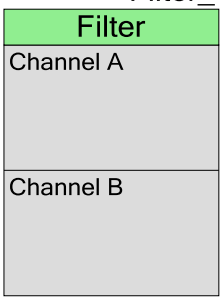
Filter
1.20

Features

Filter_1



- Allows easy user configuration of the Digital Filter Co-processor
- Two separate filtering channels
- Multiple windowing methods to tune filter responses
- Up to 4 cascaded filters in a single channel

General Description

The Filter component allows the configuration of filters on one or two digital data streams passing data in and out using either DMA or register writes through firmware. The Filter provides 128 taps to be shared between the two channels. The number of taps determines the frequency response of the filter.

When to use a Filter

Simple filter applications can be used in many different ways. Anytime frequency characteristics of a signal need to be modified a filter is used. Typical filter applications remove unwanted signals from the input.

Input/Output Connections

This section describes the various input and output connections for the Filter. An asterisk (*) in the list of I/O's states that the I/O may be hidden on the symbol under the conditions listed in the description of that I/O.

Interrupt *

If either channel is configured to generate an interrupt in response to a data-ready event, the interrupt output will be 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 *

If either channel is configured to generate a DMA request in response to a data-ready event, the DMA Request output will be 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.

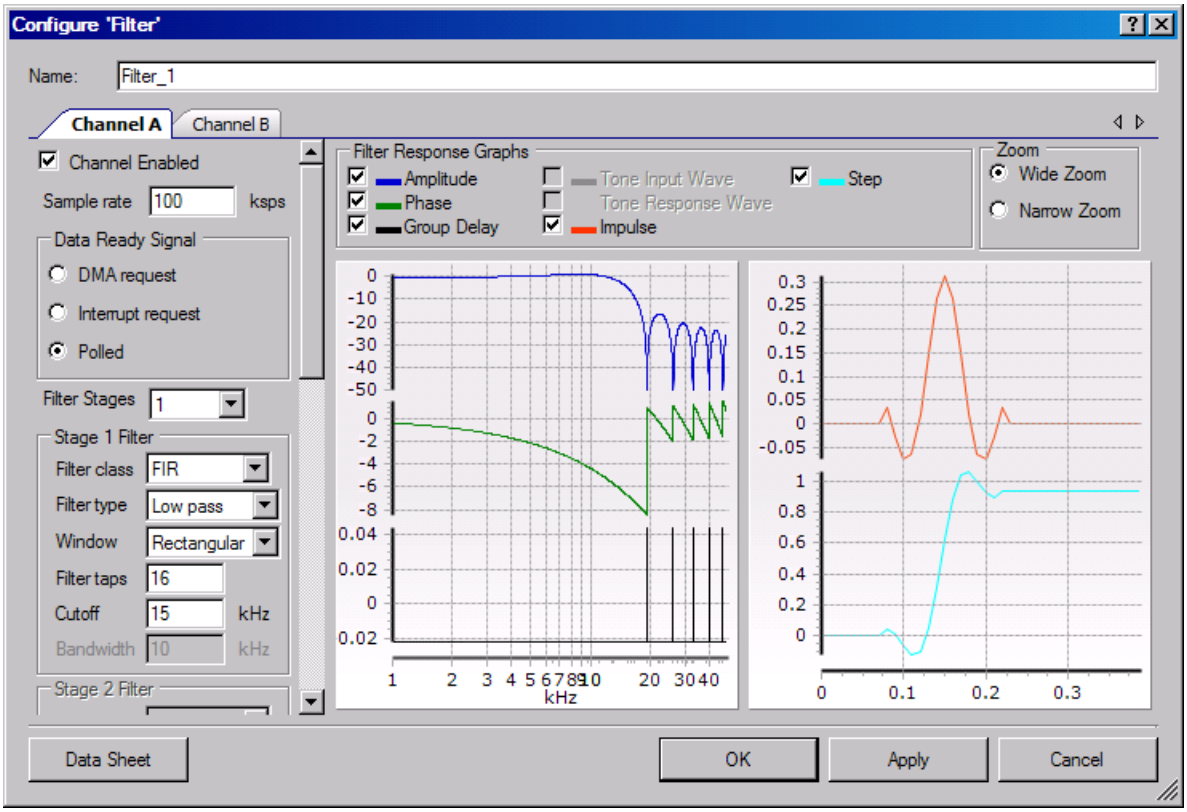
If the DMA out signal is routed to a counter, the output signal is high until read (hold register). This means you will only have a clock pulse when the register is read, whereas you may believe the counter will be clocked every time the filter finishes. However, this is only true if the holding register is also read after each filter cycle.

