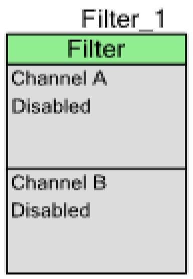
Filter
2.0

Features



- Easy user configuration of filters running on the Digital Filter Block (DFB) available in some PSoC 3 and PSoC 5 devices
- Supports two separate filter channels, each one constructed as a cascade of up to four separately designed stages.
- Multiple FIR and IIR (Biquad) filter methods (including user coefficient entry) give great flexibility
- Final coefficient values can be extracted for further analysis

General Description

The customizer for the Filter component allows you to configure digital filters on one or two data streams passed to the Digital Filter Block (DFB), using DMA, interrupts, or polling to manage data flow. The DFB's 128 data and coefficient locations are shared as needed between the two filter channels. The customizer reports (but does not set) the minimum bus clock frequency required to execute the filtering within the user-declared sample interval.

This component supports a huge number of use cases. If you encounter something unusual when using it, report it (with a good description of what you did to cause it) to psoc_creator_fb@cypress.com so Cypress can investigate.

Input/Output Connections

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

Interrupt – Output *

If either channel is configured to generate an interrupt in response to a data-ready event, the interrupt output is enabled. Connect the hardware signal to an ISR component to handle the interrupt routine. This terminal is only visible when a channel selects "Interrupt" as the data ready signal. The terminal is shared between both channels.

DMA Request – Output *

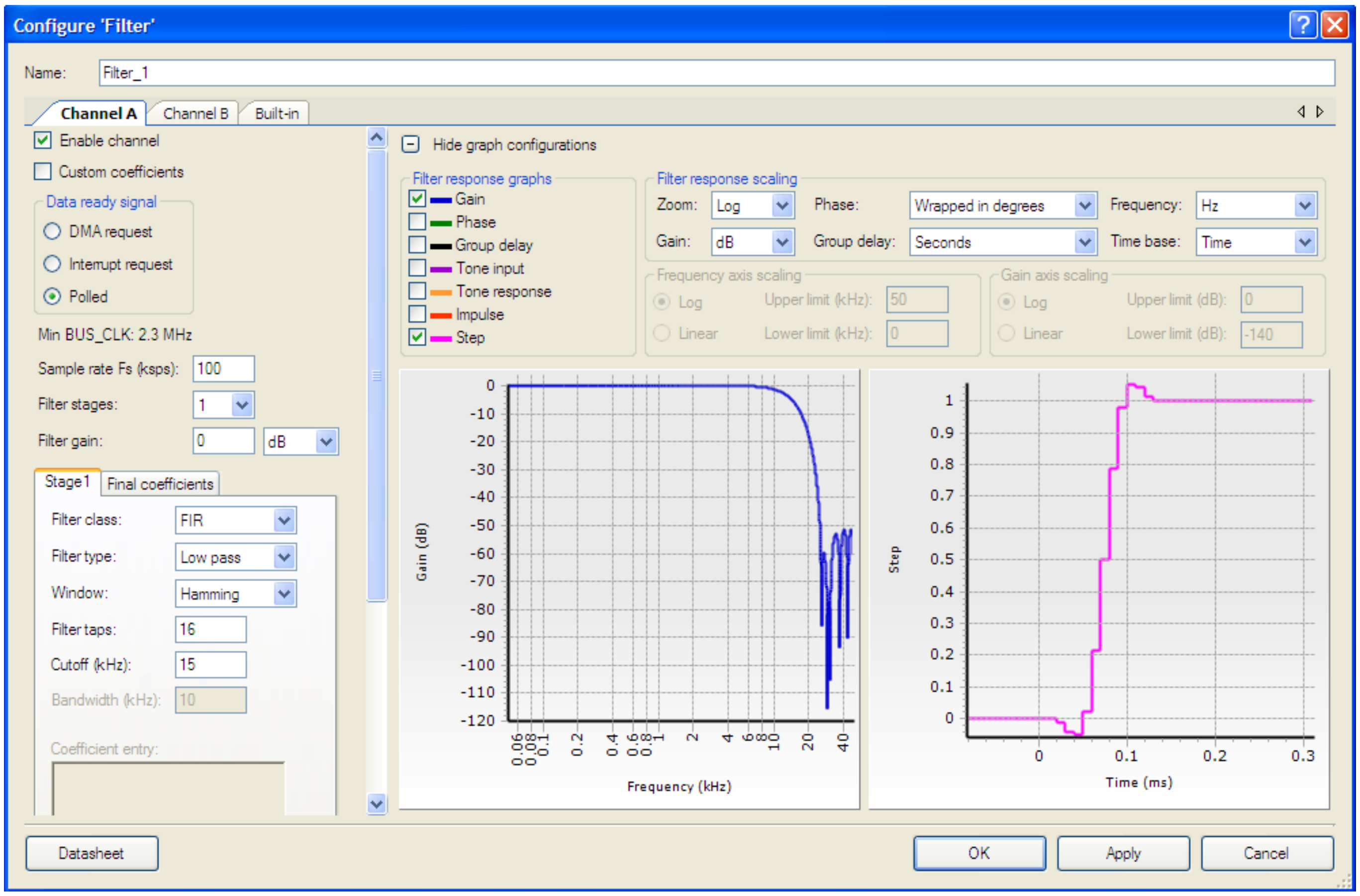
If either channel is configured to generate a DMA request in response to a data-ready event, the DMA Request output is enabled *for that channel*. Each channel has a separate DMA Request output. Connect the hardware signal to a DMA component to handle the DMA routine.

The DMA Request output signal stays high until read, and then goes low. It is your responsibility to ensure that each sample is read from the output before the next one is available because there is no local buffering. If you do not read out a sample during this period, you won't get a fresh DMA request output for the next sample.

