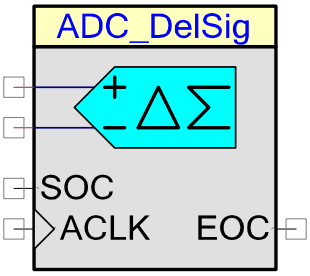
Delta Sigma Analog to Digital Converter (ADC_DelSig)

1.0

Features

- Wide range of resolutions, 8 to 20 bits
- Continuous mode
- High input impedance input buffer
- Selectable input buffer gain

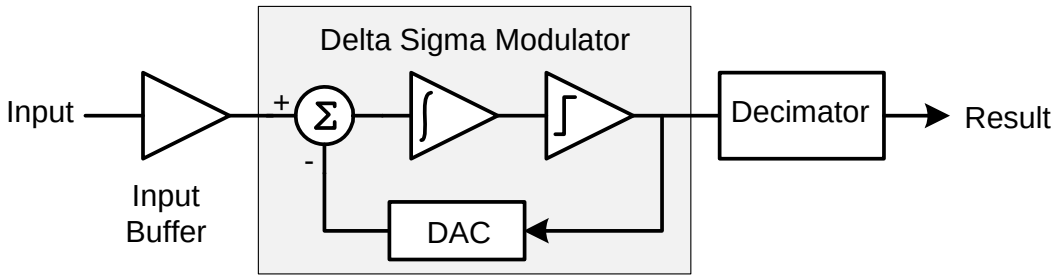


General Description

The Delta Sigma Analog to Digital Converter (ADC_DelSig) is ideal for sampling signals over a wide spectrum. The sample rate depending on mode and resolution can be adjusted between 10 and 375000 samples per second. Choices of conversion modes simplify interfacing to single streaming signals like audio, or multiplexing between multiple signal sources.

The ADC_DelSig is composed of three blocks: input amplifier, 3rd order Delta-Sigma modulator, and a decimator. The input amplifier provides a high impedance input and a user selectable input gain. The decimator block contains a 4 stage CIC Decimation filter and a post processing unit. The CIC filter operates on the data sample directly from the modulator. The post processing unit optionally performs gain, offset, and simple filter functions on the output of the CIC Decimator filter.

Figure 1: ADC_DelSig Block Diagram



When to use a ADC_DelSig

The ADC_DelSig is usable in a wide range of applications. It is capable of stereo 16-bit audio, high speed low resolution, and high precision 20-bit low speed conversions for sensors such as strain gauges, thermocouples and other high precision sensors. Care should be taken when selecting the Conversion Method in the parameter section, to best match the application.

PRELIMINARY

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 signals bandwidth. A low pass filter is used to filter out the noise outside the desired input signal bandwidth. This makes delta sigma converters good for both high speed medium resolution (8 to 16 bits) and low speed high resolution (16 – 20 bits) applications.

Input/Output Connections

This section describes the various input and output connections for the ADC_DelSig. 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.

+Input – Analog

This input is the positive analog signal input to the DS_ADC. The conversion result is a function of the +Input minus the voltage reference, which is either the –Input or Vss.

–Input – Analog *

This input is the negative analog signal input to the DS_ADC. It can also be thought of as the reference input. The conversion result is a function of the +Input minus the –Input. This pin will be visible when the “ADC_Input_Range” parameter is set to one of the two modes.

- 0.0 +/- 1.024V (Differential) -Input +/- Vref
- 0.0 +/- 2.048V (Differential) -Input +/- 2*Vref

SOC – Input *

Start of Conversion. This optional pin is shown if you have selected to use hardware to trigger a start of conversion. A rising edge on this pin will start an ADC conversion if the option is selected. See Start_of_Conversion parameter. If the parameter “Start_of_Conversion” is set to “Software” this I/O will be hidden.

ACLK – Input *

This optional pin is present if the ADC_Clock parameter is set to “External”, otherwise the pin will not be shown. This clock determines the conversion rate as a function of conversion method and resolution. If the “ADC_Clock” parameter is set to internal, this I/O will be hidden.

