参考資料

TI Designs: TIDM-1007
C2000™ MCUを使用するインターリーブCCMトーテム・ポール・ブリッジレスPFCのリファレンス・デザイン

概要

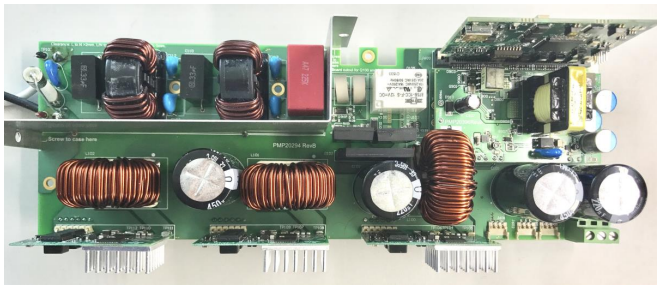
このリファレンス・デザインでは、C2000™マイクロコントローラ(MCU)とLMG3410を使用して、インターリーブ連続導通モード(CCM)トーテム・ポール(TTPL)ブリッジレス力率改善(PFC)電源段を制御する方法を紹介します。この電源トポロジは窒化ガリウム(GaN)デバイスを使用しているため、より高い効率を実現でき、電源のサイズが小さくなります。このデザインは、フェーズ・シェディングとアダプティブ・デッドタイムによって効率性を高め、入力コンデンサ補償方式によって軽負荷時の力率を向上させ、非線形電圧ループによって過渡時の電圧スパイクを低減します。このリファレンス・デザインで利用可能なハードウェアとソフトウェアにより、製品開発期間を短縮できます。

リソース

TIDM-1007	デザイン・フォルダ
TMS320F280049	プロダクト・フォルダ
TMS320F28075	プロダクト・フォルダ
LMG3410	プロダクト・フォルダ
UCC27714	プロダクト・フォルダ
OPA2376	プロダクト・フォルダ
SN74LVC1G3157DRYR	プロダクト・フォルダ
ISO7831	プロダクト・フォルダ
TLV713	プロダクト・フォルダ
C2000WARE-DIGITALPOWER-SDK	ツール・フォルダ

TI E2E™ Community

E2E™エキスパートに質問

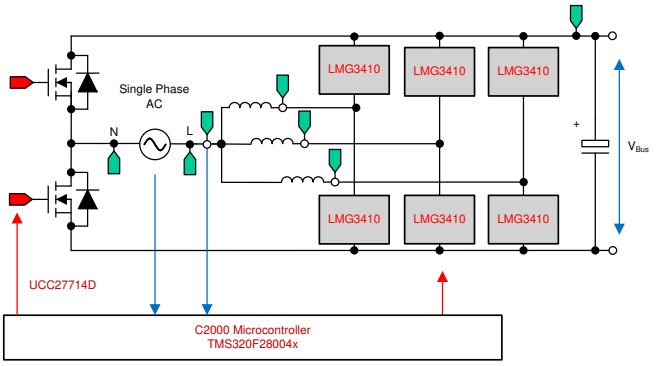


特長

- インターリーブされた3.3kW単相ブリッジレスCCMトーテム・ポールPFC段
- 100kHzのパルス幅変調(PWM)スイッチング
- 出力電圧をプログラム可能、公称出力380V DC
- 2%未満の全高調波歪み(THD)
- 98%を上回るピーク効率
- powerSUITE™サポートにより、ユーザー要件に設計を簡単に適合
- ソフトウェア周波数応答アナライザ(SFRA)により、オープン・ループ・ゲインを迅速に測定
- PWMのソフトスタートにより、TTPL PFCでのゼロ電流スパイクが低減
- ドライバ・ライブラリによるF28004xのソフトウェア・サポートに項目を
- C28xとCLAのどちらでも同じソースコードで制御ループを実行可能

アプリケーション

- 電気自動車(EV)用のオンボード充電器
- テレコム整流器
- ドライブ、溶接、その他産業用途



注: LMG3410製品とその提供状況については、http://www.ti.com/product/LMG3410を参照してください。このデバイスはAEC-Q100認定済みではありません。追加情報については、TIにお問い合わせください。

1 システム概要

インターリーブTTPL PFCは、高出力化、高効率化、高密度化の傾向にあるEV充電器や産業機器に最適な回路方式です。図 1にTTPLブリッジレスPFCのTIDM-1007基板への実装を示します。

1.1 主なシステム仕様

表 1 にインターリーブCCM TTPL PFCリファレンス・デザインの電力仕様を示します。

表 1. 主なシステム仕様

パラメータ	仕様
入力電圧(Vin)	AC 120Vrms VL-N、60Hz または AC 230Vrms VL-N、50Hz
入力電流(Iin)	16A RMS (最大値)
出力電圧(Vout)	380V DCバス(公称値)
出力電流(Iout)	10A (最大値)
定格電力	単相120Vrms時1.65KW または 単相230Vrms時3.3KW
電流THD	120Vrms L-N定格負荷時2%未満
効率	230Vrms入力時ピーク98.7%、120Vrms入力時ピーク97.7%超
1次フィルタインダクタ	478µH
出力容量	880µF
PWMスイッチング周波数	100kHz

WARNING

TI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、一般消費者向けの完成品とはみなしておりません。

TI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伴う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。

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、使用した場合に感電、火災、けがの原因となる電圧および電流で動作します。けがをしたり、物品を破損しない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。

CAUTION

電源を入れたままその場を離れないでください。

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。*感電する可能性があります。基板は、不適切に取り扱った場合に感電、火災、けがの原因となる電圧および電流で動作します。けがをしたり、物品を破損しない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。安全のため、過電圧/過電流保護機能を備え、絶縁された試験装置の使用を強くお勧めします。

TIは、電圧/絶縁要件を確認・理解した上で基板やシミュレーションにて電圧を加えることをユーザーの責任と考えます。*電圧を加える際、デバイスやその接続部品には触れないでください。*

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℃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。

2 システム概要

インターリーブTTPL PFCは、高出力化、高効率化、高密度化の傾向にあるEV充電器に最適な方式です。図 1にTTPLブリッジレスPFCのこのリファレンス・デザインへの実装を示します。

2.1 ブロック図

図 1 に主要なTI部品を使用した、このリファレンス・デザインのブロック図を示します。

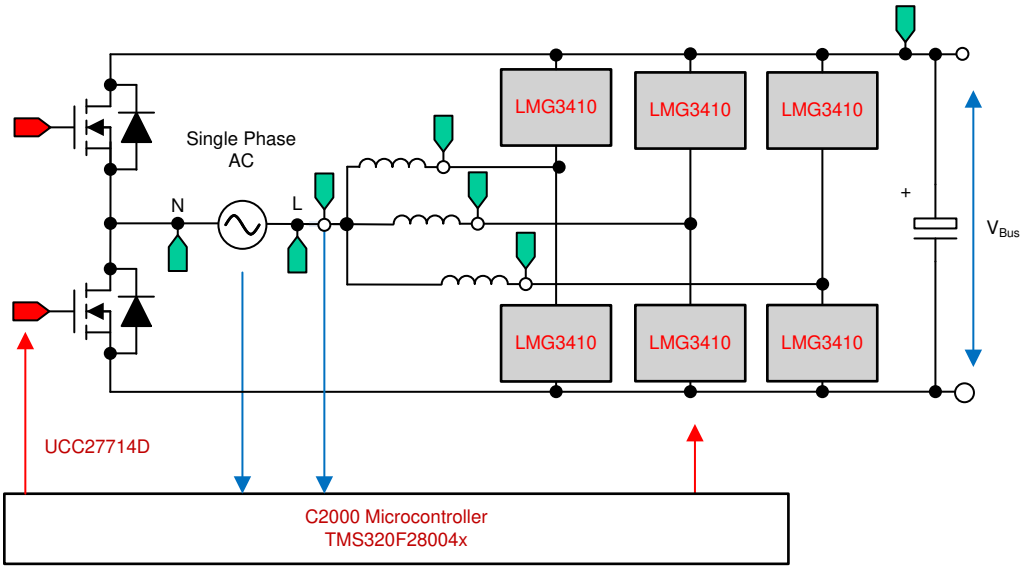


図 1. 電源回路方式ブロック図

2.2 設計上の検討事項

このリファレンス・デザインの検出回路について以下に詳述します。検出回路の詳細については、*<install_location>\solutions\tidm_1007\hardware*のC2000Ware Digital Power SDK Installディレクトリで入手可能な*calculations.xlsx*ファイルもご参照ください。

2.2.1 入力AC電圧検出

図 2に示すように、基板GNDへの分圧抵抗によりライブ側とニュートラル側を検出します。コントローラで2つの測定値を減算してVac検出を実現します。

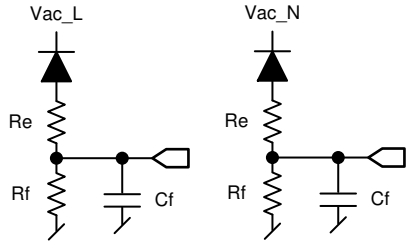


図 2. 入力AC電圧検出

2.2.2 バス電圧検出

図 3に示すように、同じく分圧抵抗によりバス電圧を検出します。

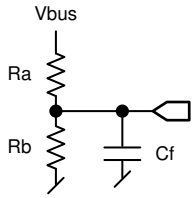


図 3. バス電圧検出回路

2.2.3 AC電流検出

ホール効果センサにより全電流を検出します。ホール効果センサにはオフセットが存在し、その範囲はADCで測定可能な範囲とは異なります。このため、図 4に示す回路を使用して、ADCの入力可能な範囲と一致するように電圧を引き上げます。

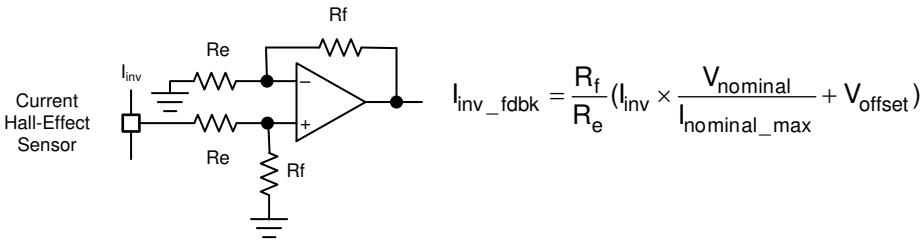


図 4. ホール効果センサを使用した電流検出

2.2.4 検出フィルタ

RCフィルタにより、コントローラに接続する前に信号がフィルタ処理されます。図 5に示すように、このリファレンス・デザインでは、すべての検出信号に共通のRCフィルタを使用します。

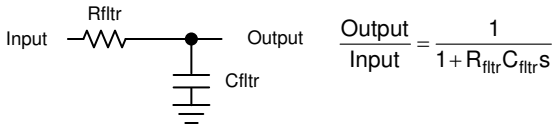


図 5. RCフィルタ

2.2.5 保護(CMPSS)

大半のパワーエレクトロニクスコンバータは、過電流から保護する必要があります。図 6に示すように、このリファレンス・デザインは複数のコンパレータを用いて、検出用の基準を生成します。

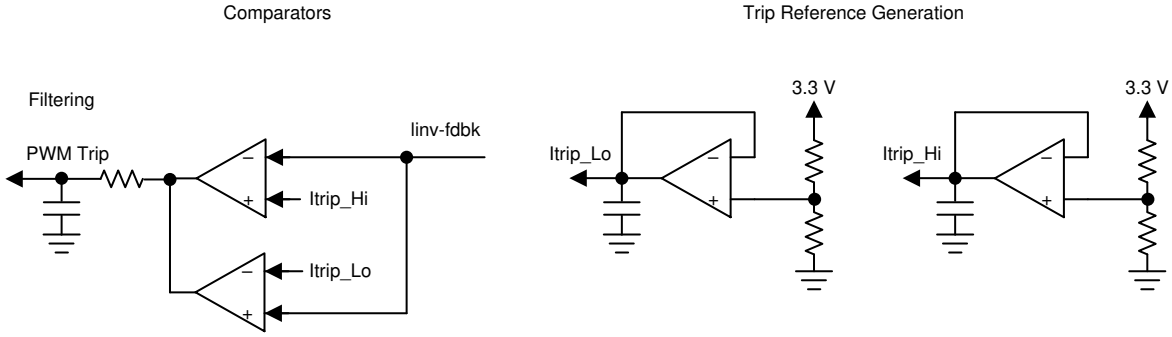


図 6. コンパレータと基準発生器を使用したPWMの検出信号生成

TMS320F28377DなどのC2000 MCUを使用すれば、CMPSSの一部として内蔵されたウィンドウコンパレータをPWMモジュールに内部接続し、PWMの高速検出を実現するため、このような回路は不要となります。図 7に示すように、オンチップのウィンドウコンパレータにより、追加部品が不要になるため、最終機器の基板面積とコストを削減できます。

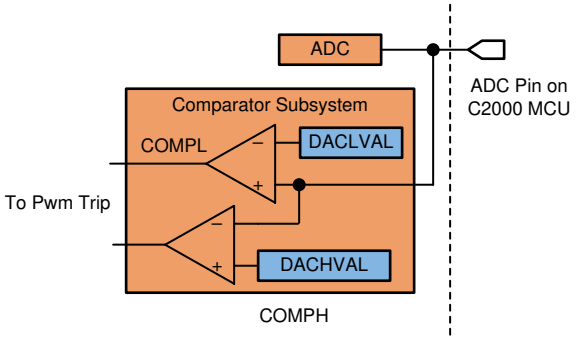


図 7. 過電流保護用のCMPSS

2.3 使用製品

2.3.1 C2000™ MCU F28004x

C2000 MCUは、リアルタイム制御機器に最適化されたMCUファミリの製品です。高速かつ高品質のA/Dコントローラにより、電流/電圧信号の正確な測定を実現し、内蔵するコンパレータサブシステム(CMPSS)により、外部デバイスなしで過電流や過電圧から保護できます。最適化されたCPUコアにより、制御ループの高速実行が可能です。オンチップの三角関数演算ユニット(TMU)により、三角関数演算が高速化されます。また、F28004xおよびF2837xのCLA (制御補償器アクセラレータ)を使用することもできます。CLAはコプロセッサであり、これを使用することによってCPUの負荷を軽減し、ループをより高速に実行したり、C2000 MCUの機能をより多く利用できます。

2.3.2 LMG3410

LMG3410シングルチャネルGaN電力段は、8mm×8mmのQFNパッケージに70mΩ、600V GaNパワートランジスタと専用ドライバを搭載しています。ダイレクトドライブアーキテクチャの採用により、ノーマリー・オフ・デバイスを実現し、GaNパワートランジスタ本来のスイッチング性能を提供します。LMG3410が電源オフになると、内蔵された低電圧シリコンMOSFETがそのソース端子を介してGaNデバイスを完全にオフにします。通常動作時は、低電圧シリコンMOSFETが継続的にオンになり、GaNデバイスは内部生成された負電圧電源により直接開閉されます。内蔵されたドライバは、追加保護機能や便利な機能を提供します。高速な過電流、過熱、および低電圧誤動作防止(UVLO)保護機能により、フェイルセーフ・システムを実現します。デバイスの状態は、異常検出出力により通知されます。内蔵する5V低ドロップアウト・レギュレータは、外部信号アイソレータに最大5mAを供給可能です。さらに、外部調整可能なスルーレートと低インダクタンスのQFNパッケージとにより、スイッチング損失やドレインリンギング、電気的雑音の発生を最小限に抑えます。

2.3.3 UCC27714

UCC27714は、4Aのソース/シンク電流容量を備えた600V 1次側/2次側ゲートドライバであり、パワーMOSFETやIGBTの駆動に適しています。このデバイスは、1つのGND基準チャネル(LO)と1つのフローティングチャネル(HO)で構成され、ブートストラップ電源で動作するように設計されています。堅牢性とノイズ耐性に優れており、HSピンで最大－8VDCの負電圧に対しても動作ロジックを保持できます(VDD＝12V時)。

2.4 システム設計理論

2.4.1 PWM

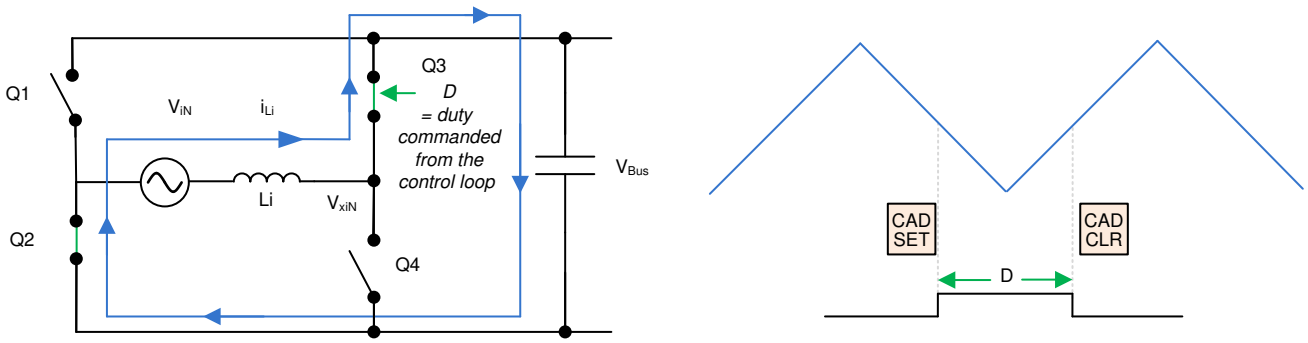


図 8. TTPL PFCの単相図

図 8 にインターリーブTTPL PFC方式の単相概略図を示します。この整流器を制御するには、デューティ・サイクルを制御することによって、電圧を直接制御します。この電圧制御は、ソフトウェア変数*Duty*または*D*が1と等しいときQ3が常時オンになるように設定した場合に可能であり、この設定により電圧 V_{xiN} はV_{bus}電圧と等しくなります。*Duty*を0に設定すると、Q3は決してオンにならず、Q4は常時接続状態になるため、 V_{xiN} 電圧は0になります。

2.4.2 電流ループモデル

電流ループモデルを理解するには、まずインダクタ電流に注目します。図 8では、スイッチQ3およびQ4に接続されたPWM変調器にデューティ・サイクル(D)が設定されます。ここから、式 1を次のように表します。

$$V_{xiN} = D \times V_{bus} \quad (1)$$

注: Dを1に設定すると、Q3は常時オンになり、Dを0に設定すると、Q3は常時オフになります。

インダクタを流れる電流を変調するには、Q3およびQ4スイッチのデューティ・サイクル制御を用いて電圧 V_{xiN} を制御します。電流の方向はACラインから整流器に流れ込む方向で正となり、DCバス・フィードフォワードおよびAC電圧フィードフォワードの使用時にはグリッドがかなり強力になると想定されます。図 9に電流ループの概略図を示し、電流ループプラントモデルを式 2のように表します。

$$H_{p\_i} = \frac{i^*_{Li}}{D} = \frac{1}{K_{v\_gain}} \times K_{i\_gain} \times K_{i\_fltr} \times G_d \times \frac{1}{Z_i} \quad (2)$$

ここで

- は、検出した最大バス電圧の逆数、です。 $K_{v\_gain} \frac{1}{V_{busMaxSense}}$
- K_{i_gain} は、検出した最大AC電流の逆数、$\frac{1}{I_{AC\_MaxSense}}$ です。
- K_{i_fltr} は、電流センサからADCピンに接続されたRCフィルタの応答です。
- G_d は、PWM更新に伴うデジタル遅延であり、デジタル制御は電流指令です。
- i^*_{Li} は電流指令です。

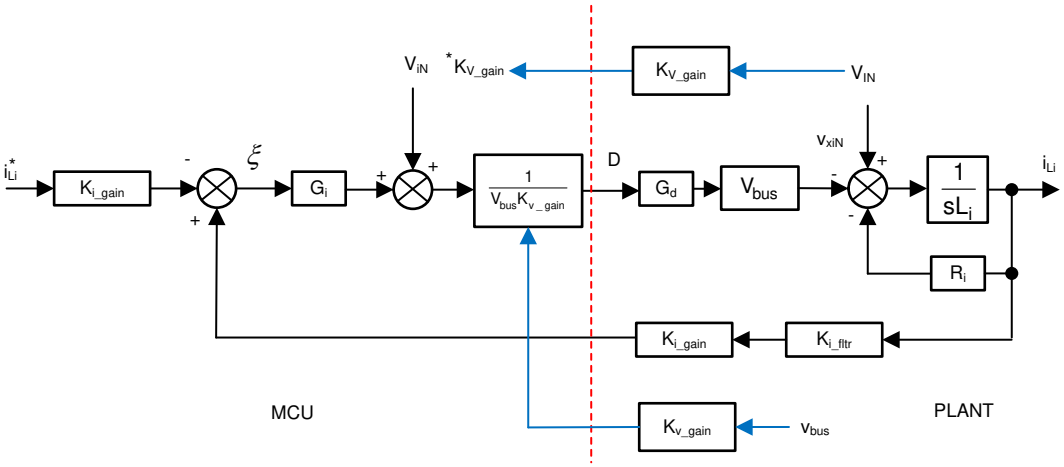


図 9. 電流ループ制御モデル

注: 基準の負記号は、電流ループが電圧 V_{xiN} を制御していると考えられるためです。電流を上げるには、V_{xiN} を下げる必要があり、このため図 9の基準と帰還では記号が逆になっています。

次に、この電流ループモデルを用いて電流補償器を設計します。電流ループには、単純な比例積分(PI)コントローラを使用します。

なお、三相インターリーブの場合は、各レッグに同じデューティ・サイクルが設定されるため、電流は単純に3倍になります。このため、プラントモデルは式 3のようになります。

$$H_{p\_i} = \frac{i_{Li}^{*}}{D} = 3 \times \frac{1}{K_{v\_gain}} \times K_{i\_gain} \times K_{i\_fltr} \times G_d \times \frac{1}{Z_i} \quad (3)$$

SFRAライブラリを使用して、このモデルをこのリファレンス・デザインで検証します。図 10はモデルと開ループ周波数応答の測定値との関係を示しており、そこに良好な相関関係があることが分かります。

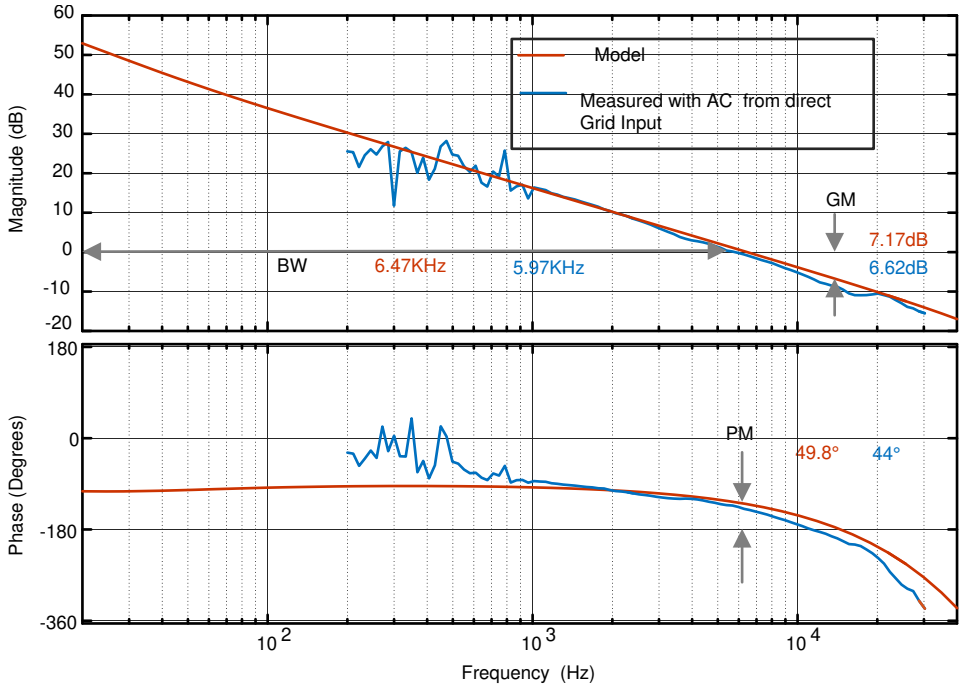


図 10. Gi、電流開ループ利得の測定値とモデルとの関係

2.4.3 DCバス電圧制御ループ

DCバス電圧制御ループは、基準電力を提供するものとされています。基準電力を入力電圧RMSの2乗で割って導電率を出し、さらに入力電圧を乗じて瞬時電流指令を求めます。

