ANALOG DEVICES

AN-1265
アプリケーション・ノート

ADSP-CM402F/ADSP-CM403F/ADSP-CM407F/ADSP-CM408F SINC フィルタと AD7401A を使用した絶縁型モーター・コントロール帰還

著者: Dara O'Sullivan、Jens Sorensen、Aengus Murray

はじめに

このアプリケーション・ノートでは、高性能モーター・コントロール・アプリケーションを中心に ADSP-CM402F/ADSP-CM403F/ADSP-CM407F/ADSP-CM408F SINC フィルタの主な機能を紹介します。

このアプリケーション・ノートの目的は、SINC フィルタ・モジュールの主要機能を説明し、アナログ・デバイセズが提供する SINC フィルタ・ドライバの使い方についてガイダンスを提供することです。SINC フィルタの機能とコンフィギュレーション・レジスタの詳細については、ADSP-CM40x Mixed-Signal Control Processor with ARM Cortex-M4 Hardware Reference Manual と ADSP-CM40x Enablement Software package 内のドキュメントを参照してください。

各 ADSP-CM402F/ADSP-CM403F/ADSP-CM407F/ADSP-CM408F SINC フィルタは、電流シャント、信号をデジタル化して絶縁する変調器、ビット・ストリームをデコードしてコントローラに渡す SINC フィルタを全て含むモーター電流帰還サブシステムの一部です。このアプリケーション・ノートでは、SINC フィルタの設定方法を説明します。

モーター電流制御アプリケーション

図 1 に、インバータ給電モーター駆動向けの絶縁型電流帰還システムの簡略回路図を示します。このシステムでは、スイッチング・パワー・インバータが発生する高電圧の共通信号の電流シャント両端に生じるアナログ信号を絶縁するという課題を克服しています。これは、絶縁型 Σ-Δ 変調器を使って信号を変換し、絶縁バリアを跨いでデジタル信号を伝えることにより実現します。

Σ-Δ 変調器は、入力電圧の関数として変調されたビット・ストリームを発生し、この信号を絶縁障壁を跨いで低電圧側にあるフィルタ回路へ伝えます。SINC フィルタは、AD7401A のような 2 次変調器からのビット・ストリームをフィルタ処理して、モーター巻線電流に相当する 16 ビット・デジタル信号を再生します。

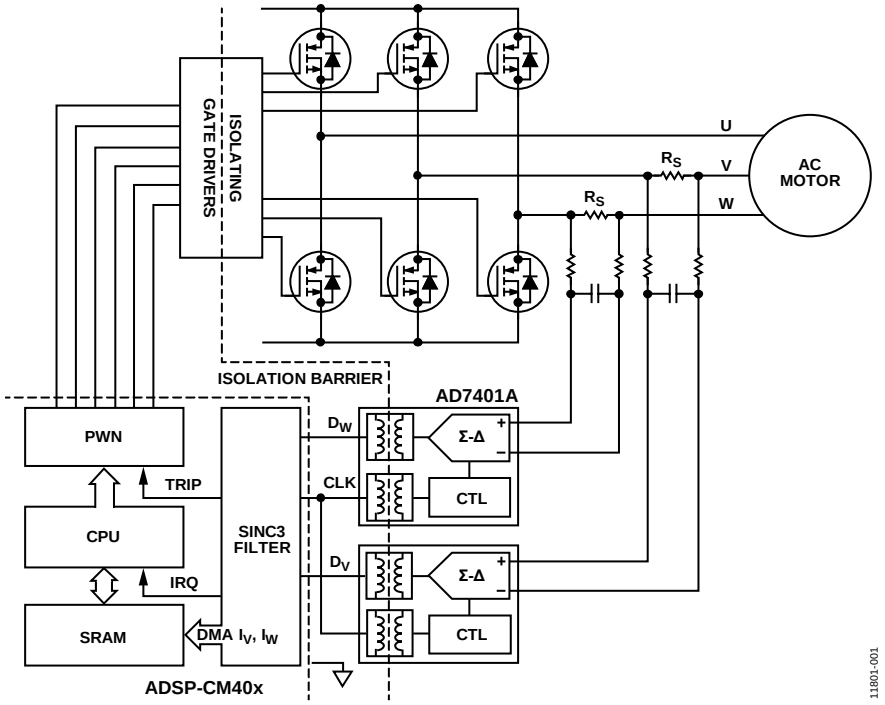

図 1. AD7401A を使用した絶縁型電流帰還

Rev. A

アナログ・デバイセズ株式会社

本　　社／〒105-6891　東京都港区海岸 1-16-1　ニューピア竹芝サウスタワービル
電話 03（5402）8200
大阪営業所／〒532-0003　大阪府大阪市淀川区宮原 3-5-36　新大阪トラストタワー
電話 06（6350）6868

目次

はじめに 1
モーター電流制御アプリケーション 1
改訂履歴 2
Sinc フィルタ・モジュールの概要 3
電流帰還システムの概要 4
電流シャントの選択 4
変調器クロック、プライマリ・フィルタ・デシメーション、データ割込みレートの選択 5
プライマリ・フィルタのスケーリング 7
セカンダリ・フィルタのスケーリングと過負荷の設定 7
SINC モジュールの故障検出機能 9
SINC フィルタのセットアップとソフトウェア・ドライバ機能 10
ピン・マルチプレクサの設定 10
データ・バッファ・メモリの割当て 10
割込みとトリガのルーティング 11
プライマリ・フィルタとセカンダリ・フィルタの設定 12
SINC フィルタ・ソフトウェアのサポート 13

改訂履歴

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 フィルタ・モジュールの概要

SINC フィルタ・ブロックは 2 つの機能を実行します。モーター制御アルゴリズム向けの忠実度の高い帰還信号の発生と、故障状態での過負荷電流の迅速な検出です。過負荷故障信号を PWM 変調器ブロックに接続すると、ソフトウェアの介入なしに PWM インバータをシャットダウンさせることができます。SINC フィルタは DMA を使ってデータをメモリへ直接転送し、設定済みのサンプル数になったときプロセッサ割込みを発生させることができます。これにより、設定後の SINC フィルタを処理するソフトウェアのオーバーヘッドを小さくしています。同じ帰還回路を、絶縁型 DC バス電圧帰還つまり DC バス電流測定に適用します。

図 2 に、SINC フィルタ・モジュールのブロック図を示します。SINC フィルタ・モジュールには 4 つの SINC フィルタ対があり、各対には帰還信号フィルタ機能と入力に接続されるデジタル・ビット・ストリーム上の過負荷検出機能が組み込まれています。フィルタ・イネーブル機能は、SINC フィルタ対を 2 つのコンフィギュレーション・レジスタ・グループの 1 つに割り当てて、フィルタ・パラメータを設定します。モーター駆動では、同じ方法で設定された複数の電流フィルタまたは電圧フィルタを必要とするものと想定しています。 SINC フィルタ・モジュールは、2 つのフィルタ対の 1 グループを各モーターに割り当て、2 つのモーターの制御をサポートします。プライマリ・フィルタ設定は、フィルタの次数、デシメーション・レート、オフセット・バイアス、ゲイン・スケーリングです。セカンダリ・フィルタ設定は、フィルタ次数、デシメーション・レート、過負荷トリップ・レベル、グリッチ・フィルタ設定値です。

その他の設定機能としては、変調器クロック周波数、割込みマスク、DMA データ転送などがあります。SINC フィルタのセットアップに必要なその他の制御ペリフェラルは、外部ピンを SINC フィルタ入力に接続するポート・コントローラと、SINC フィルタの出力信号を該当するペリフェラルへ接続するトリガ・ルーティング・ユニット （TRU)です。

