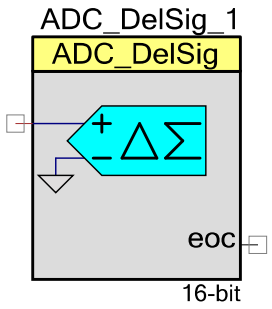
Delta Sigma Analog to Digital Converter (ADC_DelSig)

1.30

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 can be adjusted between 10 and 375000 samples per second, depending on mode and resolution. Choices of conversion modes simplify interfacing to single streaming signals like audio, or multiplexing between multiple signal sources.

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. Take care to select the appropriate **Conversion Method** parameter to best match the application.

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. 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 to 20 bits) applications.

PRELIMINARY

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 ADC_DelSig. The conversion result is a function of the +Input minus the voltage reference, which is either the –Input or Vssa.

–Input – Analog *

This input is the negative analog signal input to the ADC_DelSig. 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 following modes.

- 0.0 +/- 1.024 V (Differential) -Input +/- Vref
- 0.0 +/- 2.048 V (Differential) -Input +/- 2*Vref
- 0.0 +/- 0.512 V (Differential) -Input +/- Vref/2
- 0.0 +/- 0.256 V (Differential) -Input +/- Vref/4

soc – Input *

The Start of Conversion (soc) is an optional pin. It 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. If the **Start of Conversion** parameter is set to "Software" this I/O will be hidden.

aclk – Input *

This optional pin is present if the **Clock Source** 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.

Note The ADC Clock must be derived from the system bus clock. If a clock source external to the PSoC is required to drive the ADC_DelSig, the external clock must source the system bus clock. The clock connected to the ADC_DelSig can then be derived from any internal clock that is based on the system bus clock.

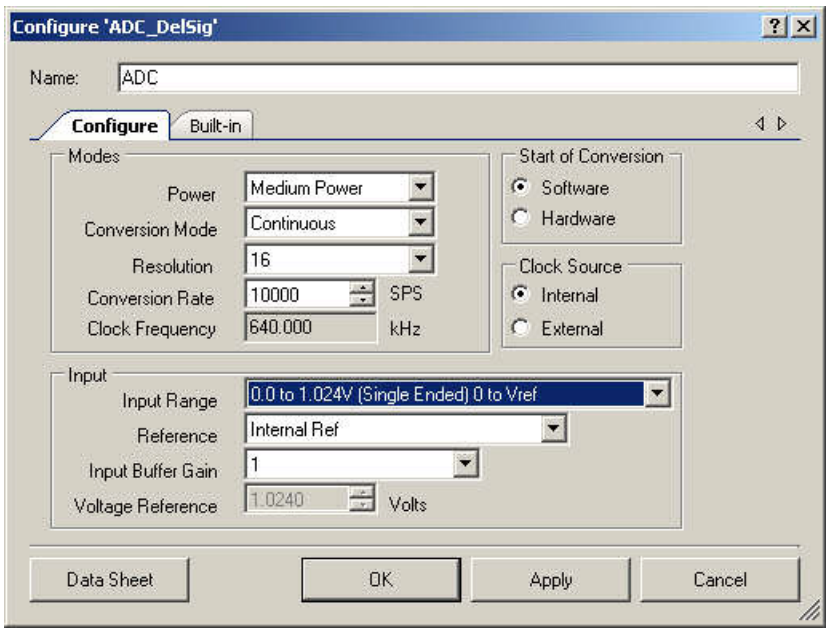
eoc – Output

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

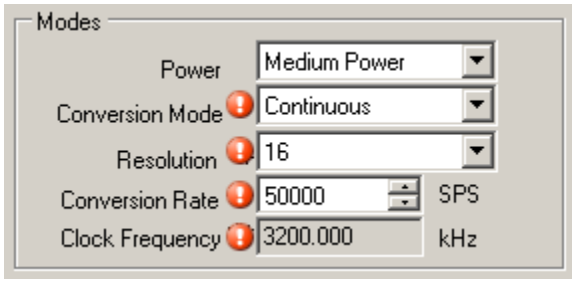
The Delta Sigma ADC is a highly configurable analog to digital converter. Drag an ADC_DelSig component onto your design and double-click it to open the Configure dialog.

