

Design Guide: TIDA-010265

C2000™ および MSPM0 を使用する 750W モーター インバータのリファレンス デザイン



概要

このリファレンス デザインは、洗濯機などのアプリケーション向けの 750W モーター駆動であり、FAST™ ソフトウェア エンコーダまたは eSMO を使用し 3 相 PMSM のセンサレス FOC 制御を実装する方法を示しています。モジュール型設計を採用したこのリファレンス デザインでは、同じマザーボード上で C2000™ マイクロコントローラと MSPM0 シリーズ マイクロコントローラのドーターボードの両方をサポートしています。このリファレンス デザインで利用できるハードウェアとソフトウェアはテスト済みであり、すぐに使用できるので、開発期間短縮に貢献します。ハードウェア設計の詳細とテスト結果は、このデザイン ガイドに記載しています。

リソース


TIDA-010265	デザイン フォルダ
TMS320F2800137	プロダクト フォルダ
MSPM0G1507	プロダクト フォルダ
UCC28881 , TPS54202 , TLV9062	プロダクト フォルダ
C2000WARE-MOTORCONTROL-SDK	ツール フォルダ

特長

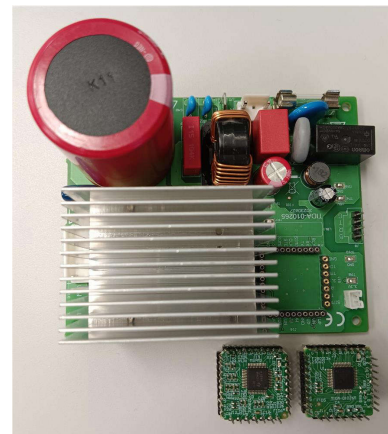
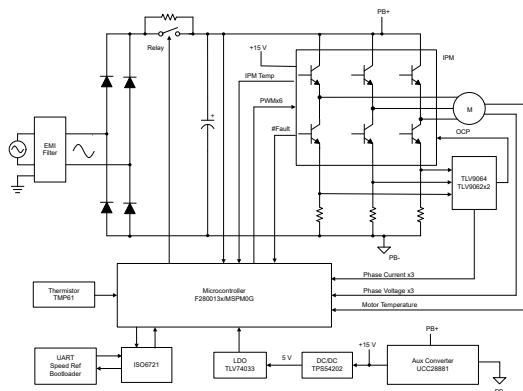
- 広い動作電圧入力範囲: AC 165V~265V, 50|60Hz
- 最大 750W のインバータ段、15kHz のスイッチング周波数、トルク補償、自動弱め界磁制御
- 同じ電源マザーボード上に C2000 または MSPM0 コントローラのドーターボードを搭載したモジュール型設計
- センサレス フィールド オリエンテッド コントロール (FOC) でのモーター制御で、FAST オブザーバ および eSMO オブザーバーの両方をサポート
- モーターの制御、識別、監視に適した使いやすいグラフィカル ユーザー インターフェイス (GUI)

アプリケーション

- 洗濯機 / 乾燥機
- エアコン室内機
- 冷蔵庫と冷凍庫
- 家電製品:コンプレッサ



テキサス・インスツルメンツの E2E™ サポート エキスパートにお問い合わせください。



1 システムの説明

現在、主要な家電製品アプリケーション向けのモーター制御は、低コスト、小型化、大電力、高エネルギー効率に対する需要の増大に対応する必要があります。永久磁石同期型モーター (PMSM) は、主要な家電アプリケーションでますます普及が進んでいます。

このリファレンス デザインでは、制御用に TMS320F2800137 または MSPM0G1507 のいずれかのドーターカードが搭載された単一の 750W インバータ マザーボードを提供しており、同じハードウェア プラットフォーム上で C2000 と MSPM0 シリーズの両方のマイコンを評価する際に好都合です。

ソフトウェアは FAST と eSMO の両方のオブザーバをサポートしているため、2 つのデバイスの性能を比較することができます。また、使いやすい GUI によって、モーターの識別や制御パラメータの調整も容易に行うことができるため、開発時間を短縮することができます。

1.1 用語

SLYZ022	テキサス・インスツルメンツ用語集:この用語集には、用語や略語の一覧および定義が記載されています。
PMSM	永久磁石同期型モーター
ブラシレス DC	ブラシレス直流電流
BEMF	逆起電力
PWM	パルス幅変調
FET、MOSFET	金属酸化膜半導体電界効果トランジスタ
IGBT	絶縁型ゲートバイポーラトランジスタ
RMS	実効値
MTPA	最大トルク / 電流
FWC	弱め界磁制御
FOC	フィールド オリエンテッド コントロール
HVAC (エアコン)	暖房、換気、空調
ESMO	拡張スライディング モード オブザーバ
PLL	位相ロック ループ
FAST	フラックス、角度、速度、およびトルクオブザーバ

1.2 主なシステム仕様

TIDA-010265 の仕様を [表 1-1](#) に示します。

表 1-1. 主なシステム仕様

パラメータ	テスト条件	最小値	公称値	最大値	単位
	システム入力特性				
入力電圧 (V_{IN})	-	165	230	265	VAC
入力周波数 (f_{LINE})	-	47	50	63	Hz
無負荷時スタンバイ電力 (P_{NL})	$V_{INAC} = 230V, I_{out} = 0A$	-	3.0	-	W
入力電流 (I_{IN})	$V_{INAC} = 230V, I_{out} = I_{MAX}$	-	8	-	A
	モーター インバータ特性				
PWM スイッチング周波数 (f_{SW})	-	-	15	20	kHz
定格出力電力 (P_{OUT})	$V_{INAC} =$ 公称値	-	500	750	W
出力電流 (I_{RMS})	$V_{INAC} =$ 公称値	-	3	-	A
インバータ効率 (η)	$V_{INAC} =$ 公称値、 $P_{OUT} =$ 公称値	-	98	-	%
モーターの電気周波数 (f)	$V_{INAC} =$ 最小値～最大値	20	200	400	Hz
フォルト保護	過電流、ストールと回復、低電圧、過電圧				
ドライブ制御方式と特長	電流センシング用に 3 つまたは 1 つのシャント抵抗を使用するセンサレス FOC				
	システム特性				
内蔵補助電源	$V_{INAC} =$ 最小値～最大値	15V \pm 10%、200mA、3.3V \pm 10%、300 mA			
動作時周囲温度	オープンフレーム	-10	25	55	$^{\circ}C$
ボードサイズ	長さ \times 幅 \times 高さ	105 mm \times 85 mm \times 60 mm			mm ³

警告

テキサス・インスツルメンツは、このリファレンス デザインをラボ環境のみで使用するものとし、このデバイスを一般消費者向けの完成品とはみなしておりません。

テキサス・インスツルメンツは、このリファレンス デザインを高電圧電気機械部品、システム、およびサブシステムの取り扱いに関連するリスクを熟知した有資格のエンジニアおよび技術者のみが使用するものとしています。

高電圧！ 基板上は高電圧状態になっており、接触するおそれがあります。基板は、不適切に取り扱ったり適用したりした場合に感電、火災、けがの原因となる電圧および電流で動作します。負傷や物品の破損を避けるために、必要な注意と適切な対策をもって機器を使用してください。

表面は高温！ 触れるとやけどの原因になることがあります。**触れないでください！** 基板の電源を入れると、一部の部品は $55^{\circ}C$ を超える高温に達することがあります。動作中は常に、また動作直後も高温の状態が続く可能性があるため、基板に触れてはいけません。

注意

電源を入れたままその場を離れないでください。

2 システム概要

2.1 ブロック図

図 2-1 に、このリファレンス デザインのブロック図を示します。

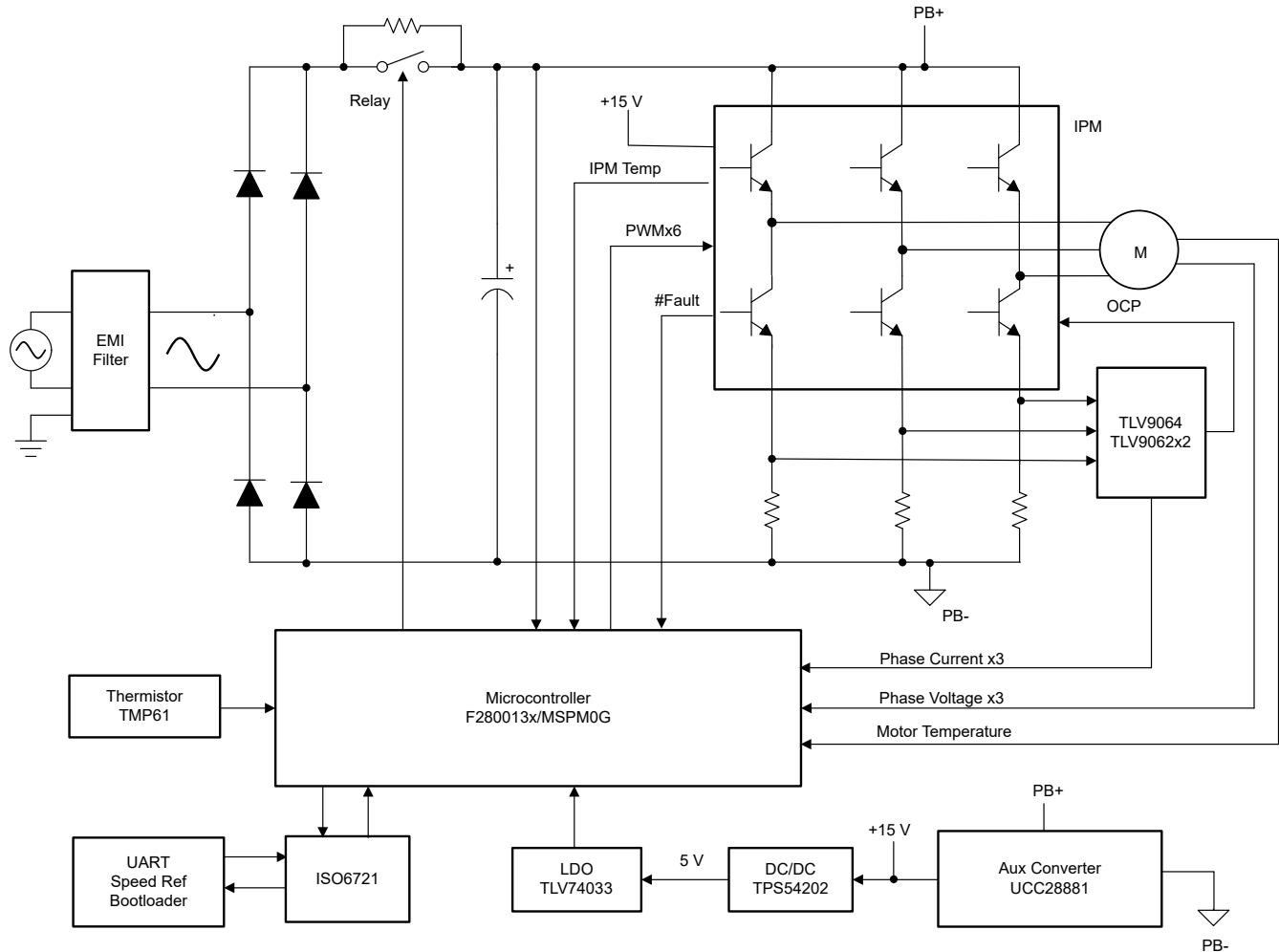


図 2-1. TIDA-010265 750W モーター インバータのブロック図

システム全体は、7 つのブロックで構成されています。

- EMI フィルタ
- ブリッジ整流器
- 3 相インバータ
- 補助電源
- TMS320F2800137 ドーターカード
- MSPM0G1507 ドーターカード

2.2 設計上の考慮事項

このデザインは C2000 コントローラまたは MSPM0 コントローラのいずれかをサポートし、単一モーターを制御するものです。高精度のモーター駆動を実現するには、ノイズ耐性の高い電流と電圧のセンシング設計が求められます。このデザインで使用されるセンシング回路と駆動回路について以下に詳述します。ハードウェア設計ファイルは、C2000Ware Motor Control SDK Install ディレクトリの <install_location> \solutions\tida_010265_wminv\hardware から入手できます。

2.3 主な使用製品

このリファレンス デザインでは、以下の主な製品を使用しています。次のセクションでは、このリファレンス デザインで使用するデバイスを選択するための主な機能について説明します。主なデバイスの詳細については、それぞれの製品データシートを参照してください。

2.3.1 TMS320F2800137

TMS320F280013x は、パワー エレクトロニクス用アプリケーションの効率を高めるために設計されたスケーラブルな超低レイテンシ デバイスである **C2000™** リアルタイム マイクロコントローラ ファミリの製品です。リアルタイム制御サブシステムは、テキサス・インスツルメンツの **32 ビット C28x** デジタル信号プロセッサ (DSP) コアをベースにしており、オンチップ フラッシュまたは **SRAM** から実行される浮動小数点または固定小数点コードに対して **120MHz** の信号処理性能を発揮します。**C28x CPU** は三角関数算術演算ユニット (TMU) と巡回冗長検査 (VCRC) 拡張命令セットによってさらに強化され、リアルタイム制御システムでよく使われる重要なアルゴリズムを高速処理します。**F280013x** リアルタイム マイクロコントローラ (MCU) に内蔵された高性能アナログ ブロックは、優れたリアルタイム シグナル チェーン性能を実現するために、処理ユニットおよび PWM ユニットと密結合されています。**14** 個の PWM チャネルはすべて周波数に依存しない分解能モードをサポートしており、**3** 相インバータから高度なマルチレベル電源トポロジまで、さまざまな電力段を制御できます。インターフェイスは、各種の業界標準通信ポート (SPI、**3** つの SCI|URAT、I2C、CAN など) によりサポートされており、優れた信号配置のための複数のピン多重化方法を備えています。

2.3.2 MSPM0G1507

MSPM0G150x マイクロコントローラ (MCU) は、最大 **80MHz** の周波数で動作する拡張 **Arm® Cortex®-M0+** **32** ビット コア プラットフォームをベースにしたミックスド シグナル処理 (MSP) の高集積超低消費電力 **32** ビット MCU ファミリの製品です。コスト最適化されたこれらの MCU は高性能アナログ ペリフェラルを統合しており、**-40°C~125°C** の拡張温度範囲をサポートしており、**1.62V~3.6V** の電源電圧で動作します。**MSPM0G150x** デバイスは、エラー訂正コード (ECC) を備えた最大 **128KB** の組み込みフラッシュプログラム メモリと、ハードウェア パリティ オプションを備えた最大 **32KB** の **SRAM** を内蔵しています。また、メモリ保護ユニット、**7** チャネル DMA、演算アクセラレータ、各種の高性能アナログ ペリフェラル (**2** つの設定可能内部共有リファレンス電圧付き **12** ビット **4MSPS** ADC、**1** つの **12** ビット **1MSPS** DAC、**3** つのリファレンス電圧 D/A コンバータ (DAC) 内蔵高速コンパレータ、**2** つのゲインをプログラム可能なゼロドリフトゼロクロスオーバー オペアンプ、**1** つの汎用アンプなど) も内蔵しています。これらのデバイスは、**3** つの **16** ビット高度制御タイマ、**3** つの **16** ビット汎用タイマ、**1** つの **24** ビット高分解能タイマ、**2** つのウィンドウ付きウォッチドッグ タイマ、**1** つのアラームおよびカレンダー モード付き RTC などのインテリジェントなデジタル ペリフェラルも備えています。これらのデバイスは、データの整合性と暗号化のペリフェラル、および豊富な通信インターフェイス (**4** つの **UART**、**2** つの **I2C**、**2** つのシリアルペリフェラル インターフェイス (SPI)) を提供します。

2.3.3 TMP6131

TMP61x リニア サーミスタは、全温度範囲にわたる線形性と安定した感度を備えているため、簡単かつ正確な方法で温度を変換できます。本デバイスは消費電力が低く、熱容量が小さいため、自己発熱の影響を最小限に抑えることができます。

本質的に高温時にフェイルセーフ挙動を示し環境変化に耐えるこれらのデバイスは、長寿命高性能向けに設計されています。また **TMP6** シリーズは、小型であるため熱源に近付けて配置でき迅速な応答が得られます。

2.3.4 UCC28881

UCC28881 は、コントローラと、**14Ω 700V** のパワー MOSFET を **1** つのモノリシック デバイスに統合したものです。また、このデバイスには高電圧電流源も内蔵されており、整流電源電圧からの直接の起動と動作が可能です。**UCC28881** は、より電流が大きい **UCC28880** と同じファミリー デバイスです。

デバイスの静止電流が小さいため、優れた効率が得られます。**UCC28881** を使用すると、降圧、昇降圧、フライバックなどの最も一般的なコンバータトポロジを、最小限の外付け部品数で構築できます。

2.3.5 TPS54202

TPS54202 は入力電圧範囲が **4.5V~28V** で、**2A** の同期整流降圧コンバータです。このデバイスには **2** つの内蔵スイッチング FET、内部的なループ補償、および **5ms** の内部ソフトスタートが搭載されているため、部品数を減らすことができます。

高度な **Eco-mode** の実装により、軽負荷時の効率が最大化され、電力損失が低減されています。

両方のハイサイド MOSFET でサイクル単位の電流制限を行い、過負荷の状況でコンバータを保護します。また、ローサイド MOSFET の電流制限を自由に設定でき、電流暴走を防止することで、さらに保護が強化されています。

2.3.6 TLV9062

TLV9062 は、レール ツー レールの入力および出力スイング機能を備えたデュアル低電圧 (**1.8V~5.5V**) オペアンプです。このデバイスは、低電圧での動作、小さな占有面積、大きな容量性負荷の駆動が必要なアプリケーション向けの、コスト効率の優れた設計です。**TLV906x** の容量性負荷駆動能力は **100pF** ですが、抵抗性オープン ループ出力インピーダンスにより、高い容量性の負荷でも簡単に安定できます。**TLV906xS** デバイスにはシャットダウン モードが備わっており、標準消費電流 **1μA** 未満で、アンプをスタンバイ モードに切り替えることができます。**TLV906xS** ファミリーはユニティ ゲイン安定で、RFI および EMI 除去フィルタが内蔵され、オーバードライブ状況で位相反転が発生しないため、システムの設計を簡素化するため役立ちます。

2.3.7 TLV74033

TLV740P 低ドロップアウト (LDO) リニアレギュレータは、静止電流の低い LDO で、ラインおよび負荷過渡性能が非常に優れており、消費電力の制限が厳しいアプリケーション向けに設計されています。このデバイスの標準精度は **1%** です。

また **TLV740P** は、デバイスの電源投入およびイネーブル時に突入電流の制御も行います。**TLV740P** は定義済みの上限値に入力電流を制限し、入力電源から大電流が流れ込むことを防止します。この機能は、バッテリーで動作するデバイスでは特に重要です。

2.4 システム設計理論

このリファレンス デザインは、洗濯機や類似する家電製品アプリケーション向けの単一モーター制御に主な焦点を当てています。

2.4.1 ハードウェア設計

一般的なモーター制御ボードには、補助電源、インバータ、電流および電圧センシング、保護回路、マイクロコントローラなどの複数のブロックがあります。このセクションでは、これらの設計コンセプトについて説明します。

2.4.1.1 モジュール形式の設計

このリファレンスデザインには、モジュール型設計を実現するために、**2** 枚のドーターボードと **1** 枚のマザーボードがあります。**1** 枚のドーターボードには、**MSPM0G1507** マイクロコントローラと、**2** つの高性能内蔵オペアンプに基づいた **2** 相電流アンプ回路が搭載されています。もう **1** 枚のドーターボードには、**TMS320F2800137** マイクロコントローラと、位相電流アンプとして **2** つの **TLV9062** デバイスが搭載されています。マザーボードには、**AC** フィルタ、整流器、インテリジェントパワー モジュール (IPM) を含むすべての電源デバイスが搭載されています。[図 2-2](#) と [図 2-3](#) にマザーボードとドーターボードを示します。

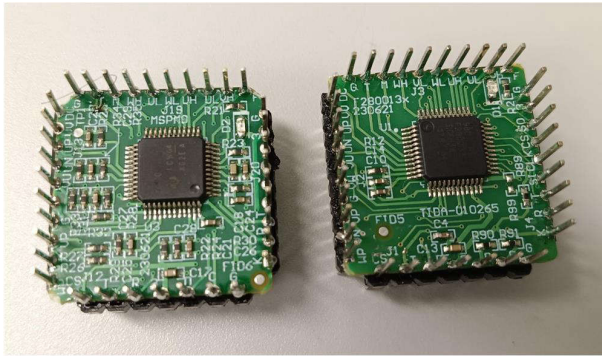


図 2-2. TIDA-010265 ドーターボード

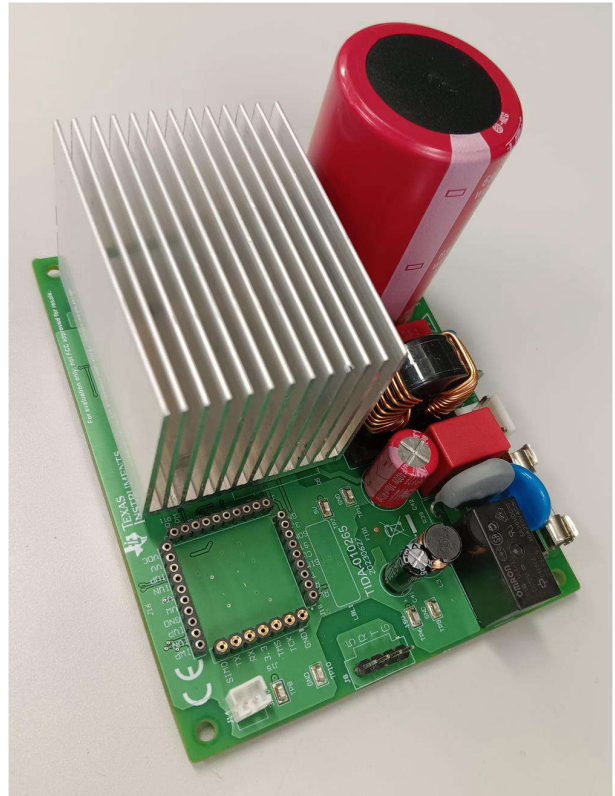


図 2-3. TIDA-010265 マザーボード

2.4.1.2 高電圧降圧補助電源

UCC28881 ベースの非絶縁型高電圧降圧電源は、このリファレンス デザインの補助電源を提供し、DC 15V で最大 200mA を供給します。図 2-4 に、UCC28881 高電圧降圧電源回路を示します。

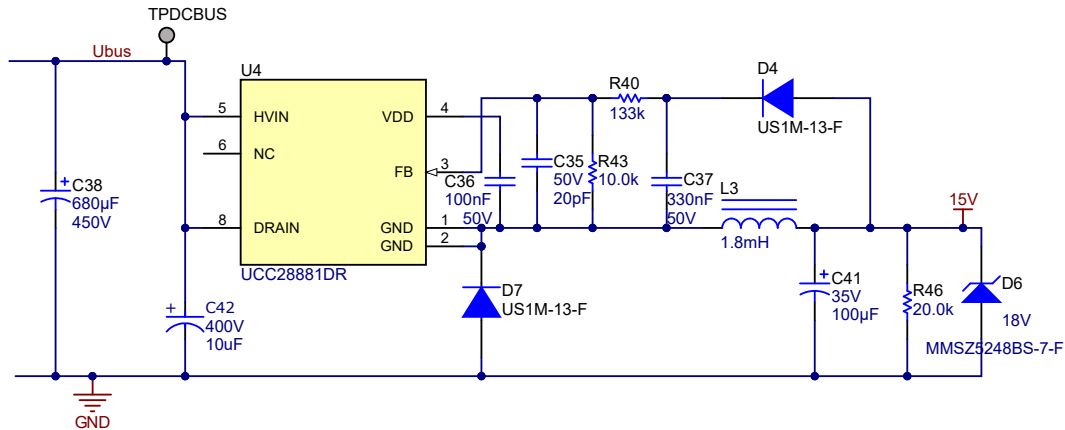


図 2-4. 高電圧降圧電源回路

2.4.1.3 DC リンク電圧検出

DC 電圧センシング回路は、図 2-5 に示すように、整流電圧信号を、低コストの抵抗回路によって生成された低電圧信号に変換するために使用されます。DC バス電圧を使用して、AC 入力電圧を推定することもできます。

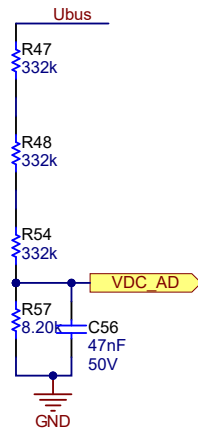


図 2-5. DC バス電圧センシング回路

2.4.1.4 モーター相電圧のセンシング

TIDA-010265 向けの C2000 ソフトウェアは、拡張スライディング モード オブザーバ (eSMO) と、フラックス、角度、速度、およびトルク (FAST™) オブザーバの両方をサポートしています。FAST オブザーバを使用すると、低速時の性能を向上させ、速度の許容誤差を抑えることができますが、FAST では 3 相電流センシングに加えて、3 つのモーター相電圧センシングが必要です。セクション 2.4.2.4.2 では、モーター相電圧センシングの設計について詳述しています。

2.4.1.5 モーター相電流のセンシング

MS320F2800137 ドーターボードは 1~3 相の電流センシングをサポートし、MSPM0G1507 ドーターボードは 1~2 相の電流センシングをサポートするように設計されています。セクション 2.4.2.4.1 では、モーター相電流のセンシング設計について詳述しています。

2.4.1.6 外部過電流保護

このリファレンス デザインは、外部コンパレータと内部コンパレータの両方を使用して過電流保護 (OCP) を実装しています。図 2-6 に外部コンパレータによる OCP を示します。この回路は 3 相の電流をまとめた後、U10 の負入力のリファレンス電圧と比較して IPM への高電圧を生成します。その後、IPM が過電流フォルト信号を生成して、マイクロコントローラに通知します。過電流保護電流の正確な値は、以下の式で計算できます。

$$V_- = \frac{3.3 \text{ V}}{R_{20} + R_{108}} \times R_{108} = \frac{3.3 \text{ V}}{20 \text{ k} + 1 \text{ k}} \times 1 \text{ k} = 0.1571 \text{ V} \quad (1)$$

$$V_+ = \frac{I_{\text{ocp}} \times R_{80}}{R_{87} + (R_{104} + R_{105})/2} \times \left(\frac{R_{104} + R_{105}}{2} \right) = \frac{0.05 \times I_{\text{ocp}}}{300 + 150} \times 150 = \frac{0.05 \times I_{\text{ocp}}}{3} \quad (2)$$

$$I_{\text{ocp}} = \frac{V_+}{R_{87}} \times 3 = \frac{0.1571}{0.05} \times 3 = 9.4286 \text{ A} \quad (3)$$

MS320F2800137 と MSPM01507 のドーターボードは、いずれもこの外部 OCP 回路によってトリガできます。

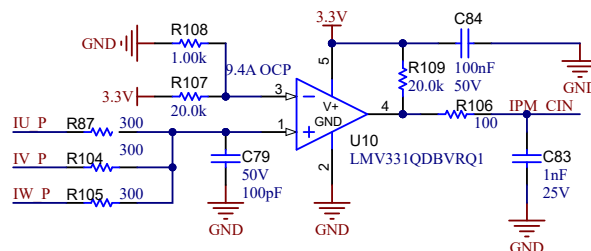


図 2-6. 外部過電流保護回路

2.4.1.7 TMS320F2800F137 の内部過電流保護

TMS320F2800F137 は、3 相電流を監視するように構成できるウィンドウ コンパレータを内蔵しています。内部コンパレータが過電流をトリガして PWM を停止させるためのソフトウェアや割り込みの遅延がないため、外部 IPM やパワー デバイスを迅速に保護することができます。

2.4.2 3 相 PMSM 駆動

永久磁石同期モーター (PMSM) は、巻線固定子、永久磁石回転子アセンブリ、回転子位置をセンシングする内部デバイスや外部デバイスを備えています。センシング デバイスは、磁石アセンブリの回転を維持するために、固定子電圧リファレンスの周波数と振幅を適切に調整するための位置帰還を提供するものです。内部の永久磁石回転子と外部巻線を組み合わせることで、回転子の慣性が小さく、放熱が効率的で、モーターを小型化できるという利点があります。

- 同期モーターの構造: 永久磁石は回転軸にしっかりと固定され、一定の回転子フラックスを生み出します。この回転子フラックスの振幅は通常一定の大きさです。通電されると、固定子巻線によって回転磁場が生成されます。回転磁場を制御するには、固定子電流を制御する必要があります。
- 回転子の実際の構造は、機械の出力範囲と定格速度によって異なります。数キロワットまでの同期機には、永久磁石が最も適しています。高電力定格の場合、回転子は通常 DC 電流が循環する巻線で構成されます。回転子の機械的構造は、必要な極数と必要なフラックス勾配に応じて設計されます。
- 固定子フラックスと回転子フラックスの相互作用によってトルクが生成されます。固定子はフレームにしっかりと取り付けられ、回転子は自由に回転できるため、[図 2-7](#) に示すように、回転子が回転することにより実際の機械的出力が得られます。
- 最大トルクを生成して、高い電気機械的変換効率を達成するには、回転子磁場と固定子磁場の間の角度を慎重に制御する必要があります。このためには、センサレス アルゴリズムを使用して速度ループを閉じた後、速度とトルクが同じ条件のもとで最小量の電流を流すように微調整する必要があります。
- 回転する固定子磁場は、回転子の永久磁場と同じ周波数で回転する必要があります。そうでない場合、回転子には正と負のトルクが急速に交互に発生します。その結果、トルク生成が不十分となり、機械部品に過度の機械的な振動やノイズ、機械的ストレスが生じることになります。さらに、回転子の慣性が原因で、回転子がこれらの振動に反応できなくなると、回転子は同期周波数での回転を停止し、静止している回転子から見た平均トルク、つまりゼロに反応することになります。これは機械がプルアウトと呼ばれる現象を起こしていることを意味しており、同期機が自己起動しない理由でもあります。
- 最高の相互トルク生成を実現するには、回転子磁場と固定子磁場の間の角度が 90° でなければなりません。この同期化には、回転子の位置を把握して、適切な固定子の磁場を生成する必要があります。
- 固定子の磁場は、異なる固定子相からの影響を組み合わせることで固定子フラックスを生成することにより、任意の方向と大きさにすることができます。

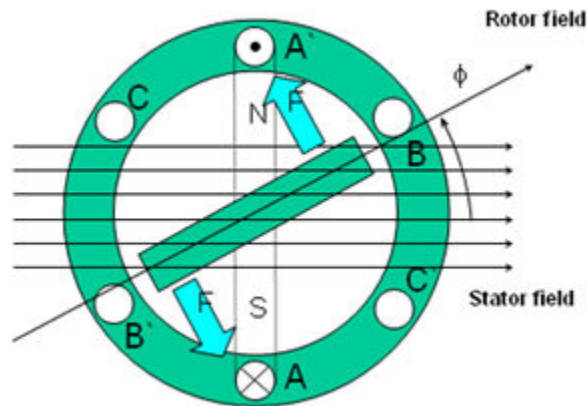


図 2-7. 回転する固定子フラックスと回転子フラックスの相互作用によって生成されるトルク

2.4.2.1 PM 同期モーターのフィールド オリエンテッド コントロール

より優れた動的性能を実現するには、より複雑な制御方式を適用して PM モーターを制御する必要があります。マイクロコントローラがもたらす数学的処理能力により、PM モーターのトルク生成と磁化機能をデカップリングする数学的変換を使

用した高度な制御方式を実装できます。トルクと磁化をデカップリングするこのような制御は、一般的に回転子フラックスオリエンテッドコントロール、または単にフィールドオリエンテッドコントロール (FOC) と呼ばれます。

図 2-8 に示すように、直流 (DC) モーターでは、固定子と回転子の励磁は別々に制御され、生成されたトルクとフラックスは別々の調整が可能です。界励磁の強さ (たとえば、界励磁電流の大きさ) によって、フラックスの値が設定されます。回転子巻線に流れる電流によって、生成されるトルクの大きさが決まります。トルク生成において特に興味深い役割を果たすのが、回転子の整流子です。整流子はブラシと接触しており、機械的構造上、最大トルクを生成するように機械的に整列された巻線が回路に切り替わるように設計されています。この配置によって、機械のトルク生成は常に非常に高いレベルに近い状態になります。ここで重要なポイントは、回転子巻線によって生成されるフラックスが固定子磁場と直交するように巻線が制御されていることです。

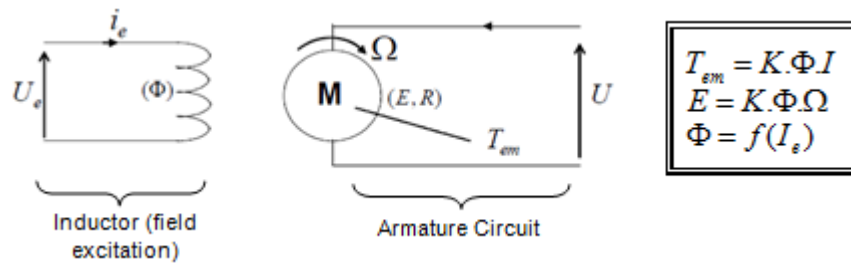


図 2-8. フラックスとトルクが別々に制御される DC モーター モデル

同期機と非同期機の FOC (ベクトル制御とも呼ぶ) の目的は、トルク生成成分とフラックス磁化成分を別々に制御できるようにすることです。FOC 制御を使用すると、固定子電流のトルク生成成分とフラックス磁化成分をデカップリングできます。磁化のデカップリング制御を行うことで、固定子フラックスのトルク生成成分は独立したトルク制御として考えることができるようになりました。トルクとフラックスをデカップリングするには、いくつかの数学的変換を行う必要があります。マイクロコントローラが最も価値を発揮するのがこの部分になります。マイクロコントローラによる処理能力によって、このような数学的変換を非常に高速で行うことができます。これは、モーターを制御するアルゴリズム全体を高速で実行できることを意味し、より高度な動的性能が実現できるのです。現在では、デカップリングに加えて、回転子フラックスの角度や回転子の速度など、数多くの量の計算にモーターの動的モデルが使用されています。つまり、効果が考慮されて、全体的な制御の質が向上しているのです。

電磁法則によれば、同期機で生成されるトルクは、式 4 に示すように、既存の 2 つの磁場のベクトル外積に等しくなります。

$$\tau_{em} = \vec{B}_{stator} \times \vec{B}_{rotor} \quad (4)$$

この式は、固定子と回転子の磁場が直交しており、負荷が 90 度である場合に、トルクが最大になることを示しています。この条件が常に満たされて、フラックスの向きを正しく保つことができれば、トルクリップルが減少し、より優れた動的応答が得られます。ただし、回転子の位置を把握していなければならないという制約があります。インクリメンタルエンコーダのような位置センサを使用すると実現できます。回転子にアクセスできないような低コストのアプリケーションでは、位置センサを排除するために異なる回転子位置オブザーバ方式が適用されます。

簡単に説明すると、回転子フラックスと固定子フラックスを直交させた状態で維持することです。固定子フラックスを回転子フラックスの q 軸に、たとえば回転子フラックスに直交するように合わせることを目標です。そのために、回転子フラックスと直交する固定子電流成分は指令されたトルクを生成するように制御され、直接成分はゼロに設定されます。固定子電流の直接成分は場合によって弱め界磁に使用することができ、回転子フラックスを逆向きにし、逆 EMF を減少させるので、より高速での動作が可能になります。

FOC は、ベクトルで表される固定子電流を制御します。この制御は、時間と速度に依存する 3 相座標系が時不変の 2 座標系 (d 座標と q 座標) に変換される投影に基づいています。このような投影によって、DC 機制御と同じような構造になります。FOC 機は、入力リファレンスとして、トルク成分 (q 座標) とフラックス成分 (d 座標) の 2 つの定数を必要とします。FOC は単純に投影に基づいているため、制御構造が瞬間的な電気量を取り扱います。これによって、あらゆる動作 (定常状態と過渡状態) について正確な制御が実現し、帯域幅に制限がある数学モデルに依存することがありません。これにより、FOC は従来の方式の問題を次のように解決します。

- 一定のリファレンスに到達しやすい (固定子電流のトルク成分とフラックス成分)

- (d, q) リファレンスフレームではトルクの式が 式 5 で定義されることから、直接トルク制御が適用しやすい

$$\tau_{em} \propto \psi_R \times i_{sq} \quad (5)$$

回転子フラックス (ψ_R) の振幅を固定値に保つことで、トルクとトルク成分 (i_{sq}) の間に線形関係が得られます。したがって、固定子電流ベクトルのトルク成分を制御することで、トルクを制御することができます。