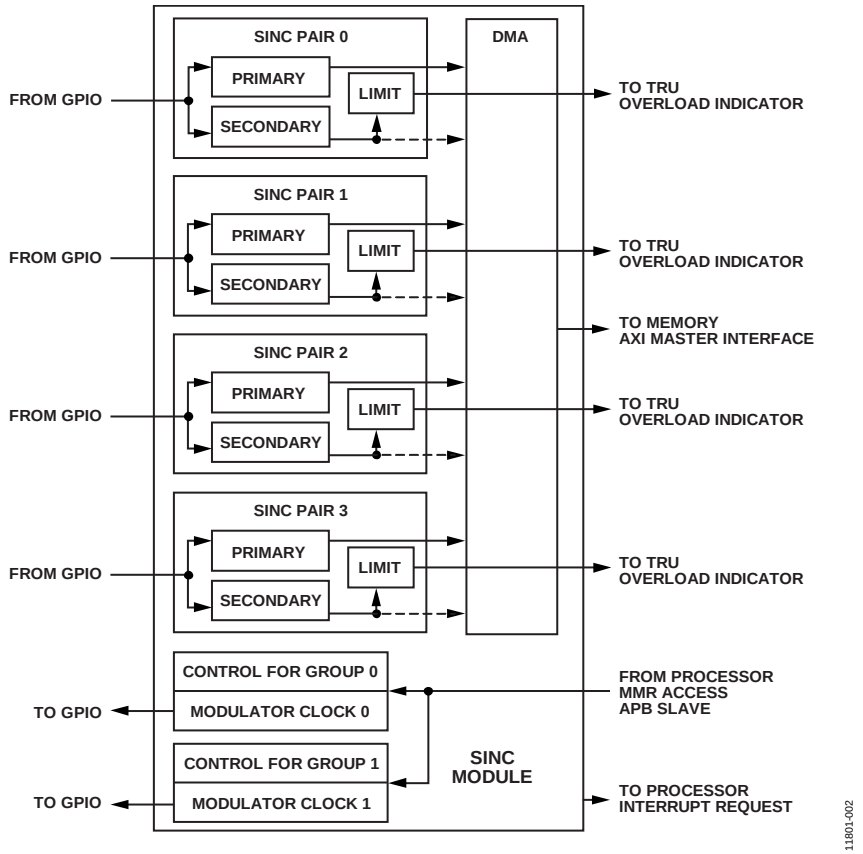

図 2. SINC フィルタ・モジュールの概要

電流帰還システムの概要

図 4 に、電流帰還システムの主要な要素を示します。シャントは、巻線電流をシャント抵抗で変換される電圧信号として検出します。AD7401A 変調器は、フルスケール入力電圧範囲でスケーリングされるパルス密度を持つ絶縁型ビット・ストリームを発生します。SINC フィルタは、フィルタ次数とデシメーション・レートに従ってパルス密度情報を取り出します。プライマリ・フィルタ・パラメータは、精度とその他のバイアスについてフィルタを最適化し、スケーリング・ブロックはデータを 16 ビット符号付き整数に変換してからメモリへ転送します。セカンダリ・パラメータはフィルタ速度を最適化し、出力からの信号は過負荷状態を検出するデジタル・コンパレータへ渡されます。上限コンパレータと下限コンパレータは電流過負荷を検出し、グリッチ・フィルタは特定ウインドウ （LWIN）内で最小過負荷カウント （LCNT）値になると、過負荷トリガ信号を発生します。過負荷トリガは、モーター・インバータを駆動する PWM 変調器に対するトリップ入力信号です。DMA 転送エンジンは、巻線電流データがメモリ内で準備できると、割込み信号を発生してアルゴリズムの実行を開始させます。

電流シャントの選択

帰還の定義に必要なシステム仕様は、変調器に対するピーク制御電流 I_{cc}（p）と最大入力電圧 V_{mod}（max)です。パワー・インバータのピーク電流能力は一般に制御電流範囲を決定しますが、他の考慮事項もあります。AD7401A 変調器の最大動作電圧規定値は ±200 mV で、変調器仕様が有効となる最大電圧範囲です。これは、±320 mV の変調器のフルスケール範囲 （V_{FS}）より狭くなっています。狭くなっている理由は、フルスケール入力に近づくと直線性と信号対ノイズ性能が大幅に低下するためです。これらの制約を満たすためには、シャント抵抗を V_{mod}（p）/I_{cc}（p）より小さくすることが必要で、最も近い公称シャント値を選択します。例えば図 3 で、電力段のピーク電流定格が 8.5 A の場合、最大シャント抵抗は 23.5 mΩ になります。最も近い公称シャント値は 20 mΩ で、これにより最大電流規定値は 10 A になります。

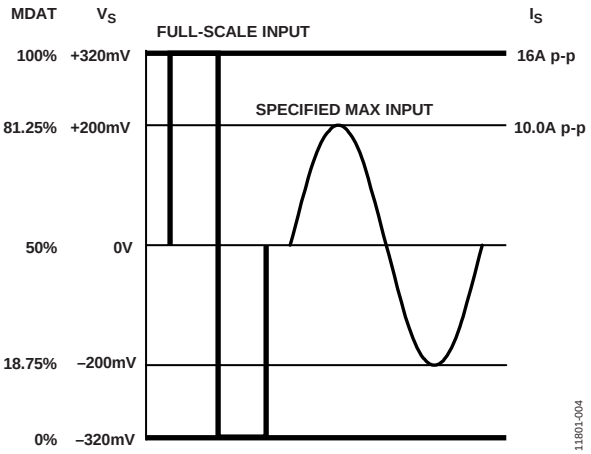

図 3. 帰還電流の動作範囲

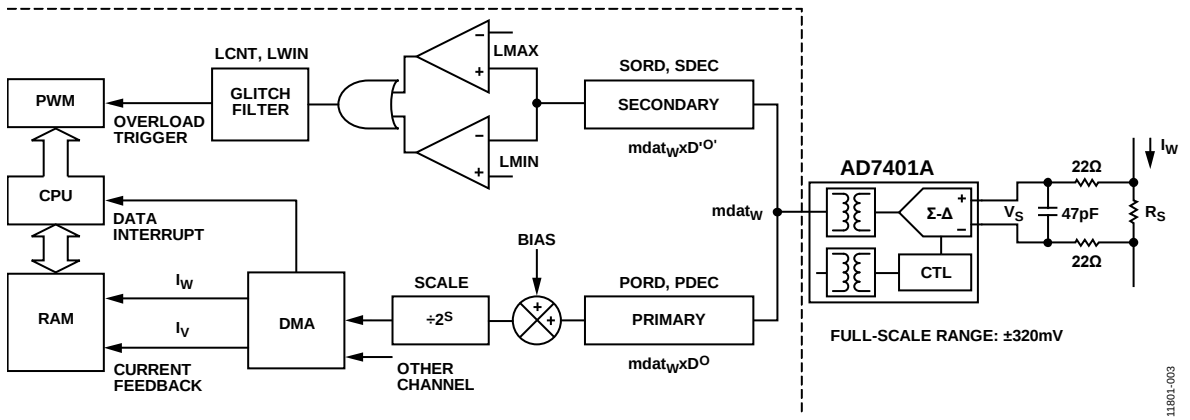

図 4. SINC フィルタの電流帰還パス

変調器クロック、プライマリ・フィルタ・デシメーション、データ割込みレートの選択

変調器クロック （f_M）とデシメーション・レート （D）は、SINC フィルタ性能を決定するパラメータです。フィルタ次数（O）は一般に、フロントエンド変調器の次数より 1 次高くなっています。このため、AD7401A を使用する場合、フィルタ次数は 3 になります。フィルタ周波数応答と群遅延の式を以下に示します。図 5 の周波数応答は、デシメーション周波数 （f_M/D）の偶数倍の周波数にゼロ点を持っています。このため、デシメーション周波数と PWM スイッチング周波数を一致させると、PWM スイッチング高調波が大幅に減衰します。その他、デシメーション・レートとフィルタの最大デシメーション限界による群遅延の増加などを考慮する必要があります。

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

$$T_d=\left(\frac{D-1}{2}\right)\frac{0}{f_M}$$

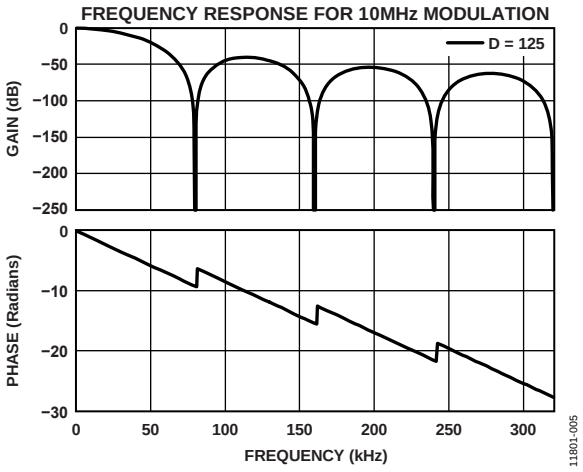

図 5. SINC フィルタの周波数応答

デシメーション・レートとフィルタ次数は、与えられたフィルタ次数に対して、フィルタの信号対ノイズ比 （SNR）と群遅延を決定するフィルタ・パラメータです。図 6 と表 1 に、SNR の変化、有効ビット数 （ENOB）および 10 MHz 変調器クロックの 3 次数フィルタでのデシメーション・レートに対する群遅延を示します。 デシメーション・レートは、11 ビット ～ 14 ビットの ENOB （さらに 67 dB ～ 86 dB の SNR）を実現するためには、85 ～ 210 の範囲である必要があります。これは電流帰還で要求されるフィルタ性能範囲です。このデシメーション・レート範囲での群遅延は、12 µs ～ 32 µs です。

