

## Design Guide: TIDA-010087

## 100A、2相デジタル制御バッテリー・テストのリファレンス・デザイン



## 概要

このリファレンス・デザインでは、C2000™ マイコンコントローラ (MCU) と高精度 ADC ADS131M08 を使用して、双方向のインターリーブ降圧コンバータ電力段の電流と電圧を正確に制御する方法を紹介します。このデザインは、C2000 マイコンの高分解能パルス幅変調 (PWM) 生成ペリフェラルを活用して、 $\pm 20\text{mA}$  未満の電流レギュレーション誤差と  $\pm 1\text{mV}$  の電圧レギュレーション誤差を実現しています。

## リソース

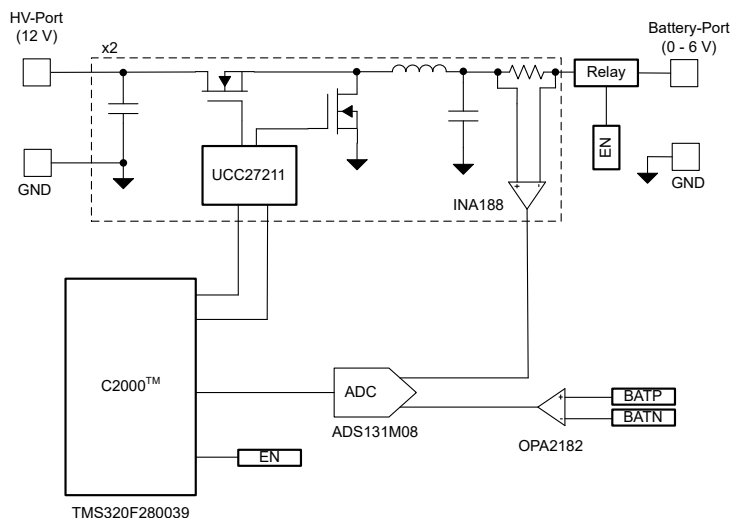
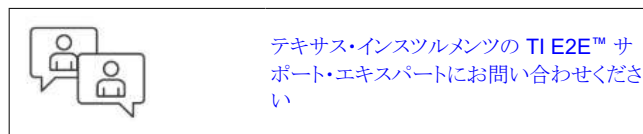
TIDA-010087	デザイン・フォルダ
TMS320F280039	プロダクト・フォルダ
ADS131M08、REF35、INA188	プロダクト・フォルダ
OPA2182、LM321LV、TLV9102、UCC27211	プロダクト・フォルダ
CSD17556Q5B、CSD16570Q5B、LMR54410	プロダクト・フォルダ
TPSI3050、TVS0500、TPS7A20	プロダクト・フォルダ
TLV1117、LM2664、TPS736	プロダクト・フォルダ
C2000WARE-DIGITALPOWER-SDK	ツール・フォルダ

## 特長

- 2相インターリーブ型、600W、双方向降圧電力段
- 100kHz のスイッチング周波数で 15.8 ビットの PWM 分解能
- 閉ループ制御用の外部デルタ・シグマ ADC
- レギュレーション誤差  $\pm 20\text{mA}$  未満の定電流充放電
- レギュレーション誤差は  $\pm 1\text{mV}$  未満で、充電と放電の両方で定電圧モードに対応
- ソフトウェア周波数応答アナライザ (SFRA) および補償デザインにより、制御ループを簡単に調整可能
- ユーザー要件に応じた設計の適用を容易にする powerSUITE をサポート

## アプリケーション

- バッテリー・セル形成とテスト機器
- プログラマブル DC 電源



## 1 システムの説明

バッテリー・テスト装置には、シングル・セル、バッテリー・モジュール、および高電圧バッテリー・パックのテストに使用されるさまざまな装置が含まれています。この試験装置は、高精度の電源とデータ収集システムを搭載しており、バッテリーの充電と放電に使用され、セルのさまざまなパラメータを測定します。

図 1-1 に、リチウムイオン・バッテリーを簡略化した製造プロセスを示します。最終段である、エンドオブライン・コンディショニングには、セル形成とテストが含まれます。形成はリチウムイオン電池の製造において重要なステップです。形成中、セルは最初の充電と放電のプロセスを経て、固体電解質界面 (SEI) 層が形成されます。SEI 層の品質は、バッテリー・セルの容量と信頼性に影響を及ぼします。形成プロセスを制御するために、セルの充電と放電に高精度のプロγραμμαブル電源が使用されます。これらの電源は、バッテリー形成システムまたはバッテリー・テストと呼ばれます。バッテリー・テストに必要な電圧および電流の精度は、通常、フルスケールの  $\pm 0.02\%$  ~  $\pm 0.05\%$  の範囲内です。

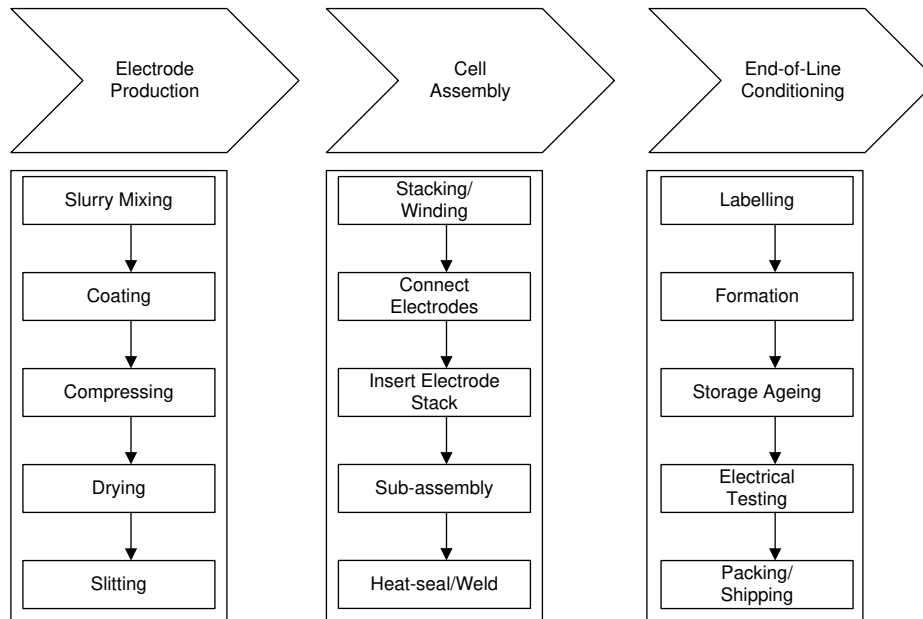


図 1-1. リチウムイオン・バッテリーの製造プロセスの簡素化

### 1.1 主なシステム仕様

パラメータ	仕様
LV ポート - バッテリー・ポート	50mV~6V
HV ポート - バス電圧	12V~15V
最大出力電流	$\pm 100A$
相あたりの最大 DC 電流	$\pm 50A$
位相数	2
スイッチング周波数	93.75kHz
電流レギュレーション誤差	$\pm 20mA$ (0.02% FS) 未満
電圧レギュレーション誤差	$\pm 1mV$ (0.02% FS) 未満

## 2 システム概要

### 2.1 ブロック図

図 2-1 は、リファレンス・デザインのブロック図です。TMS320F280039 マイコンは、同期整流降圧電力段用の高分解能 16 ビット PWM を生成し、電流および電圧制御機能を実行します。INA188 計測アンプが電流を検出し、OPA2182 オペアンプが電圧を検出します。電流および電圧信号は、外部の ADS131M08 ADC によってデジタル・データに変換されます。C2000 オンチップ・ウィンドウ・コンパレータを使用して、過電流保護を実装しています。

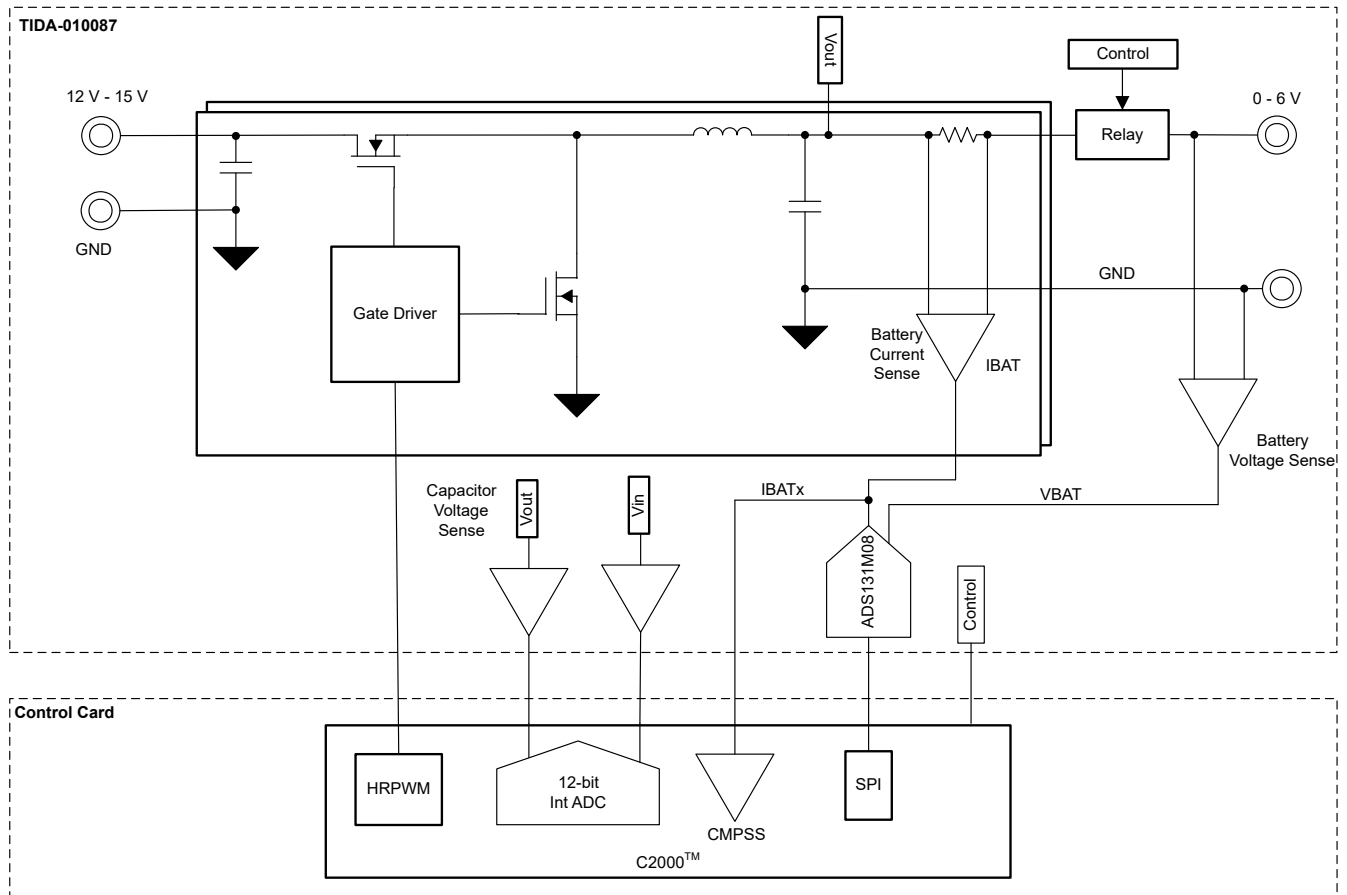


図 2-1. TIDA-010087 のブロック図

## 2.2 設計上の考慮事項

### 2.2.1 電流および電圧コントローラ

電流および電圧制御ループのソフトウェア実装を、図 2-2 に示します。電圧ループを電流にカスケード接続することで、充電モードと放電モードで定電流と定電圧の両方を実現します。バッテリー電圧が定電圧設定 (VSET) から離れている場合、電圧ループが定電流設定 (ISET) に飽和します。バッテリー電圧が VSET に近い値に達すると、電圧ループが閉じられ、ISET が低下して、バッテリー電圧が VSET 制限を超えないようにします。コントローラは充電モードと放電モードの両方で動作します。充電モードでは、VSET によって最大バッテリー電圧が制限されるため、充電が停止します。放電モードでは、VSET によって放電が停止する最小バッテリー電圧が制限されます。

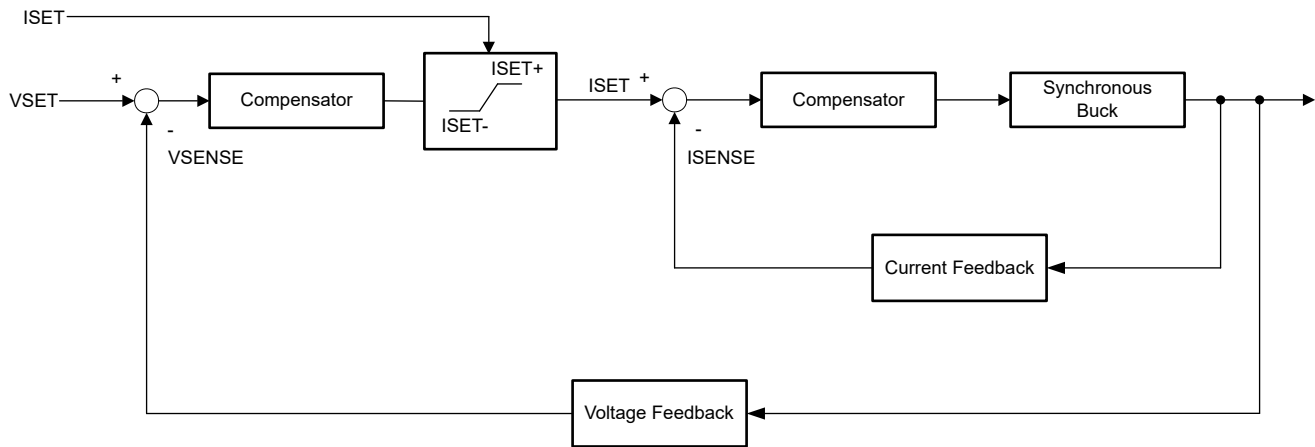


図 2-2. 電流および電圧コントローラ

### 2.2.2 高分解能 PWM 生成

高分解能を生成するには、高分解能 PWM 出力機能を持つ C2000 を使用しています。高分解能カウンタは 150ps のタイム・ステップを実行でき、これは 100MHz の CPU クロックの 100kHz PWM 周波数における 16 ビットの分解能に相当します。さまざまなスイッチング周波数での PWM 分解能を、表 2-1 に示します。