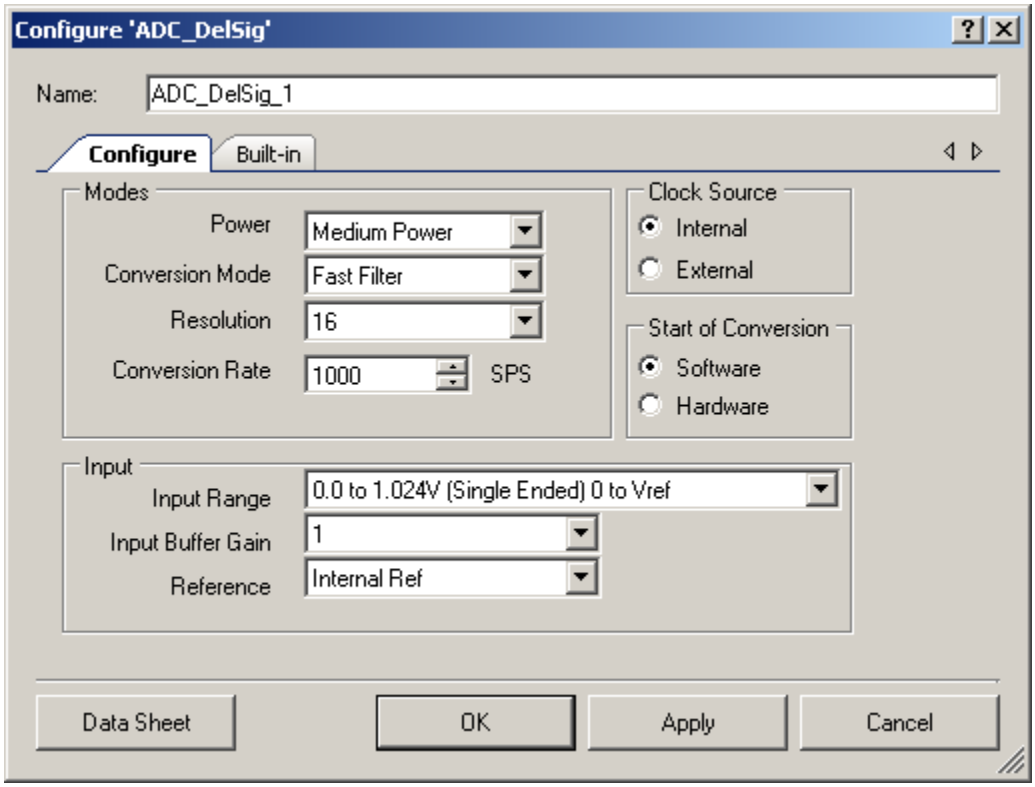
EOC – Output

A rising edge on the End Of Conversion (EOC) signals that a conversion is complete. A DMA request may be connected to this pin to transfer the conversion output to system RAM, DFB, or other component. The internal interrupt is also connected to this signal.

Parameters and Setup

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

Figure 2: Configure ADC_DelSig Dialog



The ADC_DelSig has the following parameters. The option shown in bold is the default.

ADC_Clock

This parameter allows you to select either a clock that is internal to the DS_ADC module or an external clock.

ADC_Clock	Description
Internal	Use an internal clock that is part of the DS_ADC component.
External	Use an external clock. The clock source may be analog, digital, or generated by another component.

ADC_Input_Range

This parameter configures the ADC for a given input range. This configures the input to the ADC and is independent of the input buffer gain setting.