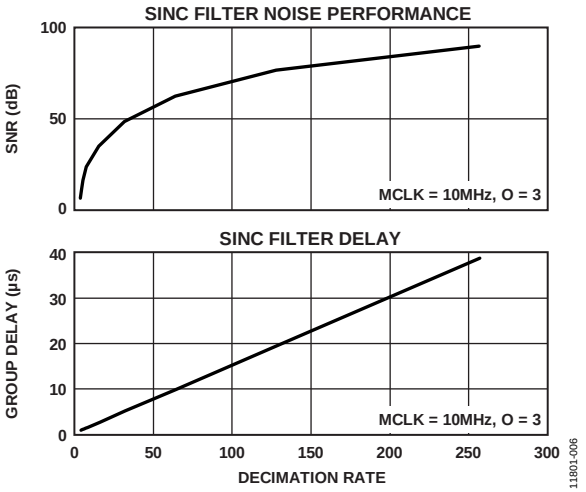

図 6. セカンダリ・フィルタの SNR

表 1. デシメーション・レートに対する SNR、ENOB、群遅延 [1]

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

[1]テスト条件は、1.22 kHz、±200 mV の正弦波。

SINC フィルタのデシメーション・レートとモーター駆動で使用される一般的な PWM スイッチング周波数を一致させることは不可能です。16 kHz のスイッチング周波数に一致させるためには、625 のデシメーション・レートが必要になり、その結果、フィルタ群遅延は 94 µs になります。このデシメーション・レートは可能な値をはるかに超えており、群遅延が電流ループの帯域幅を制限することになります。代わりに、デシメーション・レートを PWM 周波数の倍数に設定すると、群遅延は小さくなり、目標とするフィルタ SNR を実現することができます。制御アルゴリズムでは、PWM スイッチングと一致するデシメーション周波数の約数の周波数でデータをサンプリングします。このソフトウェア・デシメーション・プロセスでは、複数のデータ・サンプルをメモリ内のサーキュラ・バッファに転送して、バッファがフルになったときの割込みに応答して、直前のデータ・サンプルを読出す方法が採られています。DMA エンジンがプライマリ SINC フィルタからデータ・メモリへデータを転送し、SINC コントロール・ユニットが固定数のサンプルを転送するごとにトリガを発生します。

図 7 に、PWM スイッチング、変調器、デシメーション、データ・サンプリングの間のタイミング関係を示します。PWM 変調器からの同期パルス （PWM0_SYNC）が変調器クロックの開始を PWM 周波数に同期させます。デシメーション周波数は変調器クロックの約数で、かつ PWM 周波数の倍数です。SINC0_DAT0 トリガ・レートは、PWM 周波数に一致します。

表 2 の情報は、デシメーション・レートと PWM スイッチング周波数の選択手順を示しています。表内の最初の 3 行は、コア・クロックとペリフェラル・クロックに対するチップ・レベルの設定値です。最大コア・クロック・レートは 240 MHz で、通常、システム （ペリフェラル） クロック周波数の偶数倍です。SINC フィルタ変調器クロック （MCLK）は MDIV レジスタ・フィールド値に基づいてシステム・クロックから生成され、値は 5 MHz ～ 20 MHz の範囲に制限されます。プライマリ・デシメーション・レート （PDEC）は 125 で、この値により、フィルタ SNR は 76 dB （>12 ビット ENOB）に、群遅延は 18.6 µs に、それぞれ設定されます。この遅延は、代表的な電流制御ループ帯域幅 1.25 kHz で 8°の位相遅れに相当します。変調器クロックは 10 MHz であるため、プライマリ・デシメーション・クロック周波数は 80 kHz になり、5 のソフトウェア・デシメーション・レート （SWDEC） で、サンプリング・レートを 16 kHz の PWM 周波数 （PWM）に同期化します。SINC フィルタ・デシメーション・レートを調整して、PWM 周波数をチューニングすることができます。

変調器クロック、PWM 周波数、ハードウェアおよびソフトウェア・デシメーション・レートの間の関係は次式で与えられます。

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

ここで、
PDEC はハードウェア・デシメーション・レート、
SWDEC はソフトウェア・デシメーション・レートです。

ハードウェアおよびソフトウェア・デシメーション・レートは整数である必要があります。SINC フィルタの PCNT レジスタ・フィールド値は、ソフトウェア・デシメーション・レートを設定します。SINC フィルタ・コントロール・レジスタにロードされる PCNT 値は、割込みが発生するまでのサンプル遅延数から 1 を減算した値です。PWM_TM0 レジスタは PWM スイッチング周波数を設定するため、サンプリング・タイミングも設定します。

表 2. デシメーション・レートの選択

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 ($1/T_M$)	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	τ_d	18.6	µs
Software Decimation Rate	SWDEC	5	
Data Transfer Count	PCNT	4	
PWM Frequency ($1/T_S$)	PWM	16.00	kHz
PWM Period Count	PWMTM	2500	

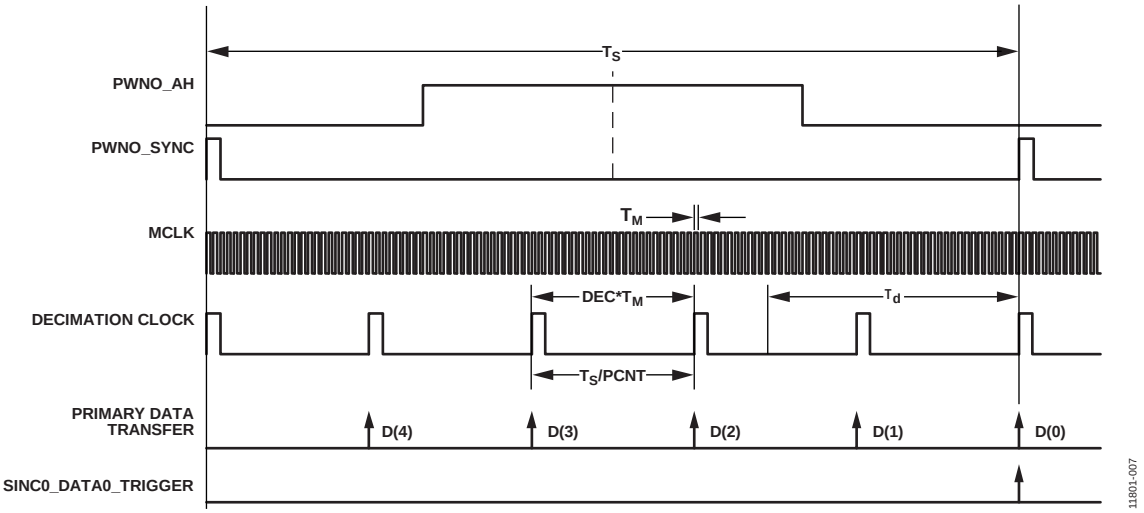

図 7. 変調器クロックとデシメーション・クロックのタイミング

プライマリ・フィルタのスケーリング

SINC フィルタ次数 （O）とデシメーション・レート （D）は、次式によりプライマリ・フィルタの DC ゲインを設定します。

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

このユニットは、メモリへ転送する前にデータを 16 ビット符号付き整数へ変換するための出力スケーリング機能とバイアス機能を持っています。有効なデータ・フォーマットは、変換に応じて範囲 ±1.0 の 16 ビット小数 （S.15）または範囲 $\pm 2^{15}$ の符号付き 16 ビット整数です。

