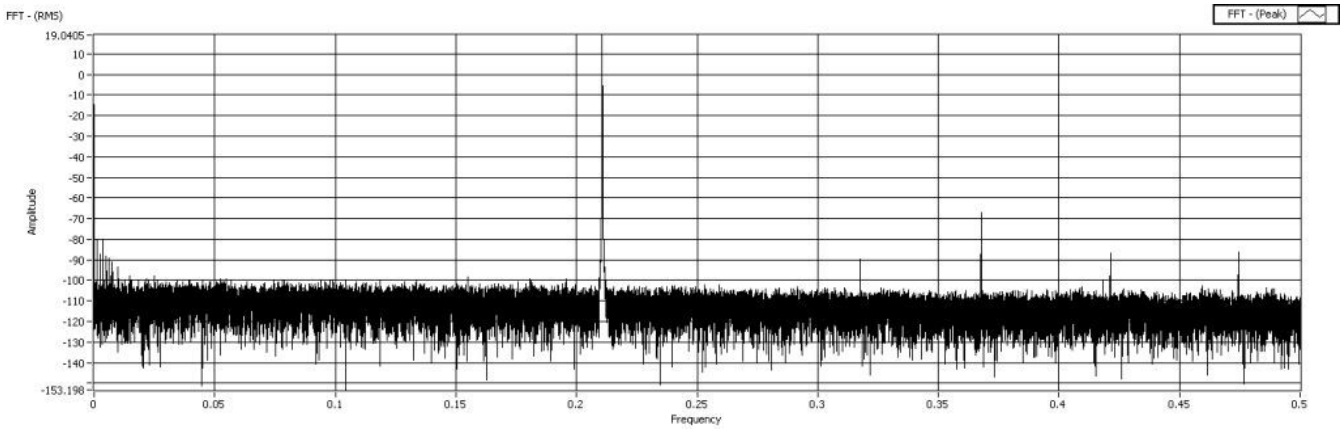
One measure of ADC performance is the distortion caused by INL, DNL, and timing errors. This distortion is measured by driving the ADC with a “perfect” sine wave then evaluating the output using an FFT (fast Fourier transform). Excess INL results in harmonic distortion with “lines” at multiples of the input sine wave frequency. Excess DNL results in an overall increase in the noise floor. Timing errors (frequency jitter) in the clock to the ADC result in spreading of the indicated fundamental from a nominal single line in the FFT. The PSoC internal clock has more frequency jitter than an external crystal oscillator. The upper plot shows the noise and spread in the FFT at the fundamental from the internal clock. The lower plot shows substantial reduction in the spreading of the fundamental as a result of the reduced frequency jitter from the external crystal oscillator.

Figure 5. Noise versus Clock

Single-Frequency FFT for Internal Oscillator



Single-Frequency FFT using External Crystal Oscillator

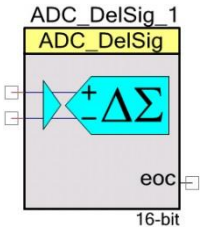


Input mode

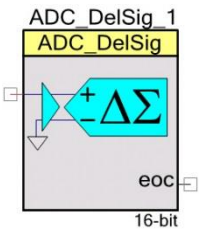
The ADC is inherently differential; however, you may use this parameter to simplify single-ended use.

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

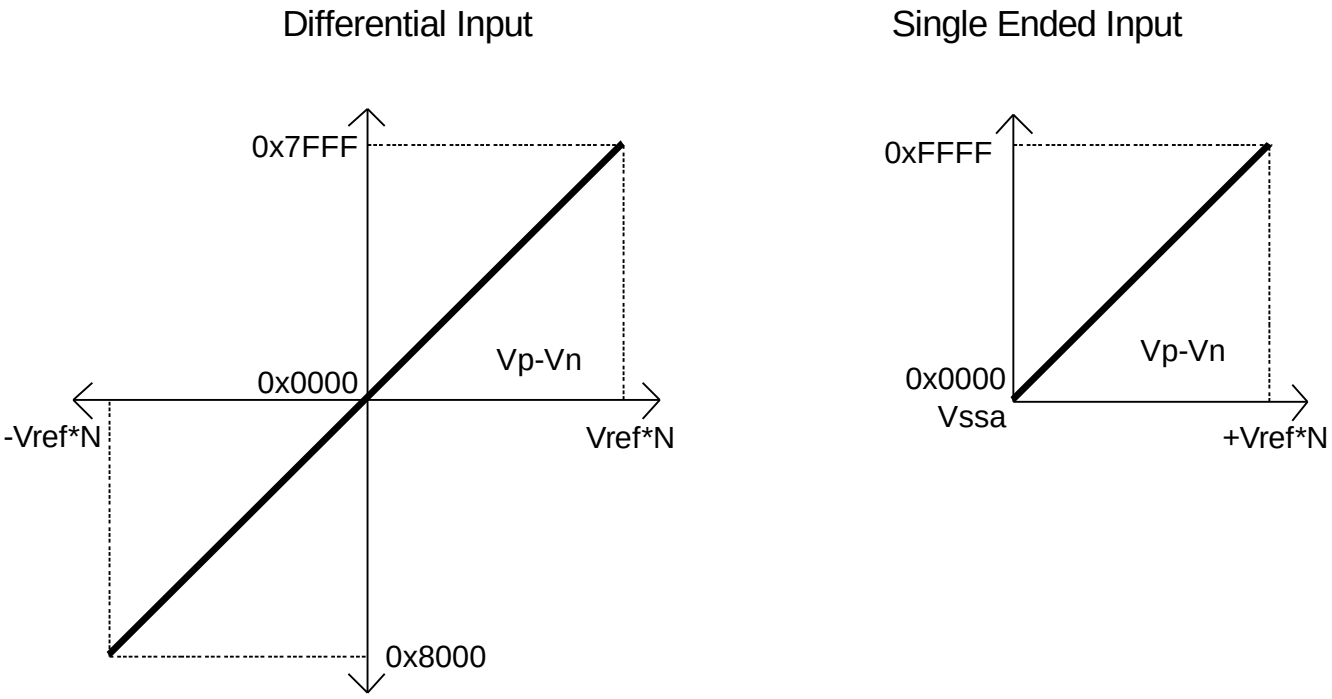


Note that that the Delta-Sigma ADC in PSoC 3 and PSoC 5LP devices inherently has differential input. Single-ended mode is implemented by internally connecting the negative input to ground.

Figure 6 shows the input-output characteristics of the delta-sigma ADC for differential and single ended input range for 16-bit resolution. The ADC is a differential ADC, although by connecting the Vn input to Vssa, it effectively makes the ADC look as if it were a single ended ADC, since all inputs will be referenced to Vssa. Even in the single ended mode, in some buffer modes the Vp input may be up to 100mV below Vssa which will cause the ADC to return a correct negative value. This is why ADC return value is always a signed number.

When ADC is in the differential mode, the full input range will always be symmetrical around zero. When in single ended mode, the input range will be from just below Vssa to the full scale value.

Figure 6. Delta-Sigma ADC I/O Characteristics



Note that when you configure the ADC for 16-bit single-ended mode, the output doesn't stop at 0x7FFF, but goes all the way up to 0xFFFF. As an example, a 16-bit single-ended ADC is implemented by considering a 17-bit differential ADC and ignoring half of the range (although the internal implementation is slightly different from that).

If your system requires both single-ended and differential modes, you must select differential mode 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.

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; otherwise, 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.

External charge pump clock

The External charge pump clock option is always enabled to guarantee the optimal charge pump clock frequency. This maximizes ADC performance. Charge pump clock frequency is automatically adjusted for optimal performance, but with the lowest power. This clock is adjusted to be 2 to 4 times that of the ADC clock frequency. This option is not available for user selection and is read-only.