2.4.2.1.1 空間ベクトルの定義と投影

AC モーターの 3 相電圧、電流、フラックスは、複素空間ベクトルとして解析できます。電流の場合、空間ベクトルは次のように定義できます。 i_a 、 i_b 、 i_c を固定子相の瞬時電流とすると、複素固定子電流ベクトルは 式 6 で定義されます。

$$\bar{i}_s = i_a + \alpha i_b + \alpha^2 i_c \quad (6)$$

ここで、

- $\alpha = e^{j\frac{2\pi}{3}}$ および $\alpha^2 = e^{j\frac{4\pi}{3}}$ は、空間演算子を表します。

図 2-9 に、固定子電流の複素空間ベクトルを示します。

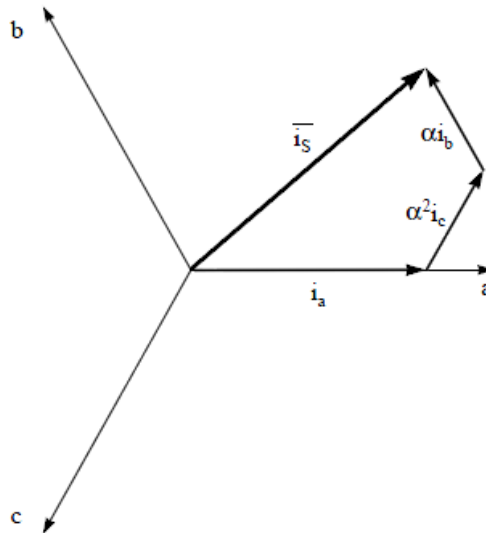


図 2-9. (a, b, c) フレームにおける固定子電流空間ベクトルと成分

ここで、

- a, b, c は 3 相座標系の軸になります。

この電流空間ベクトルは 3 相正弦波座標系を表しており、依然として、時不変の 2 座標系に変換する必要があります。この変換は、次の 2 つのステップに分けることができます。

- (a, b) \Rightarrow (α, β) (クラーク変換) は時変 2 座標系を出力します。
- (α, β) \Rightarrow (d, q) (パーク変換) は時不変 2 座標系を出力します。

2.4.2.1.1.1 (a, b) \Rightarrow (α, β) クラーク変換

空間ベクトルは、直交する 2 つの軸 (α, β) だけを持つ別のリファレンスフレームで表すことができます。a 軸と α 軸が同じ方向であると仮定すると、図 2-10 のようなベクトル図になります。

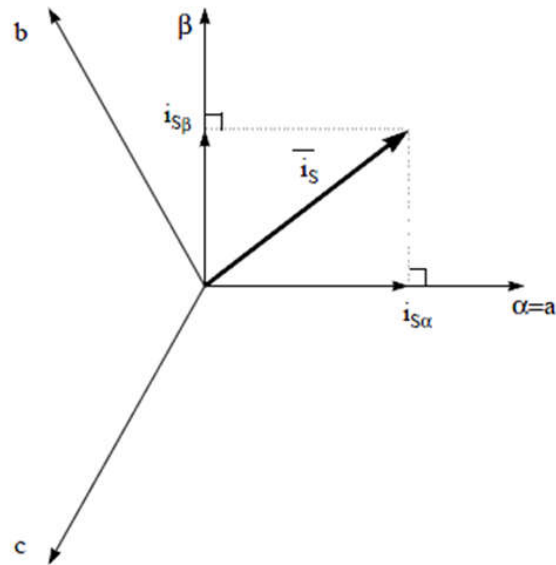


図 2-10. 固定リファレンス フレームの固定子電流空間ベクトル

3 相座標系を (α, β) 2 次元直交系に変更する投影を 式 7 に示します。

$$\begin{aligned} i_{s\alpha} &= i_a \\ i_{s\beta} &= \frac{1}{\sqrt{3}}i_a + \frac{2}{\sqrt{3}}i_b \end{aligned} \quad (7)$$

2 相 (α, β) 電流は、依然として時間と速度に依存します。

2.4.2.1.1.2 $(\alpha, \beta) \Rightarrow (d, q)$ パーク変換

これは FOC における最も重要な変換です。実際には、この投影は (d, q) 回転リファレンス フレームの 2 相直交座標系 (α, β) を変更するものです。d 軸が回転子フラックスと一直線上にあるものとして、図 2-11 に 2 つのリファレンス フレームの電流ベクトルの関係を示しています。

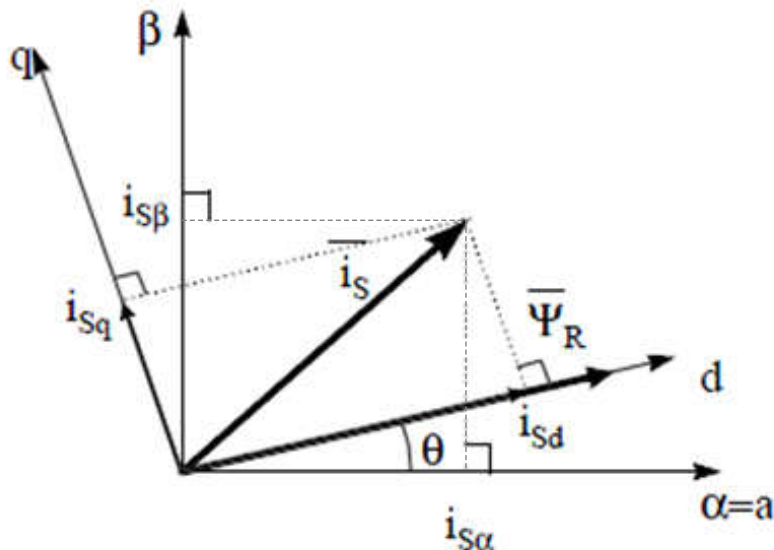


図 2-11. (d, q) 回転リファレンス フレームの固定子電流空間ベクトル

電流ベクトルのフラックス成分とトルク成分は 式 8 で決定されます。

$$\begin{aligned} i_{sd} &= i_{s\alpha}\cos(\theta) + i_{s\beta}\sin(\theta) \\ i_{sq} &= -i_{s\alpha}\sin(\theta) + i_{s\beta}\cos(\theta) \end{aligned} \quad (8)$$

ここで、

- θ は回転子フラックスの位置です。

これらの成分は、電流ベクトル (α, β) の成分と回転子フラックスの位置に依存します。適切な回転子フラックスの位置がわかると、この投影によって d、q 成分は一定になります。これで 2 相電流は DC 量 (時不変) に変わります。この時点で、一定の i_{sd} (フラックス成分) と i_{sq} (トルク成分) の電流成分が別々に制御されるため、トルク制御が容易になります。

2.4.2.1.2 AC モーターの FOC 基本方式

図 2-12 に、FOC によるトルク制御の基本方式をまとめます。

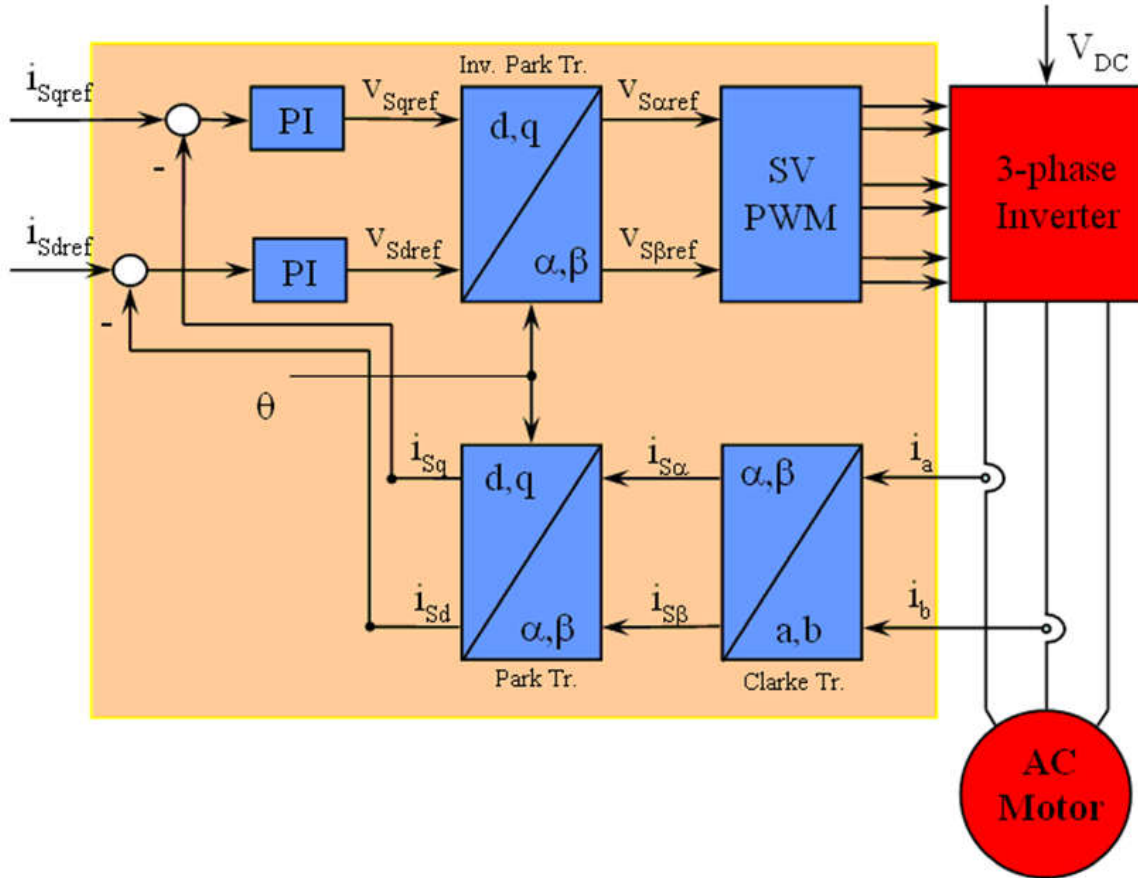


図 2-12. AC モーターの FOC 基本方式

2 つのモーター相電流が測定されて、測定値がクラーク変換モジュールに供給されます。この投影の出力は $i_{s\alpha}$ と $i_{s\beta}$ となります。この電流の 2 つの成分は、d、q 回転リファレンスフレームでの電流をもたらすパーク変換の入力です。 i_{sd} と i_{sq} 成分は、リファレンス i_{sdref} (フラックスリファレンス成分) と i_{sqref} (トルクリファレンス成分) と比較されます。ここで、この制御構造に興味深い利点があることがわかります。つまり、フラックスリファレンスを変更して、回転子フラックスの位置を取得するだけで、同期機と誘導機のどちらを制御するにもこの構造を使用できるということです。永久磁石同期モーターの場合、回転子フラックスは磁石によって固定されているため、フラックスの生成は必要ありません。したがって、PMSM を制御する場合は i_{sdref} をゼロに設定します。AC 誘導モーターは動作するために回転子フラックスを生成する必要があるため、フラックスリファレンスはゼロであってはなりません。これにより、従来の制御構造の大きな欠点の 1 つである、非同期ドライブから同期ドライブへの移行が簡単に解決されます。速度 FOC が使用されている場合、トルク指令 i_{sqref} を速度レギュレータの出力とすることができます。電流レギュレータの出力は V_{sdref} と V_{sqref} であり、逆パーク変換に適用されます。この投影の出力は、(α, β) 固定直交リファレンスフレームにおける固定子ベクトル電圧の成分である V_{saref} と V_{sbrref}

であり、空間ベクトル PWM の入力になります。このブロックの出力はインバータを駆動する信号です。パーク変換と逆パーク変換の両方には回転子フラックスの位置が必要になることに注意してください。この回転子フラックスの位置の取得方法は、AC 機のタイプ (同期機または非同期機) によって異なります。

2.4.2.1.3 回転子フラックスの位置

FOC において、回転子フラックスの位置情報を知ることが中心となります。実際、この変数に誤差があると、回転子フラックスは d 軸と一直線にならず、 i_{sd} と i_{sq} は固定子電流の正しいフラックス成分とトルク成分とはなりません。図 2-13 は、(a, b, c)、(α , β)、(d, q) の各リファレンスフレームを示し、同期速度で d, q リファレンスで回転する、回転子フラックス、固定子電流、固定子電圧の各空間ベクトルの正しい位置を示しています。

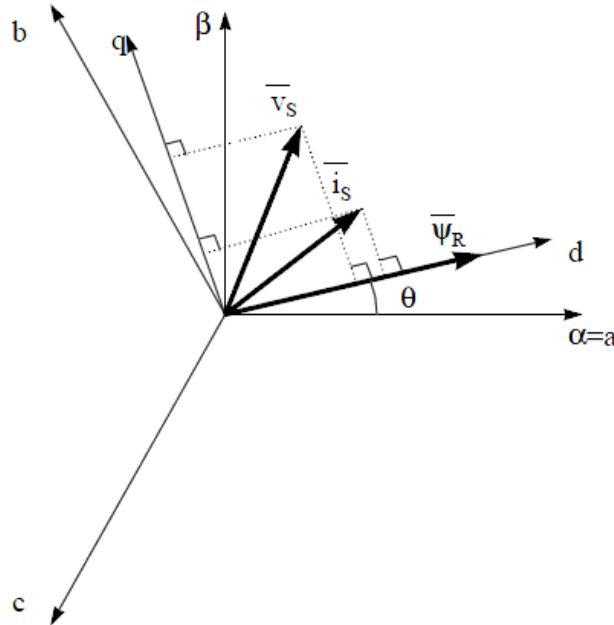


図 2-13. 回転リファレンス フレーム (d, q) の電流、電圧、回転子フラックスの各空間ベクトル

同期モーターと非同期モーターでは、回転子フラックス位置の測定方法が異なります。

- 同期モーターでは、回転子速度は回転子フラックス速度と等しくなります。したがって、 θ (回転子フラックスの位置) は位置センサによって直接測定されるか、回転子速度の積分によって求められます。
- 非同期モーターでは、回転子速度は回転子フラックス速度と等しくないため (スリップ速度があるため)、 θ の算出には特定の方法が必要になります。基本的な方法としては、d, q リファレンス フレームにおけるモーター モデルの 2 つの式を必要とする電流モデルを使用します。

理論的には、PMSM ドライブの FOC により、DC モーターの動作のようにモーター トルクをフラックスとは無関係に制御することができます。言い換えれば、トルクとフラックスは互いに切り離されていることになります。固定リファレンス フレームから同期回転リファレンス フレームへの変数変換を行うには、回転子位置を知る必要があります。この変換 (いわゆるパーク変換) の結果、q-軸の電流がトルクを制御し、d-軸の電流は強制的にゼロになります。したがって、このシステムの重要なモジュールは、拡張スライディング モード オブザーバ (eSMO) または FAST エスティメータを使用した回転子位置の推定になります。

図 2-14 に、このリファレンス デザインにおける、フライング スタートを備えた eSMO を使用した、ファン用 PMSM のセンサレス FOC の全体ブロック図を示します。

図 2-15 に、このリファレンス デザインにおける、弱め界磁制御 (FWC) と最大トルク / 電流 (MTPA) を備えた eSMO を使用した、コンプレッサ用 PMSM のセンサレス FOC の全体ブロック図を示します。

図 2-16 に、このリファレンス デザインにおける、フライング スタートを備えた FAST を使用した、ファン用 PMSM のセンサレス FOC の全体ブロック図を示します。

図 2-17 に、このリファレンスデザインにおける、弱め界磁制御 (FWC) と最大トルク / 電流 (MTPA) を備えた FAST を使用した、コンプレッサ用 PMSM のセンサレス FOC の全体ブロック図を示します。

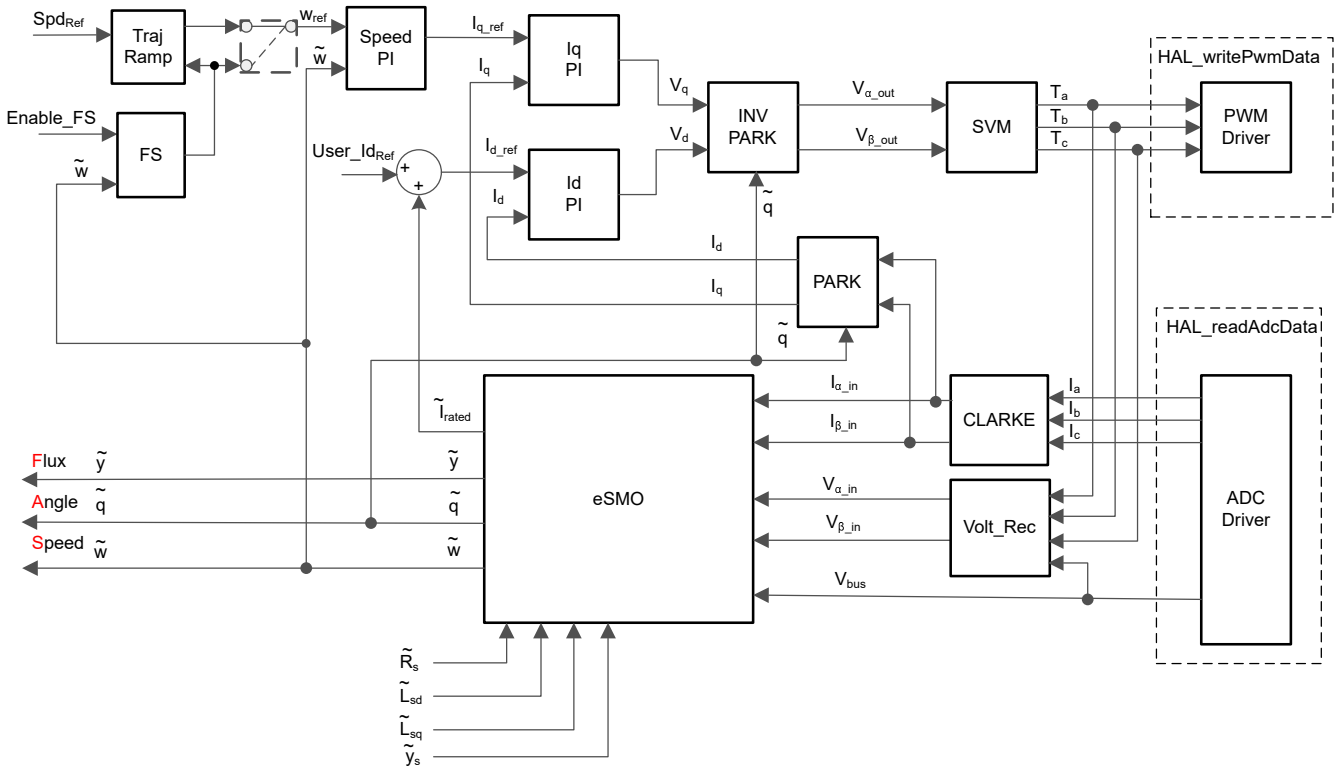


図 2-14. フライング スタート (FS) を備えた eSMO を使用した PMSM のセンサレス FOC

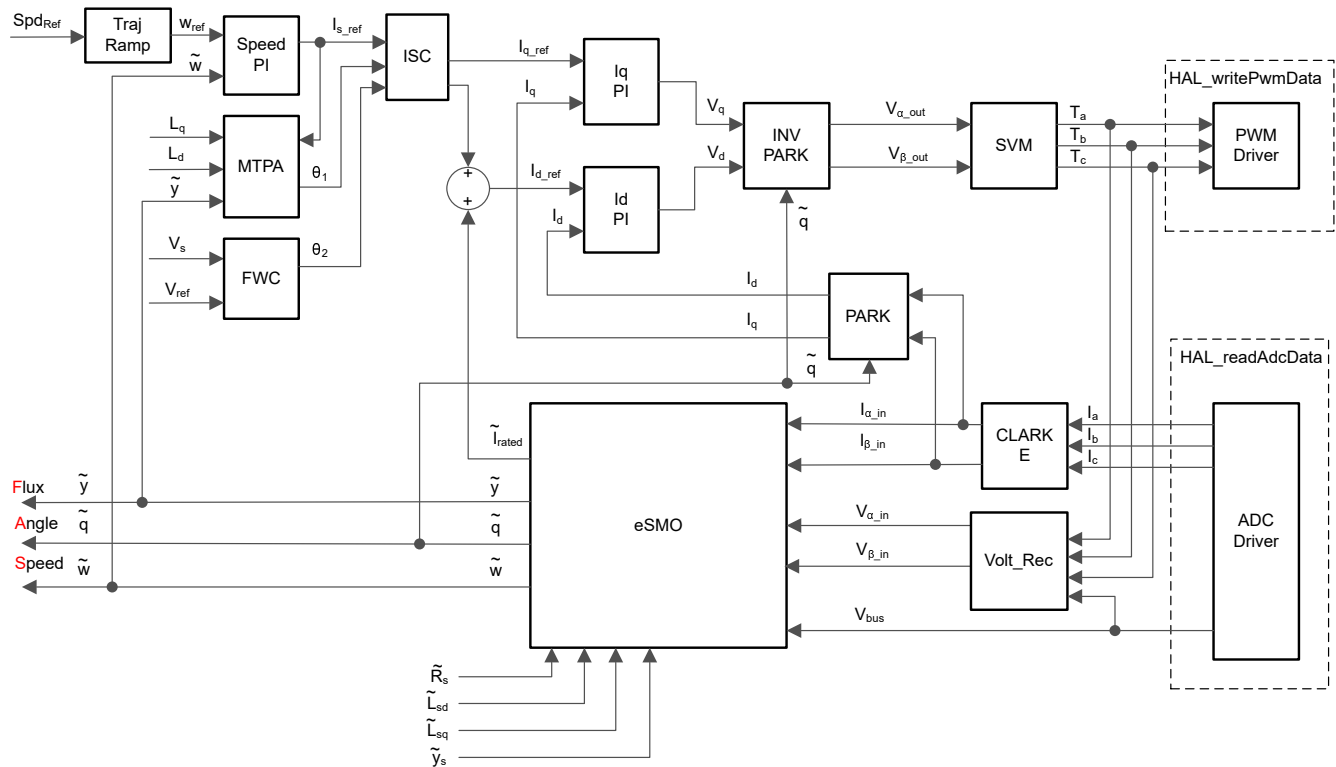


図 2-15. FWC と MTPAeSMO を備えた eSMO を使用した PMSM のセンサレス FOC

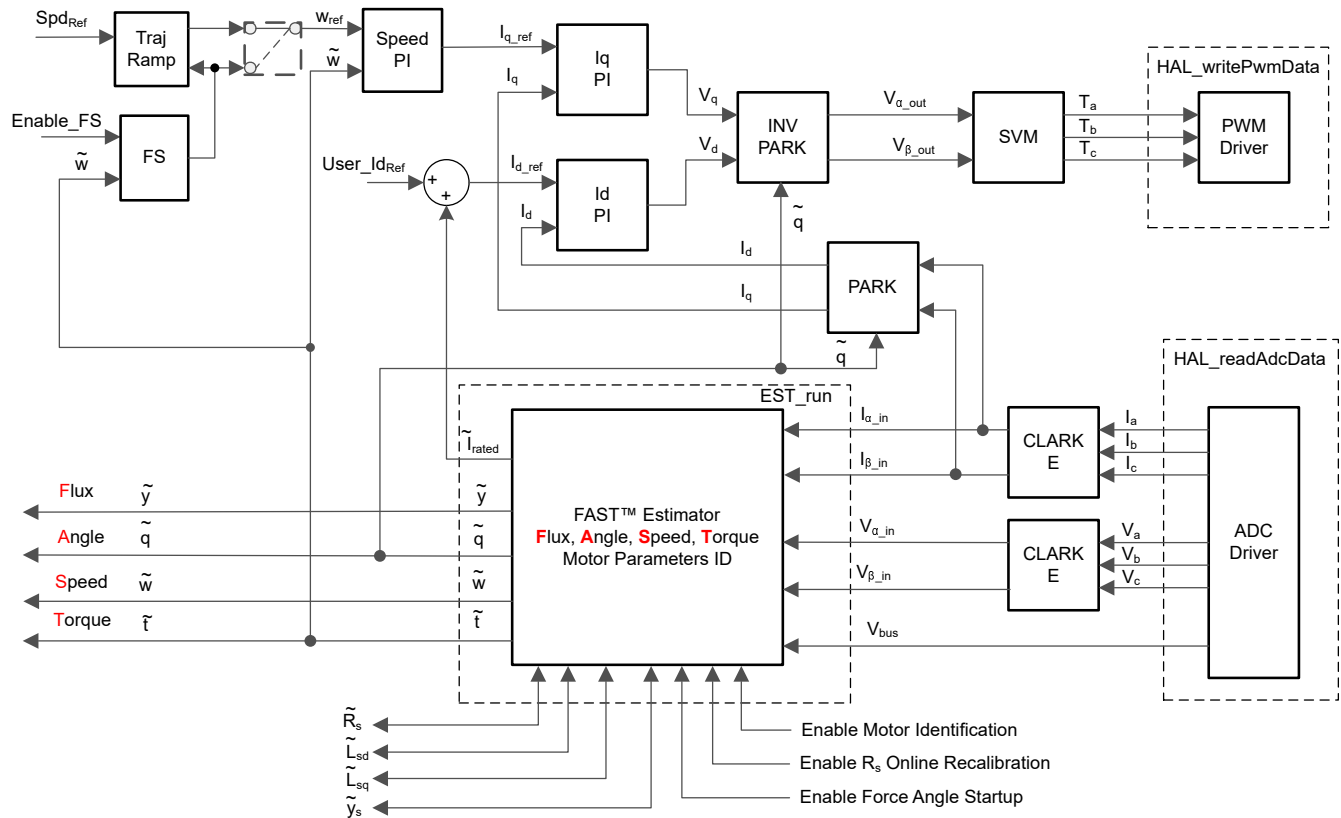


図 2-16. フライング スタート (FS) を備えた FAST を使用した PMSM のセンサレス FOC

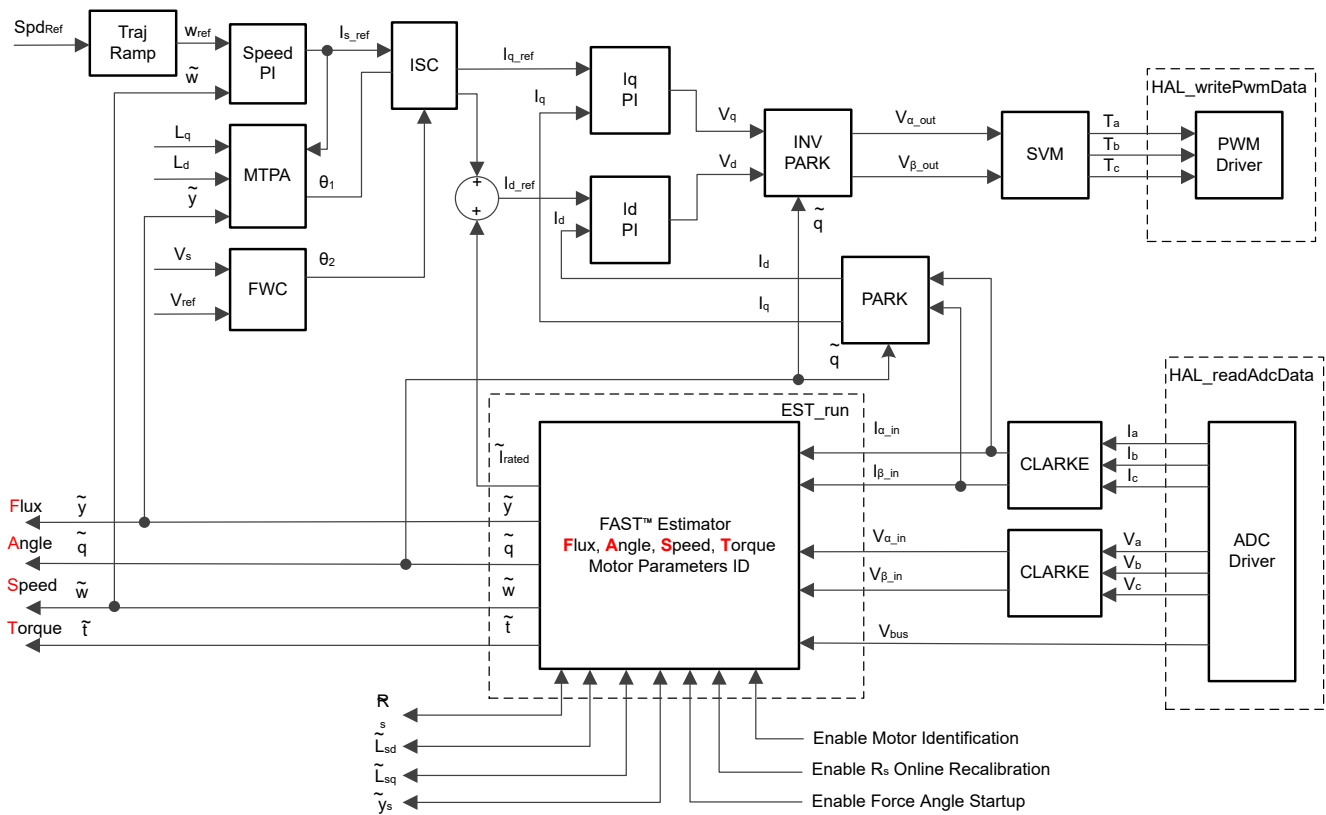


図 2-17. FWC と MTPA を備えた FAST を使用した PMSM のセンサレス FOC

2.4.2.2 PM 同期モーターのセンサレス制御

家電アプリケーションでは、機械センサを使用することにより、コスト、サイズ、信頼性の問題が増します。これらの問題を克服するために、センサレス制御方式が導入されています。機械的な位置センサを使用せずに回転子の速度と位置の情報を取得するために、複数の推定方法が使用されます。スライディング モード オブザーバ (eSMO) には、信頼性、期待される性能、システム パラメータの変動に対する堅牢性など、数多くの魅力的な特長があるため、一般的に利用されています。

2.4.2.2.1 位相ロック ループを備えた拡張スライディング モード オブザーバ

モデル ベースの手法を使用して、モーターが中速または高速で動作する場合に、IPMSM ドライブ システムの位置センサレス制御を実現しています。モデル ベースの手法では、逆 EMF モデルまたは磁束結合モデルによって回転子位置を推定します。スライディング モード オブザーバは、スライディング モード制御に基づいたオブザーバの設計手法です。システムの構造は固定ではなく、システムの現在の状態に応じて意図的に変更され、あらかじめ決められたスライディング モードの軌道に従って強制的に動かされます。応答が速く、堅牢性が優れ、パラメータの変化や外部の変動に対して影響を受けにくいのが利点です。

2.4.2.2.1.1 IPMSM の数学モデルと FOC 構造

IPMSM のセンサレス FOC 構造を 図 2-18 に示します。このシステムでは、IPMSM システムのセンサレス制御を実現するために eSMO を使用し、eSMO モデルは、逆起電力モデルと PLL モデルを組合せて、回転子の位置と速度を推定するように設計されています。

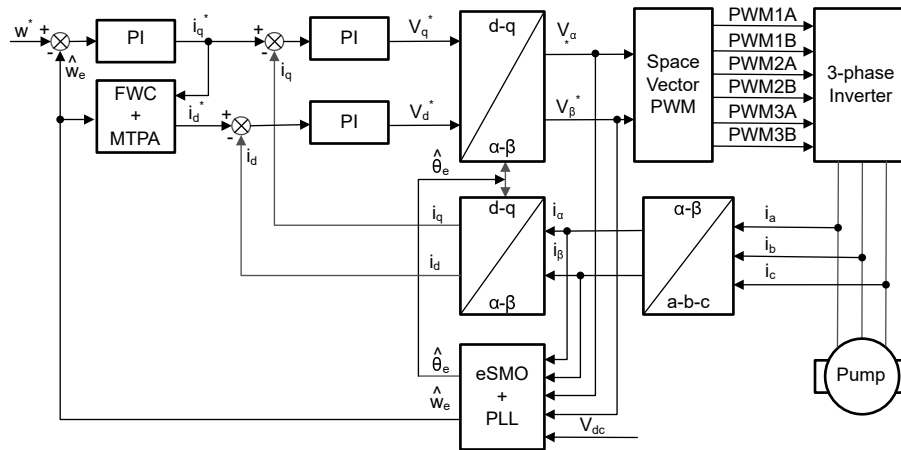


図 2-18. IPMSM システムのセンサレス FOC 構造

IPMSM は、3 相の固定子巻線 (a 軸、b 軸、c 軸) と励起用の永久磁石 (PM) 回転子で構成されています。モーターは、標準的な 3 相インバータによって制御されます。IPMSM は位相 a-b-c の量を用いてモデル化できます。適切な座標変換を行うことで、PMSM の動的モデルを d-q 回転リファレンス フレームと $\alpha\text{-}\beta$ 固定リファレンス フレームで実現できます。これらのリファレンス フレームは 式 9 のような関係にあります。一般的な PMSM の動的モデルは、d-q 回転リファレンス フレームにおいて次のように記述できます。

$$\begin{bmatrix} v_d \\ v_q \end{bmatrix} = \begin{bmatrix} R_s + pL_d & -\omega_e L_q \\ \omega_e L_d & R_s + pL_q \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_e \lambda_{pm} \end{bmatrix} \quad (9)$$

ここで、

- v_d および v_q は、それぞれ q-軸および d-軸の固定子端子電圧です。
- i_d および i_q は、それぞれ d-軸および q-軸の固定子電流です。
- L_d および L_q は、それぞれ q-軸および d-軸のインダクタンスです。
- P は微分演算子で、 $\frac{d}{dt}$ の短縮表記です。
- λ_{pm} は、永久磁石によって生成される磁束結合です。
- R_s は、固定子巻線の抵抗です。

- ω_e は、回転子の電気角速度です。

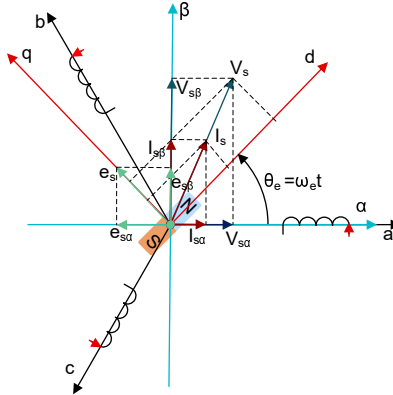


図 2-19. PMSM モデル化のための座標リファレンス フレームの定義

図 2-19 に示すように逆パーク変換を使用することで、PMSM の動的特性は、式 10 に示すように α - β 固定リファレンス フレームでモデル化できます。

$$\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \begin{bmatrix} R_s + pL_d & \omega_e(L_d - L_q) \\ -\omega_e(L_d - L_q) & R_s + pL_q \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} \quad (10)$$

ここで、

- e_α および e_β は、 α - β 軸の拡張起電力 (EEMF) の成分であり、式 11 に示すように定義できます。

$$\begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} = (\lambda_{pm} + (L_d - L_q)i_d)\omega_e \begin{bmatrix} -\sin(\theta_e) \\ \cos(\theta_e) \end{bmatrix} \quad (11)$$

式 10 と 式 11 によると、等価変換と EEMF の概念を導入することで、回転子の位置情報はインダクタンス マトリックスから切り離すことができ、その結果、EEMF が回転子の極位置情報を含む唯一の項となります。さらに、EEMF の位相情報をそのまま回転子の位置観測に利用することができます。固定子電流を状態変数として、IPMSM の電圧式 式 10 を状態式に書き換えます。

$$\begin{bmatrix} \dot{i}_\alpha \\ \dot{i}_\beta \end{bmatrix} = \frac{1}{L_d} \begin{bmatrix} -R_s & -\omega_e(L_d - L_q) \\ \omega_e(L_d - L_q) & -R_s \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \frac{1}{L_d} \begin{bmatrix} v_\alpha - e_\alpha \\ v_\beta - e_\beta \end{bmatrix} \quad (12)$$

直接測定できる唯一の物理量は固定子電流であるため、スライディング サーフェスは固定子電流の経路上で選択されます。

$$S(x) = \begin{bmatrix} \hat{i}_\alpha - i_\alpha \\ \hat{i}_\beta - i_\beta \end{bmatrix} = \begin{bmatrix} \tilde{i}_\alpha \\ \tilde{i}_\beta \end{bmatrix} \quad (13)$$