フィルタの直接出力は 0 ～ D^O の整数です。ここで、$D^O/2$ は 50% パルス密度に一致し、0 A に相当します。出力に$-D^O/2$ のバイアス値を加算すると、正しいゼロ・レベルが設定されます。結果を $D^O/2$ で除算すると、フルスケール小数出力を ±1 にスケーリングしますが、簡単化のため、このユニットはシンプルなバイナリ・スケール・ファクタを持っています。ここで、スケール・ファクタ （S）を選択してゲインを 1.0 付近に設定します。DMA エンジンは、スケーリングに無関係に、出力データの下位 16 ビットだけを転送するため、精度の低下を回避するためには正しいスケーリングが不可欠です。データ・オーバーフローを防止するため出力データを飽和させます。これにより、スケール・ファクタの選択が正しくないと、出力信号の極性が反転することがあります。フィルタは、飽和した場合オーバーフロー故障フラグを設定します。

データの浮動小数点への変換では、単純に電流シャント・ゲインの逆数でスケーリングし、フィルタの DC ゲインとスケール・ファクタの間の不一致を調整しています。

帰還スケーリングの計算

シャント電流からメモリ内のデータ・ワードまでの最終システム・ゲインは、システム内のすべての要素のゲインから求めます （図 8 参照）。この例の絶縁型変調器は AD7401A です。

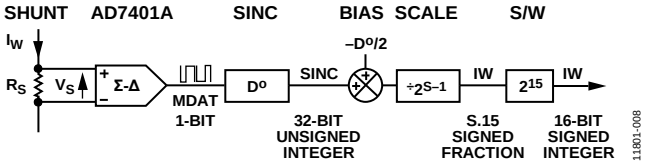

図 8. SINC プライマリ出力データのスケーリング

変調器から見たシャント電圧は、

$$v_S = i_W \times R_S$$

絶縁型変調器は バイポーラ入力を受け入れ、0 V 入力に対して 50% のパルス密度を発生します。パルス密度は、入力電圧 （v_S）の正フルスケール入力 （V_{FS}）に対する比の関数として次式のように表されます。

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

AD7401A の場合、正のフルスケール電圧は 320 mV で、最大電圧規定値 200 mV に対してパルス密度は 81.25%になります。

SINC フィルタの DC ゲインは D^O であるため、入力電圧の関数としての直接出力は、

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

この DC スケーリングは、セカンダリ・フィルタ出力に適用され、最大セカンダリ・デシメーション・レートにより直接の出力データ範囲が 16 ビット符号なし整数に制限されます。セカンダリ出力は、負のフルスケール入力で 0 に、正のフルスケール入力で D^O になります。

プライマリ出力パス内のバイアス機能とスケーリング機能が SINC データ上のバイアスを除去して、データを 16 ビット符号付き整数へ再スケーリングします。バイポーラ入力範囲で変調器に対する SINC 出力のオフセットを除去するためには、バイアス値は $-D^O/2$ である必要があります。再スケーリングでは、SINC 出力ワードから該当するビット範囲を選択します。

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

スケール・ファクタ （S）は、最大小数出力を 1.0 に設定する必要があります。すなわち、

$$\frac{D^O}{2^S} << 1 \therefore S >> \frac{\ln(D^O)}{\ln 2}$$

データを符号付き整数として読出す際の SINC 出力の式には、スケール・ファクタ 2^{15} が追加されます。

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

この場合の実際の巻線電流 （i_w）の関数としての電流測定値は、

$$IW = i_W \times \left(\frac{R_S}{0.32}\right)\left(\frac{D^O}{2^S}\right) \times 2^{15}$$

セカンダリ・フィルタのスケーリングと過負荷の設定

セカンダリ SINC フィルタのデータ出力は、過負荷コンパレータとグリッチ・フィルタに直接接続されます（図 4 参照）。セカンダリ・フィルタのデシメーション・レートは、故障状態に対する高速応答を実現するため、プライマリ・フィルタのデシメーション・レートより大幅に小さく設定されています。プロセッサ・トリガ・ルーティング・ユニット （TRU）は過負荷トリップ信号を PWM 変調器のシャットダウン入力に接続して故障をクリアします。また TRU は、その他のクリティカルな回路要素をシャットダウンするときに使用する外部 GPIO などのその他のソースに過負荷信号を接続することもできます。

代表的なパワー・インバータ・スイッチは数マイクロ秒の短絡にしか耐えられないため、過負荷回路は比較的短い検出ウインドウを持つ必要があります。SINC フィルタは 3 デシメーション・サイクル内にステップ入力へ応答できるため、デシメーション・レート 10 を使って 3 µs 内の応答が可能です（図 9 参照）。また SINC フィルタは、インバータのスイッチング・ノイズも除去します（図 10 参照）。

この図では、10 A のピーク・テスト波形に継続時間 1.5 µs の 16 A ノイズ・パルスと継続時間 40 µs の 16 A 過負荷パルスを注入しています。フィルタは狭いノイズ・パルスを除去しますが、回路は 16 A 過負荷パルスを検出します。このテストでの最大および最小のトリップ・レベルは、セカンダリ SINC 出力で±16 A に相当します。

デシメーション・レートを低くすると応答を高速化できますが、図 11 から分かるように、±10 A のシンプルな正弦波テスト電流の場合でもセカンダリ SINC 出力がトリップ・レベルを超えます。デシメーション・レート = 5 での高い SINC フィルタ・ノイズは、複数の偽トリップ信号を発生します。図 12 に、デシメーション・レートが高い場合（10）と低い場合（5）の SNR、およびトリップ信号のノイズ・マージンを示します。

セカンダリ出力グリッチ・フィルタは、トリップ・カウント・ウインドウ（WCNT）を使って最小カウント（LCNT）より短い継続時間のトリップを除去して、短い過負荷トリップをなくします。図 13 に、グリッチ・フィルタがデシメーション・レート = 5 のときトリガされる擬過負荷を除去する様子を示します。ただし、応答時間には 3 サイクルの遅延が加わります。このため、デシメーション・レートが低くなっても応答時間は短くなりません。この図は、アナログ入力上の狭いノイズ・パルスを除去するフィルタの能力を示しています。この例では、ノイズ・パルス継続時間は 1.5 µs です。

セカンダリ SINC フィルタは履歴バッファのセットを内蔵しており、トリップ発生直前の 8 個のデータ・サンプルを診断目的に格納しています。履歴レジスタ内のデータは、デバイス・ペリフェラル・メモリ・インフラストラクチャを使って直接アクセスされます。