Figure 1: Configure ADC_DelSig Dialog



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. If an invalid setting for these parameters occurs, a red circle with an exclamation point will appear, as follows:



If you hover the cursor over one of the error symbols, it will display an error message. For example, in the case above, the following error message would appear: "The ADC clock frequency of 3200.000 kHz has exceeded the maximum limit of 3072.000 kHz for the conversion mode, resolution and sample rate." If this occurs, change the parameters as needed to comply with the ADC specifications.

Power

This parameter sets the power level of the ADC. The "High Power" setting allows 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.

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

TBD: Graph of power vs. clocks.

Conversion Mode

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

Conversion Mode	Description
Fast_Filter	Continuous single sample with ADC channel resets between samples. That is, Fast Filter mode captures single samples back to back, resetting itself and Modulator between each sample. Upon completion of a sample, the next sample is initiated continuously. Fast Filter mode is simply a continuous string of Single Sample switch channel resets between them. 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. **Note** Care should be taken when switching signals between ADC conversions. Either switch the input quickly with hardware control or Stop the ADC while switching the input then restart the ADC after the new signal has been connected to the ADC. Failure to do this may result in crosstalk between signals in the ADC results.
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 multiple signals need to be measured with a single ADC. There will be a delay of three extra 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	The Fast FIR mode operates identically to the Fast Filter mode between 8 and 16 bits of resolution. For resolutions of 17 to 20 bits, the performance is about 4 times faster than the Fast Filter mode. This is because the modulator is only reset once at the end of conversion. **Note** Care should be taken when switching signals between ADC conversions. Either switch the input quickly with hardware control or Stop the ADC while switching the input then restart the ADC after the new signal has been connected to the ADC. Failure to do this may result in crosstalk between signals in the ADC results.

Resolution

This parameter sets the resolution of the ADC. The higher the resolution the slower the sample rate is for each setting.

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.
14	Sets resolution to 14 bits.
15	Sets resolution to 15 bits.
16	Sets resolution to 16 bits (default).
17	Sets resolution to 17 bits.
18	Sets resolution to 18 bits.
19	Sets resolution to 19 bits.
20	Sets resolution to 20 bits.

TBD: Equation and/or graph to show the usable sample rate and resolution.

Conversion Rate

The ADC conversion rate is selected with this parameter. It is entered in samples per second (SPS). 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.

TBD: Graph of sample rate vs resolution here.

Clock Frequency

This text box is a non-editable area that is used to inform the user what the desired clock rate is for the selected modes. When using an external clock, the customizer still calculates the desired clock rate to indicate what frequency the external clock should be.

The following table provides minimum and maximum sample rates for each mode and resolution of the ADC_DelSig component. This table is based on the minimum clock rate for all modes and resolution of 128 kHz. The maximum clock rate is 6.144 MHz for resolutions up to 14 bits. For resolutions above 14 bits (15 to 20) the maximum ADC clock is 3.072 MHz.

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	1911	91701	8000	384000	1829	87771
9	1911	91701	8000	384000	1829	87771
10	1911	91701	8000	384000	1829	87771
11	978	46900	4000	192000	956	45850
12	978	46900	4000	192000	956	45850
13	978	46900	4000	192000	956	45850
14	978	46900	4000	192000	956	45850
15	495	11861	2000	48000	489	11725
16	495	11861	2000	48000	489	11725
17	124	2965	500	12000	282	6766
18	31	741	125	3000	105	2513
19	4	93	16	375	15	357
20	2	46	8	187	8	183

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.

If a start of conversion occurs in the middle of a conversion, the SOC signal works just like the ADC_StartConvert( ) command. The ADC starts conversion on the first rising edge and continues until the ADC_StopConvert( ) is called.

Clock Source

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

Clock Source	Description
Internal	Use an internal clock that is part of the ADC_DelSig component.
External	Use an external clock. The clock source may be a standard clock component, or generated by another component. The external clock provided must be configured to be a 50% duty cycle or the ADC will not operate properly.

Input Range

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

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 3.072V. 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.