ここで、

- \hat{i}_α および \hat{i}_β は、推定電流です。
- 上付きの添え字 \wedge は推定値を示します。
- 上付きの添え字 \sim は、観測値と実測値との差を意味する変数誤差を示します。

2.4.2.2.1.2 IPMSM 向け ESMO の設計

図 2-20 に、SMO に組み込まれた通常の PLL を示します。

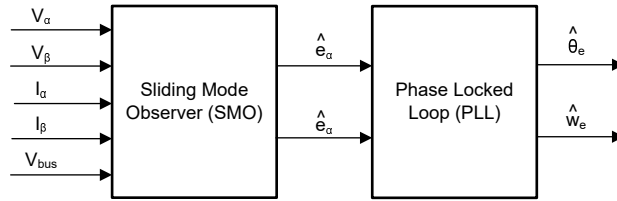


図 2-20. PMSM 向け PLL 搭載 eSMO のブロック図

従来の低次スライディングモードオブザーバは、式 14 に示す数学モデルと、図 2-21 に示すブロック図を使用して構築されます。

$$\begin{bmatrix} \hat{i}_\alpha \\ \hat{i}_\beta \end{bmatrix} = \frac{1}{L_d} \begin{bmatrix} -R_s & -\hat{\omega}_e(L_d - L_q) \\ \hat{\omega}_e(L_d - L_q) & -R_s \end{bmatrix} \begin{bmatrix} \hat{i}_\alpha \\ \hat{i}_\beta \end{bmatrix} + \frac{1}{L_d} \begin{bmatrix} V_\alpha - \hat{e}_\alpha + z_\alpha \\ V_\beta - \hat{e}_\beta + z_\beta \end{bmatrix} \quad (14)$$

ここで、

- z_α および z_β は、スライディングモードの帰還成分であり、式 15 のように定義されます。

$$\begin{bmatrix} z_\alpha \\ z_\beta \end{bmatrix} = \begin{bmatrix} k_\alpha \text{sign}(\hat{i}_\alpha - i_\alpha) \\ k_\beta \text{sign}(\hat{i}_\beta - i_\beta) \end{bmatrix} \quad (15)$$

ここで、

- k_α および k_β は、リアプノフ安定性解析によって設計された一定のスライディングモードゲインです。

仮に k_α および k_β が正で、SMO の安定動作を実現するのに十分であれば、 k_α および k_β は $k_\alpha > \max(|e_\alpha|)$ および $k_\beta > \max(|e_\beta|)$ を保持するのに十分です。

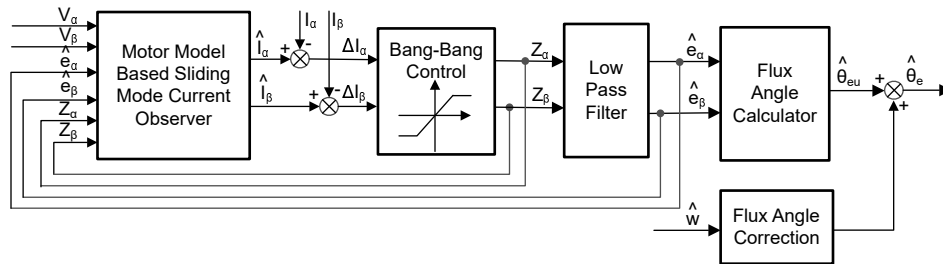


図 2-21. 従来のスライディングモードオブザーバのブロック図

α - β 軸における EEMF の推定値 (\hat{e}_α 、 \hat{e}_β) は、 z_α および z_β の不連続スイッチング信号からローパスフィルタによって求められます。

$$\begin{bmatrix} \hat{e}_\alpha \\ \hat{e}_\beta \end{bmatrix} = \frac{\omega_c}{s + \omega_c} \begin{bmatrix} z_\alpha \\ z_\beta \end{bmatrix} \quad (16)$$

ここで、

- $\omega_c = 2\pi f_c$ は LPF のカットオフ角周波数で、通常は固定子電流の基本周波数に応じて選択されます。

したがって、回転子位置は、式 17 が定義するように、逆起電力のアークタンジェントから直接計算できます。

$$\hat{\theta}_e = -\tan^{-1}\left(\frac{\hat{e}_\alpha}{\hat{e}_\beta}\right) \quad (17)$$

ローパスフィルタは、スライディングモード関数の高周波項を除去することにより、位相遅延を引き起こします。この遅延は、カットオフ周波数 ω_c と逆起電力周波数 ω_e の関係によって、式 18 で定義されているように、補正することができます。

$$\Delta \theta_e = -\tan^{-1}\left(\frac{\omega_e}{\omega_c}\right) \quad (18)$$

次に、SMO 方式によって推定された回転子位置は、式 19 で求められます。

$$\hat{\theta}_e = -\tan^{-1}\left(\frac{\hat{e}_\alpha}{\hat{e}_\beta}\right) + \Delta \theta_e \quad (19)$$

デジタル制御アプリケーションでは、SMO の時間分散式が必要です。時間分散オブザーバに変換するには、オイラー法が適切です。 α - β 座標における式 14 の時間分散システムマトリックスは、式 20 のとおりです。

$$\begin{bmatrix} \hat{i}_\alpha(n+1) \\ \hat{i}_\beta(n+1) \end{bmatrix} = \begin{bmatrix} F_\alpha \\ F_\beta \end{bmatrix} \begin{bmatrix} \hat{i}_\alpha(n) \\ \hat{i}_\beta(n) \end{bmatrix} + \begin{bmatrix} G_\alpha \\ G_\beta \end{bmatrix} \begin{bmatrix} V_\alpha^*(n) - \hat{e}_\alpha(n) + z_\alpha(n) \\ V_\beta^*(n) - \hat{e}_\beta(n) + z_\beta(n) \end{bmatrix} \quad (20)$$

ここで、

- マトリクス [F] and [G] は、式 21 と式 22 のとおりです。

$$\begin{bmatrix} F_\alpha \\ F_\beta \end{bmatrix} = \begin{bmatrix} e^{-\frac{R_s}{L_d}} \\ e^{-\frac{R_s}{L_q}} \end{bmatrix} \quad (21)$$

$$\begin{bmatrix} G_\alpha \\ G_\beta \end{bmatrix} = \frac{1}{R_s} \begin{bmatrix} 1 - e^{-\frac{R_s}{L_d}} \\ 1 - e^{-\frac{R_s}{L_q}} \end{bmatrix} \quad (22)$$

式 16 の時間分散形式は、式 23 のとおりです。

$$\begin{bmatrix} \hat{e}_\alpha(n+1) \\ \hat{e}_\beta(n+1) \end{bmatrix} = \begin{bmatrix} \hat{e}_\alpha(n) \\ \hat{e}_\beta(n) \end{bmatrix} + 2\pi f_c \begin{bmatrix} z_\alpha(n) - \hat{e}_\alpha(n) \\ z_\beta(n) - \hat{e}_\beta(n) \end{bmatrix} \quad (23)$$

2.4.2.2.1.3 PLL による回転子位置および速度の推定

アークタンジェント法では、ノイズや高調波の成分が存在するため、位置と速度の推定精度に影響が出ます。この問題を解消するために、IPMSM のセンサレス制御構造では、PLL モデルを速度と位置の推定に使用できます。セクション 2.4.2.2.1.2 に、SMO とともに使用される PLL 構造を示します。逆 EMF の推定値 \hat{e}_α および \hat{e}_β を PLL モデルで使用すると、図 2-22 に示すように、モーターの角速度と位置を推定できます。

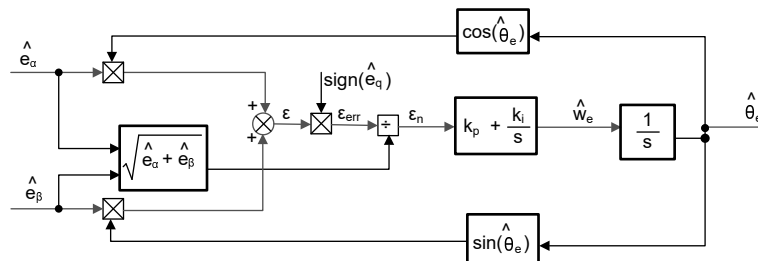


図 2-22. 位相ロックループ位置トラッカーのブロック図

一方、 $e_\alpha = E \cos(\theta_e)$ 、 $e_\beta = E \sin(\theta_e)$ 、 $E = \omega_e \lambda_{pm}$ であることから、位置誤差は式 24 のように定義できます。

$$\varepsilon = \hat{e}_\beta \cos(\hat{\theta}_e) - \hat{e}_\alpha \sin(\hat{\theta}_e) = E \sin(\theta_e) \cos(\hat{\theta}_e) - E \cos(\theta_e) \sin(\hat{\theta}_e) = E \sin(\theta_e - \hat{\theta}_e) \quad (24)$$

ここで、

- E は EEMF の大きさを、モーター速度 ω_e に比例します。

ここで、 $(\theta_e - \hat{\theta}_e) < \frac{\pi}{2}$ の場合、式 24 は式 25 のように簡略化できます。

$$\varepsilon = E(\theta_e - \hat{\theta}_e) \quad (25)$$

さらに、EEMF の正規化後の位置誤差を求めることができます (式 26)。

$$\varepsilon_n = \theta_e - \hat{\theta}_e \quad (26)$$

解析に従うと、直角位相ロックループの位置トラッカーの簡略ブロック図は、図 2-23 のようになります。PLL の閉ループ伝達関数は、式 27 のように表すことができます。

$$\frac{\hat{\theta}_e}{\theta_e} = \frac{k_p s + k_i}{s^2 + k_p s + k_i} = \frac{2\xi\omega_n s + \omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \quad (27)$$

ここで、

- k_p および k_i は、標準的な PI レギュレータの比例ゲインと積分ゲインです。

固有周波数 ω_n と減衰比 ξ は、式 28 のとおりです。

$$k_p = 2\xi\omega_n, \quad k_i = \omega_n^2 \quad (28)$$

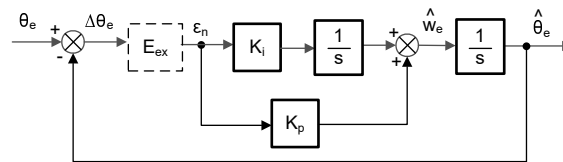


図 2-23. 位相ロック ループ位置トラッカーの概略ブロック図

2.4.2.3 弱め界磁 (FW) および最大トルク/ 電流 (MTPA) 制御

永久磁石同期型モーター (PMSM) は、高電力密度、高効率、幅広い速度範囲により、家電アプリケーションで広く使用されています。PMSM には、表面実装型 PMSM (SPM) と内部実装型 PMSM (IPM) の 2 つの主要なタイプがあります。SPM モーターは、トルクと q-軸電流が線形関係にあるため、制御が容易になっています。一方、IPMSM には、大きな突極性比による電磁トルクとリラクタンストルクがあります。総トルクは、回転子角度に対して非線形です。その結果、IPM モーターで MTPA 技術を使用して、定トルク領域でのトルク生成を最適化することができます。弱め界磁制御は、PMSM ドライブの電力と効率を最大限に高めるために最適化することが目的です。弱め界磁制御は、基本速度以上のモーター動作を可能にし、動作限界を拡大して定格速度を上回る速度に到達させ、速度と電圧の全範囲にわたって優れた制御ができるようになります。

IPMSM の数学モデルの電圧式は、式 29 と式 30 に示すように、d-q 座標で記述できます。

$$v_d = L_d \frac{di_d}{dt} + R_s i_d - p\omega_m L_q i_q \quad (29)$$

$$v_q = L_q \frac{di_q}{dt} + R_s i_q + p\omega_m L_d i_d + p\omega_m \psi_m \quad (30)$$

図 2-24 に、IPM 同期モーターの動的な等価回路を示します。

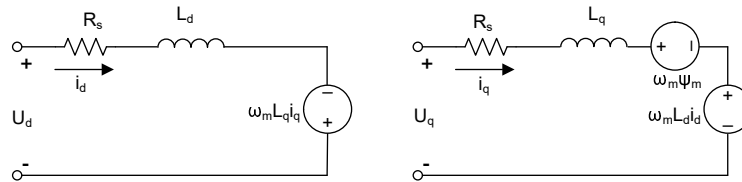


図 2-24. IPMSM 同期モーターの等価回路

IPMSM によって生成される総電磁トルクは式 31 によって表すことができ、生成されるトルクは 2 つの異なる項で構成されます。最初の項はトルク電流 i_q と永久磁石 Ψ_m の間で発生する相互作用トルクに対応し、2 番目の項は d-軸と q-軸のインダクタンスの違いによるリラクタンストルクに対応します。

$$T_e = \frac{3}{2}p[\Psi_m i_q + (L_d - L_q)i_d i_q] \quad (31)$$

ほとんどのアプリケーションでは IPMSM ドライブに速度とトルクの制約があり、これは主にインバータまたはモーターの定格電流と、使用可能な DC リンク電圧の制限によるものです。これらの制約は、数式 32 と 33 で表すことができます。

$$I_a = \sqrt{i_d^2 + i_q^2} \leq I_{\max} \quad (32)$$

$$V_a = \sqrt{v_d^2 + v_q^2} \leq V_{\max} \quad (33)$$

ここで

- V_{\max} と I_{\max} は、インバータまたはモーターの最大許容電圧と電流です。

2 レベル 3 相電圧源インバータ (VSI) によって駆動される機器では、達成可能な最大位相電圧は DC リンク電圧と PWM 方式によって制限されます。空間ベクトル変調 (SVPWM) を採用する場合、最大電圧は式 34 に示す値に制限されます。

$$\sqrt{v_d^2 + v_q^2} \leq v_{\max} = \frac{V_{dc}}{\sqrt{3}} \quad (34)$$

通常、固定子抵抗 R_s は高速動作時は無視できる程度で、定常状態では電流の微分はゼロであるため、式 35 は以下ようになります。

$$\sqrt{L_d^2 \left(i_d + \frac{\psi_{pm}}{L_d} \right)^2 + L_q^2 i_q^2} \leq \frac{V_{\max}}{\omega_m} \quad (35)$$

式 32 の電流制限により半径 I_{\max} の円が d-q 平面上に生成され、式 34 の電圧制限により、速度が上がるにつれて半径 V_{\max} が減少する楕円が生成されます。結果として得られる d-q 平面の電流ベクトルは、電流と電圧の制約に同時に従うように制御されなければなりません。これらの制約に従って、IPMSM の動作領域は、図 2-25 に示すように 3 つに分けられます。

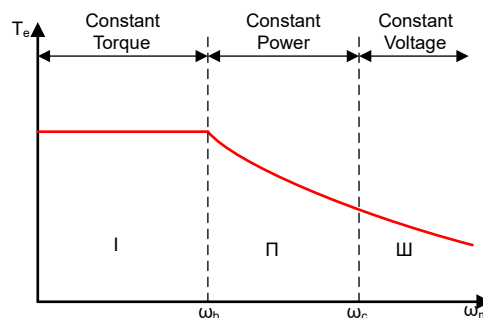


図 2-25. IPMSM 制御の動作領域

1. 定トルク領域:この動作領域では、**MTPA** を実装して最大トルクを生成することができます。
2. 定電力領域:弱め界磁制御の適用が必要であり、トルク容量は電流制約に達すると低下します。
3. 定電圧領域:この動作領域では、深い弱め界磁制御により固定子電圧が一定に保たれ、トルク生成が最大になります。

定トルク領域では、**式 31** に基づき、**IPMSM** の総トルクには、磁束結合による電磁トルクと、 L_d および L_q の間の突極性によるリラクタンストルクが含まれます。電磁トルクは q -軸電流 i_q に比例し、リラクタンストルクは d -軸電流 i_d 、 q -軸電流 i_q 、 L_d および L_q の差の乗算に比例します。

SPM モーターの通常のベクトル制御システムでは、指令された i_d を非弱め界磁モードでゼロに設定することで、電磁トルクを利用するにとどまっていた。ただし、**IPMSM** はモーターのリラクタンストルクを利用する一方で、設計者は d -軸電流も制御しなければなりません。**MTPA** 制御の目的は、リファレンス電流 i_d および i_q を計算し、生成される電磁トルクとリラクタンストルク間の比率を最大にすることです。ここで、 i_d および i_q と、固定子電流のベクトル和 I_s の関係は以下の式で示されます。

$$I_s = \sqrt{i_d^2 + i_q^2} \quad (36)$$

$$I_d = I_s \cos\beta \quad (37)$$

$$I_q = I_s \sin\beta \quad (38)$$

ここで、

- β は同期 (d - q) リファレンス フレームにおける固定子の電流角度です。

式 31 は **式 39** のように表すことができます (I_s は i_d と i_q に置き換えています)。

式 39 は、モータートルクが固定子電流ベクトルの角度に依存することを示しています。

$$T_e = \frac{3}{2} p I_s \sin\beta [\psi_m + (L_d - L_q) I_s \cos\beta] \quad (39)$$

この式は、モーターのトルク差がゼロのときに最大効率点が計算できることを示しています。**MTPA** 点は、この差分 $\frac{dT_e}{d\beta}$ が、**式 40** で示されるように、ゼロのときに見つけることができます。

$$\frac{dT_e}{d\beta} = \frac{3}{2} p [\psi_m I_s \cos\beta + (L_d - L_q) I_s^2 \cos 2\beta] = 0 \quad (40)$$

この式に従うと、**MTPA** 制御の電流角度は、**式 41** のように導くことができます。

$$\beta_{\text{mtpa}} = \cos^{-1} \frac{-\psi_m + \sqrt{\psi_m^2 + 8 \times (L_d - L_q)^2 \times I_s^2}}{4 \times (L_d - L_q) \times I_s} \quad (41)$$

したがって、実際の d -軸と q -軸のリファレンス電流は、**MTPA** 制御の電流角度を用いて、**式 42** と **式 43** で表すことができます。

$$I_d = I_s \times \cos\beta_{\text{mtpa}} \quad (42)$$

$$I_q = I_s \times \sin\beta_{\text{mtpa}} \quad (43)$$

ただし、**式 41** に示すように、**MTPA** 制御の角度 β_{mtpa} は、 d -軸と q -軸のインダクタンスに関係します。つまり、インダクタンスの変動によって、例外的な **MTPA** 点を見つけ出すことができなくなるということです。モーター駆動の効率を高めるために、 d -軸と q -軸のインダクタンスをオンラインで推定しますが、パラメータ L_d および L_q はオンラインでは簡単に測定できない上、飽和効果の影響を受けます。堅牢なルックアップ テーブル (**LUT**) 方式により、電気的パラメータが変動しても制御可能です。通常、数学モデルの簡略化のために、 d -軸と q -軸のインダクタンス間のカップリング効果は無視すること

ができます。したがって、 L_d は i_d のみで変化し、 L_q は i_q のみで変化すると仮定します。その結果、**d**-軸および **q**-軸のインダクタンスは、**式 44** と **式 45** に示すように、それぞれ **d-q** 電流の関数としてモデル化できます。

$$L_d = f_1(i_d, i_q) = f_1(i_d) \quad (44)$$

$$L_q = f_2(i_q, i_d) = f_2(i_q) \quad (45)$$

式 41 を簡略化することで、ISR の計算負担を軽減します。モーター パラメータに基づく定数 K_{mtpa} は **式 47** のように表され、 K_{mtpa} は、更新された L_d および L_q を使用してバックグラウンド ループで計算されます。

$$K_{mtpa} = \frac{\psi_m}{4 \times (L_q - L_d)} = 0.25 \times \frac{\psi_m}{(L_q - L_d)} \quad (46)$$

$$\beta_{mtpa} = \cos^{-1} \left(K_{mtpa} / I_s - \sqrt{(K_{mtpa} / I_s)^2 + 0.5} \right) \quad (47)$$

計算をさらに簡略化するために、2 番目の中間変数 G_{mtpa} (**式 48** に示す) が定義されています。また、 G_{mtpa} を使用して、MTPA 制御の角度 β_{mtpa} は、**式 49** のように計算できます。これら 2 つの計算を ISR で行い、実際の電流角度 β_{mtpa} を求めます。

$$G_{mtpa} = K_{mtpa} / I_s \quad (48)$$

$$\beta_{mtpa} = \cos^{-1} \left(G_{mtpa} - \sqrt{G_{mtpa}^2 + 0.5} \right) \quad (49)$$

いずれの場合も、直軸電流 i_d に作用して、磁束を弱め、達成可能な速度範囲を拡大することができます。この定電力動作領域に入ったことにより、定電力領域と定電圧領域で使用される MTPA 制御の代わりに、弱め界磁制御が選択されます。インバータの最大電圧が制限されるため、永久磁石界磁とモーター速度にほぼ比例する逆起電力がインバータの最大出力電圧を上回るような速度領域では、PMSM モーターは動作できません。PM モーターでは、磁束を直接制御することはできません。ただし、**d**-軸電機子反作用による減磁効果により、負の i_d を加えることでエアギャップフラックスを弱めることができます。電圧と電流の制約を考慮すると、電機子電流と端子電圧は **式 32** と **式 33** のように制限されます。インバータの入力電圧 (DC リンク電圧) の変動により、モーターの最大出力が制限されます。さらに、モーターの最大基本電圧も使用する PWM 方式によって異なります。**式 35** では、IPMSM には、永久磁石の値と、インダクタンスとフラックス電流による 2 つの要素があります。

図 2-26 に、弱め界磁を実装するために使用される代表的な制御構造を示します。 β_{fw} は弱め界磁 (FW) PI コントローラの出力で、リファレンス i_d および i_q を生成します。電圧振幅が限界に達する前は、FW の PI コントローラの入力は常に正であるため、出力は常に 0 で飽和しています。

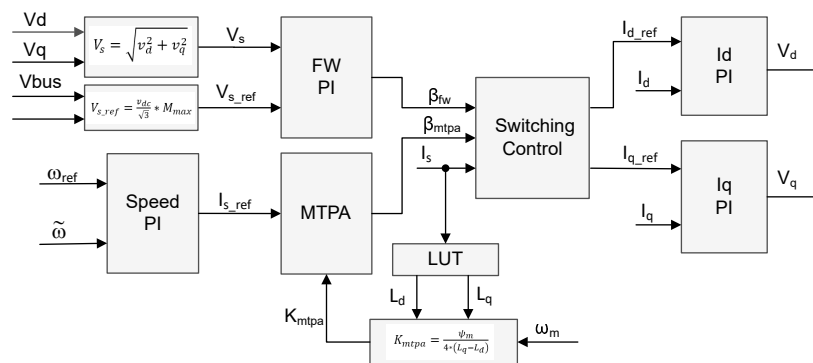


図 2-26. 弱め界磁と最大トルク / 電流制御のブロック図

図 2-15 と **図 2-17** は、FAST または eSMO ベースの FOC ブロック図の実装を示しています。これらのブロック図は、FOC システムの機能と変数を概要を示しています。モーター駆動 FOC システムには、MTPA 制御と弱め界磁制御の 2

つの制御モジュールがあります。これら 2 つのモジュールは、電流角度 β_{mtpa} および β_{fw} をそれぞれ 図 2-27 に示すような入力パラメータに基づいて生成します。

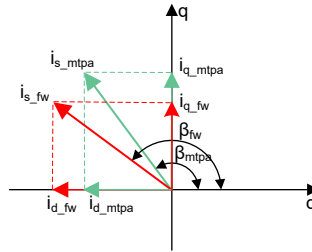


図 2-27. FW および MTPA 時の IPMSM の電流位相図

スイッチング制御モジュールは、印加角度を決定し、リファレンス i_d および i_q を、式 37 と 式 38 に示すように計算するために使用されます。電流角度は、式 50 と 式 51 のように選択されます。

$$\beta = \beta_{fw} \text{ if } \beta_{fw} > \beta_{mtpa} \quad (50)$$

$$\beta = \beta_{mtpa} \text{ if } \beta_{fw} < \beta_{mtpa} \quad (51)$$

図 2-28 のフローチャートに、メイン ループおよび割り込みで FW と MPTA を使用して InstaSPIN™-FOC を実行するために必要な手順を示します。

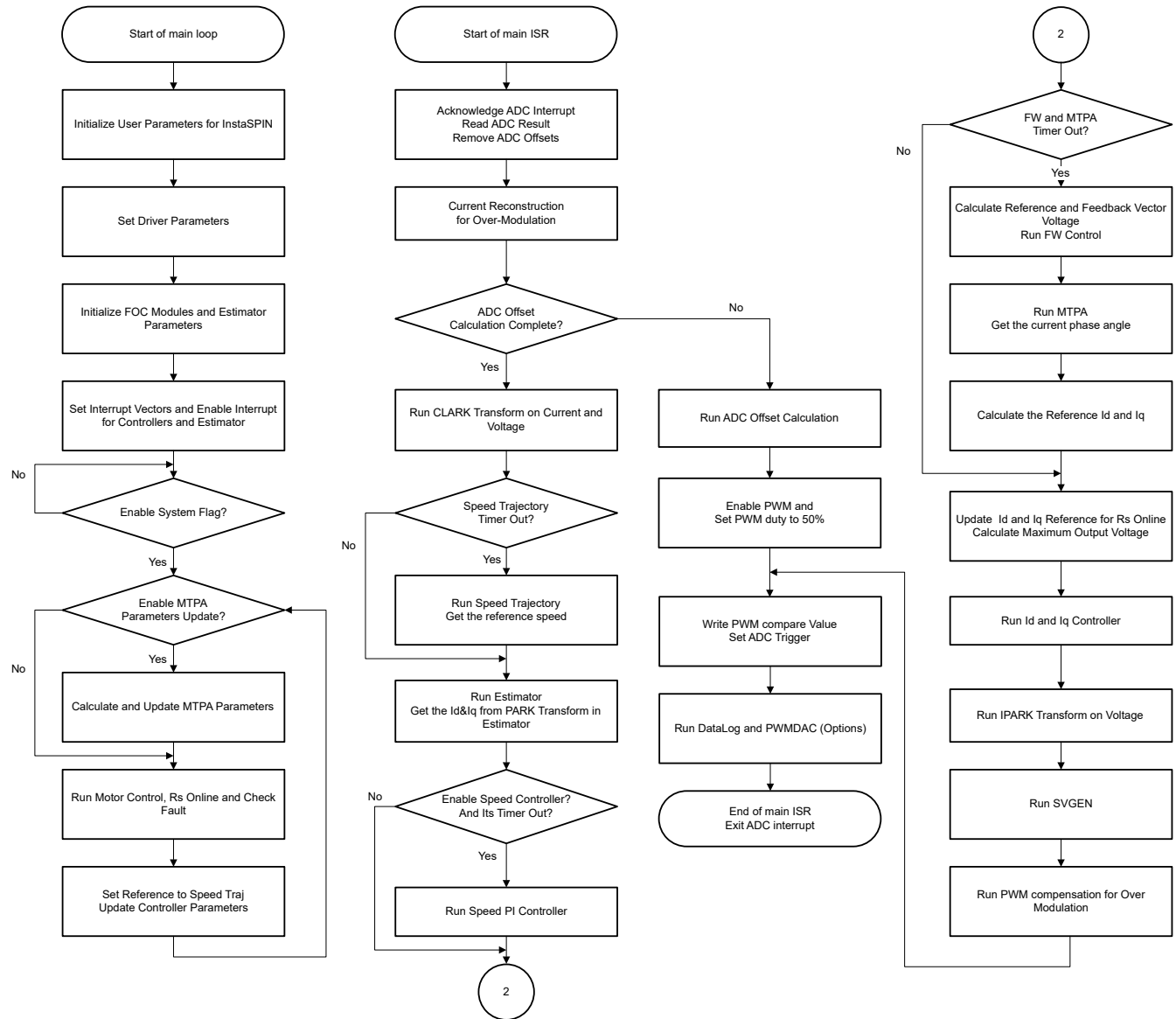


図 2-28. FW と MTPA を使用した InstaSPIN-FOC プロジェクトのフローチャート

2.4.2.4 モーター駆動のハードウェア要件

モーターの制御アルゴリズムは、DC バス電源電圧、各モーター相の電圧、各モーター相の電流など、モーターの状態に関するサンプリング測定値を利用します。モーターを正しく識別し、フィールド オリエンテッド コントロール (FOC) を使用してモーターを効果的に動作させるには、正しい設定が求められるハードウェア依存のパラメータがいくつかあります。以下のセクションでは、FAST または eSMO を使用したモーター制御のための電流スケール値、電圧スケール値、電圧フィルタ極の計算方法を示します。

2.4.2.4.1 モーター電流帰還

モーター相電流の測定には 2 つの技術が用いられています。

- 3 つのシャント電流センシング
- 1 つのシャント電流センシング

プロジェクトのビルド構成では、これら 2 つの電流センシング技術のいずれかを選択できます。セクション 3.3.2 で説明されているように、Flash_MtrInv_3SC ビルド構成は 3 つのシャント電流センシング方式をサポートし、Flash_MtrInv_1SC は 1 つのシャント電流センシング方式をサポートしています。

2.4.2.4.1.1 3 つのシャント電流センシング

モーターを流れる電流は、PWM サイクルごとにモーター制御アルゴリズムの一部としてマイクロコントローラによってサンプリングされます。TMS320F2800137 ドーターボードは 1~3 つのシャント電流センシングに対応し、MSPM0G1507 ドーターボードは 1~2 つのシャント電流センシングに対応しています。モーター相の双方向電流、つまり正負の電流を測定するには、以下の回路では 1.65V のリファレンス電圧が必要です。このオフセットリファレンス電圧は、[図 2-29](#) に示すように、電圧フォロワによって生成されます。

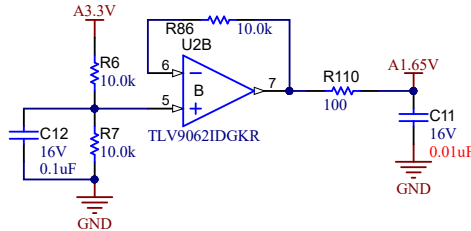


図 2-29. 3.3V 入力回路からの 1.65V リファレンス電圧

[図 2-30](#) は、TMS320F2800137 ドーターボードの場合に、フィルタリング、増幅、ADC 入力範囲の中心へのオフセットにより、モータ電流を電圧信号として表す方法を示しています。この回路は、コンプレッサおよびファンの 3 相 PMSM の各相に使用されます。この回路の伝達関数を [式 52](#) に示します。

$$V_{OUT} = V_{OFFSET} + (I_{IN} \times R_{SHUNT} \times G_i) \quad (52)$$

ここで、

- $R_{shunt} = 0.05\Omega$
- $V_{offset} = 1.65V$

計算された抵抗値により、[図 2-36](#) に示されたセンシング回路が構築されます。 G_i は [式 53](#) で求められます。

$$G_i = \frac{R_{fb}}{R_{in}} = \frac{R18}{(R97 + R15)} = \frac{10\text{ k}\Omega}{20 + 2.4\text{ k}\Omega} = 4.132 \quad (53)$$

マイクロコントローラで測定可能なピークツーピーク電流の最大値は、[式 54](#) で求められます。

$$I_{scale_max} = \frac{V_{ADC_max}}{R_{SHUNT} \times G_i} = \frac{3.3}{0.05 \times 4.132} = 15.97\text{ A} \quad (54)$$

この式ではピークツーピーク値は $\pm 7.99A$ になります。次のコードスニペットは、`user_mtr1.h` ファイルでコンプレッサモーターに対してどのように定義されているかを示しています。

```
//! \brief Defines the maximum current at the AD converter
#define USER_M1_ADC_FULL_SCALE_CURRENT_A (15.97f)
```

電流帰還の極性が正しいことも、マイクロコントローラが正確な電流測定を行う上で重要です。このハードウェアボードの構成では、接地に接続されているシャント抵抗の負のピンは、オペアンプの反転ピンにも接続されています。`user.mtr1.h` での次のコードスニペットに示すように、強調表示されている記号は、ソフトウェアで電流帰還の正しい極性を持つように設定する必要があります。

```
// define the sign of current feedback based on hardware board
#define USER_M1_SIGN_CURRENT_SF (1.0f)
```

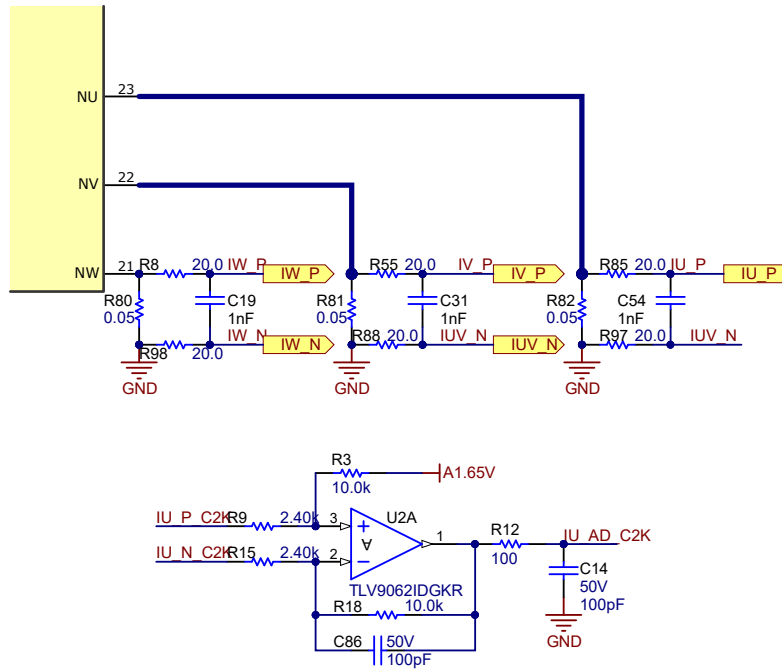


図 2-30. 3 つのシャント電流センシング (TMS320F2800137)

MSPM0 ドーターボードでは、システムのコストを削減するために、2 つの高性能内蔵アンプを使用して 2 つのシャント電流センシングが実装されています。また、アンプのゲインは 4.132 で、カットオフ周波数は 70kHz です。図 2-31 に、MSPM0G1507 ドーターボードにおける 2 つのシャント電流センシング回路を示します。

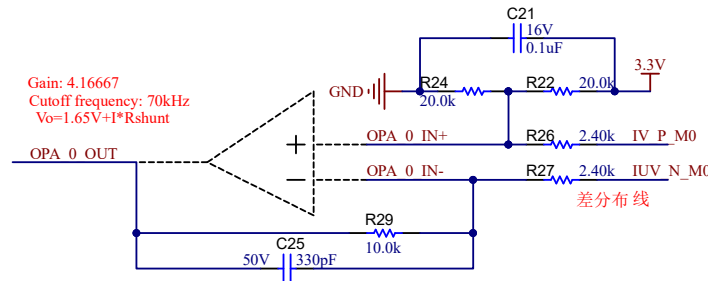


図 2-31. MSPM0G1507 における 2 つのシャント電流センシング回路

2.4.2.4.1.2 1 つのシャント電流センシング

1 つのシャント電流センシング技術により、DC リンクバス電流を測定し、電力 FET のスイッチング状態を把握した上でモーターの 3 相電流を再構築します。1 つのシャント電流センシング技術の詳細については、『単一 DC リンク シャント付き PMSM のセンサレス FOC』アプリケーション ノートを参照してください。

このリファレンスボード上では、図 2-32 に示すように、2 つのシャントを取り外し、パワー モジュールの U、V、W の接地接続を短絡することにより、1 つのシャント電流センシング技術を実装します。

1. マザーボードから電流シャント抵抗 R81 と R82 を取り外し、DC リンク電流をセンシングするシャント抵抗 R80 のみを残します。
2. TMS320F2800137 ドーターボードから C86 を取り外して、シングル シャント サンプリング用に U2A の帯域幅を広げます。
3. MSPM0G1507 ドーターボードから C29 を取り外して、シングル シャント サンプリング用に帯域幅を広げます。
4. NU、NV、NW の各ピンを太いワイヤで接続します。

