

Isolated Motor Control Feedback Using the **ADSP-CM402F/ADSP-CM403F/ADSP-CM407F/ADSP-CM408F** Sinc Filters and the **AD7403**

By Dara O’Sullivan, Jens Sorensen, and Aengus Murray

INTRODUCTION

This application note introduces the main features of the sinc filters of the [ADSP-CM402F/ADSP-CM403F/ADSP-CM407F/ADSP-CM408F](#) microprocessors, with a focus on high performance motor control applications.

The purpose of this application note is to highlight the key capabilities of the sinc filter module and to provide guidance on how to configure the sinc filter through software. For more information about the full range of sinc filter features and configuration registers, see the [ADSP-CM40x Mixed-Signal Control Processor with ARM Cortex-M4 Hardware Reference](#) and the documentation within the [ADSP-CM40x Enablement Software](#) package.

The sinc filter of each [ADSP-CM402F/ADSP-CM403F/ADSP-CM407F/ADSP-CM408F](#) microprocessor is part of a complete motor current feedback subsystem that includes a current shunt, a modulator to digitize and isolate the signal, and the sinc filter to decode the bit stream and present it to the controller. This application note describes how to set up the sinc filters.

MOTOR CURRENT CONTROL APPLICATIONS

Figure 1 shows a simplified schematic of an isolated current feedback system for inverter fed motor drives. The system overcomes the difficulty of isolating the analog signal that is generated across the current shunt from the high voltage common signal that is generated by the switching power inverter. The system converts the signal using isolated Σ - Δ modulators and then transmits a digital signal across the isolation barrier.

The Σ - Δ modulators generate a modulated bit stream as a function of the input voltage and transmit the signal across the isolation barrier to a filter circuit on the low voltage side. The sinc filter filters the bit stream from a second-order modulator, such as the [AD7403](#), to recover a 16-bit digital signal that represents the motor winding current.

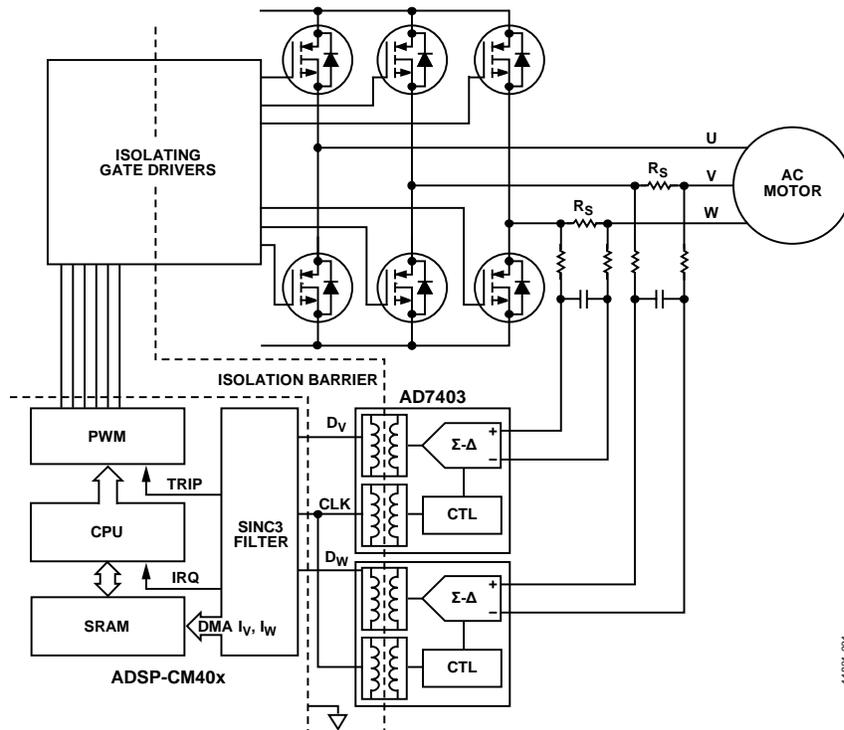


Figure 1. Isolated Current Feedback Using the [AD7403](#)

TABLE OF CONTENTS

Introduction	1
Motor Current Control Applications	1
Revision History	2
Sinc Filter Module Overview	3
Current Feedback System Overview	4
Current Shunt Selection	4
Modulator Clock, Primary Filter Decimation, and Data Interrupt Rate Selection.....	5
Aligning Sinc Impulse Response to PWM.....	6
Implementation of Impulse Response Alignment to PWM...	7
Sinc Data and Interrupt Rate	8

REVISION HISTORY

4/15—Rev. A to Rev. B

Changed AD7401A to AD7403	Universal
Changes to Introduction Section and Figure 1.....	1
Changes to Current Shunt Selection Section, Figure 3, and Figure 4	4
Changes to Modulator Clock, Primary Filter Decimation, and Data Interrupt Rate Selection Section, Figure 5, and Figure 7	5
Added Figure 6; Renumbered Sequentially	5
Added Figure 8, Aligning Sinc Impulse Response to PWM Section, Figure 9, and Figure 10	6
Added Figure 11, Figure 12, and Implementation of Impulse Response Alignment to PWM Section	7
Added Figure 13, Figure 14, and Sinc Data and Interrupt Rate Section.....	8
Changes to Figure 15.....	9
Changes to Figure 16 and Feedback Scaling Calculations Section.....	10
Changes to Figure 17 and Figure 19.....	11
Added Figure 18, Figure 20, and Figure 21	11
Changes to Figure 22 and Figure 25.....	12

Primary Filter Scaling	9
Secondary Filter Scaling and Overload Configuration	10
Sinc Module Fault Detection Functions.....	13
Sinc Filter Setup	14
Pin Multiplexer Configuration	14
Data Buffer Memory Allocation	14
Interrupt and Trigger Routing.....	15
Primary and Secondary Filter Configurations.....	16
Sinc Filter Software Support	17

Added Figure 23, Figure 24, and Figure 26.....	12
Changes to Figure 27.....	13
Added Figure 28	13
Changed Sinc Filter Setup and Software Driver Functions Section to Sinc Filter Setup Section	14
Changes to Data Buffer Memory Allocation Section, Figure 29, and Figure 31	14
Changes to Interrupt and Trigger Routing Section and Figure 32.....	15
Changes to Primary and Secondary Filter Configurations Section.....	16
Changes to Sinc Filter Software Support Section.....	17
Added Table 3; Renumbered Sequentially	17

11/13—Rev. 0 to Rev. A

Changes to Figure 1.....	1
Changes to Figure 4.....	4
Changes to Table 1	5

9/13—Revision 0: Initial Version

SINC FILTER MODULE OVERVIEW

The sinc filter block performs two functions: it generates a high fidelity feedback signal for the motor control algorithm, and it provides rapid detection of overload currents in the case of fault conditions. Connecting the overload fault signal to the pulse-width modulator (PWM) block can shut down the PWM inverter without any software intervention. The sinc filter transfers data directly to memory using direct memory access (DMA), and a processor interrupt can be generated when a preset number of samples is ready. The interrupt minimizes the software overhead to service the sinc filter after it is configured. The same feedback circuit applies to isolated dc bus voltage feedback and dc bus current measurements.

Figure 2 shows a block diagram of the sinc filter module. The sinc filter module has four sinc filter pairs that implement feedback signal filtering and overload detection on the digital bit streams connected to the inputs.

The filter enable function assigns sinc filter pairs to one of two configuration register groups to set the filter parameters. The expectation is that the motor current control requires multiple current or voltage filters configured in the same way. The sinc filter module supports control of two motors with one group of two filter pairs assigned to each motor. The primary filter settings are the filter order, decimation rate, offset bias, and gain scaling. The secondary filter settings are the filter order, decimation rate, overload trip levels, and glitch filter settings.

Other configuration functions include modulator clock frequencies, interrupt masking, and DMA data transfer. The other control peripherals required to set up the sinc filter are the port controller, which connects external pins to the sinc filter inputs, and the trigger routing unit (TRU), which connects output signals of the sinc filter to the appropriate peripheral.

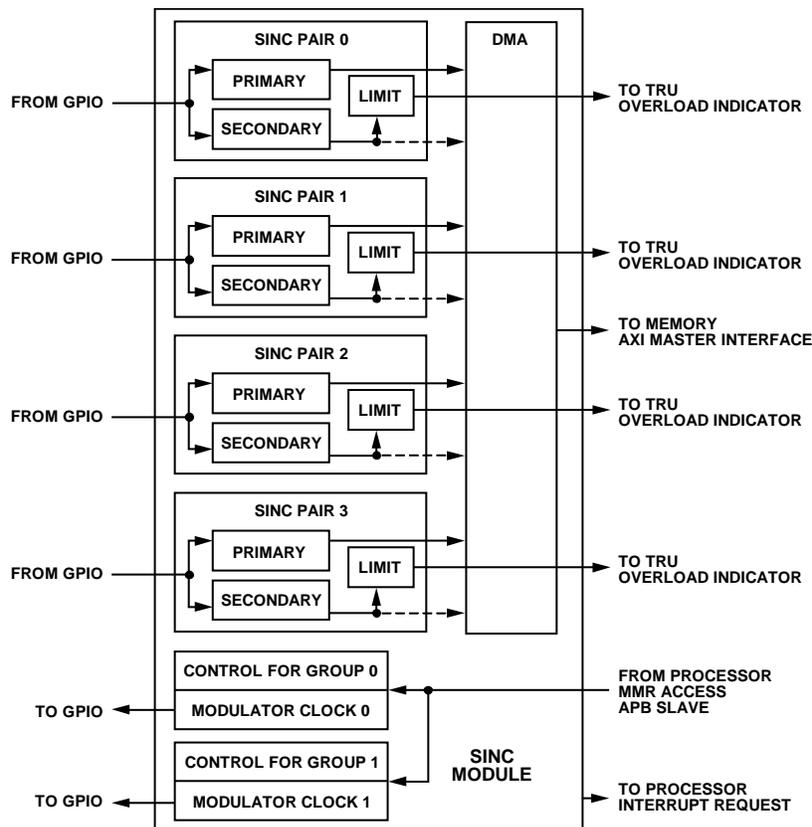


Figure 2. Sinc Filter Module Overview

11801-002

CURRENT FEEDBACK SYSTEM OVERVIEW

Figure 4 shows the key elements in the current feedback system. The shunt senses the winding current as a voltage signal that scales according to the shunt resistance. The AD7403 modulator generates an isolated bit stream with a pulse density (MDAT_w in Figure 4) that scales according to the full-scale input voltage range. The sinc filters extract the pulse density information according to the filter order (O for the primary filter and O' for the secondary filter) and decimation rate (D for the primary filter and D' for the secondary filter). The primary filter parameters optimize the filter for precision and additional bias and scaling blocks convert the data into a 16-bit, signed integer before it is transferred to memory. The secondary parameters optimize the filter for speed, and the outputs pass the signal to digital comparators that detect overload conditions. Upper and lower limit comparators detect current overloads, and a glitch filter waits for a minimum overload count (LCNT) within a specific window (LWIN) before generating an overload trigger signal. The overload trigger is a trip input signal for the PWM driving the motor inverter. The DMA transfer engine generates an interrupt signal to initiate algorithm execution when the winding current data is ready in memory.

