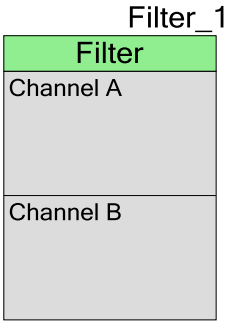
Filter
1.50

Features



- Easy user configuration of the digital filter block (DFB) coprocessor
- Two separate filtering channels
- Multiple windowing methods to tune filter responses
- Up to four cascaded filters in a single channel

General Description

The Filter component allows you to configure 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. Any time you need to modify frequency characteristics of a signal, you would use a filter. 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/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 *

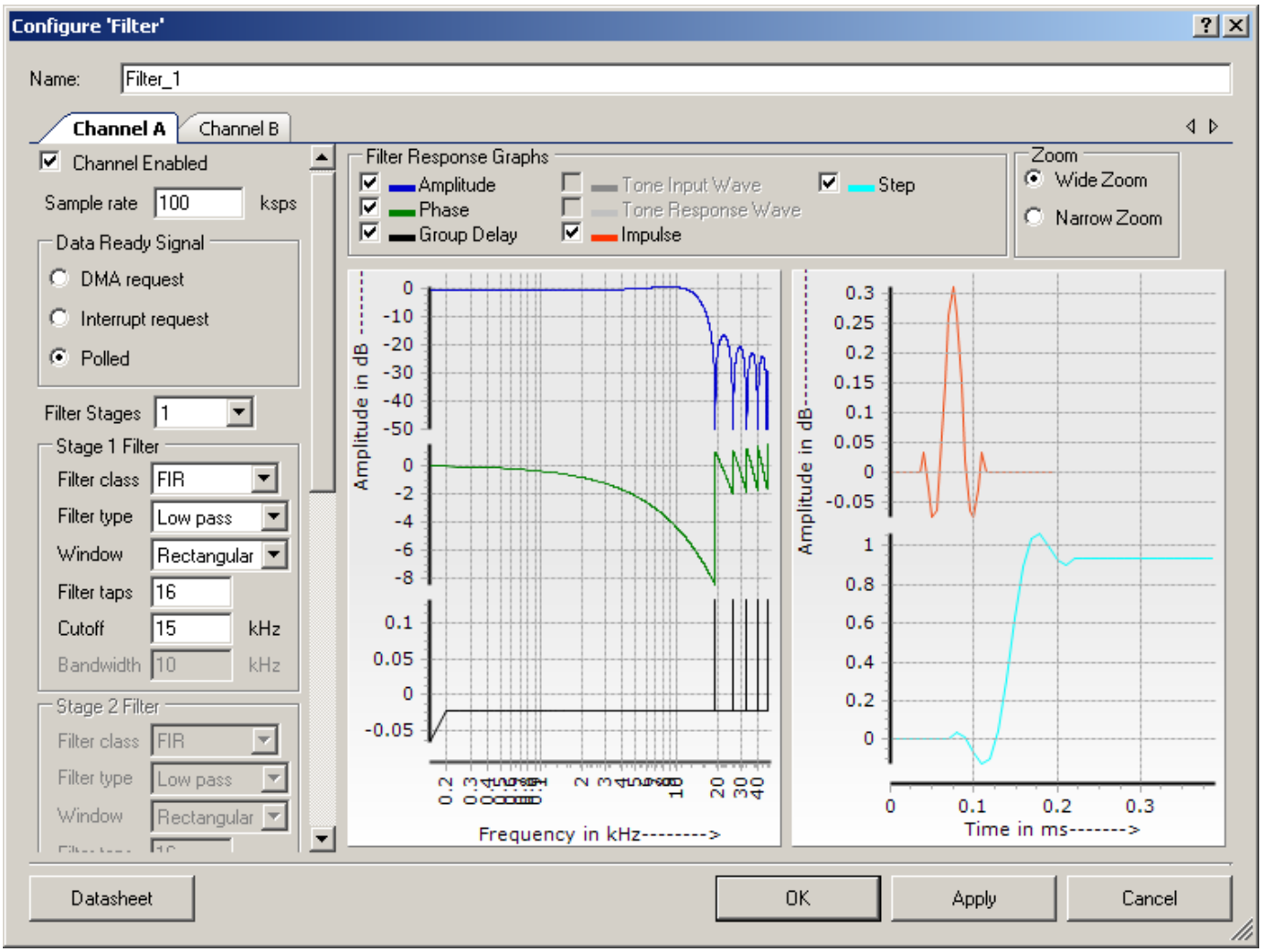
If either channel is configured to generate a DMA request in response to a data-ready event, the DMA Request output is enabled *for that channel*. Each channel has a separate DMA Request output. Connect the hardware signal to a DMA component to handle the DMA routine.