Component Parameters

Drag a Filter component onto your design and double click it to open the **Configure** dialog. Change the **Name** field to adjust the instance name of the Filter.

Note You should only place one Filter component (or any other component that uses the DFB) in a design. There is nothing to prevent you from placing two components, and the customizer will operate on both of them independently, but the design will not build and work correctly if more than one is left on the schematic when the project is built.



Name: Filter_1

✓ Enable channel
☐ Custom coefficients

Data ready signal
- DMA request
- Interrupt request
- [✓ Polled]

Min BUS_CLK: 2.3 MHz
Sample rate Fs (ksps): 100
Filter stages: 1
Filter gain: 0 dB

Filter class: FIR
Filter type: Low pass
Window: Hamming
Filter taps: 16
Cutoff (kHz): 15
Bandwidth (kHz): 10
Coefficient entry:

Filter response graphs
- ✓ Gain
- ☐ Phase
- ☐ Group delay
- ☐ Tone input
- ☐ Tone response
- ☐ Impulse
- ✓ Step

Filter response scaling
Zoom: Log Phase: Wrapped in degrees Frequency: Hz
Gain: dB Group delay: Seconds Time base: Time

Frequency axis scaling: [✓ Log] Linear Upper limit (kHz): 50 Lower limit (kHz): 0
Gain axis scaling: [✓ Log] Linear Upper limit (dB): 0 Lower limit (dB): -140

Filter Parameters

There are two groups of parameters:

- Filter parameters, on the left, determine the filter response.
- Display parameters, on the top right, determine which filter response graphs are displayed, and how they are scaled.

Enable channel

This parameter enables or disables the code generation for the channel. Each channel can only be enabled if there are enough resources for a filter on that channel. If there are not enough resources available to support a change in configuration, a warning icon will appear. Because the tracking of this error state is quite complicated, you may need to click back and forth between different settings several times to clear all of the error flags. Also, the absence of an error flag does not guarantee that there is no error.

Custom coefficients

This parameter enables or disables the custom coefficient entry for a channel. If this parameter is enabled, then custom coefficients can be entered in the **Coefficient entry** text box. These custom coefficients are used for calculating the filter response and for generating the DFB microcode. In Filter 2.0, the custom setting applies to all stages in a filter; that is, it is not possible to mix stages using custom coefficients with stages designed by the component itself. When this parameter is disabled, the customizer calculates the filter coefficients from your requirements, and the **Coefficient entry** box is blank. The final coefficients that result from this design process are aggregated across all four stages for that channel, and presented in the box on the **Final coefficients** tab. These coefficients can be copied out to the clipboard, and pasted into a text file or spreadsheet for further processing. They can be pasted back into the coefficient entry box in custom coefficients mode, enabling internal design filters to be employed in custom designs.

When this parameter is enabled, **Filter stages** parameters, except for **Filter class** and **Filter taps** are invalid, so they are disabled. Disabling this parameter re-enables the **Filter stage** parameters. In custom FIR mode, the customizer interprets each numerical input on a new line as a tap value (zero is accepted) and counts the total number of taps. In custom biquad mode, the customizer expects that the number of entries will be a multiple of five and reports an error if this is not the case. It will also check for the stability of the denominator coefficients of the biquad, and will not accept the coefficients if one or more biquads are unstable.

Data ready signal

This block configures whether you are alerted when data is ready through either a DMA request signal specific to each channel, or through an interrupt request shared between both channels. You can also poll the status register to check for new data on the DFB output. However, the interrupt must be enabled in the INT_CTRL register before starting the DFB operation so that it can be polled here.

See the Registers section for details of INT_CTRL and SR (Status register) registers. The output holding register is double buffered, but you must be sure to take the data from the output before it is overwritten.

If you are running the filter at high real sample rates, it is likely that DMA is the only method that is fast enough to keep up. Because the main CPU is not involved, the DFB can implement complicated filters at a much greater sample rate than is possible with microcontrollers that have DSP extensions to their instruction sets.

Sample rate

The rate entered into the **Sample rate** box (always in ksps) does not affect the operation of the hardware, only the design process for the filter coefficients and the scaling of the graphs used to present the results. It is your responsibility to pump data through the DFB at the desired physical sample rate. The customizer limits the declared sample rate to the range 1 sps to 10 Msps; this is only for your reporting convenience, because the maximum bus_clk rate of a PSoC 3 or PSoC 5 device does not exceed 80 MHz.

Because there is no decimation or interpolation in Filter 2.0, the sample rate is the same for every stage of a multistage filter. The sample rates for the two channels need not be the same, but Filter 2.0's DFB firmware operates most efficiently when the two rates are the same or very similar.

After you configure your filter design inputs, Filter 2.0 calculates the minimum PSoC 3/PSoC 5 bus_clk frequency that the DFB must run from to ensure that all samples are processed correctly. The filter cannot be implemented if this value exceeds the highest available clock rate in your system. No buffering is provided; this means that the Digital Filter Block code must be able to execute *both* filters in the period of time between incoming samples of the faster channel.

If you know the maximum available bus_clk rate in your system, and the count of coefficients and filter poles you will need for each of the two channels, you can calculate the maximum sample rate at which the filter will run. If you are using both channels, this is the rate at which the channel with the higher sample rate will run (or both channels, if they run at the same rate). The maximum possible sample rate is calculated by dividing the fastest available bus_clk by the number of DFB cycles that will be required by both filters. So, for each of the two channels, calculate a cycle count as follows:

FIR only:	10 + number_of_taps
Biquad even total order only:	13 + 5 × order
Biquad odd total order only:	18 + 5 × order
Both biquad even total order and FIR:	20 + number_of_taps + 5 × order
Both biquad odd total order and FIR:	25 + number_of_taps + 5 × order

Add together the results calculated for the two channels. This is the total number of bus_clk cycles that will be required to execute both channels. Dividing this into the bus_clk frequency that you plan to use will give you the value of the sample rate at which the faster of the two channels will run.