CURRENT SHUNT SELECTION

The system specifications required to define the feedback are the peak control current, I_{CC,PEAK}, and the specified maximum input voltage, V_{MOD,MAX}, for the modulator. The peak current capability of the power inverter typically defines the control current range, but other considerations may apply. The specified maximum operating voltage of the AD7403 modulator is ±250 mV, which is the maximum voltage range within which the modulator specifications are valid. The maximum operating voltage is lower than the ±320 mV full-scale range (V_{FS}) of the modulator because the linearity and signal-to-noise performance degrades significantly as inputs approach full scale. The shunt resistance must be less than V_{MOD,PEAK}/I_{CC,PEAK} to satisfy these constraints, and the closest nominal shunt value is chosen. For the example in Figure 3, given that the power stage peak current rating is 8.5 A, the maximum shunt resistance is 29.4 mΩ. For derating, a smaller standard size shunt is picked. For example, a 25 mΩ shunt yields a specified maximum current of 10 A and a peak current of 12.8 A.

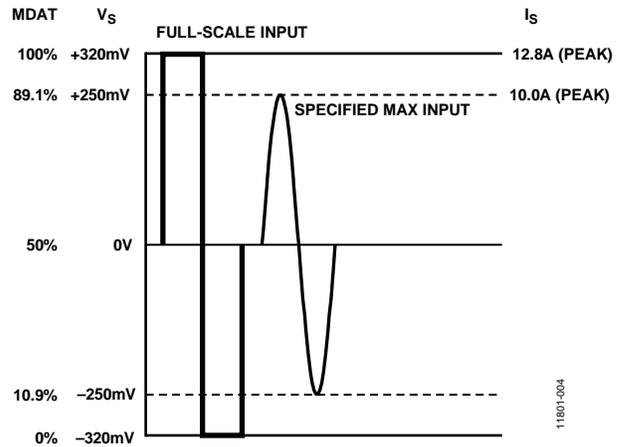


Figure 3. Feedback Current Operating Ranges

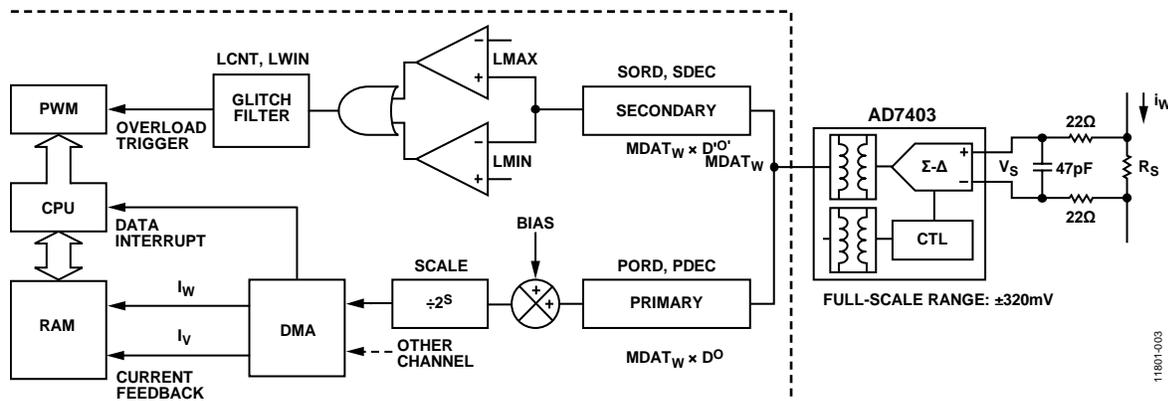


Figure 4. Sinc Filter Current Feedback Paths

MODULATOR CLOCK, PRIMARY FILTER DECIMATION, AND DATA INTERRUPT RATE SELECTION

The frequency of the modulator clock (f_M) and the decimation rate (D) are the parameters that define the sinc filter performance. The filter order (O) is typically one order higher than that of the front-end modulator. Therefore, when the AD7403 is used, the filter is third-order. The equations for the filter frequency response and filter group delay follow. The frequency response shown in Figure 5 and Figure 6 has zeroes at frequencies that are even multiples of the decimation frequency (f_M/D). Therefore, matching the decimation frequency to the PWM switching frequency substantially reduces PWM switching harmonics. Other considerations include the increase in the filter group delay with decimation rate and the maximum decimation limit of the filter.

$$H\left(e^{j\frac{f}{f_M}}\right) = \left[\frac{1}{D} \times \frac{\sin\left(D\frac{\pi f}{f_M}\right)}{\sin\left(\frac{\pi f}{f_M}\right)} \times e^{-j(D-1)\frac{\pi f}{f_M}} \right]^O$$

where:

H is the transfer function of the sinc filter in the frequency domain.
 f is the frequency.

$$\tau_d = \left(\frac{D-1}{2}\right)\frac{O}{f_M}$$

where τ_d is the filter group delay.

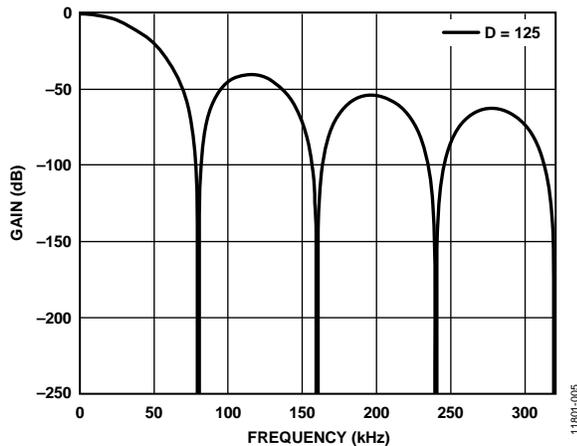


Figure 5. Sinc Filter Amplitude Response

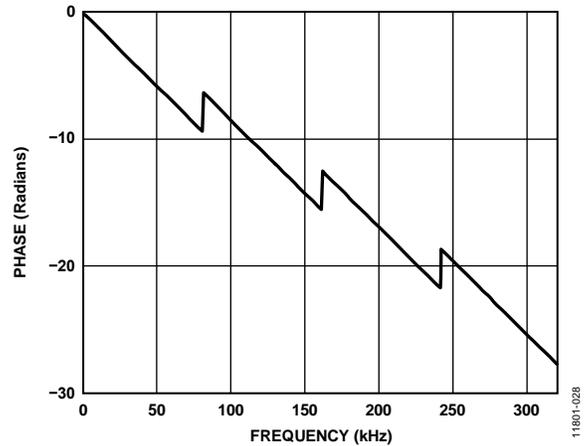


Figure 6. Sinc Filter Phase Response

For a given filter order, the decimation rate and filter order are the filter parameters that define the signal-to-noise ratio (SNR) and group delay of the filter. Figure 7, Figure 8, and Table 1 show the variation of the SNR, effective number of bits (ENOB), and group delay vs. the decimation rate for a third-order filter with a 10 MHz modulator clock. The decimation rate must be in the range of 85 to 210 to achieve an ENOB of 11 bits to 14 bits and an SNR of 67 dB to 86 dB, which is the filter performance range required for current feedback. The group delay is between 12 μ s and 32 μ s in this decimation rate range. Note that the SNR and ENOB numbers listed in Table 1 assume an ideal signal chain. The numbers represent theoretical maximum numbers and serve to illustrate the trade-off between the SNR/ENOB and the filter group delay only. Any practical implementation gives lower performance.