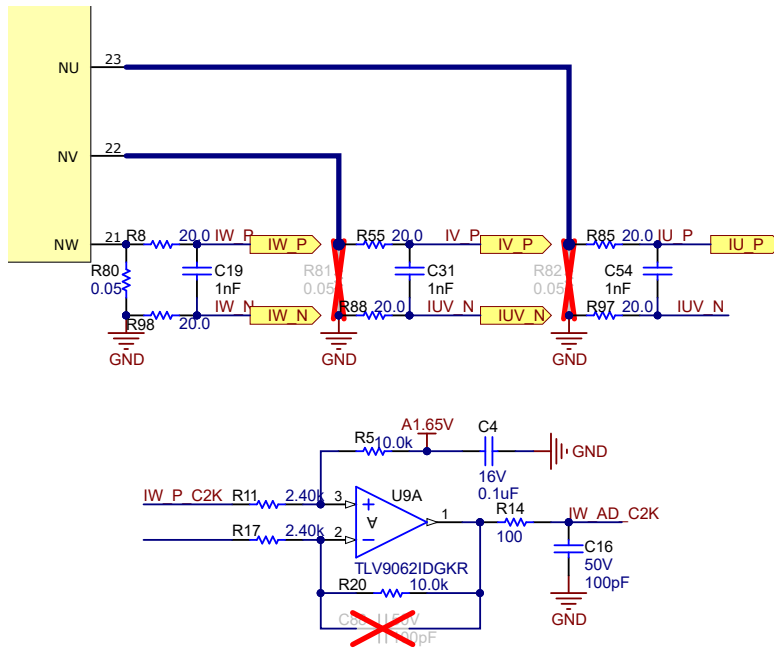


図 2-32. TMS320F2800137 における 1 つのシャント電流センシング回路

図 2-33 に、MSPM0G1507 ドーターボードにおける 1 つのシャント電流センシング回路を示します。

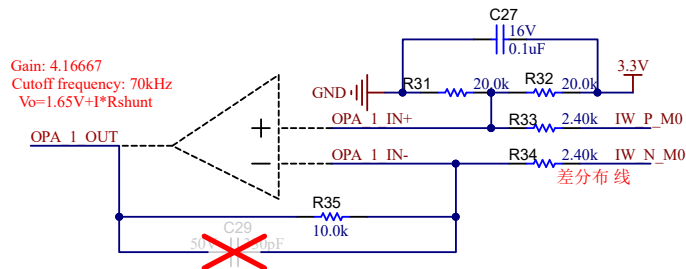


図 2-33. MSPM0G1507 における 1 つのシャント電流センシング回路

デフォルトでは、ボードには 3 つのシャント抵抗があります。図 2-34 にシャント抵抗のレイアウトを示します。1 つのシャント抵抗で動作させるには、R80 を残したまま R81 と R82 を取り外し、NU、NV、NW (R80、R81、R82 のピン 2) を一緒に半田付けします。これで、3 相電流はすべて R80 だけに流れます。

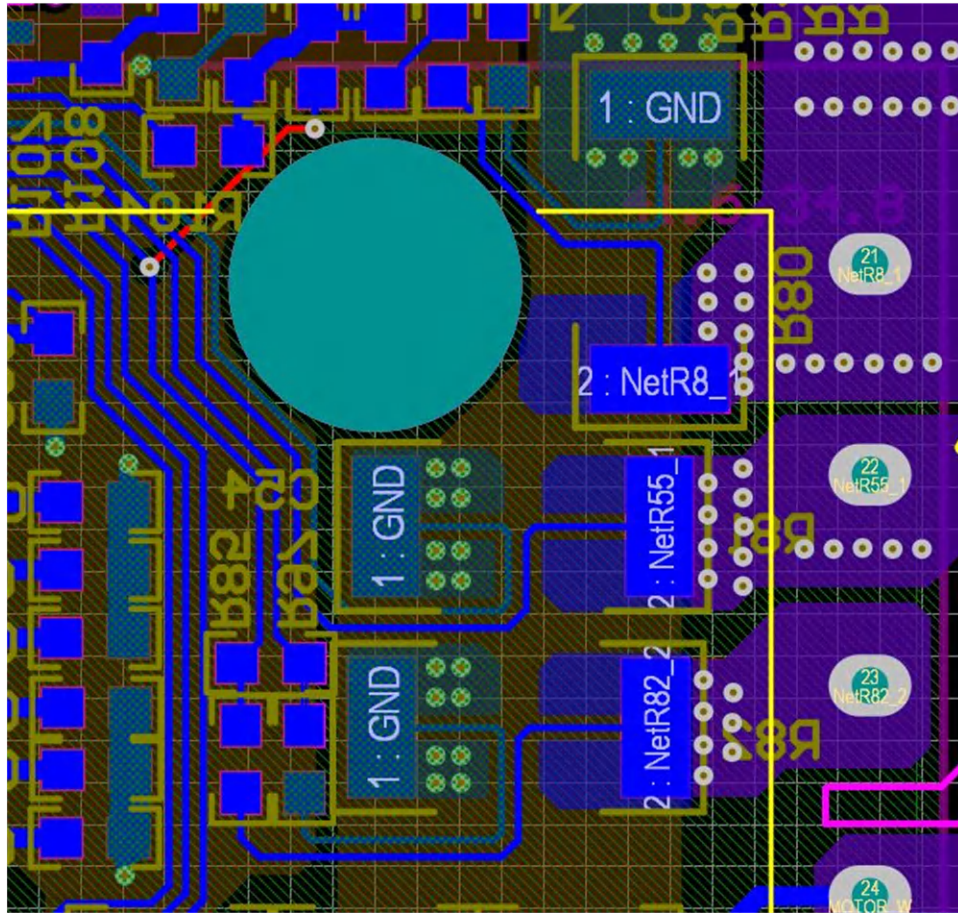


図 2-34. シャント抵抗のレイアウト

DC リンク電流は単方向信号であるため、図 2-35 に示すように、DC リンク電流の ADC サンプル範囲を改善するために、DC リンク電流オフセットは最小値または最大値に設定することができます。TMS320F2800137 ドーターボードで、リファレンス電圧のための抵抗 R7 を 10kΩ から 1kΩ/1% に変更し、DC 電流センシング用にオフセットを 0.3V に調整します。

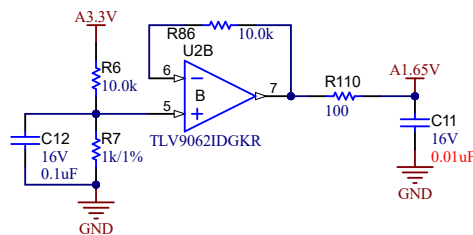


図 2-35. TMS320F2800137 ドーターボードのシングル シャント用 DC オフセットリファレンス

この電流サンプリング回路の伝達関数と、1つのシャントの場合の計算は、3つのシャントの場合と同じです。

MSPM0 ドーターボードの場合、図 2-33 に示すように、R31 を 20kΩ から 2kΩ に下げることによって、1つのシャント電流センシングのオフセットも 0.3V に下げることができます。

2.4.2.4.2 モーター電圧帰還

FAST エスティメータでは最も広い速度範囲で最高の性能を実現できるように電圧帰還が必要で、相電圧はソフトウェアによる推定ではなく、モーター相から直接測定されます。eSMO は、モーター相電圧センシング回路を使用することなく、電圧位相を表すソフトウェア推定値に依存しています。このソフトウェア値 (USER_ADC_FULL_SCALE_VOLTAGE_V)

は、モーター相からの電圧帰還をセンシングする回路に依存します。図 2-36 に、分圧抵抗に基づく電圧帰還回路を使用して、モーター電圧が ADC 入力範囲に対してどのようにフィルタリングされ、スケーリングされるかを示します。同様の回路は、コンプレッサ モーターとファン モーターの両方、および DC バスの 3 つすべての測定に使用されます。

このリファレンス デザインでマイクロコントローラによって測定可能な最大位相電圧帰還は、ADC 入力の最大電圧が 3.3V であることを考慮して、式 55 のように計算できます。

$$V_{FS} = V_{ADC_FS} \times G_V = 3.3 \text{ V} \times 122.46 = 404.13 \text{ V} \quad (55)$$

ここで

- G_V は減衰係数で、式 56 で計算されます。

$$G_V = \frac{(R62 + R67 + R70 + R74)}{R74} = \frac{(332 \text{ k}\Omega + 332 \text{ k}\Omega + 332 \text{ k}\Omega + 8.2 \text{ k}\Omega)}{8.2 \text{ k}\Omega} = 122.46 \quad (56)$$

この電圧帰還回路では、user_mtr1.h で次の設定が行われます。

```
///! \brief Defines the maximum voltage at the AD converter
#define USER_M1_ADC_FULL_SCALE_VOLTAGE_V (404.1292683f)
```

電圧帰還を正確に検出できるように、FAST エスティメータには電圧フィルタの極が必要です。PWM 信号をフィルタリングして除去し、同時に高速な電圧帰還信号がフィルタリングを通過できるように、フィルタを十分に低い周波数に設定してください。一般的なガイドラインとして、5~20kHz の PWM 周波数をフィルタリングで除去するには、数百 Hz のカットオフ周波数があれば十分です。数 kHz 程度の位相電圧周波数を生成するような超高速モーターを動作させる場合にのみ、ハードウェア フィルタを変更してください。

このリファレンス デザインでは、フィルタ極の設定は式 57 で計算できます。

$$f_{\text{filter_pole}} = \frac{1}{(2 \times \pi \times R_{\text{parallel}} \times C)} = 405.15 \text{ Hz} \quad (57)$$

where、

$C = 47\text{nF}$

$$R_{\text{parallel}} = \frac{((332 \text{ k}\Omega + 332 \text{ k}\Omega + 332 \text{ k}\Omega) \times 8.2 \text{ k}\Omega)}{(332 \text{ k}\Omega + 332 \text{ k}\Omega + 332 \text{ k}\Omega) + 8.2 \text{ k}\Omega} = 8.133 \text{ k}\Omega$$

次のサンプルコードでは、user_mtr1.h でどのように定義されているかを示しています。

```
///! \brief Defines the analog voltage filter pole location, Hz
#define USER_M1_VOLTAGE_FILTER_POLE_HZ (416.3602877f)
```

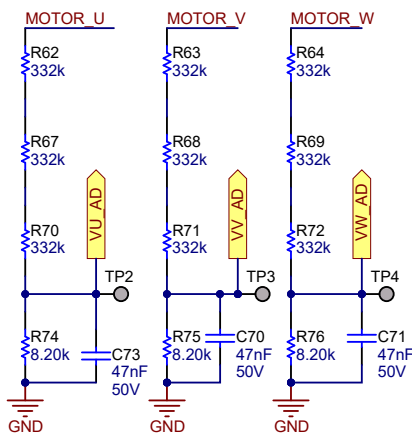


図 2-36. モーター電圧センシング回路

3 ハードウェア、ソフトウェア、テスト要件、テスト結果

3.1 ハードウェアの概要

このセクションでは、設計のボードおよびソフトウェアのテストと検証に必要な機器、テストのセットアップ、および手順について詳しく説明します。

3.1.1 ハードウェア ボードの概要

図 3-1 に、代表的なモーター インバータ システムの概要を示します。

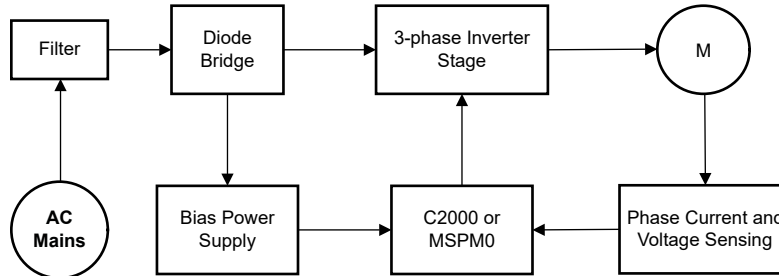


図 3-1. TIDA-010265 のハードウェア ボード ブロック図

モーター制御ボードには、完全なモーター ドライブ システムを実現する機能グループがあります。ボード上の各ブロック (および各機能) は以下のとおりです。図 3-2 は、TIDA-010265 PCB を上から見たもので、さまざまなブロックがあります。

- 電源ライン入力フィルタ
- 3 相インバータ
 - 最大 750W の 3 相インバータで PMSM または IPM をサポート
 - 15kHz のスイッチング周波数
 - 1~3 つのシャント電流センシング
- 制御
 - 48 ピン LQFP パッケージに搭載された単一 TMS320F2800137 または MSPM0G1507 シリーズ MCU
 - アナログ信号に対する増幅と入力フィルタ
- 外付けモーター温度センシング用インターフェイス
- 絶縁型 UART ポート
- 補助電源
 - オンボード電源 +3.3V、+5V、+15V

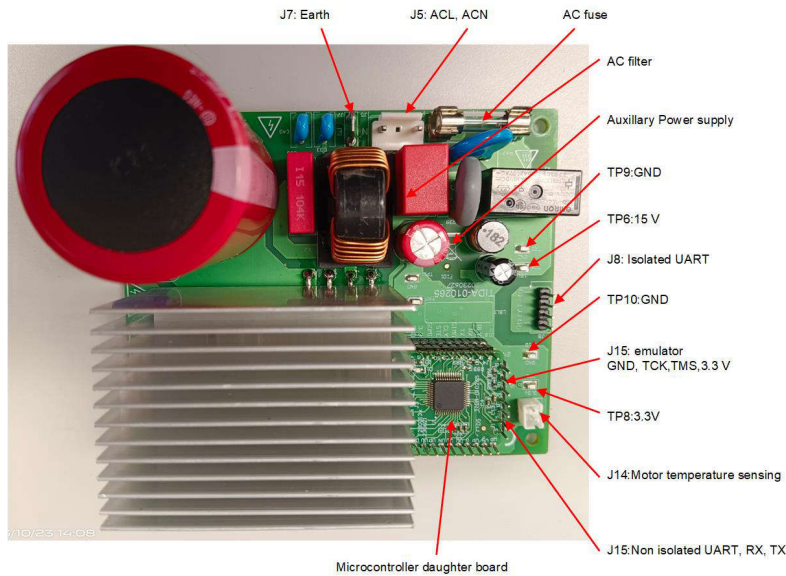


図 3-2. TIDA-010265 リファレンス デザインのボード レイアウト

ボードを使用する際には、以下の事項に注意を払うことを推奨します。

警告

- ボードに通電しているときは、ボードのどの部分にも触れないでください。また、ボードに接続されている部品にも触れないでください。
- AC 電源 (壁面コンセント) を使用してキットに電力を供給します。絶縁型 AC ソースを推奨します。
- 通電中は、ボード、キット、アSEMBリのどの部分にも触れないでください。(パワー モジュールのヒートシンクはボードから絶縁されていますが、高電圧スイッチングにより、ヒートシンク本体にある程度の容量性結合電圧が生成されます。)
- 制御グラウンドは高温になる可能性があります。

3.1.2 テスト条件

リファレンス デザイン ソフトウェアをテストする際は、以下の点に注意してください。

- 入力には、AC 電源を使用する場合は AC 165V~265V、DC 電源を使用する場合は DC 100V~400V の範囲で使用してください。入力 AC 電源の入力電流制限は 10A、DC 電源の入力電流制限は 6.5A に設定してください。ただし、ボードの初期起動時は低い電流制限から始めてください。
- 出力には、ダイナモメータを備えた 3 相 PMSM を使用してください。

3.1.3 ボードの検証に必要なテスト機器

設計者は、ボードの検証に以下の機器を使用する必要があります。

- 絶縁型 AC ソース
- 単相電力アナライザ
- デジタル オシロスコープ
- マルチメータ
- DC 電源
- 750W、3 相 PM 同期モーター
- 動力計
- 3 相電力アナライザ

3.2 GUI の概要

セクション 3.3 で説明しているように、このリファレンス デザインのソースコードは設計者がファームウェアを直接デバッグできるように提供されています。ただし、ソフトウェアのデバッグには時間がかかります。開発期間を短縮するために、UART ベースの GUI ソフトウェアが提供され、カスタマイズされたアプリケーションのパラメータを迅速に調整することができます。このセクションでは、GUI ソフトウェアを使用してモーター制御パラメータをデバッグおよび調整する方法を紹介します。

現時点では、GUI をサポートしているのは C2000 ドーターボードのファームウェアのみになります。

UART 経由で Host PC をこのリファレンス ボードに接続する場合は注意が必要です。これは、AC 整流器によって DC 出力電圧が生成され、その電圧には保護接地から浮いた電位が高い接地が含まれているためです。接地された機器をキットに接続するときは、絶縁トランスを使用する必要があります。

3.2.1 テスト設定

GUI ソフトウェアに必要なものは、Host PC とリファレンス デザイン ボード間の UART 接続のみです。図 3-3 に、GUI を使用してテストを行うためのハードウェア接続を示します。次の手順に従ってハードウェアの設定を行ってください。

1. UART-USB アダプタを介して、J15 の TX、RX、GND のみを Host PC に接続します。アダプタからの 3.3V は必要ありません。
2. モーターの配線を J10 に接続します。
3. マルチメータ、オシロスコープのプロープ、その他の測定機器を接続して、さまざまな信号やパラメータを調べたり、分析したりします。
4. DC バス電源、AC 電源、または AC 主電源を J5 と J7 でインバータに接続し、ボードに電力供給します。
 - a. DC 電源の最大出力は 380VDC です。
 - b. AC 電源の最大出力は 265VAC、50/60 Hz です。
 - c. AC 主電源は 220VAC、50/60 Hz です。

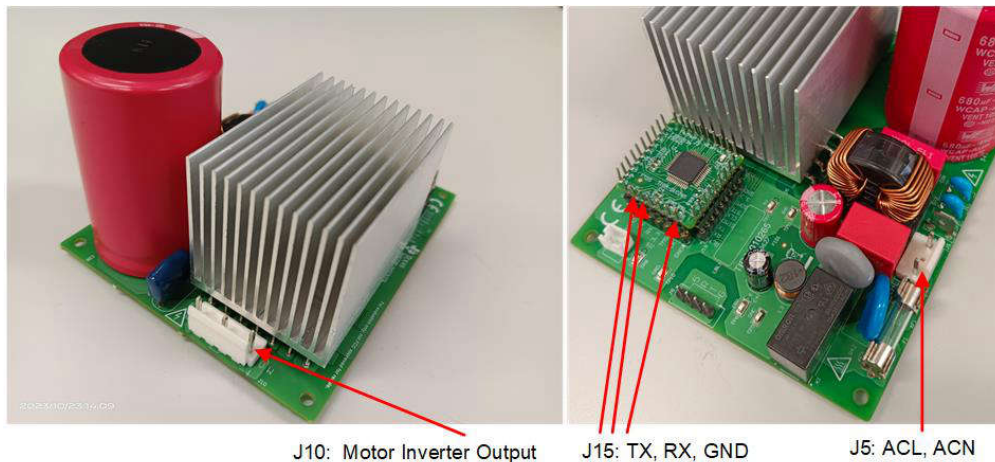


図 3-3. GUI ソフトウェアによるテスト用のハードウェア接続

注

テスト中に外付けエミュレータの接続に問題がある場合は、JTAG 信号と USB ケーブルにフェライト ビーズを追加してください。接続配線は可能な限り短くしてください。

警告

両方の電源ドメインの接地プレーンは、ハードウェア構成に応じて同じでも、異なっても構いません。個人の安全を確保し、機器の損傷を防ぐために、いかなるテスト機器もボードに接続する前に適切な絶縁要件を満たすようにしてください。ボードに電力供給する前に、GND 接続を確認してください。測定機器をボードに接続する場合は、アイソレータが必要です。

3.2.2 GUI ソフトウェアの概要

GUI ソフトウェアは、Microsoft® Windows® ベースのシステムで実行できます。GUI には図 3-4 に示すように、[Control Window]、[Debug Windows]、[Control Parameters]、[Motor Parameters]、[System Parameters]、[Communication Setting] の 6 つのタブがあります。このバージョンの GUI では、[Analysis Window] は使用できません。これらのタブには、モーター制御、識別、制御パラメータの調整、仮想オシロスコープ、MCU フラッシュの読み取り/書き込みなど、複数の機能が用意されています。

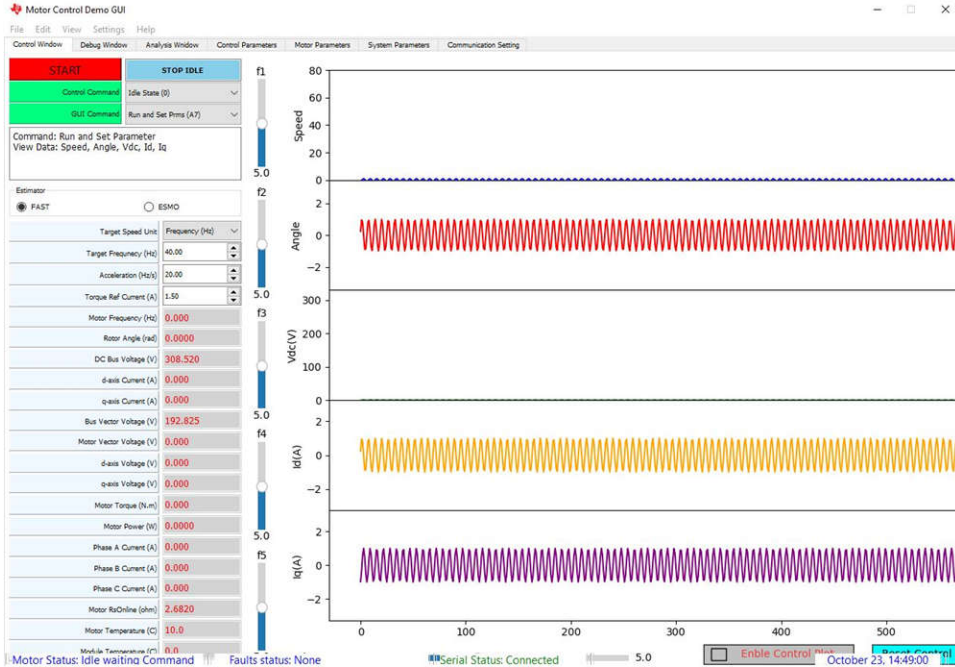


図 3-4. GUI ソフトウェア

3.2.3 シリアル ポートの設定

ホスト PC で GUI ソフトウェアを実行し、GUI ウィンドウがポップアップするまで待ちます。図 3-5 に、UART 経由でホスト PC をこのリファレンス デザイン ボードに接続する手順を示します。デフォルトのボーレートは 258600bps です。異なる UART 速度を使用する場合は、GUI と C2000 の両方でボーレート設定を変更する必要があります。

The screenshot shows the 'Communication Setting' window in the Motor Control Demo GUI. The window is titled 'Check Serial Port' and 'Check Serial Port'. It contains the following settings: Select Serial Port: COM56, Baud Rate: 384600, Data Bits: 8, Parity Bits: None, Stop Bits: 1, Read Timeout (ms): 1000, Write Timeout (ms): 0.00, Enable Baud Rate Auto Tuning: [checked], RX Update Period (ms): 1, TX Update Period (ms): 25, Save data to .csv file: [checked], Save data to .txt file: [unchecked], Save Receive Raw Data to .csv/.txt File: [unchecked], Save Receive Convert Data to .csv/.txt File: [unchecked], Log Motor Control Data to .csv/.txt File: [unchecked], Log Data Period (ms): 10, Graph Update Period (ms): 187, Control Update Period (ms): 100, Select Code File: [empty], Start Code Update: [button], Code Update: [radio], Boot Mode Update: [radio], Full Code Update: [radio], Modules Update: [radio]. The status bar at the bottom shows 'Motor Status: Idle waiting Command', 'Faults status: None', 'Serial Status: Disconnect', and the date 'September 11, 14:48:31'.

1. Select the right serial port supporting UART
2. Configure the baud rate (default rate is 384600 in C2000 project)
3. Click this button to open the serial port
4. Click this button to start transmit and receive data between PC and C2000

Configure the TX and RX period if needed (Default period is calculated per the baud rate)

図 3-5. シリアル ポートの設定

正常に接続され通信が行われているかどうかは、送信データ数と受信データ数で確認できます。図 3-6 に示すように、接続が成功すると両方の数値が連続して増加します。これらの送受信されたデータは、[View Tx Text] および [View Rx Text] をチェックすることで閲覧できます。ただし、モーターの動作中は、これらの機能をチェックしないでください。

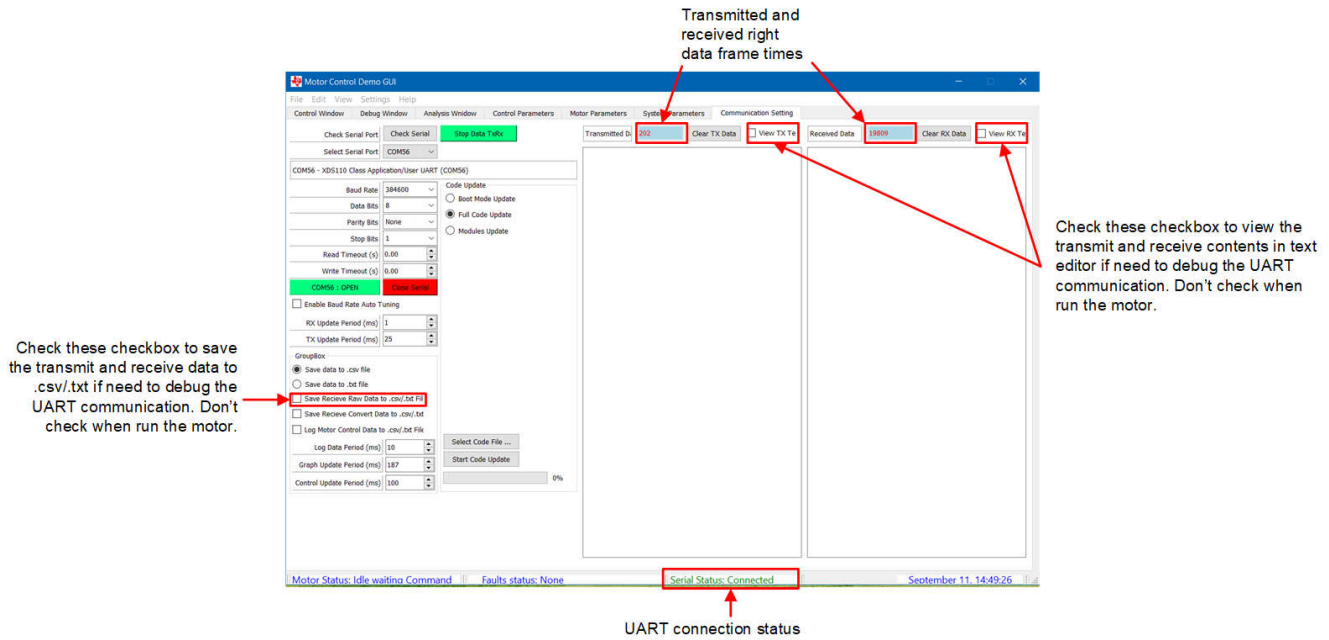


図 3-6. 通信が正常に行われている状態

3.2.4 モーターの識別

正しいモーターパラメータを実装することは、ファームウェアがモーターを正常に制御するために重要です。固定子抵抗、固定子インダクタンス、フラックスなどのパラメータがあり、ファームウェア内のデフォルトモーターに対して、これらのパラメータにデフォルト値が設定されています。

異なる PMSM モーターの場合、これらのパラメータは通常は仕様から見つけることができますが、パラメータが見つからない場合は、GUI ソフトウェアで特定できます。

まず、図 3-7 に示すように、[Control Window] タブでモーター識別コマンドを選択します。



図 3-7. モーター識別コマンド

次に [Motor Parameters] タブを選択します。モーター識別では、モーターに電流を流してモーターパラメータを推定します。固定子抵抗推定用電流、固定子インダクタ推定用電流、R/L 励起周波数 (Hz) などの識別パラメータは、変更することも、デフォルト設定のままにすることもできます。

[START] ボタンをクリックして、モーター識別を開始します。識別中は、可聴ノイズが発生するとともに、モーターが低速で回転します。識別ステータスとモーターパラメータを監視します。識別時間は全体的で 2 分程度になります。図 3-8 に、モーター識別を開始する手順を示します。

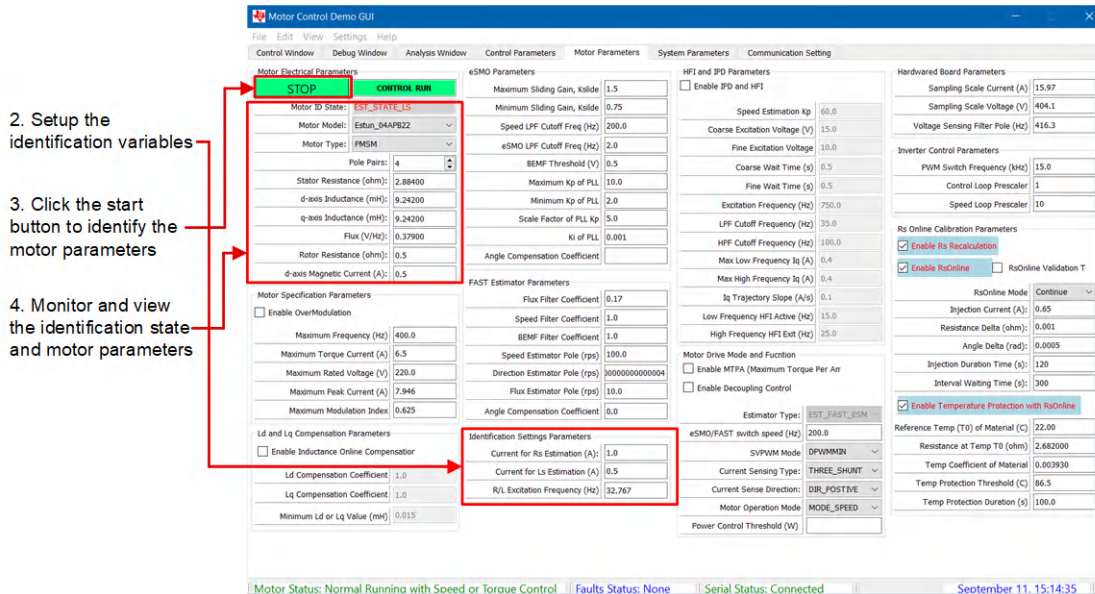


図 3-8. モーター識別の開始

識別が完了したら、モーターペア、固定子抵抗、固定子インダクタンス、フラックスなどを MCU フラッシュに書き込み、これらのパラメータが MCU 内に保存されていることを確認する必要があります。[Control Parameters] タブを選択し、モーターパラメータと制御パラメータを MCU フラッシュに保存するか、またはファイルに保存するかを選択します。書き込みが正常に行われたことを確認するには、[Read Settings from MCU Flash] ボタンをクリックした後、[Motor Parameters] タブを選択して、モーターパラメータが以前に書き込まれたものと同じであることを確認します。図 3-9 に、ここで説明した各ボタンの位置を示します。

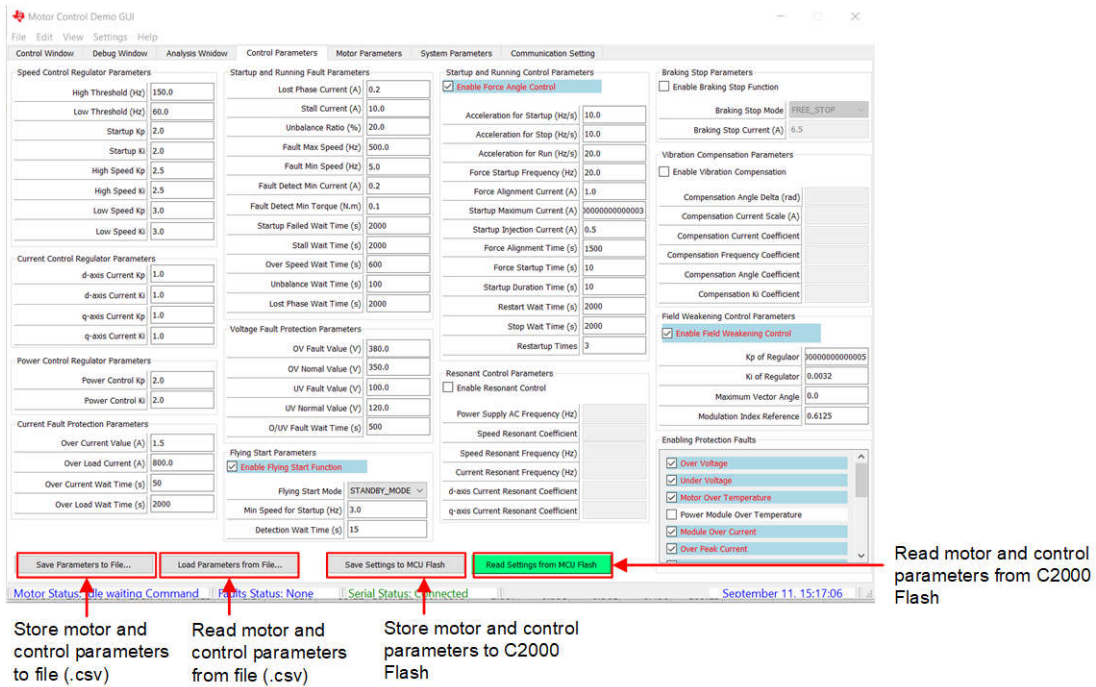


図 3-9. モーター識別結果の保存

3.2.5 モーターの回転

図 3-10 に、モーターパラメータと制御パラメータを示す [Motor Parameters] タブを示します。モーターの電気的パラメータが正しいことを確認してください。

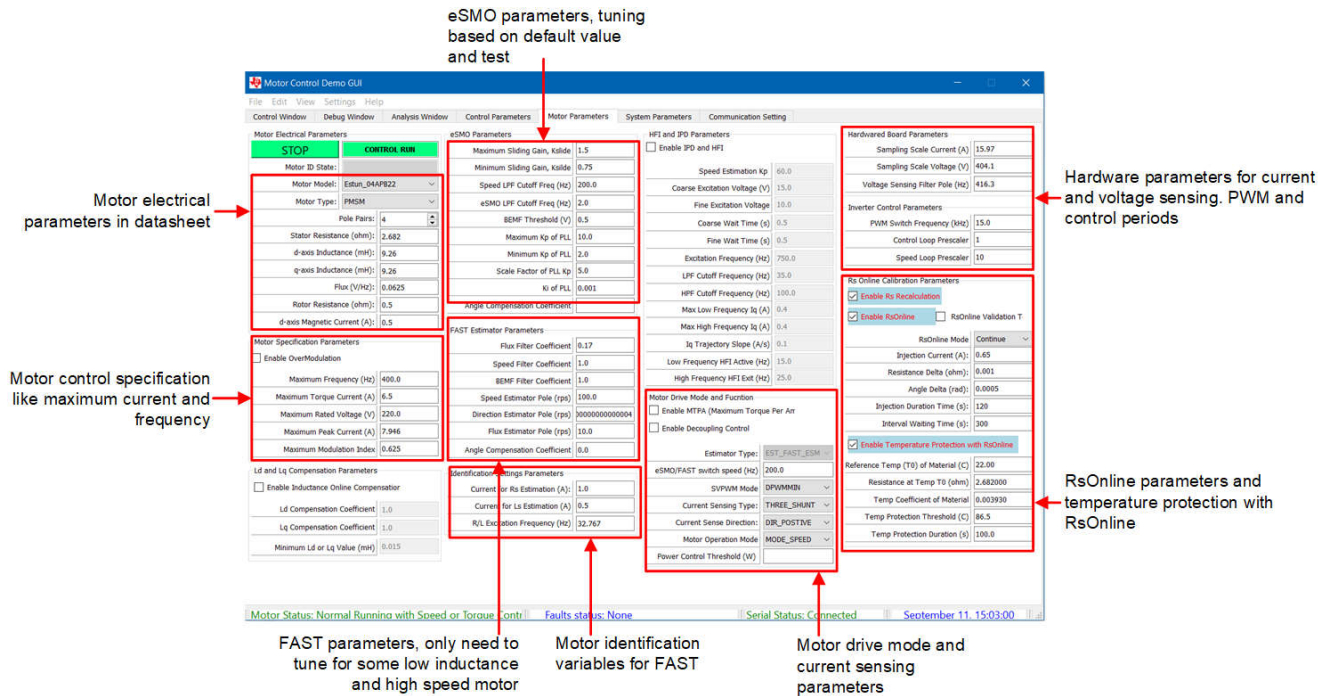


図 3-10. モーター駆動パラメータ

[Control Window] タブで、DC バス電圧が十分に高く (> 230 VDC)、[Control Command] ボタンと [GUI Command] ボタンが緑色で表示されている場合、モーターの回転コマンドは有効になっています。図 3-11 の手順に従って、モーターを回転させます。

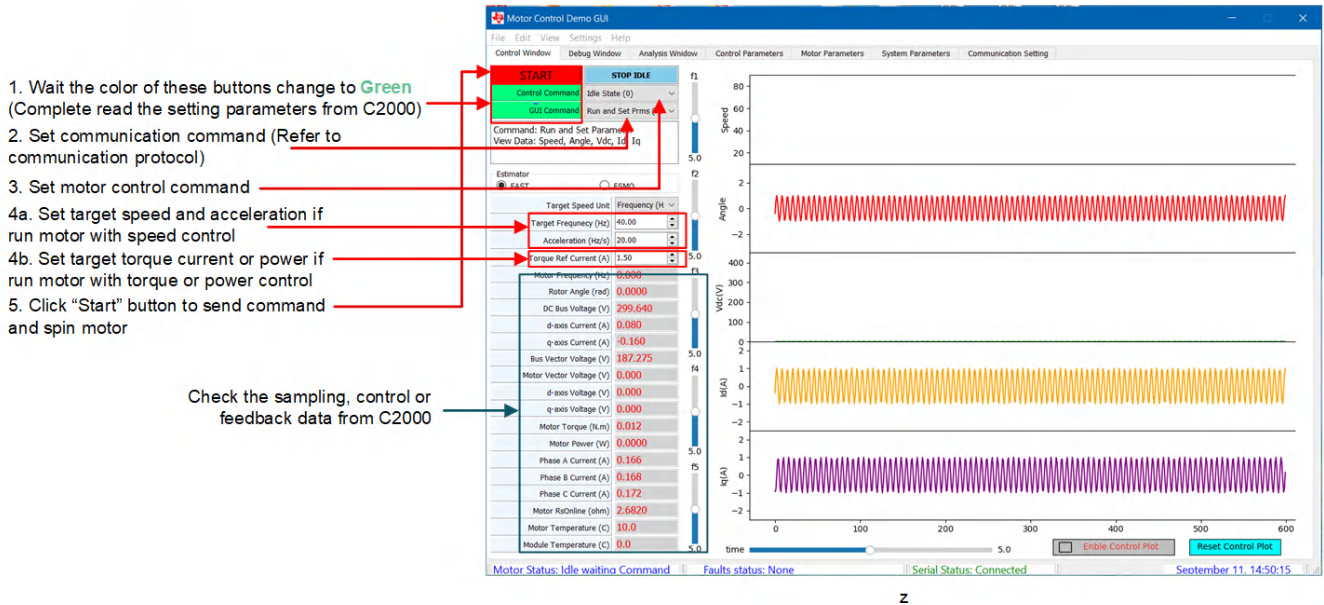


図 3-11. モーターの回転手順

3.2.6 モーターのフォルト ステータス

モーター回転中、特に、モーター パラメータが正しくない場合や、モーター制御パラメータが十分に調整されていない場合には、フォルトが見つかることがあります。[Control Window] を注意深く観察し、図 3-12 に示すように、報告されたすべてのフォルトに注意してください。



図 3-12. フォルト ステータスの監視

図 3-13 に示すように、フォルト検出は有効にも無効にもできます。

注意

一部のフォルトを無効にすると、ボードが損傷することがあります。許容される設定についてよく理解しないまま、フォルト保護を無効にしないでください。

Configure or change the control parameters for flying start, braking, startup, fault protection according to motor or system. The default parameters value are read from the value set in C2000 controller

Check the checkbox to enable/disable the related fault protection according to the system requirement

Configure or change the gain of PI regulators for speed, current, and power control. The values are coefficient, the final gains are these coefficient multiply the setting value in C2000 controller. The setting gains are calculated per the motor electrical parameters.