Input Range	Description
0.0 to 1.024V (Single Ended) 0 to Vref	When using the internal reference (1.024V), the usable input voltage to the ADC is 0 .0 to 1.024 Volts. The ADC will be configured to operate in a single ended input mode with the –Input connected internally to Vssa. The negative input buffer amplifier will be powered down and the positive input buffer will be configured to operate rail-to-rail. If an external reference voltage is used, the usable input range to the ADC will be 0.0 to Vref.
0.0 to 2.048V (Single Ended) 0 to 2*Vref	When using the internal reference (1.024V), the usable input voltage to the ADC is 0 .0 to 2.048 Volts. The ADC will be configured to operate in a single ended input mode with the –Input connected internally to Vssa. The negative input buffer amplifier will be powered down and the positive input buffer will be configured to operate rail-to-rail. If an external reference voltage is used, the usable input range to the ADC will be 0.0 to (2 * Vref).
Vssa to Vdda (Single Ended)	This mode uses the Vdda/4 reference and a ADC input (not buffer) gain of 4 so the usable input range will cover the full analog supply voltage. The ADC will put in a single ended input mode with the –Input connected internally to Vssa. The negative input buffer amplifier will be powered down and the positive input buffer will be configured to operate rail-to-rail.
0.0 +/- 1.024V (Differential) -Input +/- Vref	This mode is configured for differential inputs. When using the internal reference (1.024V), the input range will be –Input +/- 1.024V. If –Input is connected to 2.048 volts the usable input range is 2.048 +/- 1.024V or 1.024 to 2.072V. For systems where both single ended and differential signals are scanned, connect the –Input to Vssa when scanning a single ended input. This mode does not use the rail-to-rail buffer mode, but it shouldn't be required for most applications. An external reference may be used to provide a wider operating range. The usable input range can be calculated with the same equation, -Input +/- Vref.
0.0 +/- 2.048V (Differential) -Input +/- 2*Vref	This mode is configured for differential inputs. When using the internal reference (1.024V), the input range will be –Input +/- 2.048V. If –Input is connected to 2 volts the usable input range is 2.048 +/- 2.048V or 0.0 to 4.096V. For systems where both single ended and differential signals are scanned, connect the –Input to Vssa when scanning a single ended input. This mode does not use the rail-to-rail buffer mode, but it shouldn't be required for most applications. An external reference may be used to provide a wider operating range. The usable input range can be calculated with the same equation, -Input +/- 2*Vref.

ADC_Power

This parameter sets the power level of the ADC. The High Power settings allow the ADC to operate at higher clock rates for faster conversion times. The Low Power setting provides a low power alternative when fast conversion speed is not critical.

ADC_Power	Description
Low Power	Lowest power setting
Medium Power	
High Power	Highest power setting

Show a graph of power vs. clocks.

ADC_Reference

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

ADC_Reference	Description
Internal Ref	Use the internal 1.024V reference
Internal Ref, bypassed on P03	Use internal 1.024V reference, and place a bypass capacitor on pin P03.
Internal Ref, bypassed on P32	Use internal 1.024V reference, and place a bypass capacitor on pin P32.
External Ref on P03	Use external reference on pin P03.
External Rev on P32	Use external reference on pin P32.

ADC_Resolution

Sets the resolution of the ADC. The higher the resolution the slower the sample rate is for each setting.

ADC_Resolution	Description
8	Sets resolution to 8 bits.
9	Sets resolution to 9 bits.
10	Sets resolution to 10 bits.
11	Sets resolution to 11 bits.
12	Sets resolution to 12 bits.
13	Sets resolution to 13 bits.

ADC_Resolution	Description
14	Sets resolution to 14 bits.
15	Sets resolution to 15 bits.
16	Sets resolution to 16 bits.
17	Sets resolution to 17 bits.
18	Sets resolution to 18 bits.
19	Sets resolution to 19 bits.
20	Sets resolution to 20 bits.

Add an equation and/or graph to show the usable sample rate and resolution.

Buffer_Gain

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

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.
Disable Buffer	Disables the input buffer gain.

Conversion_Method

This parameter selects the mode of operation at which the ADC operates.

Conversion Method	Description
Fast_Filter	Continuous single sample with ADC channel resets between samples. 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, as with Delta-Sigma converters. There will be a delay of two 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 desired sample rate.
Continuous	Continuous sample mode operates as a normal Delta-Sigma converter. Use this mode when measuring a single input signal. This mode should not be used when

Conversion Method	Description
	multiple signals need to be measure with a single ADC. There will be a delay of two 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 desired sample rate.
Fast_FIR	TBD

Sample_Rate

The ADC sample rate is selected with this parameter. When in a single conversion mode, the conversion time will simple be the reciprocal of the sample rate. The sample rate is entered in Hertz or samples per second. The maximum sample rate is a function resolution and sample mode. The higher the resolution, the lower the sample rate. See graph below for valid sample rates for each resolution.