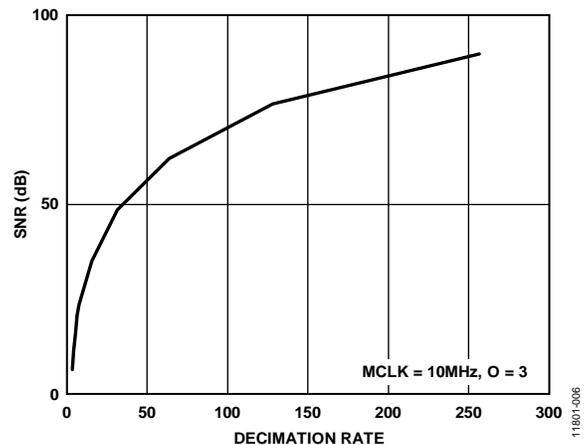


Figure 7. Sinc Filter SNR

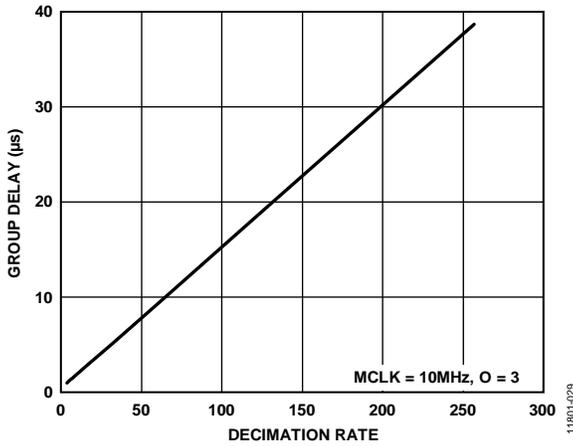


Figure 8. Sinc Filter Group Delay

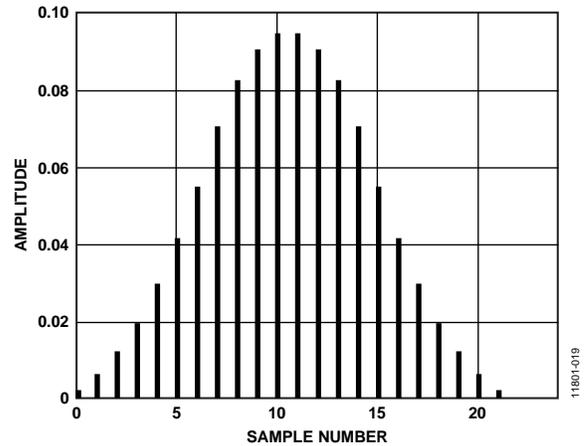


Figure 9. Impulse Response of Sinc3 Filter with Decimation Rate of 16

Table 1. SNR, ENOB, and Group Delay with Decimation Rate¹

Decimation Rate	SNR (dB)	ENOB (Bits)	Group Delay (µs)
85	68	11	12.6
113	74	12	16.8
154	80	13	23.0
210	86	14	31.4

¹The test condition is a ±200 mV sine wave at 1.22 kHz.

ALIGNING SINC IMPULSE RESPONSE TO PWM

With the selection of decimation rate and modulator clock values, the characteristics of the filter are set. It is equally important to match the filter characteristic to the application. A sinc filter has memory and the current output depends on not only current input but also previous inputs and outputs. The impulse response is useful for examining the effect of the sinc filter.

The impulse response of a system is defined as the output sequence when the system is stimulated by a unit pulse. If the system is linear and time invariant, the output response to any input sequence can be determined through convolution of the input and the impulse response as follows:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] \times h[n-k]$$

where:

y is the output sequence.

n is the sample number.

x is the input sequence.

k is the index.

h is the impulse response of the system.

The impulse response, *h*, when known, can be used to determine the response to any input. Figure 9 shows the impulse response for a sinc3 filter with a decimation rate of 16.

A third-order sinc filter has a hyperbolic impulse response. Figure 9 shows a weighted sum, which gives most weight to samples at the center and less weight to samples at the beginning and end. The effect of the weighted sum must be taken into account when measuring motor currents.

The current through a motor driven by a switching inverter can be split into two components: an average component and a switching component. For control purposes, the switching component is unwanted and must be eliminated so only the average component remains.

Figure 10 shows that there are two instances during a switching period when the average phase current can be measured. Those instances are at the beginning and middle of a switching period. Both instances are indicated by a synchronizing pulse, PWM_SYNC. Failing to measure at the point of average phase current results in signal degradation due to aliasing.

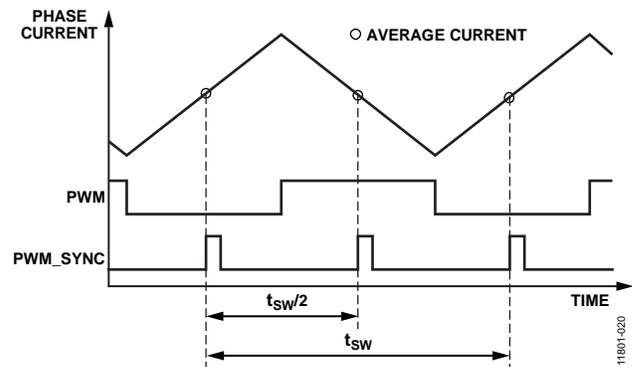


Figure 10. Motor Phase Currents and Relationship to the Inverter PWM

With a sample-and-hold-based converter, the average value of the currents is measured by letting the PWM_SYNC signal trigger the sample-and-hold circuit. However, due to the duration of the impulse response of the sinc filter, the task of suppressing the switching component requires a different approach when using Σ-Δ converters.

The impulse response is symmetrical around the center pin, meaning the sinc filter gives equal weight to samples before and after the center pin (see Figure 9). Furthermore, the switching component is symmetrical around the point of average current. That is, if x equally spaced samples taken before the point of average current are added to x equally spaced samples taken after the point of average current, the result is the average current. In other words, the switching component sums to zero.

These properties are utilized to extract the average current while also eliminating the switching component completely. If the center pin of the impulse response is aligned with the point of average current, an equal number of samples are taken before and after the desired sampling point. Because the samples before and after the center pin are given equal weight and the switching current is symmetrical around this point, the filter output is the true average current. This technique is illustrated in Figure 11.

The most weight is given around the desired sampling point. The further away from this point, the less weight is given to samples.

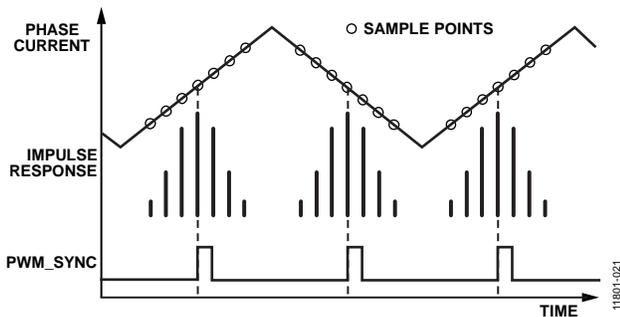


Figure 11. Aligning the Center Pin of Impulse Response to the Point of the Average Current, Equal Number of Samples Taken Before and After This Point

Aligning the center pin of the impulse response to the point of average current is equivalent to aligning the center pin to the PWM_SYNC pulse. However, to perform the alignment correctly, the actual impulse response must be known.

In most applications, high decimation rates are used; but, for simplicity, a decimation rate of 5 is used in the following example. The impulse response of a sinc3 filter is shown in Figure 12.

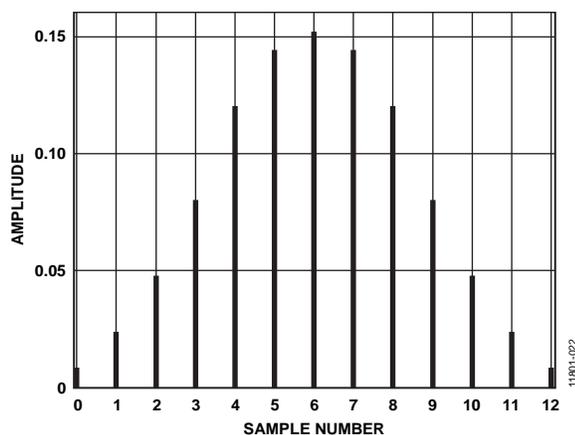


Figure 12. Impulse Response of a Third-Order Sinc Filter with Decimation Rate of 5

The number of pins in the impulse response is

$$O \times D - 2$$

Therefore, the number of pins in a third-order filter with a decimation rate of 5 is 13. It is worth noting that the number of pins is much greater than the decimation rate.

From the number of pins, the length of impulse response, in seconds, can be calculated as

$$t_M \times (O \times D - 2)$$

where t_M is the period of the modulator clock.

The duration of the impulse response is important because it tells how long it takes a sample to make its way completely through the filter.

The center pin of the impulse response is halfway through the total filter length. Therefore, the time it takes a sample to propagate halfway through the filter must be

$$\tau_d = \frac{t_M \times (O \times D - 2)}{2}$$

IMPLEMENTATION OF IMPULSE RESPONSE ALIGNMENT TO PWM

The Aligning Sinc Impulse Response to PWM section describes how the impulse response must be aligned to PWM to extract the true average motor current. This section describes how the implementation can be performed on the [ADSP-CM402F/ADSP-CM403F/ADSP-CM407F/ADSP-CM408F](#).

The task is to align the center pin of the impulse response to the PWM_SYNC pulse. As described in the Aligning Sinc Impulse Response to PWM section, the center pin is found half of an impulse response after the first pin of the filter. Measured in seconds, that is

$$\tau_d = \frac{t_M \times (O \times D - 2)}{2}$$

In other words, if the feed of input data starts ($t_M \times (O \times D - 2) / 2$) sec before the PWM_SYNC pulse, the center pin aligns with the PWM_SYNC pulse, as shown in Figure 11. The feed of data to the filter is controlled by enabling or disabling the modulator clock.

Advancing enablement of the modulator clock with respect to the PWM_SYNC pulse is impossible because it requires the generation of a negative delay. That is, when the PWM_SYNC signal is needed, PWM has not yet been started. However, instead of advancing the start of the modulator clock, exactly the same effect can be achieved by delaying the start of the modulator clock by τ_d . As long as the switching period, t_{sw} , is constant and there is an integer number of data points from the sinc filter per switching period, delaying the start of the modulator clock gives the same result as advancing it.

To generate the delay, the [ADSP-CM402F/ADSP-CM403F/ADSP-CM407F/ADSP-CM408F](#) TRU and a general-purpose timer are used, as shown in Figure 13.



Figure 13. Aligning Sinc Impulse Response to PWM Using General-Purpose Timer and Triggers

The PWM timer block outputs a trigger master, PWM_SYNC, which is routed to the TRU and onto the trigger slave of a general-purpose timer, TRGS_TIMER0_TMRx. The general-purpose timer generates a delay with respect to the PWM_SYNC pulse, which brings the impulse response in alignment with PWM. When the delay expires, the general-purpose timer generates a trigger master, TRGM_TIMER0_TMRx, which again is routed to the TRU and onto the trigger slave of a sinc filter, SINC_SYNC. The sequence is illustrated in the timing diagram in Figure 14.