表 2-1. PWM と HRPWM の C2000™ マイコンの分解能

PWM 周波数	通常の分解能 (PWM)		高分解能 PWM	
	100MHz EPWMCLK			
	ビット	%	ビット	%
(kHz)				
20	12.3	0.02	18.1	0
50	11	0.05	16.8	0.001
100	10	0.1	15.8	0.002
150	9.5	0.15	15.2	0.003
200	9	0.2	14.8	0.004
250	8.6	0.25	14.4	0.005

## 2.3 主な使用製品

### 2.3.1 TMS320F280039

TMS320F280039 C2000 デバイスを使用して、同期整流降圧電力段を制御します。このデバイスには、8 つの HRPWM チャネルと 4 つのバッテリー・テスト・チャネルまたは降圧コンバータを制御するのに十分な 4 つのウィンドウ付きコンパレータが搭載されています。詳細については、『TMS320F28003x リアルタイム・マイクロコントローラ』のデータシートを参照してください。

### 2.3.2 ADS131M08

ADS131M08 は 8 チャネルの同時サンプリング、24 ビット、デルタ・シグマ ( $\Delta\Sigma$ ) アナログ / デジタル・コンバータ (ADC) で、 $\pm 0.01\%$  の精度と 1kHz のループ帯域幅に十分な最大 32ksps の最大サンプル・レートを使用できます。

## 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 ハードウェア要件

図 3-1 に TIDA-010087 ハードウェアを示します。TIDA-010087 基板を動作させるには [F280039C controlCARD 評価基板](#) が必要です。

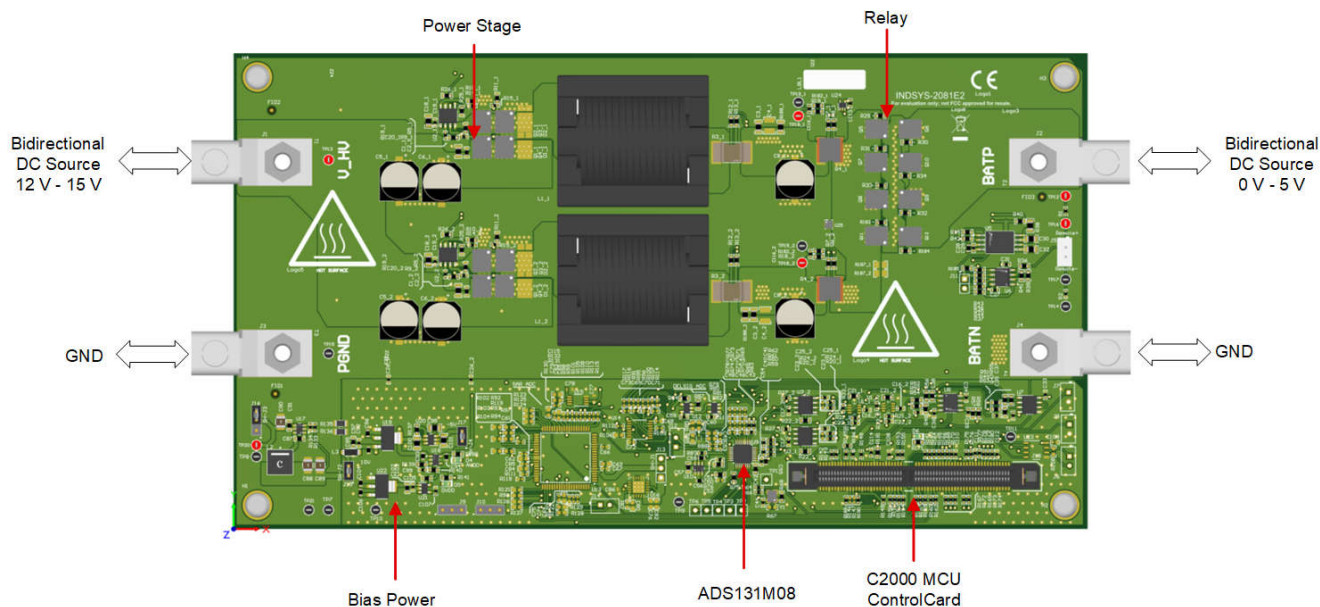


図 3-1. 基板の概要

### 3.2 ソフトウェア要件

この設計ソフトウェアは、C2000 マイコン向けの DigitalPower ソフトウェア開発キット (SDK) ([C2000WARE-DIGITALPOWER-SDK](#)) で供給され、powerSUITE フレームワーク内でサポートされています。

#### 3.2.1 Code Composer Studio 内でプロジェクトを開く

CCS でプロジェクトを開始するには、次の手順に従います。

1. [Code Composer Studio \(CCS\) 統合開発環境 \(IDE\)](#) ツール・フォルダから Code Composer Studio をインストールします。バージョン 12.3 またはそれ以降をお勧めします。
2. [C2000WARE-DIGITALPOWER-SDK](#) を次の 2 つのいずれかの方法でインストールします。
  - a. CCS にアクセスし、[View] → [Resource Explorer] をクリックします。テキサス・インスツルメンツの Resource Explorer 下で C2000WARE-DIGITAL-POWER-SDK にアクセスし、[Install] ボタンをクリックします。
  - b. C2000Ware Digital Power SDK ツール・フォルダからダウンロードします。
3. インストールが完了したら、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 を再起動しなければならない場合があります。

#### 注

デフォルトでは、powerSUITE は SDK のインストールと同時にインストールされます。

ファームウェア・プロジェクトは、次のいずれかの方法でインポートできるようになりました。

#### • Resource Explorer を使用する

1. Resource Explorer の C2000WARE-DIGITAL-POWER-SDK で、[powerSUITE] → [Solution Adapter Tool] をクリックします。
2. [DC-DC] セクションに表示される設計のリストから TIDA-010087 を選択します。

3. 開発キット・ページが表示されます。プロジェクトを実行するためのアイコンがトップ・バーに表示されます。[Run 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、GUI が図 3-2 のような設定ページが表示されます。
  5. この GUI ページが表示されない場合は、C2000WAREDIGITAL-POWER-SDK Resource Explorer の powerSUITE 下の FAQ セクションを参照してください。
- ソリューション・フォルダから直接インポートする
    1. CCS 内で [Project] → [Import CCS Projects] をクリックし、/solutions/tida\_010087/f28003x/ccs にあるソリューション・フォルダを参照して、プロジェクトを直接インポートすることもできます
    2. 2 つのプロジェクト仕様が表示されます。1 つは powerSUITE 付きで、もう 1 つは powerSUITE なしです。いずれかをクリックすると、プロジェクトの自己完結型フォルダが作成され、その中にすべての依存関係が含まれます。
    3. powerSUITE のないプロジェクトは、powerSUITE GUI の制限を見つけたお客様、または量産コードから powerSUITE を削除したいお客様向けに提供されています。
    4. このドキュメントでは powerSUITE プロジェクトについて説明しますが、powerSUITE のないプロジェクトでは、この設計ガイドに記載されている powerSUITE settings.h ファイルの関連する #defines に変更を加えることで、すべての手順を繰り返すことができます。

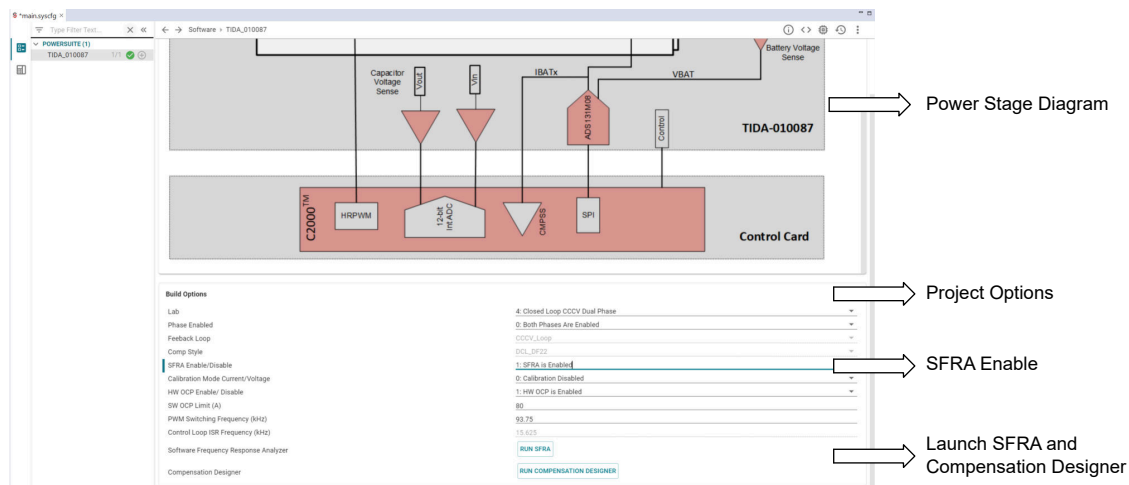


図 3-2. 設計の powerSUITE ページ

### 3.2.2 プロジェクト構造

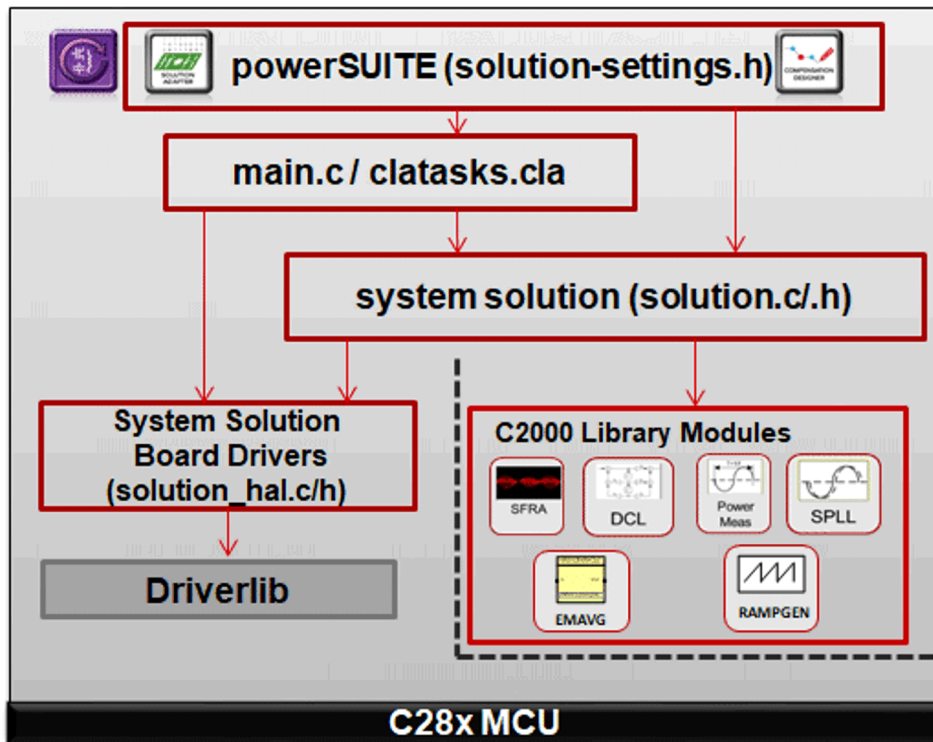


図 3-3. プロジェクト構造の概要

プロジェクトの一般構造を、図 3-3 に示します。プロジェクトがインポートされると、図 3-4 に示すように CCS 内に Project Explorer が表示されます。

#### 注

図 3-4 は F28003x 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。

コア・アルゴリズム・コードで構成されるソリューション別およびデバイスに依存しないファイルは .c/h にあります。

基板別でデバイス別のファイルは \_hal.c/h にあります。このファイルは、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。別の変調方式や別のデバイスを使用する場合は、プロジェクト内のデバイス・サポート・ファイルを変更するだけでなく、これらのファイルに変更を加えるだけで済みます。

-main.c ファイルは、プロジェクトのメイン・フレームワークで構成されています。このファイルは、システム・フレームワークの作成に役立つ基板とソリューション・ファイルへの呼び出し、割り込みサービス・ルーチン (ISR) と低速なバックグラウンド・タスクで構成されています。

この設計では、ソリューションは bt2ph です。

powerSUITE ページは、Project Explorer に表示される main.syscfg 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 \_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 C 言語を使用したファイルです。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、このファイルを手動で変更しないでください。\_user\_settings.h は \_settings.h に含まれており、ADC マッピングの #defines や GPIO など、powerSUITE ツールの範囲外の設定を保持するために使用できます。

\_cal.h ファイルは、電流と電圧を測定するためのゲイン値とオフセット値で構成されています。

Kit.json ファイルと solution.js ファイルは、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。

ソリューション名は、ソリューションで 사용되는すべての変数および定義のモジュール名としても使用されます。したがって、すべての変数および関数呼び出しの先頭には **BT2PH** 名が付きます (たとえば、**BT2PH\_userParam\_V\_I\_ch1**)。この命名規則により、ユーザーは名前の競合を回避しながら、異なるソリューションを組み合わせることができます。