Input Range	Description
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 2.028 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.
0.0 +/- 0.512V (Differential) -Input +/- 0.5*Vref	This mode is configured for differential inputs. When using the internal reference (1.024V), the input range will be –Input +/- 0.512V. If –Input is connected to 1.0 volt the usable input range is 1.0 +/- 0.512V or 0.488 to 1.512V. 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 +/- 0.5*Vref.
0.0 +/- 0.256V (Differential) -Input +/- 0.25*Vref	This mode is configured for differential inputs. When using the internal reference (1.024V), the input range will be –Input +/- 0.256V. If –Input is connected to 1.0 volt the usable input range is 1.0 +/- 0.256V or 0.744 to 1.256V. 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 +/- 0.25*Vref.

Reference

This parameter selects the ADC_DelSig 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.

Note The use of an external bypass capacitor is recommended if the internal noise caused by internal digital switching exceeds what is required for the application's analog performance. To use this option, configure either port pin P03 or P32 as an analog "Hi-Z" pin and connect an external capacitor between 0.01uF and 10uF.

TBD: Graph of cap value vs. noise.

Input 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.

Input_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.

Voltage Reference

This parameter is non-editable when using the internal 1.024 volt reference. When using an external reference, you may edit this value to match the external reference voltage. When the input range "Vssa to Vdda (Single Ended)" is selected, the Vdda supply voltage value should be entered. The voltage reference is used for the ADC count to voltage conversion functions discussed in the API section.

Placement

Not applicable

Resources

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

	Digital Blocks					API Memory (Bytes)		
Resolution	Datapaths	Macro cells	Status Registers	Control Registers	Counter7	Flash	RAM	Pins (per External I/O)
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_SetOffset(int32 offset)	Set offset of ADC.
void ADC_SetGain(int32 adcGain)	Sets ADC gain in counts per volt.
int16 ADC_CountsTo_mVolts(int32 adcCounts)	Convert ADC counts to mVolts.
Int32 ADC_CountsTo_uVolts(int32 adcCounts)	Convert ADC counts to uVolts.
float ADC_CountsTo_Volts(int32 adcCounts)	Convert ADC counts to floating point Volts.

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 this function is called while the conversion is in progress, the next conversion start is queued and will start after the current conversion is finished. If you want to start a new conversion without waiting for current conversion to finish, then call the StopConvert function to stop the current conversion in progress, and then call the StartConvert function.

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. When the ADC is configured for 8-bit single ended mode, the ADC_GetResult16() function should be used instead. This function returns only signed 8-bit values. The maximum positive signed 8-bit value is 127, but in singled ended 8-bit mode, the maximum positive value is 255.

Parameters: None

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

Side Effects: None

int16 ADC_GetResult16(void)

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

Parameters: None

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

Side Effects: None

int32 ADC_GetResult32(void)

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

Parameters: None

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

Side Effects: None

void ADC_SetOffset(int32 offset)

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

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

Return Value: None.

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

void ADC_SetGain(int32 adcGain)

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

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

Return Value: None.

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

int16 ADC_CountsTo_mVolts(int32 adcCounts)

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

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

Return Value: (int32) Result in mVolts.

Side Effects: None

int32 ADC_CountsTo_uVolts(int32 adcCounts)

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

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

Return Value: (int32) Result in uVolts.

Side Effects: None

float ADC_CountsTo_Volts(int32 adcCounts)

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

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

Return Value: (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();
}
```

Example for Multiplexing Analog Input at Runtime

```
#include <device.h>

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

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

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

Interrupt Service Routine

The ADC_DelSig contains a blank interrupt service routine in the file *ADC_DelSig_1_INT.c* file, where "ADC_DelSig_1" is the instance name. You may place custom code in the designated areas to perform whatever function is required at the end of a conversion. A copy of the blank interrupt service routine is shown below. Place custom code between the " /* `#START MAIN_ADC_ISR` */ " and "/* `#END` */ " comments. This ensures that the code will be preserved when a project is regenerated.

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

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

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

```
/* System variables */

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

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*.

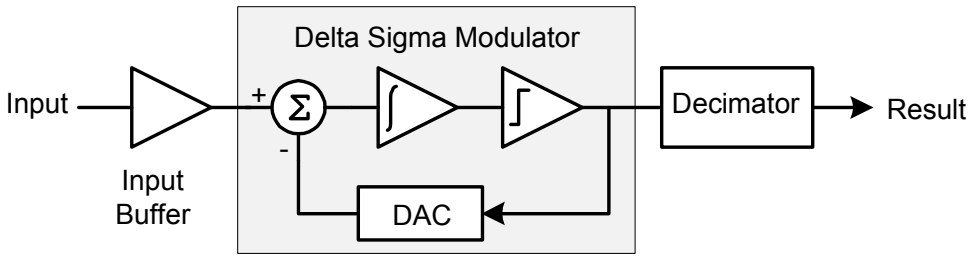
```
    /*********************************
    *      System variables
    *********************************/
    /* `#START ADC_SYS_VAR`  */
     extern int16 result;
     extern uint8 dataReady;
     /* `#END`  */

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 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 2: ADC_DelSig Block Diagram



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 placed 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.