Filter stages

Each channel can have up to four different cascaded filter stages; the number is selected with the **Filter stage** drop-down list. If appropriate resources are not available to support adding another stage, a warning notice is issued. The filter response graphs update to display the behavior of the full cascade whenever new parameters are entered. You can access each stage through its own tab.

You can configure the parameters of each filter stage separately. So, for instance, an FIR lowpass filter can be cascaded with a biquad notch filter. The customizer aggregates the coefficients and poles of all four stages in the same way, regardless of the order in which they are entered. All FIR stages are convolved to give a single FIR filter, and all biquad stages are implemented as a cascade. A very sophisticated dynamic range management algorithm inspects the sequence of all the stages and adjusts the order and internal gains in order to produce the highest-quality signal handling from the filter.

Filter gain

This parameter allows you to provide the DC gain for the resultant filtered signal. This parameter takes the gain value both in Linear and decibels. The linear gain should be within the range 0.01 to 100; the corresponding limit for dB gain is –40 dB to 40 dB. The customizer flags an error if you attempt to set a gain outside these limits. You can enter a negative number for the linear gain to invert the response, if required. This sign is not reflected in the dB display.

Note that to guarantee no overflow of output results, it is good practice to run the filter at a gain of less than unity, because most filter responses will exhibit overshoot in the time domain on some signals. The default gain is set to unity, but Cypress recommends that, in a typical design where input signals may reach digital full scale, you select a gain of 0.25x (– 12.04dB). The filter design routines in the customizer are set up so that, if this gain is selected, no internal stage of the filter can overload on any feasible digital input signal.

The Final coefficients Tab

In addition to the tabs for setting up the filter stages, the **Final coefficients** tab contains a window that shows the final coefficients that will be used in the filter. This includes all coefficient rounding, gain scaling, and biquad reordering that the customizer performs to optimize the performance. You can select and copy the contents by right-clicking in this window. You can then paste them into a spreadsheet for further analysis. The Filter Stage: Coefficient entry window section discusses the format used for both input and output of the coefficients for FIR and IIR filters.

Filter Stage: Filter class

The selectable options are **FIR** and **Biquad**. A biquad filter is implemented as a series cascade of second-order filter sections. The first-order sections that appear in odd order transfer functions are implemented as second-order biquads with zero coefficients for z^{-2}.

Filter Stage: Filter type (FIR)

For an FIR filter, the available options are **Low pass**, **High pass**, **Band pass**, **Band stop**, **Sinc^4**, and **Hilbert** filters.

The sinc^4 filter is a special-purpose lowpass filter with a gently rising passband. It is designed to compensate for the drooping frequency response of a delta-sigma ADC that uses a fourth-order sinc decimator (such as the ADC in PSoC 3 and PSoC 5 when running at resolutions of 16 bits or lower).

The Hilbert filter is a special case of a highpass filter that has an additional ninety degrees of phase shift. It is used in some communications signal processing.

For biquad filters, the available options are **Low pass, High pass**, **Band pass**, **and Band stop**.

Filter Stage: Window (FIR)

There are several windowing methods provided when using a FIR filter. The differences between them should be balanced against your needs. Pass band ripple, transition bandwidth, and stop band attenuation are affected differently by each of the windowing methods.

- **Rectangular**: This method represents the absence of a window; the sinc impulse response of the ideal lowpass filter is truncated to zero outside the number of taps used. These filters exhibit large pass band ripple, sharp roll off, and poor stopband attenuation. This window is rarely used because of the large ripple effect from Gibbs Phenomenon.
- **Hamming**: In Filter 2.0, the window used is actually a slightly modified Hamming window due to Albrecht. It exhibits a somewhat flatter and more uniform stopband. This is a good general-purpose FIR filter and is the default choice for a newly placed component.
- **Blackman**: A modified Blackman window (close to the Blackman-Nuttall window and again due to Albrecht). It gives greater stopband attenuation than the Hamming class of window, but has a wider transition band.

Filter Stage: Filter taps (FIR)

This version of the entry box is available for FIR-class filter stages. When using only FIR filters, the total combined size of all filters can be up to 128 taps. The order of an FIR filter is equal to one less than the number of taps.

Filter Stage: Shape (Biquad)

The shapes available when using the biquad filter class in Filter 2.0 are Butterworth, Bessel, and Chebyshev. Note that the Bessel implementation in Filter 2.0 uses a bilinear transform of the classical linear version, and this does not preserve the flat group delay behavior for cutoff frequencies that are a significant fraction of the sampling rate.

Filter Stage: Order (Biquad)

This version of the entry box is available for biquad-class filter stages. Lowpass and highpass filters can be either even or odd, though the implementation is always rounded up to the nearest even number because single poles are implemented with the same biquad topology within the DFB. Bandpass and bandstop filters can only be even order, and an error will be issued if you enter an odd number in this case. Chebyshev and Butterworth filters can have a maximum order of 50. At very high orders, and narrow bandpass or bandstop bandwidths, numerical restrictions in the customizer may produce unexpected results.

You can design Bessel lowpass and highpass filters up to order 25.

The number of memory locations required by a biquad cascade filter of order N is 2.5N if N is even, and 2.5(N + 1) if N is odd. The biquad filter requires three internal memory locations for additional variables, so when using biquad or biquad+FIR, the total combined size of all filters cannot exceed 125 memory locations.

Filter Stage: Cutoff

Enter the edge of the pass band frequencies for Sinc^4, Low Pass, High Pass, and Hilbert filters. The anticipated gain of the filter at this frequency depends on the filter class and type. For FIR lowpass, highpass and Hilbert filters, the response is nominally –6 dB at the entered cutoff frequency. For biquad Bessel and Butterworth filters, the response is –3 dB. For Chebychev filters, the response equals the entered ripple value, with respect to the highest gain in the passband.

