*Design Guide: TIDA-010087*

# ***100-A, Dual-Phase Digital Control Battery Tester Reference Design***

## Description

The reference design illustrates a method to precisely control the current and voltage of a bidirectional interleaved buck converter power stage using a C2000$^{TM}$ microcontroller (MCU) and a precision ADC ADS131M08. The design achieves less than ±20-mA current regulation error, and ±1-mV voltage regulation error by utilizing a high-resolution pulse-width modulation (PWM) generation peripheral of the C2000 MCU.

## Resources

| | |
|---|---|
| TIDA-010087 | Design Folder |
| TMS320F280039 | Product Folder |
| ADS131M08, REF35, INA188 | Product Folder |
| OPA2182, LM321LV, TLV9102, UCC27211 | Product Folder |
| CSD17556Q5B, CSD16570Q5B, LMR54410 | Product Folder |
| TPSI3050, TVS0500, TPS7A20 | Product Folder |
| TLV1117, LM2664, TPS736 | Product Folder |
| C2000WARE-DIGITALPOWER-SDK | Tool Folder |

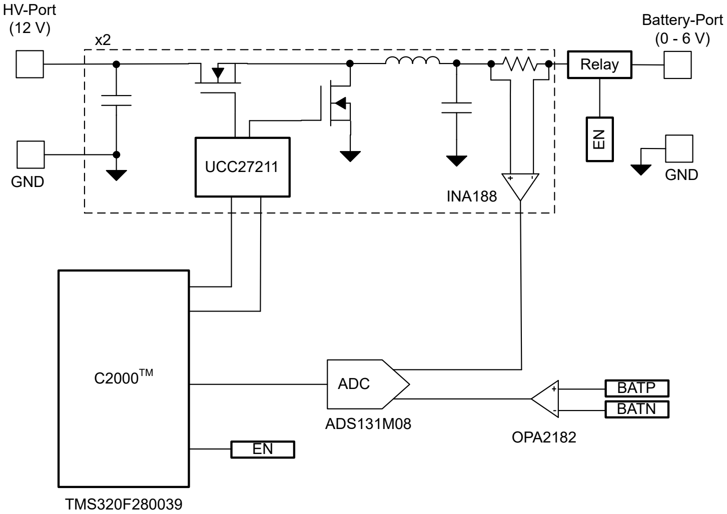
Ask our TI E2E™ support experts

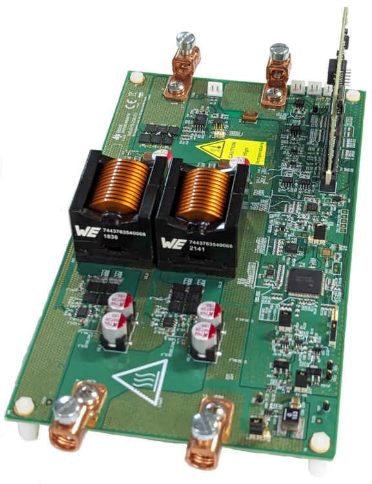
## Features

- Dual-phase interleaved, 600-W, bidirectional buck power stage
- 15.8-bits PWM resolution at 100-kHz switching frequency
- External delta-sigma ADC for closed-loop control
- Constant current charging and discharging with regulation error < ±20 mA
- Constant voltage mode supported in both charging and discharging with regulation error < ±1 mV
- Software Frequency Response Analyzer (SFRA) and compensation designer for ease of tuning of control loops
- powerSUITE support for easy adaptation of design for user requirements

## Applications

- Battery cell formation and test equipment
- Programmable DC power supply





# 1 System Description

The battery tester equipment includes a wide variety of equipment used to test single cells, battery modules, and high-voltage battery packs. The test equipment contains precision power supplies and data acquisition systems, and is used for charging and discharging of batteries, and measures various parameters of the cells.

Figure 1-1 shows a simplified Li-Ion battery manufacturing process. The Final stage, End-of-Line Conditioning, includes cell formation and testing. Formation is a critical step in the manufacturing of Li-ion cells. During formation, the cells go through a process of initial charge and discharge, which results in the formation of the solid electrolyte interface (SEI) layer. The quality of the SEI layer impacts capacity and reliability of the battery cell. To control the formation process, precise programmable power supplies are used for charging and discharging of cells. These power supplies are called battery formation systems or battery testers. The accuracy required in battery testers for voltage and current is typically between ±0.02% and ±0.05% of full-scale.