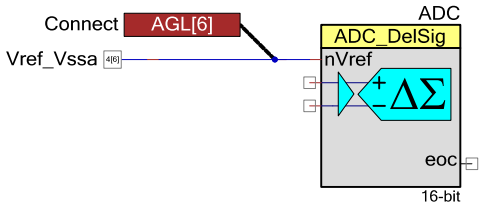
Enable Vref/Vssa

This parameter allows you to connect the negative input of the ADC's reference Vssa to the analog global AGL[6]. For high-precision 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 could 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 is directly connected to Analog Global Left 6 (AGL[6]). (See the analog routing diagram in the device datasheet for more information.) AGL[6] makes direct connections to pins P4[6], P4[2], P0[6] and P0[2]. For the best performance, make sure that Vref_Vssa is connected to one of these pins. Placing Vref_Vssa on another pin causes extra routing resources to be consumed and extra resistance to be added in series with the connection.

The manual analog routing system (MARS) components allow you to a add rule check that ensures that 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 generates an error during the build process. The error indicates that you have connected a resource to that net that has no direct connection to the pin and therefore cannot route. The following graphic is an example:



Enable modulator input

When this parameter is enabled, the modulator input terminal will be enabled on the symbol. The designer may route a signal from the DSI to drive the modulator input “mi” signal. This is an advanced user option.

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
ADC_Start()	Sets the initVar variable, calls the ADC_Init() function, and then calls the ADC_Enable() function.
ADC_Stop()	Stops ADC conversions and powers down
ADC_SetBufferGain()	Selects input buffer gain (1,2,4,8)
ADC_StartConvert()	Starts conversion
ADC_StopConvert()	Stops conversions
ADC_IRQ_Enable()	Enables interrupts at the end of conversion
ADC_IRQ_Disable()	Disables interrupts
ADC_IsEndConversion()	Returns a nonzero value if conversion is complete
ADC_GetResult8()	Returns an 8-bit conversion result
ADC_GetResult16()	Returns a 16-bit conversion result
ADC_GetResult32()	Returns a 32-bit conversion result
ADC_Read8()	Starts ADC conversions, waits for the conversion to be complete, stops ADC conversion and returns the signed 8-bit value of result.
ADC_Read16()	Starts ADC conversions, waits for the conversion to be complete, stops ADC conversion and returns the signed 16-bit value of result.
ADC_Read32()	Starts ADC conversions, waits for the conversion to be complete, stops ADC conversion and returns the signed 32-bit value of result.
ADC_SetOffset()	Sets the offset used by the ADC_CountsTo_mVolts(), ADC_CountsTo_uVolts(), and ADC_CountsTo_Volts() functions.
ADC_SelectConfiguration()	Sets one of up to four ADC configurations
ADC_SetGain()	Sets the gain used by the ADC_CountsTo_mVolts(), ADC_CountsTo_uVolts(), and ADC_CountsTo_Volts() functions.
ADC_CountsTo_mVolts()	Converts ADC counts to millivolts
ADC_CountsTo_uVolts()	Converts ADC counts to microvolts
ADC_CountsTo_Volts()	Converts ADC counts to floating point volts

Function	Description
ADC_Sleep()	Stops ADC operation and saves the user configuration
ADC_Wakeup()	Restores and enables the user configuration
ADC_Init()	Initializes or restores the ADC using the Configure dialog settings
ADC_Enable()	Enables the ADC
ADC_SaveConfig()	Saves the current configuration
ADC_RestoreConfig()	Restores the configuration
ADC_SetCoherency()	Sets the coherency register
ADC_SetGCOR()	Calculates a new GCOR value and sets the GCOR registers with this new value
ADC_ReadGCOR()	Returns the normalized GCOR register values

Global Variables

Variable	Description
ADC_initVar	The ADC_initVar variable 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 to calibrate offset. Initially, this variable is set to zero. Applications can modify it using the ADC_SetOffset() function. It affects only the ADC_CountsTo_Volts(), ADC_CountsTo_mVolts(), and ADC_CountsTo_uVolts() functions by subtracting the given offset.
ADC_CountsPerVolt	The ADC_countsPerVolt variable is used to calibrate gain. Initially, this variable is calculated for the default ADC configuration. The calculated value depends on resolution, input range, and voltage reference. Applications can modify it using the ADC_SetGain() function. It affects only the ADC_CountsTo_Volts(), ADC_CountsTo_mVolts(), and ADC_CountsTo_uVolts() functions by supplying the correct conversion between ADC counts and the applied input voltage.
ADC_convDone	The ADC_convDone variable is used as the software flag for checking the ADC conversion in single sample conversion mode for resolutions above 16 bits.

void ADC_Start(void)

Description: Sets the initVar variable, calls the ADC_Init() function, and then calls the ADC_Enable() function.

This function configures and powers up the ADC, but does not start conversions. By default, the ADC is configured for Config1 unless the ADC_SelectConfiguration() 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. The following table shows 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: Increasing the gain will lower the buffer bandwidth.

void ADC_StartConvert(void)

Description: Forces the ADC to initiate a conversion.

In Single Sample mode, call this API to start a single conversion. When the conversion completes, use ADC_IsEndConversion() API to check or wait on this event, the ADC will halts. If the ADC_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 ADC_StopConvert(). After stopping the conversion, restart the conversion by calling ADC_StartConvert().

In Multi Sample, Continuous or Multi Sample (Turbo) modes, call this API to start continuous ADC conversions until either the ADC_StopConvert() or ADC_Stop() functions are executed.

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 a 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 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*, before interrupts occur.

Parameters: None

Return Value: None

Side Effects: Enables interrupts. Reading the result clears 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: Checks 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 the following table 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 nonzero value is returned, the last conversion has completed. If the returned value is zero, the ADC is still calculating the last result.

Side Effects: When the EOC output is used to trigger the DMA to read the ADC result, the ADC_IsEndConversion() function should not be used. This is because, reading the output register with DMA clears the ADC's conversion complete flag and this API may not return when in the ADC_WAIT_FOR_RESULT mode.

int8 ADC_GetResult8(void)

Description: Returns a signed 8-bit value. The largest positive signed 8-bit value that can be represented is 127, but in single-ended 8-bit mode, the maximum positive value is 255. Hence, for 8-bit single-ended mode, use the ADC_GetResult16() function instead. Note that if the ADC resolution is set greater than 8 bits, the LSB of the result is returned.

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 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, use the ADC_GetResult32() function instead. This function returns only signed 16-bit results, 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 that has a resolution of 8 to 20 bits.

Parameters: None

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

Side Effects: None

int8 ADC_Read8(void)

Description: This function simplifies getting results from the ADC when only a single reading is required. When called, it will start ADC conversions, wait for the conversion to be complete, stop ADC conversion and return the result. This is a blocking function and will not return until the result is ready.

When the ADC is configured for 8-bit single ended mode, the ADC_Read16() 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 an ADC conversion.

Side Effects: None

int16 ADC_Read16(void)

Description: This function simplifies getting results from the ADC when only a single reading is required. When called, it will start ADC conversions, wait for the conversion to be complete, stop ADC conversion and return the result. This is a blocking function and will not return until the result is ready.

When the ADC is configured for 16-bit single ended mode, the ADC_Read32() function should be used instead. This function returns only signed 16-bit values, which allows a maximum positive value of 32767, not 65535.

Parameters: None

Return Value: int16: The 16-bit result of an ADC conversion.

Side Effects: None

int32 ADC_Read32(void)

Description: This function simplifies getting results from the ADC when only a single reading is required. When called, it will start ADC conversions, wait for the conversion to be complete, stop ADC conversion and return the result. This is a blocking function and will not return until the result is ready. 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 an 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. This API first stops the ADC and then initializes the registers with the default values for the new configuration. The custom GGOR register value, set by ADC_SetGCOR() API for particular configuration, is not overwritten to default value. If the value of the second parameter restart is 1, then ADC will be restarted. If this value is zero, then you 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: Internal interrupt is configured to the new vector, related with configuration number and is enabled. Refer to Interrupt Service Routine section for information how to remove internal interrupt.

int16 ADC_CountsTo_mVolts(int32 adcCounts)

Description: Converts the ADC output to millivolts as a 16-bit integer. For example, if the ADC measured 0.534 V, the return value would be 534 mV. The calculation of voltage depends on the value of the voltage reference. When the Vref is based on Vdda, the value used for Vdda is set for the project in the System tab of the Design Wide Resources (DWR).