The gain response for FIR sinc^4 filters is best assessed through experiment.

Filter Stage: Center Frequency and Bandwidth (Band Pass and Band Stop)

The bandwidth of a bandpass filter is the frequency difference between the upper and lower frequencies that meet the cutoff criterion given earlier for the filter class and type. The center frequency lies between the upper and lower frequencies, but its exact relationship to these frequencies depends on the filter type and class.

For FIR-class filters, the center frequency of either bandpass or bandstop filters is the arithmetic mean of the upper and lower cutoff frequencies. In other words, the response created by the customer is arithmetically symmetrical around the center frequency. An FIR bandpass or bandstop filter with center frequency of 10 kHz and bandwidth of 10 kHz will have –6-dB points at 5 kHz and 15 kHz.

The type of bandpass and bandstop transformations used in Filter 2.0 give a geometrically symmetrical frequency response. Biquad-class bandpass filters are defined by their upper and lower cutoff frequencies, and these are positioned to be *arithmetically* symmetric around the user-entered center frequency, to be consistent with the FIR case. So, if you enter a center frequency of 10 kHz and bandwidth of 10 kHz for a Butterworth biquad bandpass filter, you will get –3-dB points at 5 kHz and 15 kHz. The ‘true’ center frequency of such a filter is not at 10 kHz; in this case it is at SQRT(5 kHz × 15 kHz) = 8.66 kHz.

Biquad-class bandstop filters are designed to have their maximum attenuation at the entered center frequency. This means that the passband cutoff frequencies cannot be positioned to match the FIR case. To calculate the cutoff frequencies you actually get requires a little calculation:

$F_{lower} = SQRT(0.25 \times BW^2 + F_{center}{}^2) – 0.5 \times BW$

and of course $F_{upper} = F_{lower} + BW$

In the example with center frequency of 10 kHz and bandwidth of 10k Hz, this calculates to F_{lower} = 6.18 kHz and F_{upper} = 16.18 kHz.

Filter Stage: Ripple (IIR Chebyshev)

This parameter is only valid for a biquad-class Chebyshev filter. It determines the theoretical passband ripple of the filter. The allowable range is 0.00001 dB to 3 dB.

Filter Stage: Coefficient entry window

Use this text box to enter the custom coefficients into the selected filter stage when the **Custom Coefficients** check box is selected; see Custom coefficients.

Enter the coefficients as floating point values on sequential lines. White space is ignored, but zero is a valid entry. Nonnumeric entries are not accepted in Filter 2.0.

For a biquad filter, enter three numerator coefficients first (for z^0, z^{-1}, and z^{-2}) followed by two denominator coefficients (it is assumed that the z^0 denominator coefficient is unity). The entered coefficients are validated and if any coefficients are found to be invalid (or the total number is not divisible by five), an error indicator is displayed. The entered denominator coefficients are also tested for stability. If any biquad is found to be unstable, a warning is displayed. The biquad coefficients are applied to the same gain adjustment and sequencing algorithm that is used on internally generated coefficients, so the implementation order may not be the same as entered. Also, the peak gain of the filter in the passband will be adjusted to be the value entered in the user gain box. You can enter arbitrary values of numerator scaling and the algorithm will scale them appropriately. If you are using Filter 2.0 to implement PID or other control loops, you will need to calculate your own gain value to put in the gain box to ensure that the overall gain is what you need.

For an FIR filter, the first value entered is treated as the coefficient of z^0, the undelayed tap. For FIR-only filters, no separate gain adjustment algorithm is applied. In the FIR case, the maximum allowable range of coefficient value is –1 to 1-2-23. You can put in values outside of that range but to do that you must have a Filter Gain value that will scale those numbers down so that they are within that range. This gain value is given if you enter an invalid coefficient.

When FIR and biquad filters are combined, gain scaling is automatically applied to the FIR portion, which is implemented before the biquads are executed. You can view the scaled results of the entire cascade viewed in the **Final coefficients** tab. From there, you can paste them into another application for further analysis if required.

Display Parameters

Display parameters only affect the way the filter response is presented to you in the configuration window. They have no effect on the code generation or filter settings.

The setup parameter selections can be shown or hidden with the graph configurations button at the top of this section; this can make the graphs easier to read. Note than when many panes are shown in the graph area, the plot axes may not be numbered as expected because of limitations in the automatic plot routines.

The plots are divided into two subplot areas, for frequency and time parameters. Right-clicking on either subplot copies that side to the clipboard in bitmap format so that you can paste it into your own reports.

Filter response graphs

Gain – When enabled, displays the amplitude of the overall filter response over frequency.

Phase – When enabled, displays the phase shift of the overall filter response over frequency.

Group delay – When enabled, displays the group delay of the overall filter response over frequency.

Tone input – When enabled, displays a sinewave signal at the center or cutoff frequency of the filter, to be used as the input to the filter. If multiple filter stages are used, the tone is equal to the average center or cutoff frequency. In Filter 2.0, the frequency of this sinewave cannot be separately set.

Tone response – When enabled, displays the filter's response to the predetermined Tone Input Wave (see Tone input).

Impulse – When enabled, displays the filter's response to a positive-going single-sample impulse.

Step – When enabled, displays the filter's response to a positive-going unit step function.

Filter response scaling: Zoom

The available options are **Log**, **Linear**, and **Custom**.

The log zoom option provides a view of the filter's responses with a logarithmic frequency scale from DC to the Nyquist frequency.

The linear option provides a linear frequency scale of the pass band frequency from DC to the Nyquist frequency.