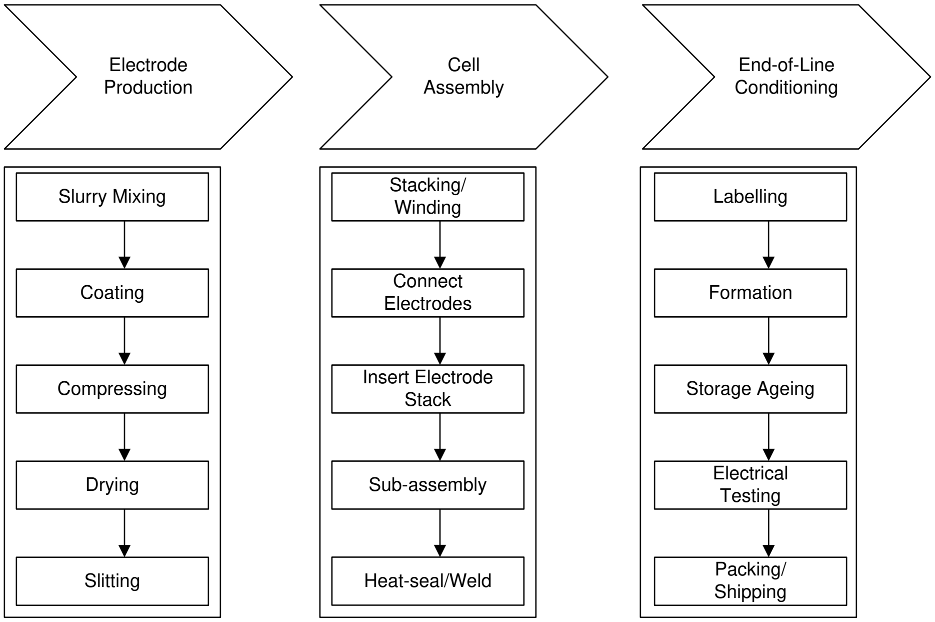


**Figure 1-1. Simplified Li-Ion Battery Manufacturing Process**

## 1.1 Key System Specifications

| PARAMETER | SPECIFICATION |
|---|---|
| LV port – Battery port | 50 mV to 6 V |
| HV port–Bus voltage | 12 V to 15 V |
| Maximum Output Current | ± 100 A |
| Maximum DC current per phase | ±50 A |
| Number of Phases | 2 |
| Switching Frequency | 93.75 kHz |
| Current Regulation Error | < ±20 mA (0.02% FS) |
| Voltage Regulation Error | < ±1 mV (0.02% FS) |

# 2 System Overview

## 2.1 Block Diagram

Figure 2-1 is a block diagram of the reference design. The TMS320F280039 MCU generates a high-resolution 16-bit PWM for a synchronous buck power stage, and performs current and voltage control functions. The INA188 instrumentation amplifier senses the current and the OPA2182 operational amplifier senses the voltage. Current and voltage signals are converted to digital data by the external ADS131M08 ADC. C2000 on-chip window comparators are used to implement overcurrent current protection.

