Introduction

This application note introduces the main features of the   
[ADSP-CM402F](http://www.analog.com/ADSP-CM402F?doc=AN-1265.pdf)/[ADSP-CM403F](http://www.analog.com/ADSP-CM403F?doc=AN-1265.pdf)/[ADSP-CM407F](http://www.analog.com/ADSP-CM407F?doc=AN-1265.pdf)/[ADSP-CM408F](http://www.analog.com/ADSP-CM408F?doc=AN-1265.pdf) SINC filters, 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 though 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 Manual](http://www.analog.com/adsp-cm40x_hrm?doc=AN-1265.pdf) and the documentation within the [ADSP-CM40x Enablement Software package](http://www.analog.com/adsp-cm40x-enablement-software?doc=AN-1265.pdf).

Each [ADSP-CM402F](http://www.analog.com/ADSP-CM402F?doc=AN-1265.pdf)/[ADSP-CM403F](http://www.analog.com/ADSP-CM403F?doc=AN-1265.pdf)/[ADSP-CM407F](http://www.analog.com/ADSP-CM407F?doc=AN-1265.pdf)/  
[ADSP-CM408F](http://www.analog.com/ADSP-CM408F?doc=AN-1265.pdf) SINC filter 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

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. It accomplishes this by converting the signal using isolated Σ-Δ modulators and then transmitting 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](http://www.analog.com/AD7403?doc=AN-1265.pdf), to recover a 16-bit digital signal that represents the motor winding current.

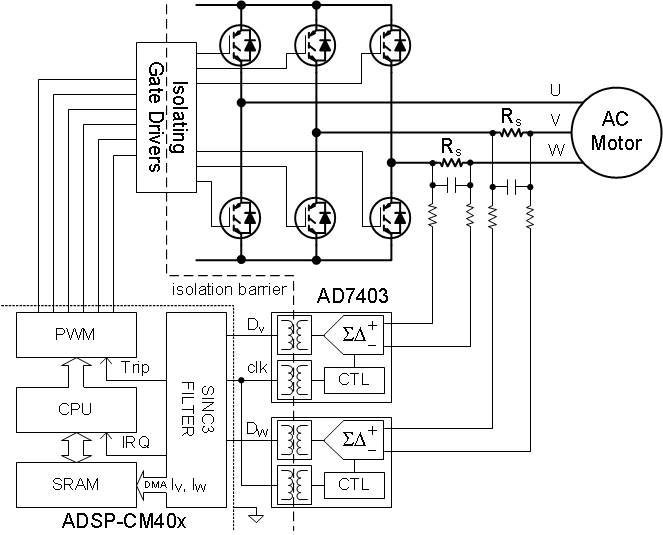


Figure 1. Isolated Current Feedback Using the [AD7403](http://www.analog.com/AD7403?doc=AN-1265.pdf)

Table of Contents

[Introduction 1](#_Toc392507724)

[Motor Current Control Applications 1](#_Toc392507725)

[Revision History 2](#_Toc392507726)

[Sinc Filter Module Overview 3](#_Toc392507727)

[Current Feedback System Overview 4](#_Toc392507728)

[Current Shunt Selection 4](#_Toc392507729)

[Modulator Clock, Primary Filter Decimation, and Data Interrupt Rate Selection 5](#_Toc392507730)

[Aligning sinc impulse response to PWM 5](#_Toc392507731)

[Implementation of impulse response alignment to PWM 7](#_Toc392507732)

[SINC data- and interrupt rate 8](#_Toc392507733)

[Primary Filter Scaling 10](#_Toc392507734)

[Secondary Filter Scaling and Overload Configuration 10](#_Toc392507735)

[SINC Module Fault Detection Functions 12](#_Toc392507736)

[SINC Filter Setup 13](#_Toc392507737)

[Pin Multiplexer Configuration 13](#_Toc392507738)

[Data Buffer Memory Allocation 13](#_Toc392507739)

[Interrupt and Trigger Routing 14](#_Toc392507740)

[Primary and Secondary Filter Configurations 15](#_Toc392507741)

[SINC Filter Software Support 16](#_Toc392507742)

Revision History

**7/7—Rev. A to Rev. ????**

- New sections (Aligning sinc impulse response to PWM and Implementation of impulse response alignment to PWM)

- Updates to most other sections

- Updates to several figures.

**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 PWM modulator block can shut down the PWM inverter without any software intervention. The SINC filter transfers data directly to memory using DMA, and a processor interrupt can be generated when a preset number of samples is ready. This minimizes the software overhead to service the SINC filter after it is configured. The same feedback circuit applies to isolated dc bus voltage feedback or dc bus current measurements.

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 drive 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 that connects external pins to the SINC filter inputs and the trigger routing unit (TRU) that connects SINC output signals to the appropriate peripheral.