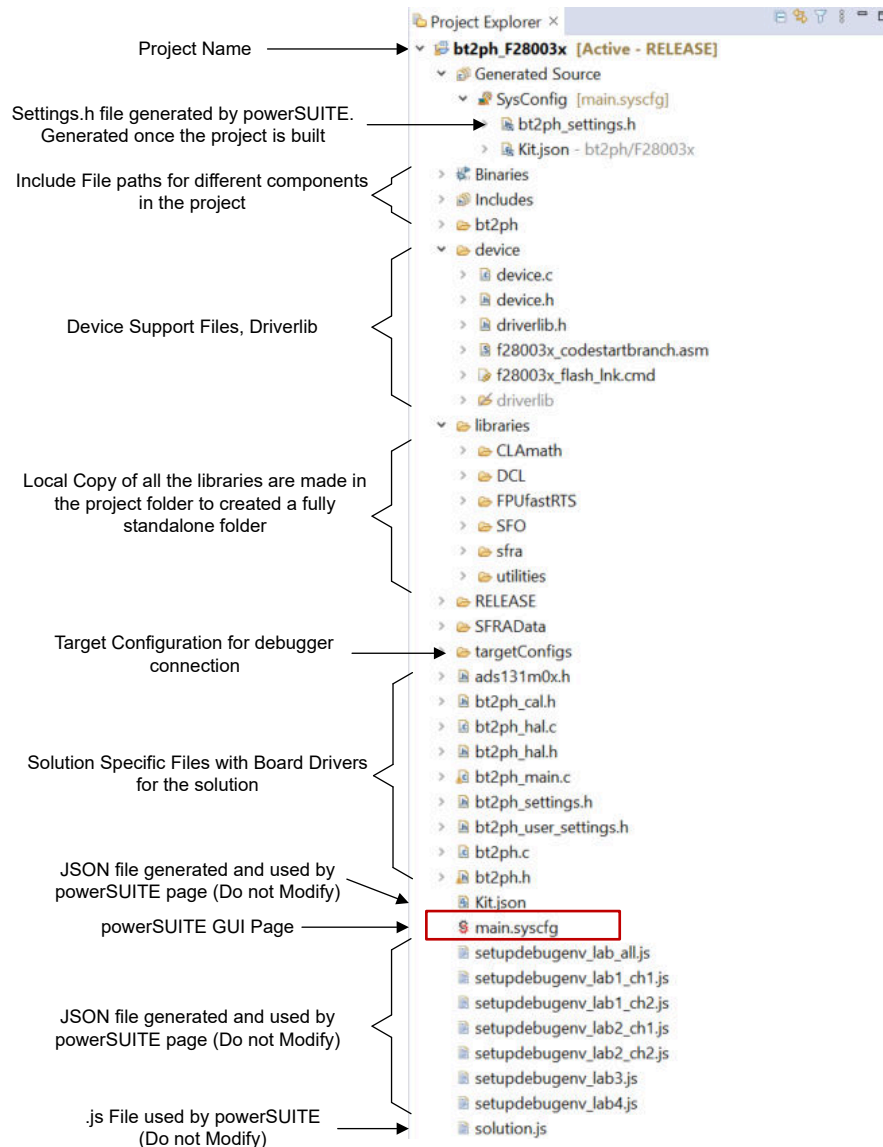


図 3-4. BT2PH プロジェクトの [Project Explorer] ビュー

BT2PH プロジェクトは 3 つの ISR (ISR3、ISR4、および ISR7) で構成されています。ISR1、ISR2、ISR5、および ISR6 は、今後の使用のために予約されています。

降圧コンバータの入力電圧とコンデンサ電圧を検出するには、**ISR3** を使用します。ISR3 は、ADCC 変換完了によってトリガされます。ADCC を使用してコンバータの入力電圧と出力電圧を検出し、DC/DC のソフト・スタートを実装します。

**ISR4** は、ADS131M08 の DRDY (データ・レディ) 信号によってトリガされます。外部 ADC は 15.625kHz のサンプル・レートにプログラムされ、これによって ISR 周波数が設定されます。ISR は、電流および電圧制御ループ機能を実行します。

**ISR7** は、SPI 受信 FIFO 割り込みによってトリガされます。ISR を使用して、FIFO レジスタから外部 ADC データを読み出します。

図 3-5 および 図 3-6 に、ISR3、ISR4、および ISR7 の所要時間を示します。3 つの ISR に要する合計時間は 8 $\mu$ s 未満です。ISR は、15.625kHz の ISR 周波数の場合、CPU リソースの 12.5% を消費します。



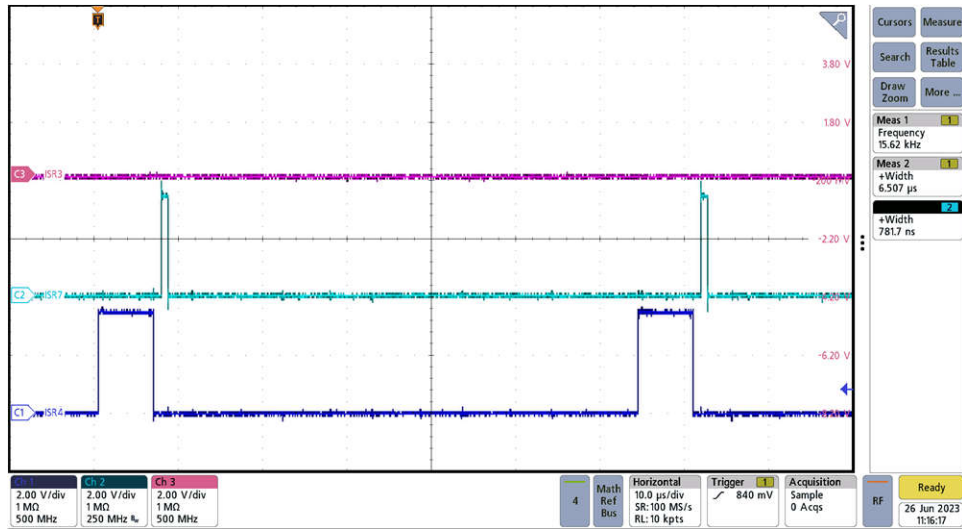


図 3-5. ISR4 および ISR7 の実行時間測定

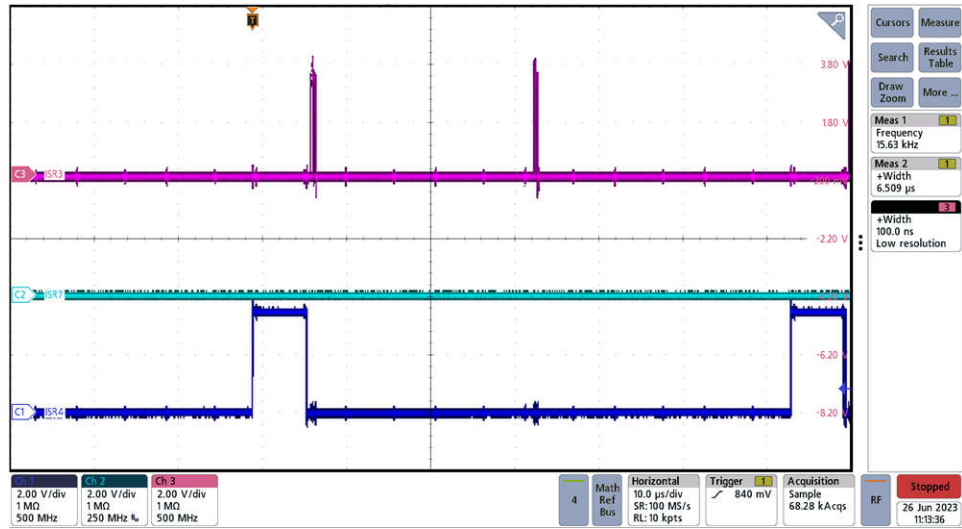


図 3-6. ISR4 および ISR3 の実行時間測定

### 3.2.3 ソフトウェア・フロー図

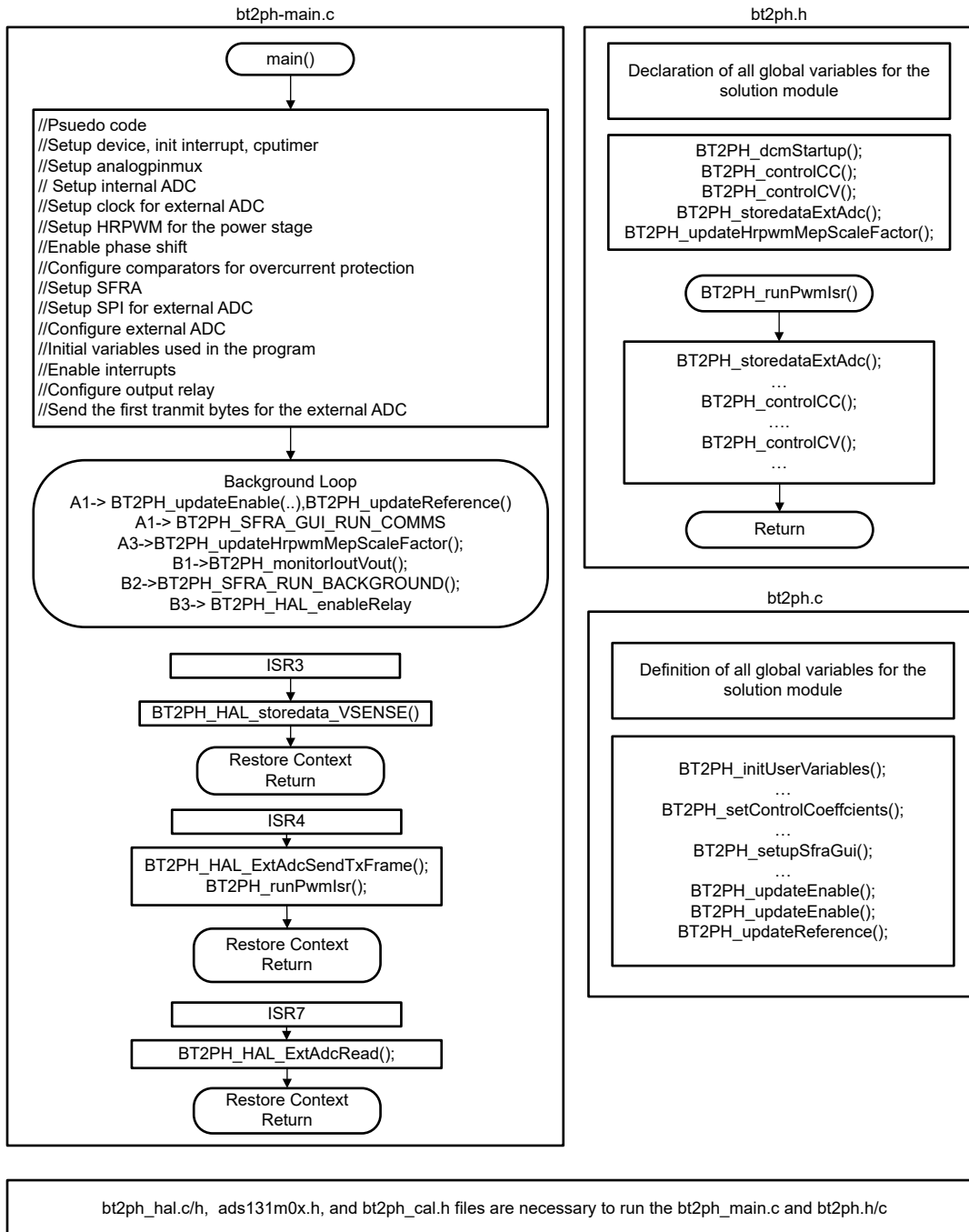


図 3-7. ソフトウェア・フロー図

### 3.3 テスト設定

#### 3.3.1 電流および電圧ループをチューニングするためのハードウェア設定

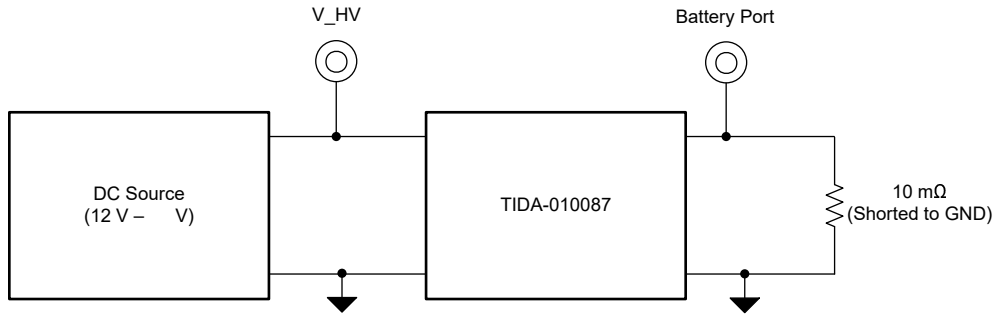


図 3-8. 電流および電圧ループをチューニングするためのハードウェア設定

#### 3.3.2 双方向の電力フローをテストするためのハードウェア設定

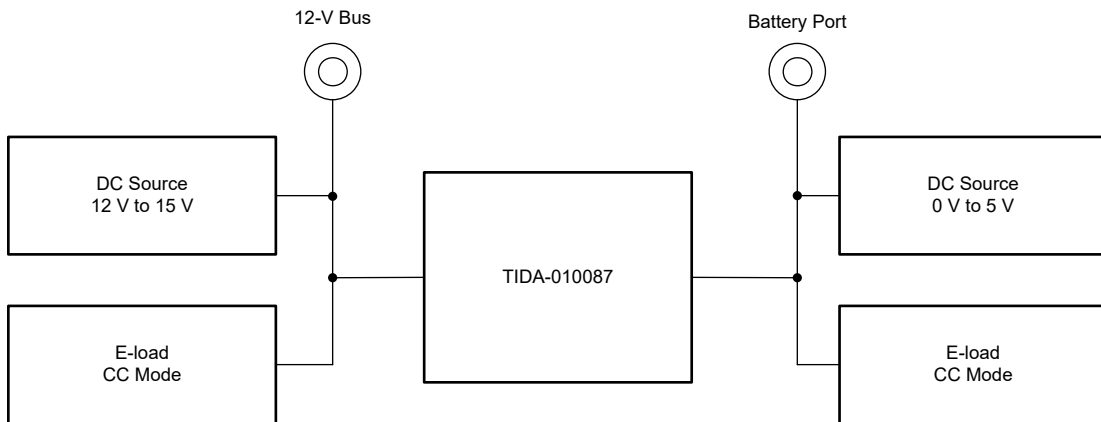


図 3-9. 双方向の電力フローをテストするためのハードウェア設定

#### 3.3.3 電流および電圧キャリブレーションのハードウェア設定

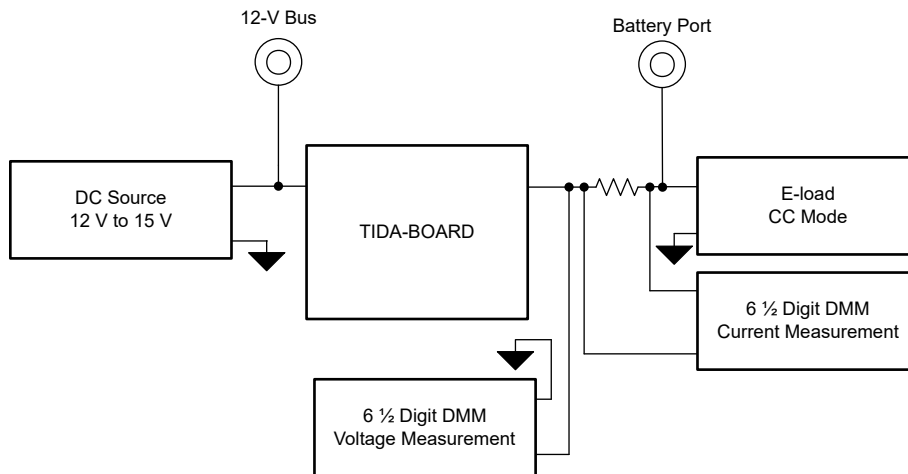


図 3-10. 電流および電圧キャリブレーションのハードウェア設定

### 3.4 テスト方法

#### 3.4.1 ラボ変数の定義

BT2PH\_userParam\_V\_I\_chx パラメータは、さまざまなラボで電力段を制御するために使用します。ラボ 1 とラボ 2 では、DC/DC コンバータの位相 1 と位相 2 を制御するために、BT2PH\_userParam\_V\_I\_ch1 と BT2PH\_userParam\_V\_I\_ch2 を使用します。ラボ 3 とラボ 4 では、BT2PH\_userParam\_V\_I\_chm 変数を使用します。パラメータの定義については、表 3-1 を参照してください。