DCバス電圧制御ループの小信号モデルは、動作点周りで式 4を線形化して作成します。

$$i_{DC} v_{bus} = \eta V_{Nrms} i_{Nrms} \rightarrow \hat{i}_{DC} = \eta \frac{\overline{V}_{Nrms}}{\overline{V}_{bus}} i_{Li} \quad (4)$$

負荷抵抗については、式 5に示すようにバス電圧および電流が相関しています。

$$\hat{V}_{bus} = \frac{R_L}{1 + sR_L C_o} \hat{i}_{DC} \quad (5)$$

DC電圧ループ制御モデルは、図 11に示すように描写できます。Vbusフィードフォワードを追加して制御ループをバス電圧から独立させており、よってバス制御のプラントモデルは式 6のように表せます。

$$H_{p\_bus} = H_{load} \times \eta \times K_{i\_gain} \times K_{v\_gain} \times K_{v\_flt} \quad (6)$$

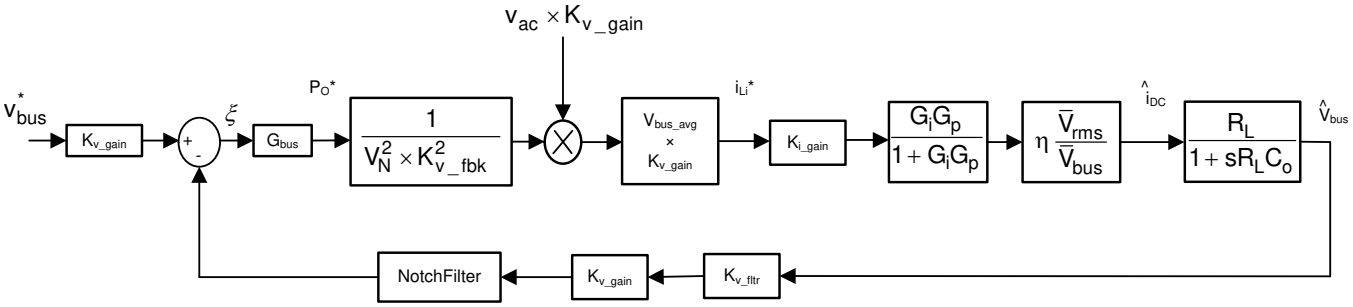


図 11. DC電圧ループ制御モデル

図 11を用いて、電圧ループ用に比例積分(PI)補償器を設計します。このループの帯域幅は、定常状態でTHDと衝突するため、狭く維持します。

SFRAライブラリを使用して、電圧ループの周波数応答を測定し、モデルを検証します。図 12に電圧ループのモデルと測定値のプロットを示します。

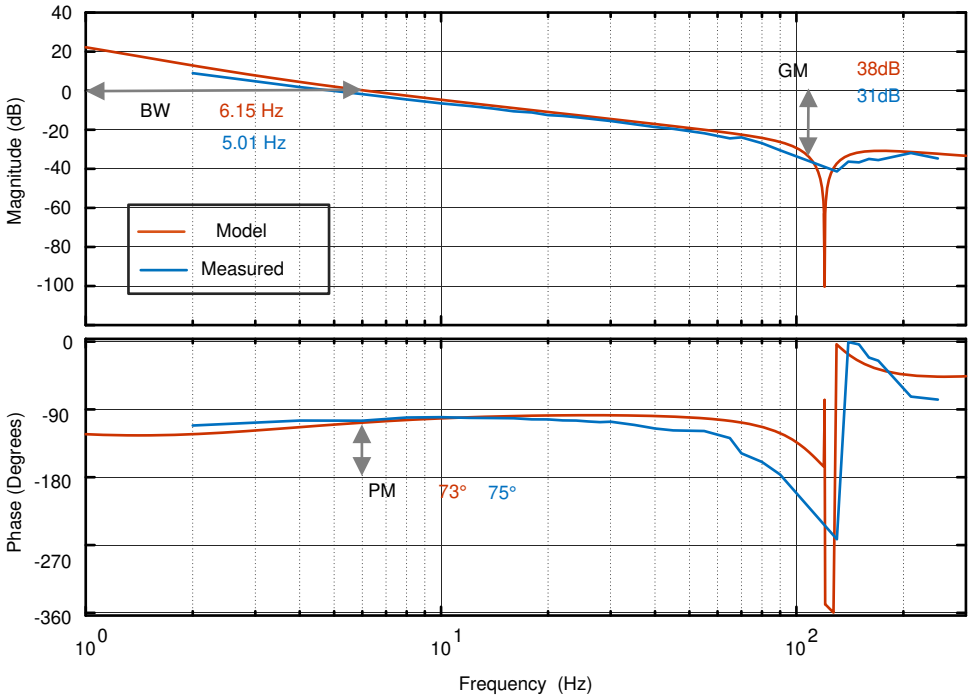


図 12. Gv、電圧ループのモデルと測定値との関係

2.4.4 ゼロクロス付近でのソフトスタートにより電流スパイクを解消または低減

ゼロクロス電流スパイクは、TTPL PFC方式における一つの課題です。ステートマシンによるソフトスタートの手法を実装して、一定のシーケンスでスイッチのオン/オフを切り替えることで、この問題を解決します。

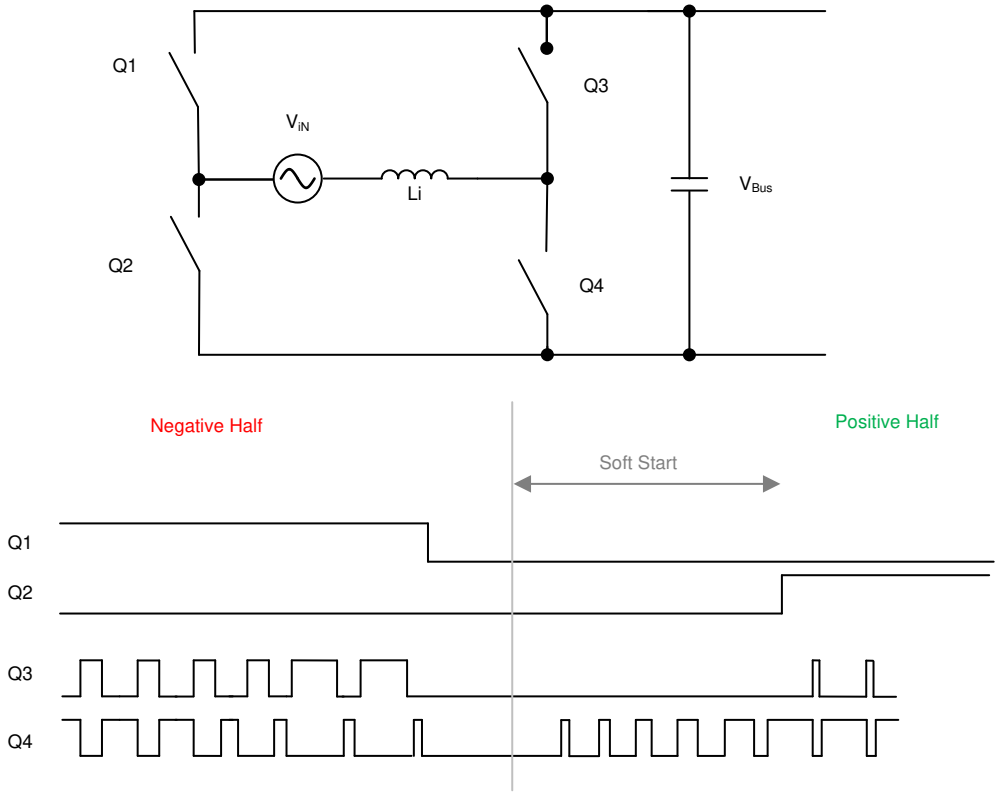


図 13. ソフトスタートによるPWMシーケンスでゼロクロス時の電流スパイクを低減

図 13 は、AC波が負から正に移行するときのスイッチング・シーケンスを示しています。負半波時にはQ1がオン、Q3がアクティブFET、Q4が同期FETとなります。この間、Q2を通した電圧はDCバス電圧となります。ACサイクルが変わると、Q2は100%またはほぼ100%オンになる必要があります。Q2がすぐにオンになると、非常に大きな正のスパイクが生じます。このため、図 13に示すようにソフトスタート・シーケンスを用いてQ4をオンにします。このソフトスタートの切り替えは、インダクタンス値およびその他の電力段パラメータ(デバイスのクロスなど)に依存します。

ゼロクロス付近で負の電流スパイクが生じるもう一つの理由は、ゼロクロス付近でAC電圧が比較的低くなることです。Q3がオンになると、デューティ・サイクルが低くても、高電圧差となり、高い負の電流スパイクが生じる可能性があります。このため、Q3が再びスイッチング動作を開始する前に、十分な遅延を要します。

また、ソフトスタートの開始後に、いくらかの遅延ののちQ2がオンになります。

3 ハードウェア/ソフトウェア/テスト要件とテスト結果

3.1 必要なハードウェアとソフトウェア

3.1.1 ハードウェア

本節では、ハードウェアならびに基板の個々の部位について詳述します。powerSUITEにより、このリファレンス・デザインのファームウェアを使用する場合には、本節を読み飛ばして構いません。

3.1.1.1 ベース基板の設定

このリファレンス・デザインはHSEC制御カードのコンセプトを採用しており、HSEC制御カードを利用できるC2000 MCU製品ファミリのデバイスであれば使用できる可能性があります。MCUの電力段の制御に使用される主要なリソースを表 2に示します。図 14に設計基板の主要な電力段とコネクタを示します。表 3に主要なコネクタとその機能を示します。まず、

1. リファレンス・デザインに電源が接続されていないことを確認します。
2. J600スロットに制御カードを挿入します。
3. 12V、1A DC電源をTP604に接続します。GND端子にはTP606を使用します。電源は投入しないでください。
4. 5V、1A DC電源をTP608に接続します。GND端子にはTP609を使用します。電源は投入しないでください。
5. 12V電源と5V電源の両方をオンにします。制御カードのLEDが点灯して、デバイスに電源が入ったことを示します。

注: MCUの電源は電力段と分離されているため、この一連の指示でシステムを安全に立ち上げることができます。

6. JTAGを接続するには、制御カードからUSBケーブルを使用してホストコンピュータに接続します。
7. 単相AC電源は入力J100に接続できます。オプションとして、一部の差分ビルドでは、システムを安全にテストするためにDC電源を必要とすることがあります。
8. 約500Ω、400Wの負荷抵抗をJ104の出力に接続します。
9. 図 14に示すように、電流/電圧プローブを接続して入力電流、入力電圧、出力電圧を観測できます。

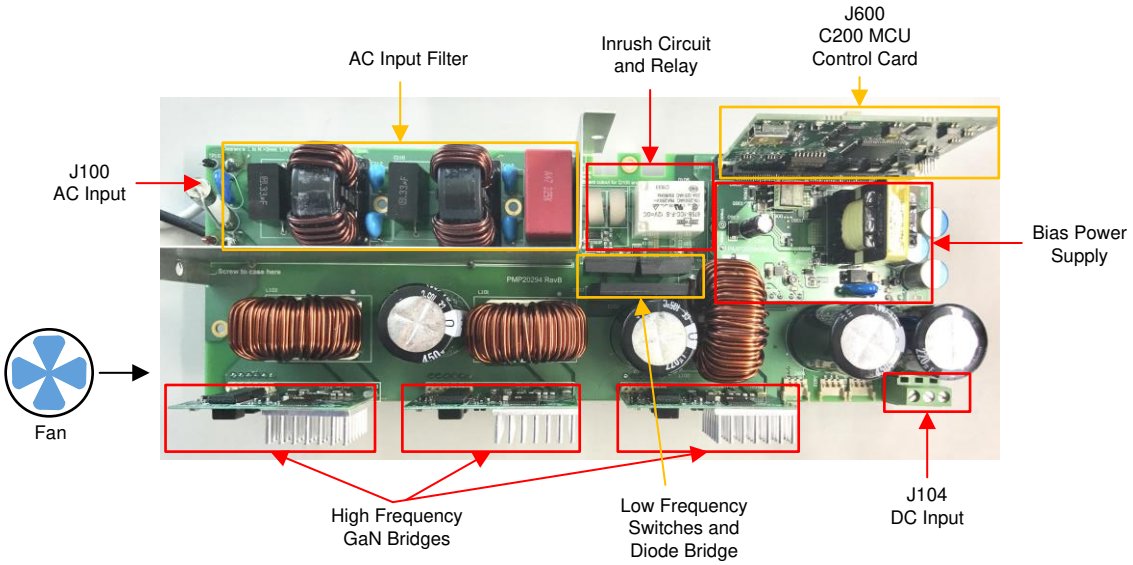


図 14. 基板の概要

表 2. 基板の電力段の制御に使用される主要なコントローラペリフェラル

信号名	HSECピン番号	機能
PWM-1A	49	PWM: 低周波MOSFETレッグ、1次側スイッチ
PWM-1B	51	PWM: 低周波MOSFETレッグ、2次側スイッチ
PWM-2A	53	PWM: 高周波GaNレッグ、1次側スイッチ、位相1
PWM-2B	55	PWM: 高周波GaNレッグ、2次側スイッチ、位相1
PWM-3A	50	PWM: 高周波GaNレッグ、1次側スイッチ、位相2
PWM-3B	52	PWM: 高周波GaNレッグ、2次側スイッチ、位相2
PWM-4A	54	PWM: 高周波GaNレッグ、1次側スイッチ、位相3
PWM-4B	56	PWM: 高周波GaNレッグ、2次側スイッチ、位相3
Iac	18	CMPSS付きADC: ACリターン電流測定
IL1	15	CMPSS付きADC: インダクタ電流測定Ph1
IL2	21	CMPSS付きADC: インダクタ電流測定Ph2
IL3	25	CMPSS付きADC: インダクタ電流測定Ph3
VL	20	ADC: ACライブ側電圧
VN	17	ADC: ACニュートラル側電圧
Vbus	24	ADC: バス電圧
突入リレー	57	GPIO: 突入リレーの制御に使用
GaNフォールト1	58	GPIO: GaNフォールト信号位相1
GaNフォールト2	60	GPIO: GaNフォールト信号位相2
GaNフォールト3	62	GPIO: GaNフォールト信号位相3
AC電流検知利得調整	63	GPIO: 利得段を制御

表 3. 主要なコネクタと機能

コネクタ名	機能
J100	入力AC電圧
J104	出力DCバス電圧
TP604	入力電源、12VDC、1A
TP608	入力電源、5VDC、1A
TP606/TP609	GND
J600	HSEC制御カードコネクタスロット

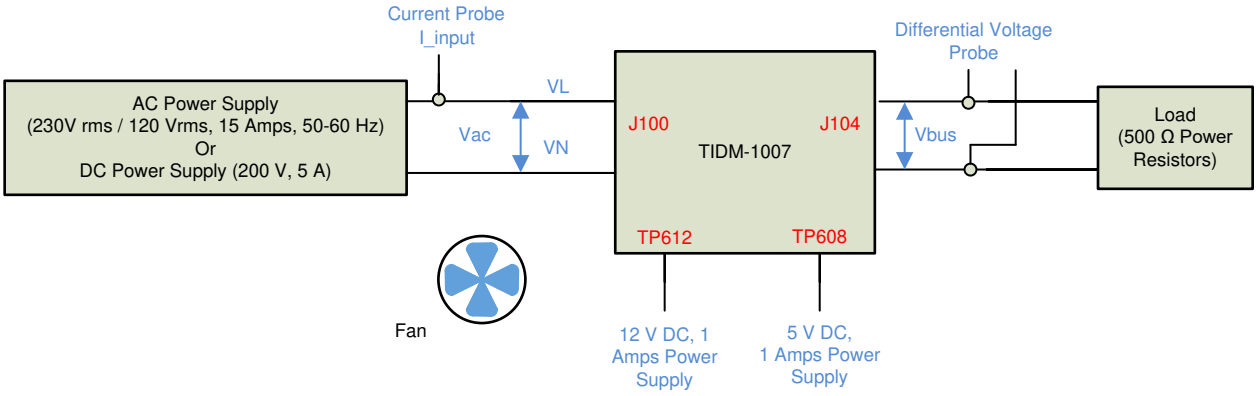


図 15. ソフトウェアを実行するためのハードウェアセットアップ

3.1.1.2 制御カードの設定

デバイスコントロールカードの設定には、JTAG経由の通信が必要であり、絶縁UARTボードを使います。また、適切なADC基準電圧も提供する必要があります。以下はF280049M制御カードのリビジョンAに必要な設定です。*<install_path>\c2000ware\boards\controlcards\TMDSCNCD280049M*のC2000Wareにある情報シートをご参照ください。また、http://www.ti.com/lit/ug/spruic4/spruic4.pdfのドキュメントでも情報が得られます。

1. JTAGのデバイスへの接続とSFRA GUI用のUART接続を確立するには、制御カードのS1:Aを両端で「ON ("上")」に設定する必要があります。このスイッチが「OFF ("下")」になっていると、制御カードに内蔵された絶縁JTAGの使用や、SFRA GUIでデバイスと通信することができません。
2. J1:Aは Code Composer Studio ™(CCS) が動作するホストPCからデバイスへの通信に使用するUSBケーブル用のコネクタです。
3. このリファレンス・デザインの制御ループ調整には3.3Vの基準電圧が必要です。F28004xの内部基準電圧を使用し、このためS8スイッチを左に移動する必要があります(VREFHIを指す)。
4. 制御カードの2つの絶縁GNDの間にコンデンサが接続されています(C26:A)。このリファレンス・デザインの性能を最大限に引き出すために、このコンデンサは外すことをお勧めします。

3.1.2 ソフトウェア

このリファレンス・デザインのソフトウェアはC2000Ware Digital Power SDKで提供されており、powerSUITEフレームワーク内でサポートされています。

3.1.2.1 CCSでのプロジェクトの開始

まず

1. *Code Composer Studio (CCS)統合開発環境(IDE)* ツールフォルダ(バージョン7.4以上を推奨)からCCSをインストールします。
2. CCSを開きます。*View → CCS App Center*を選択します。*Code Composer Studio Add-ons*下で、GUI Composer Runtime v1.0がインストールされていることを確認します。インストールされていなければ、GUI Composer Runtime v1.0をインストールします。
3. *C2000Ware Digital Power SDK* ツールフォルダのC2000Ware DigitalPower SDKをインストールします。
   - 注: デフォルトインストールでは、SDKとともにpowerSUITEがインストールされます。
4. CCSを閉じて、新しいワークスペースを開きます。CCSにより自動的にpowerSUITEが検出されます。変更を有効にするために、CCSの再起動を求められることがあります。
5. *View → Resource Explorer*を選択します。TI Resource Explorer下で、*C2000Ware DigitalPower SDK*を選択します。

リファレンス・デザイン・ソフトウェアをそのまま開くには(このリファレンス・デザインおよびハードウェアで動作していたようにファームウェアが開き、基板はこのリファレンス・デザインと全く同じものにする必要があり、powerSUITE GUIを使用してプロジェクトに変更を加えることはできません)

1. *C2000Ware DigitalPower SDK*下で、*Development Kits → CCM Totem Pole PFC TIDM-1007*を選択し、*Run <device> Project*をクリックします。
2. これでプロジェクトがインポートされ、開発キットまたは設計ページが表示されます。このページを使用して、このユーザーガイドやテストレポート、ハードウェア設計ファイルなど、リファレンス・デザインに関する情報をすべて参照できます。
3. *Run <device_name> Project*をクリックします。
4. この操作によりプロジェクトがワークスペース環境にインポートされ、図 16のようなGUIのcfgページが表示されます。
   - 注: このプロジェクトは開発キットおよびリファレンス・デザイン・ページからインポートされているため、GUIを使用して電力段に変更を加えることはできません。
5. このGUIページが表示されない場合は、*C2000Ware Digital Power SDK*リソースエクスプローラで*powerSUITE*下のFAQを参照してください。

アダプテーション用のリファレンス・デザイン・ソフトウェアを開きます。電力段パラメータを変更し、それを用いてCompensation Designerで電力段モデルを作成したり、カスタムデザイン用に電圧および電流のスケーリング値を変更することができます。

1. *C2000Ware Digital Power SDK*下で、*powerSUITE →Solution Adapter Tool* ()をクリックします。
2. 表示されたソリューションリストから*Single Phase CCM Totem Pole PFC*を選択します。
3. 次のページで、このソリューションを実行する必要があるデバイスを選択します。
4. アイコンをクリックすると、ポップアップウィンドウが表示され、プロジェクトを作成する場所を尋ねられます。ワークスペースそのものにプロジェクトを保存することもできます。場所を指定すると、プロジェクトが作成され、GUIページにそのソリューションの変更可能なオプションが表示されます(図 16)。
5. このGUIを使用して定格電力、インダクタンス、容量、検出回路パラメータなど、統合ソリューションのさまざまなパラメータを変更できます。
6. このGUIページが表示されない場合は、*C2000Ware DigitalPower SDK*リソースエクスプローラで*powerSUITE*下のFAQを参照してください。

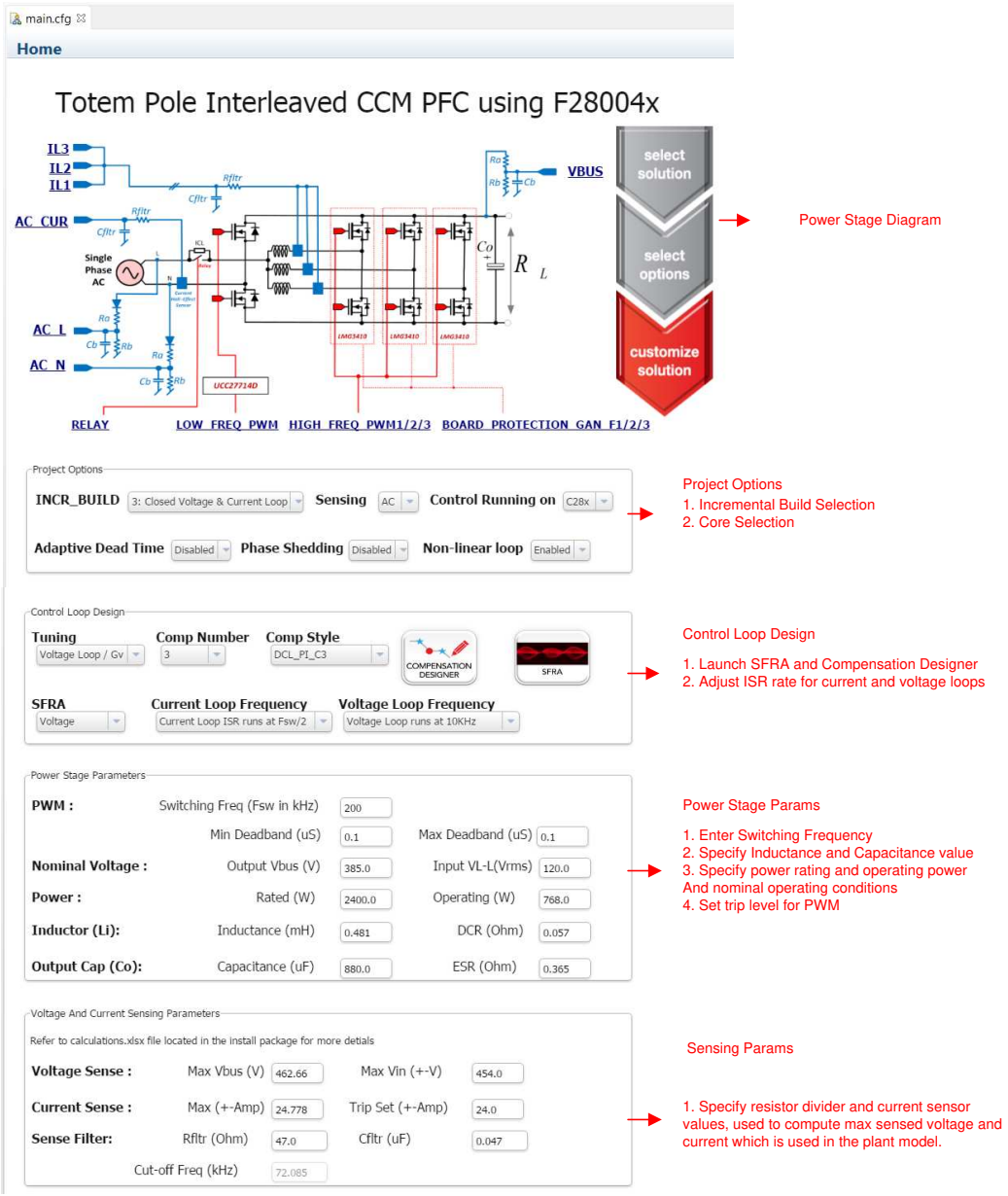


図 16. CCM TTPL PFCソリューションのpowerSUITEページ

3.1.2.2 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 17に示すようにCCS内にProject Exploreが表示されます。