W:\WW_Datasheet\Work_In_Progress\AN-1265\Editor_Janeen_Labbe\EMFS_09242013\11801-002.emf

Figure 2. SINC Filter Module Overview

# Current Feedback System Overview

describes 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](http://www.analog.com/AD7403?doc=AN-1265.pdf) modulator generates an isolated bit stream with a pulse density that scales according to the full-scale input voltage range. The SINC filters extract the pulse density information according to the filter order and decimation rate. 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 modulator 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, Icc(p), and the specified maximum input voltage, Vmod(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](http://www.analog.com/AD7403?doc=AN-1265.pdf) modulator is ±200 mV, which is the maximum voltage range within which the modulator specifications are valid. This is lower than the ±320 mV full-scale range (VFS) of the modulator because the linearity and signal-to-noise performance degrades significantly as you get close to the full-scale inputs. The shunt resistance must be less than Vmod(p)/Icc(p) to satisfy these constraints, and the closest nominal shunt value is chosen. For the example in , given that the power stage peak current rating is 8.5 A, the maximum shunt resistance is 23.5 mΩ. The closest nominal shunt value is 20 mΩ, which yields a specified maximum current of 10 A.

W:\WW_Datasheet\Work_In_Progress\AN-1265\Editor_Janeen_Labbe\EMFS_09242013\11801-004.emf

Figure 3. Feedback Current Operating Ranges

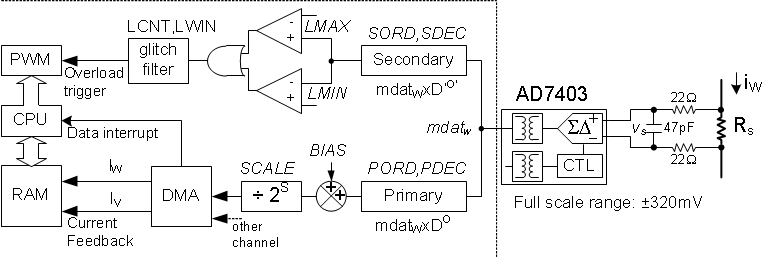


Figure 4. SINC Filter Current Feedback Paths

## Modulator Clock, Primary Filter Decimation, and Data Interrupt Rate Selection

The modulator clock (fM) and 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](http://www.analog.com/AD7403?doc=AN-1265.pdf) is used, the filter order is 3. The equations for the filter frequency response and group delay follow. The frequency response shown in Figure 5 has zeros at frequencies that are even multiples of the decimation frequency (fM/D). Therefore, matching of the decimation frequency to the PWM switching frequency substantially reduces PWM switching harmonics. Other considerations include the increase in the group delay with decimation rate and the maximum decimation limit of the filter.





W:\WW_Datasheet\Work_In_Progress\AN-1265\Editor_Janeen_Labbe\EMFS_09242013\11801-005.emf

Figure 5. SINC Filter Frequency Response

For a given filter order, the decimation rate and filter order are the filter parameters that define the filter signal-to-noise ratio (SNR) and group delay. Figure 6 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 µs and 32 µs in this decimation rate range.

W:\WW_Datasheet\Work_In_Progress\AN-1265\Editor_Janeen_Labbe\EMFS_09242013\11801-006.emf

Figure 6. Secondary Filter SNR

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 sinewave at 1.22 kHz.

## Aligning sinc impulse response to PWM

With the selection of decimation rate and modulator clock the filters characteristics have been set. Equally important is to match the filter characteristic to the application. A SINC filter has memory (states) and the current output depends on not only current input but also previous inputs and outputs. To examine the effect of the SINC filter the impulse response is useful.

A system’s impulse response 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 shown below:

*y* is the output sequence, *x* is the input sequence and *h* is the systems impulse response. The interesting part here is the impulse response *h*, because when know, it can be used to determine the response to any input. For a 3rd order SINC filter with a decimation rate of 16 the impulse response is shown in Figure 7.



Figure 7 Impulse response of 3rd order SINC filter with decimation rate of 16

A 3rd order SINC filter has a hyperbolic impulse response. As can be seen it is a weighted sum, which gives most weight to samples at the center and less weight to samples at the beginning/end. This 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 needs to be eliminated so only the average component remains.

As shown on Figure 8 the there are two instances during a switching period where the average phase current can be measured. Those instances are at the beginning- and center of a switching period. Both instances are indicated by a PWM\_SYNC pulse. Failing to measure at the instant of average phase current would result in noise due to aliasing.

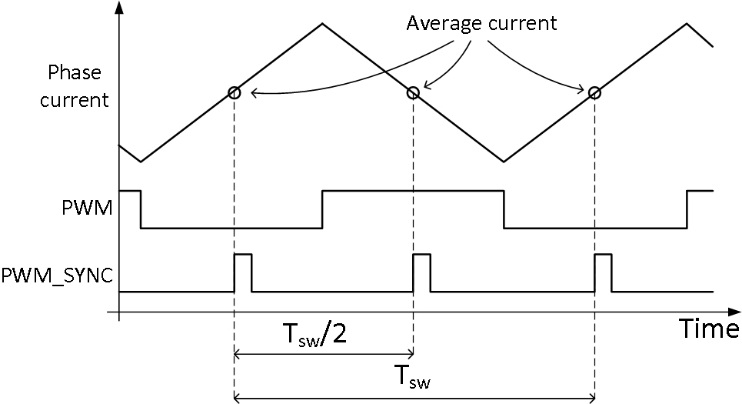


Figure 8 Motor phase currents and relationship to inverter PWM.