**表 3-1. BT2PH\_userParam 定義**

BT2PH_userParam	データの種類	備考
iref_A	フローティング	充電モードと放電モードの両方の電流を設定 [0, 100]
vrefCharge_V	フローティング	充電モードで電圧を設定 [0, 5]
vrefDischarge_V	フローティング	放電モードで電圧を設定 [0, 5]
dir_bool	符号なし整数	充電モードの場合、このパラメータを 1 に設定します 放電モードの場合、このパラメータを 0 に設定します
en_bool	符号なし整数	チャンネルをイネーブルにする場合、このパラメータを 1 に設定します
dutyRef_pu	フローティング	開ループ・モードの基準デューティ・サイクル範囲 = 0~1.0
ibatCal_pu	フローティング	キャリブレーション・モードで出力電流を設定するには、このパラメータを使用します。範囲 = 0~1.0
vbatCal_pu	フローティング	キャリブレーション・モードで出力電圧を設定するには、このパラメータを使用します。範囲 = 0~1.0
loutGain_pu	フローティング	変数には、電流ゲインのキャリブレーション・データが格納されません
ioutOffset_pu	フローティング	変数には、現在のオフセットのキャリブレーション・データが格納されます
loutGain_A	フローティング	変数には、電流ゲインのキャリブレーション・データが格納されません
loutOffset_A	フローティング	変数には、現在のオフセットのキャリブレーション・データが格納されます
vbatGain_pu	フローティング	変数には、電圧ゲインのキャリブレーション・データが格納されません
vbatOffset_pu	フローティング	変数には、電圧オフセットのキャリブレーション・データが格納されます
vbatGain_V	フローティング	変数には、電圧ゲインのキャリブレーション・データが格納されません
vbatOffset_V	フローティング	変数には、電圧オフセットのキャリブレーション・データが格納されます

#### 3.4.2 ラボ 1.開ループ電流制御単相

##### 3.4.2.1 ラボ 1 のソフトウェア・オプションの設定

1. [セクション 3.2.1](#) の概要に従って CCS プロジェクトを開きます。powerSUITE を使用している場合は、[手順 2](#) に進みます。それ以外の場合は[手順 3](#) に進みます。
2. SYSCONFIG ページを開き、[Build Options] セクションで以下を選択します。
  - ラボは [Lab 1:Open Loop CC Single Phase] を選択します。
  - [Phase Enabled] を [Phase 1] または [Phase 2] に変更します。
  - [SFRA Enable/Disable] を [1] に設定します。
  - ページを保存します。

3. powerSuite のないバージョンのプロジェクトを使用する場合、上記の設定は `solution_settings.h` ファイルで直接変更されます。

```
#define LAB_NUMBER (1)
#define PHASE_NUMBER (1)
#define SFRA_ENABLED (true)
```

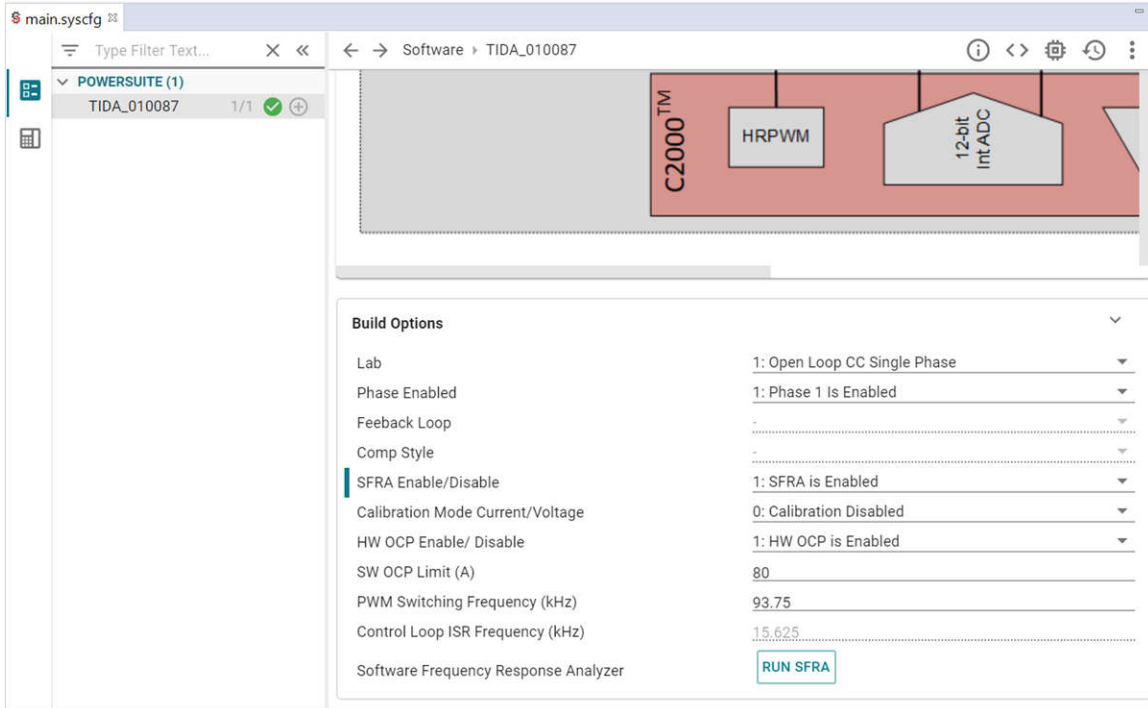


図 3-11. ラボ 1 のビルド・オプション


### 3.4.2.2 プロジェクトのビルドおよびロードとデバッグ環境の設定

プロジェクトをビルドおよびロードし、デバッグ環境を設定するには、次の手順を実行します。

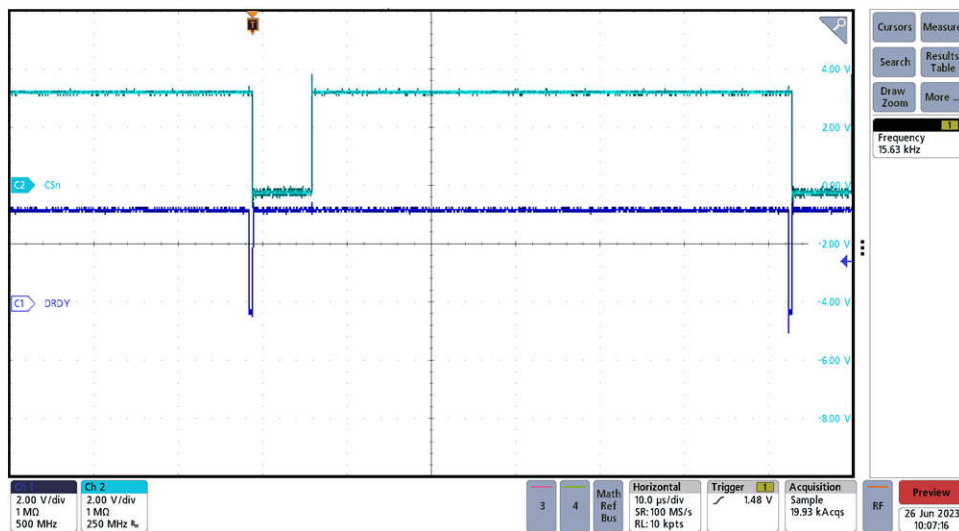
1. プロジェクト名を右クリックし、[Rebuild Project] をクリックします。
2. プロジェクトが正常にビルドされます。
3. Project Explorer で、targetConfigs 下で適切な目標構成ファイルが有効になっていることを確認します。
4. 次に、[Run] → [Debug] をクリックしてデバッグ・セッションを起動します。
5. するとプロジェクトがデバイスにロードされ、CCS デバッグ・ビューが有効になります。メイン・ルーチンの開始時にコードは停止します。
6. [Watch] および [Expressions] ウィンドウに変数を追加するには、[View] → [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_lab1_ch1.js` スクリプト・ファイルを参照します。これにより、[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。フェーズ 2 では、`setupdebugenv_lab2_ch2.js` スクリプト・ファイルを選択します。
7. [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。

### 3.4.2.3 コードの実行

ラボ 1 のコードを実行するには、次の手順に従います。

1. セクション 3.3.1 に示すようにテスト設定を使用します。
2. メニュー・バーの  をクリックしてプロジェクトを実行します。

3. 監視ビューで、[Expression] ウィンドウで BT2PH\_InputVoltageSense\_V が 12V~15V の範囲内にあるかどうかを確認します。
4. オシロスコープを使用して、周波数が 15.625kHz の場合に外部 ADC の DRDY 信号をチェックします。マイコンが動作しているときの ADS131M08 の DRDY および CS 信号を、[図 3-12](#) に示します。
5. [Expression] ウィンドウで次のパラメータを設定します。
  - BT2PH\_userParam\_V\_I\_ch1->dutyRef\_pu = 0.03
  - BT2PH\_userParam\_V\_I\_ch1->en\_bool = 1 を設定します
  - 「BT2PH\_enableRelay\_bool」を 1 に設定して、出力リレーをイネーブルにします
  - [Expression] ウィンドウの設定については、[図 3-13](#) を参照してください
6. BT2PH\_measureMultiphase\_V\_I 変数は、DC/DC コンバータの出力電流と電圧を示します。BT2PH\_userParam\_V\_I\_ch1->dutyRef\_pu を調整し、電流が約 15A であることを確認します。
7. **開ループ電流制御用の SFRA 設定** に、開ループ電流制御用のプラント・モデルを抽出するための SFRA 設定を示します。SYSCONFIG ページで Run SFRA アイコンをクリックします。SFRA GUI がポップアップ表示されます。
8. SFRA GUI でデバイスのオプションを選択します。たとえば、F280039 の場合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] ボタンをクリックします。SFRA GUI に戻り、[Connect] ボタンをクリックします。
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 掃引を開始できるようになりました。SFRA 掃引が完了するまでには数分かかります。完了すると、[図 3-15](#) に示すように測定値が表示されたグラフが表示されます。
10. 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 3-12. 外部 ADC の CSn 信号と DRDY 信号**

Expression	Type	Value
BT2PH_LabNumber	enum <unnamed>	Lab1_OpenLoopSinglePh...
BT2PH_SfraStatus	enum <unnamed>	SFRA_Enabled
BT2PH_CalibrationStatus	enum <unnamed>	Calibration_Disabled
BT2PH_CalibrationMode	enum <unnamed>	0
BT2PH_InputVoltageSense_V	float	12.3468018
BT2PH_enableRelay_bool	unsigned int	1
BT2PH_userParam_V_I_ch1	struct <unnamed>	{iref_A=1.0,vrefCharge_V...
iref_A	float	1.0
vrefCharge_V	float	4.19999981
vrefDischarge_V	float	2.79999995
dir_bool	unsigned int	1
en_bool	unsigned int	1
dutyRef_pu	float	0.0299999993
ibatCal_pu	float	0.0
vbatCal_pu	float	0.0
ioutGain_pu	float	0.0185729992
ioutOffset_pu	float	0.000677544624
ioutGain_A	float	53.8416023
ioutOffset_A	float	-0.0364800878
vbatGain_pu	float	0.167084396
vbatOffset_pu	float	0.000334143639
vbatGain_V	float	5.98499918
vbatOffset_V	float	-0.00199984945
BT2PH_measureMultiphase_V_I	struct <unnamed>	{Isense1_A=14.8733797,l...
Isense1_A	float	14.8752794
Isense2_A	float	0.0186048299
Ibatsense_A	float	14.8938847
Voutsense_V	float	0.0
Vbatsense_V	float	0.10666696