Selecting the custom option enables the settings for inputting the maximum and minimum limits of gain and frequency. In the Custom view, you can define your own limits to the frequency and gain axis. In this mode, the frequency and gain axis can be set in both linear mode and logarithmic mode.

Filter response scaling: Gain

Selecting **dB** gain displays the gain values in decibels for gain response. Selecting **Linear** displays the gain values in linear scale.

Filter response scaling: Phase

This parameter allows you to select whether you will view wrapped or unwrapped phase expressed in radians or degrees. The available options are **Unwrapped in degrees**, **Wrapped in degrees**, **Unwrapped in radians**, and **Wrapped in radians**. Note that some filters with transmission zeroes (for example, bandstop filters) will have discontinuous phase response plots even when you select an unwrapped phase.

Filter response scaling: Group delay

This parameter allows you to select Group Delay response as a time in microseconds or as a number of samples.

Filter response scaling: Frequency

This parameter allows you to select Frequency response x-axis as a value in kHz or a number of samples.

Filter response scaling: Time base

This parameter allows you to select Time response x-axis as a value in milliseconds or in sample counts.

Frequency axis scaling: Log/Linear selection

This option is valid only when **Custom** zoom is selected. Selecting **Log** sets the Frequency response x-axis in log scale. Selecting **Linear** sets the x-axis in linear scale.

Frequency axis scaling: Upper limit

This option is valid only when **Custom** zoom is selected. This sets the upper limit for the x-axis of the frequency response graph. The maximum upper limit should be less than or equal to half of the sample rate.

Frequency axis scaling: Lower limit

This option is valid only when **Custom** zoom is selected. This sets the lower limit for the x-axis of the frequency response graph. The lower limit should be less than the upper limit value and should not be less than zero.

Gain axis scaling: Log/Linear selection

This option is valid only when **Custom** zoom is selected. Selecting **Log** sets the Gain axis in log scale. Selecting **Linear** sets the gain axis in linear scale.

Gain axis scaling: Upper limit

This option is valid only when **Custom** zoom is selected. It sets the upper limit for gain response in either dB or linear scale based on whether you selected the **Log** or **Linear** button.

Gain axis scaling: Lower limit

This option is valid only when **Custom** zoom is selected. It sets the lower limit for gain response in either dB or linear scale based on whether you selected the **Log** or **Linear** button. The lower limit should be less than the upper limit value.

Placement

The Filter 2.0 component consumes the entire DFB, so a working project can contain only one placed Filter (or any other component that requires the DFB). If you place multiple Filter components, each can be set up with its own customizer and the properties will be saved. This allows initial schematics to contain multiple filters as a way of saving setups, before you build the project.

Resources

	Digital Blocks					API Memory (Bytes)		
Analog Blocks	**Datapaths**	**Macro cells**	**Status Registers**	**Control Registers**	**Counter7**	**Flash**	**RAM**	**Pins (per External I/O)**
N/A	N/A	N/A	N/A	N/A	N/A	3116	69	1

The Filter 2.0 component consumes no other resources than the DFB. To achieve maximum throughput, you will probably need to use DMA for data management.

Application Programming Interface

Application Programming Interface (API) routines allow you to interact with the component using software. The following table lists and describes the interface to each function together with related constants provided by the "include" files. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name "Filter_1" to the first instance of a component in a given project. You can rename it 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 "Filter."

Functions	Description
Filter_Start()	Configures and enables the Filter component's hardware for interrupt, DMA and filter settings.
Filter_Stop()	Stops the filters from running and powers down the hardware.
Filter_Read8()	Reads the current value on the Filter's output holding register. Byte read of the most significant byte.
Filter_Read16()	Reads the current value on the Filter's output holding register. Two-byte read of the most significant bytes.
Filter_Read24()	Reads the current value on the Filter's output holding register. Three-byte read of the data output holding register.
Filter_Write8()	Writes a new 8-bit sample to the Filter's input staging register.
Filter_Write16()	Writes a new 16-bit sample to the Filter's input staging register.
Filter_Write24()	Writes a new 24-bit sample to the Filter's input staging register.
Filter_ClearInterruptSource()	Writes the Filter_ALL_INTR mask to the status register to clear any active interrupts.
Filter_IsInterruptChannelA()	Identifies whether Channel A has triggered a data-ready interrupt.
Filter_IsInterruptChannelB()	Identifies whether Channel B has triggered a data-ready interrupt.
Filter_Sleep()	Stops and saves the user configuration.
Filter_Wakeup()	Restores and enables the user configuration.
Filter_Init()	Initializes or restores default Filter configuration.
Filter_Enable()	Enables the Filter.
Filter_SaveConfig()	Saves the configuration of Filter nonretention registers.
Filter_RestoreConfig()	Restores the configuration of Filter nonretention registers.
Filter_SetCoherency()	Sets the key coherency byte in the coherency register.

Global Variables

Variable	Description
Filter_initVar	Indicates whether the Filter has been initialized. The variable is initialized to 0 and set to 1 the first time Filter_Start() is called. This allows the component to restart without reinitialization after the first call to the Filter_Start() routine. If reinitialization of the component is required, then the Filter_Init() function can be called before the Filter_Start() or Filter_Enable() functions.