With a sample-hold-based converter it is straight forward to measure the currents at the instant they are at their average value by letting the PWM\_SYNC signal trigger the sample-hold-circuit. However, due to the duration of the SINC filter’s impulse response the task of suppressing the switching component requires a different approach when using sigma-delta converters.

Fortunately, 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 7. Furthermore, the switching component is symmetrical around the point of average current. That is, if *x* equally spaced samples taken *before* the instant of average current and added to *x* equally spaced samples taken *after* the instant of average current, result is the average current. The switching component sums to zero.

These properties are utilized extract the average current and at the same time eliminate the switching component completely. If the center pin of the impulse response is aligned with the instant of average current, an equal number of samples are taken before and after the desired sampling point. Since the samples before and after the center pin are given equal weight, and because the switching current is symmetrical around this point, the filter output is the true average current. This technique is illustrated in Figure 9.

Most emphasis (weight) is given around the desired sampling point and the further away for this instant the less weight is given to samples.

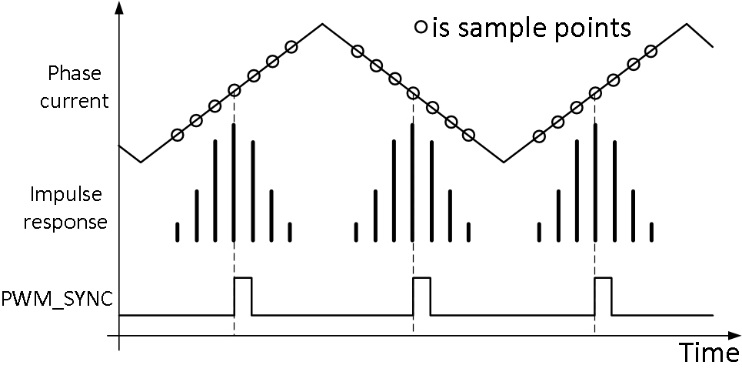


Figure 9 Aligning center pin of impulse response to the instant of average current. Equal number of samples are taken before and after this instant.

Aligning the center pin of the impulse response to the instant of average current is equivalent to aligning the center pin to the PWM\_SYNC pulse. However, to do the alignment correctly, knowledge of the actual impulse response actually is needed.

In most applications, high decimation rates are used but for the sake of simplicity, a decimation rate of 5 is used in the following. Filter order is 3rd. The impulse response of this filter is shown in Figure 10.

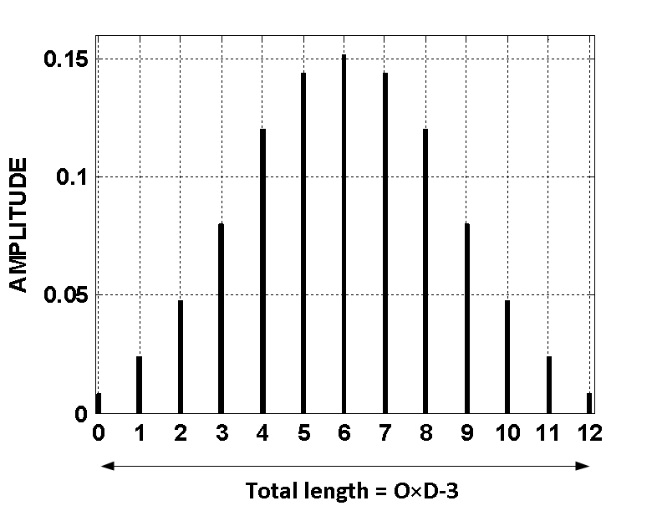


Figure 10 Impulse response of a 3rd order SINC filter with decimation rate of 5

The number of pins in impulse response is:

Therefore, the number of pins in a 3rd order filter with decimation rate of 5 is 13. It is worth noting that the number of pins (samples used by the filter) is much greater than the decimation rate. In this case 5 vs. 13.

The total length of impulse response in number of samples is one less than the number of pins:

For the 3rd order filter with decimation rate of 5, the length is 12. From this, the length of impulse response in seconds can be calculated as:

With *tM* being the period of the modulator clock. This time 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 half-way through the total filter length. Therefore, the time it takes a sample to propagate halfway through the filter must be:

## Implementation of impulse response alignment to PWM

The previous section described how the impulse response must be aligned to PWM to extract the true average motor current. This section will describe how the implementation can be done on ADSP-CM40x.

The task is to align the center pin of the impulse response to PWM\_SYNC. From the previous section, it is known the center pin is found half an impulse response after the first pin of the filter. Measured in seconds that is:

In other words, if feed of input data starts *tM⋅(O⋅D-3)/2 before* PWM\_SYNC, the center pin will align with PWM\_SYNC as shown in Figure 9. Feed of data to the filter is controlled by enable/disable of the modulator clock

Advancing enablement of the modulator clock with respect to PWM\_SYNC is impossible since it would require generation of a “negative delay”. That is, when PWM\_SYNC is needed, PWM has not yet been started. However, instead of advancing start of the modulator clock the exactly the same effect can be achieved by delaying start of the modulator clock by:

As long as the switching period*, Tsw*, is constant, delaying give the same result as advancing.