Place graph of sample rate vs resolution here.

Start_of_Conversion

This parameter determines how ADC conversions will be started.

Start_of_Conversion	Description
Software	Firmware is used to start a conversion.
Hardware	A rising edge pulse on the SOC pin will cause a conversion to start.

Placement

Not applicable

Resources

The DS_ADC uses a decimator, Delta-Sigma modulator, and a clock source. If an external reference or reference bypass is selected, P03 or P32 can be used for the external reference or bypass capacitor.

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

Application Programming Interface

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

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

Function	Description
void ADC_Start(void)	Power up ADC and reset all states.
void ADC_Stop(void)	Stop ADC conversions and power down.
void ADC_SetPower(uint8 power)	Set power level.
void ADC_SetBufferGain(uint8 gain)	Select input buffer gain (1,2,4,8)
void ADC_StartConvert(void)	Start conversion.
void ADC_StopConvert(void)	Stop Conversions
void ADC_IRQ_Enable(void)	Enables interrupts at end of conversion.
void ADC_IRQ_Disable(void)	Disables interrupts.
uint8 ADC_IsEndConversion(uint8 retMode)	Returns a non-zero value if conversion is complete.
int8 ADC_GetResult8(void)	Returns an 8-bit conversion result, right justified.
int16 ADC_GetResult16(void)	Returns a 16-bit conversion result, right justified.
int32 ADC_GetResult32(void)	Returns a 32-bit conversion result, right justified.

void ADC_Start(void)

Description: Configures and powers up the ADC, but does not start conversions.

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_SetPower(uint8 power)

Description: Sets the operational power of the ADC. The higher power settings should be used with faster clock speeds.

Parameters: (uint8) power: Power setting.

Power Options	Description
ADC_LOWPOWER	Lowest power setting
ADC_MEDPOWER	
ADC_HIGHPOWER	Highest power setting

Return Value: None

Side Effects: Power setting may affect conversion accuracy.

void ADC_SetBufferGain(uint8 gain)

Description: Sets the input buffer gain.

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

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

Return Value: None

Side Effects: None

void ADC_StartConvert(void)

Description: Forces the ADC to initiate a conversion. If in the Single_Sample mode, one conversion will be performed then the ADC will halt. If in one of the other three conversion modes, the ADC will run continuously.

Parameters: None

Return Value: None

Side Effects: None

void ADC_StopConvert(void)

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

Parameters: None

Return Value: None

Side Effects: None

void ADC_IRQ_Enable(void)

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

Parameters: None

Return Value: None

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

void ADC_IRQ_Disable(void)

Description: Disables interrupts at the end of a conversion.

Parameters: None

Return Value: None

Side Effects: None

uint8 ADC_IsEndConversion(uint8 retMode)

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

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

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

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

Side Effects: None

int8 ADC_GetResult8(void)

Description: This function will return the result of an 8-bit conversion. If the resolution is set greater than 8-bits, the LSB of the result will be returned.

Parameters: None

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

Side Effects: None

int16 ADC_GetResult16(void)

Description: Returns a 16-bit result for a conversion with a result that has a resolution of 8 to 16 bits. If the resolution is set greater than 16-bits, it will return the 16 least significant bits of the result.

Parameters: None

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

Side Effects: None

int32 ADC_GetResult32(void)

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

Parameters: None

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

Side Effects: None

Sample Firmware Source Code

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

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

```
#include <device.h>

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

Interrupt Service Routine

The ADC_DelSig contains a blank interrupt service routine in the file ADC_INT.c file, where “ADC_DelSig_1” is the instance name. You may place custom code in the designated areas to perform whatever function is required at the end of a conversion. A copy of the blank interrupt service routine is shown below. Place custom code between the “ **/* `#START MAIN_ADC_ISR` */** ” and **“/* `#END` */** “ comments. This ensures that the code will be preserved when a project is regenerated.

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

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

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