図 3-13. ラボ 1 [Expression] ウィンドウ、開ループ

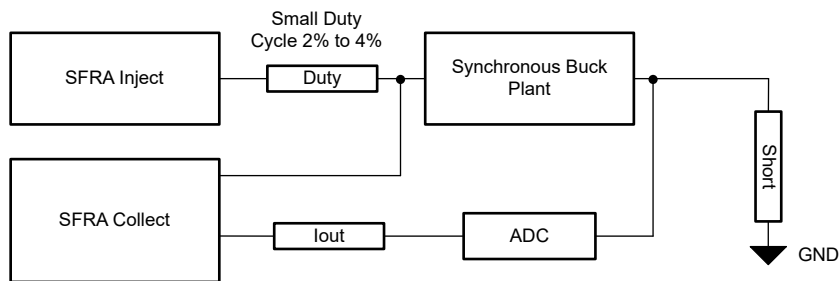


図 3-14. 開ループ電流制御用の SFRA 設定

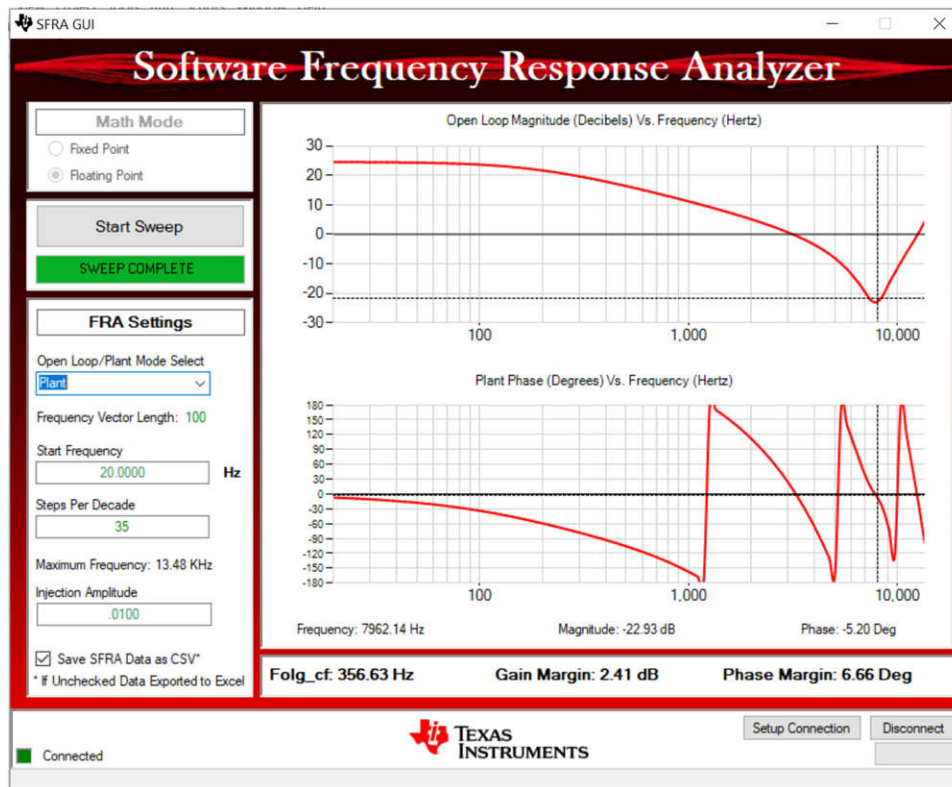



図 3-15. 電流制御の開ループ周波数応答

### 3.4.3 ラボ 2. 閉ループ電流制御単相

#### 3.4.3.1 ラボ 2 のソフトウェア・オプションの設定

- このラボを実行するには、前のセクション、[セクション 3.4.2](#) で説明したようにハードウェアが設定されていることを確認します。
- [セクション 3.2.1](#) の概要に従って CCS プロジェクトを開きます。powerSUITE を使用している場合は、[手順 3](#) に進みます。それ以外の場合は [手順 4](#) に進みます。
- SYSCONFIG ページを開き、[Build Options] セクションで以下を選択します。
  - ラボは [Lab 2: Closed Loop CC Single Phase] を選択します。
  - [Phase Enabled] を [Phase 1] または [Phase 2] に変更します。
  - [SFRA Enable/Disable] を [1] に設定します。
  - [Run Compensation Design] ボタンをクリックして、Compensation Designer  を開きます。
  - 補償デザイナーが起動し、有効な SFRA データ・ファイルを選択するように求められます。ラボ 1 の実行から補償デザイナーに SFRA データをインポートし、2 極、2 ゼロの補償器を設計します。この設計の繰り返し中により多くのマージンを確保して、ループが閉じたときにシステムが安定するようにします。
  - 電流ループの補償パラメータを、[図 3-16](#) に示します。
  - [Save Comp] ボタンをクリックして、補償を保存します。Compensation Designer ツールを閉じます。
  - SYSCONFIG ページを保存します。
- powerSUITE 以外のバージョンのプロジェクトを使用する場合、[Build Settings] は solution\_settings.h ファイルで直接変更されます。Compensation Designer は、C2000Ware\_DigitalPower\_Install\_Location\powerSUITE\source\utils に配置されています。

```
#define LAB_NUMBER (2)
#define PHASE_NUMBER (1)
#define SFRA_ENABLED (true)
```



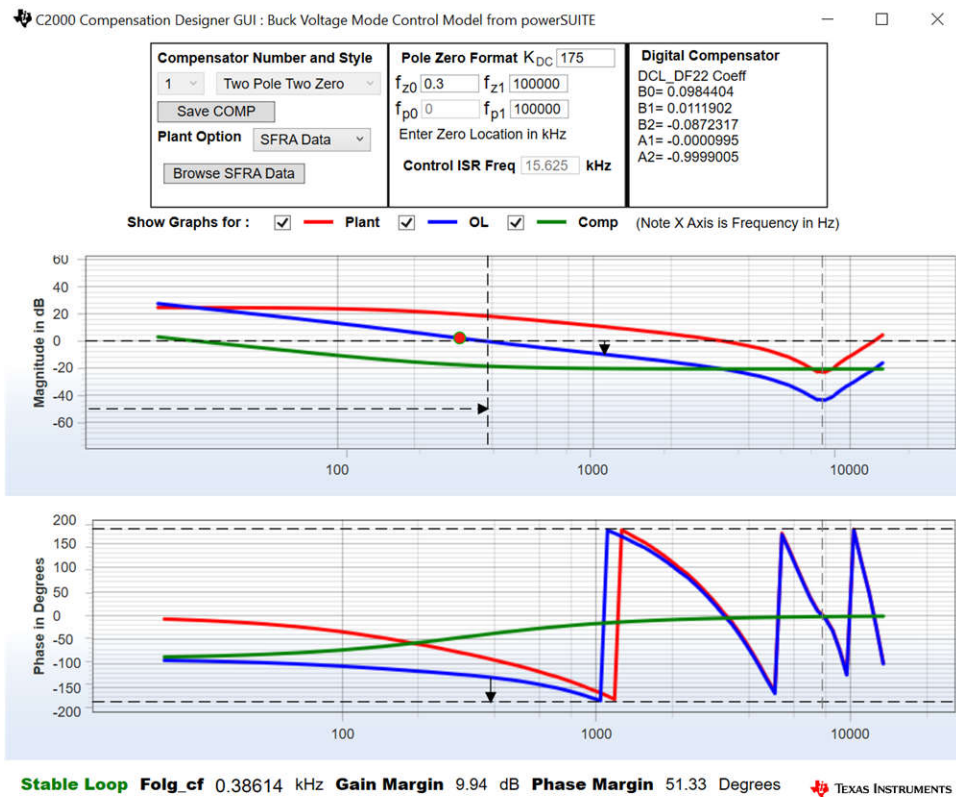


図 3-16. Compensation Designer を使用した電流ループのチューニング

### 3.4.3.2 プロジェクトのビルドおよびロードとデバッグ環境の設定

- 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。
- 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。
- 次に、[Run] → [Debug] をクリックしてデバッグ・セッションを起動します。
- 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。
- [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\_lab2\_ch1.js スクリプト・ファイルを参照します。これにより、[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。フェーズ 2 では、setupdebugenv\_lab2\_ch2.js スクリプト・ファイルを選択します。
- [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。

### 3.4.3.3 コードの実行

- このラボを実行するには、前のセクション、セクション 3.4.2 で説明したようにハードウェアが設定されていることを確認します。
- メニュー・バーの (▶) をクリックしてプロジェクトを実行します。
- 監視ビューで、[Expression] ウィンドウで BT2PH\_InputVoltageSense\_V が 12V~15V の範囲内にあるかどうかを確認します。
- [Expression] ウィンドウで次のパラメータを設定します。
  - BT2PH\_enableRelay\_bool を 1 に設定して、出力リレーをイネーブルにします。
  - BT2PH\_userParam\_V\_I\_ch1->iref\_A = 15.0。
  - BT2PH\_userParam\_V\_I\_ch1->en\_bool = 1 を設定します。

- [Expression] ウィンドウの設定については、[図 3-17](#) を参照してください。
- BT2PH\_measureMultiphase\_V\_I 変数は、DC/DC コンバータの出力電流と電圧を示します。Isense1\_A の表示値は iref\_A 設定に近く、誤差は  $\pm 1\text{mA}$  です。
  - ループ安定性をテストするための SFRA を [図 3-18](#) に示します。SYSCONFIG ページで Run SFRA アイコンをクリックします。SFRA GUI がポップアップ表示されます。
  - SFRA GUI でデバイスのオプションを選択します。たとえば、F280039 の場合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] ボタンをクリックします。
  - 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 掃引を開始できるようになりました。SFRA 掃引が完了するまでには数分かかります。完了すると、[図 3-19](#) に示すように測定値が表示されたグラフが表示されます。
  - 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。

Expression	Type	Value
BT2PH_labNumber	enum <unnamed>	Lab2_ClosedLoopSingleP...
BT2PH_SfraStatus	enum <unnamed>	SFRA_Enabled
BT2PH_CalibrationStatus	enum <unnamed>	Calibration_Disabled
BT2PH_CalibrationMode	enum <unnamed>	0
BT2PH_InputVoltageSense_V	float	12.3535156
BT2PH_enableRelay_bool	unsigned int	1
BT2PH_userParam_V_I_ch1	struct <unnamed>	{iref_A=15.0,vrefCharge_...
iref_A	float	15.0
vrefCharge_V	float	4.19999981
vrefDischarge_V	float	2.79999995
dir_bool	unsigned int	1
en_bool	unsigned int	1
dutyRef_pu	float	0.00999999978
ibatCal_pu	float	0.0
vbatCal_pu	float	0.0
ioutGain_pu	float	0.0185729992
ioutOffset_pu	float	0.000677544624
ioutGain_A	float	53.8416023
ioutOffset_A	float	-0.0364800878
vbatGain_pu	float	0.167084396
vbatOffset_pu	float	0.000334143639
vbatGain_V	float	5.98499918
vbatOffset_V	float	-0.00199984945
BT2PH_measureMultiphase_V_I	struct <unnamed>	{Isense1_A=14.9998608,I...
Isense1_A	float	14.9997711
Isense2_A	float	0.018064633
Ibatsense_A	float	15.0178356
Voutsense_V	float	0.0
Vbatsense_V	float	0.10604196

**図 3-17. ラボ 2 [Expression] ウィンドウ、閉ループ**

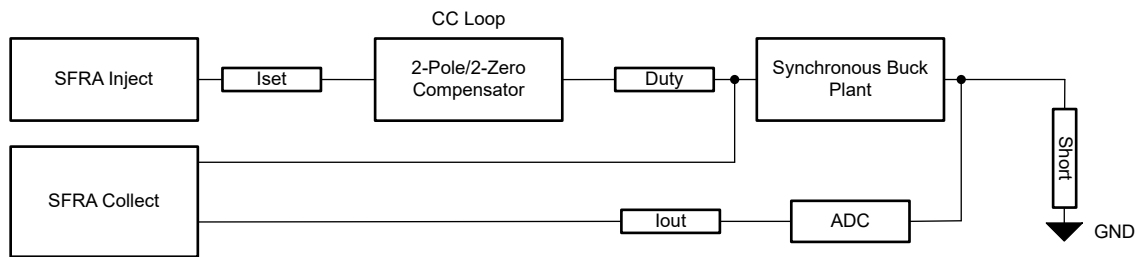


図 3-18. 閉ループ電流制御の SFRA 設定

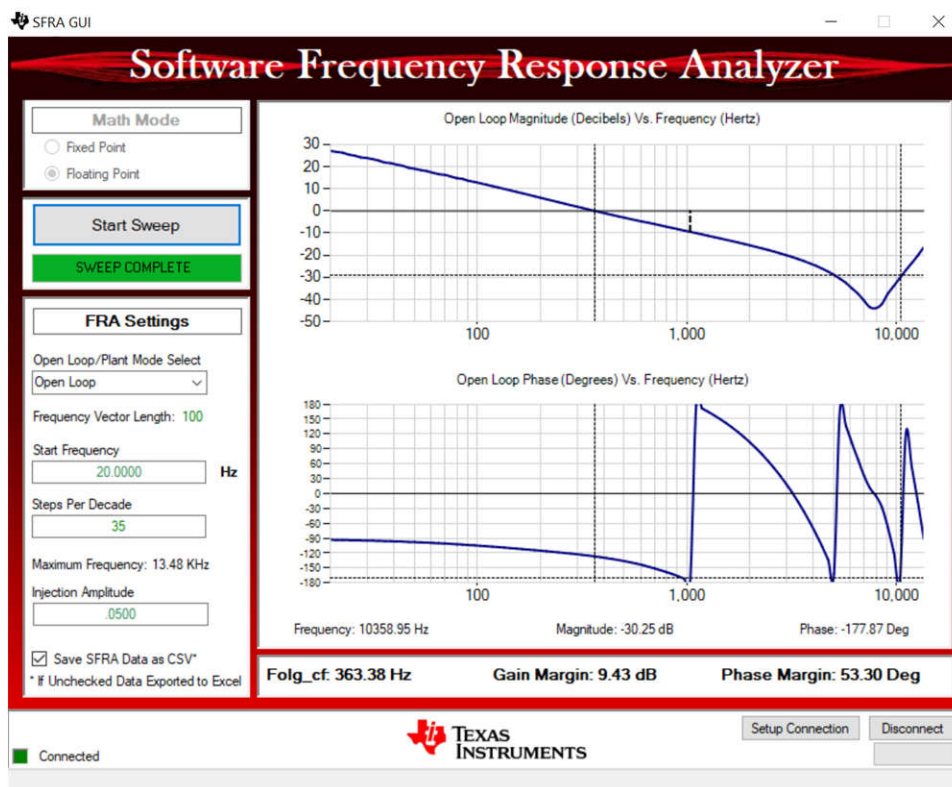


図 3-19. 電流制御の閉ループ周波数応答

### 3.4.3.4 電流キャリブレーション

- このラボを実行するには、[セクション 3.3.3](#) で説明したようにハードウェアが設定されていることを確認します。ゲイン誤差とオフセット誤差のキャリブレーションには、2 点較正法を使用します。
- 電流を測定するには、外付けの高精度抵抗と DMM を使用するか、E-Load 電流測定値を使用します。または、TIDA-010087 基板上のセンス抵抗の両端の電圧を使用して、出力電流を測定できます。
- SYSCONFIG ページを開き、[Build Options] セクションで以下を選択します。
  - ラボは [Lab 2: Closed Loop CC Single Phase] を選択します。
  - [Phase Enabled] を [Phase 1] に変更します。
  - 電流キャリブレーションは [Calibration Mode] を [1] に設定します。
  - SYSCONFIG ページを保存し、コードを実行します。
  - [Expression] ウィンドウを開きます。
  - 出力電流は、BT2PH\_userParam\_V\_I\_ch1->ibatCal\_pu パラメータを使用して更新します。
  - BT2PH\_enableRelay\_bool を 1 に設定して、出力リレーをイネーブルにします。
  - BT2PH\_userParam\_V\_I\_ch1->en\_bool = 1 を設定します。
  - BT2PH\_userParam\_V\_I\_ch1->ibatCal\_pu を「0.05」および「0.3」に設定し、出力電流読み取り値を書き留めます。
  - bt2ph\_cal.h ファイルの実際の出力電流読み取り値を更新します。

```
#define BT2PH_IBAT_ACTUAL_CH1_P1_A ((float32_t)2.6556)
#define BT2PH_IBAT_ACTUAL_CH1_P2_A ((float32_t)16.163)
#define BT2PH_IBAT_ACTUAL_CH2_P1_A ((float32_t)2.6556)
#define BT2PH_IBAT_ACTUAL_CH2_P2_A ((float32_t)16.163)
```