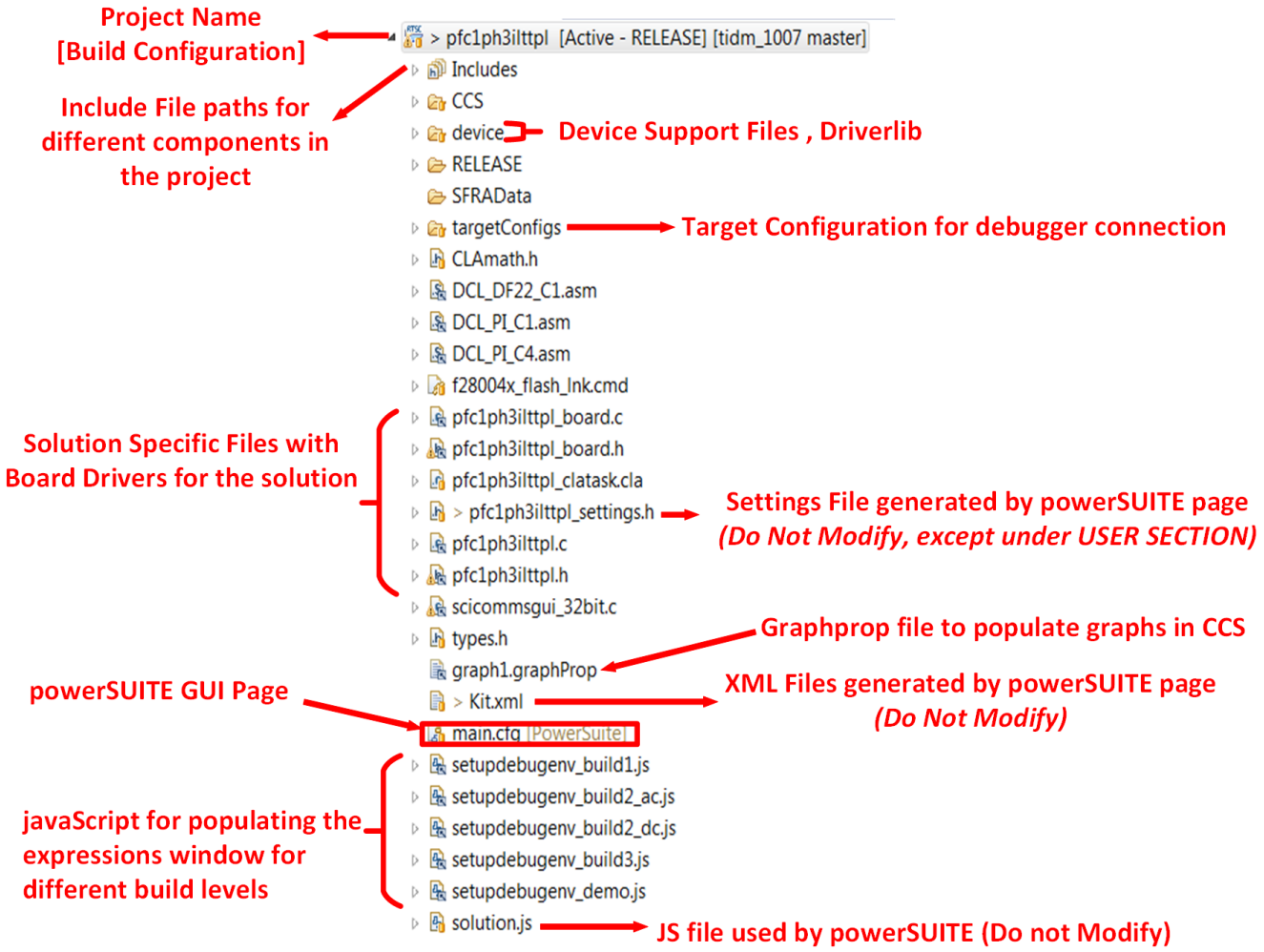


図 17. ソリューション・プロジェクトのProject Exploreビュー

プロジェクトの一般構造を図 18に示します。

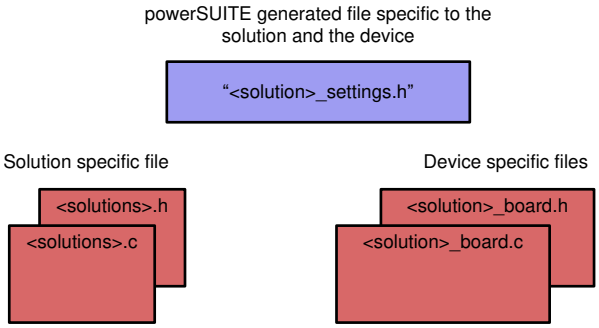


図 18. プロジェクト構造概要

注: 図 18 はF28004xのプロジェクトを示していますが、powerSUITEページから別のデバイスを選択しても、その構造は同様です。

ソリューション別でデバイスに依存しないファイルは*<solution>.c/h*です。このファイルはプロジェクトのmain.cファイルからなり、各ソリューションの制御構造に関与します。

このリファレンス・デザインの場合、<solution>は*pfc1ph3ilttpl*です。

基板別およびデバイス別のファイルは*<solution>_board.c/h*です。このファイルは、各ソリューションを実行するデバイス別ドライバで構成されています。

powerSUITEページは、Project Explorerに表示される*main.cfg*ファイルをクリックして開くことができます。powerSUITEページでは*<solution>_settings.h*ファイルが生成されます。このファイルは、powerSUITEページで生成されたプロジェクトのコンパイル時に使用する唯一のファイルです。プロジェクトが保存されるたびにpowerSUITEによって変更内容が上書きされるため、user section領域以外では、このファイルを手動で変更しないでください。

*Kit.xml*および*solution.js*ファイルも、powerSUITEにより内部で使用されるため、ユーザーが変更することはできません。これらのファイルを変更すると、プロジェクトが正常に機能しなくなります。

setupdebugenv_build.jsは、さまざまなビルドでWatchウィンドウの変数を自動入力するために提供されています。

*.graphPropファイルは、データロガーグラフの設定を自動入力するために提供されています。

プロジェクトは、PWMサイクルごとに呼び出される割り込みサービスルーチン(ISR)で構成されており、このISRの中で電流コントローラが実行されます。ほかにも電圧ループおよび計測を実行するために呼び出される、約10kHzの低速ISRがあります。少数のバックグラウンドタスク(A0-A4およびB0-B4)はポーリング方式で呼び出され、これを使用してSFRAバックグラウンドなど、絶対的なタイミング精度が要求されない低速タスクを実行できます。

図 19 にファームウェアのソフトウェア・フローチャートを示します。

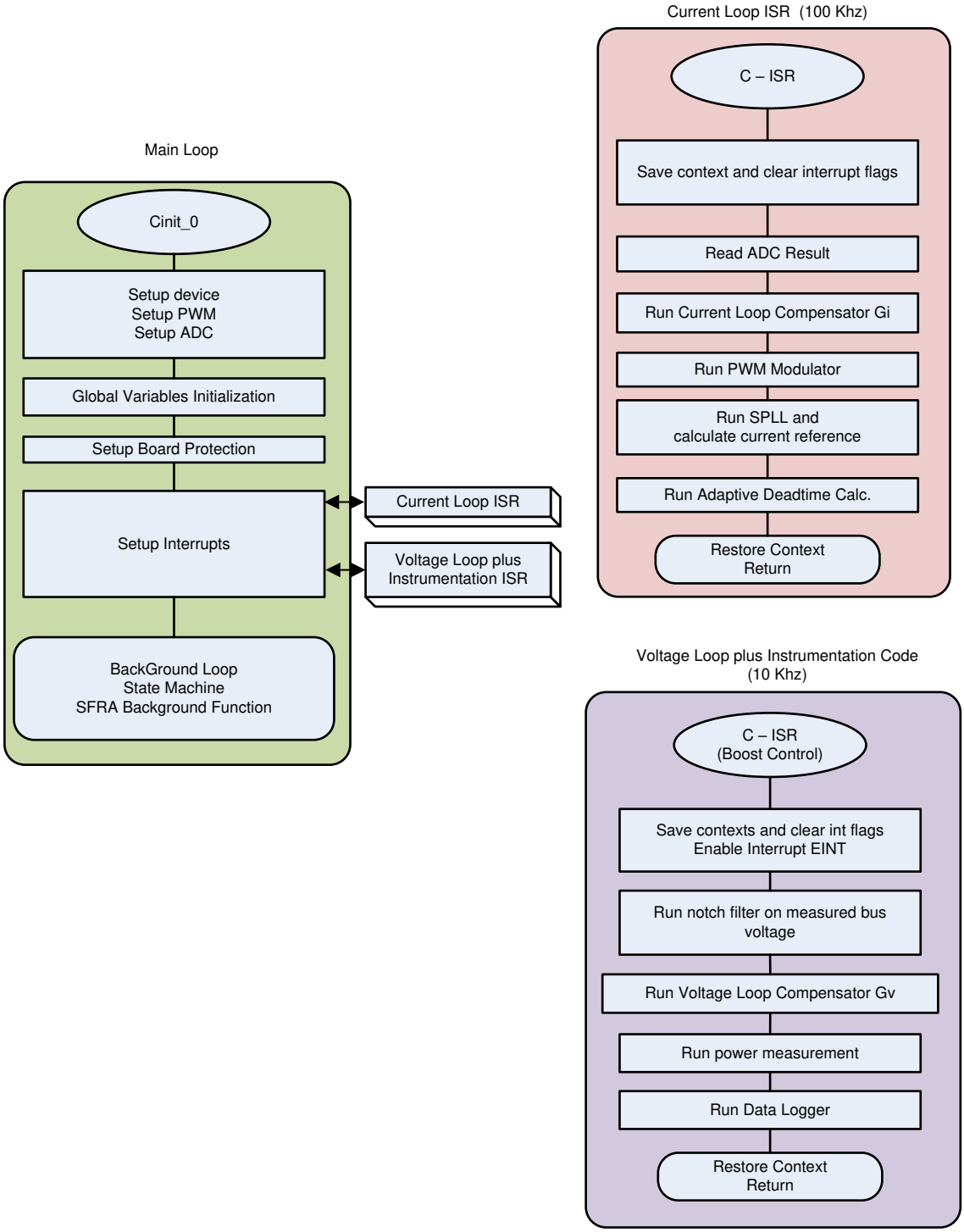


図 19. プロジェクト構造イメージ

システムの開発および設計を簡素化するために、このリファレンス・デザインのソフトウェアは、次の4つの差分ビルド(INCR_BUILD)で構成されています。

- INCR_BUILD 1: 開ループチェック、DC
- INCR_BUILD 2: 閉電流ループ: DC

- INCR_BUILD 2: 閉電流ループ: AC
- INCR_BUILD 3: 閉電圧/電流ループ

これらのビルドレベルについては 3.1.2.4に詳述します。リファレンス・デザイン・ハードウェアを使用する場合は、3.1.1で述べたようにハードウェアセットアップが完了していることを確認します。

3.1.2.3 C2000 MCUのCLAを使用してCPUの負荷を軽減

CLA (制御補償器アクセラレータ)は、C2000 MCUファミリの製品に搭載されているコプロセッサです。このコプロセッサにより、制御ISR機能をメインのC28x CPUコアからオフロードできます。

powerSUITE対応ソリューションにおいてCLAで制御ISRを実行するには、powerSUITE CFGページのドロップダウンメニューから選択します。powerSUITEソリューションのソフトウェア構造は、ドロップダウンメニューから選択するだけで、CLAにタスクがオフロードできるように設計されています。コードの重複はなく、コードをCLAとC28xのどちらで実行する場合でも1つのアルゴリズムソースで実行できます。このためソリューションを柔軟にデバッグできます。

各製品間でCLAの機能は若干異なります。例として、F2837xD、F2837xS、F2807xのCLAは、同時に1つのタスクしかサポートできず、ネスティングに対応していません。この構成では、タスクは割り込み不可能であり、CLAには1つのISRしかオフロードできません。F28004xのCLAはバックグラウンドタスクに対応しており、通常のCLAタスクへのネスティングが可能です。この構成では、CLAに2つのISRをオフロードできます。

CLAはバックグラウンドタスクに対応しており、CLAタスクへのネスティングが可能です。この構成では、CLAに2つのISRをオフロードできます。F28004xの場合には、電流ループ/電圧ループ用の制御ISR (100kHz)と計測ISR (10kHz)をCLAにオフロードします。F28004xのCPU使用率は、100kHzループ(フェーズ・シェディング、アダプティブデッドタイム、SFRA実行などの高度なオプションを含まない)で約42%、電圧ループと計測機能を実行する10kHzループで11%です。したがって、総CPU使用率は約53%になります。CLAを使用すれば、2つのISRをCLAにオフロードして、このCPU負荷をゼロにすることができます。

CLAの詳細については、CLAハンズオンワークショップおよび各製品のテクニカルリファレンスマニュアルを参照してください。

3.1.2.4 プロジェクトの実行

3.1.2.4.1 INCR_BUILD 1: 開ループ、DC

このビルドでは、固定デューティ・サイクルにより基板が開ループ方式で起動します。デューティ・サイクルはdutyPU_DC変数で制御されます。このビルドは、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。また、このビルドでは入出力電圧検出の較正も実行できます。このビルドのソフトウェア構造を図 20に示します。このシステムには、電流ループ用の高速ISR、電圧ループおよび計測機能を実行する低速ISRという2つのISRがあります。各ISRで実行するモジュールを図 20に示します。

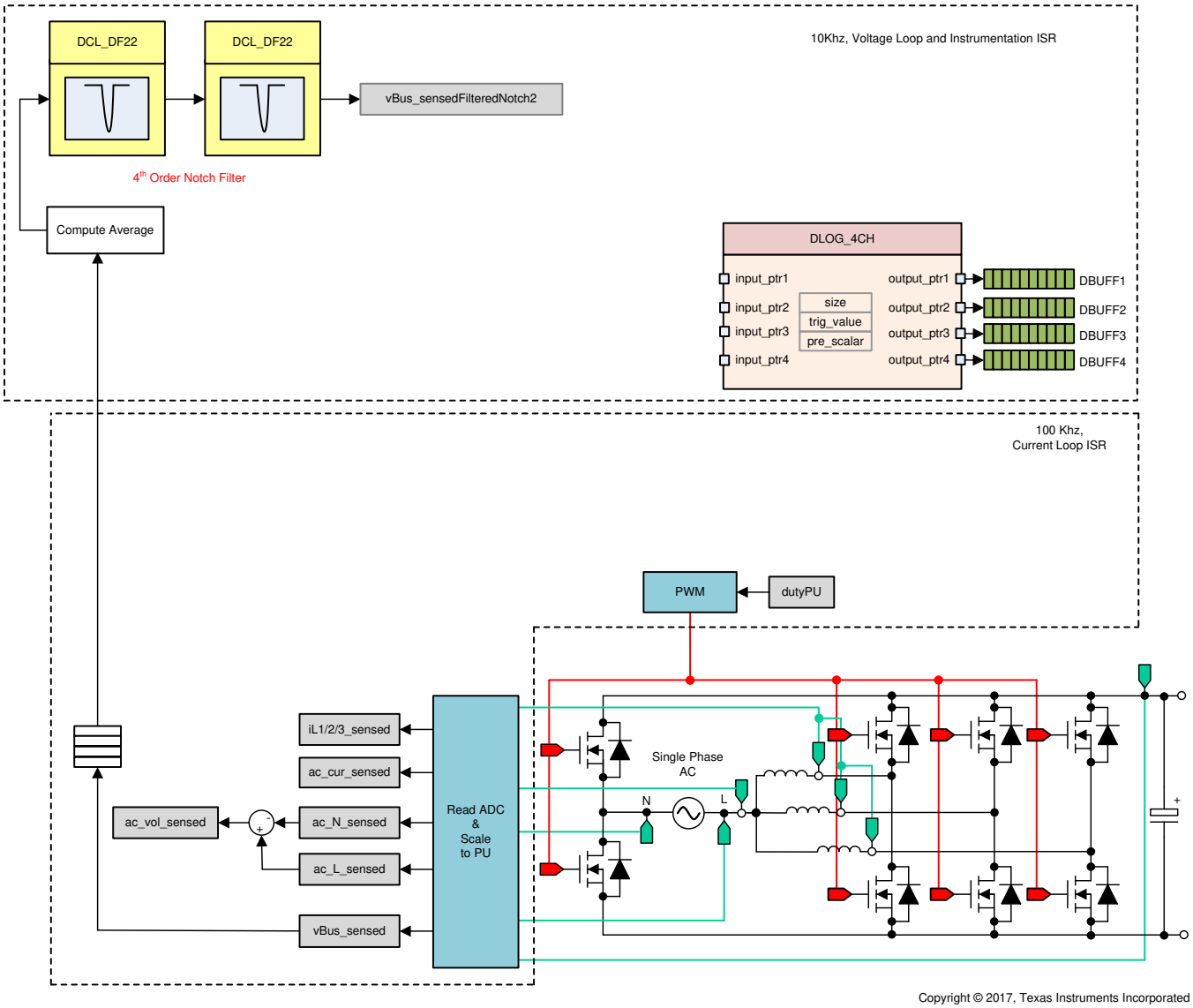


図 20. ビルドレベル1制御ソフトウェア構成図: 開ループプロジェクト

ハードウェアセットアップは、前述のものと同様とします。図 21にビルドレベル1テストのハードウェアセットアップを示します。

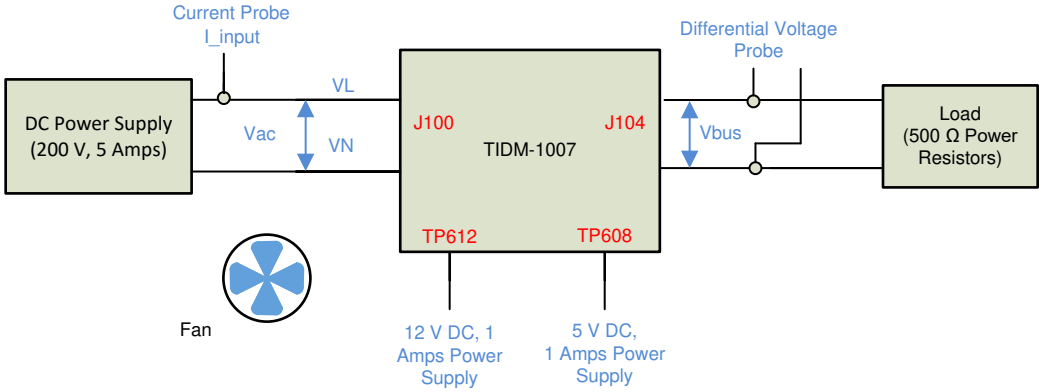


図 21. ビルドレベル1のハードウェアセットアップ

3.1.2.4.1.1 BUILD 1のソフトウェアオプションの設定

1. powerSUITEの設定: powerSUITEページの*Project Options*で次のものを選択します。
   - INCR_BUILDオプションでビルドレベルとして*Open Loop*を選択します。
   - INPUTオプションで入力をDCとします。
   - また、*Non Linear Voltage Loop*、*Adaptive Deadtime*、*Phase shedding*などの他のオプションを無効にします。
2. 適合ソリューションの場合には、*Voltage and Current Sensing Parameters*の設定を編集します。powerSUITEページの検出回路および最大範囲の計算については、*<install_location>\solutions\tidm_1007\hardware*のC2000Ware DigitalPower SDK Installディレクトリで入手可能な*calculations.xlsx*をご参照ください。
3. Power Stage Parametersで、スイッチング周波数、デッドバンド、定格電力を指定します。ページを保存します。

3.1.2.4.1.2 プロジェクトのビルドおよびロード

1. プロジェクト名を右クリックし、*Rebuild Project*をクリックします。
2. プロジェクトが正常にビルドされます。
3. *Project Explorer*で、targetconfigs下で適切な目標構成ファイルが有効になっていることを確認します(図 17)。
4. それから*Run → Debug*をクリックします。この操作によりデバッグセッションが起動します。デュアルCPUデバイスの場合には、ウィンドウが表示され、デバッグを実行する必要があるCPUを選択できます。ここでは、CPU1を選択します。
5. するとプロジェクトがデバイスにロードされ、CCSデバッグビューが有効になります。メインルーチンの開始時にコードは停止します。

3.1.2.4.1.3 デバッグ環境設定ウィンドウ

1. WatchおよびExpressionsウィンドウに変数を追加するには、*View → Scripting Console*をクリックして、Scripting Consoleダイアログボックスを開きます。このコンソールの右上隅で、Openをクリックして、プロジェクトフォルダ内にある*setupdebugenv_build1.js*スクリプトファイルを参照します。このスクリプトファイルにより、Watchウィンドウに、システムをデバッグするのに必要な適切な変数が入力されます。Watchウィンドウで Continuous Refreshボタンをクリックして、コントローラからの値の連続更新を有効にします。図 22に示すようにWatchウィンドウが表示されます。

Variables | Expressions | Registers | Breakpoints

Expression	Type	Value	Address
buildInfo	enum enum_BuildLevel	BuildLevel1_OpenLoop_DC	0x0000A817@Data
guiVbus	float	-0.000226858858	0x0000A87E@Data
guiVin	float	-0.714660585	0x0000A876@Data
guiIi	float	0.0891165435	0x0000A874@Data
ac_cur_sensed	float	0.00343942642	0x0000A8A4@Data
clearTrip	int	0	0x0000A827@Data
EPwm1Regs.TZFLG	Register	0x0000	
EPwm2Regs.TZFLG	Register	0x0000	
dutyPU	float	0.5	0x0000A858@Data
dutyPU_DC	float	0.5	0x0000A85E@Data
vBus_sensed	float	0.0	0x0000A8C0@Data
iL1_sensed	float	-0.00341796875	0x0000A8C4@Data
iL2_sensed	float	-0.00830078125	0x0000A8C6@Data
iL3_sensed	float	-0.00732421875	0x0000A8AA@Data
Add new expression			

図 22. ビルドレベル1のExpressionsビュー

2. をクリックして、プロジェクトを実行します。
3. ここでツールバーの*Halt*ボタン( )をクリックして、プロセッサを停止します。

3.1.2.4.1.4 リアルタイムエミュレーションの使用

リアルタイムエミュレーションは、MCUの動作中にCCS内のウィンドウの更新を可能にする、特別なエミュレーション機能です。この機能により、プロセッサを停止することなく、グラフおよびWatchビューが更新可能になるだけでなく、ユーザーがWatchウィンドウやMemoryウィンドウの値を変更して、その変更をシステムに反映できるようになります。