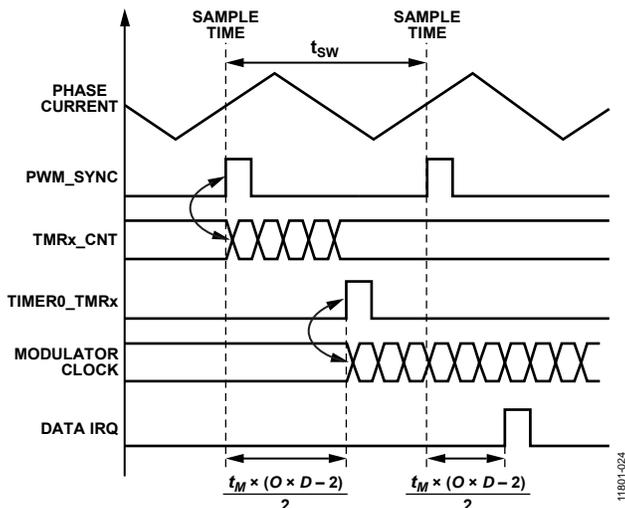


Figure 14. Startup of Sinc Filter Using a General-Purpose Timer and a TRU

In Figure 14, note that the general-purpose timer generates a delay of

$$\frac{t_M \times (O \times D - 2)}{2}$$

This delay brings the center pin of the impulse response in alignment with the PWM_SYNC pulse. Because the center pin is at the midpoint of the impulse response, it takes another half impulse response before data has propagated through the filter.

In Figure 14, note that the data interrupt occurs half of an impulse response after the PWM_SYNC pulse.

After the data interrupt is started, there is no need to realign the center pin to the PWM_SYNC pulse. The filter remains in sync and, thus, the impulse response is always aligned to PWM. Therefore, the general-purpose timer used for alignment can be reused for other purposes.

SINC DATA AND INTERRUPT RATE

It is not possible to match the sinc filter decimation rate with the typical PWM switching frequencies used in motor drives. For example, matching a switching frequency of 16 kHz requires a decimation rate of 625, and the resulting filter group delay is 94 μs. This decimation rate is well above available values, and the filter group delay limits the bandwidth of the current loop.

Instead, the decimation rate is set to a multiple of the PWM frequency to lower the group delay and still achieve the target filter SNR. The control algorithm samples the data at a submultiple of the decimation frequency matching the PWM switching. This software decimation process involves transferring multiple data samples to a circular buffer in memory and reading the most recent data sample in response to the interrupt generated when the buffer is full. The DMA engine transfers data from the primary sinc filter to data memory, and the sinc control unit generates a trigger every time it transfers a fixed number of samples.

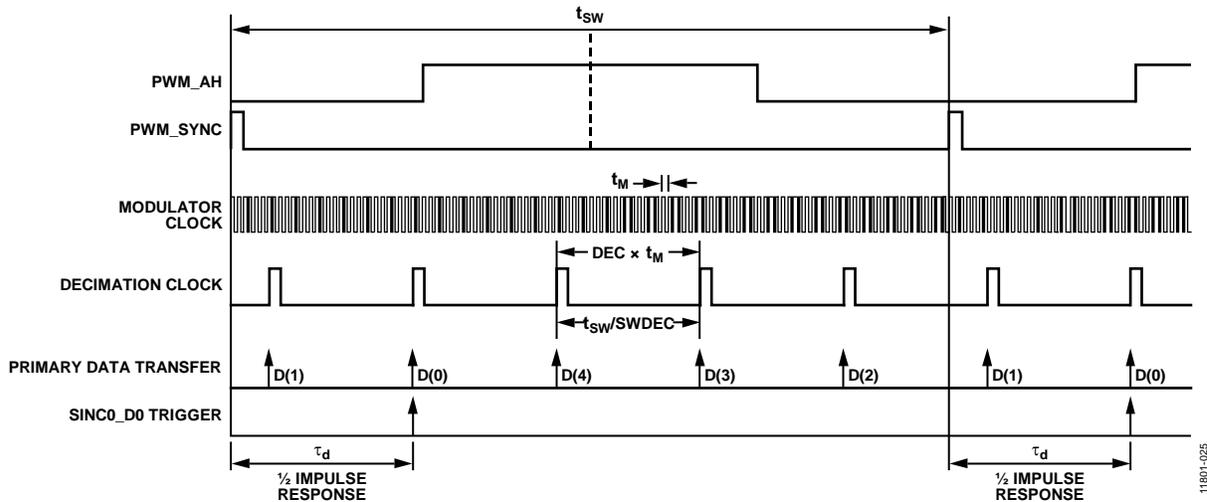


Figure 15. Modulator and Decimation Clock Timing

Figure 15 shows the alignment between PWM switching, the modulator clock, the decimation clock, and data sampling. The synchronizing pulse (PWM_SYNC) from the PWM block aligns the startup of the modulator clock with the PWM frequency. The decimation frequency is a submultiple of the modulator clock and a multiple of the PWM frequency. The SINC0_D0 trigger rate is at the PWM frequency.

The information in Table 2 illustrates the process of selecting the decimation rate and the PWM switching frequency. The first three entries in the table are chip level settings for the core and peripheral clocks. The maximum core clock rate is 240 MHz, and it is typically an even multiple of the system (peripheral) clock frequency. The sinc filter modulator clock derives from the system clock based on the MDIV register field value, and there are a limited set of values in the 5 MHz to 20 MHz range. The primary hardware decimation rate (PDEC) is 125, which sets the filter SNR at 76.0 dB (>12-bit ENOB) with a filter group delay of 18.6 μs. The modulator clock is 10 MHz; therefore, the primary decimation clock frequency is 80.0 kHz, and a software decimation rate (SWDEC) of 5 synchronizes the sample rate with a 16.00 kHz PWM frequency (PWM). To tune the PWM frequency, adjust the sinc filter decimation rate.

The equation governing the relationship among the modulator clock, the PWM frequency, and the hardware and software decimation rates is

$$\frac{MCLK}{PWM} = PDEC \times SWDEC$$

where:

MCLK is the modulator clock.

PWM is the PWM frequency.

PDEC is the primary hardware decimation rate.

SWDEC is the software decimation rate.

The hardware and software decimation rates must be integers. The PCNT register field value in the sinc filter sets the software decimation rate. The PCNT register field value loaded in the sinc filter control register is less than one of the number of sample delays before an interrupt is generated. The PWM_TM0 register sets the PWM switching frequency and, therefore, sets the sample timing.

Table 2. Decimation Rate Selection

Parameter	Symbol	Value	Unit
Core Clock	CCLK	240	MHz
System Clock Divider	SYSEL	3	
System Clock	SYSCLK	80	MHz
Modulator Clock Divider	MDIV	8	
Modulator Clock (1/t _m)	MCLK	10	MHz
Decimation Rate	PDEC	125	
Filter SNR	SNR	76.0	dB
Filter ENOB	ENOB	12.3	Bits
Decimation Clock Frequency	DCLK	80.0	kHz
Filter Group Delay	τ _d	18.6	μs
Software Decimation Rate	SWDEC	5	
Data Transfer Count	PCNT	4	
PWM Frequency (1/t _{sw})	PWM	16.00	kHz
PWM Period Count	PWMTM	2500	

PRIMARY FILTER SCALING

The sinc filter order (O) and decimation rate (D) set the primary filter dc gain (G_{DC}), given by

$$G_{DC} = D^O$$

The sinc filter block has output scaling and bias functions to convert the data to a 16-bit signed integer before it is transferred to memory. The data format is valid as a fractional 16-bit integer (S.15) in the range of ±1.0 or as a signed 16-bit integer in the range of ±2¹⁵, depending on interpretation.

The raw filter output is an integer between 0 and D^O , where $D^O/2$ aligns with a 50% pulse density corresponding to 0 A. Adding a bias value of $-D^O/2$ to the output sets the correct zero level. Dividing the result by $D^O/2$ scales the full-scale, fractional integer output to ± 1 . However, for simplicity, the unit has a simple binary scale factor (S), where the user selects S to set the gain near 1.0. Regardless of the scaling, the DMA engine only transfers the 16 least significant bits of the output data; therefore, correct scaling is essential to avoid loss of precision. The output data is saturated to prevent data overflow, which inverts the polarity of the output signal due to incorrect scale factor selection. The filter sets an overflow fault flag when saturation occurs.

Conversion of the data to a floating point involves scaling by the inverse of the current shunt gain and adjusting for the mismatch between the filter dc gain and the scale factor.

Feedback Scaling Calculations

The final system gain from the shunt current to the data-word in memory derives from the gains of all the elements in the system, as shown in Figure 16. The isolated modulator in this example is the AD7403.

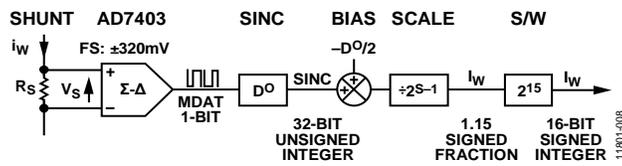


Figure 16. Sinc Primary Output Data Scaling

The shunt voltage seen by the modulator is

$$V_s = i_w \times R_s$$

where:

V_s is the input voltage.

i_w is the winding current (analog).

R_s is the resistance of the shunt.

The isolated modulator expects a bipolar input and generates a 50% pulse density for a 0 V input. The pulse density of the data stream (MDAT) is a function of the ratio of the input voltage (V_s) to the positive full-scale input (V_{FS}):

$$MDAT = 0.5 \left(\frac{V_s}{V_{FS}} + 1 \right)$$

In the case of the AD7403, the positive full-scale voltage is 320 mV, and the ones density is 89.1% for the specified maximum voltage of 250 mV.