図 3-13. モーター制御パラメータとフォルト検出の無効化

3.2.7 制御パラメータの調整

[Debug Window] タブでは、図 3-14 に示すように、PI レギュレータの速度、電流、電力制御を調整できます。値は係数であり、最終ゲインは、これらの係数と C2000 コントローラの設定値との乗算になります。設定ゲインは、モーターの電気的パラメータごとに計算されます。図 3-9 に示すように、調整された値は MCU フラッシュに書き込むことができます。

Tune the gain of PI regulators for speed, current, and power control. The values are coefficient, the final gains are these coefficient multiply the setting value in C2000 controller. The setting gains are calculated per the motor electrical parameters.

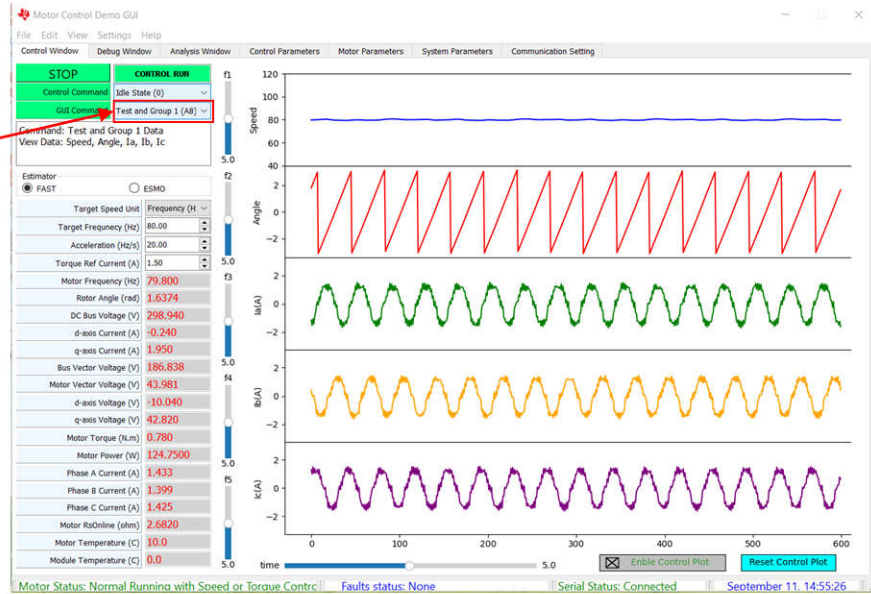
Enable data plot to monitor the speed and current response for tuning the PI regulators

図 3-14. 制御パラメータの調整

3.2.8 仮想オシロスコープ

GUI ソフトウェアには仮想オシロスコープ機能があり、角度、相電流、相電圧の波形を表示できます。図 3-15 に、回転子の角度と相電流を表示するコマンドの構成方法を示します。

Using "Test and Group 1 (A8)" communication command to view the feedback speed, rotor angle and three-phase current (Ia, Ib, and Ic). The target speed and acceleration can't be changed in this mode.

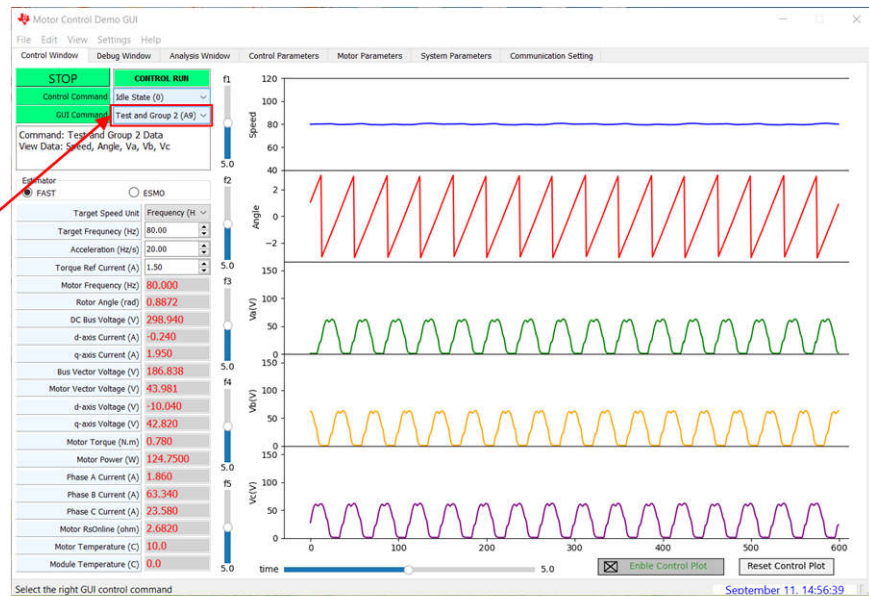


The graph quality is dependent on the maximum baud rate supporting by the hardware board. Much higher baud rate much better graph view

図 3-15. 仮想オシロスコープによる回転子の角度と相電流の表示

図 3-16 に、コマンド Test and Group 2 (A9) を示し、回転子の角度と相電圧を示します。

Using "Test and Group 2 (A9)" communication command to view the feedback speed, rotor angle and three-phase voltage (Va, Vb, and Vc). The target speed and acceleration can't be changed in this mode.

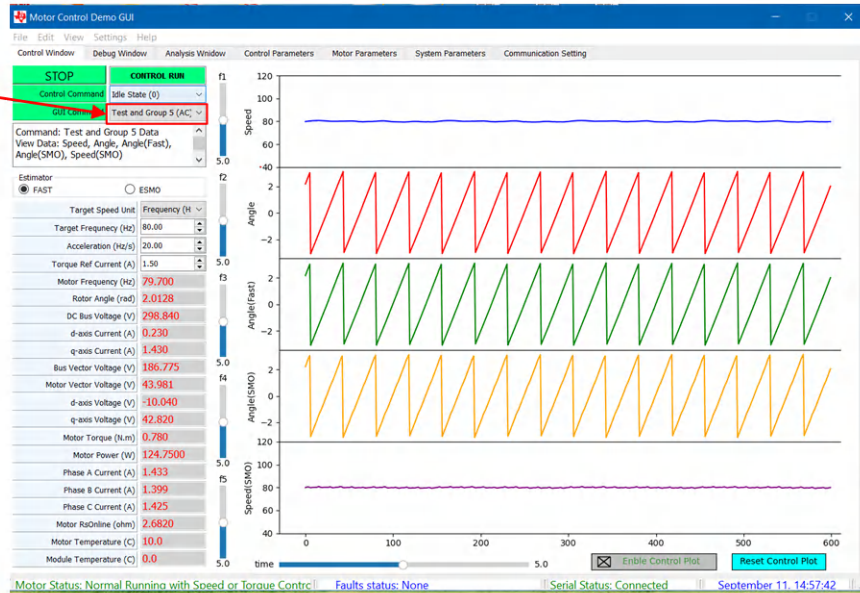


The graph quality is dependent on the maximum baud rate supporting by the hardware board. Much higher baud rate much better graph view

図 3-16. 仮想オシロスコープによる回転子の角度と相電圧の表示

図 3-17 に示すように、コマンド Test and Group 5 (AC) は FAST と EMO の回転子角度を表示することができます。

Using "Test and Group 5 (AC)" communication command to view the rotor angle and speed from FAST or eSMO. The target speed and acceleration can't be changed in this mode.



The graph quality is dependent on the maximum baud rate supporting by the hardware board. Much higher baud rate much better graph view

図 3-17. 仮想オシロスコープによる FAST と eSMO の回転子角度の表示

3.3 C2000 ファームウェアの概要

テキサス・インスツルメンツが提供するリンクから、[C2000WARE-MOTORCONTROL-SDK v5.01.00.00](#) またはそれ以降のソフトウェアをダウンロードしてインストールしてください。この Motor Control SDK ソフトウェアはデフォルトフォルダにインストールしてください。ソフトウェア プロジェクトは、`<install_location>\solutions\tida_010265_wminv\` にある C2000Ware Motor Control SDK フォルダ内に格納されます。次の手順に従って、さまざまなインクリメンタル ビルドでこのコードをビルドして実行してください。

3.3.1 ボード テストに必要なソフトウェアのダウンロードとインストール

1. [Code Composer Studio \(CCS\) 統合開発環境 \(IDE\)](#) ツール フォルダから Code Composer Studio™ IDE をダウンロードしてインストールします。バージョン 12.5 またはそれ以降をお勧めします。
2. C2000WARE-MOTORCONTROL-SDK を次のいずれかの方法でインストールします。
 - [C2000Ware MotorControl SDK](#) ツール フォルダからソフトウェアをダウンロードします。
 - CCS にアクセスし、[View] → [Resource Explorer] に進みます。テキサス・インスツルメンツの Resource Explorer 下で、[Software] → [C2000Ware_MotorControl_SDK] にアクセスし、[Install] ボタンをクリックします。
3. インストールが完了したら、[CCS] を閉じ、プロジェクトをインポートするための新しいワークスペースを作成します。

注

このリファレンスデザインは、使いやすいグラフィカル インターフェイスを使用してデバイスのピン構成とデバイス ペリフェラルの初期化を行う SysConfig をサポートしています。現在のリリースでは、この機能は参照用のみとなっています。設計者は [SysConfig](#) をダウンロードし、『[C2000 SysConfig ソフトウェア ガイド](#)』を参照して SysConfig を実装し、リファレンス デザインをボードに移行してデバイス構成を行うことができます。

3.3.2 CCS でのプロジェクトの開始

F280013x ベースのリファレンス デザイン用 projectspec ファイルは、以下のディレクトリにあります。

```
<install_location>\solutions\tida_010265_wminv\f280013x\ccs\motor_control
```

図 3-18 に示すように、CCS 内にプロジェクトをインポートし、プロジェクト名を右クリックして適切なビルド構成を選択します。HVAC リファレンス デザインに適したビルド構成を選択してください。Flash_MtrInv_3SC ビルド構成は 3 つのシャント電流センシング方式をサポートし、Flash_MtrInv_1SC は 1 つのシャント電流センシング方式をサポートします。

[Project] → [Import CCS Projects] をクリックし、プロジェクトを構成して、プロジェクト内のサポート機能を選択します。

F280013x ベースのリファレンス デザインの

```
<install_location>\solutions\tida_010265_wminv\f280013x\ccs\motor_control
```

をブラウズした後、インポートしたプロジェクト名を右クリックし、図 3-19 に示すように、[Properties] コマンドをクリックしてプロジェクトの事前定義シンボルを設定します。

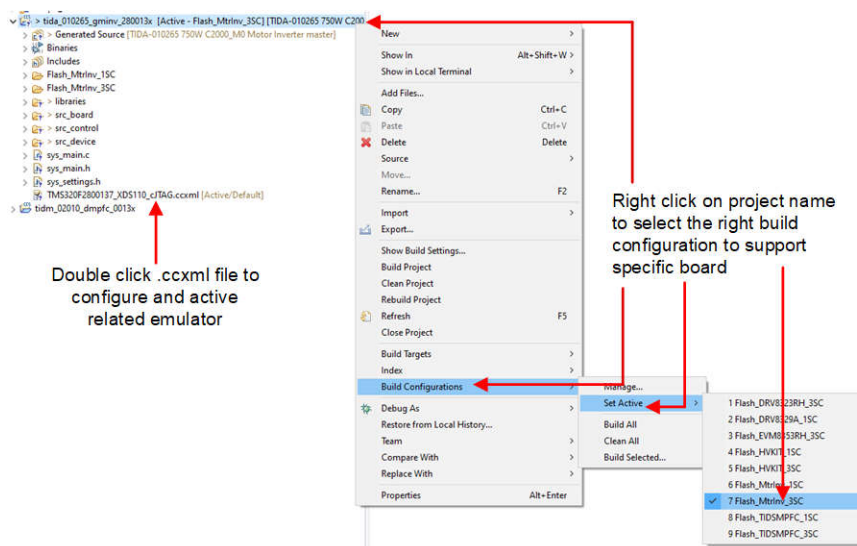


図 3-18. 適切なビルド構成の選択

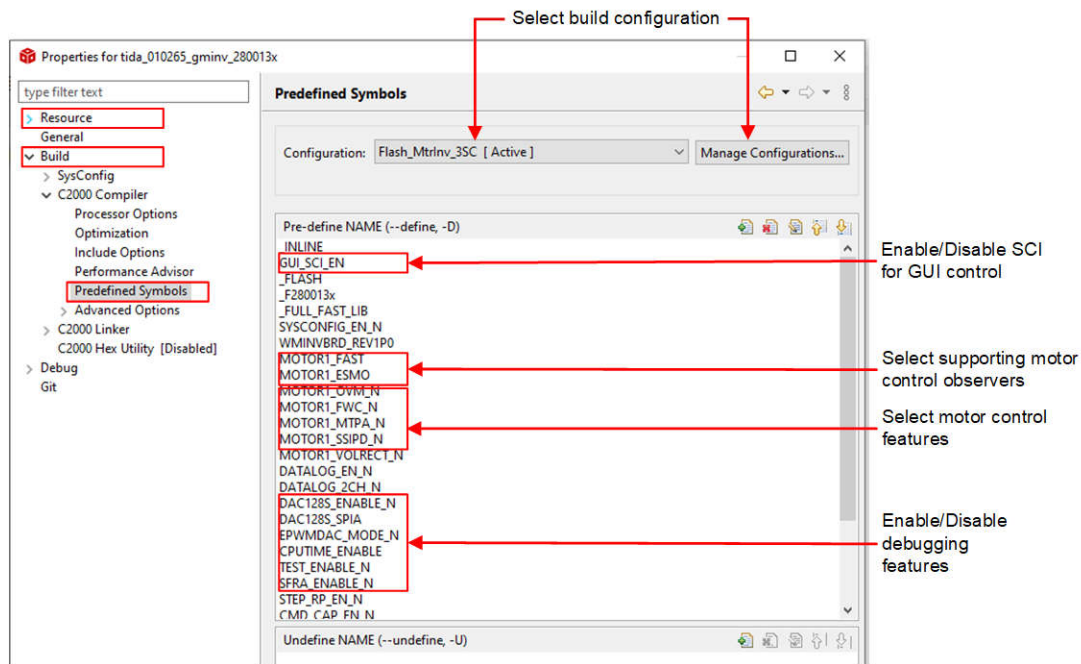


図 3-19. プロジェクト プロパティで適切な事前定義済みシンボルを選択する

3.3.3 プロジェクト構造

プロジェクトがインポートされると、図 3-20 に示すように、CCS 内にプロジェクト エクスプローラが表示されます。デバイス ペリフェラルの構成は、C2000Ware driverlib に基づいています。ユーザーは、hal.c と hal.h のコードと定義のみを変更する必要があります。

src_control フォルダ には hal.c と user_mtr1.c が含まれており、ユーザーはコードや定義を変更することができます。

src_board フォルダ には、このハードウェア ボード用のボードドライバが含まれています。

src_control フォルダには、割り込みサービス ルーチンおよびバックグラウンド タスク内でモーター制御コア アルゴリズム関数を呼び出すモーター駆動制御ファイルが含まれています。

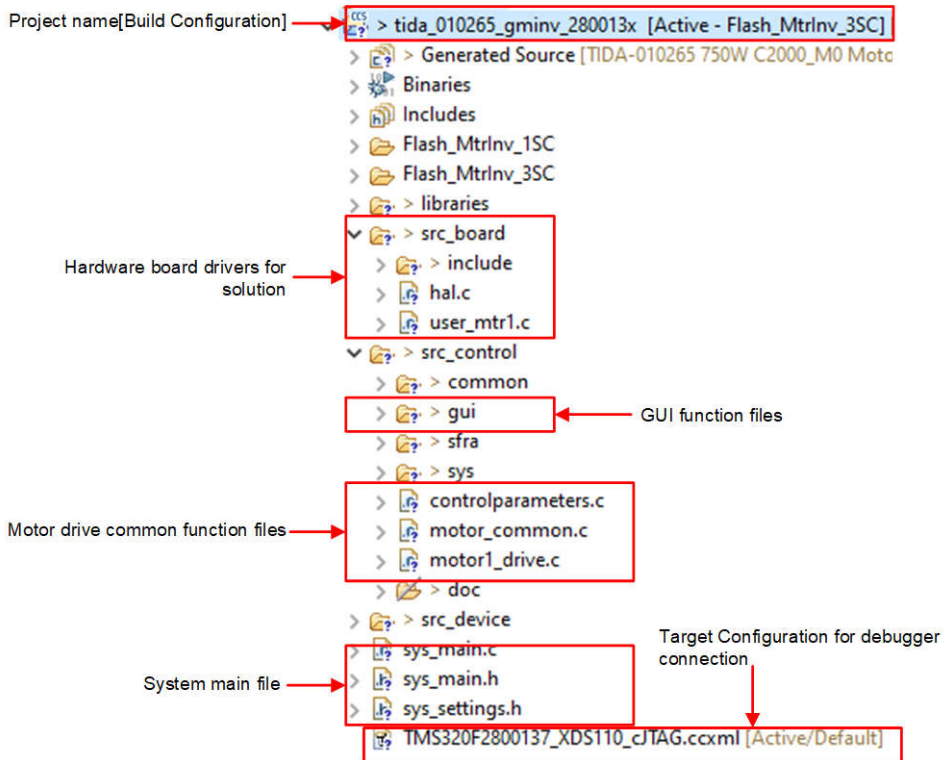


図 3-20. TIDA-010265 プロジェクト エクスプローラ ビュー

図 3-21 に、モーター制御用 ISR のプロジェクトソフトウェア フロー図を示します。これは、バックグラウンド ループでモーター制御パラメータを更新するメインループになります。

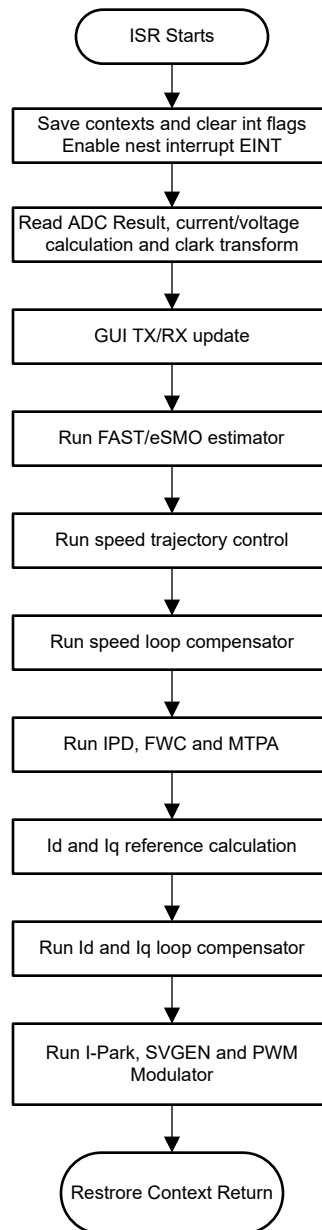


図 3-21. ファームウェア プロジェクトのフロー図

このプロジェクトは、PWM サイクルごとに呼び出されるモーター制御割り込みサービス ルーチンで構成されています。いくつかのバックグラウンド タスクが `main()` 内で永遠にループして呼び出され、絶対的なタイミング精度を必要としない低速タスクの実行や、モーター制御パラメータの更新などに使用できます。低速のバックグラウンド タスクのトリガには、CPU タイマが使用されます。

`motor1CtrlISR` は、`USER_M1_ISR_FREQ_Hz` で周期的にトリガされるモーター 1 を回転させるモーター駆動制御アルゴリズムを呼び出すために予約されています。

システムを簡素化するため、このリファレンス デザインのソフトウェアの立ち上げと設計は、インクリメンタル ビルド (`DMC_BUILDLEVEL`) を使用した 4 つのラボに分かれています。これによって、ボードとソフトウェアの学習と習熟を効果的に進めることができます。また、このアプローチは、基板のデバッグやテストにも適しています。表 3-1 に、インクリメンタル ビルド オプションの詳細を示します。特定のビルド オプションを選択するには、`sys_settings.h` で対応する

BUILDELEVEL オプションを選択します。ビルド オプションを選択したら、rebuild all コンパイラ オプションを選択して、プロジェクトをコンパイルします。セクション 3.3.4 では、各ビルド オプションを実行するための詳細を説明します。

表 3-1. インクリメンタル ビルド オプション

動作	ビルド オプション	説明
モーター駆動	DMC_LEVEL_1	50% PWM デューティ、ADC オフセット較正、PWM 出力、位相シフトの検証
	DMC_LEVEL_2	モーターの電流と電圧のセンシング信号をチェックする開ループ v/f 制御
	DMC_LEVEL_3	ハードウェア設定をチェックする閉電流ループ
	DMC_LEVEL_4	InstaSPIN-FOC または eSMO を使用したモーター パラメータの識別と実行

3.3.4 テスト方法

警告

ボードには**高電圧**が印加されています。このボードを安全に評価するには、絶縁型で電流制限された適切な電源を使用してください。ボードに電源を印加する前に、適切な抵抗性負荷または電子負荷を出力に接続する必要があります。電源が印加されているときは、ユニットの取り扱いを行わないでください。適切な定格の機器のみを使用し、適切な絶縁方法と安全対策に従ってください。

注意

スコープや他のテスト機器をボードに接続する場合は注意が必要です。これは、AC 整流器によって DC 出力電圧が生成され、その電圧には保護接地から浮いた**電位が高い接地**が含まれているためです。接地された機器をキットに接続するときは、絶縁トランスを使用する必要があります。

3.3.4.1 ビルドレベル 1: CPU とボードの構成

このビルド レベルの学習目標:

- システムの開ループ動作を評価する
- HAL オブジェクトを使用して MCU コントローラを設定し、インバータを初期化する
- PWM ドライバ モジュール と ADC ドライバ モジュールを検証する
- CCS の操作に慣れる

このシステムは開ループ制御で動作しているため、ADC での測定値はこのビルド レベルでは計測目的でのみ使用されます。このビルド レベルでは、MCU コントローラとゲートドライバ用のバイアス電源のみが使用されます。高電圧の AC 電源と DC 電源はインバータには実装されていません。

このビルド レベルでは、固定デューティ サイクルによりボードが開ループ方式で実行されます。モーターのデューティ サイクルは 50% に設定されています。このビルド レベルでは、電力段からの帰還値の検出と PWM ゲートドライバの動作を検証し、ハードウェアに問題がないことを確認します。さらに、入出力電圧センシングの較正もこのビルド レベルで実行できます。このビルド レベルのソフトウェア フローを [図 3-22](#) に示します。

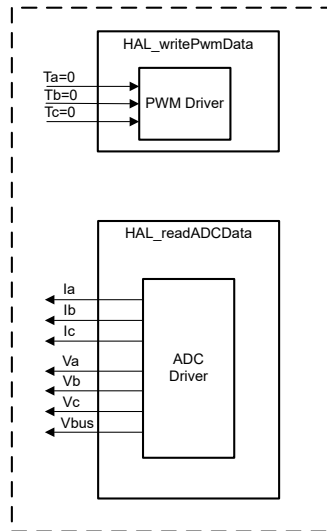


図 3-22. 制御ソフトウェアのブロック図:ビルド レベル 1 – オフセット検証

3.3.4.1.1 CCS を起動し、プロジェクトを開く

CCS を起動してプロジェクトを開くには、次の手順を行います。

1. エミュレータを J15 に接続します。
2. 図 3-23 に示すように、AC 電源または DC 電源を J5 に接続します。
3. CCSv12.5 (またはそれ以降) を開きます。プロジェクトには、C2000 コントローラ ベースのハードウェアで実行できる実行形式出力ファイル (.out) の生成に必要な、すべてのファイルとビルド オプションが含まれています。メニューバーで、[Project] → [Import CCS Project] をクリックします。[Select search-directory:] の下で C2000Ware Motor Control SDK フォルダをブラウズし、<install_location>\solutions\tida_010265_wminv を選択します。[Finish] をクリックして、関連プロジェクトを CCS にインポートします。このプロジェクトは、プロジェクトのビルドに必要なすべてのツール (コンパイラ、アセンブラ、リンカ) を呼び出します。
4. 左側のプロジェクト ウィンドウで、[Project] の左側にあるプラス記号 (+) をクリックします。プロジェクト ウィンドウの例を図 3-20 に示します。

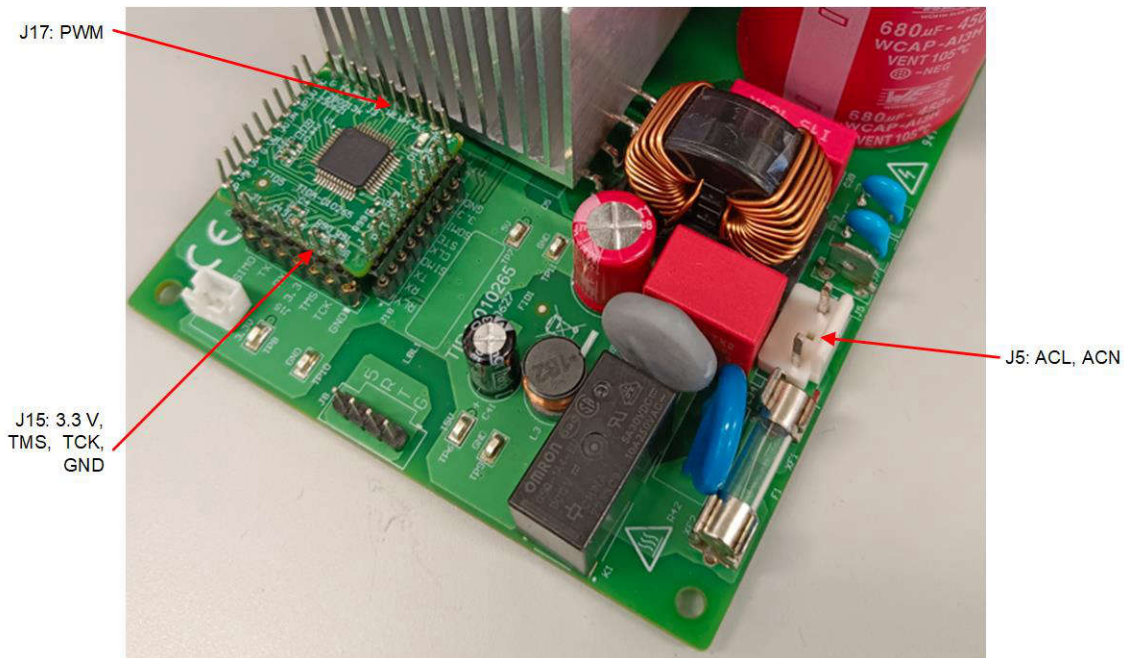


図 3-23. 外付け AC 電源または DC 電源を接続してハードウェアを検証する


3.3.4.1.2 プロジェクトのビルドとロード

プロジェクトをビルドしてロードするには、次の手順を行います。

1. プロジェクト名を右クリックし、[Properties] コマンドをクリックし、事前定義済みシンボルに移動して [GUI_SCI_EN] を [GUI_SCI_N] に変更し、[図 3-19](#) に示すように、GUI の SCI 機能を無効にします。

注

[セクション 3.2](#) で説明されているように、SCI 機能を再度有効にすると、SCI による GUI 制御が可能になります。

2. sys_settings.h ファイルを開き、DMC_BUILDLEVEL を DMC_LEVEL_1 に設定します。
3. 以前に別のビルド オプションがビルドされていた場合は、プロジェクト名を右クリックし、[Clean Project] をクリックして、[Build Project] をクリックします。ビルド ウィンドウでツールが実行されるのを確認します。プロジェクトが正常にビルドされます。
4. [Project Explorer] で、[図 3-20](#) に示すように、正しい目標構成ファイルが Active になっていることを確認します。
5. AC 電源または DC 電源をオンにして、J5 に AC 30V または DC 40V を印加し、コントローラとゲートドライバ用の +15V と 3.3V を生成します。[Debug] ボタン  をクリックするか、[Run] → [Debug] をクリックします。ビルドレベル 1 のコードをコンパイルして C2000 デバイスにロードすることができます。右上に表示されている CCS Debug アイコンは、ユーザーが [Debug Perspective] ビューにいることを示しています。プログラムは main() の最初で停止できます。


3.3.4.1.3 デバッグ環境設定ウィンドウ

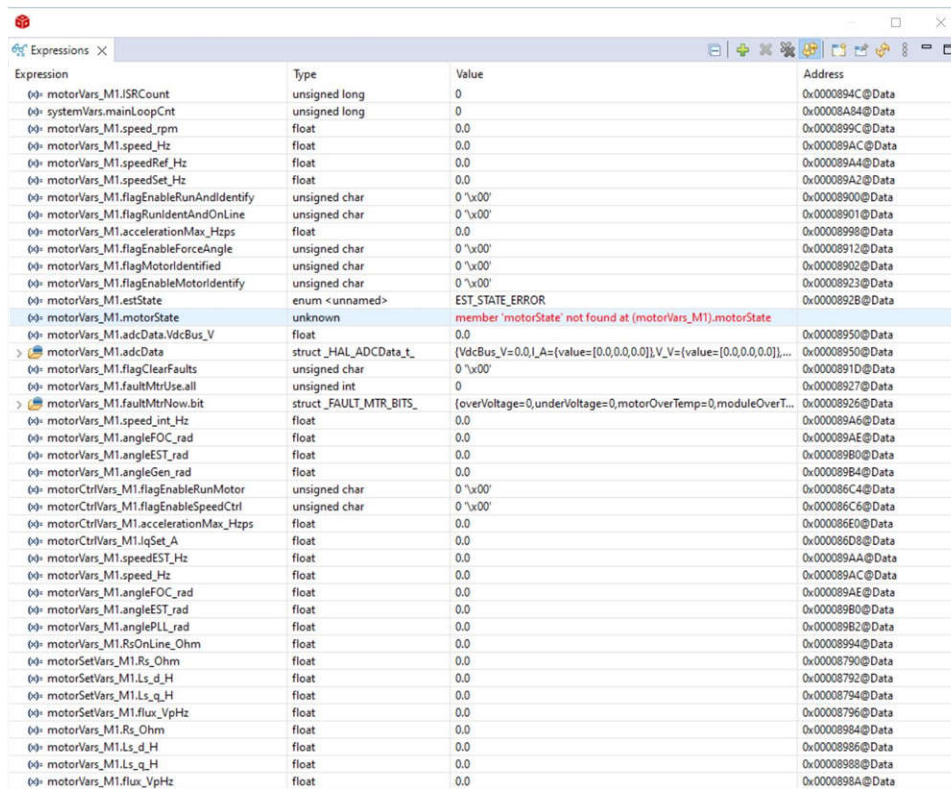
コードのデバッグ中にローカル変数やグローバル変数を注意深く観察することは、デバッグの標準的なやり方です。CCS では、このような変数の観察を行うために、メモリビューやウォッチ ビューなど、さまざまな方法を提供しています。さらに、CCS には時間 (および周波数) ドメインのプロットを作成する機能があります。この機能により、ユーザーはグラフ ツールを使用して波形を表示できます。

1. メニュー バーの [View] → [Expressions] をクリックして、[Expressions] ウォッチ ウィンドウを開きます。マウスを [Expressions] ウィンドウに移動して、プロジェクトで使用されている変数を表示します。[図 3-24](#) に示すように、[Expressions] ウィンドウに変数を追加します。このウィンドウでは、宣言時に変数に関連付けられた数値形式が使用されます。[図 3-24](#) に、[Expressions] ウィンドウの例を示します。式を右クリックして選択し、変数の数値形式を選択します。
2. 代わりに、[Expressions] ウィンドウ内で右クリックして [Import] をクリックすることで、変数のグループを [Expressions] ウィンドウにインポートすることもできます。
<install_location>\solutions\tida_010265_wminv\src_control\common\debug にあるプロジェクトのディレクトリをブラウズし、[BuildLevel1.txt] を選択して [OK] ボタンをクリックすると、[図 3-24](#) に示す変数がインポートされます。

注

メイン コードではこの時点で初期化されていない変数もあり、役に立たない値が含まれている可能性があります。

3. 構造体変数 motorVars_M1[] には、モーターの制御に関連するほとんどの変数へのリファレンスが含まれています。この変数を展開することで、必要に応じてすべての変数を表示したり、編集したりできます。
4. [Expressions] ウィンドウで [Continuous Refresh] ボタン  をクリックします。これにより、リアルタイム モードでウィンドウを実行できます。この [Expressions] ウィンドウで下矢印をクリックすることで、[Customize Continuous Refresh Interval] を選択して [Expressions] ウィンドウのリフレッシュ レートを編集できます。リフレッシュ間隔が速すぎると、パフォーマンスに影響することがあります。