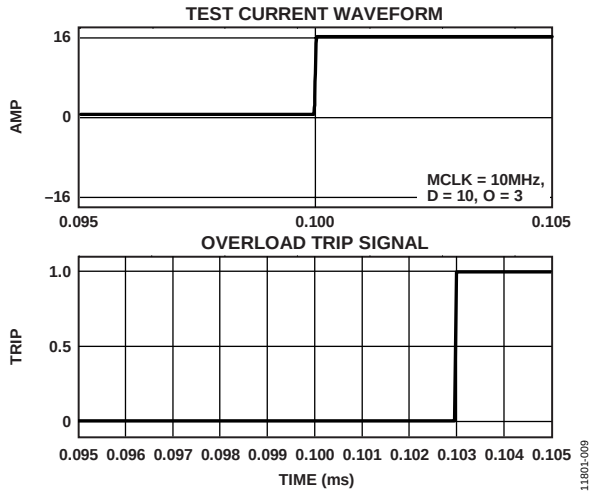

図 9. セカンダリ・フィルタの過負荷検出

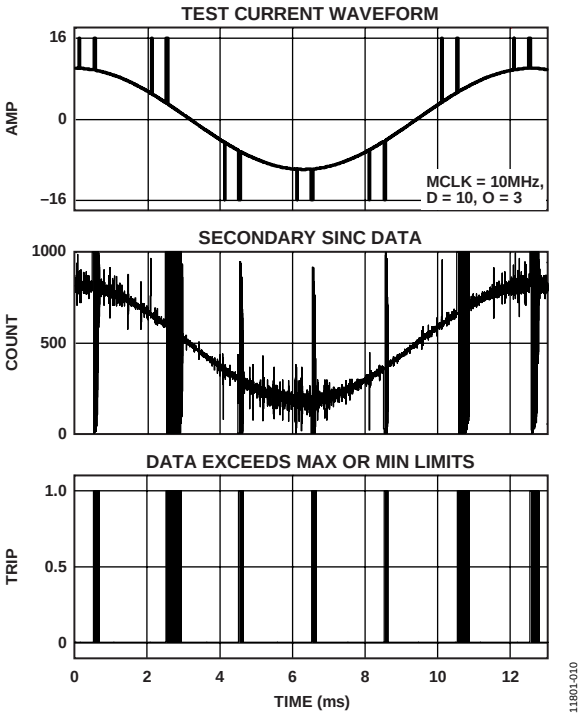

図 10. デシメーション・レート = 10 での過負荷検出

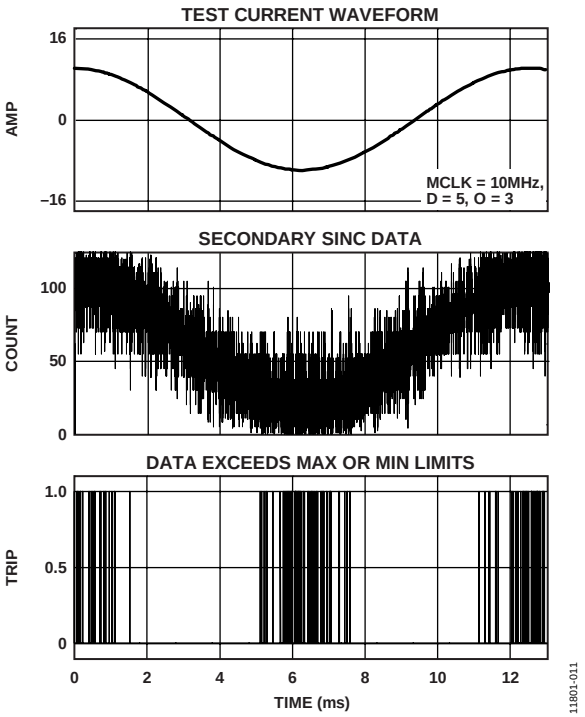

図 11. デシメーション・レート = 5 での偽過負荷検出

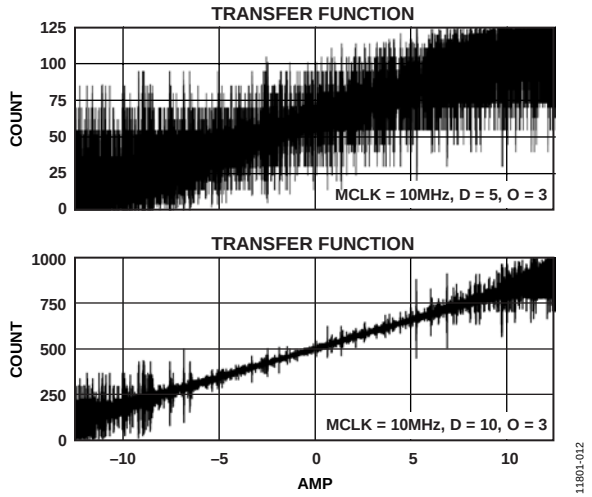

図 12. デシメーション = 5 および 10 でのセカンダリ・フィルタのゲイン・カーブ

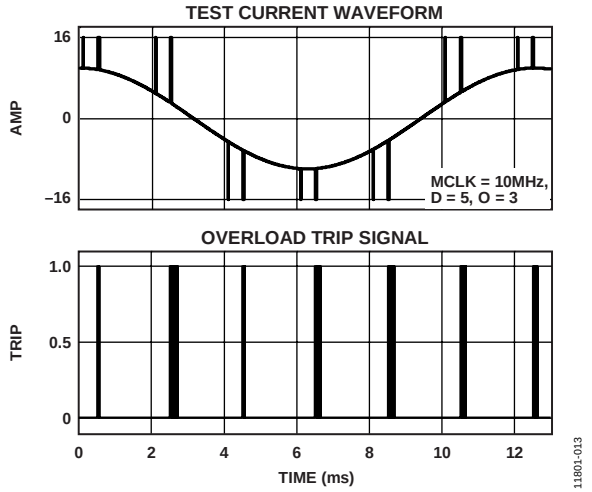

図 13. 過負荷検出（デシメーション・レート = 5、WCNT = 4 および LCNT = 4 のグリッチ・フィルタ）

セカンダリ・フィルタのスケーリングとトリップ・レベル

セカンダリ・フィルタには出力スケーリングが追加されていないため、有効な最小値と最大値は 0 ～ D^O の範囲内です。負のフルスケール電流は 0 に、正のフルスケール電流は D^O に、それぞれ対応します。最小と最大のトリップ・レベルをそれぞれ 1 と $D^O - 1$ に設定すると、トリップ機能の最大範囲が有効になります。図 12 の下側のグラフで、デシメーション・レート = 10 および 20 mΩ のシャントに対して表示された伝達関数は、10 A 入力に対するノイズ・ピークがフィルタの最大 （1000 カウント） 出力および最小 （0 カウント） 出力の範囲内であることを示しています。LMIN と LMAX のトリップ・レベルをそれぞれ 1 カウントと 999 カウントに設定して、10 A のピーク電流に対する偽トリップを防止してください。トリップが発生する実際の電流レベルは、11 A ～ 16 A （フルスケール）の範囲です。電流がフルスケール限界に近づくほどトリップが発生する可能性は大きくなります。

過負荷回路は、規定の変調器入力範囲内では多少正確に動作します。前のケースでは、5 A 入力でのピーク・ノイズは 700 カウントであり、これは 6.4 A と等価です。このため、5 A ～ 6.4 A の範囲内で動作するようにトリップを設定することができます。このケースでは、LMAX 設定値と LMIN 設定値はそれぞれ 700 カウントと 300 カウントになります。低いデシメーション・レートを使った正確なトリップ設定はさらに困難になります。

SINC モジュールの故障検出機能

セカンダリ・フィルタのスケーリングと過負荷の設定のセクションでは、所望のプライマリおよびセカンダリ・フィルタの性能を実現するために必要とされる種々のフィルタ・パラメータ設定値の選択について説明しています。過負荷故障の他に、SINC モジュールはチップ・インフラストラクチャを過負荷にする不正なフィルタ設定値から発生するデータ故障もチェックします。

プライマリ・フィルタは、出力バイアスとスケーリングの不正な設定値が存在する場合、出力データ飽和を検出します。フィルタ DMA エンジンは、フィルタが新しいデータを書き込む前にデータの転送に失敗すると、FIFO エラーを検出します。SINC_CTL レジスタの ESATx ビットと EFOVFx ビットは、飽和故障と FIFO 故障での SINC0_STAT 割込みの発生をマスクします。

SINC フィルタのセットアップとソフトウェア・ドライバ機能

フィルタが使えるようになる前に、SINC フィルタ・モジュール、信号ルーティング、データ・バッファを設定する幾つかのステップがあります。設定が済むと、DMA エンジンはプライマリ・フィルタ・データをメモリへ自動的に転送し、セカンダリ制限機能は過負荷の場合に PWM モジュールをシャットダウンさせます。システムはデータがレディになると割込みを発生するため、プロセッサは制御アルゴリズムを実行して、PWM 変調器レジスタを更新することができます。図 14 に、SINC フィルタ・ブロックと CPU、SRAM、PWM、モーター電流帰還信号を取り込む外部ピンの間に必要な相互接続の概要を示します。

SINC フィルタを使う電流帰還の設定には次の 4 ステップがあります。

1. ピン・マルチプレクサの設定
2. データ・バッファ・メモリの割当て
3. 割込みとトリガ・ルーティングの接続
4. プライマリ・フィルタとセカンダリ・フィルタの設定

このセクションではこれらのステップを詳しく説明し、セットアップ手順、およびシステムと SINC フィルタ・コントロール・レジスタを設定するアナログ・デバイセズのデバイス・ドライバ機能について説明します。