1. マウスポインタを水平ツールバーのボタンの上に置き、 ボタンをクリックして、リアルタイムモードを有効にします。

   Enable Silicon Real-time Mode (service critical interrupts when halted, allow debugger accesses while running)

2. メッセージボックスが表示されることがあります。その場合は*YES*を選択して、デバッグイベントを有効にします。この操作により、ステータスレジスタ1 (ST1)のビット1 (DGBMビット)が*0*に設定されます。DGBMは、デバッグイネーブルマスクビットです。DGBMビットが*0*に設定されると、メモリ値とレジスタ値がホストプロセッサに渡されて、デバッガのウィンドウが更新できるようになります。

3.1.2.4.1.5 コードの実行

1. ここで  をクリックして、もう一度プロジェクトを実行します。
2. 数秒後に突入リレーがオンになり、DCによりこのビルドレベルでそれを実行するようにソフトウェアがプログラミングされます。検出がクリアされ、デューティ・サイクル0.5が適用されます。
3. Watchビューで、guiVin、guiVbus、guiIi、変数が更新しているかどうか定期的にチェックします。
   - 注: 現時点では電力が印加されていないため、この値はゼロに近くなっています。
4. ここで入力DC電圧をゼロから120Vまで徐々に上げていきます。デフォルト設定で0.5PUの安定したデューティ・サイクルが適用されるため、出力電圧は昇圧を示します。大電流が引き出された場合、電圧端子が逆接続されていないかどうかを確認します。そうであれば、まず電圧をゼロに下げ、問題を修正してからテストを再開します。
5. 電圧検出の検証: *guiVin*および*guiVbus*に正しい値が表示されていることを確認します。120V DC入力の場合、guiVbusはほぼ240Vです。これにより、基板の電圧検出をある程度、検証できます。
6. 電流検出の検証: 所定のテスト条件での*guiIIi*を確認します。この値はほぼ1Aです。

Variables | Expressions | Registers | Breakpoints

Expression	Type	Value	Address
buildInfo	enum enum_BuildLevel	BuildLevel1_OpenLoop_DC	0x0000A817@Data
guiVbus	float	236.679321	0x0000A87E@Data
guiVin	float	117.730934	0x0000A876@Data
guiIi	float	1.01184416	0x0000A874@Data
ac_cur_sensed	float	0.0393730402	0x0000A8A4@Data
clearTrip	int	0	0x0000A827@Data
EPwm1Regs.TZFLG	Register	0x0000	
EPwm2Regs.TZFLG	Register	0x0000	
dutyPU	float	0.5	0x0000A858@Data
dutyPU_DC	float	0.5	0x0000A85E@Data
vBus_sensed	float	0.511474609	0x0000A8C0@Data
iL1_sensed	float	0.0161132813	0x0000A8C4@Data
iL2_sensed	float	0.01171875	0x0000A8C6@Data
iL3_sensed	float	0.0141601563	0x0000A8AA@Data
Add new expression			

図 23. ビルドレベル1: 電圧および電流の測定値を示すWatch Expression

7. これにより、PWMドライバおよびハードウェアの接続を基本的に検証でき、ユーザーはdutyPU_DCの変数を変更して、さまざまな昇圧条件での動作を確認できます。
8. 終了したら、入力電圧をゼロに下げ、バス電圧がゼロまで下がるのを見届けます。
9. これでこのビルドのチェックは完了し、このビルドが正常に終了した時点で次の項目を検証します。
   - 電圧および電流の検出とスケーリングが適正であること
   - 電流ループISRおよび電圧ループ計測ISRにおけるBUILD 1コードの割り込み生成および実行
   - PWMドライバおよびスイッチング

   問題が確認された場合には、ビルドの問題など解消するためにハードウェアを慎重に点検する必要があります。
10. 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。
11. リアルタイムモードのMCUを完全に停止するには、2段階の手順を踏みます。まず、ツールバーの*Halt*ボタン ()か*Target* → *Halt*をクリックしてプロセッサを停止します。次に、  をクリックしてMCUをリアルタイムモードから解除します。最後に、  をクリックしてMCUをリセットします。
12. *Terminate Debug Session* (*Target* → *Terminate all*)をクリックして、CCSデバッグセッションを終了します。 

3.1.2.4.2 INCR_BUILD 2: 閉電流ループDC

このBUILD 2では、内部電流ループが閉じているため、電流補償器Giを使用してインダクタ電流を制御します。DCバスと出力電圧フィードフォワードの両方をこの電流補償器の出力に印加して、インバータのデューティ・サイクルを生成します(式 7)。これにより、電流補償器用構成が簡素になり、比例(P)コントローラを使用して内部電流のループを調整できます。電流ループモデルは 2.4.2に示しています。このビルドのソフトウェア構成図を図 24に示します。

$$\text{duty1PU} = \frac{(\text{ac\_cur\_meas} - \text{ac\_cur\_ref\_inst}) \times \text{Gi} + \text{ac\_vol\_sensed}}{\text{vBus\_sensed}} \quad (7)$$

このビルドのソフトウェア構成図を図 24に示します。

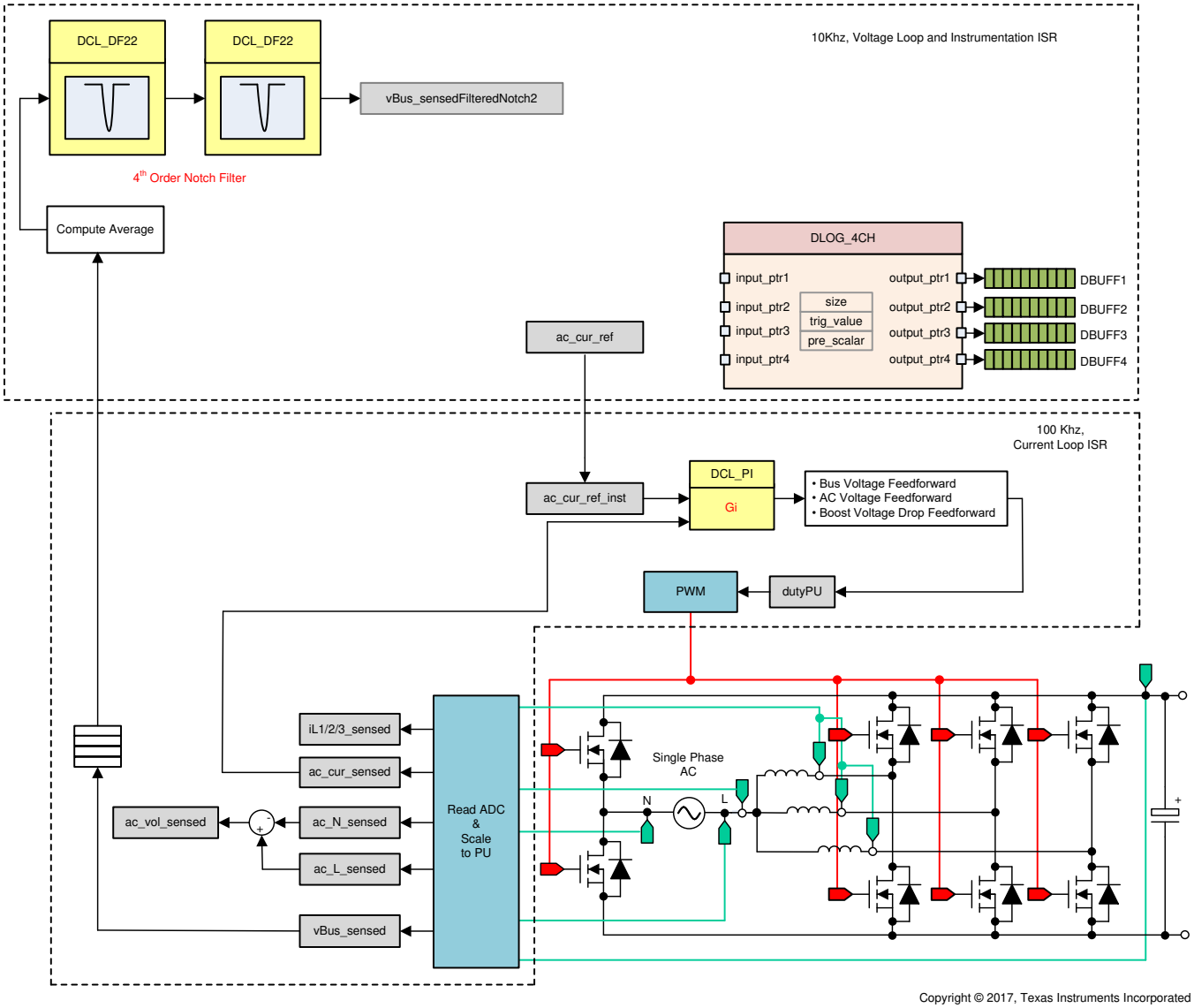


図 24. ビルドレベル2制御ソフトウェア構成図: 閉電流ループ

3.1.2.4.2.1 BUILD 2のソフトウェアオプションの設定

1. 図 21に示すようにハードウェアがセットアップされていることを確認します。高電圧(HV)電力はまだ基板に供給しないでください。
2. powerSUITEの設定: powerSUITEページの*Project Options*で次のものを選択します。
   - INCR_BUILDオプションでビルドレベルとして*Closed Current Loop*を選択します。
   - INPUTオプションで入力をDCとします。
   - また、*Non Linear Voltage Loop*、*Adaptive Deadtime*、*Phase shedding*などの他のオプションを無効にします。
3. その他のすべてのオプションは 3.1.2.4.1.1で指定したものと同じとします。
4. *Control Loop Design*で、電流ループ調整のオプションが自動的に選択されます(*Tuning* → *Current Loop* → *COMP1* → *DCL_PI_C1*)。ここで*Compensation Designer*アイコン( )をクリックします。

3.1.2.4.2.2 電流ループ補償器の設計

1. powerSUITEページで指定したパラメータの電流ループプラントモデルで、Compensation Designerが起動します。PIコントローラを極零の観点から調整して、安定した閉ループ動作を確保できます。設計した補償器を使用するシステムの安定性は、図 25に示すCompensation Designerの開ループ伝達関数プロットで利得マージンと位相マージンを観測して検証できます。

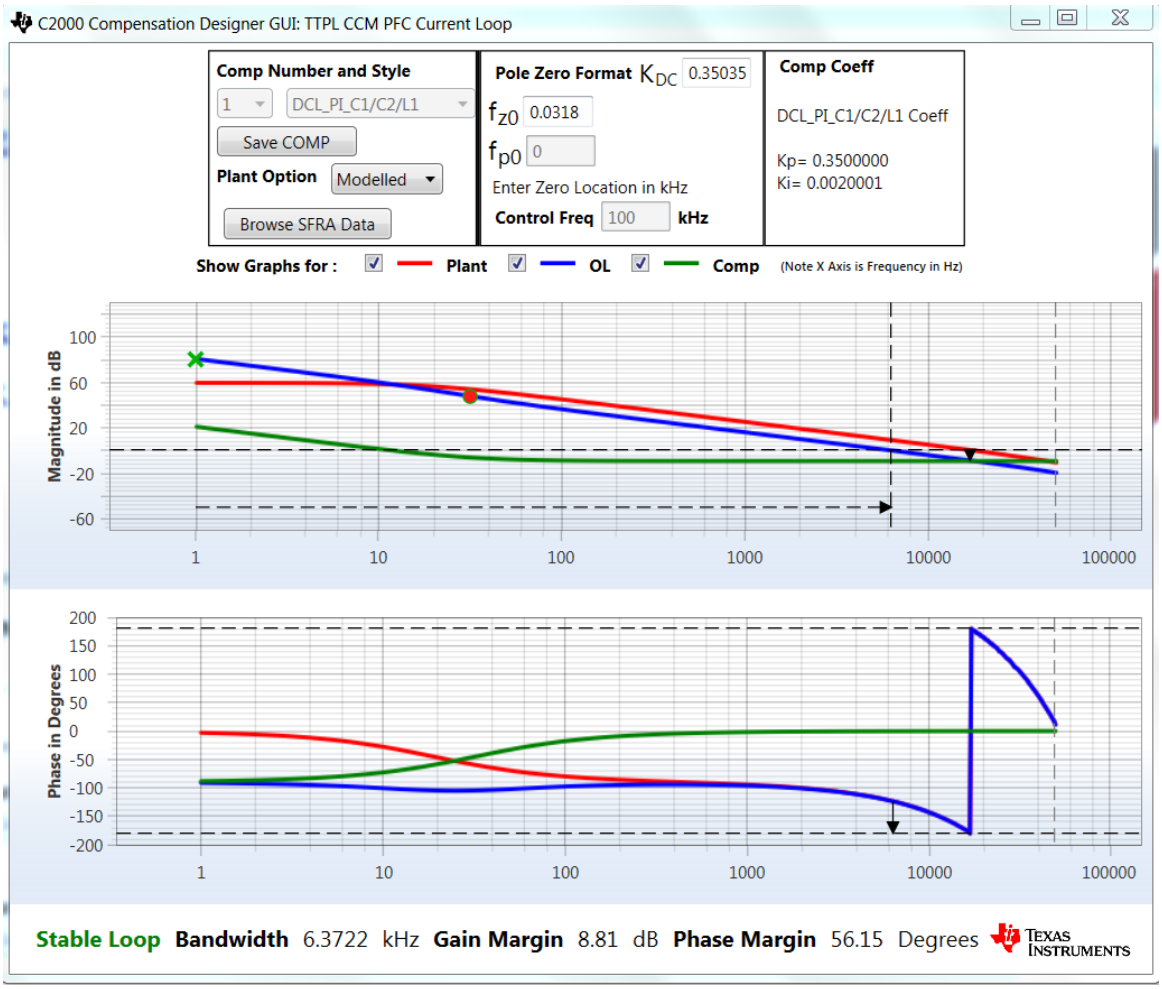


図 25. Compensation Designerを使用した電流ループ設計

2. 開ループ利得に満足したら、*Save COMP*をクリックします。この操作により、補償器の値がプロジェクトに保存されます。

- 注: そのプロジェクトがソリューションアダプタから選択されていない場合、補償器に変更を加えることはできません。ソリューションアダプタからソリューションを選択してください。

3. Compensation Designerを閉じて、powerSUITEページに戻ります。

3.1.2.4.2.3 プロジェクトのビルドおよびロードとデバッグの設定

1. プロジェクト名を右クリックし、*Rebuild Project*をクリックします。プロジェクトが正常にビルドされます。*Run → Debug*をクリックして、デバッグセッションを起動します。デュアルCPUデバイスの場合、ウィンドウが表示され、デバッグを実行する必要があるCPUを選択できます。ここでは、CPU1を選択します。するとプロジェクトがデバイスにロードされ、CCSデバッグビューが有効になります。メインルーチンの開始時にコードは停止します。
2. WatchおよびExpressionsウィンドウに変数を追加するには、*View → Scripting Console*をクリックして、Scripting Consoleダイアログボックスを開きます。このコンソールの右上隅で、*Open*をクリックして、プロジェクトフォルダ内にある*setupdebugenv_build2_dc.js*スクリプトファイルを参照します。このファイルにより、Watchウィンドウに、システムをデバッグするのに必要な適切な変数が入力されます。Watchウィンドウで*Continuous Refresh*ボタン( )をクリックして、コントローラからの値の連続更新を有効にします。図 26のようにWatchウィンドウが表示されます。

Variables | Expressions | Registers | Breakpoints

Expression	Type	Value	Address
buildInfo	enum enum_BuildLevel	BuildLevel2_CurrentLoop_DC	0x0000A80C@Data
boardStatus	enum enum_boardStatus	boardStatus_Idle	0x0000A804@Data
clearTrip	int	0	0x0000A824@Data
closeGiLoop	int	0	0x0000A819@Data
ac_cur_ref	float	0.0299999993	0x0000A838@Data
ac_cur_sensed	float	0.0118730068	0x0000A8A4@Data
guiVbus	float	0.347434014	0x0000A846@Data
guiVin	float	-0.678055584	0x0000A86C@Data
guiIi	float	0.274688691	0x0000A86A@Data
EPwm1Regs.TZFLG	Register	0x0004	
EPwm2Regs.TZFLG	Register	0x0004	
dutyPU	float	0.00999999978	0x0000A84E@Data
dutyPU_DC	float	0.5	0x0000A84C@Data
iL1_sensed	float	-0.00537109375	0x0000A8AE@Data
iL2_sensed	float	-0.00537109375	0x0000A8AC@Data
iL3_sensed	float	-0.00634765625	0x0000A8AA@Data
autoStartSlew	unsigned long	14	0x0000A87C@Data
Add new expression			

図 26. ビルドレベル2: 閉電流ループExpressionsビュー

3. マウスポインタを水平ツールバーのボタンの上に置き、 ボタンをクリックして、リアルタイムモードを有効にします。
4. をクリックしてプロジェクトを実行します。
5. ここでツールバーの*Halt*ボタン( )をクリックして、プロセッサを停止します。

3.1.2.4.2.4 コードの実行

1. このプロジェクトは、設定時間(autoStartSlew==100)を過ぎたら突入リレーを駆動し、検出をクリアするようにプログラミングされています。DCによりこのビルドレベルでそれを実行するようにソフトウェアがプログラミングされます。Runボタンをクリックしてからこのautoslewカウンタが100に達するまでに、入力電圧を印加する必要があります。入力時に電圧を印加する前にカウンタが100に達した場合は、コードをリセットする必要があります。このためコントローラをリアルタイムモードから解除する必要があり、リセットを実行し、再起動します。からの手順を繰り返します。
2. ここで をクリックしてプロジェクトを実行します。

3. autoStartSlewが100に達する前に、約50Vの入力電圧を印加します。autoStartSlewが100に達するとすぐに、突入リレーが駆動され、PWM検出がクリアされるのと同時に電流ループフラグを閉じます。

Variables | Expressions | Registers | Breakpoints

Expression	Type	Value	Address
buildInfo	enum enum_BuildLevel	BuildLevel2_CurrentLoop_DC	0x0000A80C@Data
boardStatus	enum enum_boardStatus	boardStatus_Idle	0x0000A804@Data
clearTrip	int	1	0x0000A824@Data
closeGiLoop	int	1	0x0000A819@Data
ac_cur_ref	float	0.0299999993	0x0000A838@Data
ac_cur_sensed	float	0.0300658941	0x0000A8A4@Data
guiVbus	float	127.377548	0x0000A846@Data
guiVin	float	48.3203316	0x0000A86C@Data
guiIi	float	0.707000256	0x0000A86A@Data
EPwm1Regs.TZFLG	Register	0x0000	
EPwm2Regs.TZFLG	Register	0x0000	
dutyPU	float	0.386497527	0x0000A84E@Data
dutyPU_DC	float	0.5	0x0000A84C@Data
iL1_sensed	float	0.0107421875	0x0000A8AE@Data
iL2_sensed	float	0.0087890625	0x0000A8AC@Data
iL3_sensed	float	0.009765625	0x0000A8AA@Data
autoStartSlew	unsigned long	101	0x0000A87C@Data
Add new expression			

図 27. 閉電流ループ動作が開始した後のWatch Expression、ビルドレベル2、DC

4. 入力電流は約0.7Aに制御され、出力電圧は約128Vに上昇します。
5. ここでac_cur_refを0.1すなわち2.5A入力まで徐々に上げていきます。
6. 次にVin＝120Vを徐々に上げていくと、出力電圧が350Vを上回ります。

Variables | Expressions | Registers | Breakpoints

Expression	Type	Value	Address
buildInfo	enum enum_BuildLevel	BuildLevel2_CurrentLoop_DC	0x0000A80C@Data
boardStatus	enum enum_boardStatus	boardStatus_Idle	0x0000A804@Data
clearTrip	int	1	0x0000A824@Data
closeGiLoop	int	1	0x0000A819@Data
ac_cur_ref	float	0.100000001	0x0000A838@Data
ac_cur_sensed	float	0.0993705988	0x0000A8A4@Data
guiVbus	float	380.123596	0x0000A846@Data
guiVin	float	117.478661	0x0000A86C@Data
guiIi	float	2.46380639	0x0000A86A@Data
EPwm1Regs.TZFLG	Register	0x0000	
EPwm2Regs.TZFLG	Register	0x0000	
dutyPU	float	0.308701962	0x0000A84E@Data
dutyPU_DC	float	0.5	0x0000A84C@Data
iL1_sensed	float	0.0493164063	0x0000A8AE@Data
iL2_sensed	float	0.052734375	0x0000A8AC@Data
iL3_sensed	float	0.0458984375	0x0000A8AA@Data
autoStartSlew	unsigned long	101	0x0000A87C@Data
Add new expression			

図 28. フル電圧で閉電流ループ動作が開始した後のWatch Expression、ビルドレベル2、DC

7. このビルドのソフトウェアにはSFRAが統合されているため、ハードウェアを測定して、設計した補償器が十分な利得マージンと位相マージンを提供していることを検証できます。SFRAを実行するには、プロジェクトを実行している状態で、cfgページから*SFRA*アイコンをクリックします。SFRA GUIが表示されます。
8. SFRA GUIでデバイスのオプションを選択します。例として、F28004xの場合には浮動小数点を選択します。*Setup Connection*をクリックします。ポップアップウィンドウでBoot on Connectオプションのチェックを外し、適切なCOMポートを選択します。*Boot on Connect*が選択されていないことを確認します。*OK*をクリックします。SFRA GUIに戻り、*Connect*をクリックします。

9. SFRA GUIがデバイスに接続します。これで*Start Sweep*をクリックして、SFRA掃引を開始できます。SFRA掃引が完了するまでには数分かかります。SFRA GUIのプログレスバーを確認したり、UARTの動作を示す制御カード裏面の青色LEDの点滅をチェックすることで、動作を監視できます。終了すると、開ループプロットによるグラフが表示されます。これを測定プロットと比較すると、図 29に示すようにモデルと測定値の間に良好な相関関係があります。これにより、設計した補償器が確かに安定しており、モデルが正確であることを検証できます。注: 200Hzを下回る低周波時の偏差が予想され、既知の現象とされており、ここに示す測定値はDC電源で取得したものです。AC電源を使用してDC電源をエミュレートした場合、AC電源の出力インピーダンスは以下にプロットしたグラフとの偏差を生じる可能性があります。