Parameters: int32 adcCounts: Result from the ADC conversion.

Return Value: int16: Result in mV.

Side Effects: None

int32 ADC_CountsTo_uVolts(int32 adcCounts)

Description: Converts the ADC output to microvolts as a 32-bit integer. For example, if the ADC measured –0.02345 V, the return value would be –23450 µV. The calculation of voltage depends on the value of the voltage reference. When the Vref is based on Vdda, the value used for Vdda is set for the project in the System tab of the Design Wide Resources (DWR).

Parameters: int32 adcCounts: Result from the ADC conversion.

Return Value: int32: Result in µV.

Side Effects: None

float32 ADC_CountsTo_Volts(int32 adcCounts)

Description: Converts the ADC output to volts as a floating point number. For example, if the ADC measures a voltage of 1.2345 V, the returned result would be +1.2345 V. The calculation of voltage depends on the value of the voltage reference. When the Vref is based on Vdda, the value used for Vdda is set for the project in the System tab of the Design Wide Resources (DWR).

Parameters: int32 adcCounts: Result from the ADC conversion.

Return Value: float: Result in volts.

Side Effects: None

void ADC_Sleep(void)

Description: The ADC_Sleep() function checks to see if the component is enabled and saves that state. Then it calls the ADC_Stop() function and calls ADC_SaveConfig() to save the user configuration.

Call the ADC_Sleep() function before calling the CyPmSleep() or the CyPmHibernate() function. Refer to the *PSoC Creator System Reference* Guide for more information about power management functions.

Parameters: None

Return Value: None

Side Effects: None

void ADC_Wakeup(void)

Description: The ADC_Wakeup() function calls the ADC_RestoreConfig() function to restore the user configuration. If the component was enabled before the ADC_Sleep() function was called, the ADC_Wakeup() function will re-enable the component.

Parameters: None

Return Value: None

Side Effects: Calling the ADC_Wakeup() function without first calling the ADC_Sleep() or ADC_SaveConfig() function may produce unexpected behavior.

void ADC_Init(void)

Description: Initializes or restores the component parameters per the Configure dialog settings. 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 reinitializes the component.

void ADC_Enable(void)

Description: Enables the clock and power for ADC.

Parameters: None

Return Value: None

Side Effects: Enables internal interrupt. Refer to Interrupt Service Routine section for information how to remove internal interrupt.

void ADC_SaveConfig(void)

Description: This function saves the component configuration. This will save nonretention registers. This function will also save the current component parameter values, as defined in the Configure dialog or as modified by appropriate APIs. This function is called by the ADC_Sleep() function.

Parameters: None

Return Value: None

Side Effects: None.

void ADC_RestoreConfig(void)

Description: This function restores the component configuration. This will restore nonretention registers. This function will also restore the component parameter values to what they were prior to calling the ADC_Sleep() function.

Parameters: None

Return Value: None

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

void ADC_SetCoherency(uint8 coherency)

Description: This function allows you to change which of the ADC's three-word results will trigger a coherency unlock. The ADC's result will not be updated until the set byte is read by either the ADC or DMA. By default, the LSB is the coherency byte. If DMA or if a custom API is written where the LSB is not the last byte read, this use this API to set the last byte of the ADC result that is read. If a multi-byte read is performed either by DMA or the ARM processor, the coherency can be set to any byte in the last word read.

Parameters: uint8 coherency: Coherency settings. The following table shows the valid coherency values.

Coherency Options	Description
ADC_COHER_LOW	The LSB should be the last byte read.
ADC_COHER_MID	The middle byte should be the last byte read.
ADC_COHER_HIGH	The MSB should be the last byte read.

Return Value: None

uint8 ADC_SetGCOR(float gainAdjust)

Description: This function calculates a new GCOR (ADC Gain) value and writes it into the GCOR registers. The GCOR value is a 16-bit value that represents a gain of 0 to 2. The ADC result is multiplied by this value before it is placed in the ADC output registers.

When executing the function, the old GCOR value is multiplied by **gainAdjust** input and reloaded into the GCOR register. The GCOR value is normalized based on the GVAL register.

The value, calculated by this API, is also stored into the RAM for each active configuration and used by SelectConfiguration() API to initialize the GCOR register.

Parameters: float gainAdjust: Gain multiplier coefficient for GCOR correction. Resultant GCOR value should be in range of 0.000 to 1.9999.

Return Value: uint8: A nonzero value will be returned, if the correction value is outside the GCOR value range of 0.00 to 1.9999.

Side Effects: Actual GCOR value depends on the GVAL register. The GVAL register is set to a value that is equal to number of valid bits in the GCOR register minus one. If GVAL is 15 (0x0F), all 16 bits of the GCOR registers are valid. If GVAL is 11 (0x0B), only 12 bits are valid. The least four bits are lost when the GCOR value is shifted four places to the right. The GVAL register is automatically set by Init() and SelectConfiguration() APIs and doesn't need to be updated by user.

uint16 ADC_ReadGCOR(void)

Description: This function returns the current GCOR register value, normalized based on the GVAL setting. For example, if the GCOR value is 0x0812 and the GVAL register is set to 11 (0x0B), the returned value is shifted by four bits to the left. (Actual GCOR value = 0x0812, returned value = 0x8120)

Parameters: None

Return Value: uint16: Normalized GCOR value

Side Effects: The GCOR value depends on the GVAL register. If GVAL is set to less than 15 (0x0F), the GCOR value is shifted right by 15 minus GVAL. GVAL register is automatically set by Init() and SelectConfiguration() APIs and doesn't need to be updated by user.

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 the EOC pin from the ADC. You can use the DMA Wizard to configure DMA operation as follows:

Name of DMA Source	Length	Direction	DMA Req Signal	DMA Req Type	Description
ADC_DelSig_DEC_SAMP_PTR	1	source	EOC	Rising Edge	Receives 1-byte conversion result for input analog value that has a resolution of 8 bits.
ADC_DelSig_DEC_SAMP_PTR	2	source	EOC	Rising Edge	Receives 2-byte conversion result for input analog value that has a resolution of 9 to 16 bits. Note that the coherency should be set to medium (MID) byte. Use ADC_SetCoherency() API to change coherency.
ADC_DelSig_DEC_SAMP_PTR	3	source	EOC	Rising Edge	Receives 3-byte conversion result for input analog value that has a resolution of 17 to 20 bits. Note that the coherency should be set to MSB (HIGH) byte. Use ADC_SetCoherency() API to change coherency.

MISRA Compliance

This section describes the MISRA-C:2004 compliance and deviations for the component. There are two types of deviations defined:

- project deviations – deviations that are applicable for all PSoC Creator components
- specific deviations – deviations that are applicable only for this component

This section provides information on component-specific deviations. Project deviations are described in the MISRA Compliance section of the *System Reference Guide* along with information on the MISRA compliance verification environment.

The ADC_DelSig component does not have any specific deviations.

This component has the following embedded components: Interrupt, Clock, AMux. Refer to the corresponding component datasheet for information on their MISRA compliance and specific deviations.

Interrupt Service Routine

The ADC_DelSig contains four interrupt service routines in the file *ADC_INT.c* file, where "ADC" is the instance name. The service routines are labeled *ADC_ISR1*, *ADC_ISR2, ADC_ISR3* and *ADC_ ISR4* and related with configuration number. ADC_SelectConfiguration() function changes the interrupt vector address to the corresponding interrupt routine.

You can place custom code in the designated areas to perform whatever function is required at the end of a conversion. Place custom code between the "`/* `#START MAIN_ADC_ISR1` */`" and "`/* `#END` */`" comments. This ensures that the code will be preserved when a project is regenerated.

There is advanced feature to remove the internal interrupt. It is useful when external interrupt component is connected to EOC output and internal ISR isn't required. Set **rm_int** parameter to true in Expression view of the Common tab(right click on Common tab menu of the component customizer) to suppress the internal interrupt. Make sure that Expression view is enabled in Creator menu: Tools/Option…/Design Entry/Component Catalog/**Enable Param Edit Views**.