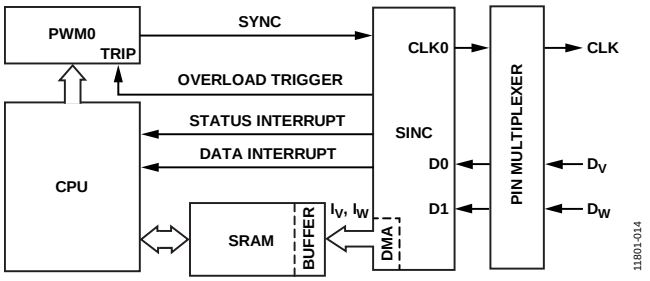

図 14. SINC フィルタ・システムの設定

ピン・マルチプレクサの設定

ピン・マルチプレクサは、フロントエンド変調器クロックとデータ・ピンを SINC モジュールに接続します。2 つの変調器クロック出力 （SINC0_CLK0 と SINC0_CLK1）と、4 本の SINC データ入力ピン （SINC0_D0、SINC0_D1、SINC0_D2、 SINC0_D3）があります。PORT_MUX レジスタは、マルチプレクスされた各ピンに対する 4 つの入力信号または出力信号からこれらのピンへの割当てを制御します。ADSP-CM40x Enablement Software package で供給する Java アプリケーション・プログラムである PinMux64.jar と PinMux32.jar が、ユーザーのポート選択をイネーブルする C コードを自動的に生成します。図 15 に、PinMux64.jar アプリケーション・ウインドウのスナップショットを示します。

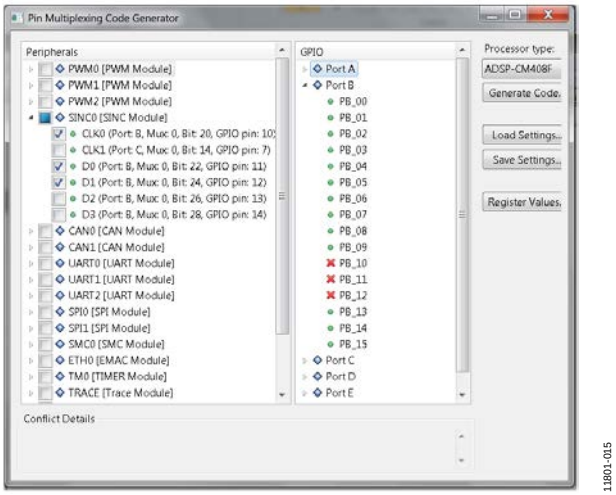

図 15. ピン・マルチプレクサ選択ツール

データ・バッファ・メモリの割当て

制御アルゴリズムがデータを使用できるようにするため、プライマリ・フィルタのデータ・バッファを定義して、メモリ・スペースを割り当てる必要があります。ソフトウェア・デシメーション・レートと帰還チャンネル数によりバッファ・サイズが決まります。データは、グループごとにチャンネル順に並べられます。直前のデータ・セットに対するポインタが SINC_PPTRx レジスタに格納されます。割込みとトリガのルーティングのセクションで説明するデバイス・ドライバが、バッファとポインタの管理を行います。

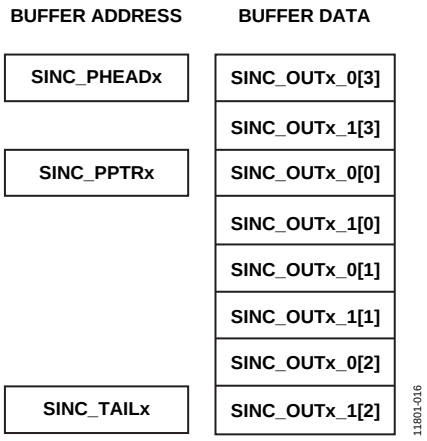

図 16. データ・バッファの構造

割込みとトリガのルーティング

図 17 に、SINC フィルタと、割込み信号およびトリガ信号を使用するその他のペリフェラル機能との相互接続を示します。SINC_STAT は、SINC フィルタ・モジュールのシングル・プロセッサ割込み信号です。トリガ・ルーティング・ユニット（TRU）は、その他のトリガ信号を SINC フィルタ・モジュールのペリフェラルおよびプロセッサ割込みへ接続します。トリガ・マスター・アドレスを TRU のトリガ・スレーブ・レジスタにロードすると、ルーティングが接続されます。

TRU は SINC フィルタ変調器クロックとデシメーション・クロックを PWM 変調器周波数に同期させて、図 7 で定義するタイミングを満たすようにします。TRU は SINC フィルタ・データ転送トリガを制御ソフトウェア割込みに接続して、制御アルゴリズムの実行を開始させます。

TRU は両方の SINC 過負荷トリガを PWM 変調器の TRIP1 入力に接続して、過電流保護機能をイネーブルします。TRIP0 入力は外部トリップ信号にのみ接続されます。PWM 変調器、TRIP0 入力、TRIP1 入力は、これらのトリガを受け入れるように設定する必要があります。過負荷故障時に生成される割込みトリガは 2 つあり、CPU に直接接続される SINC_STAT 割込みと、SINC 過負荷トリガで生成される PWM_TRIP1 割込みです。SINC フィルタ・ソフトウェアのサポートのセクションの終わりに示すコード・セグメントには、これらの接続を実現するデバイス・ドライバ呼出しが含まれています。

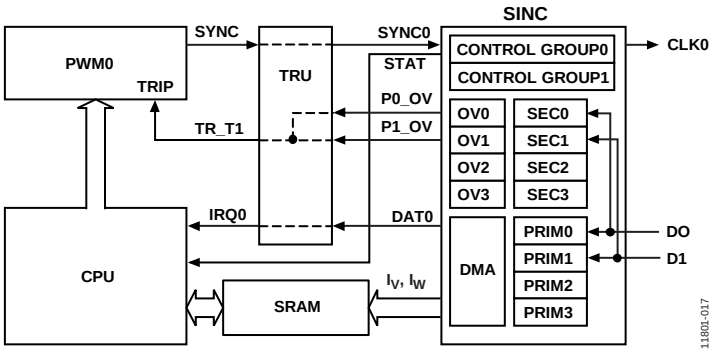

図 17. SINC フィルタ・トリガのルーティング

プライマリ・フィルタとセカンダリ・フィルタの設定

フィルタ・チャンネルはグループで構成されています。これは、2 個または 3 個の帰還信号が同じフィルタ・パラメータを必要とすることが一般的だからです。SINC モジュールは、2 グループのコンフィギュレーション・レジスタを持っています。任意のグループ内の複数のチャンネルが同じクロックを共用し、フィルタ次数、デシメーション・レート、スケーリング、バイアスなどの共通のフィルタ・パラメータを持っています。例外は過負荷制限レジスタと履歴レジスタです。これらはチャンネルごとに構成されています。フィルタ・チャンネルをイネーブルすると、設定グループが割り当てられます。コンフィギュレーション・レジスタは、変調器クロック、フィルタ・パラメータ、DMA データ転送、過負荷検出を決定します。

図 18 に、フィルタ・パラメータとシステム・パラメータの Group0 レジスタへの割当てを示します。Group1 レジスタの構成は同じです。SINC_CTL レジスタは、各チャンネルをイネーブルして制御グループを割り当てます。推奨プロセスは、フィルタ・グループを設定した後にグループ内のチャンネルをイネーブルすることです。SINC_CTL レジスタは、 SINC_STAT 割込みのマスクも行います。システム・ステータス・レジスタ SINC_STAT は、故障ステータスとデータ・トリガ・カウント・ステータスを通知します。

グループあたり 3 個のレジスタとクロック・レジスタによって、プライマリ・フィルタ・パラメータとセカンダリ・フィルタ・パラメータを決定します。SINC_RATE0 と SINC_RATE1 は、プライマリ・フィルタとセカンダリ・フィルタのデシメーション・レート （PDEC、SDEC） とプライマリ・フィルタの位相 （通常は 0°）を設定します。SINC_LEVEL0 と SINC_LEVEL1 は、プライマリ・フィルタとセカンダリ・フィルタの次数 （PORD、SORD）とプライマリ・フィルタのスケール （PSCALE）を決定します。BIAS0 と BIAS1 は、プライマリ・フィルタのデータ・オフセットを決定します。SINC_CLK レジスタは CLK0 および CLK1 変調器クロック周波数を決定し、外部トリガとの同期をイネーブルすることができます。このレジスタも、必要に応じてクロック位相を調整する手段を持っています。

グループあたり 3 個のレジスタがプライマリ DMA チャンネルをサポートします。 SINC_PHEAD0 と SINC_TAIL0 は、Group0 プライマリ出力データ・バッファのメモリ・アドレスを決定します。SINC_PPTR0 レジスタは、バッファ内の直前のデータに対するポインタを格納します。SINC_LEVEL0 レジスタの PCNT ビットは、データ割込みあたりのデータ転送数 （PCNT + 1）を指定してソフトウェア・デシメーション・レートを設定します。