Defines

- **Filter_CHANNEL_x** – Filter_CHANNEL_A or Filter_CHANNEL_B. Use when specifying which channel an operation occurs on for function calls.
- **Filter_CHANNEL_x_INTR** – Mask for the CHANNEL_A or CHANNEL_B interrupt of the Status Register.
- **Filter_ALL_INTR** – Mask for the possible interrupts of the Status Register.

void Filter_Start(void)

Description:	This is the preferred method to begin component operation. Configures and enables the Filter component's hardware for interrupt, DMA, and filter settings.
Parameters:	None
Return Value:	None
Side Effects:	None

void Filter_Stop(void)

Description:	Stops the Filter hardware from running and powers it down.
Parameters:	None
Return Value:	None
Side Effects:	None

uint8 Filter_Read8(uint8 channel)

Description:	Reads the highest order byte of Channel A's or Channel B's output holding register.
Parameters:	uint8 channel: Which filter channel should be read. Options are Filter_CHANNEL_A and Filter_CHANNEL_B.
Return Value:	8-bit filter output value
Side Effects:	None

uint16 Filter_Read16(uint8 channel)

Description:	Reads the two highest-order bytes of Channel A's or Channel B's output holding register.
Parameters:	uint8 channel: Which filter channel should be read. Options are Filter_CHANNEL_A and Filter_CHANNEL_B.
Return Value:	16-bit filter output value
Side Effects:	None

uint32 Filter_Read24(uint8 channel)

Description:	Reads all three bytes of Channel A's or Channel B's output holding register.
Parameters:	uint8 channel: Which filter channel should be read. Options are Filter_CHANNEL_A and Filter_CHANNEL_B.
Return Value:	Unsigned 32-bit representation of the sign extended 24-bit output value
Side Effects:	None

void Filter_Write8(uint8 channel, uint8 sample)

Description:	Writes to the highest-order byte of Channel A's or Channel B's input staging register.
Parameters:	uint8 channel: Which filter channel should be read. Options are Filter_CHANNEL_A and Filter_CHANNEL_B. uint8 sample: Value to be written to the input register
Return Value:	None
Side Effects:	This function writes only the most significant byte. This could result in noise being added to the input samples if the lowest-order bytes have not been set to zero.

void Filter_Write16(uint8 channel, uint16 sample)

Description:	Writes to the two highest-order bytes of Channel A's or Channel B's input staging register.
Parameters:	uint8 channel: Which filter channel should be read. Options are Filter_CHANNEL_A and Filter_CHANNEL_B. uint16 sample: Value to be written to the input register
Return Value:	None
Side Effects:	None

void Filter_Write24(uint8 channel, uint32 sample)

Description:	Writes to all three bytes of Channel A's or Channel B's input staging register.
Parameters:	uint8 channel: Which filter channel should be read. Options are Filter_CHANNEL_A and Filter_CHANNEL_B. uint32 sample: Value to be written to the input register
Return Value:	None
Side Effects:	None

void Filter_ClearInterruptSource(void)

Description:	Writes the Filter_ALL_INTR mask to the status register to clear any active interrupt. See the earlier Defines section for the definition of this mask.
Parameters:	None
Return Value:	None
Side Effects:	None

uint8 Filter_IsInterruptChannelA(void)

Description:	Identifies whether Channel A has triggered a data-ready interrupt.
Parameters:	None
Return Value:	0 if no interrupt on Channel A; Positive value otherwise.
Side Effects:	None

uint8 Filter_IsInterruptChannelB(void)

Description:	Identifies whether Channel B has triggered a data-ready interrupt.
Parameters:	None
Return Value:	0 if no interrupt on Channel B; positive value otherwise.
Side Effects:	None

void Filter_SetCoherency(uint8 channel, unit8 byte_select)

Description:	Sets the value in the DFB coherency register. This value determines the key coherency byte. The key coherency byte is the software's way of telling the hardware which byte of the field will be written or read last when an update to the field is desired.
Parameters:	uint8 channel: The options are Filter_Channel_A and Filter_Channel_B. uint8 byte_select: Determines which of the three bytes to be set as key coherency byte. Options are High byte, Med byte and Low byte.
Return Value:	None.
Side Effects:	By default, High byte is the key coherency byte. Using this API to change the default behavior, and then using the Filter_Read() and Filter_Write() APIs mentioned earlier, may cause unexpected behavior.

void Filter_Sleep(void)

Description:	Stops the DFB operation. Saves the configuration registers and the component enable state. Should be called just before entering sleep.
Parameters:	None
Return Value:	None
Side Effects:	Filter output registers are nonretention and they will not be saved while going to sleep. So before going to sleep, make sure that there are no pending conversions.

void Filter_Wakeup(void)

Description:	This is the preferred API to restore the component to the state when Filter_Sleep() was called. The Filter_Wakeup() function calls the Filter_RestoreConfig() function to restore the configuration. If the component was enabled before the Filter_Sleep() function was called, the Filter_Wakeup() function will also re-enable the component.
Parameters:	None
Return Value:	None
Side Effects:	Calling the Filter_Wakeup() function without first calling the Filter_Sleep() or Filter_SaveConfig() function may produce unexpected behavior.

void Filter_Init(void)

Description:	Initializes or restores the component according to the customizer Configure dialog settings. It is not necessary to call Filter_Init() because the Filter_Start() API calls this function and is the preferred method to begin component operation.
Parameters:	None
Return Value:	None
Side Effects:	All registers will be reset to their initial values. This reinitializes the component.

void Filter_Enable(void)

Description:	Activates the hardware and begins component operation. It is not necessary to call Filter_Enable() because the Filter_Start() API calls this function, which is the preferred method to begin component operation.
Parameters:	None
Return Value:	None
Side Effects:	None

void Filter_SaveConfig(void)

Description:	This function saves the component configuration and nonretention registers. It also saves the current component parameter values, as defined in the Configure dialog or as modified by appropriate APIs. This function is called by the Filter_Sleep() function.
Parameters:	None
Return Value:	None
Side Effects:	None

void Filter_RestoreConfig(void)

Description:	This function restores the component configuration and nonretention registers. It also restores the component parameter values to what they were before calling the Filter_Sleep() function.
Parameters:	None
Return Value:	None
Side Effects:	Calling this function without previously calling Filter_SaveConfig() or Filter_Sleep() produces unexpected behavior

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 Filter component generates the necessary code for the DFB's coprocessor and configures the filter component. Multistage filters are mathematically combined into a single filter through convolution, resulting in one larger filter for each channel. Future modes will include support for IIR-FIR streams through each channel.