Sample Firmware Source Code

PSoC Creator provides many example projects that include schematics and example code in the Find Example Project dialog. For component-specific examples, open the dialog from the Component Catalog or an instance of the component in a schematic. For general examples, open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

Refer to the "Find Example Project" topic in the PSoC Creator Help for more information.

Functional Description

The delta-sigma channel contains the following blocks:

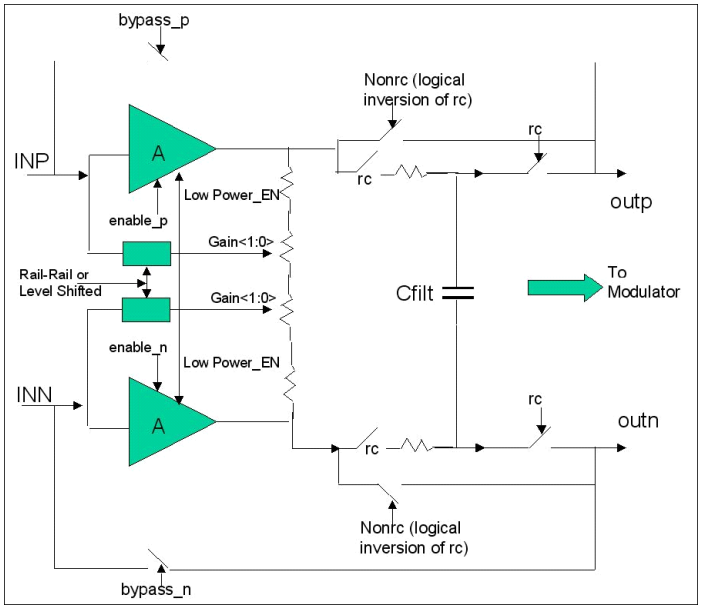
- A high-input-impedance, front-end buffer (with programmable gain) that can be bypassed (and powered down) when not needed.
- A fully differential programmable third-order switched capacitor modulator.
- A downstream digital-filtering consisting of: a fourth-order Cascaded Integrator-Comb (CIC) filter (also called the decimator) and the postprocessing engine (cicdec4_pproc), which 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.

Without an input buffer, a switched-capacitor input stage consumes current to charge the capacitor during each cycle. In that case, the equivalent input resistance is of the order of 1/(fs × C), or 1/((3 MHz) × (5 pF)) = 66 kΩ. 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 handle signals closer to ground in some applications and must work closer to the supply rail in others. Input buffer architecture contains two single-ended buffers, which create a differential channel. You can select either buffer 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. You can power down the buffers individually. There are two main modes of operation for the buffer:

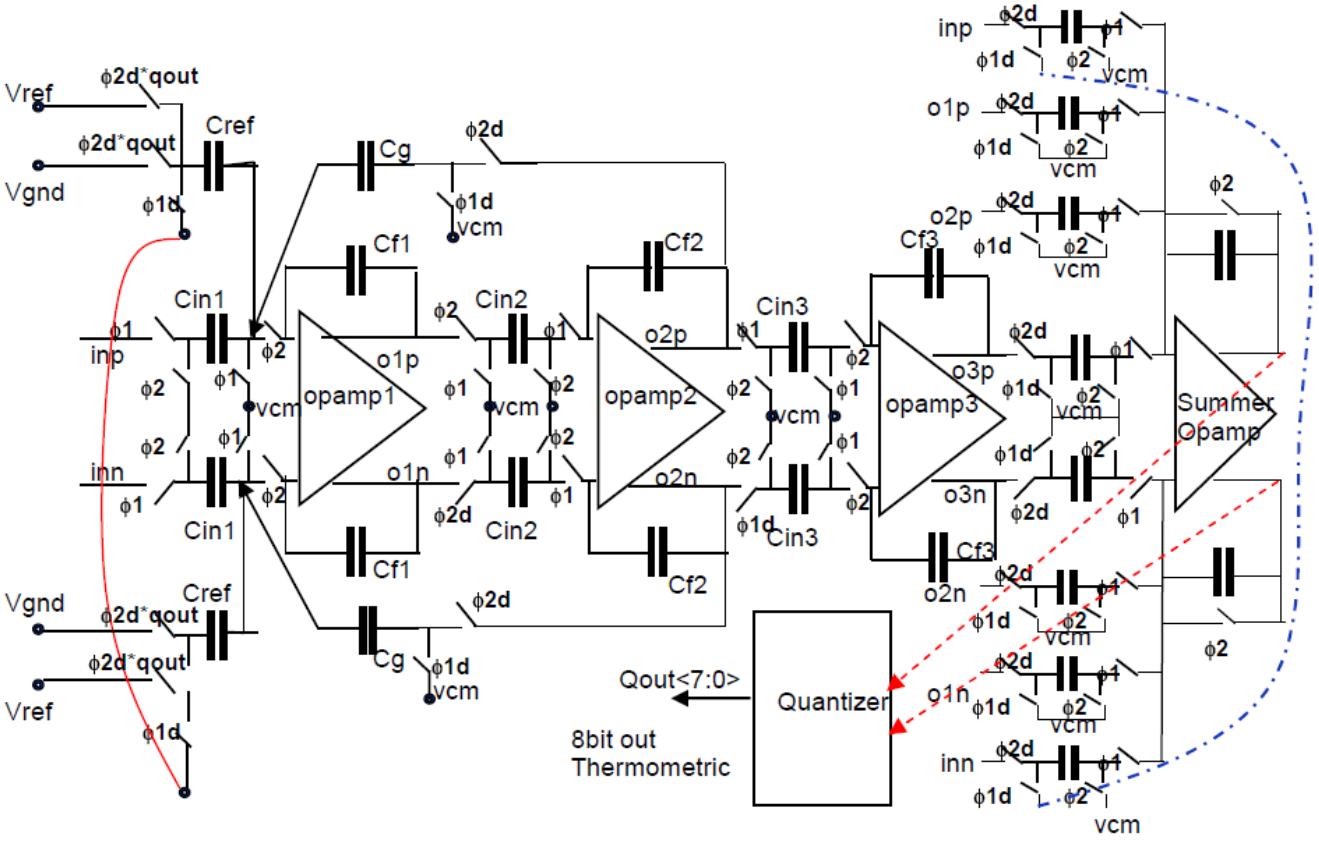
- Level-Shifted Mode: Buffer output can be level shifted up from the input when the input is close to 0-V input common mode voltage range.
- Rail to Rail Mode: This mode is used when input is rail to rail.

Figure 7. ADC Buffer Structure



The switched capacitor implementation is shown in Figure 8. A dynamic element matching (DEM) block shapes the errors caused by 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 thermometric format. The conversion of thermometer quantizer code into 2’s complement format, for use in the decimator (Sinc4 and Sinc1), is performed in the ANAIF block.

Figure 8. Switched Capacitor Delta Sigma Modulator Structure



Registers

Sample Registers

The ADC results can be between 8 and 20 bits of resolution. The output is divided into three 8-bit registers. The CPU or DMA can access these registers 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]							

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.

API Memory Usage

The component memory usage varies significantly, depending on the compiler, device, number of APIs used and component configuration. The following table provides the memory usage for all APIs available in the given component configuration.

The measurements have been done with the associated compiler configured in Release mode with optimization set for Size. For a specific design the map file generated by the compiler can be analyzed to determine the memory usage.