To generate the delay, CM40x trigger routing unit (TRU) and a general purpose timer is used as shown in Figure 11.

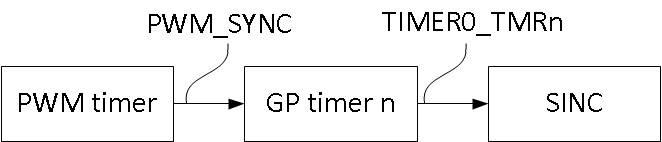


Figure 11 Aligning SINC impulse response to PWM using general purpose timer and triggers.

The PWM timer block outputs a trigger master, PWM\_SYNC, which is routed to the TRU and on to a GP timer’s trigger slave, TIMER0\_TMRn. The GP timer generates a delay with respect to PWM\_SYNC, which brings the impulse response in alignment with PWM. When the delay expires, the GP timer generates a trigger master, TIMER0\_TMRn, which again is routed to the TRU and on to the SINC filters trigger slave, SINC\_SYNC. The sequence is illustrated in the timing diagram in Figure 12,

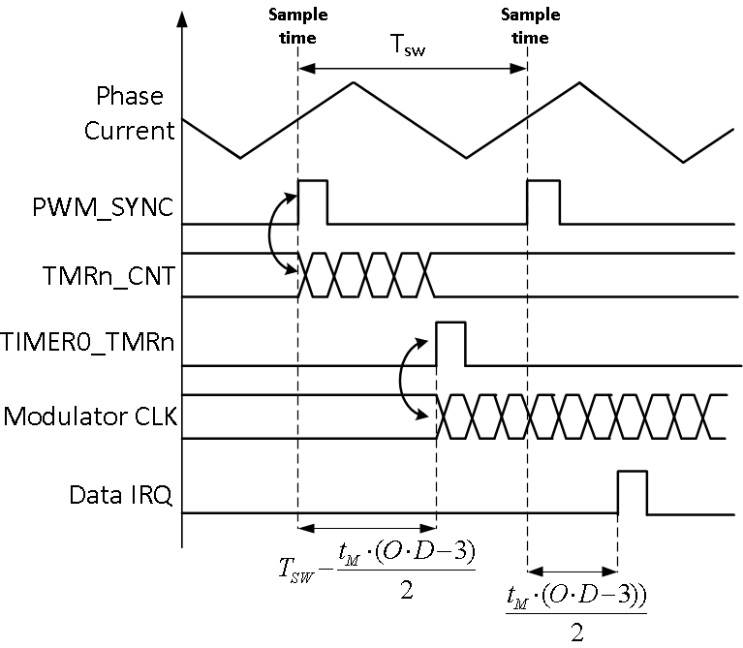


Figure 12 Startup of SINC filter suing GP timer and trigger routing unit.

In Figure 12, note the GP generates a delay of:

This brings the center pin of impulse response in alignment with PWM\_SYNC. Since the center pin is at the midpoint of the impulse response, it takes another half impulse response , with half an impulse response given by:

In Figure 12, note the Data IRQ happens half an impulse response *after* PWM\_SYNC.

Once started, there is no need to realign to PWM\_SYNC. The filter will remain in sync and hence the impulse response will always be aligned to PWM. Therefore, the general-purpose timer used to do the alignment can be reused for other purposes.

## SINC data- and interrupt rate

with the typical PWM switching frequencies used in motor drives. Matching a switching frequency of 16 kHz would require a decimation rate of 625, and the resultant filter group delay would be 94 µs. This decimation rate is well above available values, and the group delay would limit 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 13 describes the alignment between PWM switching, modulator, decimation, and data sampling. The synchronizing pulse (PWM0\_SYNC) from the PWM modulator 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\_DAT0 trigger rate is at the PWM frequency. The information in 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 (MCLK) 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 decimation rate (PDEC) is 125, which sets the filter SNR at 76 dB (>12-bit ENOB) with a group delay of 18.6 µs. The delay corresponds to a phase lag of only 8° at a typical current control loop bandwidth of 1.25 kHz. The modu­lator clock is 10 MHz; therefore, the primary decimation clock frequency is 80 kHz, and a software decimation rate (SWDEC) of 5 synchronizes the sample rate with a 16 kHz PWM frequency (PWM). Tuning of the PWM frequency is possible by adjusting 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



where:  
*PDEC* is the 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 value loaded in the SINC filter control register is one less than 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 | SYSSEL | 3 |  |
| System Clock | SYSCLK | 80 | MHz |
| Modulator Clock Divider | MDIV | 8 |  |
| Modulator Clock (1/TM) | MCLK | 10 | MHz |
| Decimation Rate | PDEC | 125 |  |
| Filter SNR | SNR | 76.0 | dB |
| Filter ENOB | ENOB | 12.3 | Bits |
| Decimation Frequency | DCLK | 80.0 | kHz |
| Filter Group Delay | τd | 18.6 | µs |
| Software Decimation Rate | SWDEC | 5 |  |
| Data Transfer Count | PCNT | 4 |  |
| PWM Frequency (1/TS) | PWM | 16.00 | kHz |
| PWM Period Count | PWMTM | 2500 |  |