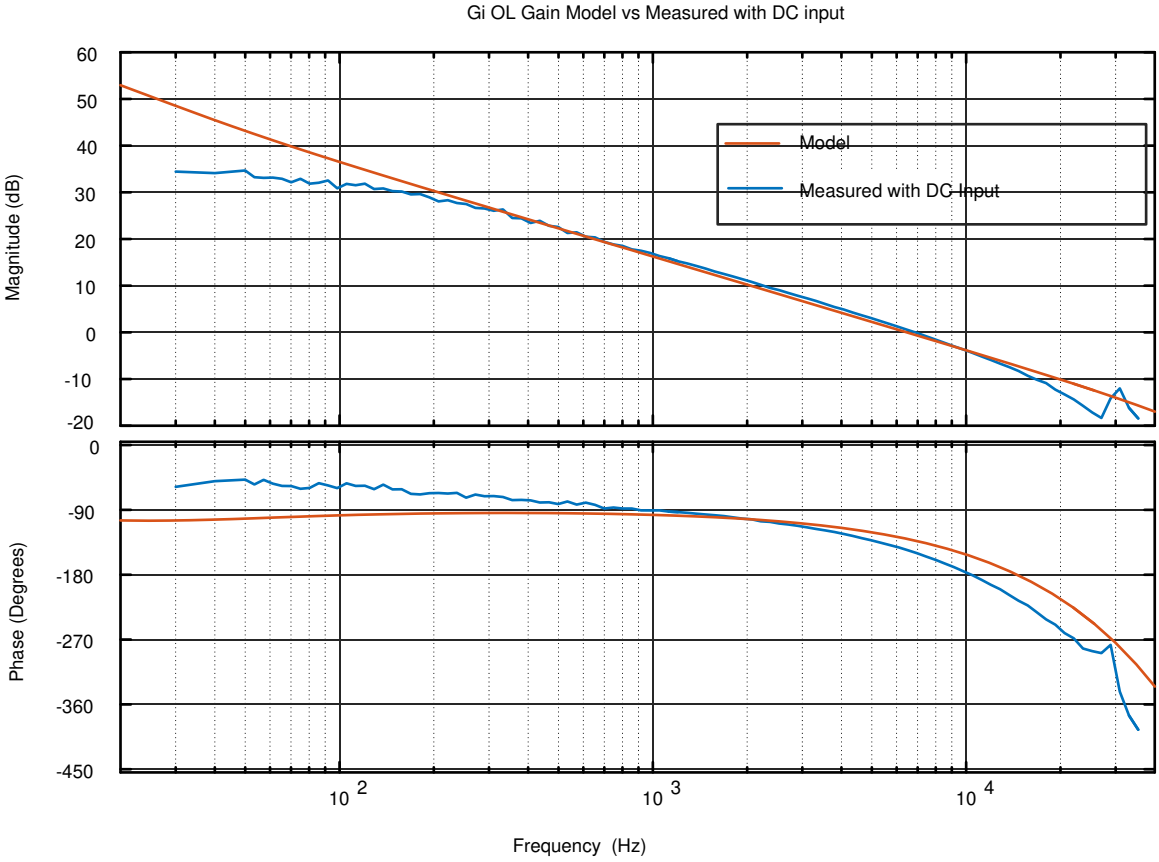


図 29. SFRA実行と閉電流ループ、開ループ利得モデルとの関係

また、周波数応答データはSFRAデータフォルダ下のプロジェクトフォルダに保存され、SFRA実行時のタイムスタンプが記録されます。

10. オプションとして、CFGページから再度Compensation Designerをクリックし、GUIでプラントオプションに*SFRA Data*を選択することで、プラントの周波数応答の測定値を用いて電流補償器を設計できます。これは、測定したプラントの情報を用いて補償器を設計するものです。このオプションを使用して補償を微調整できます。デフォルトでは、Compensation Designerは最新のSFRA実行を示しています。過去のSFRA実行プラント情報を使用する必要がある場合、ユーザーは*Browse SFRA Data*をクリックし、参照してSFRAData.csvファイルを選択できます。
11. この操作により、電流補償器設計を検証できます。
12. システムを安全に停止させるには、入力DC電圧をゼロまで下げ、guiVbusもゼロに下がっていることを確認します。
13. リアルタイムモードのMCUを完全に停止するには、2段階の手順を踏みます。まず、ツールバーの*Halt*ボタン( )か*Target → Halt*をクリックしてプロセッサを停止します。次に、 をクリックしてMCUをリアルタイムモードから解除します。最後に、MCUをリセットします( )。
14. *Terminate Debug Session* (*Target → Terminate all*)をクリックして、CCSデバッグセッションを終了します。

3.1.2.4.3 INCR_BUILD 2: 閉電流ループ、AC

このBUILD 2では、内部電流ループが閉じているため、電流補償器Giを使用してインダクタ電流を制御します。DCバスと出力電圧フィードフォワードの両方をこの電流補償器の出力に印加して、ゼロクロス周りのPWMソフトスタートとともにインバータのデューティ・サイクルを生成します。

このビルドのソフトウェア構成図を図 30に示します。

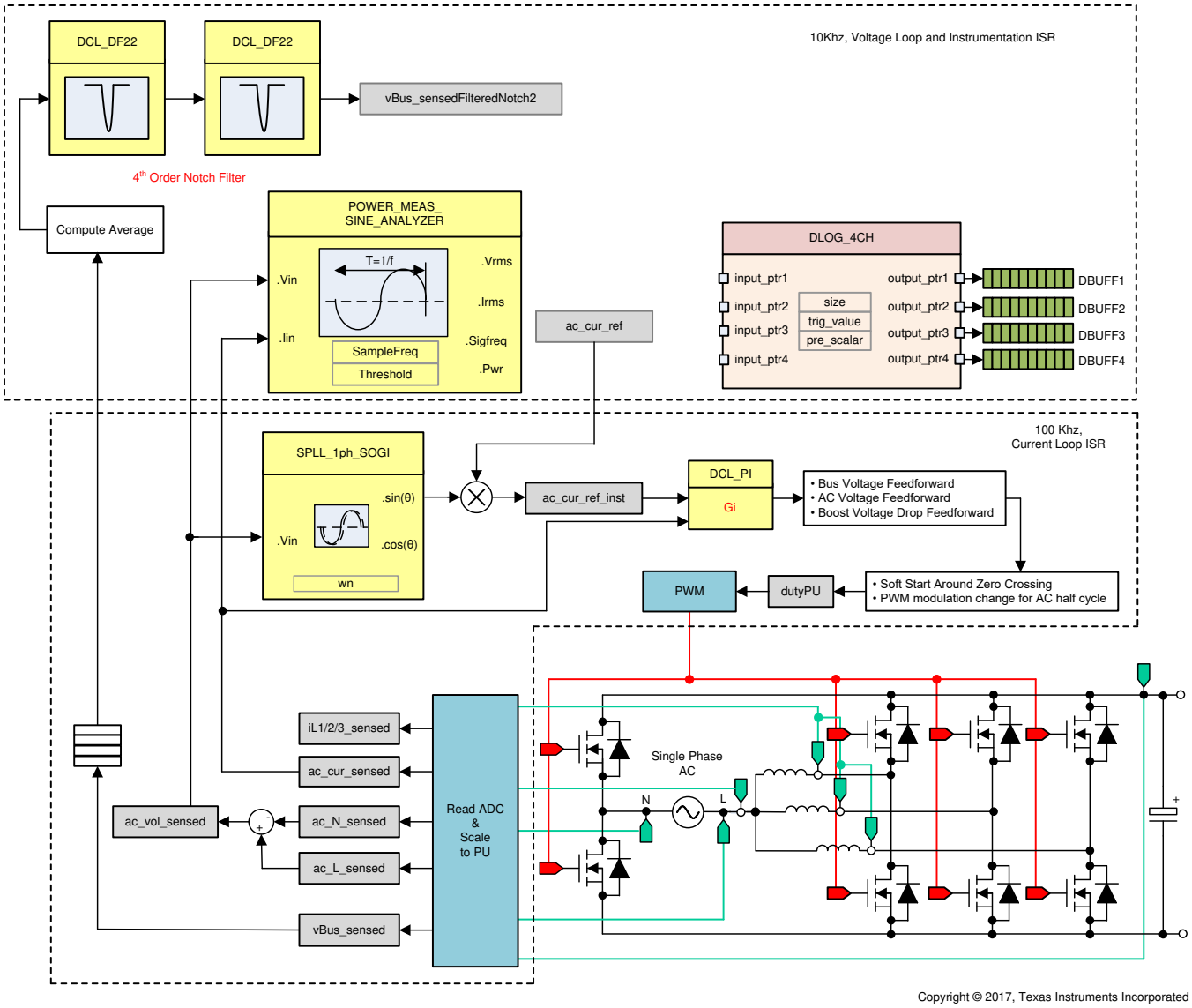


図 30. ビルドレベル2制御ソフトウェア構成図: 閉電流ループAC

このビルドレベルを実行するには、図 31に示すようにハードウェアがセットアップされていることを確認します。高電圧(HV)電力はまだ基板に供給しないでください。

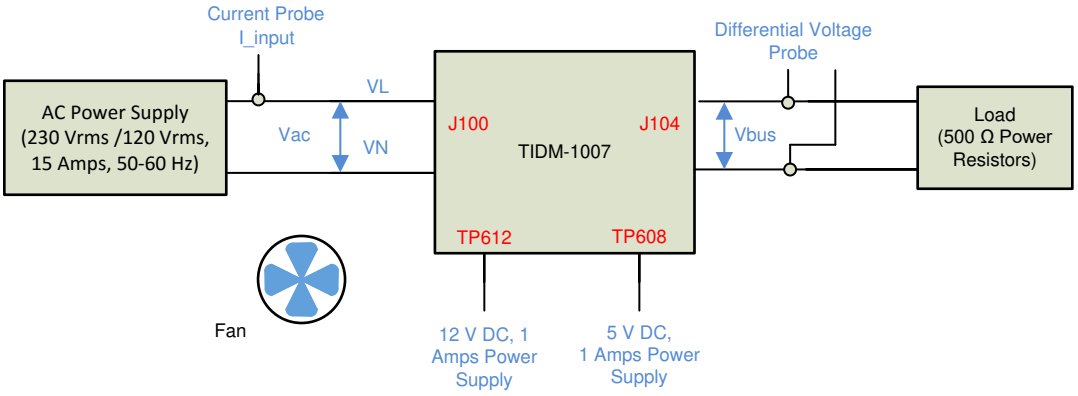


図 31. AC入力のハードウェアセットアップ

3.1.2.4.3.1 BUILD 2のソフトウェアオプションの設定

1. powerSUITEの設定: *powerSUITE*ページで、*Project Options*にで述べたオプションが選択されていると仮定し、ビルドレベルとして*Closed Current Loop*とAC入力を選択します。ページを保存します。
2. 過去のビルドの電流補償器をこのビルドで再利用するため、ビルドの電流ループを調整する追加手順は必要ありません。

3.1.2.4.3.2 プロジェクトのビルドおよびロードとデバッグのセットアップ

1. プロジェクト名を右クリックし、*Rebuild Project*をクリックします。プロジェクトが正常にビルドされます。*Run → Debug*をクリックして、デバッグセッションを起動します。デュアルCPUデバイスの場合、ウィンドウが表示され、デバッグを実行する必要があるCPUを選択できます。ここでは、CPU1を選択します。するとプロジェクトがデバイスにロードされ、CCSデバッグビューが有効になります。メインルーチンの開始時にコードは停止します。
2. WatchおよびExpressionsウィンドウに変数を追加するには、*View → Scripting Console*をクリックして、Scripting Consoleダイアログボックスを開きます。このコンソールの右上隅で、*Open*をクリックして、プロジェクトフォルダ内にある*setupdebugenv_build2_ac.js*スクリプトファイルを参照します。このファイルにより、Watchウィンドウに、システムをデバッグするのに必要な適切な変数が入力されます。Watchウィンドウで*Continuous Refresh*ボタン( )をクリックして、コントローラからの値の連続更新を有効にします。のようにWatchウィンドウが表示されます。

Variables | Expressions | Registers | Breakpoints

Expression	Type	Value	Address
buildInfo	enum enum_BuildLevel	BuildLevel2_CurrentLoop_AC	0x0000A806@Data
pwmSwState	enum enum_pwmSwState	pwmSwState_defaultState	0x0000A824@Data
boardStatus	enum enum_boardStatus	boardStatus_InputUnderVoltageTrip	0x0000A814@Data
clearTrip	int	0	0x0000A809@Data
closeGiLoop	int	0	0x0000A807@Data
ac_cur_ref	float	0.0299999993	0x0000A840@Data
ac_cur_sensed	float	0.010755837	0x0000A8A8@Data
guiVbus	float	0.344914794	0x0000A874@Data
guiVin	float	-0.563325524	0x0000A882@Data
guiVrms	float	0.0	0x0000A872@Data
guiIrms	float	0.0	0x0000A878@Data
guiPrms	float	0.0	0x0000A86E@Data
guiFreqAvg	float	0.0	0x0000A880@Data
guiPowerFactor	float	0.0	0x0000A87C@Data
EPwm1Regs.TZFLG	Register	0x0004	
EPwm2Regs.TZFLG	Register	0x0004	
dutyPU	float	0.00999999978	0x0000A86C@Data
dutyPU_DC	float	0.5	0x0000A862@Data
iL1_sensed	float	-0.00634765625	0x0000A8C4@Data
iL2_sensed	float	-0.00830078125	0x0000A8BE@Data
iL3_sensed	float	-0.0112304688	0x0000A8C0@Data
autoStartSlew	unsigned long	0	0x0000A85E@Data
Add new expression			

図 32. ビルドレベル2 AC: 閉電流ループExpressionsビュー

3. マウスポインタを水平ツールバーのボタンの上に置き、 ボタンをクリックして、リアルタイムモードを有効にします。

3.1.2.4.3.3 コードの実行

1. このプロジェクトは、入力電圧が約70Vrmsを上回るまで待機してから突入リレーを駆動し、検出をクリアするようにプログラミングされています。
2.  をクリックしてプロジェクトを実行します。
3. ここで約120Vの入力電圧を印加します。基板は低電圧状態から脱し、突入リレーが駆動されます。検出がクリアされ、約0.55A RMSの少量の電流が引き出されます。Watchウィンドウの外観は図 33と同様です。バス電圧はほぼ180Vです。

Variables | Expressions | Registers | Breakpoints

Expression	Type	Value	Address
buildInfo	enum enum_BuildLevel	BuildLevel2_CurrentLoop_AC	0x0000A806@Data
pwmSwState	enum enum_pwmSwState	pwmSwState_positiveHalf	0x0000A824@Data
boardStatus	enum enum_boardStatus	boardStatus_NoFault	0x0000A814@Data
clearTrip	int	1	0x0000A809@Data
closeGiLoop	int	1	0x0000A807@Data
ac_cur_ref	float	0.0299999993	0x0000A840@Data
ac_cur_sensed	float	-0.00663924217	0x0000A8A8@Data
guiVbus	float	180.061981	0x0000A874@Data
guiVin	float	-49.6501122	0x0000A882@Data
guiVrms	float	117.459831	0x0000A872@Data
guiIrms	float	0.551513135	0x0000A878@Data
guiPrms	float	64.2371902	0x0000A86E@Data
guiFreqAvg	float	59.8999023	0x0000A880@Data
guiPowerFactor	float	0.978407621	0x0000A87C@Data
EPwm1Regs.TZFLG	Register	0x0000	
EPwm2Regs.TZFLG	Register	0x0000	
dutyPU	float	-0.880984187	0x0000A86C@Data
dutyPU_DC	float	0.5	0x0000A862@Data
iL1_sensed	float	0.0180664063	0x0000A8C4@Data
iL2_sensed	float	-0.0048828125	0x0000A8BE@Data
iL3_sensed	float	-0.0283203125	0x0000A8C0@Data
autoStartSlew	unsigned long	5	0x0000A85E@Data
Add new expression			

図 33. 閉電流ループ動作が開始した後のWatch Expression、ビルドレベル2、AC

4. ここでac_cur_refを0.14すなわち2.4A入力まで徐々に上げていくと、バス電圧が380Vまで上昇します。電圧および電流波形を図 34に示します。

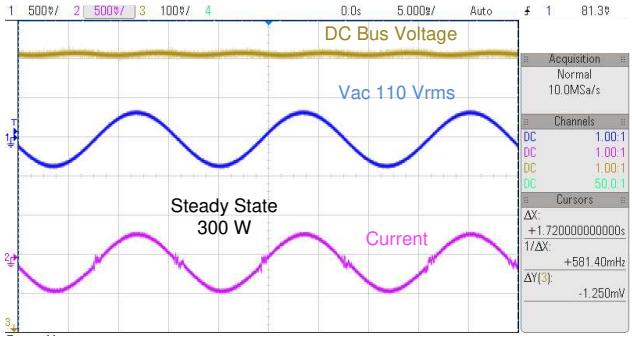


図 34. 入力AC電圧、電流および出力DC電圧の波形

5. このビルドのソフトウェアにはSFRAが統合されているため、ハードウェアを測定して、設計した補償器が十分な利得マージンと位相マージンを提供していることを検証できます。SFRAを実行するには、プロジェクトを実行している状態で、cfgページから*SFRA*アイコンをクリックします。SFRA GUIが表示されます。
6. SFRA GUIでデバイスのオプションを選択します。例として、F28377Dの場合には浮動小数点を選択します。*Setup Connection*をクリックします。ポップアップウィンドウでBoot on Connectオプションのチェックを外し、適切なCOMポートを選択します。*OK*をクリックします。SFRA GUIに戻り、*Connect*をクリックします。

7. SFRA GUIがデバイスに接続します。これで*Start Sweep*をクリックして、SFRA掃引を開始できます。SFRA掃引が完了するまでには数分かかります。SFRA GUIのプログレスバーを確認したり、UARTの動作を示す制御カード裏面の青色LEDの点滅をチェックすることで、動作を監視できます。終了すると、開ループプロットによるグラフが表示されます(図 35)。これはDC条件下のプロットと似ていますが、測定周波数付近にAC高調波による若干の追加ノイズが確認されます。BW、PM、GMの数値はDCの場合と非常に似通っています。図 35に示すグラフは、ダイレクトグリッドAC入力により取得しました。AC電源を使用したAC電源出力時には、インピーダンスが観測され、制御利得に影響する可能性があります。

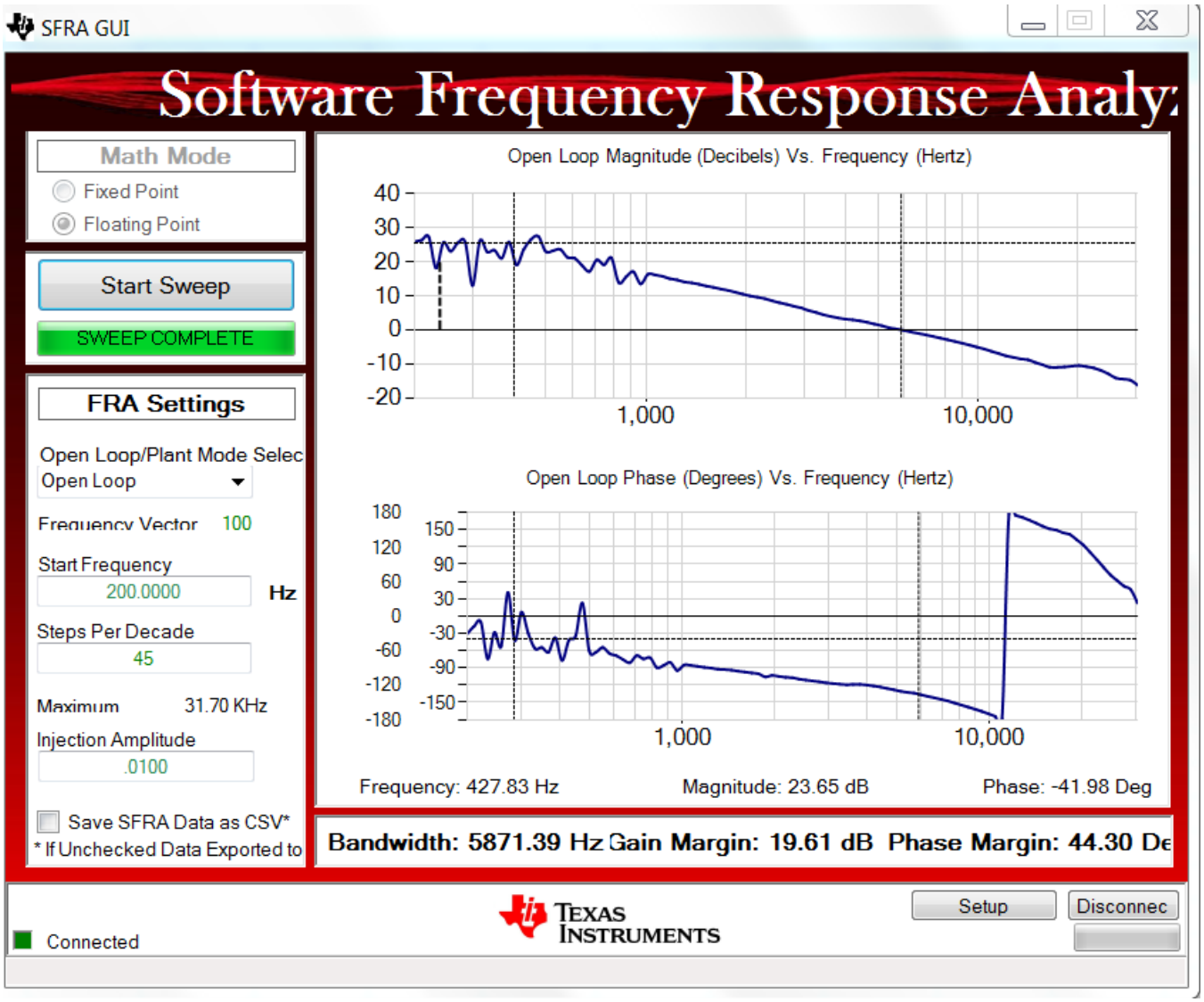


図 35. SFRA実行、閉電流ループ、開ループ利得

8. システムを安全に停止させるには、AC電源からの出力をオフにすることにより、入力AC電圧をゼロまで下げ、guiVbusもゼロに下がっていることを確認します。
9. リアルタイムモードのMCUを完全に停止するには、2段階の手順を踏みます。まず、ツールバーの*Halt*ボタン( )か*Target → Halt*をクリックしてプロセッサを停止します。次に、 をクリックしてMCUをリアルタイムモードから解除します。最後に、MCUをリセットします( )。
10. *Terminate Debug Session* (*Target → Terminate all*)をクリックして、CCSデバッグセッションを終了します。

3.1.2.4.4 INCR_BUILD 3: 閉電圧/電流ループ

このビルドでは、外部電圧ループも内部電流ループも閉じています。外部電圧ループのモデルは 2.4.3で導き出されています。PI補償器を使用して、外部電圧ループをCompensation Designerで調整します。

図 36 にこのビルドのソフトウェア構成図を示します。

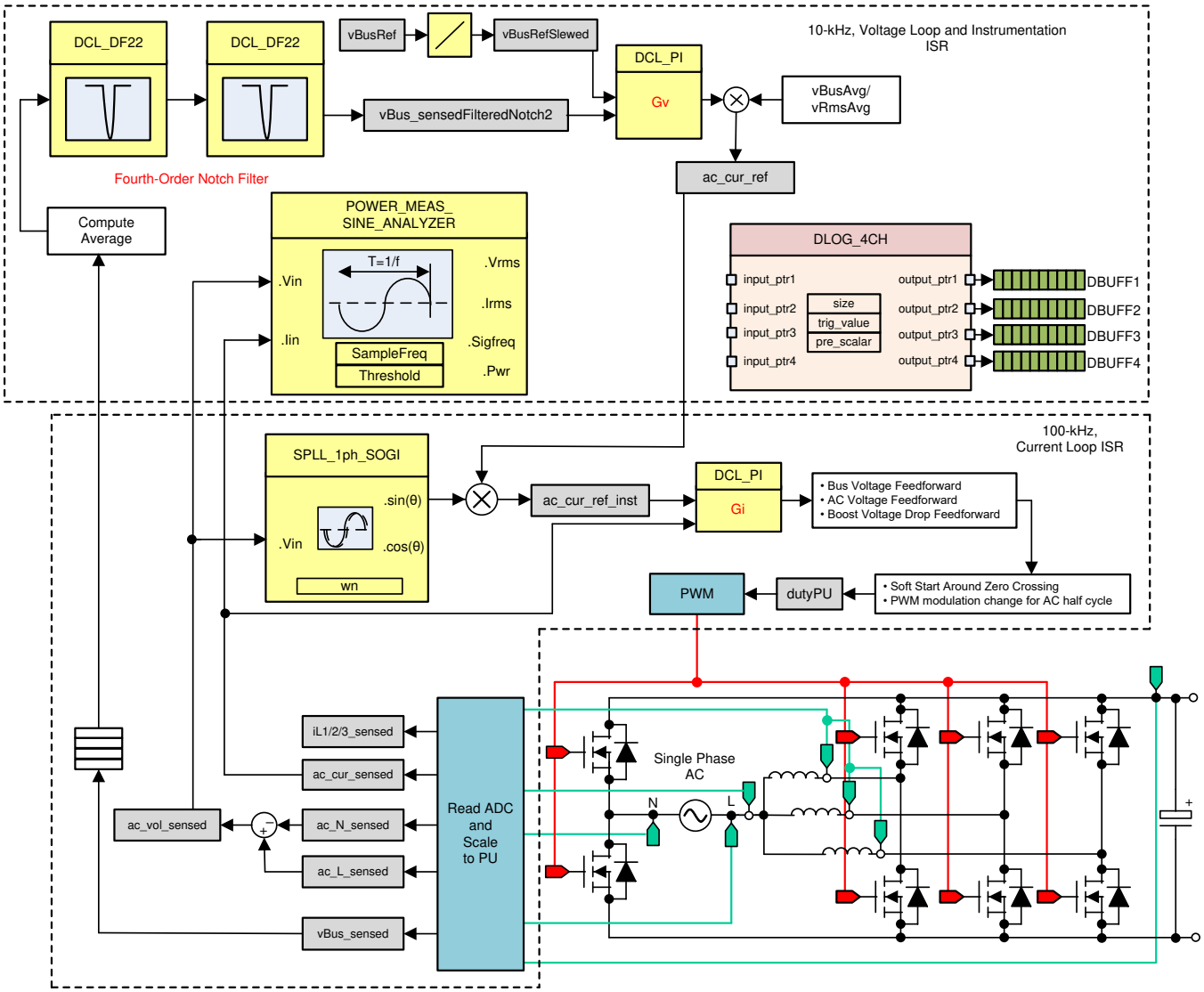


図 36. ビルドレベル3制御図: 内部電流ループによる出力電圧制御

3.1.2.4.4.1 BUILD 3のソフトウェアオプションの設定

1. 図 31に示すようにハードウェアがセットアップされていることを確認します。高電圧(HV)電力はまだ基板に供給しないでください。
2. powerSUITEの設定: powerSUITEページの*Project Options*で次のものを選択します。
   - INCR_BUILDオプションでビルドレベルとして*Closed Voltage & Current Loop*を選択します。
   - INPUTオプションで入力にACを選択します。
   - また、*Non Linear Voltage Loop*、*Adaptive Deadtime*、*Phase shedding*などの他のオプションを無効にします。
3. その他のすべてのオプションは 3.1.2.4.1.1で指定したものと同じとします。
4. *Control Loop Design*で、*Tuning*を*Voltage Loop*とします。Styleは*DCL PI*にプリセットされています。*Ctrl*＋*S*