DMA

The DMA component can be used to transfer converted results from Filter output registers to the RAM. The DMA component can also be used to transfer input sample values to Filter input registers. The Filter data ready signal should be connected to the data request signal of the DMA. The DMA Wizard can be used to configure DMA operation as shown in the following table.

Filter Resolution	Name of DMA Source/Destination in DMA Wizard	Direction	DMA Req Signal	DMA Req Type	Description
8-bit	Filter_HOLDBH_PTR	source	DMA_Req_B	Level	Receives 8-bit filtered output values from the filter output register Filter_HOLDB
	Filter_HOLDAH_PTR	source	DMA_Req_A	Level	Receives 8-bit filtered output values from the filter output register Filter_HOLDA
	Filter_STAGEAH_PTR	destination	DMA_Rea_A or N/A	Level or N/A	Receives 8-bit input sample values from RAM to filter input register Filter_STAGEA
	Filter_STAGEBH_PTR	destination	DMA_Req_B or N/A	Level or N/A	Receives 8-bit input sample values from RAM to filter input register Filter_STAGEB
16-bit	Filter_HOLDB_PTR	source	DMA_Req_B	Level	Receives 16-bit filtered output values from the filter output register Filter_HOLDB
	Filter_HOLDA_PTR	source	DMA_Req_A	Level	Receives 16-bit filtered output values from the filter output register Filter_HOLDA

Filter Resolution	Name of DMA Source/Destination in DMA Wizard	Direction	DMA Req Signal	DMA Req Type	Description
	Filter_STAGEA_PTR	destination	DMA_Rea_A or N/A	Level or N/A	Receives 16-bit input sample values from RAM to filter input register Filter_STAGEA.
	Filter_STAGEB_PTR	destination	DMA_Req_B or N/A	Level or N/A	Receives 16-bit input sample values from RAM to filter input register Filter_STAGEB
24-bit	Filter_HOLDB_PTR	source	DMA_Req_B	Level	Receives 24-bit filtered output values from the filter output register Filter_HOLDB
	Filter_HOLDA_PTR	source	DMA_Req_A	Level	Receives 24-bit filtered output values from the filter output register Filter_HOLDA
	Filter_STAGEA_PTR	destination	DMA_Rea_A or N/A	Level or N/A	Receives 24-bit input sample values from RAM to filter input register Filter_STAGEA
	Filter_STAGEB_PTR	destination	DMA_Req_B or N/A	Level or N/Aw	Receives 24-bit input sample values from RAM to filter input register Filter_STAGEB

Registers

Staging

Each of the Filter component’s two channels has a 24-bit dedicated input staging register. When not processing data, the Filter enters a wait state where it waits for one of these to registers to be written before starting a new pass through the filter design.

Holding

After processing input data, the latest output sample is placed in the 24-bit output holding register. There are three options regarding system notification of a ready output sample: Interrupt, DMA request, or Polling.

Data Align (Filter_DALIGN) Register

The DFB requires that input data is MSB aligned, and the delivered output results are also MSB aligned. The DFB hardware provides a data alignment feature in the input Staging registers and in the output Holding registers for convenience to the system software.

Both Staging and both Holding registers support byte accesses, which addresses alignment issues for input and output samples of eights bits or less. Likewise, all four of these registers are mapped as 32-bit registers (only three of the four bytes are used) so there are no alignment issues for samples between 17 and 24 bits. However, for sample sizes between 9 and 16 it is convenient to be able to read/write these samples on bus bits 15:0 while they source and sink on bits 23:8 of the Holding/Staging registers.

The bits for the Data Align register provide an alignment feature that allows System bus bits 15:0 to either be source from Holding register bits 23:8 or sink to Staging register bits 23:8. Each Staging and Holding register can be configured individually with a bit in the DALIGN register. If the bit is set high, the effective byte shift occurs.

Example If an output sample from the delta-sigma ADC is 12 bits wide and aligned to bit 23 of the ADC's Output Sample register, and you want to stream this value to the DFB, the data alignment feature of the ADC can be enabled. This allows the 16 bits of the ADC's Output Sample register to be read on bus bits 15:0. Setting the alignment feature in the DFB for the Staging A input register, this 16 bits can be written on bus bits 15:0, but can also be written into bits 23:8 of the Staging A register when required.

Coherency (Filter_COHER) Register

Coherency refers to the hardware included in the DFB to protect against malfunctions of the block in cases where register fields are wider than the bus access. This case can leave intervals in time when fields are partially written/read (incoherent).

Coherency checking is an option that is enabled in the COHER register. The hardware provides coherency checking on both Staging and Holding registers.

The Staging registers are protected on writes so that the underlying hardware does not use the field when it is only partially updated by the system software. The Holding registers are protected on reads so that the underlying hardware does not update the field when it is partially read by the system software or DMA. Depending on the configuration of the block, not all bytes of the Staging and Holding registers may be needed. The coherency method allows for any size output field and handles it properly.

The bit fields of this register are used to select which of the three bytes of the Staging and Holding registers will be used as the Key Coherency byte. In the COHER register, coherency is enabled and a Key Coherency Byte is selected. The Key Coherency Byte is the user (software) telling the hardware which byte of the field will be written or read last when an update to the field is desired.

Filter Register and DMA Wizard Settings

Filter Resolution	Bytes per Burst *	Filter_COHER and Filter_DALIGN Register Settings	Endian Flag *	
			PSoC 3	PSoC 5
8-bit	1	Filter_COHER_REG = 0xAA	OFF	OFF
		Filter_DALIGN_REG = 0x00		
16-bit	2	Filter_COHER_REG = 0x00	TD_SWAP_EN | TD_INC_DST_ADR	OFF
		Filter_DALIGN_REG = 0x0F		
24-bit	4	Filter_COHER_REG = 0xAA	TD_SWAP_EN | TD_SWAP_SIZE4 | TD_INC_DST_ADR	TD_SWAP_SIZE4
		Filter_DALIGN_REG = 0x00		

* To be set in the DMA wizard tool.