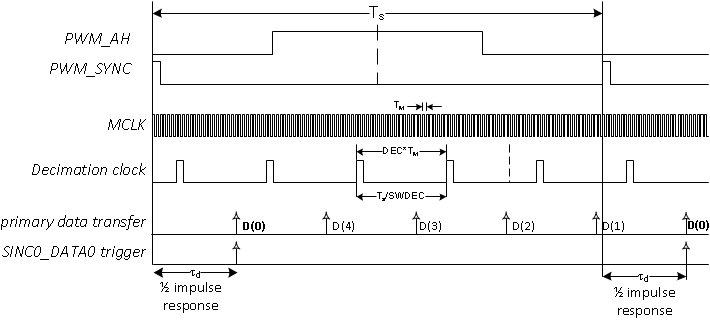


Figure 13. Modulator and Decimation Clock Timing

## Primary Filter Scaling

The SINC filter order (O) and decimation rate (D) set the primary filter dc gain, given by

*Gdc* = *DO*

The unit 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 ±1.0 or as a signed 16-bit integer in the range ±215, depending on interpretation.

The raw filter output is an integer between 0 and DO, where DO/2 aligns with a 50% pulse density corresponding to 0 A. Adding a bias value of −DO/2 to the output sets the correct zero level. Dividing the result by DO/2 would scale the full-scale fractional integer output to ±1. However, for simplicity, the unit has a simple binary scale factor 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 could invert 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 simply 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 . The isolated modulator in this example is the [AD7403](http://www.analog.com/AD7403?doc=AN-1265.pdf).

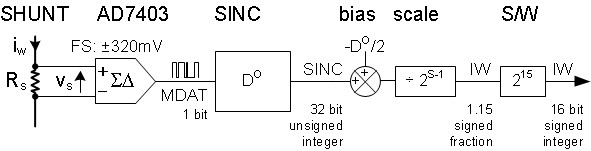


Figure 14. SINC Primary Output Data Scaling

The shunt voltage seen by the modulator is

*vS* = *iW* × *RS*

The isolated modulator expects a bipolar input and generates a 50% pulse density for a 0 V input. The pulse density is a function of the ratio of the input voltage (vs) to the positive full-scale input (*VFS*):



In the case of the [AD7403](http://www.analog.com/AD7403?doc=AN-1265.pdf), the positive full-scale voltage is 320 mV, and the ones density is 81.25% for the specified maximum voltage of 200 mV.

The SINC filter dc gain is DO; therefore, the raw output as a function of the input voltage is



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 DO 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 –DO/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.



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



The SINC output equation when reading the data as a signed integer adds a scale factor of 215.



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



## 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 trigger routing unit (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 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 µs is possible using a decimation rate of 10, as shown in Figure 15. The SINC filter also filters out inverter switching noise, as shown in Figure 16. In this figure, a 10 A peak test waveform has injected 16 A noise pulses of 1.5 µs duration and 16 A overload pulses of 40 µs 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.

A faster response is possible at a lower decimation rate, but as seen from Figure 17, 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 18 illustrates the SNR at high (10) and low (5) decimation rates and the noise margin for the trip signal.

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). illustrates 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. The figure illustrates the ability of the filter to reject short noise pulses on the analog input. In this example, the noise pulse is 1.5 µs 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 are accessed directly through the device peripheral memory infrastructure.

W:\WW_Datasheet\Work_In_Progress\AN-1265\Editor_Janeen_Labbe\EMFS_09242013\11801-009.emf

Figure 15. Secondary Filter Overload Detection

W:\WW_Datasheet\Work_In_Progress\AN-1265\Editor_Janeen_Labbe\EMFS_09242013\11801-010.emf

Figure 16. Overload Detection with Decimation Rate of 10

W:\WW_Datasheet\Work_In_Progress\AN-1265\Editor_Janeen_Labbe\EMFS_09242013\11801-011.emf

Figure 17. False Overloads Detected with Decimation Rate of 5

W:\WW_Datasheet\Work_In_Progress\AN-1265\Editor_Janeen_Labbe\EMFS_09242013\11801-012.emf

Figure 18. Secondary Filter Gain Curves for Decimations of 5 and 10

W:\WW_Datasheet\Work_In_Progress\AN-1265\Editor_Janeen_Labbe\EMFS_09242013\11801-013.emf

Figure 19. Overload Detection with 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 values are within the range of 0 to DO. The negative full-scale current maps to 0, and the positive full-scale current maps to DO. Setting the minimum and maximum trip levels to 1 and DO − 1 enables the maximum range of the trip function. The transfer function shown in the bottom graph of for a decimation rate of 10 and a 20 mΩ 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 gets to the full-scale limits.