チャンネルあたり 5 個のレジスタが、セカンダリ過負荷検出機能をサポートします。SEC_LIMIT0 は最大過負荷スレッショールドと最小過負荷スレッショールドを決定し、P0SEC_HIST0、P0SEC_HIST1、P0SEC_HIST2、P0SEC_HIST3 は、過負荷トリップ直前の 8 個のセカンダリ・フィルタ出力を格納します。SINC_LEVEL0 レジスタと SINC_LEVEL1 レジスタは、対応するグループ内のチャンネルに対してセカンダリ・フィルタのグリッチ・パラメータ （LWIN、LCNT）を設定します。

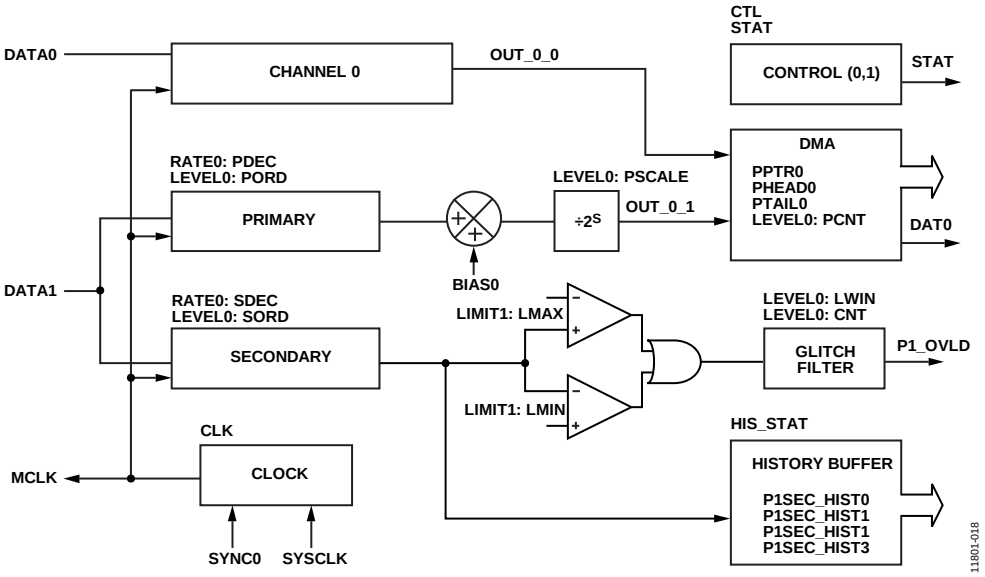

図 18. SINC レジスタのマッピング

SINC フィルタ・ソフトウェアのサポート

次に示すコード・セグメントは、電流帰還の 2 つのチャンネルに対するプライマリ・フィルタとセカンダリ・フィルタの設定方法の例です。これらのコードは、クローズド・ループ・モーター制御評価用プラットフォームでテストした動作コードからの抜粋です。デバイス・ドライバによりある程度オーバーヘッドが増えますが、SINC モジュール・レジスタの設定が大幅に簡素化されます。関数呼出し定数名がこのドキュメントで使用したパラメータ名に一致しているため、大部分のコードは一目瞭然です。

コードの最初のブロック （行 1 ～ 21） は、パラメータ定数の数を決定します。行 9 と行 10 は、プライマリ SINC データのデータ・バッファ・サイズを決定します。コードの次のブロック（行 22～36）は、プロトタイプ関数を設定し、メモリを割り当てます。行 28 と行 29 で定義された SINC コールバック関数は、SINC_DATA0 割込みと SINC_STAT 割込みを処理します。

SetupTRU コード・ブロック （行 37～46）は、すべてのトリガ・ルーティングを含みます。SetupPWM コード・ブロック （行 47 ～ 74）は、PWM タイマ周波数、同期パルス、トリップ機能のセットアップを含みます。外部ハードウェア・トリップは TRIP0 に接続され、内部 SINC_Px_OVLD トリガは TRIP1 に接続されます。TRIP1 割込みは、SINC 過負荷で生成される割込みの 1 つです。過負荷も SINC_STAT 割込みを発生します。

SetupSINC コード・ブロック （行 75～106）は、SINC フィルタ・パラメータを設定します。行 78 ～ 80 はデバイス・ドライバをオープンし、コールバック関数をセットアップします。行 81 ～ 85 は、次数やデシメーション・レートなど種々のグループ・パラメータを設定します。行 87 ～ 89 は、フルスケール範囲に対する過負荷制限値の初期設定値を制御してフィルタ起動時の偽トリップを防止します。行 86 はプライマリ SINC データのサーキュラ・バッファを設定し、行 94 と行 95 はデータ・チャンネルをバッファに割り当てます。行 91 と行 92 は、変調器クロックを設定します。行 91 で、ドライバがシステム・クロックと変調器周波数から分周比を計算します。行 92 で、ドライバ呼出しがクロックをイネーブルし、同期モードを設定します。行 97 は SINC_STAT 割込みマスクをイネーブルします。行 98 と行 99 は、Group0 に割り当てられたフィルタ・チャンネル 0 とフィルタ・チャンネル 1 をイネーブルします。行 100 と行 101 は、行 103 で SINC 割込みマスクを指定し、行 104 と行 105 でセカンダリ・フィルタ過負荷制限値を設定する前に短い遅延を挿入します。