を押してページを保存し、Compensation Designerボタン( )をクリックします。

5. 基板の出力に接続している負荷がpowerSUITE cfgページに正しく入力されていることを確認します。これは、この負荷の値を電圧補償器の設計に用いるためです。

3.1.2.4.4.2 電圧ループ補償器の設計

1. Compensation Designerが、図 37に示す電圧ループプラントモデルを起動します。希望する利得マージンと位相マージンが得られるようにPI補償器を編集できますが、電圧ループの帯域幅と実現するTHDとの間には逆相関関係があることに注意してください。通常、PFCシステムでは、この帯域幅は約10Hzに維持されます。

図 37. Compensation Designerを使用した電圧ループPI補償調整

2. 補償器設計に満足したら、*Save COMP*をクリックします。この操作により、補償器の値がプロジェクトに保存されます。
   - 注: そのプロジェクトがソリューションアダプタから選択されていない場合、補償器に変更を加えることはできません。個々の設計では、ソリューションアダプタからソリューションを選択してください。
3. Compensation Designerを閉じて、powerSUITEページに戻ります。*Ctrl*＋*S*を押して保存します。

3.1.2.4.4.3 プロジェクトのビルドおよびロードとデバッグのセットアップ

1. プロジェクト名を右クリックし、*Rebuild Project*をクリックします。プロジェクトが正常にビルドされます。*Run → Debug*をクリックして、デバッグセッションを起動します。デュアルCPUデバイスの場合、ウィンドウが表示され、デバッグを実行する必要があるCPUを選択できます。ここでは、CPU1を選択します。するとプロジェクトがデバイスにロードされ、CCSデバッグビューが有効になります。メインルーチンの開始時にコードは停止します。
2. WatchおよびExpressionsウィンドウに変数を追加するには、*View → Scripting Console*をクリックして、Scripting Consoleダイアログボックスを開きます。このコンソールの右上隅で、*Open*をクリックして、プロジェクトフォルダ内にある*setupdebugenv_build3.js*スクリプトファイルを参照します。このファイルにより、Watchウィンドウに、システムをデバッグするのに必要な適切な変数が入力されます。Watchウィンドウで*Continuous Refresh*ボタン( )をクリックして、コントローラからの値の連続更新を有効にします。
図 38に示すようなWatchウィンドウが表示されます。

Variables | Expressions | Registers

Expression	Type	Value	Address
buildInfo	enum enum_BuildLevel	BuildLevel3_VoltageLoop_AC	0x0000A811@Data
pwmSwState	enum enum_pwmSwState	pwmSwState_defaultState	0x0000A826@Data
boardStatus	enum enum_boardStatus	boardStatus_InputUnderVoltageTrip	0x0000A80F@Data
clearTrip	int	0	0x0000A81A@Data
closeGvLoop	int	0	0x0000A819@Data
vBusRef	float	0.821337461	0x0000A850@Data
vBus_sensed	float	0.000651041686	0x0000A8C0@Data
closeGiLoop	int	0	0x0000A81C@Data
ac_cur_senseOffset	float	0.502499998	0x0000A888@Data
guiVbus	float	0.353723764	0x0000A858@Data
guiVin	float	0.375192761	0x0000A82C@Data
guiVrms	float	0.0	0x0000A842@Data
guiIrms	float	0.0	0x0000A832@Data
guiPrms	float	0.0	0x0000A834@Data
guiPowerFactor	float	0.0	0x0000A82E@Data
guiFreqAvg	float	0.0	0x0000A844@Data
EPwm1Regs.TZFLG	Register	0x0004	
EPwm2Regs.TZFLG	Register	0x0004	
dutyPU	float	0.00999999978	0x0000A86E@Data
dutyPU_DC	float	0.5	0x0000A86C@Data
iL1_sensed	float	-0.00390625	0x0000A884@Data
iL2_sensed	float	-0.00634765625	0x0000A880@Data
iL3_sensed	float	-0.00244140625	0x0000A882@Data
autoStartSlew	unsigned long	0	0x0000A83E@Data
Add new expression			

図 38. ビルドレベル3: Expressionsビュー

3. マウスポインタを水平ツールバーのボタンの上に置き、 ボタンをクリックして、リアルタイムモードを有効にします。
4. をクリックしてプロジェクトを実行します。
5. ここでツールバーの*Halt*ボタン( )をクリックして、プロセッサを停止します。

3.1.2.4.4.4 コードの実行

1. このプロジェクトは、入力電圧が約70Vrmsを上回るまで待機してから突入リレーを駆動し、検出をクリアするようにプログラミングされています。
2.  をクリックしてプロジェクトを実行します。
3. ここで約120Vの入力電圧を印加します。基板は低電圧状態から脱し、突入リレーが駆動されます。検出がクリアされ、出力が380V DCまで上昇します。AC入力から正弦波電流が引き出されます。この段でプログラムを実行しているときのWatchウィンドウを図 39に示します。

Variables | Expressions | Registers

Expression	Type	Value	Address
buildInfo	enum enum_BuildLevel	BuildLevel3_VoltageLoop_AC	0x0000A811@Data
pwmSwState	enum enum_pwmSwState	pwmSwState_negativeHalf	0x0000A826@Data
boardStatus	enum enum_boardStatus	boardStatus_NoFault	0x0000A80F@Data
clearTrip	int	1	0x0000A81A@Data
closeGvLoop	int	1	0x0000A819@Data
vBusRef	float	0.821337461	0x0000A850@Data
vBus_sensed	float	0.822998047	0x0000A8C0@Data
closeGiLoop	int	1	0x0000A81C@Data
ac_cur_senseOffset	float	0.502499998	0x0000A888@Data
guiVbus	float	380.081421	0x0000A858@Data
guiVin	float	-152.073486	0x0000A82C@Data
guiVrms	float	120.093376	0x0000A842@Data
guiIrms	float	2.40836215	0x0000A832@Data
guiPrms	float	277.007263	0x0000A834@Data
guiPowerFactor	float	0.990778685	0x0000A82E@Data
guiFreqAvg	float	60.0219727	0x0000A844@Data
EPwm1Regs.TZFLG	Register	0x0000	
EPwm2Regs.TZFLG	Register	0x0000	
dutyPU	float	-0.4262546	0x0000A86E@Data
dutyPU_DC	float	0.5	0x0000A86C@Data
iL1_sensed	float	0.0561523438	0x0000A884@Data
iL2_sensed	float	-0.0673828125	0x0000A880@Data
iL3_sensed	float	-0.0434570313	0x0000A882@Data
autoStartSlew	unsigned long	5	0x0000A83E@Data
Add new expression			

図 39. ビルドレベル3: AC電圧を印加した後のExpressionsビュー

4. このビルドのソフトウェアにはSFRAが統合されているため、ハードウェアを測定して、設計した補償器が十分な利得マージンと位相マージンを提供していることを検証できます。SFRAを実行するには、プロジェクトを実行している状態で、cfgページから*SFRA*アイコンをクリックします。SFRA GUIが表示されます。
5. SFRA GUIでデバイスのオプションを選択します。例として、F28004xの場合には浮動小数点を選択します。*Setup Connection*をクリックし、ポップアップウィンドウでBoot on Connectオプションのチェックを外して、適切なCOMポートを選択します。*OK*をクリックします。SFRA GUIに戻り、*Connect*をクリックします。

6. SFRA GUIがデバイスに接続します。これで*Start Sweep*をクリックして、SFRA掃引を開始できます。SFRA掃引が完了するまでには数分かかります。SFRA GUIのプログレスバーを確認したり、UARTの動作を示す制御カード裏面の青色LEDの点滅をチェックすることで、動作を監視できます。終了すると、図 40のように開ループプロットによるグラフが表示されます。この操作により、設計した補償器が確かに安定していることを検証できます。

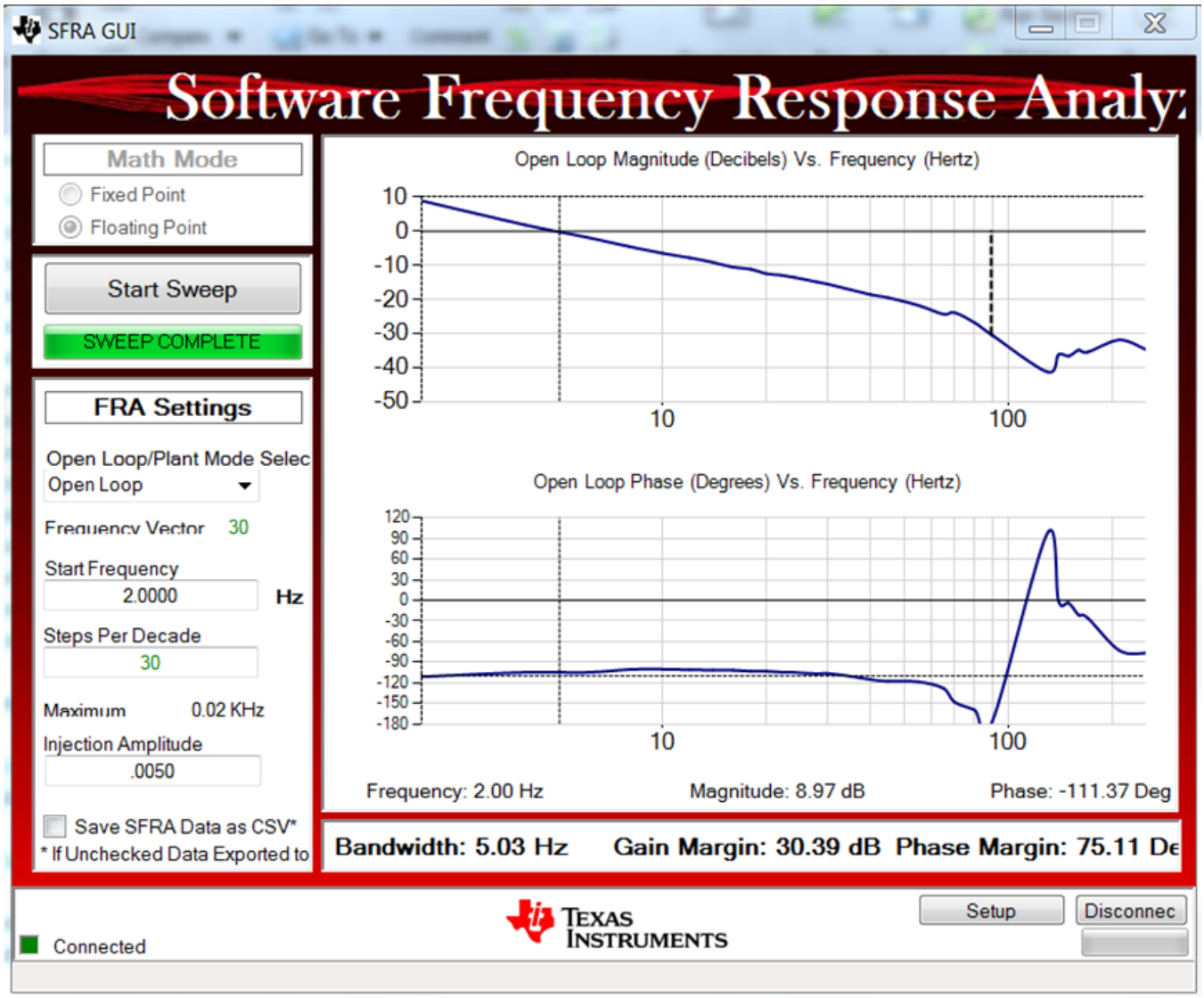


図 40. 閉電圧ループでのSFRA実行

また、周波数応答データはSFRAデータフォルダ下のプロジェクトフォルダに保存され、SFRA実行時のタイムスタンプが記録されます。

図 12に示すように、利得マージンと位相マージンの測定値はモデルの値に近接しています。

7. オプションとして、CFGページから再度Compensation Designerをクリックし、GUIでプラントオプションに*SFRA Data*を選択します。このオプションにより、測定したプラント情報を用いて補償器を設計し、補償を微調整できます。デフォルトでは、Compensation Designerは最新のSFRA実行を示しています。過去のSFRA実行プラント情報を使用する必要がある場合、ユーザーは*Browse SFRA Data*をクリックし、参照してSFRAData.csvファイルを選択できます。Compensation Designerを閉じて、完了したcfgページに戻ります。
8. これにより、電流補償器設計を検証できます。
9. システムを安全に停止させるには、入力AC電圧をゼロまで下げ、guiVBusもゼロに下がっていることを確認します。
10. リアルタイムモードのMCUを完全に停止するには、2段階の手順を踏みます。まず、ツールバーの*Halt*ボタン()か*Target* → *Halt*をクリックしてプロセッサを停止します。次に、  をクリックしてMCUをリアルタイムモードから解除します。最後に、MCUをリセットします()。
11. *Terminate Debug Session* (*Target* → *Terminate all*)をクリックして、CCSデバッグセッションを終了します。 

3.1.2.5 CLAでのコード実行

このソリューションは、オプションによりCLAでコードを実行できます。このオプションは、powerSUITE main.cfgページでProject Optionsのドロップダウンボックスを使用して選択します。どのビルドレベルについても、CLAでの実行を選択できます。

注: SFRAライブラリはCLAに対応していないため、CLA使用時にSFRAを実行することはできません。また、CLA使用時にはDLOGを使用しないため、データロググラフを使用することもできません。

オプションを変更したら、CFGファイルを保存し、プロジェクトを再コンパイルする必要があります。再コンパイルしたら、各差分ビルドレベルの資料に記載されている手順に従います。

製品によって異なりますが、たとえばF28004xのCLAはCLAタスクとバックグラウンドタスクに対応しているため、100kHz ISRと10kHz ISRの両方をCLAにオフロードできます。デフォルトでは、powerSUITEページで選択した場合、高速のISRはCLAタスクに移行され、低速のISRはバックグラウンドタスクに移行されます。10kHz ISRをCLAで実行したくない場合は、solutions-settings.hファイルの「USER SECTION」でそのように設定できます。

```
#if CONTROL_RUNNING_ON == CLA_CORE
#define INSTRUMENTATION_ISR_RUNNING_ON CLA_CORE
#else
#define INSTRUMENTATION_ISR_RUNNING_ON C28x_CORE
#endif
```

3.1.2.6 その他の設定

本節では、力率(PF)を高める高度な設定について説明し、テスト結果をもってそれぞれの総体的影響を数値化します。

3.1.2.6.1 軽負荷時のPF向上のための入力容量補償

式 8および図 41(a)に示すように基準電流が電圧と完全に同期した状態に維持されると、入力容量によりPFが低下します。

$$i_{ref\_dpllvc} = i^{*}_{ref} \sin(\omega t) - i_{input\_cap\_comp} \cos(\omega t) \quad (8)$$

基準電流をベクトルで調整することにより、PFの低下をオフセットできます(式 9、図 41(b))。位相ロックループからの角度を用いてベクトルを計算することから、この手法はデジタル位相ロックループベクトルキャンセレーション(DPLLVC)法と呼ばれています。これにより、軽負荷・高入力電圧時のPFが大幅に向上します。

$$i_{ref\_DPLLVC} = i^{*}_{ref} \sin(\omega t) - i_{input\_cap\_comp} \cos(\omega t) \quad (9)$$

印加される補正量は入力容量値によって異なります。例として、このリファレンス・デザインの入力容量は2.2µF

$$\frac{220}{\frac{1}{2} \times pi \times 60 \times 2.2\,\mu F} = 0.1823\ A$$

です。これは、高入力電圧時に に等しい容量性電流が引き出されることを意味します。軽負荷時に、これは大量の電流となり、力率損失を招きます。このリファレンス・デザインの電流センサ利得は約24Aであるため、高入力電圧条件において約0.01puの調整ということになります。

さらに、低消費電力条件下では追従誤差が生じます。この追従誤差は、図 41(c)に示す電流指令の調整によりオフセットできます。この追従誤差量は、システムの最高性能に応じて経験的に調整されます。したがって総基準電流は式 10で求められます。

$$i_{ref\_DPLLVC\_TC} = i^{*}_{ref} \sin(\omega t) - i_{input\_cap\_comp} \cos(\omega t) - i_{tracking\_error}\Big|_{power\_dependent} \cos(\omega t) \quad (10)$$

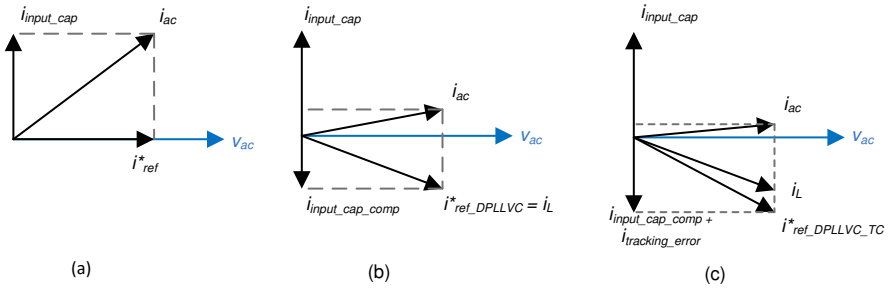


図 41. 力率(a)調整なし(b) DPLLVC (c) DPLLVC＋追従誤差補償

図 42にPF向上の結果をグラフ化します。

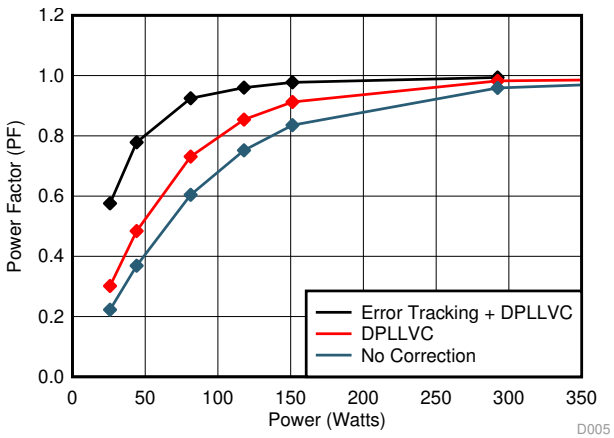


図 42. 入力容量電流補償による向上を示す220Vrms時のPFグラフと消費電力との関係

この機能は、<solution>-settings.hファイルでINPUT_CAP_COMPENSATION #defineの変数を書き換えることによってオフまたはオンにできます。この設定はヘッダファイルの「USER SECTION」領域にあり、ユーザーにより変更可能です。調整値は、#define HIGH_LINE_INPUT_CAP_COMP_ADJUSTおよびLOW_LINE_INPUT_CAP_COMP_ADJUSTにより制御されます。これらはユニット単位形式であり、この値の決定についてはすでに述べたとおりです。

```
#define INPUT_CAP_COMPENSATION 1
#define HIGH_LINE_INPUT_CAP_COMP_ADJUST -0.01823
#define LOW_LINE_INPUT_CAP_COMP_ADJUST  -0.00911
```

3.1.2.6.2 アダプティブデッドタイムによる効率向上

連続導通モードでは、短絡保護と効率の観点から同期整流のデッドタイム制御が重要とされます。最適なデッドタイムにより、シュートスルーのリスクを解消できるだけでなく、同期FETのボディダイオード導通による過剰な導通損失も防止できます。したがって、最適なデッドタイムの目標は、アクティブFETと同期FETを同時にオンにせず、同期FETの冗長な第3象限導通を最小限に抑えることです。

この最適なデッドタイムは測定した電流とデバイス出力容量から計算でき、これは式 11で求められます。

$$t_{deadtime\_optimal} = \frac{2C_{oss}V_{out}}{i_{L\_peak}} \quad (11)$$

図 43 に実装のブロック図を示します。

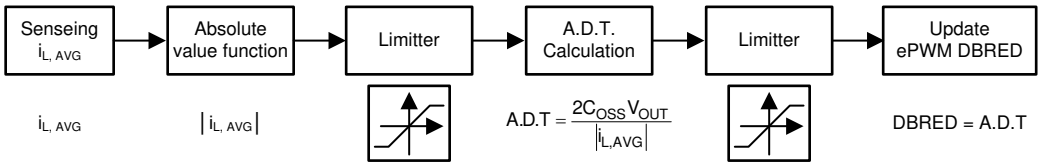


図 43. アダプティブデッドタイムの実装

このオプションにより、図 44の高入力電圧のケースで示すとおり、固定デットタイムに比べて消費電力を削減できます。低消費電力時のシュートスルーを回避することで、消費電力を大幅に削減できるものと推測されます。ただし、アダプティブデッドタイム調整を実装することにより、シュートスルーを回避した後、消費電力削減量は最初は低下し、その後、消費電力が増えるに従って徐々に増加し、ダイオード導通時間が短縮されます。

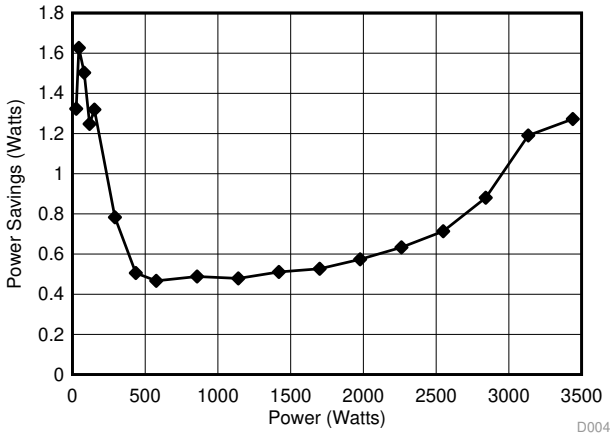


図 44. 高入力電圧230Vrms時のアダプティブデッドタイムによる消費電力削減

アダプティブデッドタイムを有効にするには、ソリューションの*CFG/powerSUITE*ページで*Project Options*下のドロップダウンボックスを選択します。立ち下がりエッジ遅延(FED)には、CFGページで指定した固定値を使用します。アダプティブデッドタイムを有効にすると、立ち上がりエッジ遅延(RED)が変調されるため、*<solution>-settings.h*の*user section*で最小/最大範囲を指定します。以下は調整可能な#defineです。