Configuration	PSoC 3 (Keil_PK51)		PSoC 5LP (GCC)	
	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes
8-20 bits	3824	27	2968	23

DC and AC Electrical Characteristics for PSoC 3

Specifications are valid for –40 °C ≤ T_A ≤ 85 °C and T_J ≤ 100 °C, except where noted. Specifications are valid for 1.71 V to 5.5 V, except where noted. Typical values are for T_A = 25°C

Operating conditions are:

- Operation in continuous sample mode
- fclk = 3.072 MHz for resolution = 16 to 20 bits; fclk = 6.144 MHz for resolution = 8 to 15 bits
- Reference = 1.024-V internal reference bypassed on P3[2] or P0[3]
- Unless otherwise specified, all charts and graphs show typical value

Delta-Sigma ADC DC Specifications for PSoC 3

Parameter	Description	Conditions	Min	Typ	Max	Units
	Resolution		8	–	20	bits
	Number of channels, single ended		–	–	No. of GPIO	–
	Number of channels, differential	Differential pair is formed using a pair of GPIOs.	–	–	No. of GPIO/2	–
	Monotonic	Yes	–	–	–	–
Ge	Gain error [2]	Buffered, buffer gain = 1, input range = ±1.024 V, 16-bit mode	–	–	±0.2	%
Gd	Gain drift	Buffered, buffer gain = 1, input range = ±1.024 V, 16-bit mode	–	–	50	ppm/°C
V_{OS}	Input offset voltage [3]	Buffered	–	–	±0.1	mV
TCV_{OS}	ADC TC input offset voltage	Temperature coefficient, input offset voltage	–	–	0.55	µV/°C
	Input voltage range, single ended [4]		V_{SSA}	–	V_{DDA}	V
	Input voltage range, differential unbuffered [5]		V_{SSA}	–	V_{DDA}	V
	Input voltage range, differential, buffered [5]		V_{SSA}	–	V_{DDA} – 1	V
PSRRb	Power supply rejection ratio, buffered [5]	Buffer gain = 1, 16-bit, Range = ±1.024 V	90	–	–	dB
CMRRb	Common mode rejection ratio, buffered [5]	Buffer gain = 1, 16-bit, Range = ±1.024 V	85	–	–	dB
INL20	Integral nonlinearity [5]	Range = ±1.024 V, unbuffered	–	–	±32	LSB
DNL20	Differential nonlinearity [5]	Range = ±1.024 V, unbuffered	–	–	±1	LSB
INL16	Integral non linearity [5]	Range = ±1.024 V, unbuffered	–	–	±2	LSB
DNL16	Differential nonlinearity [5]	Range = ±1.024 V, unbuffered	–	–	±1	LSB
INL12	Integral nonlinearity [5]	Range = ±1.024 V, unbuffered	–	–	±1	LSB

[2] Total gain error is sum of ADC error and buffer error.

[3] Total offset voltage error is sum of buffer V_{OS} and ADC V_{OS}.

[4] V_{SSA} to 6 × V_{BG} range is used for direct measurement of V_{DDA} power supply. Actual scale is limited to V_{DDA}.

[5] Based on device characterization (not production tested).

Parameter	Description	Conditions	Min	Typ	Max	Units
DNL12	Differential nonlinearity [5]	Range = ±1.024 V, unbuffered	–	–	±1	LSB
INL8	Integral nonlinearity [5]	Range = ±1.024 V, unbuffered	–	–	±1	LSB
DNL8	Differential nonlinearity [5]	Range = ±1.024 V, unbuffered	–	–	±1	LSB
Rin_Buff	ADC input resistance	Input buffer used	10	–	–	MΩ
Rin_ADC16	ADC input resistance	Input buffer bypassed, 16-bit, Range = ±1.024 V	–	74 [6]	–	kΩ
Rin_ADC12	ADC input resistance	Input buffer bypassed, 12 bit, Range = ±1.024 V	–	148 [6]	–	kΩ
V_{EXTREF}	ADC external reference input voltage	Pins P0[3], P3[2]	0.9	–	1.3	V
Current Consumption						
I_{DD_20}	Current consumption, 20-bit [5]	187 sps, unbuffered	–	–	1.5	mA
I_{DD_16}	Current consumption, 16-bit [5]	48 ksps, unbuffered	–	–	1.5	mA
I_{DD_12}	Current consumption, 12-bit [5]	192 ksps, unbuffered	–	–	1.95	mA
I_{BUFF}	Buffer current consumption [5]		–	–	2.5	mA

Delta-Sigma ADC AC Specifications for PSoC 3

Parameter	Description	Conditions	Min	Typ	Max	Units
	Startup time		–	–	4	Samples
	ADC_SelectConfiguration() API execution time	CPU clock = 24 Mhz	–	90	-	µs
THD	Total harmonic distortion [7]	Buffer gain = 1, 16 bit, Range = ±1.024 V	–	–	0.0032	%
20-Bit Resolution Mode						
SR20	Sample rate [7]	Range = ±1.024 V, unbuffered	7.8	–	187	sps
BW20	Input bandwidth at max sample rate [7]	Range = ±1.024 V, unbuffered	–	40	–	Hz

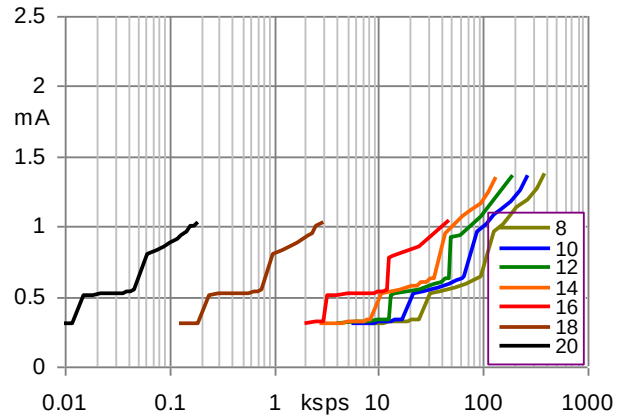
[6] Using switched capacitors at the ADC input creates an effective input resistance. Holding the gain and number of bits constant, the resistance is proportional to the inverse of the clock frequency. This value is calculated, not measured. For more information see the Technical Reference Manual.

[7] Based on device characterization (not production tested).

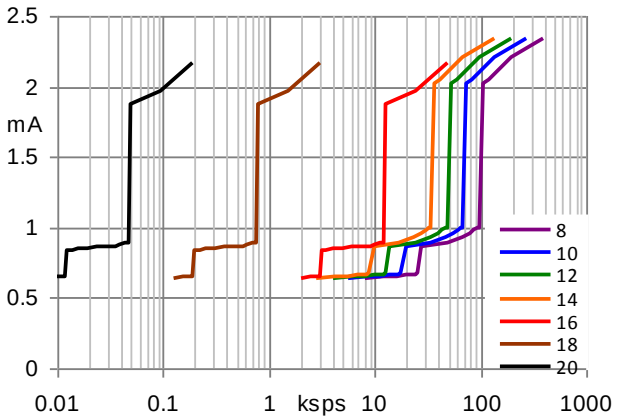
Parameter	Description	Conditions	Min	Typ	Max	Units
16-Bit Resolution Mode						
SR16	Sample rate [7]	Range = ±1.024 V, unbuffered	2	–	48	ksps
BW16	Input bandwidth at max sample rate [7]	Range = ±1.024 V, unbuffered	–	11	–	kHz
SINAD16int	Signal-to-noise ratio, 16-bit, internal reference [7]	Range = ±1.024V, unbuffered	81	–	–	dB
SINAD16ext	Signal-to-noise ratio, 16-bit, external reference [7]	Range = ±1.024 V, unbuffered	84	–	–	dB
12-Bit Resolution Mode						
SR12	Sample rate, continuous, high power [7]	Range = ±1.024 V, unbuffered	4	–	192	ksps
BW12	Input bandwidth at max sample rate [7]	Range = ±1.024 V, unbuffered	–	44	–	kHz
SINAD12int	Signal-to-noise ratio, 12-bit, internal reference [7]	Range = ±1.024 V, unbuffered	66	–	–	dB
8-Bit Resolution Mode						
SR8	Sample rate, continuous, high power [7]	Range = ±1.024 V, unbuffered	8	–	384	ksps
BW8	Input bandwidth at max sample rate [7]	Range = ±1.024 V, unbuffered	–	88	–	kHz
SINAD8int	Signal-to-noise ratio, 8-bit, internal reference [7]	Range = ±1.024 V, unbuffered	43	–	–	dB