Expression	Type	Value	Address
motorVars_M1.ISRCount	unsigned long	0	0x0000894C@Data
systemVars.mainLoopCnt	unsigned long	0	0x00008A84@Data
motorVars_M1.speed_rpm	float	0.0	0x0000899C@Data
motorVars_M1.speed_Hz	float	0.0	0x000089A0@Data
motorVars_M1.speedRef_Hz	float	0.0	0x000089A4@Data
motorVars_M1.speedSet_Hz	float	0.0	0x000089A8@Data
motorVars_M1.flagEnableRunAndIdentify	unsigned char	0 '\x00'	0x00008900@Data
motorVars_M1.flagRunIdentAndOnLine	unsigned char	0 '\x00'	0x00008901@Data
motorVars_M1.accelerationMax_Hzps	float	0.0	0x00008998@Data
motorVars_M1.flagEnableForceAngle	unsigned char	0 '\x00'	0x00008912@Data
motorVars_M1.flagMotorIdentified	unsigned char	0 '\x00'	0x00008902@Data
motorVars_M1.flagEnableMotorIdentify	unsigned char	0 '\x00'	0x00008923@Data
motorVars_M1.estState	enum <unnamed>	EST_STATE_ERROR	0x00008928@Data
motorVars_M1.motorState	unknown	member 'motorState' not found at (motorVars_M1).motorState	
motorVars_M1.adcData.VdcBus_V	float	0.0	0x00008950@Data
motorVars_M1.adcData	struct_HAL_ADCData_t	{VdcBus_V=0.0,La=[value=[0.0,0.0,0.0]],V_V=[value=[0.0,0.0,0.0]]...	0x00008950@Data
motorVars_M1.flagClearFaults	unsigned char	0 '\x00'	0x0000891D@Data
motorVars_M1.faultMtrUse.all	unsigned int	0	0x00008927@Data
motorVars_M1.faultMtrNow.bit	struct_FAULT_MTR_BITS_	{overVoltage=0,underVoltage=0,motorOverTemp=0,moduleOverT...	0x00008926@Data
motorVars_M1.speed_int_Hz	float	0.0	0x000089A6@Data
motorVars_M1.angleFOC_rad	float	0.0	0x000089AE@Data
motorVars_M1.angleEST_rad	float	0.0	0x000089B0@Data
motorVars_M1.angleGen_rad	float	0.0	0x000089B4@Data
motorCtrlVars_M1.flagEnableRunMotor	unsigned char	0 '\x00'	0x000086C4@Data
motorCtrlVars_M1.flagEnableSpeedCtrl	unsigned char	0 '\x00'	0x000086C6@Data
motorCtrlVars_M1.accelerationMax_Hzps	float	0.0	0x000086E0@Data
motorCtrlVars_M1.lqSet_A	float	0.0	0x000086D8@Data
motorVars_M1.speedEST_Hz	float	0.0	0x000089AA@Data
motorVars_M1.speed_Hz	float	0.0	0x000089AC@Data
motorVars_M1.angleFOC_rad	float	0.0	0x000089AE@Data
motorVars_M1.angleEST_rad	float	0.0	0x000089B0@Data
motorVars_M1.anglePLL_rad	float	0.0	0x000089B2@Data
motorVars_M1.RsOnLine_Ohm	float	0.0	0x00008994@Data
motorSetVars_M1.Rs_Ohm	float	0.0	0x00008790@Data
motorSetVars_M1.Ls_d_H	float	0.0	0x00008792@Data
motorSetVars_M1.Ls_q_H	float	0.0	0x00008794@Data
motorSetVars_M1.flux_VpHz	float	0.0	0x00008796@Data
motorVars_M1.Rs_Ohm	float	0.0	0x00008984@Data
motorVars_M1.Ls_d_H	float	0.0	0x00008986@Data
motorVars_M1.Ls_q_H	float	0.0	0x00008988@Data
motorVars_M1.flux_VpHz	float	0.0	0x0000898A@Data

図 3-24. ビルドレベル 1:リセット時の [Expressions] ウォッチ ウィンドウ


3.3.4.1.4 コードの実行

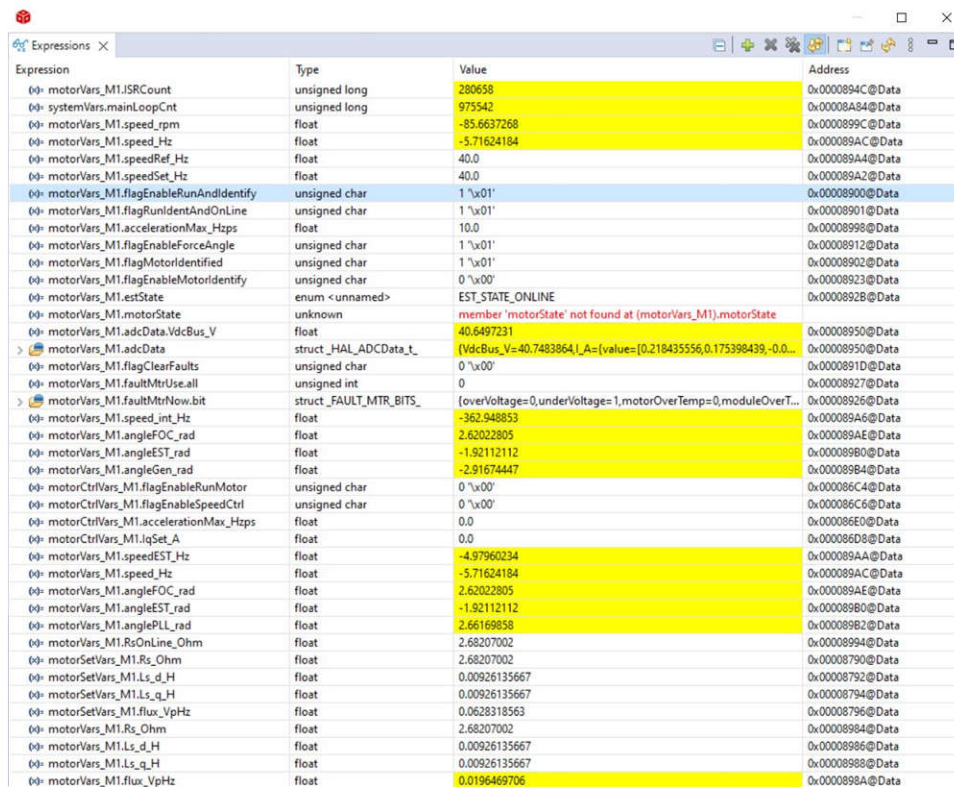
コードを実行するには、次の手順を行います。

- ボタン  をクリックしてプロジェクトを実行するか、[Debug] タブで [Run] → [Resume] をクリックします。
- [Expressions] ウォッチ ウィンドウで、systemVars.flagEnableSystem が自動的に 1 に設定された後、変数 motorVars_M1.flagEnableRunAndIdentify を 1 に設定します。
- これでプロジェクトが実行できるようになり、このプロジェクトを使用している間は、[図 3-25](#) に示すように、グラフと [Expressions] ウィンドウの値は連続的に更新されます。ウィンドウのサイズは、お好みに合わせて変更できます。
- ウォッチ ウィンドウでは、変数 motorVars_M1.flagRunIdentAndOnLine を自動的に 1 に設定できます。[ISRCount] は連続的に増加しています。
- モーターの較正オフセットをチェックします。[図 3-25](#) に示すように、モーター相電流センシングのオフセット値は、ADC のスケール電流の半分程度にすることができます。
- [図 3-23](#) に示すように、モーター駆動制御用の PWM 出力を J15 でオシロスコープを使用して測定します。このビルドレベルではすべての PWM デューティが 50% に設定され、PWM 出力の波形は [図 3-26](#) のようになります。motor_1 の PWM スイッチング周波数は 15kHz です。
- これでコントローラを停止し、デバッグ接続を終了できます。まずツールバーの [Halt] ボタン  をクリックするか、[Target] → [Halt] をクリックして、コントローラを完全に停止します。最後に、 をクリックするか、[Run] → [Reset] をクリックして、コントローラをリセットします。
- [図 3-27](#) に示すように、[Tools] → [On-Chip Flash] をクリックして次のビルドレベルのコントローラのコードを消去し、[On-Chip Flash] タブで [Erase Flash] をクリックします (すべてのフラッシュ バンクがチェックされていることを確認してください)。この操作により、フラッシュに保存されているすべてのプログラムコードが消去されます。(このステップはオプションです。ユーザーはこのステップを無視して、次のビルドレベルで新しいプログラムコードをロードできます。)

注

フラッシュの消去中は、[Cancel] をクリックしたり、ボードの電源をオフにしたり、エミュレータの接続を解除したりしないでください。

9. [Terminate Debug Session] ボタン  をクリックするか、[Run] → [Terminate] をクリックして、CCS デバッグ セッションを終了します。



Expression	Type	Value	Address
motorVars_M1.ISRCCount	unsigned long	280658	0x0000894C@Data
systemVars.mainLoopCnt	unsigned long	975542	0x00008A84@Data
motorVars_M1.speed_rpm	float	-85.6637268	0x0000895C@Data
motorVars_M1.speed_Hz	float	-5.71624184	0x000089AC@Data
motorVars_M1.speedRef_Hz	float	40.0	0x000089A4@Data
motorVars_M1.speedSet_Hz	float	40.0	0x000089A2@Data
motorVars_M1.flagEnableRunAndIdentify	unsigned char	1 '\u001'	0x00008900@Data
motorVars_M1.flagRunIdentAndOnLine	unsigned char	1 '\u001'	0x00008901@Data
motorVars_M1.accelerationMax_Hzps	float	10.0	0x00008998@Data
motorVars_M1.flagEnableForceAngle	unsigned char	1 '\u001'	0x00008912@Data
motorVars_M1.flagMotorIdentified	unsigned char	1 '\u001'	0x00008902@Data
motorVars_M1.flagEnableMotorIdentify	unsigned char	0 '\u000'	0x00008923@Data
motorVars_M1.estState	enum <unnamed>	EST_STATE_ONLINE	0x0000892B@Data
motorVars_M1.motorState	unknown	member 'motorState' not found at (motorVars_M1).motorState	
motorVars_M1.adcData.VdcBus_V	float	40.6497231	0x00008950@Data
motorVars_M1.adcData	struct _HAL_ADCData_t	{VdcBus_V=40.7483864, I_A=([value]=[0.218435556,0.175398439,-0.0...]	0x00008950@Data
motorVars_M1.flagClearFaults	unsigned char	0 '\u000'	0x0000891D@Data
motorVars_M1.faultMtrUse.all	unsigned int	0	0x00008927@Data
motorVars_M1.faultMtrNow.bit	struct _FAULT_MTR_BITS_	{overVoltage=0, underVoltage=1, motorOverTemp=0, moduleOverT...	0x00008926@Data
motorVars_M1.speed_int_Hz	float	-362.948853	0x000089A6@Data
motorVars_M1.angleFOC_rad	float	2.62022805	0x000089AE@Data
motorVars_M1.angleEST_rad	float	-1.92112112	0x000089B0@Data
motorVars_M1.angleGen_rad	float	-2.91674447	0x000089B4@Data
motorCtrlVars_M1.flagEnableRunMotor	unsigned char	0 '\u000'	0x000086C4@Data
motorCtrlVars_M1.flagEnableSpeedCtrl	unsigned char	0 '\u000'	0x000086C6@Data
motorCtrlVars_M1.accelerationMax_Hzps	float	0.0	0x000086E0@Data
motorCtrlVars_M1.iqSet_A	float	0.0	0x000086D8@Data
motorVars_M1.speedEST_Hz	float	-4.97960234	0x000089AA@Data
motorVars_M1.speed_Hz	float	-5.71624184	0x000089AC@Data
motorVars_M1.angleFOC_rad	float	2.62022805	0x000089AE@Data
motorVars_M1.angleEST_rad	float	-1.92112112	0x000089B0@Data
motorVars_M1.anglePLL_rad	float	2.66169858	0x000089B2@Data
motorVars_M1.RsOnLine_Ohm	float	2.68207002	0x00008994@Data
motorSetVars_M1.Rs_Ohm	float	2.68207002	0x00008790@Data
motorSetVars_M1.Ls_d_H	float	0.00926135667	0x00008792@Data
motorSetVars_M1.Ls_q_H	float	0.00926135667	0x00008794@Data
motorSetVars_M1.flux_VpHz	float	0.0628318563	0x00008796@Data
motorVars_M1.Rs_Ohm	float	2.68207002	0x00008984@Data
motorVars_M1.Ls_d_H	float	0.00926135667	0x00008986@Data
motorVars_M1.Ls_q_H	float	0.00926135667	0x00008988@Data
motorVars_M1.flux_VpHz	float	0.0196469706	0x0000898A@Data

図 3-25. ビルドレベル 1: 実行時の [Expressions] ウィンドウ

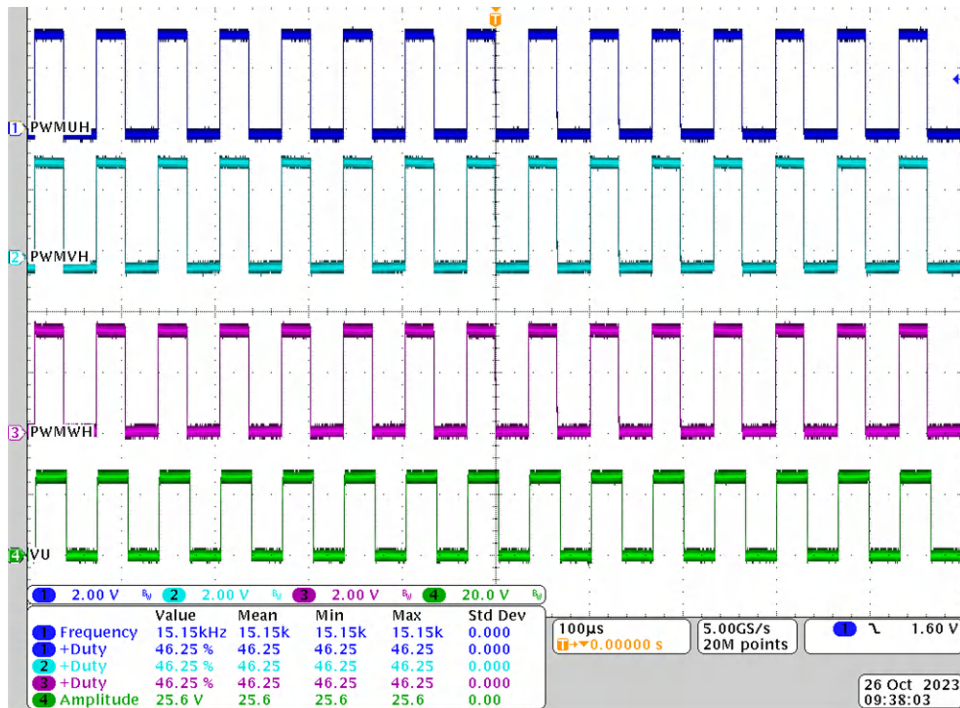


図 3-26. ビルドレベル 1: MCU PWM 出力と IPM 出力

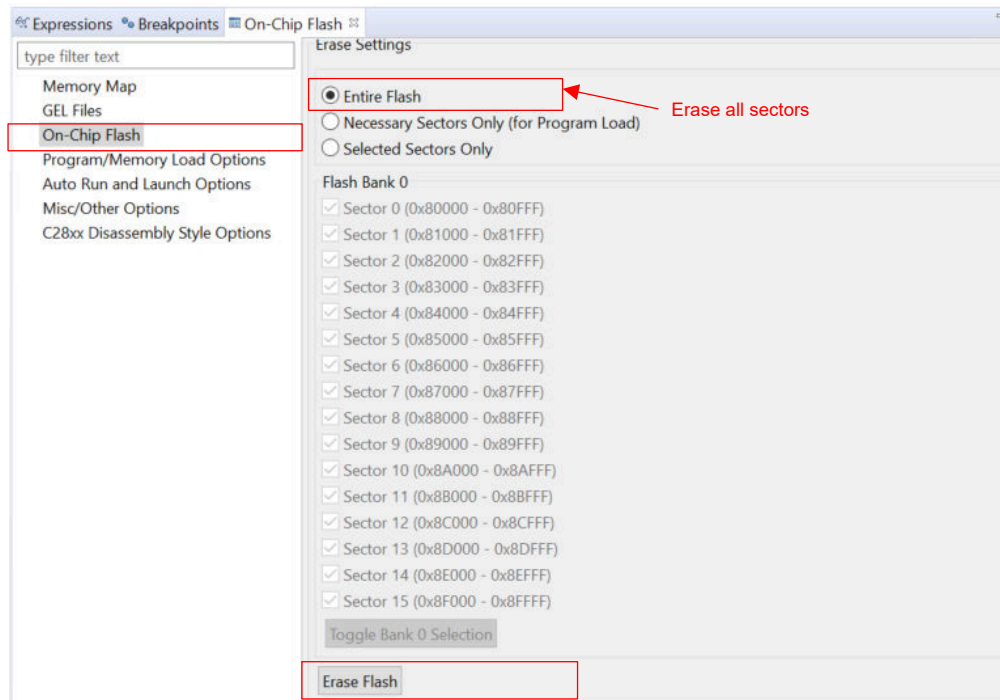


図 3-27. ビルドレベル 1: 次のビルドレベルのためにフラッシュのプログラム コードを消去する

3.3.4.2 ビルドレベル 2: ADC 帰還を使用した開ループチェック

このビルドレベルの学習目標:

- 電流と電圧のセンシング回路と IPM 回路の検証のために、モーター駆動用の簡単なスカラー v/f 制御を実装する
- モーター制御用の InstaSPIN-FOC FAST モジュールまたは eSMO モジュールをテストする

このシステムは開ループ制御で動作しているため、ADC での測定値はこのビルドレベルでは計測目的でのみ使用されます。高電圧 DC 電源はインバータに実装されており、MCU コントローラと IPM 用のバイアス電源は補助電源モジュールから供給されます。このビルドレベルのソフトウェア フローを [図 3-28](#) に示します。

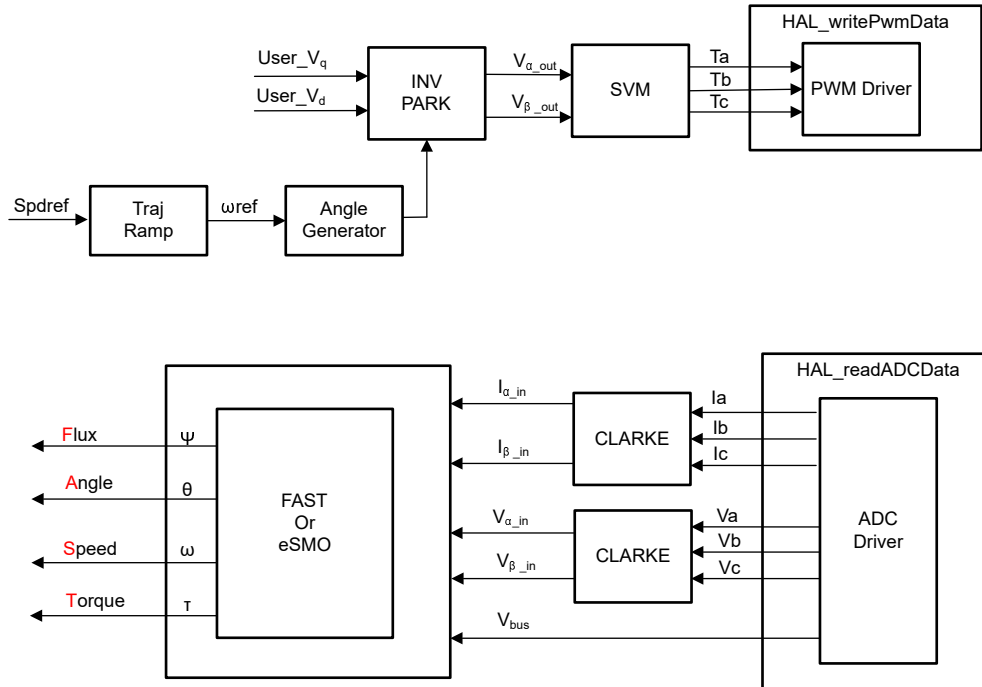


図 3-28. 制御ソフトウェアのブロック図:ビルドレベル 2 – 開ループ制御

3.3.4.2.1 CCS を起動し、プロジェクトを開く

CCS を起動してプロジェクトを開くには、次の手順を行います。

1. [図 3-23](#) に示すように、最大 750W のユニバーサル入力可能な絶縁型 AC 電源をリファレンスボードの入力端子 (コネクタ J5) に接続します。電源の電流制限を 2A に設定します。このとき、電源はオンにしないでください。
2. モーターを J10 に接続します。
3. [セクション 3.3.4.1.1](#) のステップ 2 と 3 に従って、プロジェクトを開きます。

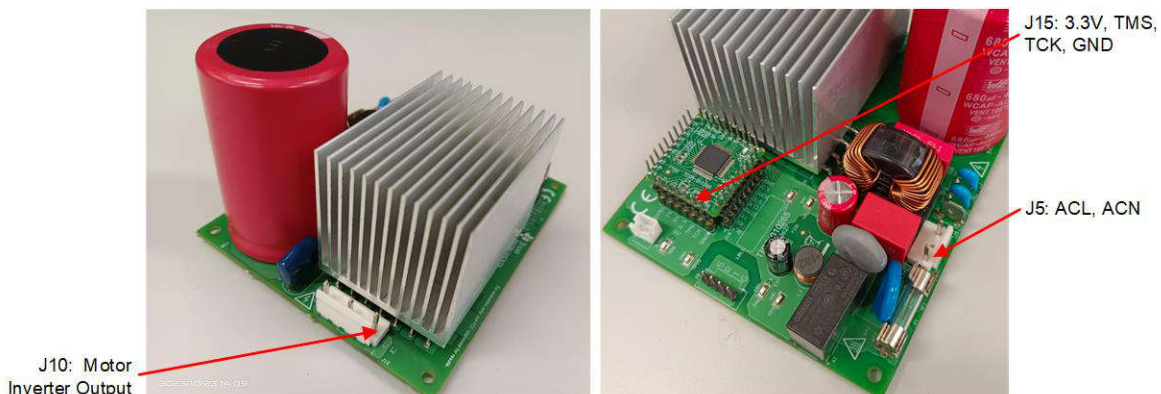


図 3-29. 外付け AC 電源または DC 電源を接続してハードウェアを検証する

3.3.4.2.2 プロジェクトのビルドとロード

プロジェクトをビルドしてロードするには、次の手順を行います。


1. DMC_BUILDLEVEL to DMC_LEVEL_2 を設定します。
2. セクション 3.3.4.1.2 のステップ 2~4 に従って、プロジェクトをビルドし、コードをコントローラにロードします。

3.3.4.2.3 デバッグ環境設定ウィンドウ




セクション 3.3.4.1.3 のステップ 1~4 に従って、[BuildLevel2.txt] を選択して、変数を [Expressions] ウィンドウにインポートします。図 3-30 に示すように、[Expressions] ウィンドウが表示されます。

3.3.4.2.4 コードの実行

コードを実行するには、次の手順を行います。

1. AC 電源出力を 0V に設定し、AC 電源をオンにして、出力電圧を AC 0V から 100V までゆっくりと上げます。
2.  ボタンをクリックしてプロジェクトを実行するか、[Debug] タブで [Run] → [Resume] をクリックします。モーターフォルトフラグ motorVars_M1.faultMtrUse.all は 0 でなければなりません、そうでない場合は、セクション 3.3.4.1 で説明されているように、電流および電圧のセンシング回路をチェックしてください。
3. モーター用インバータの電流および電圧のセンシング回路を検証するには、図 3-30 に示すように、[Expressions] ウィンドウで変数 motorVars_M1.flagEnableRunAndIdentify を 1 に設定します。motor_1 は v/f 開ループで動作する必要があります。モーターが滑らかに回転しない場合、モーターの仕様に合わせて user_mtr1.h の v/f プロファイルパラメータを以下のように調整してください。

```
#define USER_MOTOR1_FREQ_LOW_HZ      (10.0f)      // Hz
#define USER_MOTOR1_FREQ_HIGH_HZ     (200.0f)     // Hz
#define USER_MOTOR1_VOLT_MIN_V       (10.0f)      // volt
#define USER_MOTOR1_VOLT_MAX_V       (200.0f)     // volt
```

4. これでモーターは、変数 motorVars_M1.speedRef_Hz の設定速度で回転するようになります。[Expressions] ウィンドウで motorVars_M1.speed_Hz の値をチェックしてください。図 3-30 に示すように、非常に近い値でなければなりません。
5. 図 3-31 に示すように、オシロスコープの電圧プローブと電流プローブを接続して、モーターの相電圧と相電流を注意深く観察します。
6. 変数 motorVars_M1.overCurrent_A の値を小さくすることで、過電流フォルト保護を検証します。過電流保護は CMPSS モジュールによって実装されています。motorVars_M1.overCurrent_A が実際の電流より小さい値に設定され、PWM 出力が無効化され、motorVars_M1.flagEnableRunAndIdentify が 0 にクリアされ、motorVars_M1.faultMtrUse.all が 0x10 に設定されると、過電流フォルトがトリガされます。
7. これでコントローラを停止し、デバッグ接続を終了できます。まずツールバーの [Halt] ボタン  をクリックするか、[Target] → [Halt] をクリックして、コントローラを完全に停止します。最後に、 をクリックするか、[Run] → [Reset] をクリックして、コントローラをリセットします。
8. [Terminate Debug Session] ボタン  をクリックするか、[Run] → [Terminate] をクリックして、CCS デバッグセッションを終了します。

Expression	Type	Value	Address
motorVars_M1.ISRCCount	unsigned long	544363	0x0000894C@Data
systemVars.mainLoopCnt	unsigned long	2059177	0x00008A84@Data
motorVars_M1.speed_rpm	float	1176.95312	0x0000899C@Data
motorVars_M1.speed_Hz	float	80.0597916	0x000089AC@Data
motorVars_M1.speedRef_Hz	float	80.0	0x000089A4@Data
motorVars_M1.speedSet_Hz	float	80.0	0x000089A2@Data
motorVars_M1.flagEnableRunAndIdentify	unsigned char	1 '\x01'	0x00008900@Data
motorVars_M1.flagRunIdentAndOnLine	unsigned char	1 '\x01'	0x00008901@Data
motorVars_M1.accelerationMax_Hzps	float	20.0	0x00008998@Data
motorVars_M1.flagEnableForceAngle	unsigned char	1 '\x01'	0x00008912@Data
motorVars_M1.flagMotorIdentified	unsigned char	1 '\x01'	0x00008902@Data
motorVars_M1.flagEnableMotorIdentify	unsigned char	0 '\x00'	0x00008923@Data
motorVars_M1.estState	enum <unnamed>	EST_STATE_ONLINE	0x0000892B@Data
motorVars_M1.motorState	unknown	member 'motorState' not found at (motorVars_M1).motorState	
motorVars_M1.adcData.VdcBus_V	float	141.188721	0x00008950@Data
motorVars_M1.adcData.Vstruct_HAL_ADCData_t	struct	{VdcBus_V=136.650162, I_A=(value=[2.74908686,3.78245902,4.2660...	0x00008950@Data
motorVars_M1.flagClearFaults	unsigned char	0 '\x00'	0x0000891D@Data
motorVars_M1.faultMtrUseAll	unsigned int	0	0x00008927@Data
motorVars_M1.faultMtrNow.bit	struct_FAULT_MTR_BITS_	{overVoltage=0,underVoltage=0,motorOverTemp=0,moduleOverT...	0x00008926@Data
motorVars_M1.speed_int_Hz	float	80.0	0x000089A6@Data
motorVars_M1.angleFOC_rad	float	-1.86383724	0x000089AE@Data
motorVars_M1.angleEST_rad	float	0.0245006047	0x000089B0@Data
motorVars_M1.angleGen_rad	float	1.99211061	0x000089B4@Data
motorCtrlVars_M1.flagEnableRunMotor	unsigned char	0 '\x00'	0x000086C4@Data
motorCtrlVars_M1.flagEnableSpeedCtrl	unsigned char	0 '\x00'	0x000086C6@Data
motorCtrlVars_M1.accelerationMax_Hzps	float	0.0	0x000086ED@Data
motorCtrlVars_M1.lqSet_A	float	0.0	0x000086D8@Data
motorVars_M1.speedEST_Hz	float	78.2935638	0x000089AA@Data
motorVars_M1.speed_Hz	float	80.0597916	0x000089AC@Data
motorVars_M1.angleFOC_rad	float	-1.86383724	0x000089AE@Data
motorVars_M1.angleEST_rad	float	0.0245006047	0x000089B0@Data
motorVars_M1.anglePLL_rad	float	0.653951049	0x000089B2@Data
motorVars_M1.RsOnLine_Ohm	float	2.68207002	0x00008994@Data
motorSetVars_M1.Rs_Ohm	float	2.68207002	0x00008790@Data
motorSetVars_M1.Ls_d_H	float	0.00926135667	0x00008792@Data
motorSetVars_M1.Ls_q_H	float	0.00926135667	0x00008794@Data
motorSetVars_M1.flux_VpHz	float	0.0628318563	0x00008796@Data
motorVars_M1.Rs_Ohm	float	2.68207002	0x00008984@Data
motorVars_M1.Ls_d_H	float	0.00926135667	0x00008986@Data

図 3-30. ビルドレベル 2: 実行時の [Expressions] ウィンドウ

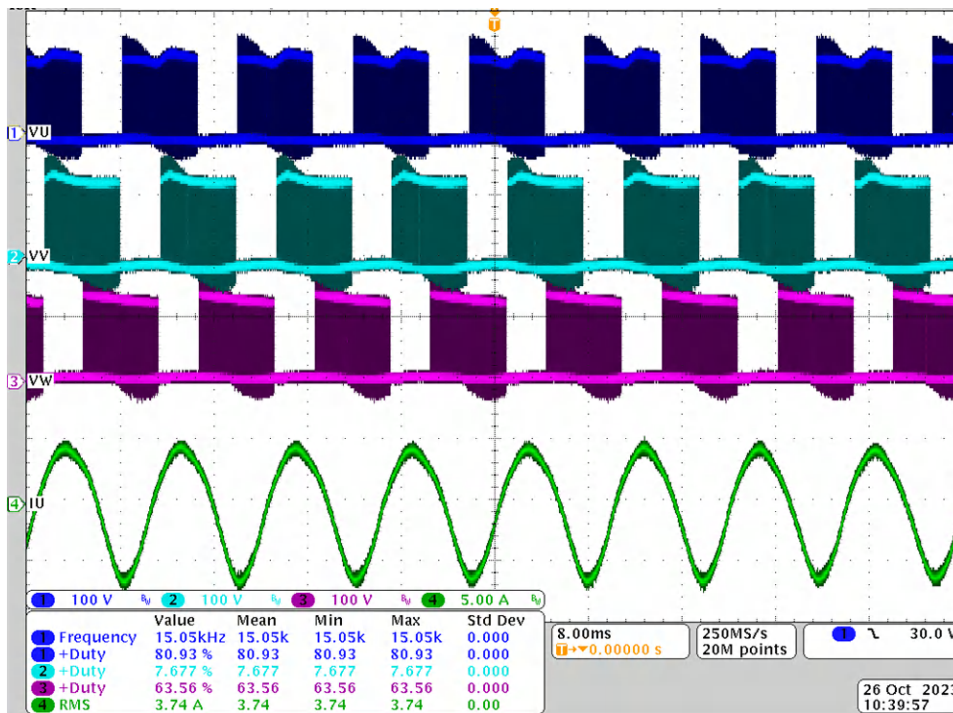


図 3-31. ビルドレベル 2: モーターの相電圧と相電流

3.3.4.3 ビルドレベル 3: 閉電流ループ チェック

このビルドレベルの学習目標:

- モーター動作の閉電流ループを評価する

このビルドレベルでは、モーターは *if* 制御を使用して制御され、回転子角度はランプ ジェネレータ モジュールから生成されます。このビルドレベルのソフトウェア フローを [図 3-32](#) に示します。

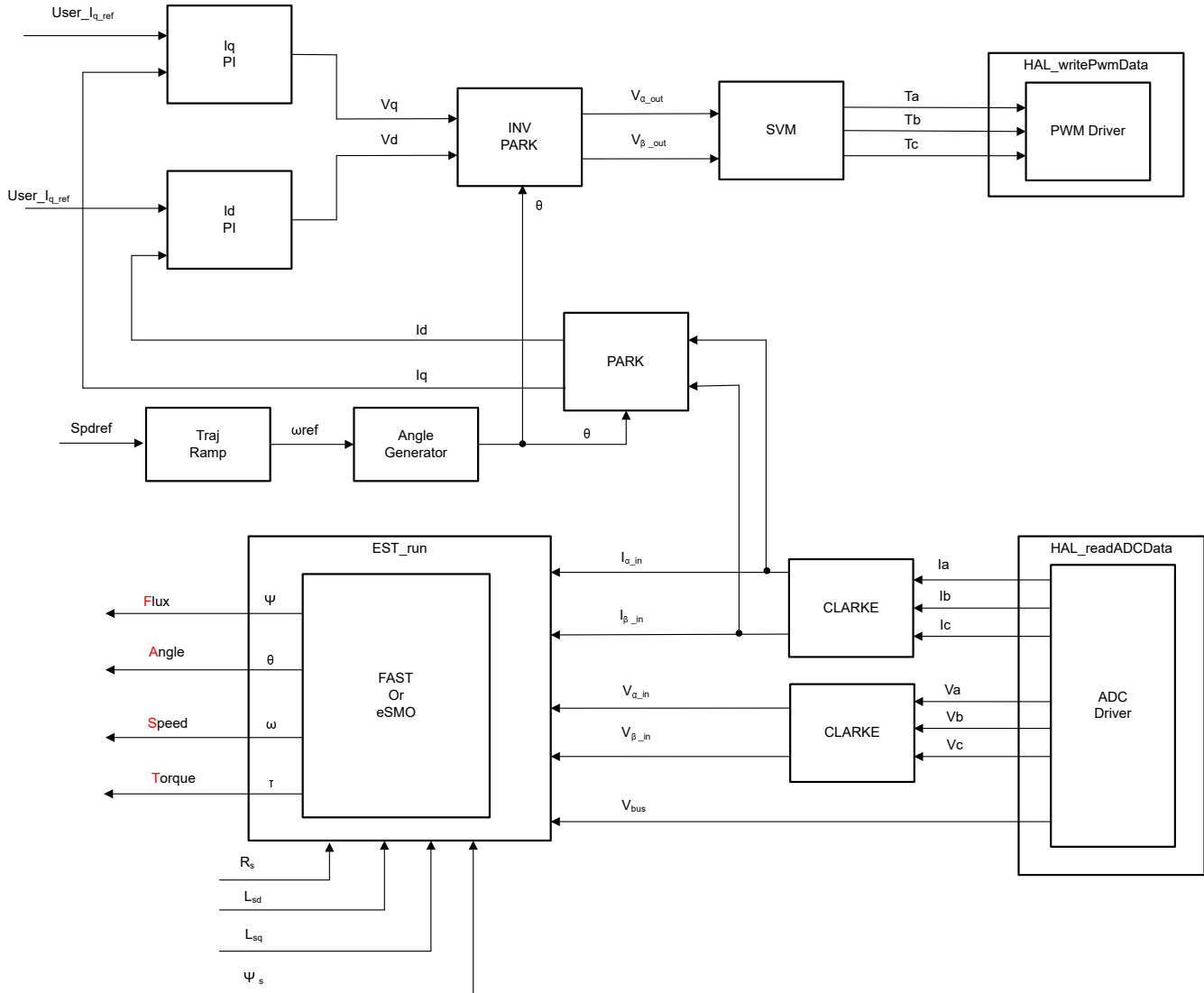


図 3-32. 制御ソフトウェアのブロック図: ビルドレベル 3 - 電流の閉ループ制御

3.3.4.3.1 CCS を起動し、プロジェクトを開く

CCS を起動してプロジェクトを開くには、次の手順を行います。

- [図 3-29](#) に示すように、最大 750W のユニバーサル AC 入力を供給できるプログラマブル絶縁型 AC 電源をリファレンスボードの入力端子 (コネクタ J5) に接続します。電源の電流制限を 2A に、出力周波数を 50/60Hz に設定します。このとき、電源はオンにしないでください。
- モーター (コンプレッサ) を J10 に接続します。
- [セクション 3.3.4.1.1](#) のステップ 2 と 3 に従って、プロジェクトを開きます。

3.3.4.3.2 プロジェクトのビルドとロード

プロジェクトをビルドしてロードするには、次の手順を行います。





1. `sys_settings.h` ファイルを開き、`DMC_BUILDLEVEL` を `DMC_LEVEL_3` に設定します。
2. [セクション 3.3.4.1.2](#) のステップ 2~4 に従って、プロジェクトをビルドし、コードをコントローラにロードします。

3.3.4.3.3 デバッグ環境設定ウィンドウ

[セクション 3.3.4.1.3](#) のステップ 1~4 に従って、`BuildLevel3.txt` を選択して変数を [Expressions] ウィンドウにインポートします。[図 3-30](#) に示すように、[Expressions] ウィンドウが表示されます。