- コンバータのフェーズ 2 についても、この手順を繰り返します。
  - キャリブレーションをディセーブルにするには [Calibration Mode] を [0] に設定します。
4. powerSUITE 以外のバージョンのプロジェクトを使用する場合、[Build Settings] は solution\_settings.h ファイルで直接変更されます。

```
#define LAB_NUMBER (2)
#define PHASE_NUMBER (1)
#define CALIBRATION_ENABLED (true)
#define CALIBRATION_MODE (1)
```

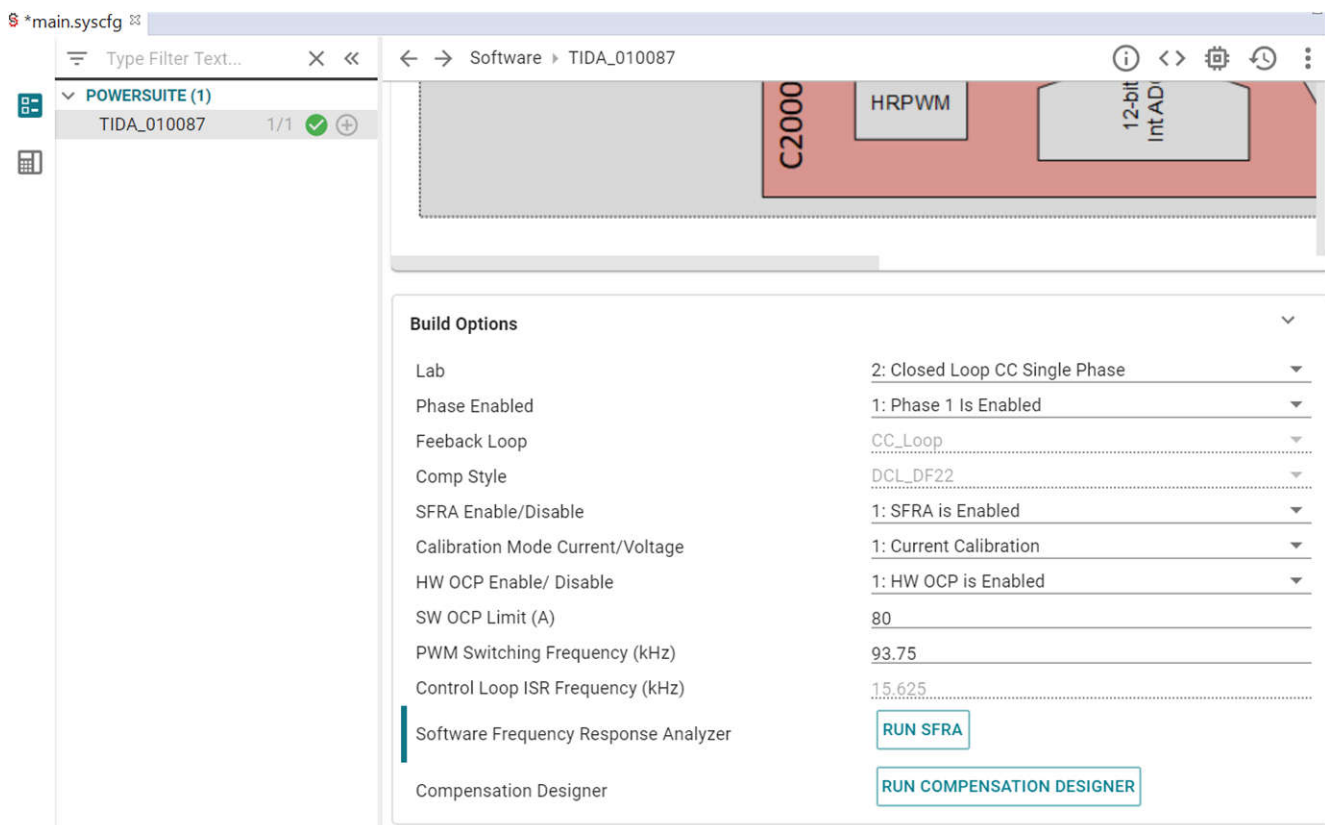


図 3-20. 電流キャリブレーションのビルド・オプション

### 3.4.4 ラボ 3.閉ループ電流制御 2 相


#### 3.4.4.1 ラボ 3 のソフトウェア・オプションの設定

1. セクション 3.3.1 に示すようにテスト設定を使用します。
2. セクション 3.2.1 の概要に従って CCS プロジェクトを開きます。powerSUITE を使用している場合は、手順 3 に進みます。それ以外の場合は手順 4 に進みます。
3. SYSCONFIG ページを開き、[Build Options] セクションで以下を選択します。
  - ラボは [Lab 3:Closed-Loop CC Dual Phase] を選択します。
  - 2 相モードは [Phase Enabled] を [0] に変更します。
  - [SFRA Enable/Disable] を [1] に設定します。


- ページを保存します。
4. powerSUITE 以外のバージョンのプロジェクトを使用する場合、上記の設定は `solution_settings.h` ファイルで直接変更されます。

```
#define LAB_NUMBER (3)
#define PHASE_NUMBER (0)
#define SFRA_ENABLED (true)
```

#### 3.4.4.2 プロジェクトのビルドおよびロードとデバッグ環境の設定

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。
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。
4. 次に、[Run] → [Debug] をクリックしてデバッグ・セッションを起動します。
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。
6. [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_lab3.js` スクリプト・ファイルを参照します。これにより、[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。
7. [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。

#### 3.4.4.3 コードの実行

1. セクション 3.3.1 に示すようにテスト設定を使用します。
2. メニュー・バーの  をクリックしてプロジェクトを実行します。
3. 監視ビューで、[Expression] ウィンドウで `BT2PH_InputVoltageSense_V` が 12V~15V の範囲内にあるかどうかを確認します。
4. [Expression] ウィンドウで次のパラメータを設定します。
  - `BT2PH_enableRelay_bool` を 1 に設定して、出力リレーをイネーブルにします。
  - `BT2PH_userParam_V_I_chm->iref_A` = 15.0。
  - `BT2PH_userParam_V_I_ch1->en_bool` = 1 を設定します。
  - [Expression] ウィンドウの設定については、[図 3-21](#) を参照してください。
5. `BT2PH_measureMultiphase_V_I` 変数は、DC/DC コンバータの出力電流と電圧を示します。`lbatsense_A` の表示値は `iref_A` 設定に近く、誤差は  $\pm 1\text{mA}$  です。
6. [図 3-22](#) に、開ループ電圧制御周波数応答を測定するための SFRA 設定を示します。
7. SYSCONFIG ページで Run SFRA アイコンをクリックします。SFRA GUI がポップアップ表示されます。
8. SFRA GUI でデバイスのオプションを選択します。たとえば、F280039 の場合は浮動小数点を選択します。[Setup Connection] ボタンをクリックします。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] ボタンをクリックします。
9. SFRA GUI がデバイスに接続します。これで [Start Sweep] ボタンをクリックして、SFRA 掃引を開始できるようになりました。SFRA 掃引が完了するまでには数分かかります。完了すると、[図 3-23](#) に示すように測定値が表示されたグラフが表示されます。
10. 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。

Expression	Type	Value
BT2PH_labNumber	enum <unnamed>	Lab3_ClosedLoopDualPh...
BT2PH_SfraStatus	enum <unnamed>	SFRA_Enabled
BT2PH_CalibrationStatus	enum <unnamed>	Calibration_Disabled
BT2PH_CalibrationMode	enum <unnamed>	0
BT2PH_InputVoltageSense_V	float	12.3400879
BT2PH_enableRelay_bool	unsigned int	1
BT2PH_userParam_V_I_chm	struct <unnamed>	{iref_A=15.0,vrefCharge_...
iref_A	float	15.0
vrefCharge_V	float	4.19999981
vrefDischarge_V	float	2.79999995
dir_bool	unsigned int	1
en_bool	unsigned int	1
dutyRef_pu	float	0.00999999978
ibatCal_pu	float	0.0
vbatCal_pu	float	0.0
ioutGain_pu	float	0.00927747134
ioutOffset_pu	float	0.000643853913
ioutGain_A	float	107.787994
ioutOffset_A	float	-0.0693997219
vbatGain_pu	float	0.167084396
vbatOffset_pu	float	0.000334143639
vbatGain_V	float	5.98499918
vbatOffset_V	float	-0.00199984945
BT2PH_measureMultiphase_V_I	struct <unnamed>	{Isense1_A=7.49065685,I...
Isense1_A	float	7.49119616
Isense2_A	float	7.50912523
Ibatsense_A	float	15.0003214
Voutsense_V	float	0.0
Vbatsense_V	float	0.106659822