Parameters and Setup

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 Only one filter component should be placed in a design. There is nothing to prevent placing two components but the design will not build and work correctly.

Figure 1 Configure Filter Component: Channel A



Filter Parameters

There are two groups of parameters. Filter Parameters, on left, determine filter response. Display parameters, on the top right, determine which filter response graphs are displayed.

Channel Enabled

This enables or disables the code generation for the channel. A channel can only be enabled if sufficient resources exist for a filter on that channel; If there are currently not enough resources available a warning icon will appear, informing you of the issue.

Sample Rate

The sample rate has two caveats attached to it:

1. This is the design sample rate. The operational sample rate is determined by your data source for the filter.
2. Because there are no decimation or interpolation stages, the sample rate applies to every stage of the channel.

Consideration of Nyquist sampling theory and bus clock should be applied when selecting a sample rate. The maximum sample rate of a single filter channel is given by

$$f_{sMax} = \frac{Clk_{bus}}{9 + ChannelDepth}$$

Where Clk_{bus} is the bus clock speed, and Channel Depth is the total sum of taps for a single channel.

When both channels are used the equation changes to:

$$f_{sMax} = \frac{Clk_{bus}}{19 + ChannelDepth_A + ChannelDepth_B}$$

Data Ready Signal

The block can alert you of ready data through either a DMA request signal specific to each channel, or through an interrupt request shared between both channels. Finally, you have the option to poll the status register to check for new data on the DFB output. The output holding register is double buffered, but care must be taken to take the data from the output before it is over written.

Filter Stages

Each channel can, depending on resource availability, have up to 4 different cascaded filters. If appropriate resources are not available, a warning notice will be issued next to the Filter Stage selection box. The filter response graphs will update to display the behavior of the full cascade. Each filter stage has a set of parameters which affect its behavior.

Filter Stage: Filter Class

Currently, only FIR filters are selectable. In the future IIR filter generation will be available as well.

Filter Stage: Filter Type

Choose between Low Pass, High Pass, Band Pass, and Band Stop and Sinc4 compensation filters.

Filter Stage: Window (FIR)

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

Window types:

- Rectangular: Large pass band ripple, sharp roll off and poor stop band attenuation. This window is rarely used because of the large ripple effect from Gibbs Phenomenon. Each of the following windows smooth out the rectangular window to reduce the discontinuity and the effect of the phenomenon.
- Hamming: Smoothed pass band, wider transition band, better stop band attenuation than Rectangular.
- Gaussian: Wider transition band, but greater stop band attenuation and smaller stop band lobes than Hamming.
- Blackman: Steeper roll off than Gaussian window, similar stop band attenuation, but larger lobes in the stop band.

Filter Stage: Taps (Order)

Specify the size of the filter. When using only FIR filters the total combined size of all filters cannot exceed 128 taps.

Filter Stage: Cutoff Frequency

Specify the edge of the pass band frequencies for Sinc4, Low Pass and High Pass filters.

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

This is the *arithmetic* mean of the band pass or band stop filters. That is the center frequency is defined by the equation:

$$f_c = \frac{f_u + f_l}{2}$$

Where f_u is the upper cutoff frequency and f_l is the lower cutoff frequency.

Filter Stage: Bandwidth (Band Pass and Band Stop)

Bandwidth is defined as the difference between the upper cutoff frequency and the lower cutoff frequency:

$$f_U - f_L$$

The bandwidth will be evenly split on either side of the arithmetic center frequency defined above.

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.

Frequency Response

Displays the amplitude of the filter cascade's response over frequency

Phase Delay

Displays the phase delay of the filter cascade's response over frequency

Group Delay

The filter cascade's phase delay differentiated over frequency.

Zoom

Wide zoom provides a view of the filter's responses with a logarithmic frequency scale from DC to the Nyquist frequency. Narrow zoom provides a linear frequency scale of the pass band frequencies.