The overload circuit operates a little 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 could be set to operate within the range of 5 A to 6.4 A. The LMAX and LMIN settings in this case would be 700 counts and 300 counts. Attaining precise trip settings using lower decimation rates is more difficult.

## SINC Module Fault Detection Functions

The section describes the selection of the various filter parameter settings required to achieve the desired primary and secondary filter performance. 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 FIFO error if it fails to transfer data before the filter writes new data. The ESATx and EFOVFx bits in the SINC\_CTL register mask the SINC0\_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. Once 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 modulator registers. outlines the interconnections required between the SINC filter block and the CPU, SRAM, PWM, and external pins to capture motor current feedback signals.

There are four steps to 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.

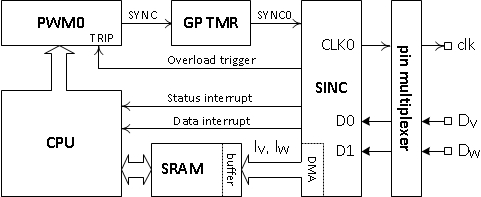


Figure 20. SINC Filter System Configuration

## Pin Multiplexer Configuration

The pin multiplexer connects the front-end modulator clock and data pins to the SINC module. There are two available modulator clock outputs (SINC0\_CLK0 and SINC0\_CLK1) and four available SINC data input pins (SINC0\_D0, SINC0\_D1, SINC0\_D2, and SINC0\_D3). The PORT\_MUX registers control 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](http://www.analog.com/adsp-cm40x-enablement-software?doc=AN-1265.pdf), automatically generate C code to enable the user port selections. Figure 21 is a snapshot of the PinMux64.jar application window.

W:\WW_Datasheet\Work_In_Progress\AN-1265\Editor_Janeen_Labbe\EMFS_09242013\11801-015.emf

Figure 21. Pin Mux Selection Tool

## 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.

W:\WW_Datasheet\Work_In_Progress\AN-1265\Editor_Janeen_Labbe\EMFS_09242013\11801-016.emf

Figure 22. Data Buffer Organization

## Interrupt and Trigger Routing

Figure 23 describes the SINC filter interconnection with other peripheral functions using interrupt and trigger signals. The SINC\_STAT is the single processor interrupt signal of the SINC filter module. The trigger routing unit (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, TMRn, the TRU synchronizes the SINC filter modulator and decimation clocks with the PWM modulator frequency to meet the timing defined in Figure 13. 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 PWM modulator TRIP1 input to enable overcurrent protection. The TRIP0 input connects to the external trip signal only. The PWM modulator as well as TRIP0 and TRIP1 inputs must be configured to accept these triggers. There are two interrupt triggers produced by an overload fault: the SINC\_STAT interrupt connected directly to the CPU and the PWM\_TRIP1 interrupt generated by the SINC overload trigger.

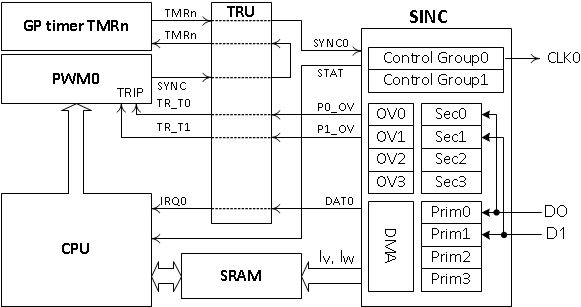


Figure 23. SINC Filter Trigger Routing

## 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.

describes the assignment of filter and system parameters to Group0 registers. The Group1 registers organization 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. SINC\_RATE0 and SINC\_RATE1 set the primary and secondary filter decimation rates (PDEC, SDEC) and the primary filter phase (normally 0°). SINC\_LEVEL0 and SINC\_LEVEL1 define the primary and secondary filter order (PORD, SORD) and the primary filter scale (PSCALE). BIAS0 and BIAS1 define the primary filter data offset. The SINC\_CLK register defines the CLK0 and CLK1 modulator clock frequency 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. SINC\_PHEAD0 and SINC\_TAIL0 define the memory addresses for the Group0 primary output data buffer. The SINC\_PPTR0 register stores the pointer to the most recent data in the buffer. The PCNT bits in the SINC\_LEVEL0 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. SEC\_LIMIT0 defines the maximum and minimum overload threshold, and P0SEC\_HIST0, P0SEC\_HIST1, P0SEC\_HIST2, and P0SEC\_HIST3 store the last eight secondary filter outputs before an overload trip. The SINC\_LEVEL0 and SINC\_LEVEL1 registers set the secondary filter glitch parameters (LWIN, LCNT) for the channels in the associated group.

W:\WW_Datasheet\Work_In_Progress\AN-1265\Editor_Janeen_Labbe\EMFS_09242013\11801-018.emf

Figure 24. 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. These code snippets are extracts from working code tested on a closed-loop motor control evaluation platform. Main focus of the code example is setup and handling of the SINC filter but setting up the Trigger Routing Unit is required too. An example of TRU setup is also included below. Please note, the TRU is setup using device drivers while the SINC filter is setup using direct register access. Analog Devices Inc. provide a complete set of device drivers for all peripherals, including SINC filter. The device driver adds some overhead but simplifies the programming while direct register setup give the most efficient code.