図 3-21. ラボ 3 [Expression] ウィンドウ、閉ループ

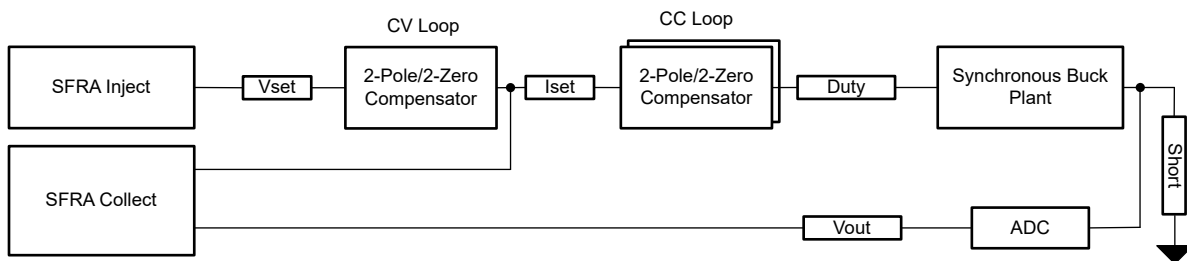


図 3-22. 開ループ電流制御の SFRA 設定

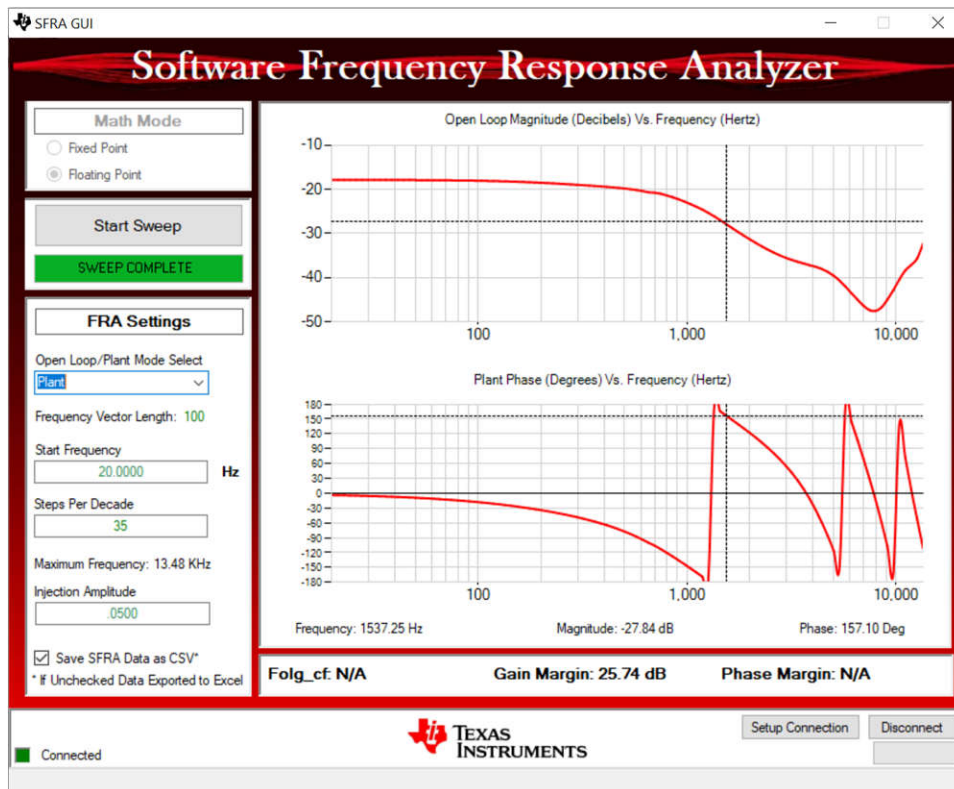



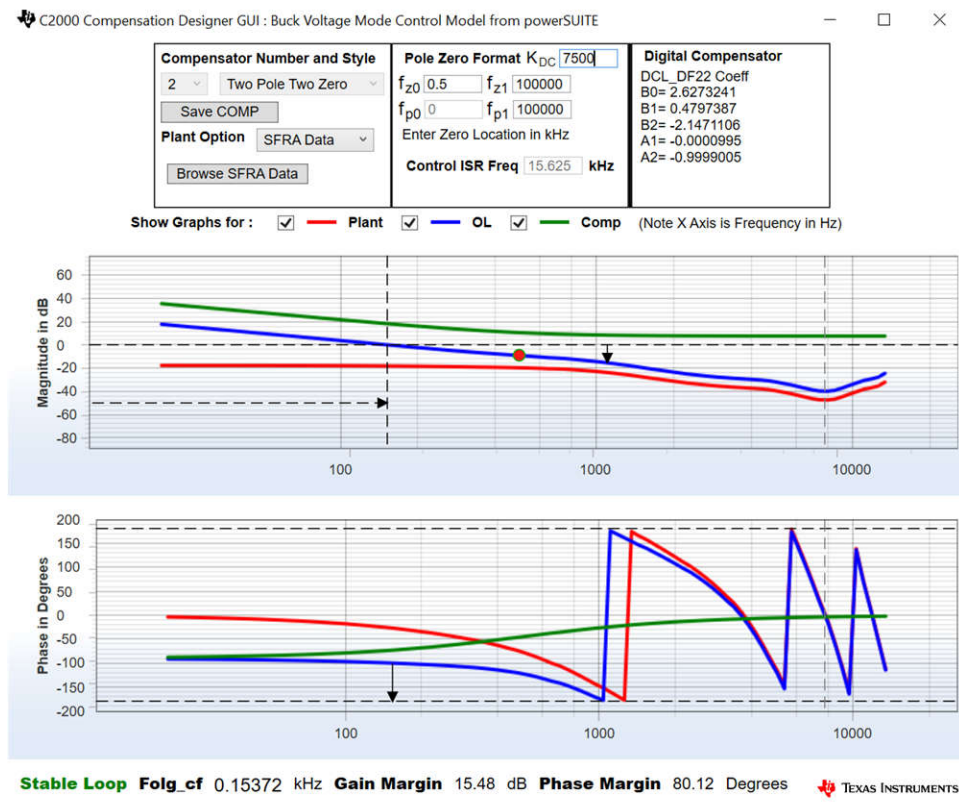
図 3-23. 電圧制御の開ループ周波数応答

### 3.4.5 ラボ 4.閉ループ電流および電圧制御

#### 3.4.5.1 ラボ 4 のソフトウェア・オプションの設定

1. セクション 3.3.1 に示すようにテスト設定を使用します。
2. セクション 3.2.1 の概要に従って CCS プロジェクトを開きます。powerSUITE を使用している場合は、手順 3 に進みます。それ以外の場合は手順 4 に進みます。
3. SYSCONFIG ページを開き、[Build Options] セクションで以下を選択します。
  - ラボは [Lab 4:Closed-Loop CCCV Dual Phase] を選択します。
  - 2 相動作は [Phase Enabled] を [0] に変更します。
  - [SFRA Enable/Disable] を [1] に設定します。
  - [Run Compensation Design] ボタンをクリックして、Compensation Designer  を開きます。
  - 補償デザイナーが起動し、有効な SFRA データ・ファイルを選択するように求められます。ラボ 1 の実行から補償デザイナーに SFRA データをインポートし、2 極、2 ゼロの補償器を設計します。この設計の繰り返し中により多くのマージンを確保して、ループが閉じたときにシステムが安定するようにします。
  - 電圧ループの補償パラメータを、図 3-24 に示します。
  - [Save Comp] ボタンをクリックして、補償を保存します。Compensation Designer ツールを閉じます。
  - SYSCONFIG ページを保存します。
4. powerSuite のないバージョンのプロジェクトを使用する場合、[Build Settings] は solution\_settings.h ファイルで直接変更されます。Compensation Designer は、C2000Ware\_DigitalPower\_Install\_Location\powerSUITE\source\utils に配置されています


```
#define LAB_NUMBER (4)
#define PHASE_NUMBER (0)
#define SFRA_ENABLED (true)
```


**図 3-24. Compensation Designer を使用した電圧ループのチューニング**

### 3.4.5.2 プロジェクトのビルドおよびロードとデバッグ環境の設定

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。
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。
4. 次に、[Run] → [Debug] をクリックしてデバッグ・セッションを起動します。
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。
6. [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\_lab4.js スクリプト・ファイルを参照します。これにより、[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。
7. [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。

### 3.4.5.3 コードの実行

1. セクション 3.3.1 に示すようにテスト設定を使用します。
2. メニュー・バーの  をクリックしてプロジェクトを実行します。
3. 監視ビューで、[Expression] ウィンドウで BT2PH\_InputVoltageSense\_V が 12V~15V の範囲内にあるかどうかを確認します。
4. [Expression] ウィンドウで次のパラメータを設定します。
  - BT2PH\_enableRelay\_bool を 1 に設定して、出力リレーをイネーブルにします。
  - BT2PH\_userParam\_V\_I\_chm->iref\_A = 30.0。
  - BT2PH\_userParam\_V\_I\_chm->vrefCharge\_V = 0.12。
  - BT2PH\_userParam\_V\_I\_ch1->en\_bool = 1 を設定します。
  - [Expression] ウィンドウの設定については、[図 3-25](#) を参照してください。



5. BT2PH\_measureMultiphase\_V\_I 変数は、DC/DC コンバータの出力電流と電圧を示します。Vbatsense\_V の表示値が vrefCharge\_V に近く、誤差は  $\pm 0.5\text{mV}$  です。
6. [図 3-26](#) に、閉ループ電圧制御周波数応答を測定するための SFRA 設定を示します。
7. SYSCONFIG ページで Run SFRA アイコンをクリックします。SFRA GUI がポップアップ表示されます。
8. SFRA GUI でデバイスのオプションを選択します。たとえば、F280039 の場合は浮動小数点を選択します。[Setup Connection] ボタンをクリックします。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] ボタンをクリックします。
9. SFRA GUI がデバイスに接続します。これで [Start Sweep] ボタンをクリックして、SFRA 掃引を開始できるようになりました。SFRA 掃引が完了するまでには数分かかります。完了すると、[図 3-27](#) に示すように測定値が表示されたグラフが表示されます。
10. 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。

Variables Expressions Registers		
Expression	Type	Value
BT2PH_labNumber	enum <unnamed>	Lab4_ClosedLoopDualPh...
BT2PH_SfraStatus	enum <unnamed>	SFRA_Enabled
BT2PH_CalibrationStatus	enum <unnamed>	Calibration_Disabled
BT2PH_CalibrationMode	enum <unnamed>	0
BT2PH_InputVoltageSense_V	float	12.3132324
BT2PH_enableRelay_bool	unsigned int	1
BT2PH_userParam_V_I_chm	struct <unnamed>	{iref_A=30.0,vrefCharge_...
iref_A	float	30.0
vrefCharge_V	float	0.119999997
vrefDischarge_V	float	2.79999995
dir_bool	unsigned int	1
en_bool	unsigned int	1
dutyRef_pu	float	0.00999999978
ibatCal_pu	float	0.0
vbatCal_pu	float	0.0
ioutGain_pu	float	0.00927747134
ioutOffset_pu	float	0.000643853913
ioutGain_A	float	107.787994
ioutOffset_A	float	-0.0693997219
vbatGain_pu	float	0.167084396
vbatOffset_pu	float	0.000334143639
vbatGain_V	float	5.98499918
vbatOffset_V	float	-0.00199984945
BT2PH_measureMultiphase_V_I	struct <unnamed>	{Isense1_A=8.51648712,I...
Isense1_A	float	8.51335526
Isense2_A	float	8.53360748
Ibatsense_A	float	17.0511131
Voutsense_V	float	0.0
Vbatsense_V	float	0.12000452

**図 3-25. ラボ 4 [Expression] ウィンドウ、閉ループ**

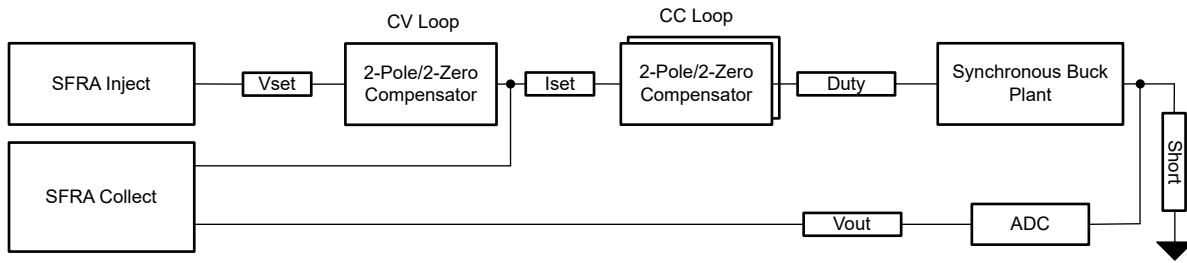


図 3-26. 閉ループ電流制御の SFRA 設定

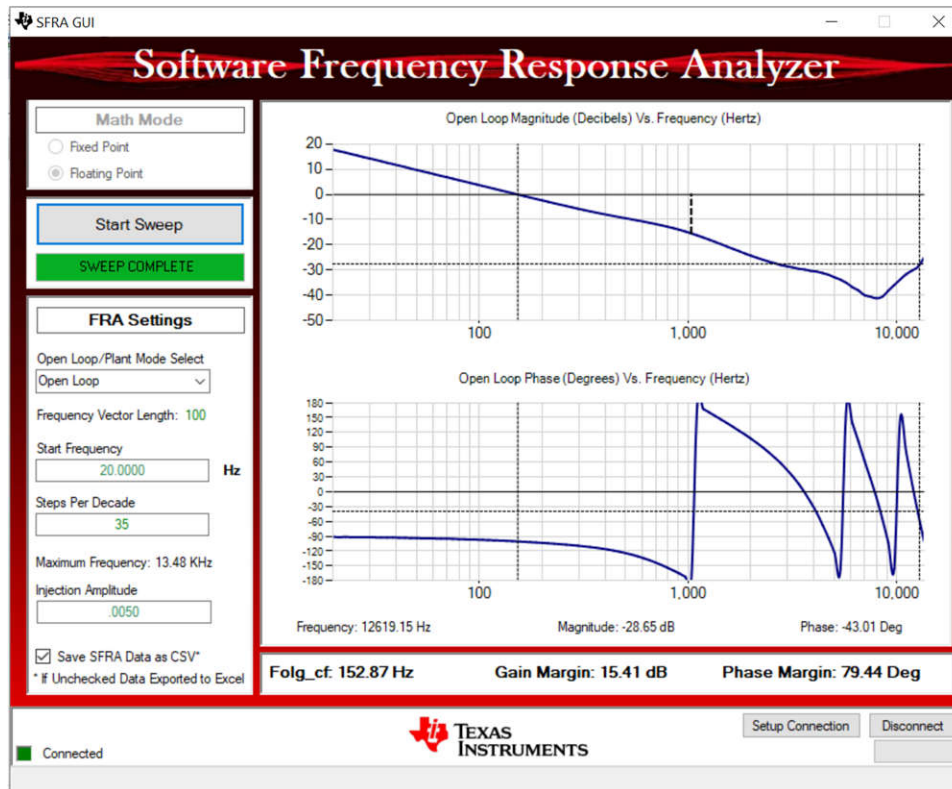


図 3-27. 電圧制御の閉ループ周波数応答

### 3.4.5.4 電圧キャリブレーション

- このラボを実行するには、[セクション 3.3.3](#) で説明したようにハードウェアが設定されていることを確認します。キャリブレーション中に E-Load をオフにすることができますが、そうでない場合、E-Load 電流設定を iref\_A より小さい値に設定して回路を定電圧モードにします。ゲイン誤差とオフセット誤差のキャリブレーションには、2 点キャリブレーション法を使用します。
- 電圧を測定するには、DMM を使用します。
- SYSCONFIG ページを開き、[Build Options] セクションで以下を選択します。
  - ラボは [Lab 4:Closed-Loop CCCV Dual Phase] を選択します。
  - 2 相モードは [Phase Enabled] を [0] に変更します。
  - 電圧キャリブレーションは [Calibration Mode] を [2] に設定します。
  - SYSCONFIG ページを保存し、コードを実行します。
  - [Expression] ウィンドウを開きます。
  - 出力電流は、BT2PH\_userParam\_V\_I\_chm->ibatCal\_pu パラメータを使用して更新します。
  - BT2PH\_enableRelay\_bool を 1 に設定して、出力リレーをイネーブルにします。
  - BT2PH\_userParam\_V\_I\_chm->en\_bool = 1 を設定します。
  - BT2PH\_userParam\_V\_I\_chm->vbatCal\_pu を「0.2」および「0.6」に設定し、出力電圧読み取り値を書き留めます。

- bt2ph\_cal.h ファイルの実際の出力電圧読み取り値を更新します。

```
#define BT2PH_VBAT_ACTUAL_CH1_P1_V ((float32_t)1.1995)
```

```
#define BT2PH_VBAT_ACTUAL_CH1_P2_V ((float32_t)3.599)
```

- キャリブレーションをディセーブルにするには [Calibration Mode] を [0] に設定します。

- powerSuite の古いバージョンのプロジェクトを使用する場合、[Build Settings] は solution\_settings.h ファイルで直接変更されます。

```
#define LAB_NUMBER (4)
```

```
#define PHASE_NUMBER (0)
```

```
#define CALIBRATION_ENABLED (true)
```

```
#define CALIBRATION_MODE (2)
```