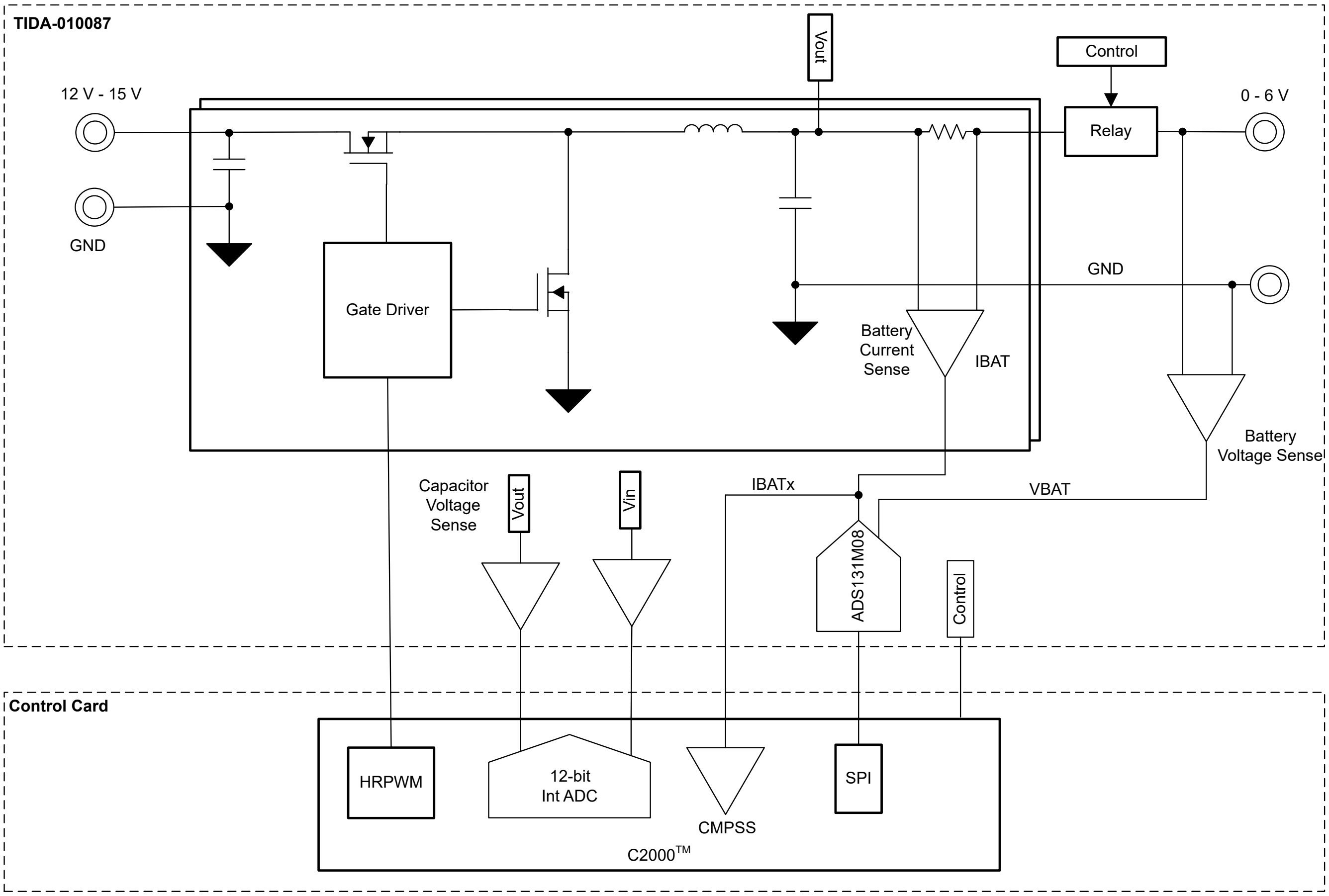


**Figure 2-1. TIDA-010087 Block Diagram**

## 2.2 Design Considerations

### 2.2.1 Current and Voltage Controller

Figure 2-2 shows the software implementation of current and voltage control loops. Voltage loop is cascaded to the current to achieve both constant-current and constant-voltage in charging and discharging modes. When the battery voltage is far away from the constant-voltage setting (VSET), the voltage loop gets saturated to constant current setting (ISET). When battery voltage reaches close to VSET, the voltage loop is closed, and ISET is reduced to make sure the battery voltage does not exceed the VSET limit. The controller works in both charge and discharge modes. In the charge mode, VSET limits the maximum battery voltage, thus stops the charging. While in the discharge mode, VSET limits the minimum battery voltage which stops the discharging.