DC Specifications

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	
	Monotonicity		Yes	-	-	

Parameter	Description	Conditions	Min	Typ	Max	Units
	Gain error	Input buffer bypassed	-	-	±0.2	%
	Input offset voltage		-	-	±0.1	mV
	Current consumption					
	48 ksps, 16-Bit Mode	High power mode, ADC clock = 3.072 MHz	-	1	3.4	mA
	192 ksps, 12-Bit Mode	High power mode, ADC clock = 6.144 MHz	-	-	3.75	mA
		Medium power mode	-	-	2.72	mA
		Low power mode	-	-	2.56	mA
	Input voltage range - single ended		Vssa	-	Vdda	V
	Input voltage range - differential		Vssa	-	Vdda	V
	Input voltage range - differential (buffered)		Vssa		Vdda - 1	V
	Input resistance	Input buffer used	10	-	-	MΩ
		Input buffer bypassed, 16 bits, ADC clock = 3.072 MHz, gain = 1	-	74	-	kΩ
		Input buffer bypassed, 12 bits, ADC clock = 6.144 MHz, gain = 1	-	148	-	kΩ
THD	Total harmonic distortion	Input buffer used	-	-	0.0032	%

AC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
	Startup time		-	-	4	samples
PSRR	Power supply rejection ratio	Input buffer used	90	-	-	dB
CMRR	Common mode rejection ratio	Input buffer used	90	-	-	dB
	20-Bit Resolution Mode					
	Sample rate	ADC clock = 3.072 MHz , continuous sample mode	-	-	180	sps
SNR	Signal-to-noise ratio (SNR)	Vdda ≥ 2.7V, input buffer bypassed	110	-	-	dB
	Input bandwidth		-	40	-	Hz
INL	Integral non linearity		-	-	16	LSB
DNL	Differential non linearity		-	-	1	LSB
	16-Bit Resolution Mode					
	Sample rate	ADC clock = 3.072 MHz,	-	-	48	ksps

Parameter	Description	Conditions	Min	Typ	Max	Units
		continuous sample mode				
SNR	Signal-to-noise ratio (SNR)	Vdda ≥ 2.7V, input buffer bypassed	90	-	-	dB
	Input bandwidth		-	11	-	kHz
INL	Integral non linearity		-	-	1	LSB
DNL	Differential non linearity		-	-	1	LSB
	12-Bit Resolution Mode					
	Sample rate	ADC clock = 6.144 MHz, continuous sample mode, input buffer bypassed	-	-	192	ksps
		ADC clock = 3.072 MHz, continuous sample mode, input buffer used	-	-	160	ksps
SNR	Signal-to-noise ratio (SNR)	Vdda ≥ 2.7V, input buffer bypassed	70	-	-	dB
	Input bandwidth		-	44	-	kHz
INL	Integral non linearity		-	-	1	LSB
DNL	Differential non linearity		-	-	1	LSB

Component Changes

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

Note The 1.30 version of the ADC DelSig component does not support PSoC 3 ES3 silicon; use version 2.0 instead.

Version	Description of Changes	
1.30.b	Added information to the component that advertizes its compatibility with silicon revisions.	The tool reports an error/warning if the component is used on incompatible silicon. If this happens, update to a revision that supports your target device.
1.30.a	Data sheet correction: fixed typo in ADC_SetGain( ) API. Updated internal expression for the CY_REMOVE parameter; no affect on component behavior.	
1.30	Removed SW3 configuration in ADC_Start() API. Added power management changes to ADC_Stop() API. Modified firmware write to ADC Power Management Register CR5. Max SPS for buffered mode is 160 KSPS for 12-15 bit resolution.	

© Cypress Semiconductor Corporation, 2009-2010. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks and of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.