- デバイス出力容量: #define GAN_COSS 0.000000000145
- 最小許容デッドタイム: #define PFC_DEADBAND_RED_MIN_US 0.020
- 最大許容デッドタイム: #define PFC_DEADBAND_RED_MAX_US 0.200

このオプションを変更した場合は、これらの調整を行った後、プロジェクトをコントローラに保存、再コンパイル、ロードする必要があります。3.1.2.4.4に述べるハードウェアセットアップおよびソフトウェアに関する指示に従って、基板の動作を確認し、効率を測定できます。

3.1.2.6.3 フェーズ・シェディングによる効率向上

フェーズ・シェディングは、導通損失およびスイッチング損失を最適化して、インターリーブ制御の効率を高める効果的な手法となり得ます。図 45に示すように、このリファレンス・デザインには3つの位相があることから、3種類の構成が可能です。

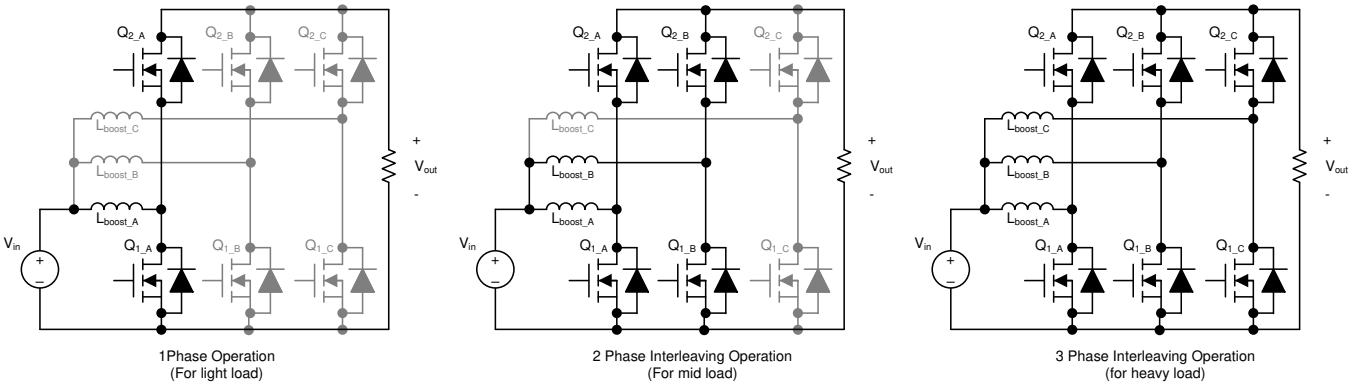


図 45. TTPL PFCのフェーズ・シェディングオプション

これらのモードのそれぞれの間で位相シフトを調整する必要があります。二相モード時には、PWM間で180°の位相シフトが必要とされ、三相モード時には120°の位相シフトが必要とされます。

フェーズ・シェディングを実行する決定は、RMS電流、消費電力、ピークインダクタ電流など、さまざまなパラメータに基づいて下すことができます。RMS電流を用いると、位相の変更は大幅に遅れる可能性があります。図 46に、決定がRMS電流に基づく場合のフェーズ・シェディングを示します。位相を追加するまでに、コードは複数のACサイクルを要します。

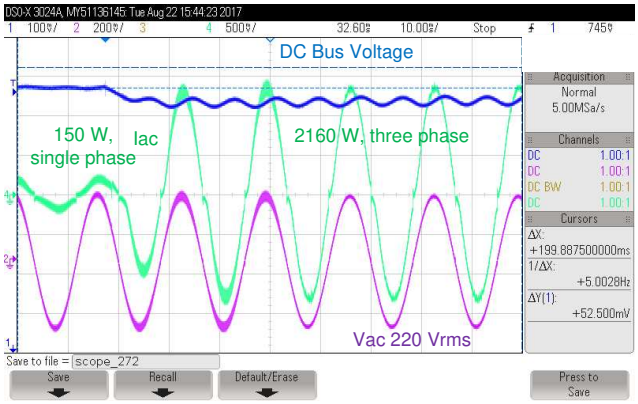


図 46. 位相を追加する決定がRMS計算に基づく場合の波形

多くの機器では、このような遅延は許容されません。したがって、位相を削減したり追加したりする決定点として、電圧コントローラ出力を選択します。図 47に示すようにステートマシンを構成し、フェーズ・シェディング点周りにヒステリシスを構築します。

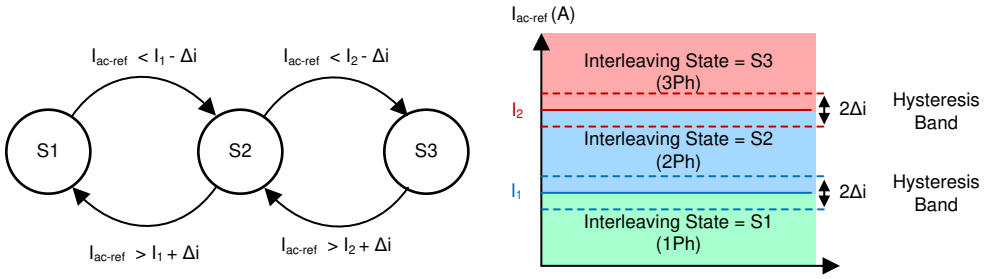


図 47. フェーズ・シェディング制御のステートマシン

位相を追加/削減することにより、想定外のパルスが生成されるおそれがあります。このため、位相を追加/削減するための実装は、GPIOピンMUXレジスタを使用して、GPIOおよびPWMペリフェラルスイッチで行います。すべてのPWM対応ピンが構成され、GPIO出力はLOレベルに駆動されます。ここで、印加する必要がある位相の数に基づき、それに応じてGPIOピンMUXが変更されます。PWMのレジスタがシャドウイングされるため、ACサイクルのどの点でも、GPIOピンMUXスイッチを使用して位相を有効/無効にすることができます。図 48にC2000 MCUへのフェーズ・シェディングの実装の詳細を示します。

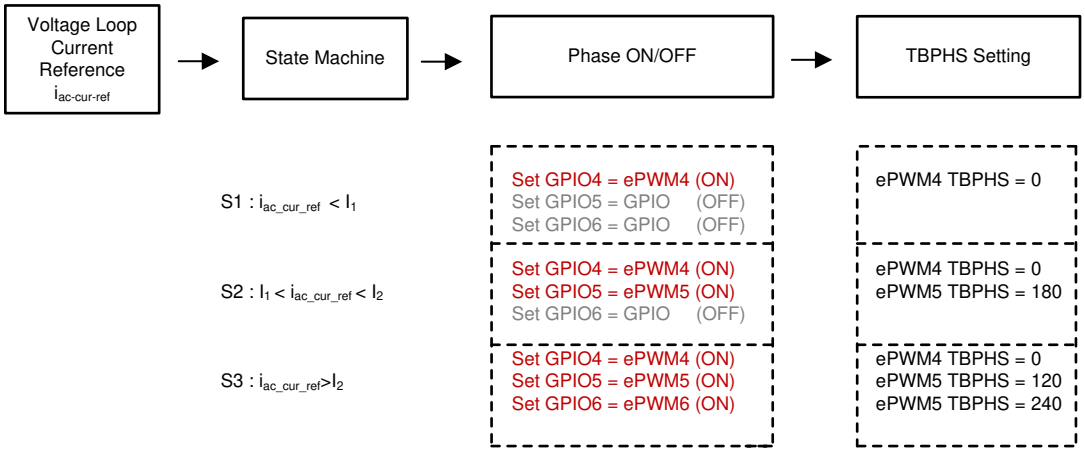


図 48. C2000™ MCUを使用したTTPL PFCへのフェーズ・シェディングの実装

フェーズ・シェディングを有効にするには、*<solution>-settings.h*ファイルのコードのuser sectionで設定し、*PHASE_SHEDDING_ENABLED*というdefineを修正します。位相を追加/削減する点は、*PHASE_SHEDDING_1PH_2PH_TRANSITION_CURRENT*および*PHASE_SHEDDING_2PH_3PH_TRANSITION_CURRENT*というdefineを変更して設定し、それぞれ図 48に示すI1およびI2に該当します。コードを再コンパイルし、ロードして、3.1.2.4.4で述べた手順を繰り返して、この機能をテストします。この機能を実装することにより、過渡時に位相が素早く削除/追加されます。

図 49および図 50に、110Vrms時の1.3KW→150Wおよび150W→1.3KWという過渡事象を示します。決定は電圧ループで生成される基準電流に基づくため、過渡時に素早く位相が追加/削除されます。

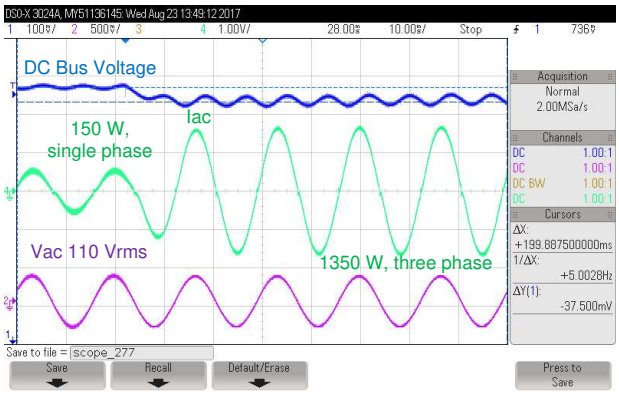


図 49. 120Vrms、60Hz時に過渡時に素早く追加される位相

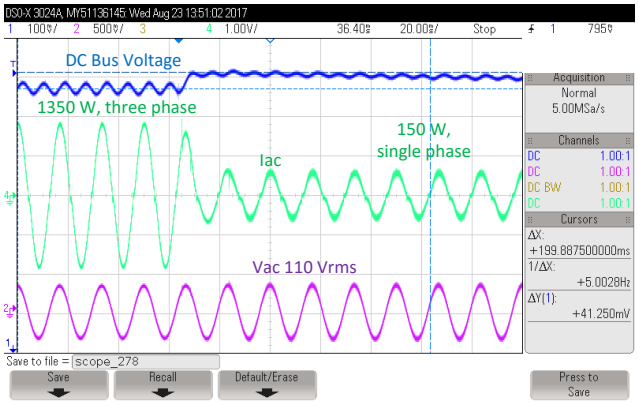


図 50. 120Vrms、60Hz時に過渡時に素早く削減される位相

同様に高入力電圧時には、2KWを上回る過渡電力が印加され、位相はほぼ一瞬で単相から三相に移行します。図 51および図 51に、高入力電圧時の150W→2.16KWおよび2.16KW→150Wという過渡時でのテスト波形を示します。

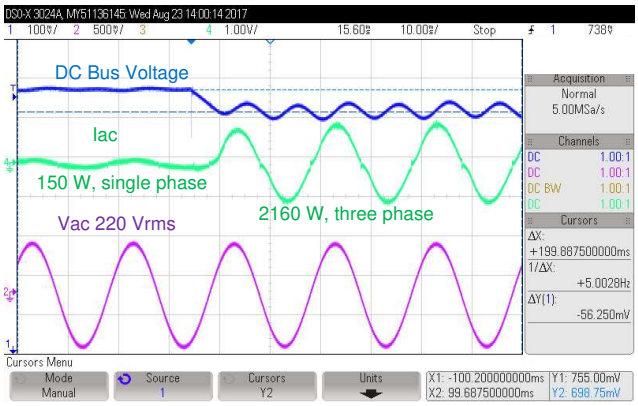


図 51. 230Vrms、50Hz時に過渡時に素早く追加される位相

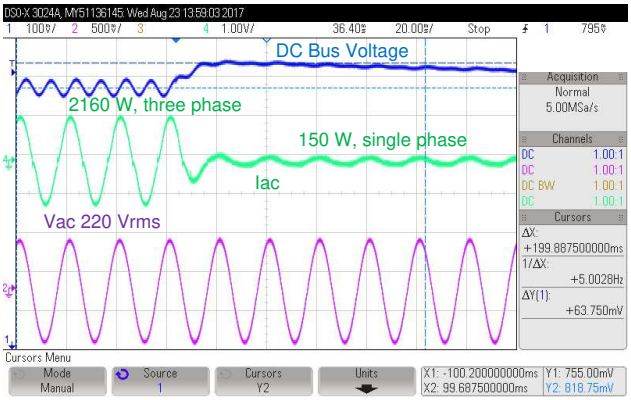


図 52. 230Vrms、50Hz時に過渡時に素早く削減される位相

図 53 にフェーズ・シェディングによる230Vrms時の効率向上を示します。

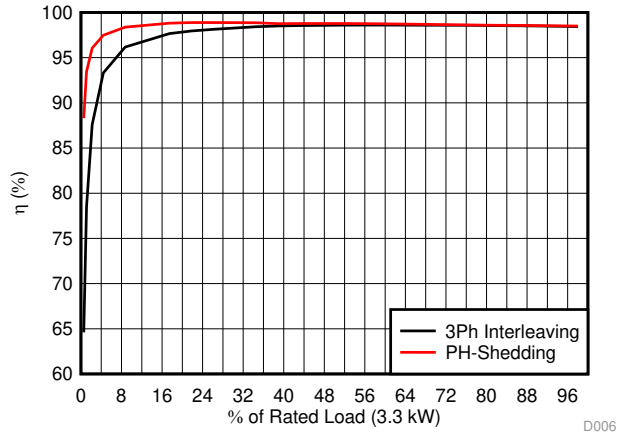


図 53. フェーズ・シェディングあり/なしでの230Vrms時の効率比較

3.1.2.6.4 非線形電圧ループによる過渡事象の抑制

PFC段制御は、入力電圧に追従しようとする内部電流ループと、出力時に一定のDCバス電圧を維持しようとする外部電圧ループで構成されます。このように電圧ループは電流ループと対立するため、適切な力率を達成するには、極めて狭い帯域幅(約10Hz)になるように設計する必要があります。低速な電圧ループは、過渡時に重大なオーバーシュートやアンダーシュートを引き起こします(図 54を参照)。

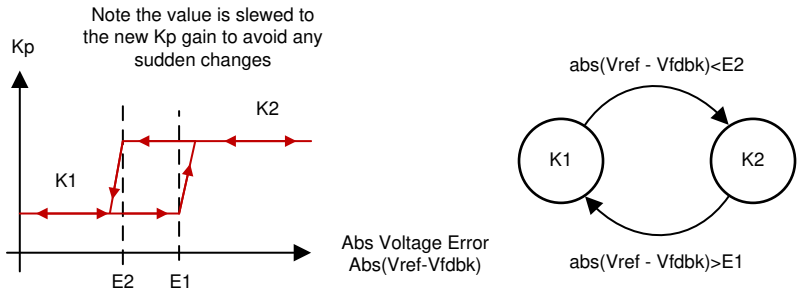


図 54. ヒステリシス付き非線形電圧ループ

電圧オーバーシュートおよびアンダーシュートを改善するには、図 55に示すように、適切な力率を維持しながら非線形電圧制御ループを実装します。非線形電圧ループにヒステリシスバンドを追加することにより、高利得モード/低利得モード間の変動を回避できます。さらに、利得の変動に追従して、突然の変化を回避します。図 56に非線形電圧ループの結果を示します。

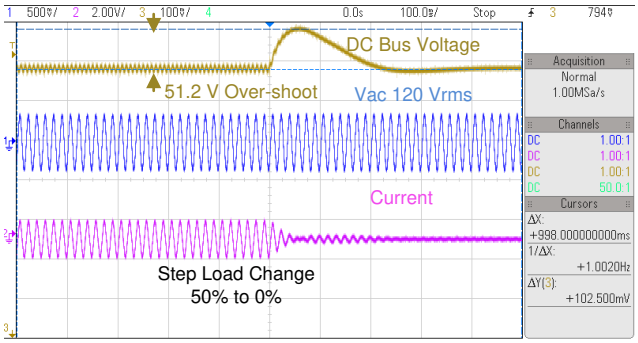


図 55. 非線形電圧ループなしの電圧過渡、Vin 120Vrms、880W→0W過渡事象、オーバーシュート51.2V

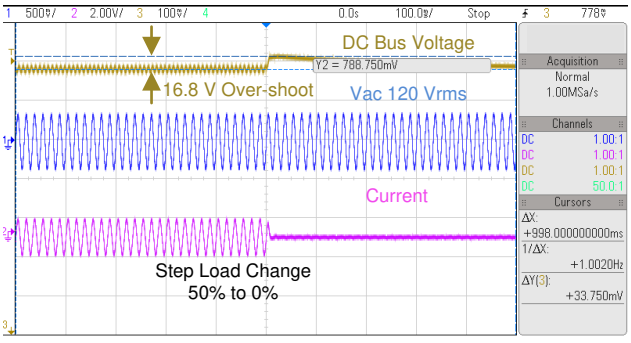


図 56. 非線形電圧ループありの電圧過渡、Vin 120Vrms、880W→0W過渡事象、オーバーシュート16.8V

非線形電圧ループを有効にするには、ソリューションの*CFG/powerSUITE*ページで*Project Options*下のドロップダウンボックスを選択します。過渡条件下の比例項には、デフォルト値である5倍の利得が適用されます。この値は、*<solution>-settings.h*ファイルの*user section*で、*NON_LINEAR_V_LOOP_KP_MULTIPLIER*というdefineを変更することにより調整できます。このオプションを変更した場合は、プロジェクトをコントローラに保存、再コンパイル、ロードする必要があります。ビルドレベル3のハードウェアセットアップおよびソフトウェアに関する指示に従って、過渡条件下の基板の動作を確認できます。

3.1.2.6.5 ソフトウェア位相ロックループ法: SOGI - FLL

業界標準を満たすには、周波数過渡条件下でこのリファレンス・デザインをテストする必要があります。PLLが周波数変動に適応できない場合、PLL角度を用いてPWM信号を駆動すると、これにより問題が生じます。*『Grid Synchronization of Power Converters Using Multiple Second Order Generalized Integrators』* [1]で提案されている周波数ロックループ法の採用により、ソフトウェア位相ロックループ周波数を適応性のあるものにすることができます。

このモジュールについては、SOGI PLLモジュールと同じ基本構造を採用し、*『Software PLL Design Using C2000 MCUs Single Phase Grid Connected Inverter』* [2]で説明する実装を行います。図 57に示すように、周波数ロックループによる周波数アダプティブ機能が追加されます。

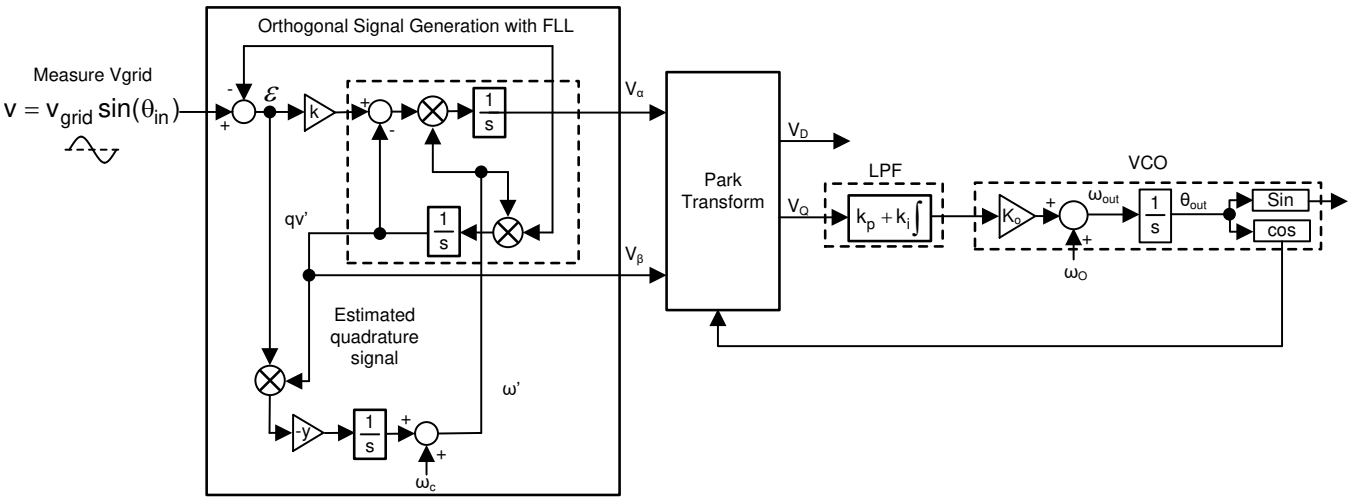


図 57. 2次汎用積分器と周波数ロックループを採用したソフトウェア位相ロックループ構造

PLL法を選択するには、*<solution>-settings.h*の*user section*を開きます。以下は調整可能な#defineです。

- #define SPLL_METHOD_SELECT SPLL_1PH_SOGI_FLL_SEL

この調整を行った後、プロジェクトをコントローラに保存、再コンパイル、ロードする必要があります。ビルドレベル3のハードウェアセットアップおよびソフトウェアに関する指示に従って、新しいPLL方式による基板の動作を確認できます。

3.2 テストと結果

3.2.1 入力120Vrms、60Hz、出力380V DC時のテスト結果

3.2.1.1 起動

入力単相120Vrms VL-N、380Vに電圧が制御された出力バス、1.6kW負荷および無負荷での電力段の起動シーケンスを図 58に示します。

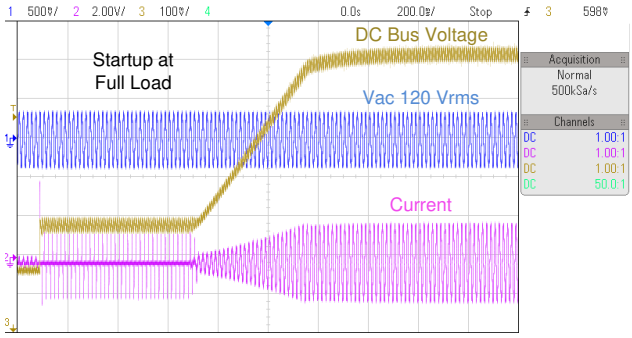


図 58. 120Vac IN、380V DC OUT、1.6KW負荷時のPFC動作の起動

図 59 に無負荷時の起動を示します。

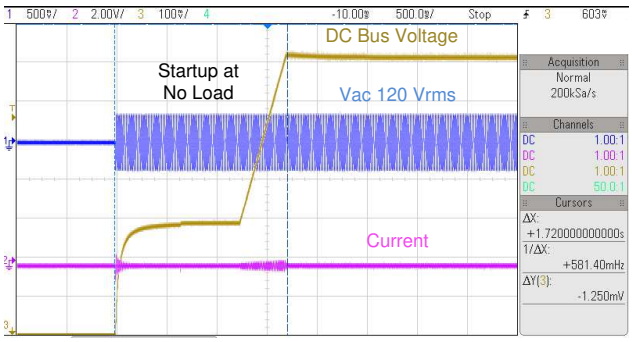


図 59. 120Vac IN、380V DC OUT、0%負荷時のPFCの起動

3.2.1.2 定常状態条件

さまざまな負荷条件での定常状態電流波形を図 60、図 61、図 62に示します。これらのデータについては、フェーズ・シェディングが無効になっています。

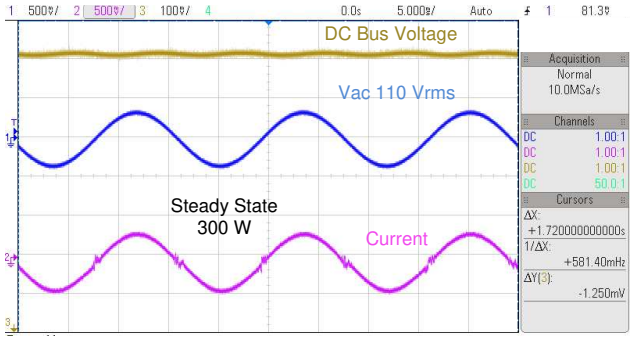


図 60. 定常状態120Vac IN、380V DC OUT、300W、iTHD5.5%

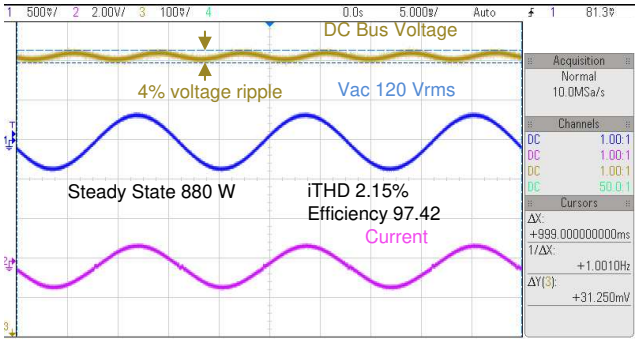


図 61. 定常状態120Vac IN、380V DC OUT、880W、iTHD2.15%

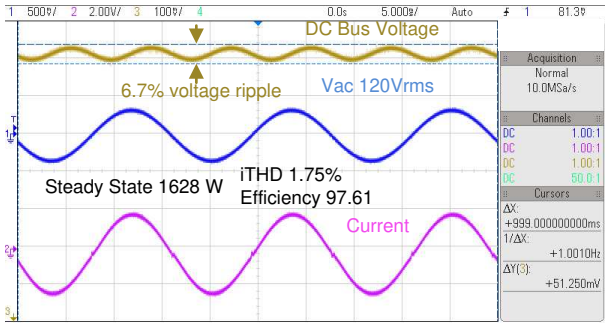


図 62. 定常状態120Vac IN、380V DC OUT、1.674KW、iTHD1.75%

表 4 に、120Vac入力、380V DC出力、さまざまな負荷条件でのこのリファレンス・デザインの詳細なテスト結果を示します。このデータについては、フェーズ・シェディング無効、アダプティブデッドタイム有効、ハードスイッチエッジの固定デッドタイムとして100nsを選択、ソフトスイッチングエッジのデッドタイムは20ns～200nsの範囲で変動するものとします。