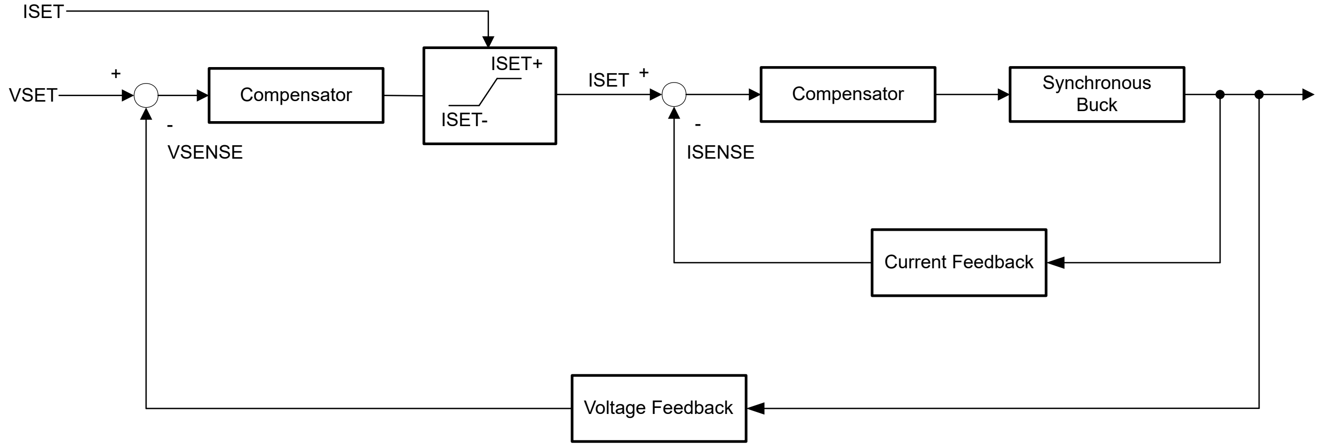


**Figure 2-2. Current and Voltage Controller**

### 2.2.2 High-Resolution PWM Generation

To generate high resolution, a C2000 with high-resolution PWM output capability is used. The high-resolution counter is capable of 150-ps time step, which is equivalent to 16-bit resolution at 100-kHz PWM frequency for a 100-MHz CPU clock. Table 2-1 shows PWM resolution at different switching frequencies.

**Table 2-1. C2000™ MCU Resolution for PWM and HRPWM**

| PWM FREQUENCY | REGULAR RESOLUTION (PWM) | | HIGH RESOLUTION PWM | |
|---|---|---|---|---|
| | 100-MHz EPWMCLK | | | |
| (kHz) | BITS | % | BITS | % |
| 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 Highlighted Products

### 2.3.1 TMS320F280039

The TMS320F280039 C2000 device is used control the synchronous buck power stage. The device has eight HRPWM channels and four windowed comparators that are sufficient to control four battery test channels or buck converters. For more information, see the *TMS320F28003x Real-Time Microcontrollers* data sheet.

### 2.3.2 ADS131M08

The ADS131M08 is an eight-channel, simultaneously sampling, 24-bit, delta-sigma (ΔΣ), analog-to-digital converter (ADC) that allows maximum sample rates up to 32 ksps and is sufficient for ± 0.01% accuracy and 1-kHz loop bandwidth.

# 3 Hardware, Software, Testing Requirements, and Test Results

## 3.1 Hardware Requirements

Figure 3-1 shows the TIDA-010087 hardware. The TIDA-010087 board requires a F280039C controlCARD Evaluation Module to operate.

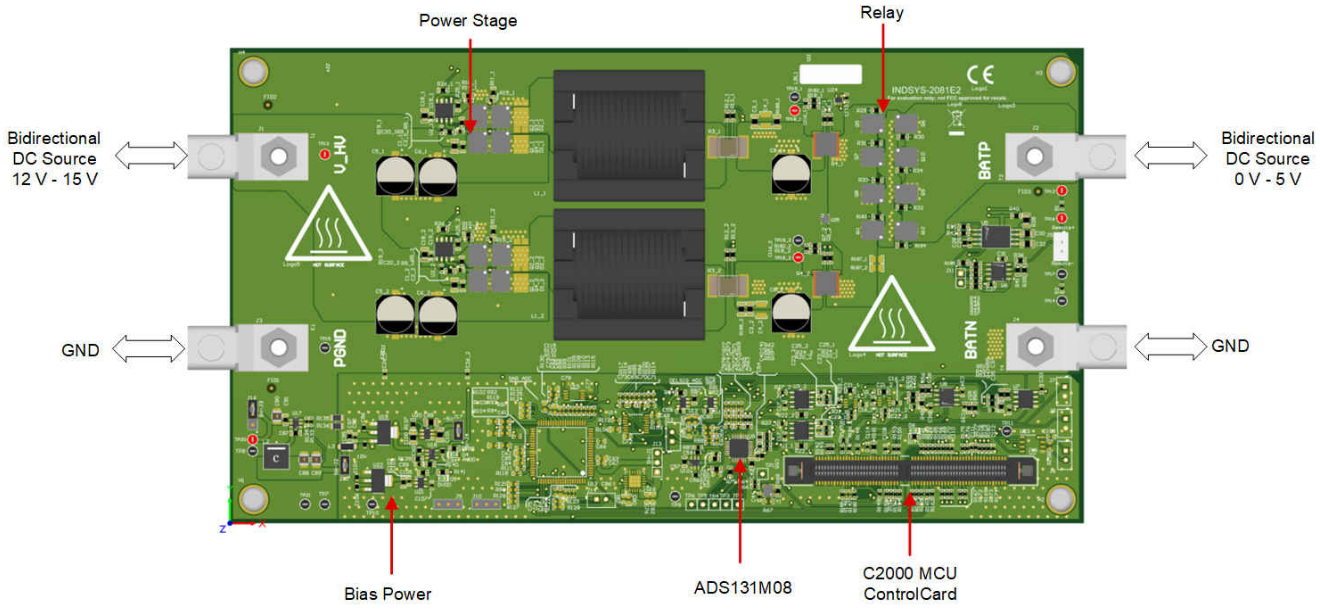

Power Stage
Relay
Bidirectional DC Source 12 V - 15 V
Bidirectional DC Source 0 V - 5 V
GND
GND
Bias Power
ADS131M08
C2000 MCU ControlCard