Figures

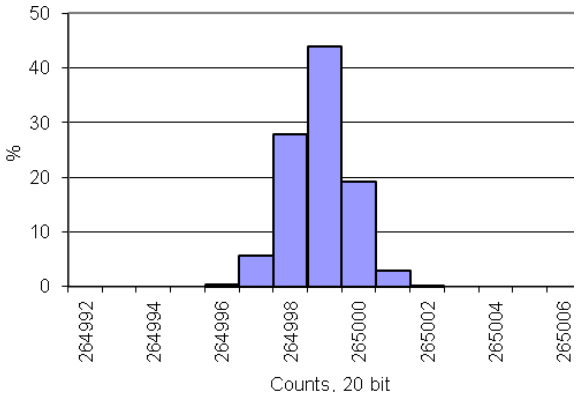
Delta-Sigma ADC I_{DD} versus sps, Unbuffered



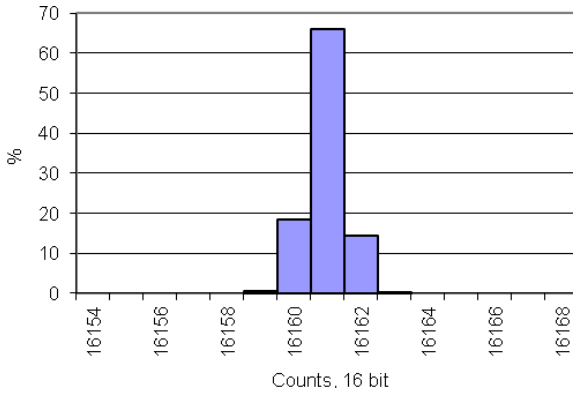
Delta-Sigma ADC I_{DD} versus sps, Buffered



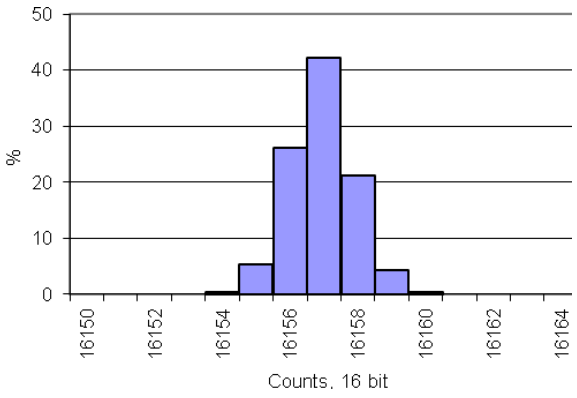
Delta-Sigma ADC Noise Histogram, 1000 Samples, 20-Bit, 187 sps, Ext Ref, $V_{IN} = V_{REF}/2$, Range = ±1.024 V



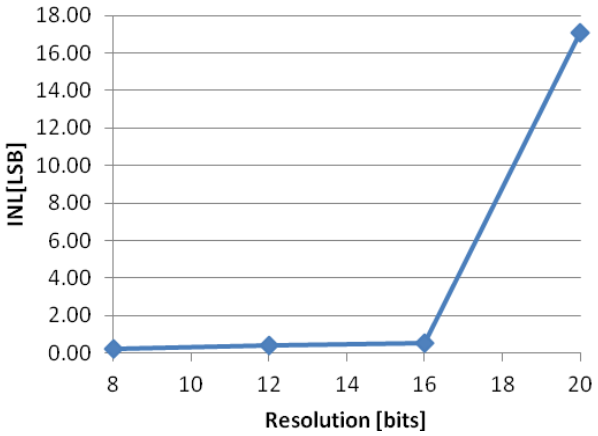
Delta-Sigma ADC Noise Histogram, 1000 samples, 16-bit, 48 ksps, Ext Ref, $V_{IN} = V_{REF}/2$, Range = ±1.024 V



Delta-Sigma ADC Noise Histogram, 1000 samples, 16-bit, 48 ksps, Int Ref, $V_{IN} = V_{REF}/2$, Range = ±1.024 V



Delta-Sigma ADC INL versus Resolution at Max Sample Rate



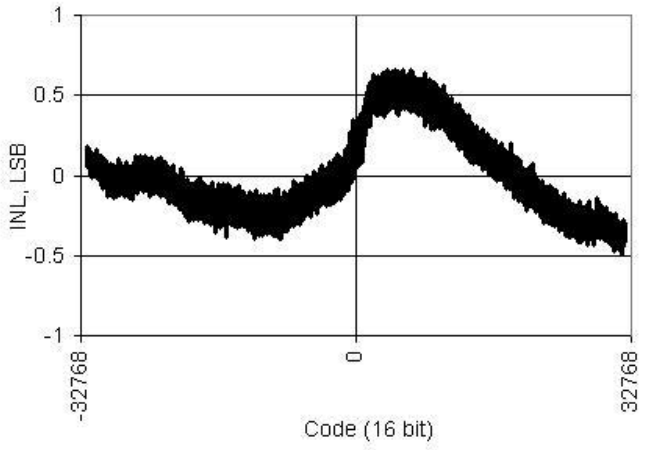
Delta-Sigma ADC RMS Noise in Counts versus Input Range and Sample Rate, 20-bit, External Reference, Single-Ended

Sample rate, SPS	Input Voltage Range			
	0 to VREF	0 to Vref × 2	VSSA to VDDA	0 to Vref × 6
8	1.28	1.24	6.02	0.97
23	1.33	1.28	6.09	0.98
45	1.77	1.26	6.28	0.96
90	1.65	0.91	6.84	0.95
187	1.87	1.06	7.97	1.01

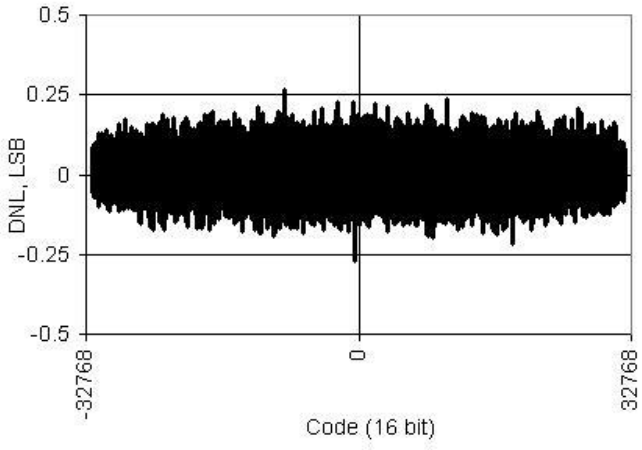
Delta-Sigma ADC RMS Noise in Counts versus Input Range and Sample Rate, 20-bit, External Reference, Differential

Sample rate, SPS	Input Voltage Range				
	±VREF	±Vref/2	±Vref/4	±Vref/8	±Vref/16
8	0.70	0.84	1.02	1.40	2.65
11.3	0.69	0.86	0.96	1.40	2.69
22.5	0.73	0.82	1.25	1.77	2.67
45	0.76	0.94	1.02	1.76	2.75
61	0.75	1.01	1.13	1.65	2.98
170	0.75	0.98	INVALID OPERATING REGION		
187	0.73				

Delta-Sigma ADC INL versus Output Code, 16-bit, 48 ksps, 25 °C V_{DDA} = 3.3 V



Delta-Sigma ADC DNL versus Output Code, 16-bit, 48 ksps, 25 °C V_{DDA} = 3.3 V



Input Capacitance Values (in pF) Used by Each of the Input Ranges for Resolutions 8 to 20

Input Range	Input Capacitance in pF	
	Resolution 8 to 15	Resolution 16 to 20
V_{SS} to V_{REF} (Single)	0.896	3.888
V_{SS} to V_{REF}*2 (Single)	0.896	3.888
V_{SS} to V_{DD} (V_{DD}/3 Single)	0.592	1.296

Input Range	Input Capacitance in pF	
	Resolution 8 to 15	Resolution 16 to 20
V_{SS} to V_{DD} (V_{DD}/4 Single)	0.496	0.992
V_{SS} to V_{REF}*6 (Single)	0.400	0.688
± V_{REF} (Diff)	0.896	3.888
± V_{REF}*2 (Diff)	0.400	1.888
± V_{REF}*6 (Diff)	0.400	0.688
± V_{REF}/2 (Diff)	1.600	6.000
± V_{REF}/4 (Diff)	2.800	12.000
± V_{REF}/8 (Diff)	3.488	17.588
± V_{REF}/16 (Diff)	5.696	17.200