Filter_SR_REG

Filter Status Register. Read this to get the sources of the interrupt. Use the Filter_ClearInterruptSource() macro to clear it.

Bits	7	6	5	4	3	2	1	0
Value	INTR SEM2	INTR SEM1	INTR SEM0	INTR HOLDING REG B	INTR HOLDING REG A	RND MODE	SAT MODE	RAM SEL

This register contains five bits indicating the status of block-generated interrupts and three bits of status from the Datapath unit.

Note If the system software wants to poll for an event and not have an interrupt generated, the interrupt must be enabled in the INT_CTRL register so that it can be polled here.

- Bit 7: Semaphore 2 Interrupt – If this bit is high, semaphore register bit 2 is the source of the current interrupt. Write a '1' to this bit to clear it.
- Bit 6: Semaphore 1 Interrupt – If this bit is high, semaphore register bit 1 is the source of the current interrupt. Write a '1' to this bit to clear it.
- Bit 5: Semaphore 0 Interrupt – If this bit is high, semaphore register bit 0 is the source of the current interrupt. Write a '1' to this bit to clear it.
- Bit 4: Holding Register B Interrupt – If this bit is high, Holding register B is the source of the current interrupt. Write a '1' to this bit to clear it. Reading the Holding register B also clears this bit.
- Bit 3: Holding Register A Interrupt – If this bit is high, Holding register A is the source of the current interrupt. Write a '1' to this bit to clear it. Reading the Holding register A also clears this bit.
- Bit 2: Round Mode – Indicates that the DP is in Round mode. This means that any result passing out of the DP unit is being rounded to a 16-bit value.
- Bit 1: Saturation Mode – Indicates that the DP unit is in Saturation mode. This means that executing any mathematic operation that produces a number outside the range of a 24-bit 2's complement number is clamped to the mode positive or negative number allowed. Saturation mode is set or unset under Assembly control in the DFB Controller.
- Bit 0: RAM Select – Shows which CS RAM is in use.

Filter_INT_CTRL_REG

This register controls which events generate an interrupt. The events enabled by the bits in this register are ORed together to produce the dfb_intr signal.

Bits	7	6	5	4	3	2	1	0
Value	resvd	resvd	resvd	EN SEM2	EN SEM1	EN SEM0	EN HOLDING REG B	EN HOLDING REG A

If you want to use the polling method, enable either bit 0 or 1 of this register, based on the Filter channel selected. This generates an interrupt when data is ready in Filter output registers. Corresponding status bits are set in the Status register. Firmware can poll the corresponding bits in the status register to read the Filter output data.

- Bit 7 to 5: Reserved
- Bit 4: ENABLE Semaphore 2 – If this bit is set high, an interrupt is generated each time a ‘1’ is written into the semaphore register bit 2.
- Bit 3: ENABLE Semaphore 1 – If this bit is set high, an interrupt is generated each time a ‘1’ is written into the semaphore register bit 1.
- Bit 2: ENABLE Semaphore 0 – If this bit is set high, an interrupt is generated each time a ‘1’ is written into the semaphore register bit 0.
- Bit 1: ENABLE HOLDING Register B – If this bit is set high, an interrupt is generated each time new valid data is written into the output Holding register B.
- Bit 0: ENABLE HOLDING Register A – If this bit is set high, an interrupt is generated each time new valid data is written into the output Holding register A.

DC and AC Electrical Characteristics for PSoC 3

Filter DC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
	DFB operating current	64-tap FIR at F_{DFB}				
		100 kHz (1.3 ksps)	--	0.03	0.05	mA
		500 kHz (6.7 ksps)	--	0.16	0.27	mA
		1 MHz (13.4 ksps)	--	0.33	0.53	mA
		10 MHz (134 ksps)	--	3.3	5.3	mA
		48 MHz (644 ksps)	--	15.7	25.5	mA
		67 MHz (900 ksps)	--	21.8	35.6	mA

Filter AC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
F_{DFB}	DFB operating frequency		DC	--	67	MHz

DC and AC Electrical Characteristics for PSoC 5

Filter DC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
	DFB operating current	64-tap FIR at F_{DFB}				
		100 kHz (1.3 ksps)	--	0.03	0.075	mA
		500 kHz (6.7 ksps)	--	0.16	0.3	mA
		1 MHz (13.4 ksps)	--	0.33	0.57	mA
		10 MHz (134 ksps)	--	3.3	5.5	mA
		48 MHz (644 ksps)	--	15.7	26	mA
		67 MHz (900 ksps)	--	21.8	35.6	mA

Filter AC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
F_{DFB}	DFB operating frequency		DC	--	67	MHz

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
2.0	Support for IIR filter	IIR filter was not supported in older version
	Redesign of FIR filter.	There were issues with windowed version of FIR filter in the old version.
	Filter DMA wizard related updates	Filter can be set as destination for data transfer using DMA without setting the DMA as the data ready signal.
	Added a display field in the configure window to display the required bus clock frequency based on the filter parameter settings	To know the bus clock requirement based on the parameter selection.
	Filter configure window updates	To provide an option for entering custom coefficients and also improvements for good look and feel.
1.50.a	Added characterization data to the datasheet	
	Minor datasheet edits and updates	
1.50	Added Sleep/Wakeup and Init/Enable APIs.	To support low power modes, as well as to provide common interfaces to separate control of initialization and enabling of most components.
	Added DMA capabilities file to the component.	This file allows the Filter to be supported by the DMA Wizard tool in PSoC Creator.