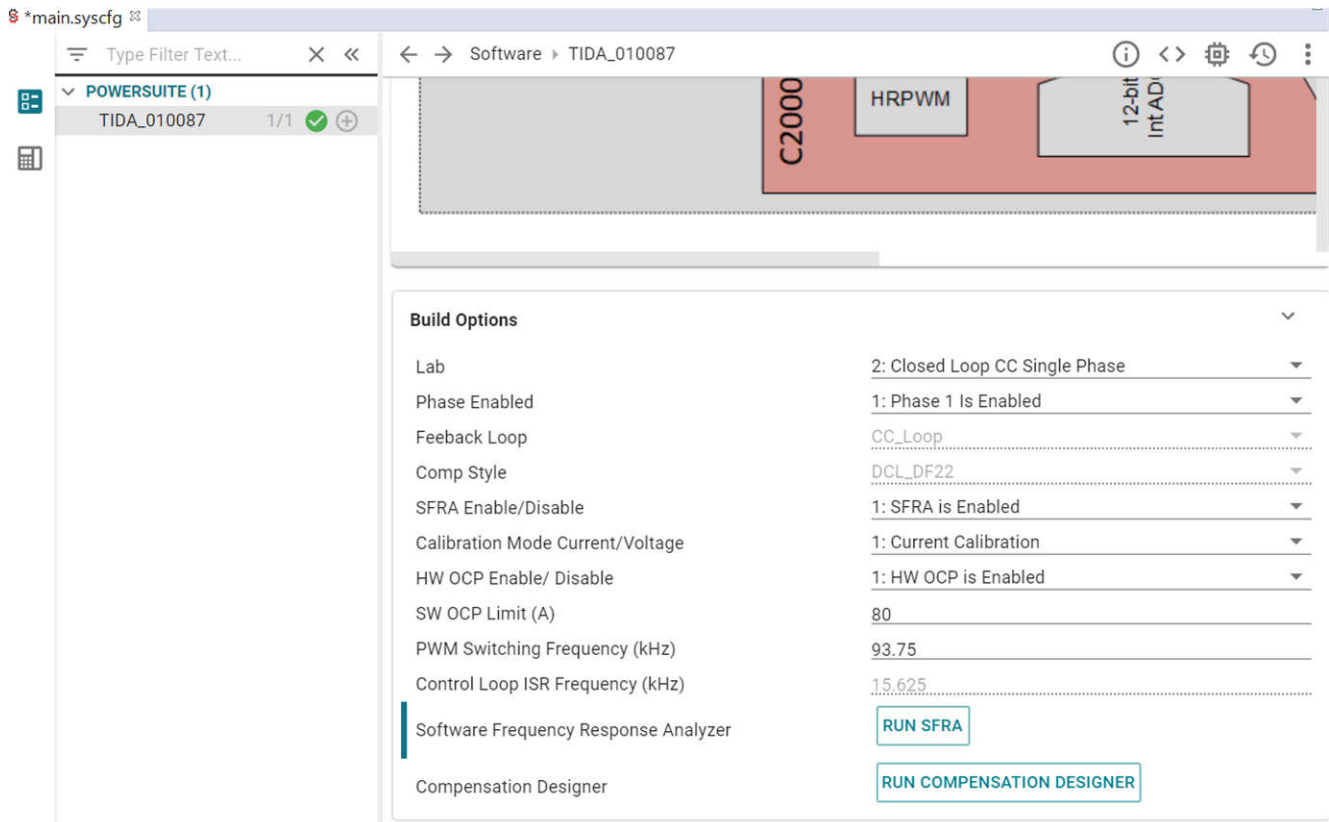


図 3-28. 電流キャリブレーションのビルド・オプション

## 3.5 テスト結果

### 3.5.1 電流ループ負荷レギュレーション誤差

表 3-2. 充電モードでの電流ループ負荷レギュレーション誤差

FSR (A)	100								
TIDA_010087 ISET (A)	0.1	0.5	1	5	10	20	25	30	39
ELOAD CV モード	E-LOAD 読み取り								
VSET (V)	lactual (A)	lactual (A)	lactual (A)	lactual (A)	lactual (A)	lactual (A)	lactual (A)	lactual (A)	lactual (A)
1	0.118	0.516	1.015	5.011	10.006	19.996	25.004	30.006	39.009
2	0.117	0.514	1.015	5.012	10.009	20.004	25.003	30.005	39.006
3	0.115	0.513	1.012	5.01	10.007	20.003	25.002	30.003	39.004

表 3-2. 充電モードでの電流ループ負荷レギュレーション誤差 (continued)

4.2	0.112	0.51	1.01	5.008	10.005	20.001	25	30.001	39.001
誤差 (mA)	0.018	0.016	0.015	0.012	0.009	0.004	0.004	0.006	0.009
誤差 (% FSR)	0.018	0.016	0.015	0.012	0.009	0.004	0.004	0.006	0.009

### 3.5.2 電圧ループ負荷レギュレーション誤差

表 3-3. 充電モードでの電圧ループ負荷レギュレーション誤差

フルスケール値 (V)	6.25			
TIDA_010087 VSET (V)	1	2	3	4.2
ELOAD CC モード				
ISET (A)	Vactual (V)	Vactual (V)	Vactual (V)	Vactual (V)
1	1.00012	2.00038	3.0002	4.2
10	1.00013	2.00039	3.00022	4.20002
20	1.00018	2.00035	3.00018	4.2
30	1.00012	2.00037	3.0002	4.19998
誤差 (mV)	0.00018	0.00039	0.00022	2E-05
誤差 (% FSR)	0.00288	0.00624	0.00352	0.00032

### 3.5.3 無負荷時の電圧遷移

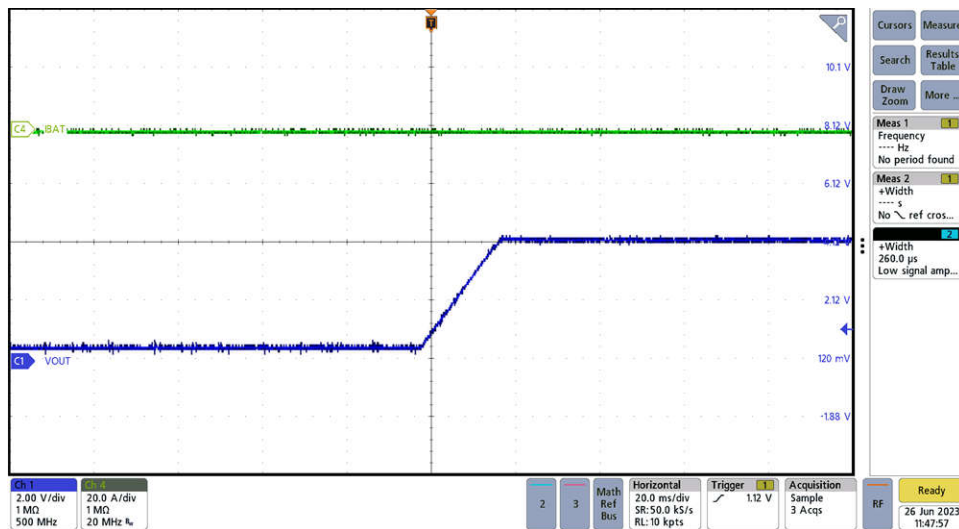


図 3-29. 無負荷時の電圧遷移

### 3.5.4 スタートアップ時の過渡応答

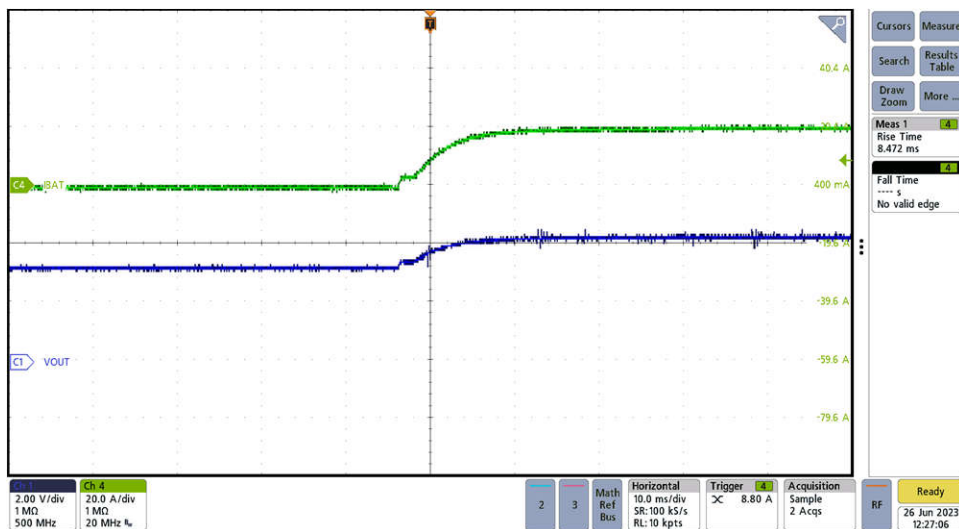


図 3-30. スタートアップ時の過渡応答

### 3.5.5 双方向電流スイッチング時間

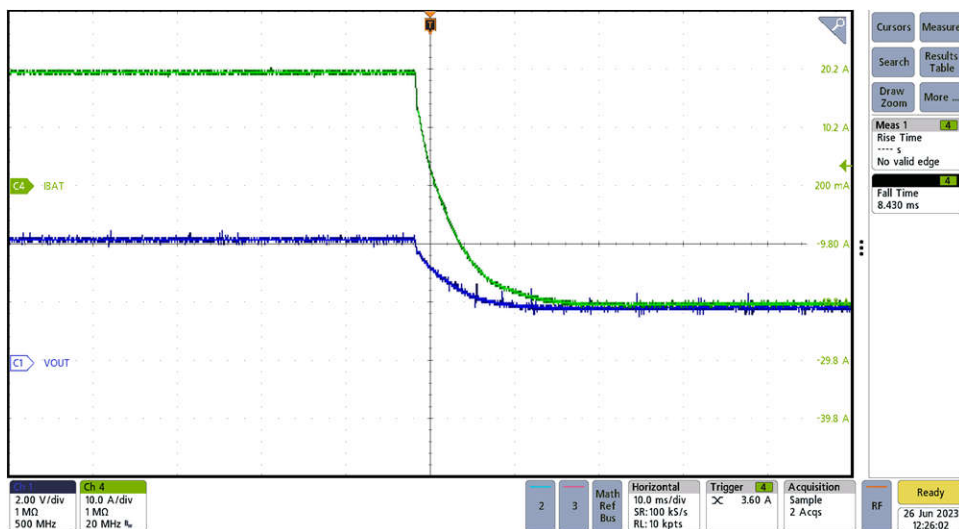


図 3-31. 電流遷移、充電モードから放電モードへ

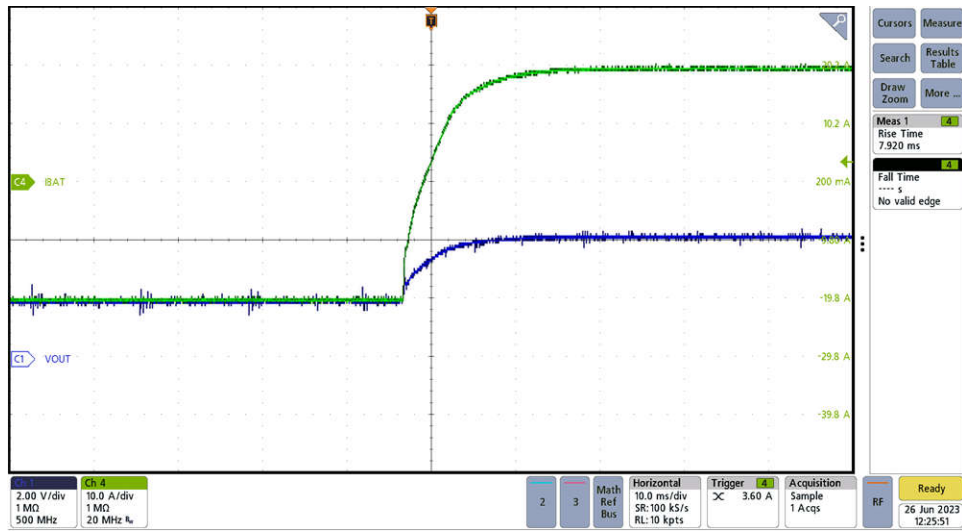


図 3-32. 電流遷移、放電モードから充電モードへ

## 4 設計とドキュメントのサポート

### 4.1 設計ファイル

#### 4.1.1 回路図

回路図をダウンロードするには、[TIDA-010087](#) の設計ファイルを参照してください。

#### 4.1.2 BOM

部品表 (BOM) をダウンロードするには、[TIDA-010087](#) の設計ファイルを参照してください。

### 4.2 ツールとソフトウェア

#### ツール

[TMDSCNCD280039C](#)

F280039C controlCARD 評価基板

#### ソフトウェア

[CCSTUDIO](#)

Code Composer Studio (CCS) 統合開発環境 (IDE)

[C2000WARE-DIGITALPOWER-SDK](#)

C2000™ マイコン向け DigitalPower ソフトウェア開発キット (SDK)

### 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、[『TMS320F28003x リアルタイム・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、[『ADS131M08 8 チャンネル、同時サンプリング、24 ビット、デルタ・シグマ ADC』データシート](#)

### 4.4 サポート・リソース

[TI 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、該当する貢献者により、現状のまま提供されるものです。これらは TI の仕様を構成するものではなく、必ずしも TI の見解を反映したものではありません。TI の[使用条件](#)を参照してください。

### 4.5 商標

TI E2E™ and C2000™ are trademarks of Texas Instruments.

すべての商標は、それぞれの所有者に帰属します。

## 5 著者

**SHAURY ANAND** は、テキサス・インスツルメンツのシステム・エンジニアであり、テストおよび測定アプリケーションのリファレンス・デザイン開発を担当しています。インド工科大学ローキー校で電気工学の学士号 (B.Tech) を取得しています。

著者は、このリファレンス・デザインでいただいたサポートに対して、VICTOR SALOMON と OZINO ODHARO に感謝しています。

## 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](https://www.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 © 2023, Texas Instruments Incorporated