**Figure 3-1. Board Overview**

## 3.2 Software Requirements

The design software is available in the DigitalPower Software Development Kit (SDK) for C2000 MCUs (C2000WARE-DIGITALPOWER-SDK), and is supported inside the powerSUITE framework.

### 3.2.1 Opening the Project Inside Code Composer Studio

Use the following steps to start a project in CCS:

1. Install Code Composer Studio from the Code Composer Studio (CCS) integrated development environment (IDE) tools folder. Version 12.3 or above is recommended.
2. Install C2000WARE-DIGITALPOWER-SDK in one of two ways:
   a. Go to CCS and click on *View* → *Resource Explorer*. Under the TI Resource Explorer, go to C2000WARE-DIGITAL-POWER-SDK, and click on the install button.
   b. Through the C2000Ware Digital Power SDK tools folder.
3. Once installation completes, close CCS, and open a new workspace. CCS automatically detects powerSUITE. Sometimes CCS must be restarted for the change to take effect.

**Note**

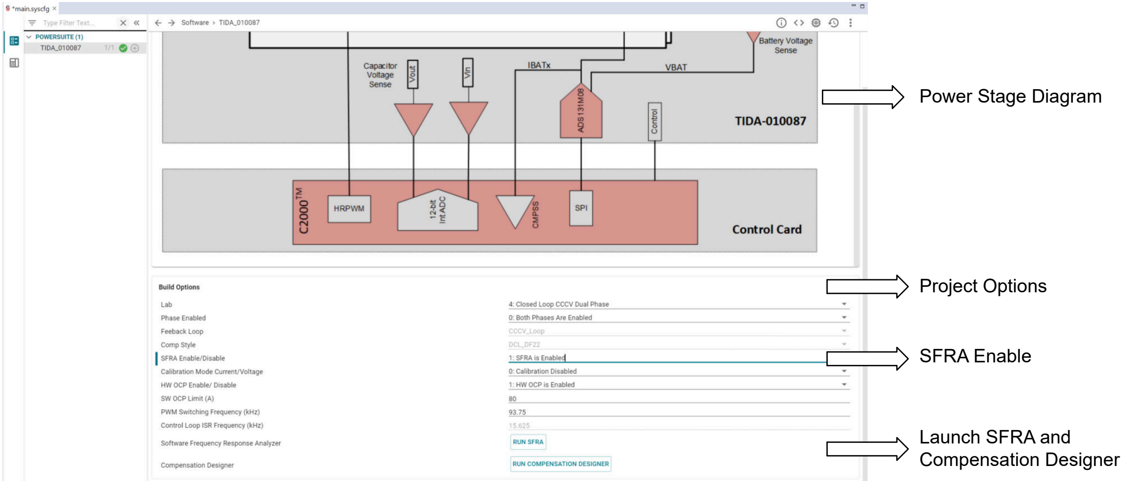
By default, powerSUITE is installed with the installation of SDK.

The firmware project can now be imported using one of the following methods:

- **Using *Resource Explorer***
  1. In the *Resource Explorer*, under C2000WARE-DIGITAL-POWER-SDK, click on *powerSUITE* → *Solution Adapter Tool*.
  2. Select TIDA-010087 from the list of designs presented under DC-DC section.
  3. The development kit page is displayed. The icon to run the project appears in the top bar. Click *Run Project*.
  4. This action imports the project into the workspace environment, and a configuration page with a GUI similar to Figure 3-2 appears.

5. If this GUI page does not appear, refer to the FAQ section under powerSUITE in the C2000WAREDIGITAL-POWER-SDK resource explorer.

- **Direct import from the solution folder**
  1. The user can also directly import the project by going inside CCS to click Project → *Import CCS Projects* and browsing to the solution folder located at /solutions/tida_010087/f28003x/ccs.
  2. Two project specifications appear: one of the project with powerSUITE, and the other without powerSUITE. Clicking on either creates a self-contained folder of the project with all the dependencies inside.
  3. The non-powerSUITE project is provided for customers who find the powerSUITE GUI limiting or want to remove powerSUITE for production code.
  4. This document guides the user through the powerSUITE project, but all the steps can be repeated with the non-powerSUITE project with modification to the relevant `#defines` in the powerSUITE settings.h file, which are documented in this design guide.