```
/*********************************/
/* System variables               */
/*********************************/
/* `#START ADC_SYS_VAR`  */

    /* Place user code here. */

/* `#END`  */
```

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

```
#include <device.h>

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

           /* More user code */
        }
    }
}
```

Interrupt code segments in the file ADC_DelSig_1_INT.c.

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

CY_ISR(ADC_DelSig_1_ISR )
{
    /*********************************************/
    /* Place user ADC ISR code here.             */
    /* This can be a good place to place code    */
    /* that is used to switch the input to the   */
```

```
        /* ADC.  It may be good practice to first     */
        /* Stop the ADC before switching the input    */
        /* then restart the ADC.                      */
        /********************************************/
        /* `#START MAIN_ADC_ISR`  */
            result = ADC_DelSig_1_GetResult16();
            dataReady = 1;
        /* `#END`  */
    }
```

Functional Description

The following table provides preliminary minimum and maximum sample rates for each mode and resolution of the ADC_DelSig component.

Table 1 Minimum and Maximum Sample Rates (Samples/Sec) for ADC_DelSig

	Fast Filter		Continuous		Fast FIR	
Resolution	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
8	1000	93000	4000	375000	1000	85000
9	1000	93000	4000	375000	1000	85000
10	1000	93000	4000	375000	1000	85000
11	500	47000	2000	190000	500	44000
12	500	47000	2000	190000	500	44000
13	500	47000	2000	190000	500	44000
14	500	47000	2000	190000	500	44000
15	250	11500	2000	48000	500	11000
16	250	11500	2000	48000	500	11000
17	50	2300	500	12000	100	2200
18	15	700	125	3000	30	650
19	10	85	16	375	10	90
20	10	45	10	185	10	45

(Add theory of operation and more details.)

User Registers

Sample Registers

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

ADC_DEC_SAMP (ADC Output Data Sample Low Register)

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

ADC_DEC_SAMPM (ADC Output Data Sample Middle Register)

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

ADC_DEC_SAMPH (ADC Output Data Sample High Register)

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

Gain Correction Registers

Each ADC result may be multiplied by a 16-bit gain correction value. This gain can be used to compensate for system gain errors or in conjunction with the Offset Correction Registers to automatically convert the ADC output to a useful unit such as temperature or pressure. The Gain Correction Register is 16-bits across two 8-bit registers. The ADC output sample is multiplied by the gain correction value after the offset correction has been added.

ADC_DEC_GCOR (Gain Correction Low Register)

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

ADC_DEC_GCORH (Gain Correction High Register)

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

Offset Correction Registers

The post processing stage of the ADC provides a method to add a constant offset to the ADC conversion result before the result is place in the output sample register. This offset can be used to calibrate system offsets or in conjunction with the Gain Correction Registers to automatically convert the ADC output to a useful unit such as temperature or pressure. This Offset Correction Register is 24-bits across three 8-bit registers. The offset correction value is added to the sample result prior to the gain correction.

ADC_DEC_OCOR (ADC Offset Correction Low Register)

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

OCOR[7:0]: Offset Correction Register

ADC_DEC_OCORM (ADC Offset Correction Middle Register)

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

OCOR[15:8]: Offset Correction Register

ADC_DEC_OCORH (ADC Offset Correction High Register)

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

OCOR[23:16]: Offset Correction Register

DC and AC Electrical Characteristics

The following values are indicative of expected performance and based on initial characterization data.

5.0V/3.3V DC and AC Electrical Characteristics

Parameter	Typical	Min	Max	Units	Conditions and Notes
Input					
Input Voltage Range	---		Vss to Vdd	V	
Input Capacitance	---		---	pF	
Input Impedance	---		---	Ω	
Maximum Clock Rate	---		100	MHz	
Resolution			8 to 20	bits	
Maximum sample rate				sps	
SNR				dB	
DC Accuracy					
DNL				LSb	
INL				LSb	
Offset Error				uV	
Gain Error				uV	
Operating Current					
Low Power				uA	
Medium Power				uA	
High Power				uA	

PRELIMINARY