コードの最終ブロック（行 106 ～ 129）は、SINC データと過負荷割込みに対するコールバック関数を含みます。SincDataCallback 関数はバッファからのデータをモーター制御変数へコピーして、制御関数を呼び出します。SincStatusCallback は、フォルト・ハンドラ・ルーチンを呼び出します。

```
/*************************************
SINC FILTER SETUP CODE SNIPPETS
*************************************/
/****   Include file #define code    ***/
/* SINC definitions */
#define SINC_DEV                0
#define SINC_NUM_SAMPLES        4
/* this determines how often a data
interrupt is generated */
#define SINC_NUM_PAIRS          2
#define SINC_DATA_SIZE
(SINC_NUM_PAIRS * SINC_NUM_SAMPLES)
#define CIRC_BUF_SIZE
(SINC_NUM_SAMPLES*20)
/* size for the device circular buffer */
#define SINC_MODCLK             (5000000)
/* modulator clock freq */
#define PDEC                125
/* primary decimation */
#define PSCALE              24
/* Primary scale */
#define SDEC                5
/* secondary decimation */
#define LWIN                4
/* Glitch window    */
#define LCNT                4
/* Glitch count */
#define LMAX           124
/* Overload max limit */
#define LMIN           1
/* Overload min limit */
/* TRU definitions*/
#define TRU_DEV_NUM             0
#define ADI_TRU_REQ_MEMORY      4u


/*  SINC related P R O T O T Y P E S  and
memory allocation                */
/* Function prototypes */
void SetupPWM(void);
void SetupTRU(void);
void SetupSINC(void);
/* Prototype for callback functions */
static void SincDataCallback(void *pHandle,
uint32_t event, void *pArg);
static void SincStatusCallback(void* pHandle,
uint32_t event, void* pArg);
/* SINC handler and data buffers */
static uint8_t
SincMemory[ADI_SINC_MEMORY_SIZE];
static ADI_SINC_HANDLE hSINC;
static int16_t sincData1[SINC_DATA_SIZE];
static int16_t sincData2[SINC_DATA_SIZE];
static int16_t
sincCircBuffer[CIRC_BUF_SIZE];
/*********************************/


void SetupTRU(void){
/*****       Function: SetupTRU  (code
snippet for SINC related setup)  *******/
ADI_TRU_RESULT result;
result = adi_tru_Open (TRU_DEV_NUM,
&TruDevMemory[0], ADI_TRU_REQ_MEMORY,
&hTru);
// Setup TRU for SINC. Slave is SINC0 SYNC0,
master is PWM0 sync pulse
result = adi_tru_TriggerRoute (hTru,
TRGS_SINC0_SYNC0, TRGM_PWM0_SYNC);  // TRU
device, slave, master
result = adi_tru_TriggerRoute (hTru,
TRGS_PWM0_TRIP_TRIG1, TRGM_SINC0_P0_OVLD);
/* connect SINC_Px_OVLD trigger to
PWM0_TRIP_TRIG1. slave, master */
result = adi_tru_TriggerRoute (hTru,
TRGS_PWM0_TRIP_TRIG1, TRGM_SINC0_P1_OVLD);
/* Both overload detection on TRIP1. TRIP0
used by HW */
result = adi_tru_Enable (hTru, true); //
Enable TRU
}


void SetupPWM(void){
/****   Function: SetupPWM  (code snippet
for SINC related setup)  *******/
static ADI_PWM_RESULT result;
uint32_t temp = 0;


result = adi_pwm_Open(PWM_DEV, &PwmMemory,
ADI_PWM_MEMORY_SIZE, &hPWM, PwmCallback,
NULL); // Open driver


temp = (uint32_t)(fsysclk / (2u * FPWM));
// Calculate switching period as number of
sys clocks (up-down counter)
result = adi_pwm_SetReferencePeriod(hPWM,
temp);


temp = (uint32_t)(fsysclk *
SYNC_PULSE_WIDTH);       // Calculate sync
pulse width as number of sys clocks (up-down
counter)
result = adi_pwm_SetSyncWidth(hPWM, temp);
result = adi_pwm_ExtSyncEnable(hPWM, false,
false);          // Internal sync used
```

```
result = adi_pwm_SetIntSyncTimerMode(hPWM,
    ADI_PWM_TIMER0);  // Use timer 0 to generate
    sync.

result = adi_pwm_SetTripEnable(hPWM,
    ADI_PWM_CHANNEL_A, ADI_PWM_TRIP0_SRC, true);
    // Enable Trip0 and trip on all channels
result = adi_pwm_SetTripEnable(hPWM,
    ADI_PWM_CHANNEL_B, ADI_PWM_TRIP0_SRC, true);
result = adi_pwm_SetTripEnable(hPWM,
    ADI_PWM_CHANNEL_C, ADI_PWM_TRIP0_SRC, true);
result = adi_pwm_SetTripEnable(hPWM,
    ADI_PWM_CHANNEL_A, ADI_PWM_TRIP1_SRC, true);
    // Enable Trip1 and trip on all channels
result = adi_pwm_SetTripEnable(hPWM,
    ADI_PWM_CHANNEL_B, ADI_PWM_TRIP1_SRC, true);
result = adi_pwm_SetTripEnable(hPWM,
    ADI_PWM_CHANNEL_C, ADI_PWM_TRIP1_SRC, true);

result = adi_pwm_SetTripMode(hPWM,
    ADI_PWM_CHANNEL_A, ADI_PWM_TRIP0_SRC,
    false);     // Stop PWM and report fault at
    trip. Do not restart
result = adi_pwm_SetTripMode(hPWM,
    ADI_PWM_CHANNEL_B, ADI_PWM_TRIP0_SRC,
    false);
result = adi_pwm_SetTripMode(hPWM,
    ADI_PWM_CHANNEL_C, ADI_PWM_TRIP0_SRC,
    false);
result = adi_pwm_SetTripMode(hPWM,
    ADI_PWM_CHANNEL_A, ADI_PWM_TRIP1_SRC,
    false);
result = adi_pwm_SetTripMode(hPWM,
    ADI_PWM_CHANNEL_B, ADI_PWM_TRIP1_SRC,
    false);
result = adi_pwm_SetTripMode(hPWM,
    ADI_PWM_CHANNEL_C, ADI_PWM_TRIP1_SRC,
    false);

result = adi_pwm_InterruptEnable(hPWM,
    ADI_PWM_INTERRUPT_TIMER0, true);
    // Enable sync irq
result = adi_pwm_InterruptEnable(hPWM,
    ADI_PWM_INTERRUPT_TRIP0, true);
    // Enable trip0 irq
result = adi_pwm_InterruptEnable(hPWM,
    ADI_PWM_INTERRUPT_TRIP1, true);
    // Enable trip1 irq

/*** other PWM setup code **/
}

void SetupSINC(void){
/***           Function: SetupSINC
******/

static ADI_SINC_RESULT result;

result = adi_sinc_Open(SINC_DEV, SincMemory,
    ADI_SINC_MEMORY_SIZE, &hSINC);

result = adi_sinc_RegisterDataCallback
    (hSINC, SincDataCallback, 0);
result = adi_sinc_RegisterStatusCallback
    (hSINC, SincStatusCallback, 0);

/* Specify Group Parameters */
result = adi_sinc_SetRateControl (hSINC,
    ADI_SINC_GROUP_0, ADI_SINC_FILTER_PRIMARY,
    PDEC, 0);
result = adi_sinc_SetRateControl (hSINC,
    ADI_SINC_GROUP_0, ADI_SINC_FILTER_SECONDARY,
    SDEC, 0);
result = adi_sinc_SetLevelControl (hSINC,
    ADI_SINC_GROUP_0, LWIN, LCNT,
    SINC_NUM_SAMPLES, PSCALE);
result = adi_sinc_SetFilterOrder (hSINC,
    ADI_SINC_GROUP_0,
    ADI_SINC_FILTER_THIRD_ORDER,
    ADI_SINC_FILTER_THIRD_ORDER);
result = adi_sinc_SetCircBuffer(hSINC,
    ADI_SINC_GROUP_0, sincCircBuffer,
    CIRC_BUF_SIZE);

/* Reset overload amplitude detection limits
    to 0 – FullScale  */
result = adi_sinc_SetAmplitudeLimit (hSINC,
    ADI_SINC_PAIR_0, 0x0000, 0xFFFF);
result = adi_sinc_SetAmplitudeLimit (hSINC,
    ADI_SINC_PAIR_1, 0x0000, 0xFFFF);

/* Specify Modulator Clock frequency, phase
    & startup synchronization  */
result = adi_sinc_ConfigModClock (hSINC,
    ADI_SINC_GROUP_0, fsysclk, SINC_MODCLK, 0,
    false);
result = adi_sinc_EnableModClock (hSINC,
    ADI_SINC_GROUP_0,
    ADI_SINC_MOD_CLK_PWM_SYNC);

/* submit buffers to receive SINC data */
result = adi_sinc_SubmitBuffer(hSINC,
    ADI_SINC_GROUP_0, sincData1, 16);
```

```
result = adi_sinc_SubmitBuffer(hSINC,
    ADI_SINC_GROUP_0, sincData2, 16);


/* route the TRU interrupt */
result = adi_sinc_EnableDataInterrupt (hSINC,
    ADI_SINC_GROUP_0, ADI_SINC_DATA_INT_0,
    true);


result = adi_sinc_EnablePair(hSINC,
    ADI_SINC_PAIR_0, ADI_SINC_GROUP_0, true);
result = adi_sinc_EnablePair(hSINC,
    ADI_SINC_PAIR_1, ADI_SINC_GROUP_0, true);


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


   /* Enable & assign used SINC filter pair,
    and specify interrupt masks */
   result = adi_sinc_SetControlIntMask
    (hSINC,
    ADI_SINC_INT_EPCNT0|ADI_SINC_INT_EFOVF0 |
    ADI_SINC_INT_EPCNT1|ADI_SINC_INT_EFOVF1 |
    ADI_SINC_INT_ELIM0);


   result = adi_sinc_SetAmplitudeLimit
    (hSINC, ADI_SINC_PAIR_0, LMIN, LMAX);
   result = adi_sinc_SetAmplitudeLimit
    (hSINC, ADI_SINC_PAIR_1, LMIN, LMAX);
   }
   /***         Function: SincDataCallback
    ****/
   static void SincDataCallback(void*
    pHandle, uint32_t event, void* pArg){
   static uint16_t *bufferPtr;
   bufferPtr = (uint16_t*)pArg;  /* pointer
    to sincData1 or sincData2 */
   switch((ADI_SINC_EVENT)event){
   case ADI_SINC_EVENT_DATA0:
   Mctrl_U.ibc_sinc[1] = *bufferPtr;
   Mctrl_U.ibc_sinc[0] = *(bufferPtr+1);
   MotorControl();       /* Algorithm call
    */
   break;
   case ADI_SINC_EVENT_STATUS:
   break;
   default:
   break;
   }
   }
   /*  Function: SincStatusCallback
    */
   static void SincStatusCallback(void*
    pHandle, uint32_t event, void* pArg){
   ADI_SINC_EVENT eEvent =
    (ADI_SINC_EVENT)event;
   uint32_t status = (uint32_t)pArg;
   if (status & ADI_SINC_STATUS_GLIM0){
   SINC_TRIP_Fault_handler();
   }
   }
```