The sinc filter dc gain is D^O ; therefore, the raw output as a function of the input voltage is

$$SINC = \frac{D^O}{2} \left(\frac{V_s}{V_{FS}} + 1 \right)$$

This dc scaling applies to the secondary filter outputs, and the maximum secondary decimation rate restricts the raw output data range to a 16-bit unsigned integer. The secondary output is 0 at the negative full-scale input and D^O at the positive full-scale input.

The bias and scale functions in the primary output path remove the bias on the sinc data and rescale the data to a 16-bit signed integer. The bias value must be $-D^O/2$ to eliminate the offset in the sinc output for a modulator with a bipolar input range. The rescaling selects the appropriate bit range from the sinc output word.

$$I_w = \frac{SINC - \frac{D^O}{2}}{2^{S-1}} = \frac{D^O}{2^S} \left(\frac{V_s}{V_{FS}} \right)$$

where I_w is the winding current (digital).

The scale factor must set the maximum fractional integer output at 1.0, which is true when

$$\frac{D^O}{2^S} \ll 1 \therefore S \gg \frac{\ln(D^O)}{\ln 2}$$

The sinc output equation, when reading the data as a signed integer, adds a scale factor of 2^{15} .

$$I_w = \frac{D^O}{2^S} \left(\frac{V_s}{V_{FS}} \right) \times 2^{15}$$

The current reading as a function of the actual winding current in this case is

$$I_w = i_w \times \left(\frac{R_s}{0.32} \right) \left(\frac{D^O}{2^S} \right) \times 2^{15}$$

SECONDARY FILTER SCALING AND OVERLOAD CONFIGURATION

The secondary sinc filter data outputs connect directly to overload comparators and a glitch filter, as shown in Figure 4. The secondary filter decimation rate is set significantly lower than that of the primary filter to achieve fast response to fault conditions. The processor TRU connects the overload trip signal to the PWM modulator shutdown input to clear the fault. The TRU can also connect the overload signal to other sources, such as an external general-purpose input/output (GPIO) used to shut down other critical circuit elements.

Typical power inverter switches can withstand a short circuit for a few microseconds; therefore, the overload circuit must have a relatively short detection window. Because the sinc filter can respond to a step input within three decimation cycles, a response within 3 μ s is possible using a decimation rate of 10, as shown in Figure 17 and Figure 18. The sinc filter also filters out inverter switching noise, as shown in Figure 19, Figure 20, and Figure 21. In Figure 19, Figure 20, and Figure 21, a 10 A peak test waveform injects 16 A noise pulses of 1.5 μ s in duration and 16 A overload pulses of 40 μ s in duration. The filter rejects the short noise pulses, but the circuit detects the 16 A overload pulses. The maximum and minimum trip levels in this test are at secondary sinc outputs corresponding to ± 16 A.

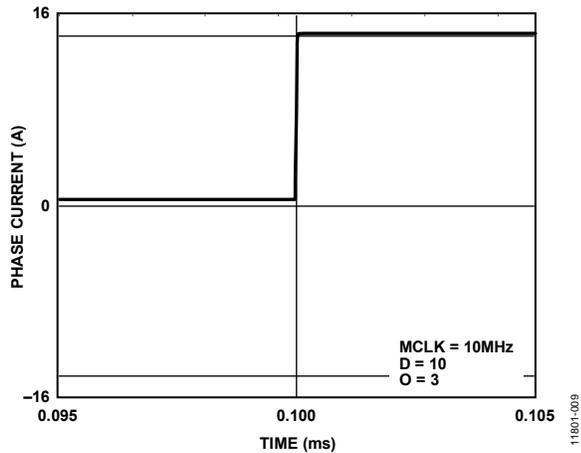


Figure 17. Secondary Filter Overload Detection: Test Current Waveform

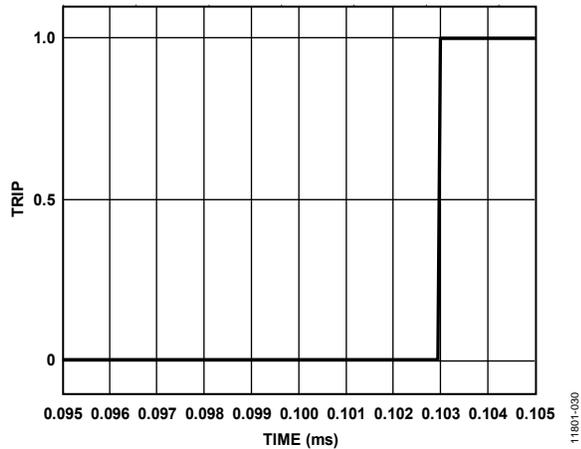


Figure 18. Secondary Filter Overload Detection: Overload Trip Signal

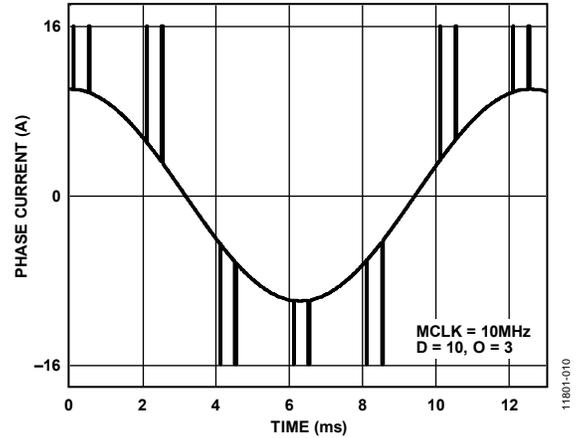


Figure 19. Test Current Waveform

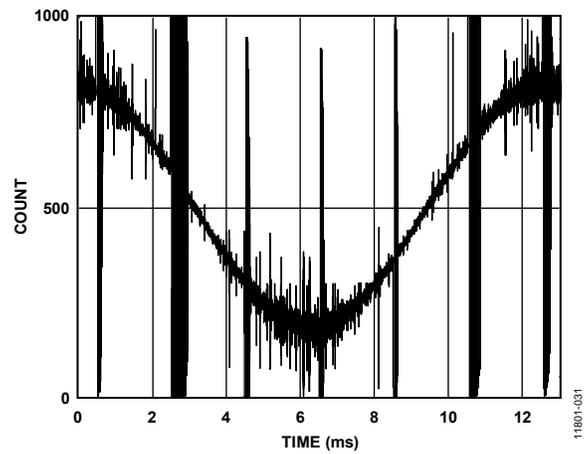


Figure 20. Secondary Sinc Data with a Decimation Rate of 10

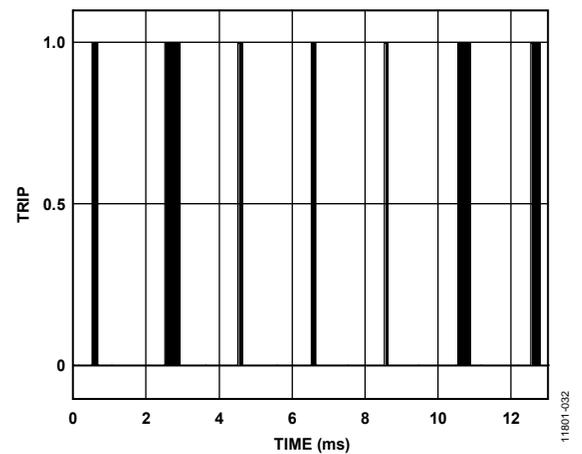


Figure 21. Signal Indicating if Data Exceeds Maximum or Minimum Limits

A faster response is possible at a lower decimation rate; but, as shown in Figure 22, Figure 23, and Figure 24, the secondary sinc output exceeds the trip levels even for a simple sinusoidal test current of ± 10 A. The higher sinc filter noise at a decimation rate of 5 generates multiple false trip signals. Figure 25 and Figure 26 illustrate the SNR at high (10) and low (5) decimation rates and the noise margin for the trip signal.