表 4. 120Vac IN、380V DC OUT、さまざまな負荷電力での詳細なテスト結果

Vin (V RMS)	Vout (V)	Pin (W)	Iout (A)	Pout (W)	効率%	iTHD%	PF	%定格負荷	θオフセット	GI KP
120.05	382.02	154.27	0.375	143.47	92.98	10.54	0.9927	9.0	−0.014	0.35
119.86	382.01	301.30	0.750	286.36	95.14	5.50	0.9974	17.9	−0.01	0.35
119.49	382.01	444.40	1.120	427.76	96.30	4.16	0.9987	26.7	−0.01	0.35
119.42	382.03	579.10	1.469	561.40	96.94	2.89	0.9950	35.1	−0.01	0.35
119.16	382.02	721.30	1.837	701.80	97.30	2.42	0.9995	43.9	−0.01	0.35
119.02	382.05	863.00	2.202	841.50	97.52	2.15	0.9995	52.6	0	0.35
118.78	381.96	1007.20	2.573	983.30	97.64	1.92	0.9995	61.5	0	0.35
118.63	382.08	1152.00	2.944	1125.30	97.69	1.82	0.9995	70.3	0	0.35
118.40	382.08	1298.40	3.319	1268.20	97.70	1.72	0.9994	79.3	0	0.35
118.25	382.08	1442.00	3.685	1408.30	97.69	1.87	0.9991	88.0	0	0.3
118.03	382.08	1593.80	4.071	1555.50	97.65	1.80	0.9991	97.2	0	0.3
117.98	382.05	1716.40	4.449	1674.80	97.61	1.75	0.9991	104.7	0	0.3

3.2.1.3 ステップ負荷変動による過渡応答テスト

3.2.1.3.1 0%→50%負荷ステップ変動

図 63 に入力120Vrmsで負荷ステップ50%を電力段に印加したときの過渡応答を示します。

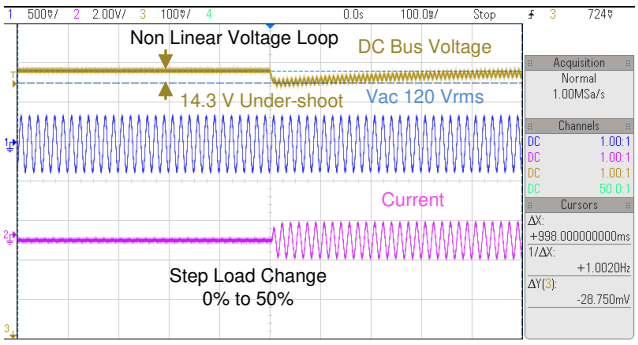


図 63. 120Vrms、60Hz、0%→50%負荷ステップ時の過渡応答

3.2.1.3.2 50%→100%負荷ステップ変動

図 64 に入力120Vrmsで負荷を50%から100%に上げたときの過渡応答を示します。

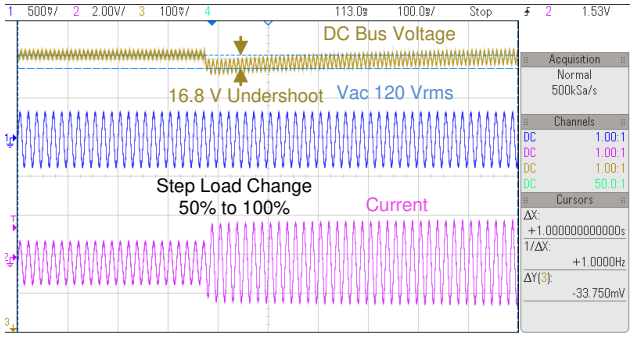


図 64. 120Vrms、60Hz、50%→100%負荷ステップ時の過渡応答

3.2.1.3.3 100%→50%負荷ステップ変動

図 65 に入力120Vrmsで負荷を100%から50%に下げたときの過渡応答を示します。

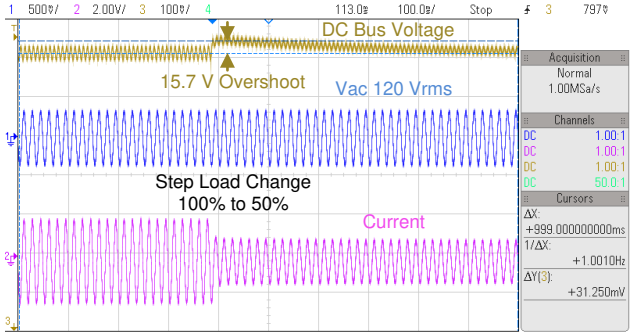


図 65. 120Vrms、60Hz、100%→50%負荷ステップ時の過渡応答

3.2.1.3.4 50%→0%負荷ステップ変動

図 66 に入力120Vrmsで負荷を50%から0%に下げたときの過渡応答を示します。

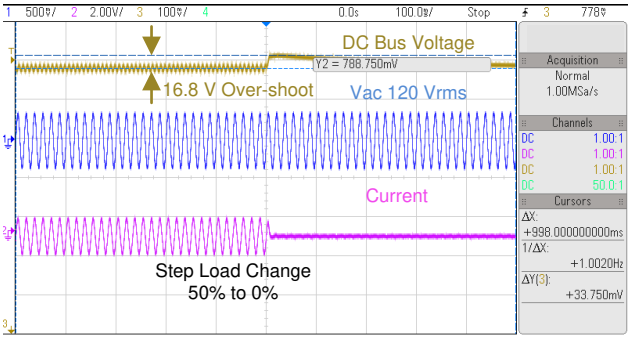


図 66. 120Vrms、60Hz、50%→0%負荷ステップ時の過渡応答

3.2.2 入力230Vrms、50Hz、出力380V DC時のテスト結果

3.2.2.1 起動

図 67 に入力単相230Vrms VL-N、380Vに電圧が制御された出力バス、880W負荷での電力段の起動シーケンスを示します。

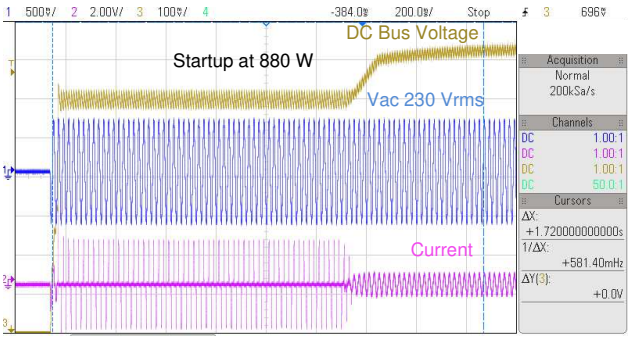


図 67. 230Vac IN、380V DC OUT、880W負荷時のPFC動作の起動

図 68 に無負荷、230Vrms時のPFCの起動を示します。

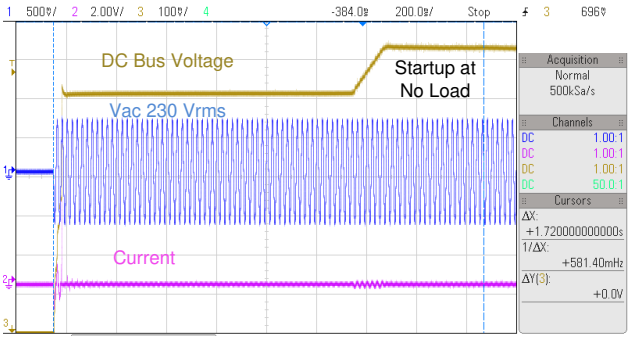


図 68. 230Vac IN、380V DC OUT、無負荷時のPFC動作の起動

3.2.2.2 定常状態条件

図 69、図 70、図 71に、さまざまな負荷条件での定常状態電流波形を示します。これらのデータについては、フェーズ・シェディングが無効になっています。

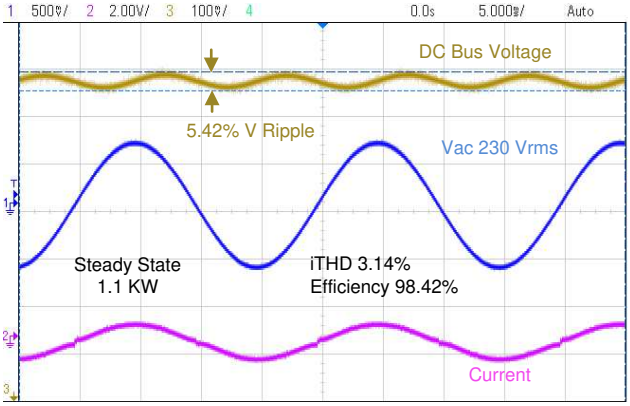


図 69. 定常状態230Vac IN、380V DC OUT、1.1KW、iTHD3.14%

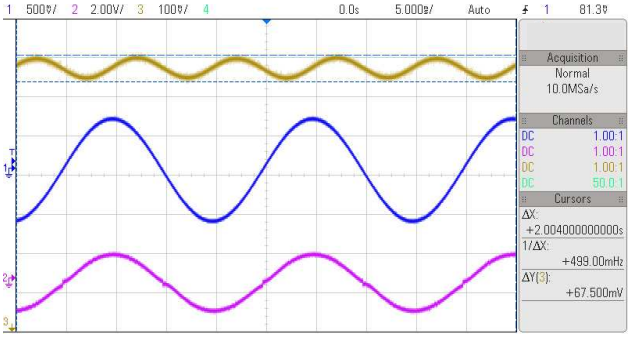


図 70. 定常状態230Vac IN、380V DC OUT、2.2KW、iTHD2.62%

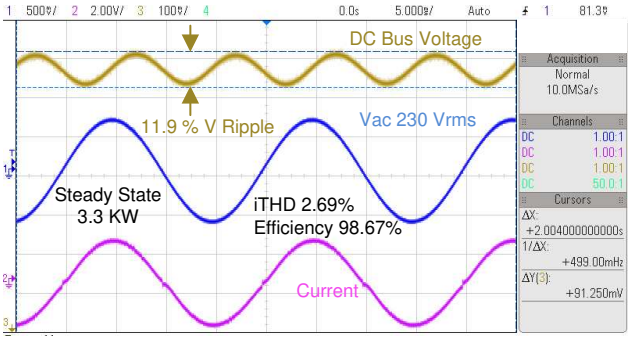


図 71. 定常状態230Vac IN、380V DC OUT、3.3KW、iTHD2.69%

表 5 に、230Vac入力、380V DC出力、さまざまな負荷条件でのこのリファレンス・デザインの詳細なテスト結果を示します。以下のデータについては、フェーズ・シェディング無効、アダプティブデッドタイム有効、ハードスイッチエッジの固定デッドタイムとして100nsを選択、ソフトスイッチングエッジのデッドタイムは20ns～200nsの範囲で変動するものとします。

表 5. 230Vac IN、380V DC OUT、さまざまな負荷電力での詳細なテスト結果

Vin (V RMS)	Vout (V)	Pin (W)	Iout (A)	Pout (W)	効率%	iTHD%	PF	%定格負荷	θオフセット	Gi Kp
230.68	381.98	151.28	0.372	142.16	94.03	18.20	0.9775	4.4	−0.025	0.35
230.43	382.00	292.24	0.736	281.41	96.29	9.15	0.9936	8.8	−0.02	0.35
230.25	382.03	435.90	1.109	423.62	97.18	6.12	0.9938	13.2	−0.01	0.35
230.06	382.06	576.40	1.473	562.86	97.66	4.85	0.9972	17.6	−0.01	0.35
229.80	382.05	856.80	2.201	841.00	98.15	4.16	0.9974	26.3	0	0.35
229.70	382.11	1140.10	2.935	1121.90	98.42	3.14	0.9989	35.1	0	0.35
229.52	382.08	1418.80	3.659	1398.40	98.57	2.42	0.9993	43.7	0	0.3
229.28	382.08	1699.20	4.386	1676.40	98.66	2.74	0.9995	52.4	0	0.3
229.06	382.09	1977.70	5.106	1951.90	98.71	2.56	0.9996	61.0	0	0.3
229.09	382.11	2261.50	5.840	2232.40	98.73	2.62	0.9995	69.8	0	0.25
228.91	382.11	2548.30	6.580	2515.60	98.73	2.50	0.9994	78.6	0	0.25
228.86	382.14	2840.60	7.332	2803.20	98.71	2.89	0.9990	87.6	0	0.2
228.51	382.12	3132.80	8.083	3091.10	98.69	2.80	0.9989	96.6	0	0.2
228.22	382.03	3439.10	8.873	3392.30	98.65	2.69	0.9988	106.0	0	0.2

3.2.2.3 ステップ負荷変動による過渡応答テスト

以下にステップ負荷変動による過渡応答テストの結果を示します。

3.2.2.3.1 33%→100%負荷ステップ変動

図 72 に入力230Vrmsで負荷を33%から100%に上げたときの過渡応答を示します。

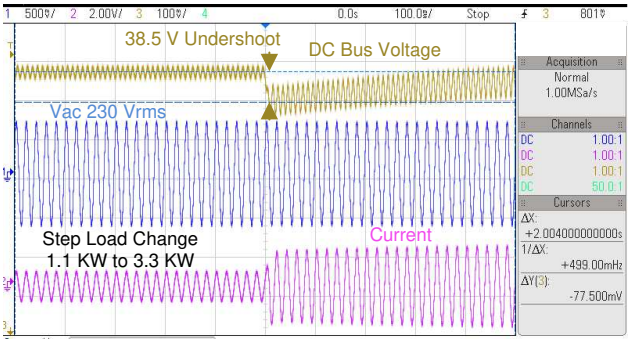


図 72. 230Vrms、50Hz、33%→100%負荷ステップ時の過渡応答

3.2.2.3.2 100%→33%負荷ステップ変動

図 73 に入力230Vrmsで負荷を100%から33%に下げたときの過渡応答を示します。

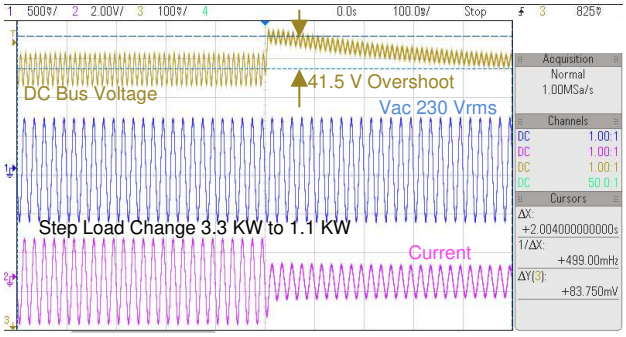


図 73. 230Vrms、50Hz、100%→33%負荷ステップ時の過渡応答

3.2.3 テスト結果のグラフ

このリファレンス・デザインについて、さまざまなテスト条件でプロットした効率性データを図 74に示します。

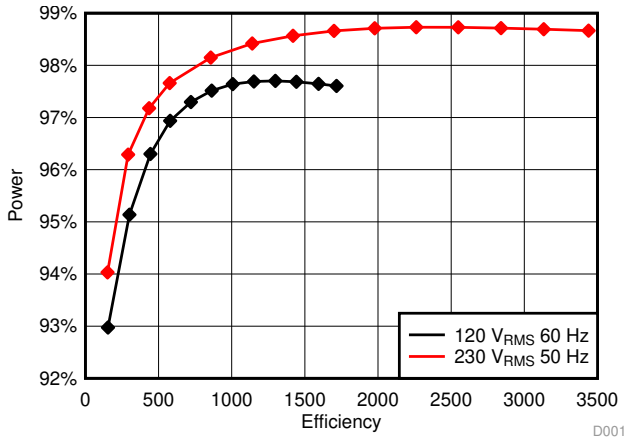


図 74. 230Vrms入力時および120Vrms入力時の効率性

これらのテスト条件でプロットしたTHDデータを図 75に示します。

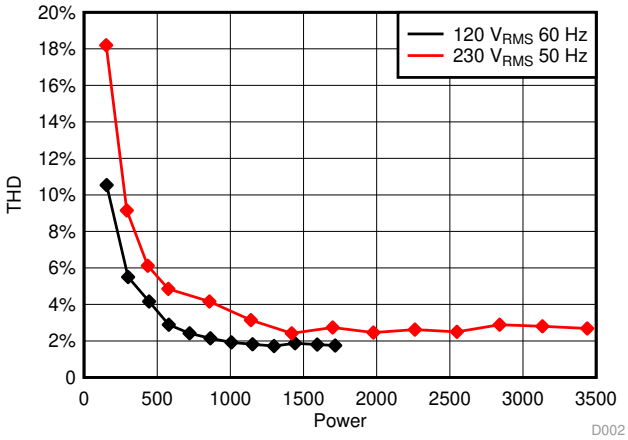


図 75. 230Vrms入力時および120Vrms入力時のTHD

これらのテスト条件でプロットしたPFデータを図 76に示します。

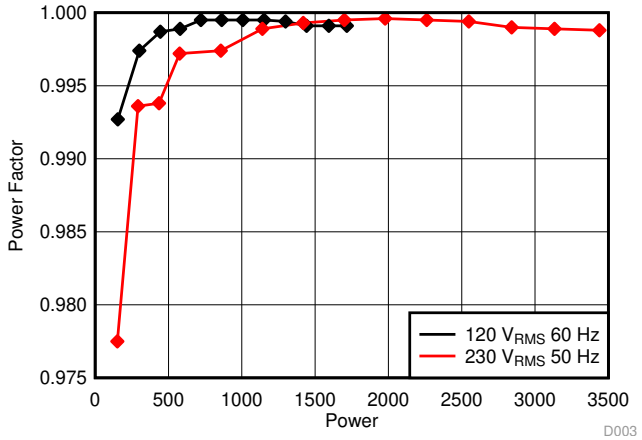


図 76. 230Vrms時および120Vrms時の負荷の変化に対するPF

4 Design Files

4.1 Schematics

回路図をダウンロードするには、TIDM-1007のデザインファイルを参照してください。

4.2 部品表

部品表(BOM)をダウンロードするには、TIDM-1007のデザインファイルを参照してください。

4.3 PCBレイアウトに関する推奨事項

4.3.1 レイアウト・プリント

レイヤプロットをダウンロードするには、TIDM-1007のデザインファイルを参照してください。

4.4 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、TIDM-1007のデザインファイルを参照してください。

4.5 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、TIDM-1007のデザインファイルを参照してください。

4.6 組立図面

組立図面をダウンロードするには、TIDM-1007のデザインファイルを参照してください。

5 ソフトウェア・ファイル

ソフトウェア・ファイルをダウンロードするには、TIDM-1007のデザインファイルを参照してください。

6 関連資料

1. テキサス・インスツルメンツ、*『トーテムポール型PFCにおいてACゼロクロス時の電流スパイクを抑える方法』テクニカルブリーフ*
2. Z. Ye, A. Aguilar, Y. Bolurian and B. Daugherty, *GaN FET-Based High CCM Totem-Pole Bridgeless PFC*, Texas Instruments Power Supply Design Seminar, 2014-15.
3. L. Xue, Z. Shen, D. Boroyevich and P. Mattavelli, *GaN-based High Frequency Totem-Pole Bridgeless PFC Design with Digital Implementation*, IEEE 2015 Applied Power Electronics Conference, 2015, pp. 759-766.
4. H.-S. Youn, J.-B. Lee, J.-I. Black and G.-W. Moon, *A Digital Phase Leading Filter Current Compensation (PLFCC) Technique for CCM Boost PFC Converter to Improve PF in High Ligne Voltage and Light Load Condition*, IEEE Transactions on Power Eelectronics , vol. 31, no. 9, pp. 6596-6606, 2016.
5. D. M. V. d. Sype, K. D. Gusseme, A. P. M. V. d. Bossche and J. A. Melkebeek, *Duty-Ratio Feeforward for Digitally Controlled Boost PFC Converters*, IEEE Transactions on Industrial Electronics, vol. 52, no. 1, pp. 108-115, February 2005.
6. "Rodriguez, P.,Luna, A., Candela, I., Teodorescu, R., and Blaabjerg, F. *Grid Synchronization of Power Converters Using Multiple Second Order Generalized Integrators* , In Proceedings of IEEE industrial Electronics Conference (IECON'08), November 2008, pp 755-760"
7. テキサス・インスツルメンツ、*『C2000 MCU単相グリッドをインバータに接続したソフトウェアPLL設計』アプリケーションレポート*
8. テキサス・インスツルメンツ、*『TMS320F28004x Piccolo™マイクロコントローラ』データマニュアル*
9. テキサス・インスツルメンツ、*『LMG3410 600V 12Aシングルチャネル GaN電力段』データシート*

6.1 商標

C2000, E2E, powerSUITE, Code Composer Studio are trademarks of Texas Instruments.
すべての商標および登録商標はそれぞれの所有者に帰属します。

7 著者について

MANISH BHARDWAJは、テキサス・インスツルメンツC2000マイクロコントローラ・システム・ソリューション・グループのシステム・アプリケーション・エンジニアであり、デジタルパワー、モータ制御、およびソーラーパワー用リファレンス・デザインの開発を担当しています。2009年にTIに入社する以前、2008年にアトランタのジョージア工科大学で電気工学・コンピュータ工学の修士号を取得し、2007年にインドのデリー大学ネタージ・サブハス工科大学の工学士号を取得しています。

リビジョンBの改訂履歴

資料番号末尾の英字は改訂を表しています。その改訂履歴は英語版に準じています。

Revision A (March 2018) から **Revision B** に変更 **Page**

- 特長 追加 1
- F28004xを 2.3.1に、テキストを更新 追加 6
- 表 3で電源接続をTP612からTP604に 変更 12
- 表 3で電源接続をTP609からTP606/TP609に 変更 12
- 3.1.2.3 追加 19
- 3.1.2.5 追加 39

リビジョンAの改訂履歴

2017年11月発行のものから更新 **Page**

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が適用される各種規格や、その他のあらゆる安全性、セキュリティ、またはその他の要件を満たしていることを確実にする責任を、お客様のみが単独で負うものとします。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（www.tij.co.jp/ja-jp/legal/termsofsale.html）、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。

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が適用される各種規格や、その他のあらゆる安全性、セキュリティ、またはその他の要件を満たしていることを確実にする責任を、お客様のみが単独で負うものとします。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（www.tij.co.jp/ja-jp/legal/termsofsale.html）、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。