The code example below only relates to the SINC filter and cannot work on its own It needs to be included in a complete SW project.

The first block of code (Lines[1:19]) defines a number of parameter constants. The next block of code (Lines[20:27]) defines prototype functions and allocates memory for the SINC circular buffer. The function defined on Line 26 implements prototype for the SINC\_DATA0 interrupt service routine.

The SetupTRU code block (Lines[28:44]) includes setup of all the trigger routing. Among the triggers is overload detection. Handling of overload and shut down of PWM is handled by the PWM block. Code example for this is not included below.

The external hardware trip connects to TRIP0, and the internal SINC\_Px\_OVLD triggers connect to TRIP1. The TRIP1 interrupt is one of the interrupts generated by a SINC overload. The overload can also generate a SINC\_STAT interrupt.

The SetupSINC code block (Lines[45:102]) is the main configuration block. Line [48:68] , , modulator clock, registers service function for data interrupt and set up priority.

Lines[60:61] is 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, Line [70:71]. To let data propagate through the filter a 10us delay is provided, Line [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, Line [78:79].

Startup and alignment of impulse response to PWM is handled by Line [90:101]. Utilizing triggers, a General Purpose timer is used to generate the required interrupt. When the delay expires, the GP timers 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. /\* SINC definitions \*/
5. #define SINC\_NUM\_SAMPLES\_HDR 4 /\* determines how often a data interrupt is generated \*/
6. #define SINC\_NUM\_PAIRS 2
7. #define CIRC\_BUF\_SIZE\_HDR (SINC\_NUM\_SAMPLES\_HDR\*2) /\* size of the circular buffer \*/
8. #define SINC\_MODCLK (8000000) /\* modulator clock freq \*/
9. #define S\_HDR 23 /\* Primary scale \*/
10. #define HDR 200 /\* primary decimation \*/
11. #define TRIP\_DR 5 /\* Decimation rate of TRIP filter \*/
12. #define SINC\_N 3 /\* SINC order \*/
13. #define LWIN 4 /\* Glitch window \*/
14. #define LCNT 4 /\* Glitch count \*/
15. #define LMAX 124 /\* Overload max limit \*/
16. #define LMIN 1 /\* Overload min limit \*/
17. // TRU definitions
18. #define TRU\_DEV\_NUM 0
19. #define ADI\_TRU\_REQ\_MEMORY 4u
20. // Function prototypes
21. void SetupTRU(void);
22. void SetupSINC(void);
23. // SINC Data
24. void SincData0Handler(uint32\_t, void\* );
25. // SINC handler and data buffers
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. ADI\_TRU\_RESULT result;
30. static uint8\_t TruDevMemory[ADI\_TRU\_REQ\_MEMORY]; // TRU Device memory
31. static ADI\_TRU\_HANDLE hTru; // Handle to TRU
32. result = adi\_tru\_Open (TRU\_DEV\_NUM, &TruDevMemory[0], ADI\_TRU\_REQ\_MEMORY, &hTru);
33. // Setup TRU for SINC data interrupt. Slave is TRU interrupt. Master is SINC\_DATAx
34. result = adi\_tru\_TriggerRoute (hTru, TRGS\_TRU0\_IRQ0, TRGM\_SINC0\_DATA0); // HDR
35. // Setup TRU for GP timer enable. Slave is TIMER0\_TMRx, master is PWM0\_SYNC
36. result = adi\_tru\_TriggerRoute (hTru, TRGS\_TIMER0\_TMR2, TRGM\_PWM0\_SYNC);
37. // Setup TRU for SINC enable. Slave is SINC0 SYNC0, master is TIMER0\_TMRx
38. result = adi\_tru\_TriggerRoute (hTru, TRGS\_SINC0\_SYNC0, TRGM\_TIMER0\_TMR2);
39. // Setup TRU for overload detection enable. Slave is PWM0\_TRIP\_TRIG1, master is SINC0\_Px\_OVLD
40. /W
41. result = adi\_tru\_TriggerRoute (hTru, TRGS\_PWM0\_TRIP\_TRIG1, TRGM\_SINC0\_P0\_OVLD);
42. result = adi\_tru\_TriggerRoute (hTru, TRGS\_PWM0\_TRIP\_TRIG1, TRGM\_SINC0\_P1\_OVLD);
43. result = adi\_tru\_Enable (hTru, true); // 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 comp. due to drift is handled by the application code
54. \*pREG\_SINC0\_BIAS0 = -(HDR\*HDR\*HDR)/2;
55. // Setup 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 & startup 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\_TRU0\_INT0, SincData0Handler, NULL, true);
67. NVIC\_SetPriority((IRQn\_Type)INTR\_TRU0\_INT0, 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);