3.3.4.3.4 コードの実行

コードを実行するには、次の手順を行います。

1. AC 電源出力を 50/60Hz で 0V に設定し、AC 電源をオンにして、入力電圧を AC 0V から 220V までゆっくりと上げます。
2.  ボタンをクリックしてプロジェクトを実行するか、[Debug] タブで [Run] → [Resume] をクリックします。一定の時間が経過したら、`systemVars.flagEnableSystem` を 1 に設定します。これによって、オフセット較正が完了し、導通時には電力リレーがオンになります。モーター フォルト フラグ `motorVars_M1.faultMtrUse.all` は 0 でなければなりません。そうでない場合は、[セクション 3.3.4.1](#) で説明されているように、電流および電圧のセンシング回路をチェックしてください。
3. モーターの電流閉ループ制御を検証するには、[図 3-33](#) に示すように、[Expressions] ウィンドウで変数 `motorVars_M1.flagEnableRunAndIdentify` を 1 に設定します。モーターは、変数 `motorVars_M1.speedRef_Hz` の設定速度で角度ジェネレータからの角度を使用して閉ループ制御で動作する必要があります。[Expressions] ウィンドウで `motorVarsM1.speed_Hz` の値をチェックしてください。両方の変数値は非常に近い値でなければなりません。
4. モーター電流 `Iq` は、`motorVars_M1.Idq_Set_A.value[1]` で設定および変更できます。
5. [図 3-34](#) に示すように、オシロスコープのプロープを IPM 出力に接続して、モーターの相電圧と相電流を注意深く観察します。[Expressions] ウィンドウで `Idq_set_A[0].value[1]` を変更します。モーター相電流は適宜増加させる必要があります。
6. 電流閉ループでモーターが動作せず、過電流フォルトが発生しているように見える場合は、`adcData[0].current_sf` の符号と `userParams[0].current_sf` の値がハードウェア ボードに応じて正しく設定されているかどうかをチェックします。
7. これでコントローラを停止してから `motorVars_M1.flagEnableRunAndIdentify` を 0 に設定し、デバッグ接続を終了できます。まずツールバーの [Halt] ボタン  をクリックするか、[Target] → [Halt] をクリックして、コントローラを完全に停止します。最後に、 をクリックするか、[Run] → [Reset] をクリックして、コントローラをリセットします。
8. [Terminate Debug Session] ボタン  をクリックするか、[Run] → [Terminate] をクリックして、CCS デバッグ セッションを終了します。

Expression	Type	Value	Address
motorVars_M1.ISRCCount	unsigned long	1284134	0x0000894C@Data
systemVars.mainLoopCnt	unsigned long	5307166	0x00008A84@Data
motorVars_M1.speed_rpm	float	592.226624	0x0000899C@Data
motorVars_M1.speed_Hz	float	40.6853447	0x000089AC@Data
motorVars_M1.speedRef_Hz	float	40.0	0x000089A4@Data
motorVars_M1.speedSet_Hz	float	40.0	0x000089A2@Data
motorVars_M1.flagEnableRunAndIdentify	unsigned char	1 '\x01'	0x00008900@Data
motorVars_M1.flagRunIdentAndOnLine	unsigned char	1 '\x01'	0x00008901@Data
motorVars_M1.accelerationMax_Hzps	float	20.0	0x00008998@Data
motorVars_M1.flagEnableForceAngle	unsigned char	1 '\x01'	0x00008912@Data
motorVars_M1.flagMotorIdentified	unsigned char	1 '\x01'	0x00008902@Data
motorVars_M1.flagEnableMotorIdentify	unsigned char	0 '\x00'	0x00008923@Data
motorVars_M1.estState	enum <unnamed>	EST_STATE_ONLINE	0x00008928@Data
motorVars_M1.controlStatus	enum <unnamed>	MOTOR_CTRL_RUN	0x0000892A@Data
motorVars_M1.adcData.VdcBus_V	float	308.918152	0x00008950@Data
motorVars_M1.adcData	struct _HAL_ADCData_t	{VdcBus_V=308.7208251_A=[value=[-1.88685691,-0.674133778,0.91...	0x00008950@Data
motorVars_M1.flagClearFaults	unsigned char	0 '\x00'	0x0000891D@Data
motorVars_M1.faultMtrUse.all	unsigned int	0	0x00008927@Data
motorVars_M1.faultMtrNrn.bit	struct _FAULT_MTR_BITS_	{overVoltage=0,underVoltage=0,motorOverTemp=0,moduleOverT...	0x00008926@Data
motorVars_M1.speed_int_Hz	float	40.0	0x000089A6@Data
motorVars_M1.angleFOC_rad	float	2.30622911	0x000089AE@Data
motorVars_M1.angleEST_rad	float	-1.95582139	0x000089B0@Data
motorVars_M1.angleGen_rad	float	-2.92251492	0x000089B4@Data
motorCtrlVars_M1.flagEnableRunMotor	unsigned char	0 '\x00'	0x000086C4@Data
motorCtrlVars_M1.flagEnableSpeedCtrl	unsigned char	0 '\x00'	0x000086C6@Data
motorCtrlVars_M1.accelerationMax_Hzps	float	0.0	0x000086ED@Data
motorCtrlVars_M1.iqSet_A	float	0.0	0x000086D8@Data
motorVars_M1.speedEST_Hz	float	40.7923813	0x000089AA@Data
motorVars_M1.speed_Hz	float	40.6853447	0x000089AC@Data
motorVars_M1.angleFOC_rad	float	2.30622911	0x000089AE@Data
motorVars_M1.angleEST_rad	float	-1.95582139	0x000089B0@Data
motorVars_M1.anglePLL_rad	float	1.01841605	0x000089B2@Data
motorVars_M1.RsOnLine_Ohm	float	2.68207002	0x00008994@Data
motorSetVars_M1.Rs_Ohm	float	2.68207002	0x00008790@Data
motorSetVars_M1.Ls_d_H	float	0.00926135667	0x00008792@Data
motorSetVars_M1.Ls_q_H	float	0.00926135667	0x00008794@Data
motorSetVars_M1.flux_VpHz	float	0.0628318563	0x00008796@Data
motorVars_M1.Rs_Ohm	float	2.68207002	0x00008984@Data
motorVars_M1.Ls_d_H	float	0.00926135667	0x00008986@Data
motorVars_M1.Ls_q_H	float	0.00926135667	0x00008988@Data
motorVars_M1.flux_VpHz	float	0.378431171	0x0000898A@Data
motorVars_M1.flux_Wb	float	0.0603588633	0x0000898C@Data
angleGen_M1	struct _ANGLE_GEN_Obj_	{freq_Hz=40.0,angleDeltaFactor=0.000418879034,angleDelta_rad=...	0x0000861C@Data
motorVars_M1.Idq_set_A.value[0]	float	0.0	0x00008A2A@Data
motorVars_M1.Idq_set_A.value[1]	float	2.0	0x00008A2C@Data

図 3-33. ビルドレベル 3:実行時の [Expressions] ウィンドウ

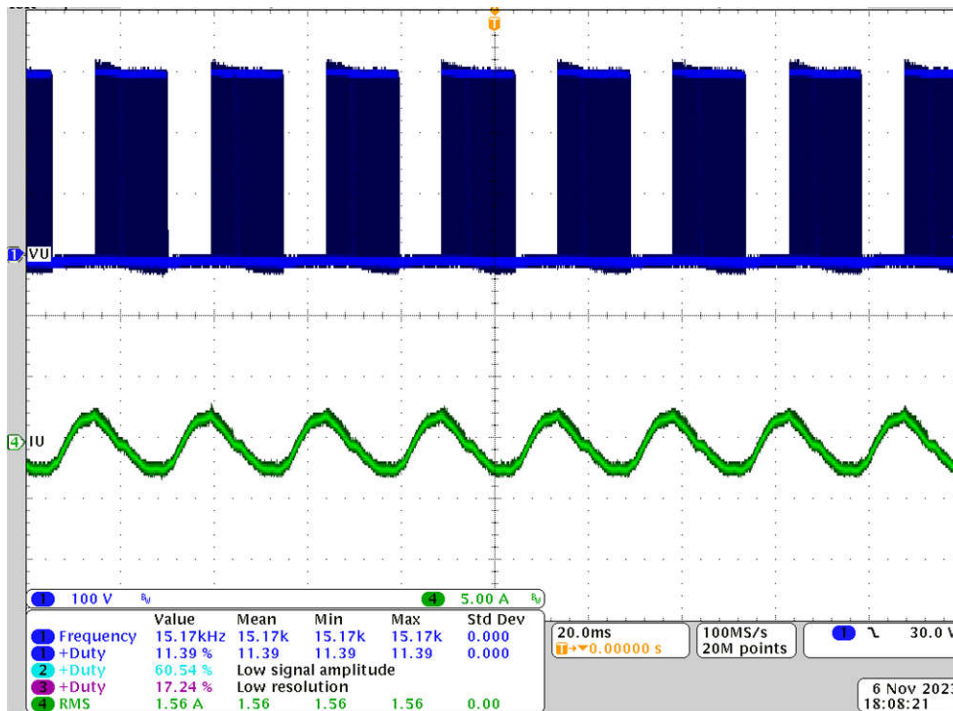
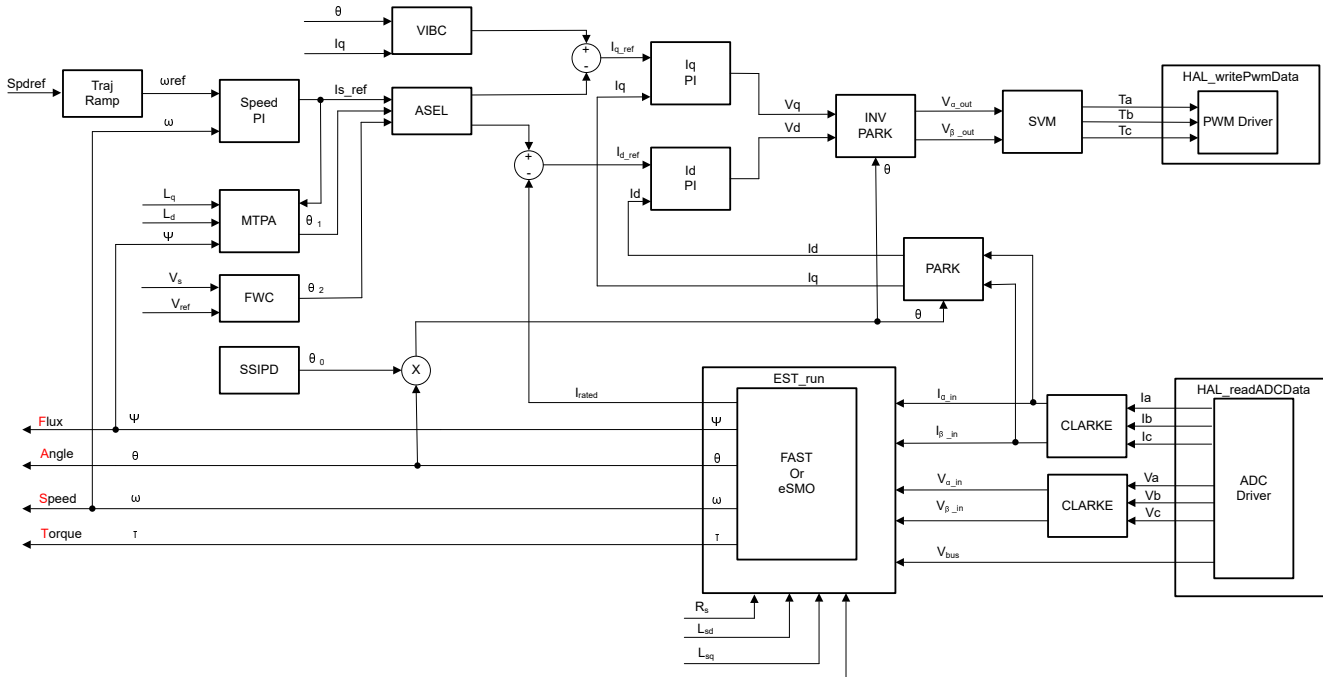


図 3-34. ビルドレベル 3:2A I_q 設定時のモーター電流

3.3.4.4 ビルドレベル 4: 完全なモーター駆動制御

このビルドレベルの学習目標:

- 完全なモーター駆動を評価する
- 追加機能およびモーターの弱め界磁制御を評価する
- 完成したシステムを評価する

このビルドレベルでは、回転子角度が FAST または eSMO エスティメータ モジュールから得られるように、モーターの外側の速度ループは内側の電流ループで閉じられます。このビルドレベルのソフトウェア フローを  3-35 に示します。

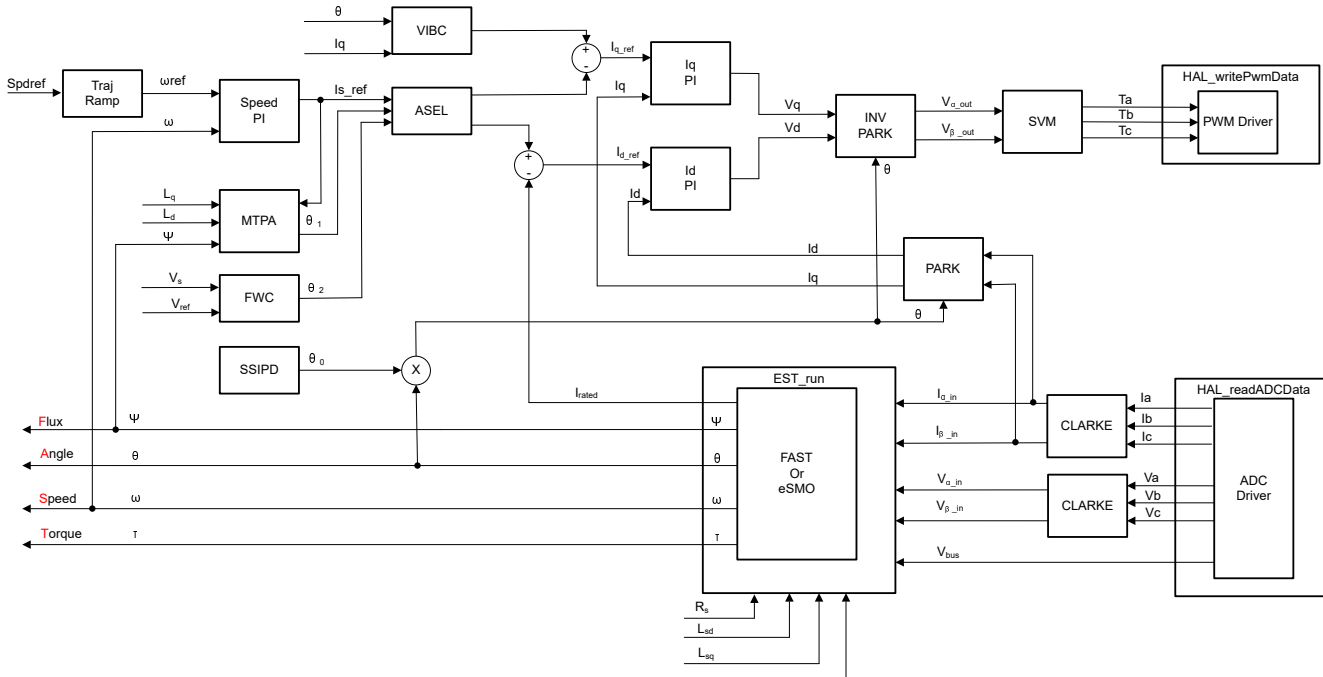
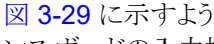


図 3-35. 制御ソフトウェアのブロック図: ビルドレベル 4 – 速度と電流の閉ループ制御

3.3.4.4.1 CCS を起動し、プロジェクトを開く

CCS を起動してプロジェクトを開くには、次の手順を行います。

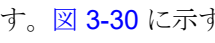
1.  3-29 に示すように、最大 750W のユニバーサル AC 入力を提供できるプログラマブル絶縁型 AC 電源をリファレンスボードの入力端子 (コネクタ J5) に接続します。AC 電源の電流制限を 8A に設定します。このとき、電源はオンにしないでください。
2. モーター (コンプレッサ) を J10 に接続します。
3. セクション 3.3.4.1.1 のステップ 2 と 3 に従って、プロジェクトを開きます。

3.3.4.4.2 プロジェクトのビルドとロード

プロジェクトをビルドしてロードするには、次の手順を行います。

1. sys_settings.h ファイルを開き、DMC_BUILDLEVEL を DMC_LEVEL_4 に設定します。
2. セクション 3.3.4.1.2 のステップ 2~4 に従って、プロジェクトをビルドし、コードをコントローラにロードします。

3.3.4.4.3 デバッグ環境設定ウィンドウ

セクション 3.3.4.1.3 のステップ 1~4 に従って、BuildLevel4.txt を選択して変数を [Expressions] ウィンドウにインポートします。  3-30 に示すように、[Expressions] ウィンドウが表示されます。

3.3.4.4.4 コードの実行

コードを実行するには、次の手順を行います。

- AC 電源出力を 50/60Hz で 0V に設定し、AC 電源をオンにして、入力電圧を AC 0V から 220V までゆっくりと上げます。
- 次のサンプルコードに示すように、必要なモーター パラメータはヘッダー ファイル (`user_mtr1.h`) に記録する必要があります。モーター パラメータが不明な場合、リファレンス デザインに FAST エスティメータが実装されていれば、モーター 識別を使用してモーター パラメータを取得できます。





```
#define USER_MOTOR1_Rs_Ohm           (2.68207002f)
#define USER_MOTOR1_Ls_d_H           (0.00926135667f)
#define USER_MOTOR1_Ls_q_H           (0.00926135667f)
#define USER_MOTOR1_RATED_FLUX_VpHz (0.381890297f)
```

- `userParams_M1.flag_bypassMotorId` の値を `false` に変更し、次のモーターのサンプルコードのように、モーター 識別を有効にします。

```
// true->enable identification, false->disable identification
userParams[MTR_1].flag_bypassMotorId = false;
```

- モーターの仕様に合わせて、`user_mtr1.h` に正しい識別変数の値を設定します。

```
#define USER_MOTOR1_RES_EST_CURRENT_A (1.0f) // A - 10~30% of rated current of the
motor
#define USER_MOTOR1_IND_EST_CURRENT_A (-1.0f) // A - 10~30% of rated current of the
motor, just enough to enable rotation
#define USER_MOTOR1_MAX_CURRENT_A (6.5f) // A - 30~150% of rated current of the
motor
#define USER_MOTOR1_FLUX_EXC_FREQ_HZ (40.0f) // Hz - 10~30% of rated frequency of
the motor
```

- プロジェクトをリビルドしてコードをコントローラにロードします。  ボタンをクリックしてプロジェクトを実行するか、[Debug] タブで [Run] → [Resume] をクリックします。一定の時間が経過したら、`systemVars.flagEnableSystem` を 1 に設定する必要があります。これによって、オフセット較正が完了し、導通時には電力リレーがオンになります。モーター フォルト フラグ `motorVars_M1.faultMtrUse.all` は 0 でなければなりません、そうでない場合は、[セクション 3.3.4.1](#) で説明されているように、電流および電圧のセンシング回路をチェックしてください。
- [図 3-36](#) に示すように、[Expressions] ウィンドウで変数 `motorVars_M1.flagEnableRunAndIdentify` を 1 に設定すると、モーター 識別を実行されます。このプロセスには約 150 秒かかります。`motorVars_M1.flagEnableRunAndIdentify` が 0 になると、モーター パラメータが識別されます。新しく定義したモーター パラメータでウォッチ ウィンドウの値を、`user_mtr1.h` に次のように記録します。
 - `USER_MOTOR1_Rs` = `motorVars_M1.Rs_オーム` の値
 - `USER_MOTOR1_Ls_d` = `motorVars_M1.Ls_d_H` の値
 - `USER_MOTOR1_Ls_q` = `motorVars_M1.Ls_q_H` の値
 - `USER_MOTOR_RATED_FLUX` = `motorVars_M1.flux_VpHz` の値
- モーター パラメータの識別が正常に完了したら、`userParams_M1.flag_bypassMotorId` を両方 `true` に設定し、プロジェクトをリビルドして、コードをコントローラにロードします。
 - 変数 `motorVars_M1.flagEnableRunAndIdentify` を再度 1 に設定し、モーターの動作を開始します。
 - 変数 `motorVars_M1.speedRef_Hz` を異なる値に設定し、モーター シャフトの速度がどのように変化するかを注意深く観察します。
 - 加速度を変更するには、変数 `motorVars_M1.accelerationMax_Hzps` と `motorVars_M1.accelerationMax_Hzps` に異なる加速度値を入力します。
- これでコントローラを停止してから `motorVars_M1.flagEnableRunAndIdentify` を 0 に設定し、デバッグ接続を終了できます。まずツールバーの [Halt] ボタン  をクリックするか、[Target] → [Halt] をクリックして、コントローラを完全に停止します。最後に、 をクリックするか、[Run] → [Reset] をクリックして、コントローラをリセットします。
- [Terminate Debug Session] ボタン  をクリックするか、[Run] → [Terminate] をクリックして、CCS デバッグ セッションを終了します。

Expression	Type	Value	Address
motorVars_M1.ISRCCount	unsigned long	367093	0x0000894C@Data
systemVars.mainLoopCnt	unsigned long	1277708	0x00008A84@Data
motorVars_M1.speed_rpm	float	1501.23352	0x0000899C@Data
motorVars_M1.speed_Hz	float	100.178963	0x000089AC@Data
motorVars_M1.speedRef_Hz	float	100.0	0x000089A4@Data
motorVars_M1.speedSet_Hz	float	100.0	0x000089A2@Data
motorVars_M1.flagEnableRunAndIdentify	unsigned char	1 '\x01'	0x00008900@Data
motorVars_M1.flagRunIdentAndOnLine	unsigned char	1 '\x01'	0x00008901@Data
motorVars_M1.accelerationMax_Hzps	float	20.0	0x00008998@Data
motorVars_M1.flagEnableForceAngle	unsigned char	1 '\x01'	0x00008912@Data
motorVars_M1.flagMotorIdentified	unsigned char	1 '\x01'	0x00008902@Data
motorVars_M1.flagEnableMotorIdentify	unsigned char	0 '\x00'	0x00008923@Data
motorVars_M1.estState	enum <unnamed>	EST_STATE_ONLINE	0x00008928@Data
motorVars_M1.controlStatus	enum <unnamed>	MOTOR_CTRL_RUN	0x0000892A@Data
motorVars_M1.adcData.VdcBus_V	float	309.510132	0x00008950@Data
motorVars_M1.adcData	struct _HAL_ADCData_t	{VdcBus_V=309.510132,I_A={value=[0.198781252,-0.0507163107,-0...	0x00008950@Data
motorVars_M1.flagClearFaults	unsigned char	0 '\x00'	0x0000891D@Data
motorVars_M1.faultMtrUseAll	unsigned int	0	0x00008927@Data
motorVars_M1.faultMtrNow.bit	struct _FAULT_MTR_BITS_	{overVoltage=0,underVoltage=0,motorOverTemp=0,moduleOverT...	0x000089A6@Data
motorVars_M1.speed_int_Hz	float	100.0	0x000089A6@Data
motorVars_M1.angleFOC_rad	float	-0.980699837	0x000089AE@Data
motorVars_M1.angleEST_rad	float	0.295398921	0x000089B0@Data
motorVars_M1.angleGen_rad	float	-2.30981064	0x000089B4@Data
motorCtrlVars_M1.flagEnableRunMotor	unsigned char	0 '\x00'	0x000086C4@Data
motorCtrlVars_M1.flagEnableSpeedCtrl	unsigned char	0 '\x00'	0x000086C6@Data
motorCtrlVars_M1.accelerationMax_Hzps	float	0.0	0x000086E6@Data
motorCtrlVars_M1.lqSet_A	float	0.0	0x000086D8@Data
motorVars_M1.speedEST_Hz	float	100.423782	0x000089AA@Data
motorVars_M1.speed_Hz	float	100.178963	0x000089AC@Data
motorVars_M1.angleFOC_rad	float	-0.980699837	0x000089AE@Data
motorVars_M1.angleEST_rad	float	0.295398921	0x000089B0@Data
motorVars_M1.anglePLL_rad	float	2.96600747	0x000089B2@Data
motorVars_M1.RsOnLine_Ohm	float	2.68207002	0x00008994@Data
motorSetVars_M1.Rs_Ohm	float	2.68207002	0x00008990@Data
motorSetVars_M1.Ls_d_H	float	0.00926135667	0x00008792@Data
motorSetVars_M1.Ls_q_H	float	0.00926135667	0x00008794@Data
motorSetVars_M1.flux_VpHz	float	0.0628318563	0x00008796@Data
motorVars_M1.Rs_Ohm	float	2.68207002	0x00008994@Data
motorVars_M1.Ls_d_H	float	0.00926135667	0x00008996@Data
motorVars_M1.Ls_q_H	float	0.00926135667	0x00008998@Data
motorVars_M1.flux_VpHz	float	0.389726102	0x0000899A@Data
motorVars_M1.flux_Wb	float	0.0620205812	0x0000899C@Data
angleGen_M1	struct _ANGLE_GEN_Obj_	{freq_Hz=100.0,angleDeltaFactor=0.000418879034,angleDelta_rad...	0x0000861C@Data

図 3-36. ビルドレベル 4:実行時の [Expressions] ウィンドウ

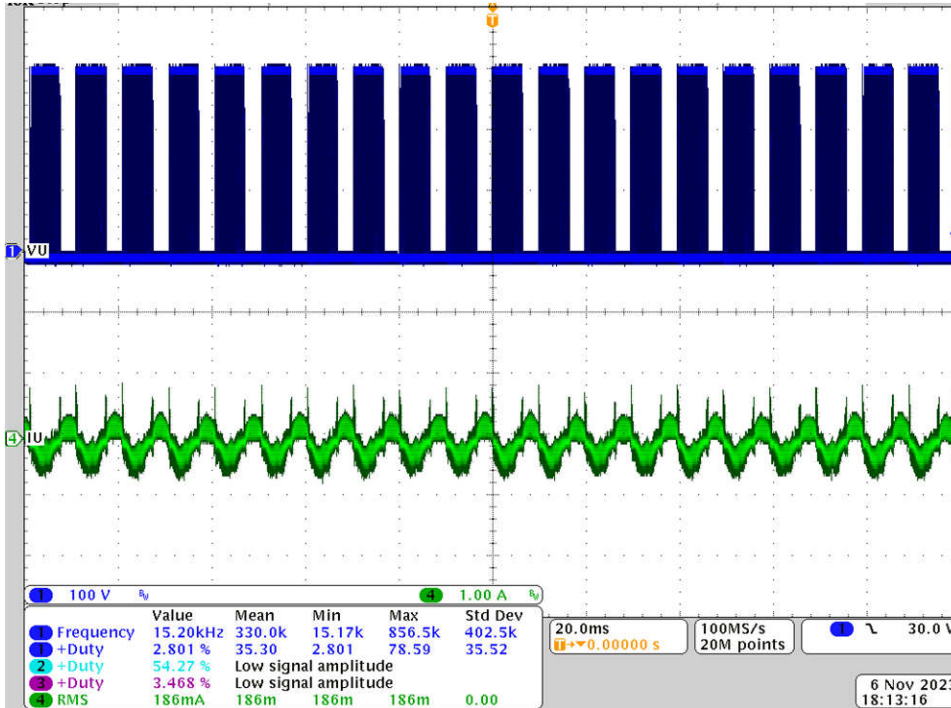


図 3-37. ビルドレベル 4:モーターの回転子角度と相電流

3.3.4.4.5 モーター駆動 FOC パラメータの調整

スライディング モード電流オブザーバは、モデル ベースの電流オブザーバと、推定されたモーター電流と実際のモーター電流との間の誤差によって駆動されるバンバン制御方式のジェネレータで構成されています。F および G のパラメータは、[セクション 2.4.2.2.1.2](#) で説明されているように、モーター パラメータ R_s と L_s に基づいて計算されます。バンバン制御方式のオブザーバ ゲイン k 、LPF のカットオフ周波数、PLL 角度トラッカの K_p と K_i は、テスト状態に応じて調整し、最適なパラメータが得られるようにする必要があります。ユーザーは FAST エスティメータと eSMO を並列に実行し、eSMO からの角度を検証してパラメータを調整できます。初期パラメータは、user-mtr1.h ファイルで定義されています。

```
// Only for eSMO
#define USER_MOTOR1_KSLIDE_MAX      (1.50f)
#define USER_MOTOR1_KSLIDE_MIN      (0.75f)

#define USER_MOTOR1_PLL_KP_MAX      (10.0f)
#define USER_MOTOR1_PLL_KP_MIN      (2.0f)
#define USER_MOTOR1_PLL_KP_SF      (5.0f)

#define USER_MOTOR1_BEMF_THRESHOLD (0.5f)
#define USER_MOTOR1_BEMF_KSLF_FC_HZ (2.0f)
#define USER_MOTOR1_THETA_OFFSET_SF (1.0f)
#define USER_MOTOR1_SPEED_LPF_FC_HZ (200.0f)
```

速度と電流の PI レギュレータ ゲインは、モーター パラメータに応じて計算されているので、ユーザーはこれらのゲインをオンラインで調整し、システムの制御性能を最適化することができます。

- CCS Debug Perspective の [Expressions] ウィンドウに、motorVars[0].Kp_spd、motorVars[0].Ki_spd、motorVars[0].Kp_lq、motorVars[0].Ki_lq、motorVars[0].Kp_id、motorVars[0].Ki_id を追加します。コンプレッサモーター駆動の PI ゲインを変更し、値を記録します。

3.3.4.4.6 弱め界磁および MTPA 制御パラメータの調整

FWC 関数と MTPA 関数が追加され、電流角度を計算するためにモーター駆動 ISR で呼び出されて、d-軸および q-軸のリファレンス電流を計算します。

1. [セクション 3.3.2](#) で説明されているように、プロジェクトのビルド構成に事前定義シンボル MOTOR1_FWC と MOTOR1_MTPA を追加して、FWC と MTPA をそれぞれ有効にします。
2. user_mtr1.h ファイルで、モーター パラメータが既知であり、正しく設定されていることを確認します。mtpa.h で、各表が適切に設定され、モーターの仕様に合わせて計算が設定されていることを確認します。
3. CCS Debug Perspective の [Expressions] ウィンドウに変数 VsRef_pu、Kp_fwc、Ki_fwc を追加し、モーターとシステムに応じて弱め界磁制御で期待される性能を達成するためにこれらのパラメータを調整します。
4. これらの変数を調整して修正したら、user_mtr1.h ファイルに新しく定義したパラメータを使用してウォッチ ウィンドウの値を記録します。

USER_M1_FWC_VREF = VsRef_pu の値。弱め界磁制御のリファレンス電圧の係数。

USER_M1_FWC_KP = Kp_fwc の値。弱め界磁制御用 PI レギュレータの Kp ゲイン。

USER_M1_FWC_KI = Ki_fwc の値。弱め界磁制御用 PI レギュレータの Ki ゲイン

5. MTPA 制御パラメータは、モーター パラメータ、 L_d 、 L_q 、 ψ_m に従って計算されるため、オンラインで調整する追加パラメータはありません。

3.3.4.4.7 電流センシング回路の調整

高精度の電流センシングは、回転子の角度と速度を推定し、さらに最適な動的モーター制御を実現するために重要です。電流センシング パラメータは、次の関連パラメータを設定することで、ハードウェアと整合させる必要があります。

- デッド バンド時間、立ち上がりエッジ遅延時間は、パワー モジュールの (ハイサイド ターンオン時間) + (ローサイド ターンオフ時間) よりも大きく、立ち下がりエッジ遅延時間は、パワー モジュールの (ハイサイド ターンオフ時間) + (ローサイド ターンオン時間) よりも大きくする必要があります。リファレンス デザインで使用されるパワー モジュールの設定は以下のとおりです。

```
//! \brief Defines the PWM deadband falling edge delay count (system clocks)
#define MTR1_PWM_DBFED_CNT (uint16_t)(2.5f * 120.0f) // 2.5us, (>2.0us)
```

```

//! \brief Defines the PWM deadband rising edge delay count (system clocks)
#define MTR1_PWM_DBRED_CNT (uint16_t)(2.5f * 120.0f) // 2.50us, (>2.0us)
  
```

- パルス幅 PWM の最小持続時間は、(ハードウェア遅延時間 + デッドバンド時間 + リンギング期間 + ADC サンプルング時間) よりも大きくなるように指定します。

```

//! \brief Defines the minimum duration, clock cycle
#define USER_M1_DCLINKSS_MIN_DURATION (450U)
  
```

- サンプル / ホールド遅延時間は、電流センシングのために、PWM 出力から ADC サンプルング時間までの遅延時間を指定します。遅延時間はハードウェアに依存し、ゲートドライバ回路の伝搬遅延と電力 FET のターンオン / ターンオフ遅延を含み、(最小持続時間 - ADC サンプルング時間) 以下となります。

```

//! \brief Defines the sample delay, clock cycle
#define USER_M1_DCLINKSS_SAMPLE_DELAY (430U)
  
```

3.4 テスト結果

以下のセクションには、デザインの特性評価から得られたテスト データを示します。テスト結果は複数のセクションに分かれており、ファンとコンプレッサのモーターの定常状態の性能とデータ、機能性能の波形、過渡性能の波形を網羅しています。

3.4.1 負荷および熱のテスト

図 3-38 は、500W dyno 負荷における 3000RPM (200Hz) の波形です。波形表示は次のとおりです。

- CH1 (青): DC バス電圧
- CH2 (水色): AC 入力電圧
- CH4 (緑): 位相 U の電流

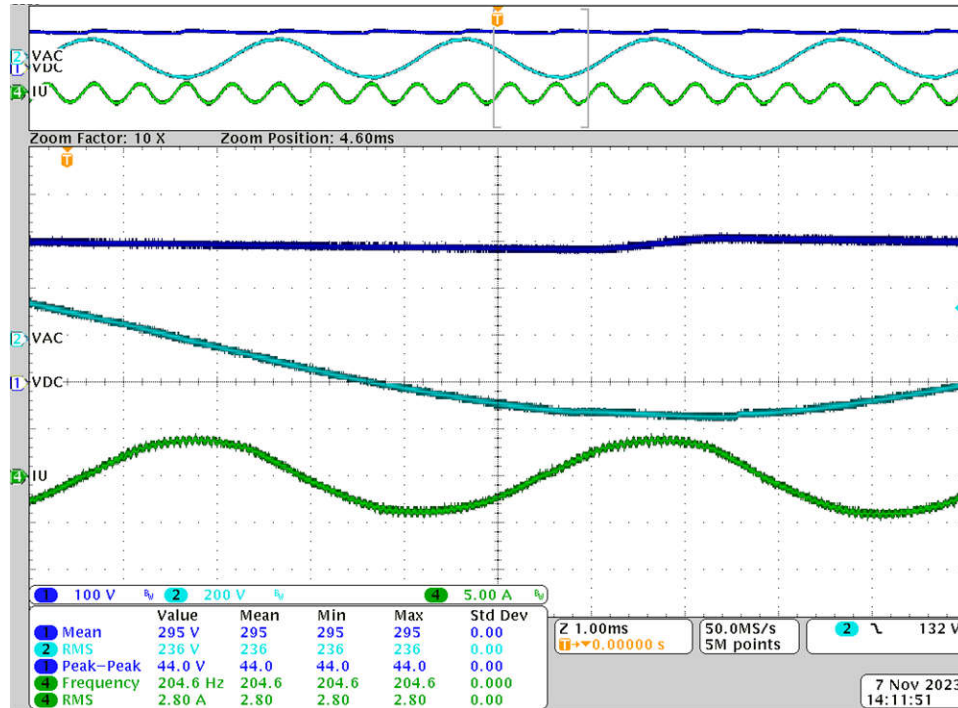


図 3-38. モーターの相電流と相電圧の波形 (500W、200Hz 時)

図 3-38 は、弱め界磁を有効にした状態の 300W dyno 負荷における 3300RPM (220Hz) の波形です。テスト対象のモーターは定格 3000RPM (200Hz) で、このとき弱め界磁の状態で作動しています。