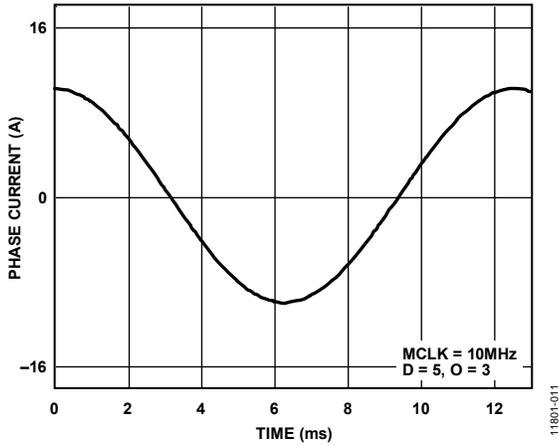


Figure 22. Test Current Waveform

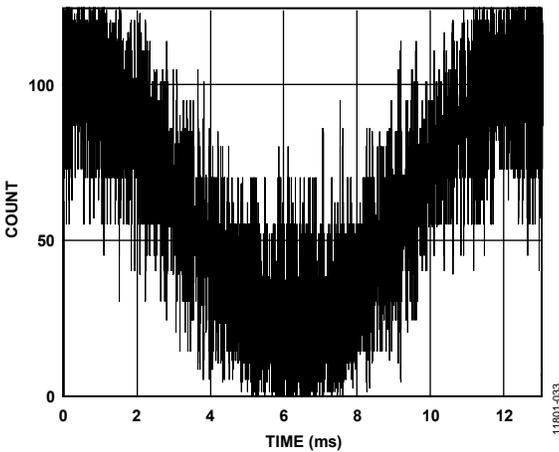


Figure 23. Secondary Sinc Data with a Decimation Rate of 5

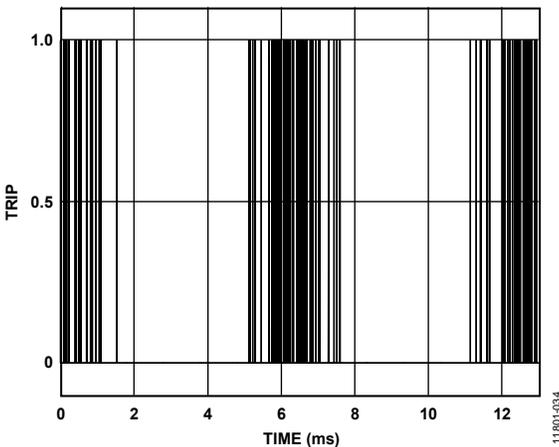


Figure 24. False Overloads Detected

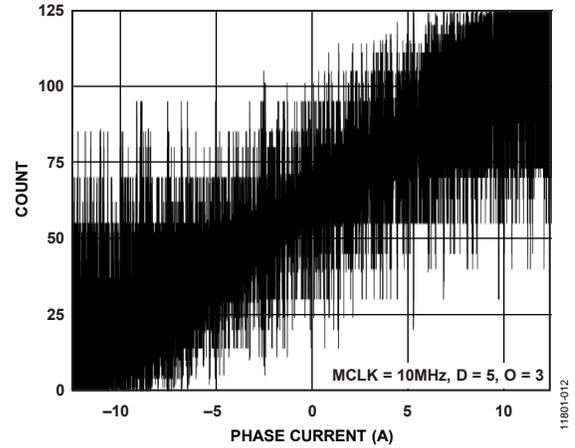


Figure 25. Secondary Filter Gain Curve for Decimation Rates of 5

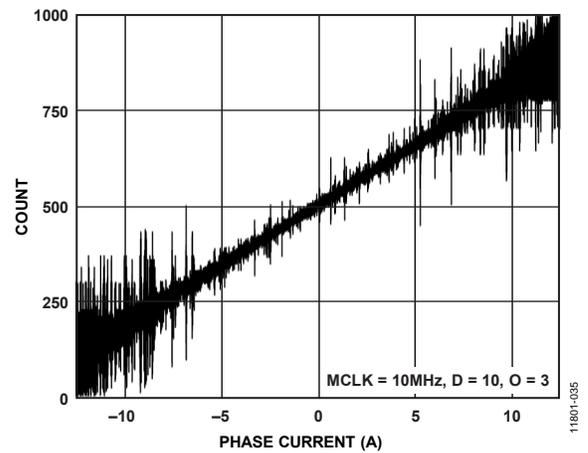


Figure 26. Secondary Filter Gain Curve for Decimation Rates of 10

The secondary output glitch filter rejects short overload trips by eliminating trips with durations less than a minimum count (LCNT) with a trip count window (WCNT). Figure 27 and Figure 28 illustrate how the glitch filter eliminates the spurious overload that is triggered when the decimation rate is 5; however, there is an additional three cycle delay in the response time. Therefore, there is no reduction in response time from the lower decimation rate. Figure 27 and Figure 28 illustrate the ability of the filter to reject short noise pulses on the analog input. In this example, the noise pulse is 1.5 μ s in duration.

The secondary sinc filter includes a set of history buffers that capture the eight most recent data samples before a trip is generated for diagnostic purposes. The data in the history registers is accessed directly through the device peripheral memory infrastructure.

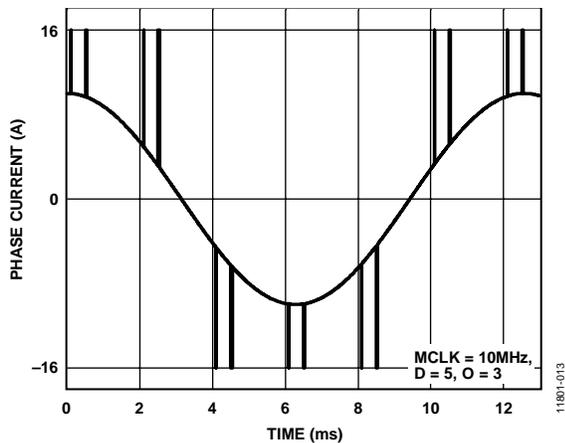


Figure 27. Test Current Waveform with Overload Events

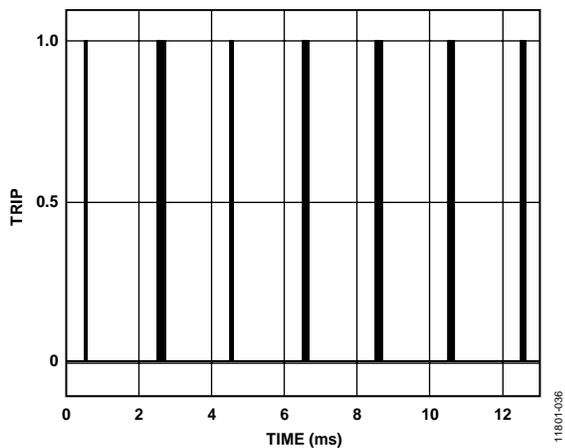


Figure 28. Overload Trip Signal with a Decimation Rate of 5 and Glitch Filter with WCNT = 4 and LCNT = 4

Secondary Filter Scaling and Trip Level

There is no extra output scaling on the secondary filters; therefore, valid minimum and maximum trip levels are within the range of 0 to D^O . The negative, full-scale current maps to 0, and the positive, full-scale current maps to D^O . Setting the minimum and maximum trip levels to 1 and $D^O - 1$ enables the maximum range of the trip function. The transfer function shown in Figure 25 and Figure 26 (for a decimation rate of 10 and a 20 m Ω shunt) shows that the noise peaks for a 10 A input are within the maximum (1000 counts) and minimum (0 counts) outputs of the filter. Set the LMIN and LMAX trip levels to 1 count and 999 counts to avoid spurious trips for 10 A peak current. The actual current level at which the trip is triggered ranges between 11 A and the full scale of 16 A. The likelihood of a trip increases the closer the current is to the full-scale limits.

The overload circuit operates slightly more precisely within the specified modulator input range. For the previous case, the peak noise at 5 A input is 700 counts, which is equivalent to 6.4 A. Therefore, the trip is set to operate within the range of 5 A to 6.4 A. The LMAX and LMIN settings, in this case, are 700 counts and 300 counts. Attaining precise trip settings using lower decimation rates is more difficult.

SINC MODULE FAULT DETECTION FUNCTIONS

In addition to overload faults, the sinc module checks for data faults that can arise from incorrect filter settings overloading the chip infrastructure.

The primary filter detects output data saturation when there is an incorrect setting of the output bias and scaling. The filter DMA engine detects a first in, first out (FIFO) error if it fails to transfer data before the filter writes new data. The ESATx bits and the EFOVfx bits in the SINC_CTL register mask the SINC_STAT interrupt generation on saturation and FIFO faults.

SINC FILTER SETUP

There are several steps to set up the sinc filter module as well as the signal routing and data buffers before the filter is ready for use. After it is configured, the DMA engine automatically streams primary filter data to memory, and the secondary limit function shuts down the PWM module in the case of an overload. The system generates an interrupt when data is ready; therefore, the processor can execute the control algorithm and update the PWM duty cycle registers. Figure 29 outlines the interconnections required between the sinc filter block and the CPU, SRAM, PWM, and external pins to capture motor current feedback signals.

The following four steps set up current feedback using the sinc filter:

1. Configure the pin multiplexer.
2. Allocate the data buffer memory.
3. Connect the interrupt and trigger routing.
4. Configure the primary and secondary filters.

This section further describes these steps, detailing the setup process and programming the sinc filter control registers.

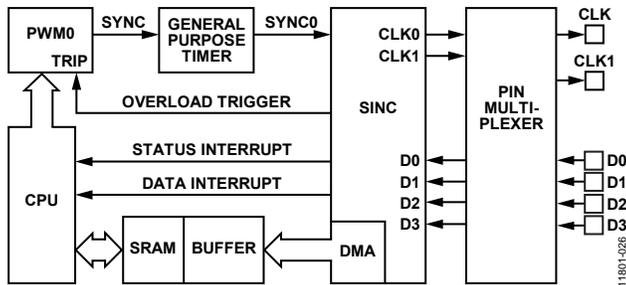


Figure 29. Sinc Filter System Configuration

PIN MULTIPLEXER CONFIGURATION

The pin multiplexer connects the front-end modulator clock and data pins to the sinc module. Two modulator clock outputs are available, the SINC0_CLK0 and SINC0_CLK1 pins. Four sinc data input pins are available, the SINC0_D0 pin, the SINC0_D1 pin, the SINC0_D2 pin, and the SINC0_D3 pin. The PORT_MUX register controls the selection of these pins from four alternate input or output signals for each of the multiplexed pins. The PinMux64.jar and PinMux32.jar Java® application programs, which are supplied with the [ADSP-CM40x Enablement Software package](#), automatically generate C code to enable the user port selections. Figure 30 is a snapshot of the PinMux64.jar Java application window.

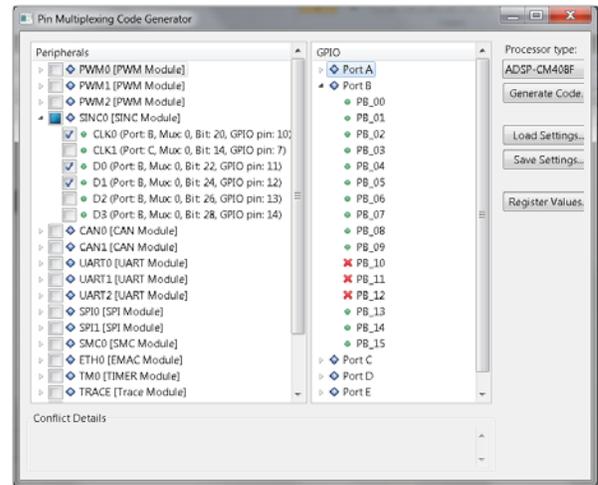


Figure 30. Pin Multiplexing Code Generator

DATA BUFFER MEMORY ALLOCATION

The primary filter data buffers must be defined and assigned memory space to allow the control algorithm to use the data. The software decimation rate and the number of feedback channels define the buffer size. The data is ordered on a per group basis in channel sequence. The pointer to the most recent data set is stored in the SINC_PPTRx register. Figure 31 shows how the data buffer is organized and how the head and tale specify the start and end of the buffer, where SINC_OUT_x_M[n] is the data for the nth most recent sample in the Mth channel in the filter group x, and n = 0 is the most recent data.