DC and AC Electrical Characteristics for PSoC 5LP

Specifications are valid for –40 °C ≤ T_A ≤ 85 °C and T_J ≤ 100 °C, except where noted.
Specifications are valid for 1.71 V to 5.5 V, except where noted. Typical values are for T_A = 25°C

Operating conditions are:

- Operation in continuous sample mode
- fclk = 3.072 MHz for resolution = 16 to 20 bits; fclk = 6.144 MHz for resolution = 8 to 15 bits
- Reference = 1.024-V internal reference bypassed on P3[2] or P0[3]
- Unless otherwise specified, all charts and graphs show typical value

Delta-Sigma ADC DC Specifications for PSoC 5LP

Parameter	Description	Conditions	Min	Typ	Max	Units
	Resolution		8	–	20	bits
	Number of channels, single-ended		–	–	No. of GPIOs	
	Number of channels, differential	Differential pair is formed using a pair of GPIOs	–	–	No. of GPIOs/2	
	Monotonic	Yes	–	–	–	

Parameter	Description	Conditions	Min	Typ	Max	Units
Ge	Gain error	Buffered, buffer gain = 1, Range = ±1.024 V, 16-bit mode, 25 °C	–	–	±0.4	%
Gd	Gain drift	Buffered, buffer gain = 1, Range = ±1.024 V, 16-bit mode	–	–	50	ppm/°C
V_{OS}	Input offset voltage	Buffered, 16-bit mode, full voltage range, 25 °C	–	–	±0.2	mV
		Buffered, 16-bit mode, V_{DDA} = 1.7 V, 25 °C	–	–	±0.1	mV
TCV_{OS}	Temperature coefficient, input offset voltage	Buffer gain = 1, 16-bit, Range = ±1.024 V	–	–	0.55	µV/°C
	Input voltage range [8]	Single-ended, unbuffered	V_{SSA}	–	V_{DDA}	V
		Differential, unbuffered	V_{SSA}	–	V_{DDA}	V
		Differential, buffered	V_{SSA}	–	V_{DDA} – 1	V
PSRRb	Power supply rejection ratio, buffered [8]	Buffer gain = 1, 16-bit, Range = ±1.024 V	90	–	–	dB
CMRRb	Common mode rejection ratio, buffered [8]	Buffer gain = 1, 16-bit, Range = ±1.024 V	85	–	–	dB
INL20	INL for 20-bit [8]	Differential range ±1.024 V	–	–	±32	LSB
DNL20	DNL for 20-bit [8]	Differential range ±1.024 V	–	–	±1	LSB
INL16	Integral nonlinearity, 16-bit [8]	Differential range ±1.024 V	–	–	±2	LSB
DNL16	Differential nonlinearity 16-bit [8]	Differential range ±1.024 V	–	–	±1	LSB
INL12	INL for 12-bit [8]	Differential range ±1.024 V	–	–	±1	LSB
DNL12	DNL for 12-bit [8]	Differential range ±1.024 V	–	–	±1	LSB
INL8	INL for 8-bit [8]	Differential range ±1.024 V	–	–	±1	LSB
DNL8	DNL for 8-bit [8]	Differential range ±1.024 V	–	–	±1	LSB
Rin_Buff	ADC input resistance	Input buffer used	10	–	–	MΩ
Rin_ADC_16	ADC input resistance	Input buffer bypassed, 16-bit, range = ±1.024 V	–	74 [9]	–	kΩ

[8] Based on device characterization (not production tested).

[9] Using switched capacitors at the ADC input creates an effective input resistance. Holding the gain and number of bits constant, the resistance is proportional to the inverse of the clock frequency. This value is calculated, not measured. For more information, see the Technical Reference Manual.

Parameter	Description	Conditions	Min	Typ	Max	Units
Rin_ADC_12	ADC input resistance	Input buffer bypassed, 12-bit, range = ±1.024 V	–	148 [9]	–	kΩ
V_{EXTREF}	ADC external reference input voltage, see also internal reference in Voltage Reference Specifications	Pins P0[3], P3[2]	0.9	–	1.3	V
Current consumption						
I_{DD_20}	Current consumption, 20-bit, 187 sps [8]	Unbuffered	–	–	1.5	mA
I_{DD_16}	Current consumption, 16-bit, 48 ksps [8]	Unbuffered	–	–	1.5	mA
I_{DD_12}	Current consumption, 12-bit, 192 ksps [8]	Unbuffered	–	–	1.95	mA
I_{DD_8}	Current consumption, 8-bit, 384 ksps [8]	Unbuffered	–	–	1.95	mA
I_{BUFF}	Buffer current consumption [8]		–	–	2.5	mA

Delta-Sigma ADC AC Specifications for PSoC 5LP

Parameter	Description	Conditions	Min	Typ	Max	Units
	Startup time		–	–	4	Samples
	ADC_SelectConfiguration() API execution time	CPU clock = 24 Mhz	–	50	–	µs
THD	Total harmonic distortion, 16-bit [10]	Unbuffered, range ±1.024 V	–	–	0.0032	%
20-Bit Resolution Mode						
SR20	Sample Rate, 20-bit [10]	Differential range ±1.024 V	7.8	–	187	sps
BW20	Bandwidth, 20-bit [10]	Differential range ±1.024 V	–	40	–	Hz
16-Bit Resolution Mode						
SR16	Sample Rate, 16-bit [10]	Differential range ±1.024 V	2	–	48	ksps

[10] Based on device characterization (not production tested).

Parameter	Description	Conditions	Min	Typ	Max	Units
BW16	Bandwidth, 16-bit [10]	Differential range ±1.024 V	–	11	–	kHz
SINAD16int	Signal to noise + distortion - 16-bit, internal reference [10]	Unbuffered, range = ±1.024 V	81	–	–	dB
SINAD_16ext	Signal to noise + distortion, 16-bit, external reference [10]	Unbuffered, range = ±1.024 V	84	–	–	dB
12-Bit Resolution Mode						
SR12	Sample Rate, 12-bit [10]	Differential range ±1.024 V	4	–	192	ksps
BW12	Bandwidth, 12-bit [10]	Differential range ±1.024 V	–	44	–	kHz
SINAD12int	Signal to noise + distortion, 12-bit, internal reference [10]	Unbuffered, range = ±1.024 V	66	–	–	dB
8-Bit Resolution Mode						
SR8	Sample Rate, 8-bit [10]	Differential range ±1.024 V	8	–	384	ksps
BW8	Bandwidth, 8-bit [10]	Differential range ±1.024 V	–	88	–	kHz
SINAD8int	Signal to noise + distortion, 8-bit, internal reference [10]	Unbuffered, range = ±1.024 V	43	–	–	dB

Voltage Reference Specifications

Also see the ADC external reference specifications under V_{EXTREF} in Delta-Sigma ADC DC Specifications for PSoC 5LP.

Parameter	Description	Conditions	Min	Typ	Max	Units
V_{REF}	Precision reference voltage	Initial trimming	1.023 (–0.1%)	1.024	1.025 (+0.1%)	V
	Temperature drift [11]		–	–	30	ppm/°C
	Long term drift		–	100	–	ppm/Khr
	Thermal cycling drift (stability)[11]		–	100	–	ppm

[11] Based on device characterization (not production tested).

Component Errata

This section lists known problems with the component.

Cypress ID	Component Version	Problem	Workaround
191257	v3.0	This component was modified without a version number change in PSoC Creator 3.0 SP1. For more information, see Knowledge Base Article KBA94159 (www.cypress.com/go/kba94159).	No workaround is necessary. There is no impact to designs.
184285	v3.0	The **Buffer mode** parameter setting does not correspond to the associated configuration tab. That is, the setting made for **Config1** applies to **Config1** and **Config2**, the setting for **Config2** applies to **Config3**, and for **Config3** to **Config4**.	If different **Buffer mode** settings are needed in a design, use **Config1** and **Config3**, but configure the parameter for **Config3** using the **Config2** configuration tab.