1. // Wait 10us to let data propagate through the filter before setting trip limits.
2. for (int i=0; i<500; i++)
3. asm("nop;");
4. // Specify interrupt masks
5. \*pREG\_SINC0\_CTL |= (BITM\_SINC\_CTL\_EPCNT0 | BITM\_SINC\_CTL\_EFOVF0 | BITM\_SINC\_CTL\_ELIM0);
6. // Now the correct trip limits can be set
7. \*pREG\_SINC0\_LIMIT0 = (LMAX<<BITP\_SINC\_LIMIT0\_LMAX) | LMIN; // Limits for filter 0
8. \*pREG\_SINC0\_LIMIT1 = (LMAX<<BITP\_SINC\_LIMIT1\_LMAX) | LMIN; // Limits for filter 1

1. // SINC filter is now set up but not yet started. We want to sync modulator clock to PWM\_SYNC
2. // To do so let PWM SYNC pulse start GP timer. GP timer creates a phase shift which is half
3. // the duration of the impulse response of the filter. In that way the true average of the
4. // motor current can be measured.
5. // Set required phase shift in TIMER0\_TMRx\_DLY register.
6. // Do not start timer here. PWM\_SYNC pulse starts timer through TRU.
7. // Enable timer slave (to start the timer) and trigger master (to start SINC mod clock)
8. // Note, to enable master trigger both TRG\_MSK register and valid IRQ mode must be set or
9. // trigger won't happen.
10. // Disable Timer First
11. \*pREG\_TIMER0\_STOP\_CFG\_SET = BITM\_TIMER\_STOP\_CFG\_TMR02;
12. \*pREG\_TIMER0\_RUN\_CLR = BITM\_TIMER\_RUN\_SET\_TMR02;
13. \*pREG\_TIMER0\_TMR2\_CFG = ENUM\_TIMER\_TMR\_CFG\_PWMSING\_MODE | ENUM\_TIMER\_TMR\_CFG\_IRQMODE1 |
14. ENUM\_TIMER\_TMR\_CFG\_TRIGSTART | ENUM\_TIMER\_TMR\_CFG\_POS\_EDGE |
15. ENUM\_TIMER\_TMR\_CFG\_PADOUT\_EN |ENUM\_TIMER\_TMR\_CFG\_EMU\_CNT;
16. // Set timer delay to half an impulse response: t\_mod\*(N\*DR-3)/2
17. \*pREG\_TIMER0\_TMR2\_DLY = (uint32\_t)((fsysclk/SINC\_MODCLK) \* (SINC\_N\*HDR-3)/2);
18. // Width register just has to be greater than delay register -> multiply with 2
19. \*pREG\_TIMER0\_TMR2\_WID = \*pREG\_TIMER0\_TMR2\_DLY << 1;
20. // Enable trigger. On next PWM\_SYNC pulse TMR is started. When delay expires, SINC is started
21. \*pREG\_TIMER0\_TRG\_MSK &= ~BITM\_TIMER\_TRG\_MSK\_TMR02;
22. \*pREG\_TIMER0\_TRG\_IE |= BITM\_TIMER\_TRG\_IE\_TMR02;
23. }
24. void SincData0Handler(uint32\_t iid, void\* handlerArg){
25. // Data are stored in a circular buffer that wraps around every time it is full.
26. // By keeping the length of the buffer and integer times the number of samples per
27. // data irq the buffer never wraps around in the middle of a data set. This is not
28. // required, but makes handling of the buffer easier.
29. // PPTR0 point at the end of the latest data set so we need to find the beginning.
30. // Data are interleaved: pair0, pair1,..., pairx. First, move pointer to beginning
31. // of latest data set. That's SINC\_NUM\_PAIRS-1u addresses ahead of current address.
32. // Then move to the very first sample belonging to this data frame. That's
33. // SINC\_NUM\_PAIRS \* pcnt ahead.
34. static int16\_t \*pData;
35. static uint32\_t pcnt;

1. pcnt = (\*pREG\_SINC0\_LEVEL0 & BITM\_SINC\_LEVEL0\_PCNT) >> BITP\_SINC\_LEVEL0\_PCNT;
2. pData = (uint16\_t\*)\*pREG\_SINC0\_PPTR0;
3. pData = pData - (SINC\_NUM\_PAIRS-1u);
4. pData = pData - (SINC\_NUM\_PAIRS \* pcnt);
5. PMSMctrl\_U.ibc\_sinc[1] = \*pData;
6. PMSMctrl\_U.ibc\_sinc[0] = \*(pData+1);
7. ib\_sinc\_raw\_HDR = PMSMctrl\_U.ibc\_sinc[0];
8. ic\_sinc\_raw\_HDR = PMSMctrl\_U.ibc\_sinc[1];
9. sMcAlgorithm(); // Call application code
10. \*pREG\_SINC0\_STAT |= (1u << BITP\_SINC\_STAT\_PCNT0);
11. }

|  |  |
| --- | --- |
| ©2013 Analog Devices, Inc. All rights reserved. Trademarks and   registered trademarks are the property of their respective owners.  AN11801-0-11/13(A) | **word_signoff** |