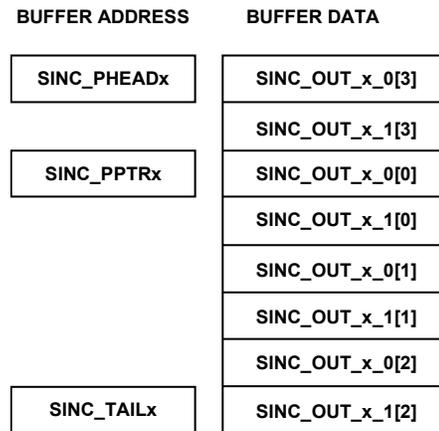


Figure 31. Data Buffer Organization

INTERRUPT AND TRIGGER ROUTING

Figure 32 shows the sinc filter interconnection with other peripheral functions using interrupt and trigger signals. The SINC_STAT interrupt is the single processor interrupt signal of the sinc filter module. The TRU connects the other trigger signals to the peripherals and processor interrupts of the sinc filter module. Loading the trigger master address into the trigger slave registers in the TRU connects the routing.

Through a general-purpose timer, TMRx, the TRU synchronizes the sinc filter modulator and decimation clocks with the PWM frequency to meet the timing defined in Figure 15. The TRU connects the sinc filter data transfer trigger to the control software interrupt to start execution of the control algorithm.

The TRU connects both of the sinc overload triggers to the TR_T1 input of the PWM to enable overcurrent protection. The TR_T0 input connects to the external trip signal only. The PWM, as well as the TR_T0 and TR_T1 inputs, must be configured to accept these triggers. There are two interrupt triggers produced by an overload fault: the STAT interrupt, connected directly to the CPU, and the TR_T1 interrupt, generated by the sinc overload trigger.

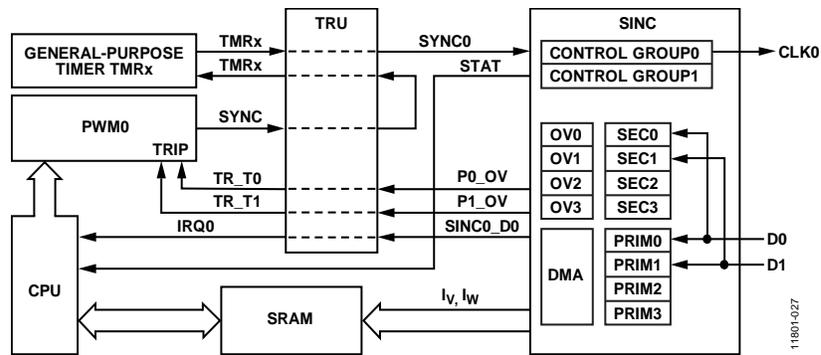


Figure 32. Sinc Filter Trigger Routing

11801-027

PRIMARY AND SECONDARY FILTER CONFIGURATIONS

Filter channels are organized in groups because it is typical for two or three feedback signals to need the same filter parameters. The sinc module has two groups of configuration registers. The channels in any one group share the same clock and have common filter parameters, such as filter order, decimation rate, scaling, and bias. The exception is the overload limit and history registers, which have a per channel organization. Enabling a filter channel assigns it to a configuration group. The configuration registers define the modulator clocks, filter parameters, DMA data transfer, and overload detection.

Figure 33 describes the assignment of filter and system parameters to Group 0 registers. The organization of Group 1 registers is the same. The SINC_CTL register enables each channel and assigns the control group. The recommended process is to configure the filter group before enabling the channels in the group. The SINC_CTL register also masks the SINC_STAT interrupt. The system status register, SINC_STAT, reports the fault and data trigger count status.

Three registers per group and the clock register define the primary and secondary filter parameters. The SINC_RATE0 register and the SINC_RATE1 register set the primary and secondary filter decimation rates (PDEC, SDEC) and the primary filter phase (typically 0°). The SINC_LEVEL0 register and the

SINC_LEVEL1 register define the primary and secondary filter order (PORD, SORD) and the primary filter scale (PSCALE). The SINC_BIAS0 register and the SINC_BIAS1 register define the primary filter data offset. The SINC_CLK register defines the CLK0 and CLK1 modulator clock frequencies and can enable synchronization with an external trigger. This register also includes a means to adjust the clock phase if required.

Three registers per group support the primary DMA channels. The SINC_PHEAD0 register and the SINC_PTAIL0 register define the memory addresses for the Group 0 primary output data buffer. The SINC_PPTRx register stores the pointer to the most recent data in the buffer. The PCNT bits in the SINC_LEVELx register set the software decimation rate by defining the number of data transfers per data interrupt (PCNT + 1).

Five registers per channel support the secondary overload detection function. The SINC_LIMITx register defines the maximum and minimum overload threshold, and the SINC_PxSEC_HIST0 register, the SINC_PxSEC_HIST1 register, the SINC_PxSEC_HIST2 register, and the SINC_PxSEC_HIST3 register store the last eight secondary filter outputs before an overload trip. The SINC_LEVEL0 register and the SINC_LEVEL1 register set the secondary filter glitch parameters (LWIN, LCNT) for the channels in the associated group.

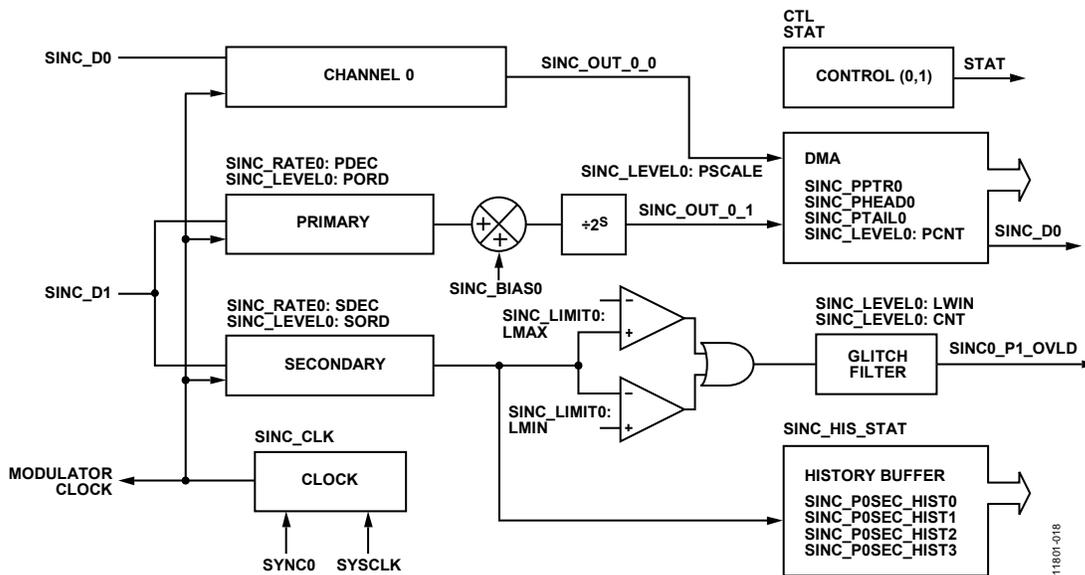


Figure 33. Sinc Register Mapping

SINC FILTER SOFTWARE SUPPORT

The code segment that follows is an example of how to set up the primary and secondary filters for two channels of current feedback. The example was developed for the configuration in Table 3.

Table 3. Configuration of Software Example

Parameter	Symbol	Value	Unit
Core Clock	CCLK	240	MHz
System Clock	SYSCLK	80	MHz
Modulator Clock (1/t _m)	MCLK	8	MHz
Decimation Rate	PDEC	200	
Decimation Frequency	DCLK	40.0	kHz
Software Decimation Rate	SWDEC	4	
PWM Frequency (1/t _{sw})	PWM	16.00	kHz

These code snippets are extracts from working code tested on a closed-loop motor control evaluation platform. The main focus of the code example is the setup and handling of the sinc filter, but setting up the TRU is also required, and is included in the code.

The code example only relates to the sinc filter and cannot work on its own. The code must be included in a complete software project.

The first block of code (Lines[1:18]) defines a number of parameter constants. The next block of code (Lines[19:27]) defines prototype functions and allocates memory for the sinc circular buffer. The function defined on Line 24 implements a prototype for the SINC_DATA0 interrupt service routine.

The TRU setup code block (Lines[28:44]) includes the setup of the trigger routing. Among the triggers is overload detection. Both the handling of overload and the shutdown of PWM is handled by the PWM block. A code example for the PWM setup is not included.

The external hardware trip connects to the TRIP0 input pin, and the internal SINC_Px_OVLD triggers connect to the TRIP1 and TRIP2 trigger slaves.

The sinc setup code block (Lines[45:102]) is the main configuration block. Lines[48:68] set up various group parameters, including order, decimation rate, modulator clock, registers service function for data interrupt, and setup priority.

Lines[60:61] are the initial setting of the overload limits to their full range to avoid a spurious trip when the filter starts. To set the application specific overload limits, a defined sequence must be followed. First, the filter is enabled (Lines[70:71]). To let data propagate through the filter, a 10 μ s delay is provided (Lines[73:74]). At this point, the correct current levels have been determined and the overload interrupt masks can be cleared (Line 76). Finally, the application specific trip levels are set (Lines[78:79]).

Startup and alignment of impulse response to PWM is handled by Lines[90:101]. Utilizing triggers, a general-purpose timer is used to generate the required interrupt. When the delay expires, the general-purpose timer generates a trigger that starts the filter.