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
3.0.c	Minor datasheet edit.	
3.0.b	Edited datasheet to add Component Errata section.	Document known problems with the component.
3.0.a	Minor datasheet edit.	
3.0	Added MISRA Compliance section.	The component does not have any specific deviations.
	Added Alignment option to Configuration tabs in customizer.	The component improvement.
	Adjusted the input capacitor ratio and gain correction register to allow accurate measuring of signals that swing the entire input range.	The change has impact on the ADC measurement results. The results can differ between version 3.0 and previous component versions.
	Fixed an issue with end of conversion status detection by ADC_IsEndConversion() API in Single Sample mode with resolution higher that 16 bits.	A nonzero value was always returned.
	For External clock mode, Sample rate field was made disabled and recalculated based on an external clock frequency.	Cannot build project w/o proper internal clock

Version	Description of Changes	Reason for Changes / Impact
	Added ADC_Read8(), ADC_Read16(), ADC_Read32() APIs.	This function simplifies getting results from the ADC when only a single reading is required.
	Added optional modulator input (mi).	This input is used to dynamically control the polarity of the signal in the modulator.
	The gain coefficient, calculated and written by ADC_SetGCOR() API is stored into the RAM and used by the following SelectConfiguration() API calls to initialize the GCOR register.	This change prevent from overwriting custom GCOR value for particular configuration.
	Added advanced parameter (rm_int) to optionally remove internal interrupt.	It is useful when external interrupt component is connected to EOC output and internal ISR isn't required.
	Fixed an issue when ADC_Wakeup() API started conversion after sleep mode even when conversion was disabled before.	ADC_Sleep() API saves the conversion status. ADC_Wakeup() API starts conversion only when conversion was enabled before sleep mode.
2.30.a	Fixed ADC Range diagram.	Diagram was incorrect.
2.30	Fixed an issue with ADC clock divider setting when using multiple configurations and ADC clock is not derived from BUS Clock	Clock divider settings issue with multiple ADC configurations
	Fixed an issue with charge pump clock divider settings when using multiple configurations	Charge pump clock divider settings issue when using multiple ADC configurations
	Fixed an issue where device gets reset when enabling reference to ADC with VDDA value greater than 5.3Volts	Device reset when enabling reference to ADC
	Fixed an issue with reference buffer setting when using multiple configurations	Reference buffer setting is wrong when using multiple ADC configurations.
	Added all APIs with the CYREENTRANT keyword when they are included in the .cyre file.	Not all APIs are truly reentrant. Comments in the component API source files indicate which functions are candidates. This change is required to eliminate compiler warnings for functions that are not reentrant used in a safe way: protected from concurrent calls by flags or Critical Sections.
	Added support PSoC 5LP silicon.	
	Updated Figure 4. Delta Sigma ADC I/O Characteristics in the datasheet.	To fix issues with Figure 4.
	Updated SetBufferGain() API description	Updates to mention side effects of using buffer gain.
	Added a note about using internal reference for ADC in the Reference parameter description section .	Internal reference is shared between SAR and DelSig ADC.
	Updated Side Effects section for ADC_IsEndConversion() API.	Updates to mention side effects of using DMA and ADC_IsEndConversion() API together.

Version	Description of Changes	Reason for Changes / Impact
2.20.a	Datasheet corrections	
2.20	Three new APIs ADC_SetCoherency(), ADC_SetGCOR() and ADC_ReadGCOR() are implemented	ADC_SetCoherency() for setting the coherency register. Helps to easily change the byte coherency. ADC_SetGCOR() for setting the Gain correction register. ADC_ReadGCOR()for reading the normalized GCOR register value.
	Added four new Bypass reference options.	Bypass reference options for Vdda/3 and Vdda/4 reference
	Added comment and config name parameters	The config parameter helps to uniquely name the ADC configurations. The comment parameter describes configuration use.
	ADC_Stop() API changes for PSoC 5	Change required to prevent the component from impacting unrelated analog signals when stopped, when used with PSoC 5.
	ADC power setting changes based on the ADC clock frequency	To optimize the power use by ADC component.
	Fixed an issue with ADC_SelectConfiguration() API.	Calling ADC_SelectConfiguration() API without calling ADC_Start() now works as expected.
	Added PSoC 5 characterization data to datasheet	
	Updated placeholder graphs in datasheet with real data	
	External Charge pump clock is the default option and cannot be user selectable.	ADC now always uses an external charge pump clock instead of an internal clock. This is to guarantee the optimal charge pump clock frequency to maximize ADC performance.
2.10	Two differential input range ±0.128 V(–Input ± Vref/8) ±0.064 V(–Input ± Vref/16) names are changed.	These input ranges have an error in their name. Note: Updating an existing design that uses an older version than 2.0 with the version 2.10 of the ADC_DelSig, may result in a parameter evaluation failed error message. To fix the problem, open the Configure dialog, and then toggle the selection for Input mode. Click OK to close the dialog and then build the project.
	Added two new reference options: Internal VDDA/4 and Internal VDDA/3.	The reference input for the PSoC 3 ADC has a narrow range, 0.9 V to 1.3 V. This may cause the VSS to VDDA ratiometric range to operate incorrectly when VDDA is less than 3.6 V. This new VDDA/3 reference corrects this problem.
	Changing the reference value affects the input range list.	Input range list updates are based on the reference voltage selected.

Version	Description of Changes	Reason for Changes / Impact
	Error provider is added to Vref Value parameter.	To warn the user if the Vref value exceeds the expected range 0.9V to 1.3V.
	ADC clock frequency is now dependent on input range selection.	ADC clock frequency depends on the selected input range. ADC maximum clock frequency differs based on the input range. If ADC clock frequency exceeds the maximum clock frequency for the selected input range, then an error symbol is set on the clock frequency text box.
	Fixed an issue with offset calculation.	0 to 2*Vref single ended mode for resolutions 9, 10, 11, 13, 14, and 15 now works as expected.
	Fixed an issue with DMA wizard tool generated code.	DMA wizard tool uses correct ADC output register when resolution is above 8 bits.
	Added characterization data to datasheet	
	Minor datasheet edits and updates	
2.0	Changed the Input Range parameter: added four new input ranges: Vssa to 6*Vref ±6.144 V(–Input ± 6*Vref) ±0.125 V(–Input ± Vref/8) ±0.0625 V(–Input ± Vref/16) **Note** This change will likely break existing designs.	When updating an existing project with the 2.0 version of the ADC_DelSig, this change may result in a parameter evaluation failed error message. To fix the problem, open the Configure dialog, and then toggle the selection for Input mode. Click **OK** to close the dialog and then build the project.
	Updated to support PSoC 3 ES2 or later and PSoC 5 or later.	This version supports PSoC 3 ES2 or later and PSoC 5 or later. Older versions of the component will display an error message when used with newer versions of the silicon.
	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 new parameters: Input Mode, and Buffer Mode. Removed the Power parameter and added new Conversion Mode “Single Sample.”	These new parameters were not present in the older versions of the ADC_DelSig component. When updating to version 2.0 of the component, the new parameters are given default values. The default value for the Input Mode parameter is “Differential.” The Input Mode parameter drives the selection of the Input Range. Therefore, an updated component previously configured with a “single-ended” Input Range will get a default value of “Differential.”
	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.	Adds the capability for customers to specify individual generated functions as reentrant.

Version	Description of Changes	Reason for Changes / Impact
	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 V_{SSA} to V_{DDA} input range is selected.
	Trim values are incorporated into the ADC implementation for the selected input ranges.	Trim values are 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, and so on, so that you can use them in your applications.
	New optional connection (nVref) has been added to the ADC DelSig component	This can be used to connect the ADC's reference Vssa to the analog global (AGL[6]).
	Charge pump power setting has been enabled depending on the clock frequency.	There was a problem with 8-bit ADC range. The problem was caused by 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 nonfunctioning 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.

© Cypress Semiconductor Corporation, 2013-2016. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage (“Unintended Uses”). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.