Step Response

Enables or disables the display of the filter cascade's response to a positive step function.

Impulse Response

Enables or disables the display of the filter cascade's response to an input of the unit impulse.

Tone Input Wave

The tone input wave and its response are only available when a band pass filter is selected in the channel's cascade of filters. When selected, it displays the tone input wave at the center

frequency of the band pass. If multiple band pass filters are used the tone is equal to the average center frequency.

Tone Response Wave

Enable the display of the cascade's response to the Tone Input Wave. (See Tone Input Wave above)

Clock Selection

Not Applicable.

Placement

A single Filter component can be placed at once as it consumes the entire DFB.

Note Only one filter component should be placed in a design. There is nothing to prevent placing two components but the design will not build and work correctly.

Resources

One Digital Filter Block.

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(uint8 channel)	Read the current value on the Filter's output holding register. Byte read of the most significant byte.

Functions	Description
Filter_Read16(uint8 channel)	Read the current value on the Filter's output holding register. 2-byte read of the most significant bytes.
Filter_Read32(uint8 channel)	Read the current value on the Filter's output holding register. 3-byte read of the data output holding register.
Filter_Write8(uint8 channel, uint8 sample)	Write a new 8-bit sample to the Filter's input staging register.
Filter_Write16(uint8 channel, uint16 sample)	Write a new 16-bit sample to the Filter's input staging register.
Filter_Write24(uint8 channel, uint32 sample)	Write a new 24-bit sample to the Filter's input staging register.
void Filter_ClearInterruptSource(void)	Writes the Filter_ALL_INTR mask to the status register to clear any active interrupts.
uint8 Filter_IsInterruptChannelA(void)	Checks if Channel A has triggered a data-ready interrupt.
uint8 Filter_IsInterruptChannelB(void)	Checks if Channel B has triggered a data-ready interrupt.

void Filter_Start(void)

Description: 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 only writes 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 the 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.

Parameters: None

Return Value: None

Side Effects: None

uint8 Filter_IsInterruptChannelA(void)

Description: Checks if 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:	Checks if 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

Defines

Filter_CHANNEL_x

(Filter_CHANNEL_A or Filter_CHANNEL_B) Use when specifying which channel an operation occurs on for function calls.

Filter_SR

Status Register. Read this to get the source(s) of the interrupt; Use the Filter_ClearInterruptSource() macro to clear it.

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.

Sample Firmware Source Code

To be provided in Application Note.

Interrupt Service Routine

Not Applicable.

Functional Description

The Filter component generates the necessary code for the Digital Filter Block's (DFB) coprocessor and configures the filter component. Multi-stage 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.

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

DFB inherently requires that input data be MSB aligned and 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 SW.

Both Staging and both Holding registers support byte accesses which addresses alignment issues for input/output samples of 8 bits or less. Likewise, all 4 of these registers are mapped as 32-bits registers (only 3 of the 4 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 Data Align register provides 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 Decimator is 12-bit wide and aligned to bit 23 of the Decimator Output Sample register and it is desired to stream this value to the DFB, the like data alignment feature of the Decimator can be enabled which allows the 16 bits of the Decimator 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 will be written into bits 23:8 of the Staging A register – where required.

Coherency (Filter_COHER) Register

Coherency refers to the HW added to DFB block to protect against malfunctions of the block in cases where register fields are wider than the bus access, leaving intervals in time when fields are partially written/read (incoherent).

Coherency checking is an option and is enabled in the COHER register. The HW provides coherency checking on both STAGING and HOLDING registers

The Staging registers are protected on writes – so the underlying HW doesn't incorrectly use the field when it is partially updated by the System SW. The Holding registers are protected on reads so that the underlying HW doesn't update it when partially read by the System SW or DMA. Depending on the configuration of the block, not all bytes of the Staging/Holding registers may be needed. The coherency methodology allows for any size output field and handles it properly.

The bit fields of this register are used to select which of the 3 bytes of the STAGING and HOLDING registers will be used as the Key Coherency byte. In the COHER register, coherency is both enabled and a Key Coherency Byte is selected. The Key Coherency Byte is basically the user (SW) telling the HW which byte of the field will be written/read last when an update to the field is desired.

Block Diagram and Configuration

Not Applicable

References

Not Applicable

DC and AC Electrical Characteristics

Not Applicable