- CH1 (青): DC バス電圧
- CH2 (水色): AC 入力電圧
- CH4 (緑): 位相 U の電流

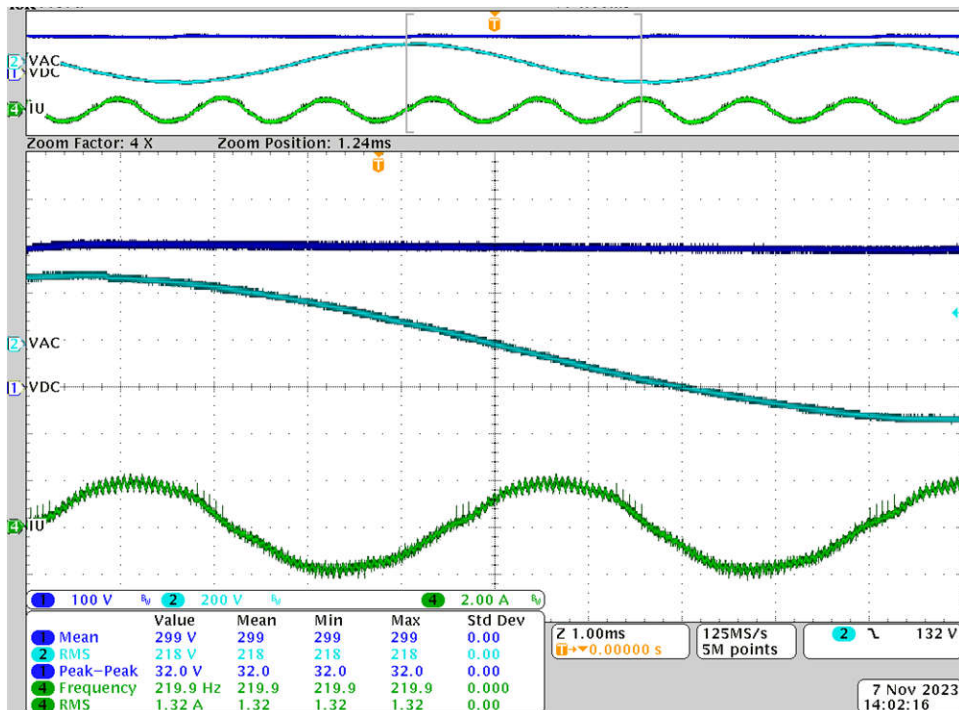


図 3-39. 弱め界磁テスト (300W、220Hz 時)

このボードは 750W で短時間 (≤ 1 分) 動作するように設計されています。温度の上昇に注意してください。ボードを大電力で動作させたり、長時間動作させたりする場合、外付け冷却ファンを使用してヒートシンクを冷却してください。図 3-38 は、500W、3000RPM (200Hz) でのボードの温度上昇を示しています。

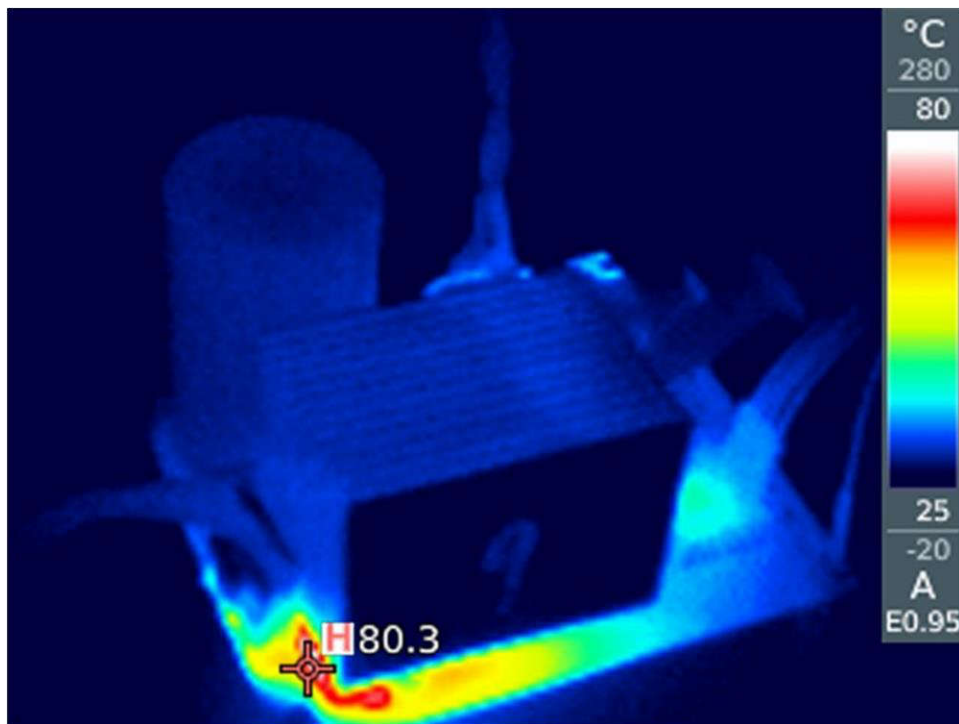


図 3-40. 熱テスト (AC220V、500W、200Hz 時)

3.4.2 外部コンパレータによる過電流保護

セクション 2.4.1.6 で説明しているように、外部過電流保護のためにコンパレータ U10 があります。図 3-41 に、外部過電流保護の波形を示します。R80 の電流が U10 の負入力で設定されたリファレンスポイントを上回ると、U10 の出力 (正味 IPM_CIN) が高レベルになり、高レベルの IPM_CIN が IPM フォルト保護をトリガして、マイクロコントローラに接続されている IPM_FAULT に 低レベルの信号を出力します。

- CH1 (青): IPM_FAULT
- CH2 (水色): IPM_CIN
- CH4 (緑): R80 の電流

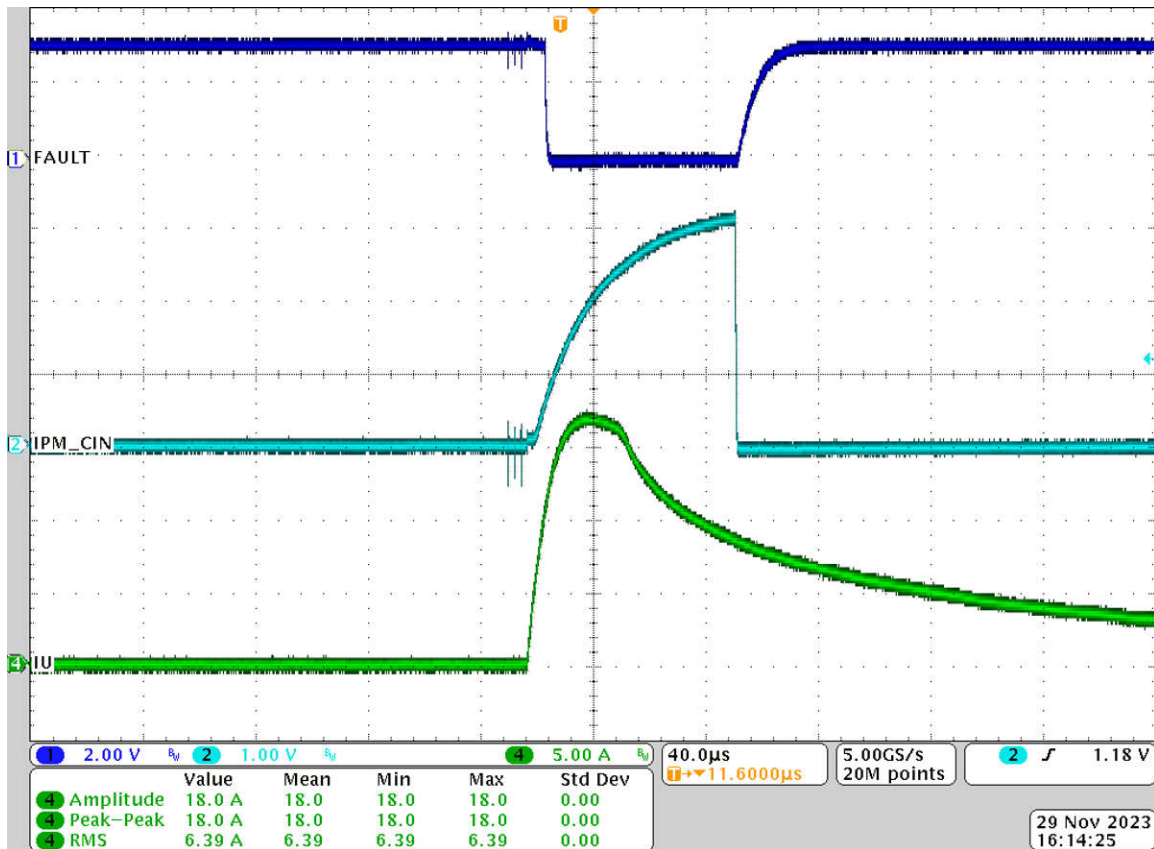


図 3-41. 外部コンパレータによる過電流保護

3.4.3 内部 CMPSS による過電流保護

セクション 2.4.1.7 で説明しているように、内部 CMPSS は過電流保護用に構成できます。図 3-42 は内部過電流保護の波形です。IPM_FAULT と IPM_CIN は両方ともトリガされないため、内部 CMPSS によってトリガされます。過電流は以下のコードで設定できます。

```
objSets->maxPeakCurrent_A = USER_M1_ADC_FULL_SCALE_CURRENT_A * 0.4975f;
```

- CH1 (青): IPM_FAULT
- CH2 (水色): IPM_CIN
- CH4 (緑): 位相 U の電流

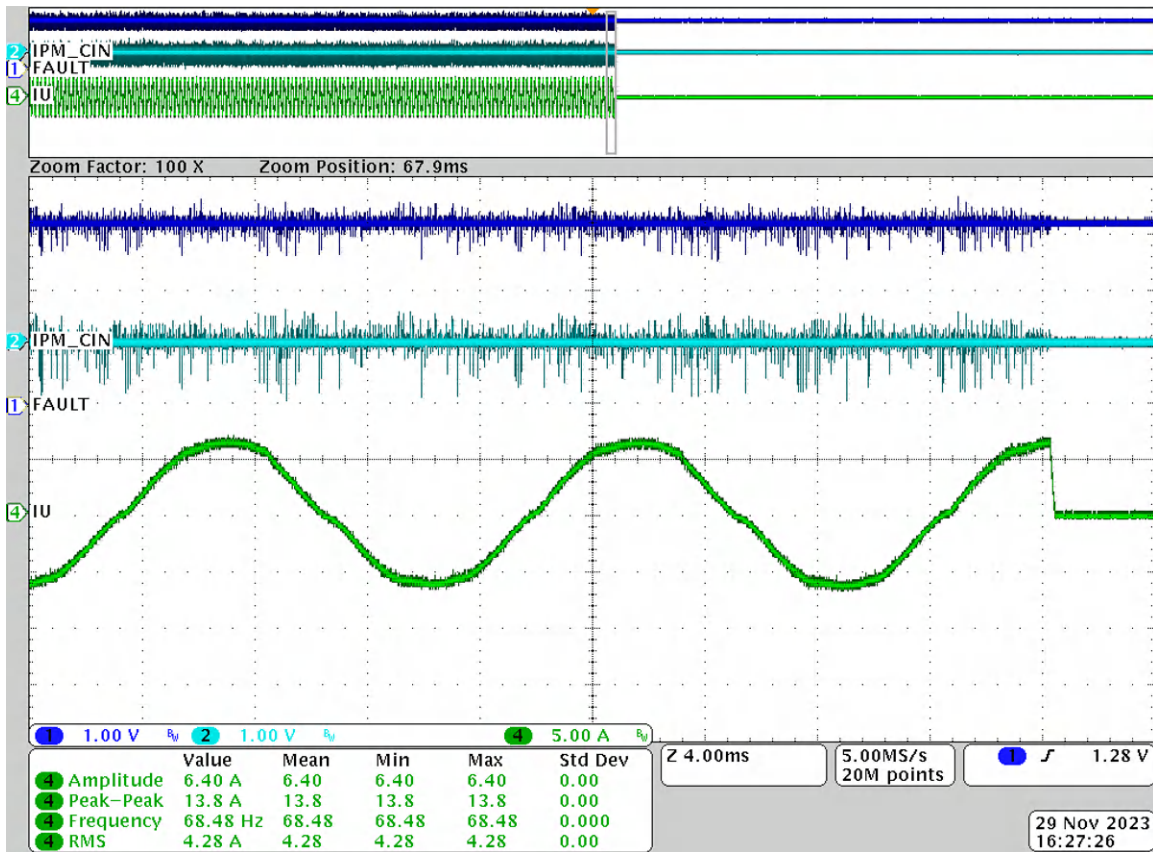


図 3-42. 内部 CMPSS による過電流保護

3.5 新しいハードウェア ボードへのファームウェアの移行

設計者がリファレンス デザインをハードウェア ボードに移行する場合は、以下のセクションで説明されているように、hal.c、hal.h、user_mtr1.h の各ファイルでモーター関連の PWM、CMPSS、ADC ペリフェラルの構成、ハードウェア パラメータ、モーター パラメータを適宜変更してください。

3.5.1 PWM、CMPSS、ADC モジュールの構成

モーターを制御するためのアプリケーション パラメータは、ハードウェアに基づいて hal.h に #define として、PWM、CMPSS、ADC モジュールのベース アドレスを構成するように記述されます。コンプレッサ モーターの PWM、CMPSS、ADC は、以下のコードで定義されています。

モーター駆動用の PWM と CMPSS のベースアドレスを構成してください。

```
// EPWM
#define MTR1_PWM_U_BASE      EPWM2_BASE
#define MTR1_PWM_V_BASE      EPWM3_BASE
#define MTR1_PWM_W_BASE      EPWM4_BASE

// CMPSS->Iu/Iv/Iw
#define MTR1_CMPSS_U_BASE     CMPSSLITE4_BASE
#define MTR1_CMPSS_V_BASE     CMPSSLITE2_BASE
#define MTR1_CMPSS_W_BASE     CMPSSLITE3_BASE
```

モーター駆動用の ADC のベースアドレスとチャンネルを構成してください。

```
// Three shunts
// Using ADCA/ADCC for current sensing
#define MTR1_ADC_TRIGGER_SOC  ADC_TRIGGER_EPWM2_SOCA // EPWM2_SOCA

#define MTR1_ADC_I_SAMPLEWINDOW  14
#define MTR1_ADC_V_SAMPLEWINDOW  20
```

```
#define MTR1_IU_ADC_BASE      ADCC_BASE          // ADCC-A7/C3*/CMP4  -SOC1
#define MTR1_IV_ADC_BASE      ADCA_BASE          // ADCA-A4*/C14/CMP2 -SOC2
#define MTR1_IW_ADC_BASE      ADCA_BASE          // ADCA-A0*/C15/CMP3 -SOC1

#define MTR1_IU_ADCRES_BASE    ADCCRESULT_BASE    // ADCC-A7/C3*
#define MTR1_IV_ADCRES_BASE    ADCARESULT_BASE    // ADCA-A4*/C14
#define MTR1_IW_ADCRES_BASE    ADCARESULT_BASE    // ADCA-A0*/C15

#define MTR1_IU_ADC_CH_NUM     ADC_CH_ADCIN3      // ADCC-A7/C3*
#define MTR1_IV_ADC_CH_NUM     ADC_CH_ADCIN4      // ADCA-A4*/C14
#define MTR1_IW_ADC_CH_NUM     ADC_CH_ADCIN0      // ADCA-A0*/C15

#define MTR1_IU_ADC_SOC_NUM     ADC_SOC_NUMBER1    // ADCC-A7/C3*  -SOC1-PPB1
#define MTR1_IV_ADC_SOC_NUM     ADC_SOC_NUMBER2    // ADCA-A4*/C14 -SOC2-PPB2
#define MTR1_IW_ADC_SOC_NUM     ADC_SOC_NUMBER1    // ADCA-A0*/C15 -SOC1-PPB1

#define MTR1_IU_ADC_PPB_NUM     ADC_PPB_NUMBER1    // ADCC-A7/C3*  -SOC1-PPB1
#define MTR1_IV_ADC_PPB_NUM     ADC_PPB_NUMBER2    // ADCA-A4*/C14 -SOC2-PPB2
#define MTR1_IW_ADC_PPB_NUM     ADC_PPB_NUMBER1    // ADCA-A0*/C15 -SOC1-PPB1
```

モーター駆動制御用のペリフェラル割り込みを構成してください。

```
// Interrupt
#define MTR1_PWM_INT_BASE      MTR1_PWM_U_BASE     // EPWM1
#define MTR1_ADC_INT_BASE      ADCC_BASE          // ADCC-A11/C0* -SOC4
#define MTR1_ADC_INT_NUM       ADC_INT_NUMBER1    // ADCC_INT1  -SOC4
#define MTR1_ADC_INT_SOC       ADC_SOC_NUMBER4     // ADCC_INT1  -SOC4

#define MTR1_PIE_INT_NUM       INT_ADCC1          // ADCC_INT1  -SOC4
#define MTR1_CPU_INT_NUM       INTERRUPT_CPU_INT1 // ADCC_INT1-CPU_INT1
#define MTR1_INT_ACK_GROUP     INTERRUPT_ACK_GROUP1 // ADCC_INT1-CPU_INT1
```

ハードウェアに基づいて hal.h に ADC ピンと CMPSS モジュール間の接続を構成してください。詳細については、『[TMS320F280013x リアルタイム マイクロコントローラ テクニカル リファレンス マニュアル](#)』の表、アナログピン、内部接続を参照してください。

```
// CMPSS->Iu/Iv/Iw
#define MTR1_CMPSS_U_BASE      CMPSSLITE4_BASE
#define MTR1_CMPSS_V_BASE      CMPSSLITE2_BASE
#define MTR1_CMPSS_W_BASE      CMPSSLITE3_BASE

#define MTR1_IU_CMPHP_SEL       ASYSCTL_CMPPHMUX_SELECT_4 // CMPSS4H-A7/C3*
#define MTR1_IU_CMPLP_SEL       ASYSCTL_CMPLPMUX_SELECT_4 // CMPSS4L-A7/C3*
#define MTR1_IV_CMPHP_SEL       ASYSCTL_CMPPHMUX_SELECT_2 // CMPSS2H-A4*/C14
#define MTR1_IV_CMPLP_SEL       ASYSCTL_CMPLPMUX_SELECT_2 // CMPSS2L-A4*/C14

#define MTR1_IW_CMPHP_SEL       ASYSCTL_CMPPHMUX_SELECT_3 // CMPSS3H-A0*/C15
#define MTR1_IW_CMPLP_SEL       ASYSCTL_CMPLPMUX_SELECT_3 // CMPSS3L-A0*/C15

#define MTR1_IU_CMPHP_MUX       1 // CMPSS4H-A7/C3*
#define MTR1_IU_CMPLP_MUX       1 // CMPSS4L-A7/C3*

#define MTR1_IV_CMPHP_MUX       0 // CMPSS2H-A4*/C14
#define MTR1_IV_CMPLP_MUX       0 // CMPSS2L-A4*/C14

#define MTR1_IW_CMPHP_MUX       2 // CMPSS3H-A0*/C15
#define MTR1_IW_CMPLP_MUX       2 // CMPSS3L-A0*/C15
```

ハードウェアに基づいて hal.h に EPWM と GPIO 出力に渡される CMPSS からのトリップ信号を構成してください。詳細については、『[TMS320F280013x リアルタイム マイクロコントローラ テクニカル リファレンス マニュアル](#)』の表、ePWM X-BAR MUX 構成表と表、OUTPUT X-BAR MUX 構成表を参照してください。

```
// XBAR-EPWM
#define MTR1_XBAR_TRIP_ADDR_L    XBAR_O_TRIP8MUX0T015CFG
#define MTR1_XBAR_TRIP_ADDR_H    XBAR_O_TRIP8MUX16T031CFG

#define MTR1_XBAR_INPUT1         XBAR_INPUT1
#define MTR1_TZ_OSHT1           EPWM_TZ_SIGNAL_OSHT1

#define MTR1_XBAR_TRIP           XBAR_TRIP8
#define MTR1_DCTRIPIN           EPWM_DC_COMBINATIONAL_TRIPIN8y
```

```
// XBAR-EPWM->Iu/Iv/Iw
#define MTR1_IU_XBAR_EPWM_MUX XBAR_EPWM_MUX06_CMPSS4_CTRIPH_OR_L // CMPSS4-HP&LP, A7/C3*
#define MTR1_IV_XBAR_EPWM_MUX XBAR_EPWM_MUX02_CMPSS2_CTRIPH_OR_L // CMPSS2-HP&LP, A4*/C14
#define MTR1_IW_XBAR_EPWM_MUX XBAR_EPWM_MUX04_CMPSS3_CTRIPH_OR_L // CMPSS3-HP&LP, A0*/C15

#define MTR1_IU_XBAR_MUX XBAR_MUX06 // CMPSS4-HP&LP, A7/C3*
#define MTR1_IV_XBAR_MUX XBAR_MUX02 // CMPSS2-HP&LP, A4*/C14
#define MTR1_IW_XBAR_MUX XBAR_MUX04 // CMPSS3-HP&LP, A0*/C15
```

関連する ADC チャンネルは、ピンがコンパレータ サブシステム (CMPSS) に内部接続されているモーター電流センシングに使用されます。以下のコードに示すように、hal.c ファイルの HAL_setupCMPSSs() 関数で CMPSS レジスタを構成してください。3 つの CMPSS モジュールは、モーターの U 相、V 相、W 相に対する正負の過電流保護を実装するために使用されます。

```
void HAL_setupCMPSSmTR(HAL_MTR_Handle handle)
{
    HAL_MTR_Obj *obj = (HAL_MTR_Obj *)handle;

    #if defined(DMCPFC_REV3P2) || defined(DMCPFC_REV3P1)
    #if !defined(MOTOR1_DCLINKSS) || !defined(MOTOR2_DCLINKSS)
        uint16_t cmpsaDACH;
    #endif // !(MOTOR1_DCLINKSS || MOTOR2_DCLINKSS)
        uint16_t cmpsaDACL;
        ...
    #else // !MOTOR1_DCLINKSS, Three-shunt
        cmpsaDACH = MTR1_CMPSS_DACH_VALUE;
        cmpsaDACL = MTR1_CMPSS_DACL_VALUE;

        ASysCtl_selectCMPHPMux(MTR1_IU_CMPHP_SEL, MTR1_IU_CMPHP_MUX);

        ASysCtl_selectCMPHPMux(MTR1_IV_CMPHP_SEL, MTR1_IV_CMPHP_MUX);

        ASysCtl_selectCMPLPMux(MTR1_IW_CMPLP_SEL, MTR1_IW_CMPLP_MUX);
        ...
    } // end of HAL_setupCMPSSs() function
    return;
```

CMPSS で生成された信号は X-Bar に送られ、さまざまな独自の方法で信号を組み合わせて、IPM #Fault からの外部 TZ 信号を含む複数のソースからの独自のトリップ イベントにフラグを立て、フォルト保護を実装することができます。フォルトには、CMPSS からの過電流信号と、パワー モジュールからのフォルト インジケータ出力が含まれます。以下のコードに示すように、hal.c ファイルの HAL_setupMtrFaults() 関数で XBAR レジスタを構成してください。

```
void HAL_setupMtrFaults(HAL_MTR_Handle handle)
{
    HAL_MTR_Obj *obj = (HAL_MTR_Obj *)handle;
    uint16_t cnt;

    // Configure TRIP 7 to OR the High and Low trips from both
    // comparator 5, 3 & 1, clear everything first
    EALLOW;
    HWREG(XBAR_EPWM_CFG_REG_BASE + MTR1_XBAR_TRIP_ADDRL) = 0;
    HWREG(XBAR_EPWM_CFG_REG_BASE + MTR1_XBAR_TRIP_ADDRH) = 0;
    EDIS;
    ...
    // what do we want the OST/CBC events to do?
    // TZA events can force EPWMxA
    // TZB events can force EPWMxB
    EPWM_setTripZoneAction(obj->pwmHandle[cnt],
        EPWM_TZ_ACTION_EVENT_TZA,
        EPWM_TZ_ACTION_LOW);

    EPWM_setTripZoneAction(obj->pwmHandle[cnt],
        EPWM_TZ_ACTION_EVENT_TZB,
        EPWM_TZ_ACTION_LOW);
    ...
    // clear any spurious fault
    EPWM_clearTripZoneFlag(obj->pwmHandle[0], HAL_TZFLAG_INTERRUPT_ALL);
    EPWM_clearTripZoneFlag(obj->pwmHandle[1], HAL_TZFLAG_INTERRUPT_ALL);
    EPWM_clearTripZoneFlag(obj->pwmHandle[2], HAL_TZFLAG_INTERRUPT_ALL);
```

```

    return;
}

```

以下のコードに示すように、ハードウェアに基づいて hal.c ファイルの HAL_setupGPIOs() で GPIO を構成してください。

```

void HAL_setupGPIOs(HAL_Handle handle)
{
    ...
    // GPIO2->EPWM2A->M1_UH
    GPIO_setPinConfig(GPIO_2_EPWM2_A);
    GPIO_writePin(2, 0);
    GPIO_setDirectionMode(2, GPIO_DIR_MODE_OUT);
    GPIO_setPadConfig(2, GPIO_PIN_TYPE_STD);

    // GPIO3->EPWM2B->M1_UL
    GPIO_setPinConfig(GPIO_3_EPWM2_B);
    GPIO_writePin(3, 0);
    GPIO_setDirectionMode(3, GPIO_DIR_MODE_OUT);
    GPIO_setPadConfig(3, GPIO_PIN_TYPE_STD);
    ...
    return;
} // end of HAL_setupGPIOs() function

```

使用するモーター制御用 CMPSS に応じて、構成コードは hal.h ファイルの HAL_enableMtrPWM() と HAL_clearMtrFaultStatus() で、以下の太字のように変更する必要があります。

```

static inline void HAL_enableMtrPWM(HAL_MTR_Handle handle)
{
    HAL_MTR_Obj *obj = (HAL_MTR_Obj *)handle;

    obj->flagEnablePWM = true;

    #if defined(DMCPFC_REV3P2) || defined(DMCPFC_REV3P1)
        if(obj->motorNum == MTR_1)
        {
            #if defined(MOTOR1_DCLINKSS)
                // Clear any comparator digital filter output latch
                CMPSS_clearFilterLatchLow(obj->cmpssHandle[0]);
            #else
                // !MOTOR1_DCLINKSS
                // Clear any comparator digital filter output latch
                CMPSS_clearFilterLatchHigh(obj->cmpssHandle[0]);

                CMPSS_clearFilterLatchHigh(obj->cmpssHandle[1]);

                CMPSS_clearFilterLatchLow(obj->cmpssHandle[2]);
            #endif
        }
    #endif
    return;
} // end of HAL_enableMtrPWM() function

```

```

static inline void HAL_clearMtrFaultStatus(HAL_MTR_Handle handle)
{
    HAL_MTR_Obj *obj = (HAL_MTR_Obj *)handle;
    ...
    #if defined(HVMTRPFC_REV1P1) || defined(WMINVBRD_REV1P0) || defined(TIDSMPFC_REV3P2)
        // Clear any comparator digital filter output latch
        CMPSS_clearFilterLatchHigh(obj->cmpssHandle[0]);
        CMPSS_clearFilterLatchLow(obj->cmpssHandle[0]);

        CMPSS_clearFilterLatchHigh(obj->cmpssHandle[1]);
        CMPSS_clearFilterLatchLow(obj->cmpssHandle[1]);

        CMPSS_clearFilterLatchHigh(obj->cmpssHandle[2]);
        CMPSS_clearFilterLatchLow(obj->cmpssHandle[2]);
    #endif
    ...
    return;
} // end of HAL_clearMtrFaultStatus() function

```

3.5.2 ハードウェア ボード パラメータの設定

user_mtr1.h ファイルには、モーター制御用のすべてのユーザー パラメータが格納されます。AD コンバータへの入力における相電流と相電圧の最大値は、ハードウェアに依存するものであり、電流と電圧のセンシングおよび ADC 入力に

対するスケーリングに基づく必要があります。使用される電流センサと電圧 (相) センサの数は `user_mtr1.h` で定義されており、ハードウェアに依存します。

構成可能なパラメータはすべて、`user_mtr1.h` ファイルで定義されています。これらのパラメータは、`Motor_Drive_Parameters_Calculation.xlsx` Microsoft® Excel® スプレッドシートを使用して計算できます。このファイルは、フォルダ (`.\solutions\tida_010265\docs`) にある TIDA-010265 のアーカイブ ファイルに含まれています。これらの値を計算し、太字で示されたこれらのパラメータを以下のコードのように `user_mtr1.h` にコピーしてください。

```

///  
// Full scale voltage of AD converter, not the current voltage  
#define USER_M1_ADC_FULL_SCALE_VOLTAGE_V      (404.1292683f)  
  
///  
//!  
#define USER_M1_VOLTAGE_FILTER_POLE_HZ        (416.3602877f)  
  
///  
//!  
#define USER_M1_ADC_FULL_SCALE_CURRENT_A      (15.972f)

```

3.5.3 フォルト保護パラメータの構成

このシステムには、過電流、過電圧、低電圧、ストール、過負荷、起動失敗などのフォルト管理が実装されています。フォルト保護パラメータは、以下のコードに示すように `user_mtr1.h` で定義されており、ハードウェア ボード、モーター、およびシステムに依存します。

```

///  
#define USER_MOTOR1_OVER_CURRENT_A            (6.5f)           // A  
  
///  
#define USER_M1_LOST_PHASE_CURRENT_A          (0.2f)  
  
///  
#define USER_M1_UNBALANCE_RATIO              (0.2f)

```

```

///  
#define USER_M1_OVER_VOLTAGE_FAULT_V          (380.0f)  
  
///  
#define USER_M1_OVER_VOLTAGE_NORM_V          (350.0f)  
  
///  
#define USER_M1_UNDER_VOLTAGE_FAULT_V        (100.0f)

```

3.5.4 モーターの電気的パラメータの設定

`user_mtr1.h` に含まれる PMSM モーターおよび BLDC モーターのパラメータは、以下のコードに示すとおりです。モーターのパラメータは、このモーターに FAST 技術が実装されている場合、またはモーターのデータシートから取得することで、特定できます。

```

#define USER_MOTOR1_TYPE                       MOTOR_TYPE_PM  
#define USER_MOTOR1_NUM_POLE_PAIRS            (4)  
#define USER_MOTOR1_Rr_Ohm                    (0.0f)  
#define USER_MOTOR1_Rs_Ohm                    (2.68207002f)  
#define USER_MOTOR1_Ls_d_H                     (0.00926135667f)  
#define USER_MOTOR1_Ls_q_H                     (0.00926135667f)  
#define USER_MOTOR1_RATED_FLUX_VpHz          (0.381890297f)

```

3.6 MSPM0 ファームウェアの概要

MSPM0G1507 ドーターボードのファームウェアについては、テキサス・インスツルメンツの担当者にお問い合わせください。

4 設計とドキュメントのサポート

4.1 デザイン ファイル

4.1.1 回路図

回路図をダウンロードするには、[TIDA-010265](#) のデザイン ファイルを参照してください。

4.1.2 部品表 (BOM)

部品表 (BOM) をダウンロードするには、[TIDA-010265](#) のデザイン ファイルを参照してください。

4.1.3 PCB レイアウトに関する推奨事項

このリファレンス デザインは、2 層で 2 オンスの銅を使用した PCB を採用しており、コストとボード面積を節約するために底面に SMD 部品を配置して実装されています。PCB を設計する際には、注意すべき重要な点がいくつかあります。以下に、システムレベルの配置と各ブロックのレイアウトについて説明します。

- 各部品を高電圧と低電圧、高電流と低電流、高い独立性と低い独立性のグループにそれぞれ分けます。マイクロコントローラ関連の信号や IPM の入力側など、低電圧で高インピーダンスの部品と信号はまとめて配置し、配線してください。これらの領域には、銅流し込みを使用して、統合された GND プレーンを設けてください。AC 入力、フィルタ、整流器、IPM 出力の各側は、高電圧、大電流、低インピーダンスの部品や信号であるため、大電流経路を作るために幅広のパターンや銅箔で配線し、干渉を抑えるために上記の低電圧や高インピーダンスの信号と分離してください。
- 大電力経路にある部品は、PCB の外縁に可能な限り最短距離で配置します。マイクロコントローラは、制御が必要なすべてのパワー ブロックからの最適な距離を考慮して、中央に配置します。ピン配置は、制御信号または帰還信号のトレースの長さ、アナログ信号とデジタル信号の交差を最小限に抑えるように設定されます。
- AC ライン保護と EMI フィルタ
 - AC ライン保護部品は、接続経路までの最小距離内に近接して配置されます。保護回路と EMI フィルタ回路の周囲には、アース接続の保護を設けています。
- モータードライブ
 - 高リップルの要件に対応するため、モーター駆動はフィルム コンデンサと DC バス コンデンサ バンクのできるだけ近くに配置されます。
 - 電流センシングには、4 線式センシングのローサイド シャント抵抗方式が採用されます。シャント抵抗からオープン回路へのセンシング信号の接続には、インピーダンス マッチング抵抗による差動ペアが使用されます。シャント抵抗はモジュールの近くに配置され、直接接地された銅プレーンに接続されます。
- 補助電源
 - 補助電源の GND は、DC バス コンデンサ バンクを直接、独立して接続し、インバータの大電流で高周波の GND パターンから低電流を分離します。

4.1.4 Altium プロジェクト

Altium プロジェクト ファイルをダウンロードするには、[TIDA-010265](#) のデザイン ファイルを参照してください。

4.1.5 ガーバー ファイル

ガーバー ファイルをダウンロードするには、[TIDA-010265](#) のデザイン ファイルを参照してください。

4.2 ソフトウェア ファイル

[CCSTUDIO](#) にある Code Composer Studio 統合開発環境をダウンロードしてください。

[C2000WARE-MOTORCONTROL-SDK](#) の設計ファイルから、[TIDA-010265](#) ハードウェア固有のソフトウェア設計ファイルをダウンロードしてください。

4.3 ドキュメントのサポート

1. テキサス・インスツルメンツ、『[TMS320F280013x マイクロコントローラ](#)』データシート
2. テキサス・インスツルメンツ、『[TMS320F280013x リアルタイム マイクロコントローラ テクニカル リファレンス マニュアル](#)』
3. テキサス・インスツルメンツ、『[InstaSPIN-FOC および InstaSPIN-MOTION](#)』ユーザー ガイド
4. テキサス・インスツルメンツ、『[モーター制御 SDK ユニバーサル プロジェクト およびラボ](#)』ユーザー ガイド
5. テキサス・インスツルメンツ、『[C2000™ ソフトウェア周波数応答アナライザ \(SFRA\) ライブラリおよび補償デザイナー](#)』ユーザー ガイド
6. テキサス・インスツルメンツ、『[単一 DC リンク シャント付き PMSM のセンサレス FOC](#)』アプリケーション ノート
7. テキサス・インスツルメンツ、『[C2000 SysConfig](#)』アプリケーション ノート

4.4 サポート・リソース

テキサス・インスツルメンツ E2E™ サポート・フォーラムは、エンジニアが検証済みの回答と設計に関するヒントをエキスパートから迅速かつ直接得ることができる場所です。既存の回答を検索したり、独自の質問をしたりすることで、設計に必要な支援を迅速に得ることができます。

リンクされているコンテンツは、各寄稿者により「現状のまま」提供されるものです。これらはテキサス・インスツルメンツの仕様を構成するものではなく、必ずしもテキサス・インスツルメンツの見解を反映したものではありません。テキサス・インスツルメンツの[使用条件](#)を参照してください。

4.5 商標

FAST™, C2000™, テキサス・インスツルメンツの E2E™, InstaSPIN™, Code Composer Studio™, and テキサス・インスツルメンツ E2E™ are trademarks of Texas Instruments.

Arm® and Cortex® are registered trademarks of Arm Limited.

Microsoft®, Windows®, and Excel® are registered trademarks of Microsoft Corporation.

すべての商標は、それぞれの所有者に帰属します。

5 著者について

HELY ZHANG は テキサス・インスツルメンツのシステム アプリケーション エンジニアで、家電製品に関連する電力供給とモーター インバータの開発を担当しています。Hely は 2002 年に安徽理工大学からパワー エレクトロニクスの修士号を取得し、テキサス・インスツルメンツに入社する前は SolarEdge と General Electric で働いていました。

重要なお知らせと免責事項

TI は、技術データと信頼性データ(データシートを含みます)、設計リソース(リファレンス・デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、TI 製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した TI 製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとし、

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている TI 製品を使用するアプリケーションの開発の目的でのみ、TI はその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。TI や第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、TI およびその代理人を完全に補償するものとし、TI は一切の責任を拒否します。

TI の製品は、[TI の販売条件](#)、または [ti.com](#) やかかる TI 製品の関連資料などのいずれかを通じて提供する適用可能な条項の下で提供されています。TI がこれらのリソースを提供することは、適用される TI の保証または他の保証の放棄の拡大や変更を意味するものではありません。

お客様がいかなる追加条項または代替条項を提案した場合でも、TI はそれらに異議を唱え、拒否します。

郵送先住所 : Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2024, Texas Instruments Incorporated