The final block of code (Lines[103:125]) includes the interrupt service routine called when the data buffer has been transferred to memory. The SincData0Handler function copies data from the buffer to the motor control variables and calls the control function.

```

1. /*****
2. SINC FILTER SETUP CODE SNIPPETS
3. *****/
4.
5. /* SINC definitions */
6. #define SINC_NUM_SAMPLES_HDR 4 /*
   determines how often a data interrupt is
   generated */
7. #define SINC_NUM_PAIRS      2
8. #define CIRC_BUF_SIZE_HDR
   (SINC_NUM_SAMPLES_HDR*2) /* size of the
   circular buffer */
9. #define SINC_MODCLK      (8000000)
   /* modulator clock frequency */
10. #define S_HDR           23 /*
   Primary scale */
11. #define HDR             200 /*
   primary decimation */
12. #define TRIP_DR         5 /*
   Decimation rate of TRIP filter */
13. #define SINC_N          3 /*
   SINC order */
14. #define LWIN            4 /*
   Glitch window */
15. #define LCNT            4 /*
   Glitch count */
16. #define LMAX            124 /*
   Overload max limit */
17. #define LMIN            1 /*
   Overload min limit */
18.
19. // Function prototypes
20. void SetupTRU(void);
21. void SetupSINC(void);
22.
23. // SINC Data
24. void SincData0Handler(uint32_t, void* );
25. #pragma data_alignment = 2 // Make
   sure buffer starts at an even address
   (16-bit aligned)
26. static int16_t
   sincCircBuffer_HDR[CIRC_BUF_SIZE_HDR];
27. static int16_t ib_sinc_raw_HDR,
   ic_sinc_raw_HDR;
28. void SetupTRU(void){
29. *pREG_TRU0_GCTL |= BITM_TRU_GCTL_RESET;
   // Reset the TRU
30.
31. // Setup TRU for SINC data interrupt.
   Slave is TRU interrupt. Master is
   SINC_DATAx
32. *pREG_TRU0_SSR12 = TRGM_SINC0_DATA0;
   // Slave is TRU0_IRQ0 (12), master is
   SINC0_DATA0
33.
34. // Setup TRU for GP timer enable. Slave
   is TIMER0_TMR2, master is PWM0_SYNC
35. *pREG_TRU0_SSR4 = TRGM_PWM0_SYNC;
   // Slave is TIMER0_TMR2 (4), master is
   PWM0_SYNC
36. // Setup TRU for SINC enable. Slave is
   SINC0_SYNC0, master is TIMER0_TMR2
37. *pREG_TRU0_SSR57 = TRGM_TIMER0_TMR2;
   // Slave is SINC0_SYNC0 (57), master is
   TIMER0_TMR2
38.
39. // Setup TRU for SINC overload detection.
   Slave is PWM0_TRIP_TRIGx, master is
   SINC0_Px_OVLD
40. *pREG_TRU0_SSR49 = TRGM_SINC0_P0_OVLD; //
   Slave is PWM0_TRIP_TRIG1 (49), master:
   PWM0_TRIP_TRIG1
41. *pREG_TRU0_SSR50 = TRGM_SINC0_P1_OVLD; //
   Slave is PWM0_TRIP_TRIG2 (50), master:
   PWM0_TRIP_TRIG2
42.
43. *pREG_TRU0_GCTL |= BITM_TRU_GCTL_EN; //
   Enable TRU
44. }
45. void SetupSINC(void){
46. uint8_t mdiv_temp;
47. // Specify Group 0 Parameters for primary
   and secondary filter
48. *pREG_SINC0_RATE0 =
   (TRIP_DR<<BITP_SINC_RATE0_SDEC) | HDR;
49. *pREG_SINC0_LEVEL0 =
   (0<<BITP_SINC_LEVEL0_PORD) |
   (S_HDR<<BITP_SINC_LEVEL0_PSCALE) |
50. (SINC_NUM_SAMPLES_HDR-1 <<
   BITP_SINC_LEVEL0_PCNT) |
51. (0<<BITP_SINC_LEVEL0_SORD) |
   (LCNT<<BITP_SINC_LEVEL0_LCNT) |
52. (LWIN<<BITP_SINC_LEVEL0_LWIN);
53. // Calculate bias as -DR^N/2. Offset
   compensation due to drift is handled by
   the application code
54. *pREG_SINC0_BIAS0 = -(HDR*HDR*HDR)/2;

```

```

55. // Set up head and tail address of result
    buffers
56. *pREG_SINC0_PHEAD0 =
    (uint32_t)&sincCircBuffer_HDR;
57. *pREG_SINC0_PTAIL0 =
    (uint32_t)&sincCircBuffer_HDR + 2u *
    (CIRC_BUF_SIZE_HDR-1);

58. // Reset overload amplitude detection
    limits to 0 - FullScale
59. *pREG_SINC0_LIMIT0 =
    (0xFFFF<<BITP_SINC_LIMIT0_LMAX) | 0x0000;
    // Limits for filter 0
60. *pREG_SINC0_LIMIT1 =
    (0xFFFF<<BITP_SINC_LIMIT1_LMAX) | 0x0000;
    // Limits for filter 1

61. // Specify modulator clock frequency,
    phase and start-up synchronization
62. mdiv_temp =
    (uint8_t)(fsysclk/SINC_MODCLK);
63. // Scalers for MCLK and specify start
    condition as "Enable and Commence on Next
    Rising Edge"
64. *pREG_SINC0_CLK =
    (mdiv_temp<<BITP_SINC_CLK_MDIV0) |
    (3<<BITP_SINC_CLK_MCEN0);

65. //Install interrupt handler for data IRQ
    and specify priority
66. adi_int_InstallHandler((IRQn_Type)INTR_TR
    U0_INT0, SincData0Handler, NULL, true);
67. NVIC_SetPriority((IRQn_Type)INTR_TRU0_INT
    0, 0);

68. //Enable filters
69. *pREG_SINC0_CTL = (3<<BITP_SINC_CTL_EN3)
    | (3<<BITP_SINC_CTL_EN2) |
70.
    (2<<BITP_SINC_CTL_EN1)
    | (2<<BITP_SINC_CTL_EN0);

71. // Wait 10 µs to let data propagate
    through the filter before setting trip
    limits.
72. for (int i=0; i<500; i++)
73.     asm("nop;");

74. // Specify interrupt masks
75. *pREG_SINC0_CTL |= (BITM_SINC_CTL_EPCNT0
    | BITM_SINC_CTL_EFOVF0 |
    BITM_SINC_CTL_ELIM0);

76. // Now the correct trip limits can be set

77. *pREG_SINC0_LIMIT0 =
    (LMAX<<BITP_SINC_LIMIT0_LMAX) | LMIN; //
    Limits for filter 0
78. *pREG_SINC0_LIMIT1 =
    (LMAX<<BITP_SINC_LIMIT1_LMAX) | LMIN; //
    Limits for filter 1

79. // SINC filter is now set up but not yet
    started. We want to sync modulator clock
    to PWM_SYNC
80. // To do so let PWM SYNC pulse start GP
    timer. GP timer creates a phase shift
    which is half
81. // the duration of the impulse response
    of the filter. In that way the true
    average of the
82. // motor current can be measured.
83. // Set required phase shift in
    TIMER0_TMRx_DLY register.
84. // Do not start timer here. PWM_SYNC
    pulse starts timer through TRU.
85. // Enable timer slave (to start the
    timer) and trigger master (to start SINC
    mod clock)
86. // Note, to enable master trigger both
    TRG_MSK register and valid IRQ mode must
    be set or
87. // trigger won't happen.

88. // Disable Timer First
89. *pREG_TIMER0_STOP_CFG_SET =
    BITM_TIMER_STOP_CFG_TMR02;
90. *pREG_TIMER0_RUN_CLR =
    BITM_TIMER_RUN_SET_TMR02;

91. *pREG_TIMER0_TMR2_CFG =
    ENUM_TIMER_TMR_CFG_PWMSING_MODE |
    ENUM_TIMER_TMR_CFG_IRQMODE1 |
92.
    ENUM_TIMER_TMR_CFG_TRIGSTART |
    ENUM_TIMER_TMR_CFG_POS_EDGE |
93.
    ENUM_TIMER_TMR_CFG_PADOUT_EN
    |ENUM_TIMER_TMR_CFG_EMU_CNT;

94. // Set timer delay to half an impulse
    response: t_mod × (N × DR-2)/2
95. *pREG_TIMER0_TMR2_DLY =
    (uint32_t)((fsysclk/SINC_MODCLK) *
    (SINC_N*HDR-2)/2);
96. // Width register just has to be greater
    than delay register -> multiply with 2
97. *pREG_TIMER0_TMR2_WID =
    *pREG_TIMER0_TMR2_DLY << 1;

```

```

98. // Enable trigger. On next PWM_SYNC pulse
    TMR is started. When delay expires, SINC
    is started
99. *pREG_TIMER0_TRG_MSK &=
    ~BITM_TIMER_TRG_MSK_TMR02;
100. *pREG_TIMER0_TRG_IE |=
    BITM_TIMER_TRG_IE_TMR02;
101. }
102. void SincData0Handler(uint32_t iid,
    void* handlerArg){
103. // Data is stored in a circular buffer
    that wraps around every time it is full.
104. // By keeping the length of the buffer
    and integer times the number of samples
    per
105. // data irq, the buffer never wraps
    around in the middle of a data set. This
    is not
106. // required, but makes handling of the
    buffer easier.
107. // PPTR0 point at the latest data point.
    Data are interleaved: pair0, pair1,...,
    pairx.
108. static int16_t *pData;
109. pData = (uint16_t*)*pREG_SINC0_PPTR0;
110. PMSMctrl_U.ibc_sinc[0] = *(pData);
111. PMSMctrl_U.ibc_sinc[1] = *(pData-1);
112. ib_sinc_raw_HDR =
    PMSMctrl_U.ibc_sinc[0];
113. ic_sinc_raw_HDR =
    PMSMctrl_U.ibc_sinc[1];
114. sMcAlgorithm(); // Call application
    code
115. *pREG_SINC0_STAT |= (1u <<
    BITP_SINC_STAT_PCNT0);
116. }

```