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

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 in a design. There is nothing to prevent you from placing two components, but the design will not build and work correctly.



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.

Filter Parameters

Channel Enabled

This parameter 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 not enough resources available, a warning icon will appear.

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

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

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:

$$f_{sMax} = \frac{Clk_{bus}}{19 + ChannelDep\ th_A + ChannelDep\ th_B}$$

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. 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.

Refer to 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.

Filter Stages

Each channel can, depending on resource availability, have up to four 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**, **Band stop**, **Sinc^4**, and **Hilbert** 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.

- **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 smooths 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: Filter taps

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

Filter Stage: Cutoff

Enter the edge of the pass band frequencies for Sinc^4, 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 is evenly split on either side of the previously defined arithmetic center frequency.

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).

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

Analog Blocks	Digital Blocks						API Memory (Bytes)		Pins (per External I/O)
	DFB	Datapaths	Macro cells	Status Registers	Control Registers	Counter7	Flash	RAM	
None	1	N/A	N/A	N/A	N/A	N/A	3054	69	1

One DFB.

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.

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.

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 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: 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_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 re-initializes 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() will produce unexpected behavior

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.

Sample Firmware Source Code

PSoC Creator provides numerous 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. 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.

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_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

DFB inherently requires that input data is 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 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 Decimator is 12 bits wide and aligned to bit 23 of the Decimator Output Sample register, and you want to stream this value to the DFB, the data alignment feature of the Decimator can be enabled. This 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 can also be written into bits 23:8 of the Staging A register when required.

Coherency (Filter_COHER) Register

Coherency refers to the hardware added to 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 and 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 the underlying hardware doesn't incorrectly use the field when it is partially updated by the System software. The Holding registers are protected on reads so that the underlying hardware doesn't update it when partially read by the System software or DMA. Depending on the configuration of the block, not all 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 Filter_DALIGN_REG = 0x00	OFF	OFF
16-bit	2	Filter_COHER_REG = 0x00 Filter_DALIGN_REG = 0x0F	TD_SWAP_EN | TD_INC_DST_ADR	OFF
24-bit	4	Filter_COHER_REG = 0xAA Filter_DALIGN_REG = 0x00	TD_SWAP_EN | TD_SWAP_SIZE4 | TD_INC_DST_ADR	TD_SWAP_SIZE4

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.

* To be set in the DMA wizard tool.

- 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 make use of polling method, then you should enable either bit 0 or 1 of this register based on 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 pole 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 FDFB				
		100 kHz (1.3 ksps)	--	0.03	0.075	mA

Parameter	Description	Conditions	Min	Typ	Max	Units
		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
1.50.e	Minor datasheet edits.	
1.50.d	Minor datasheet edits.	
1.50.c	Added PSoC 5 characterization data to datasheet	
1.50.b	Added characterization data to the datasheet	
1.50.a	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.

© Cypress Semiconductor Corporation, 2010-2016. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC (“Cypress”). This document, including any software or firmware included or referenced in this document (“Software”), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress’s patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

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

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