Build Options

Lab: 4: Closed Loop CCCV Dual Phase
Phase Enabled: 0: Both Phases Are Enabled
Feeback Loop: CCCV_Loop
Comp Style: DCL_DF22
SFRA Enable/Disable: 1: SFRA is Enabled
Calibration Mode Current/Voltage: 0: Calibration Disabled
HW OCP Enable/ Disable: 1: HW OCP is Enabled
SW OCP Limit (A): 80
PWM Switching Frequency (kHz): 93.75
Control Loop ISR Frequency (kHz): 15.625
Software Frequency Response Analyzer
Compensation Designer

Power Stage Diagram

Project Options

SFRA Enable

Launch SFRA and Compensation Designer

**Figure 3-2. powerSUITE Page for the Design**

### 3.2.2 Project Structure

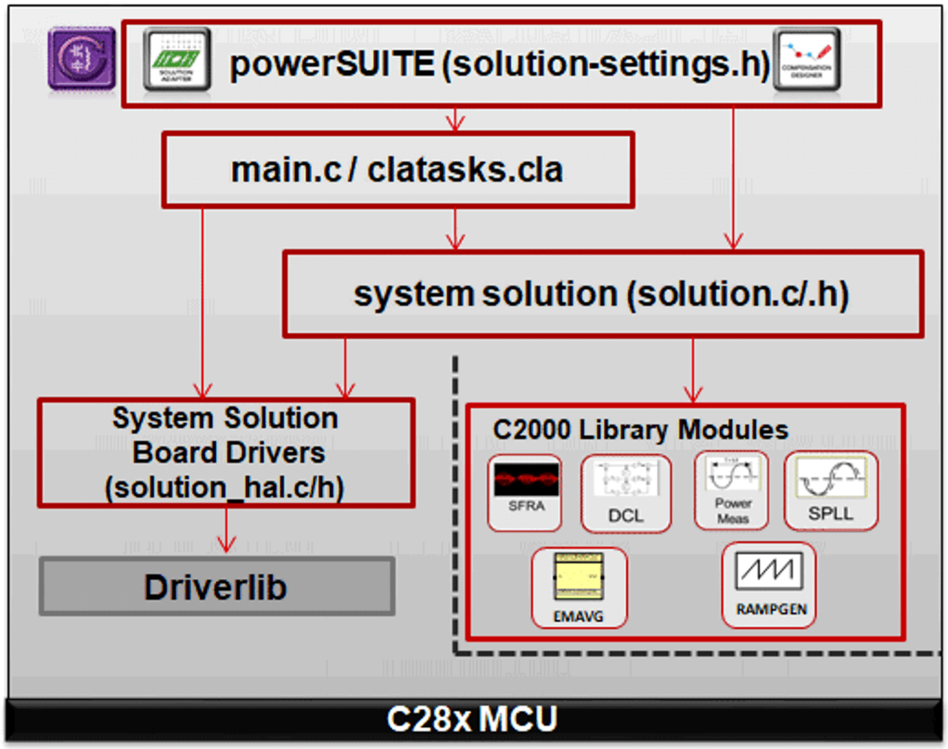

powerSUITE (solution-settings.h)
main.c / clatasks.cla
system solution (solution.c/.h)
System Solution Board Drivers (solution_hal.c/h)
Driverlib
C2000 Library Modules
SFRA
DCL
Power Meas
SPLL
EMAVG
RAMPGEN
C28x MCU

**Figure 3-3. Project Structure Overview**

Figure 3-3 shows the general structure of the project. Once the project is imported, the Project Explorer appears inside CCS as shown in Figure 3-4.

**Note**

Figure 3-4 shows the project for F28003x; however, if a different device is chosen from the powerSUITE page, the structure is similar.

Solution-specific and device-independent files that consist of the core algorithmic code are in *.c/h*.

Board-specific and device-specific files are in *_hal.c/h*. This file consists of device-specific drivers to run the solution. If the user wants to use a different modulation scheme or a different device, the user is required only to make changes to these files, besides changing the device support files in the project.

The *-main.c* file consists of the main framework of the project. This file consists of calls to the board and solution file that help in creating the system framework, along with the interrupt service routines (ISRs) and slow background tasks.

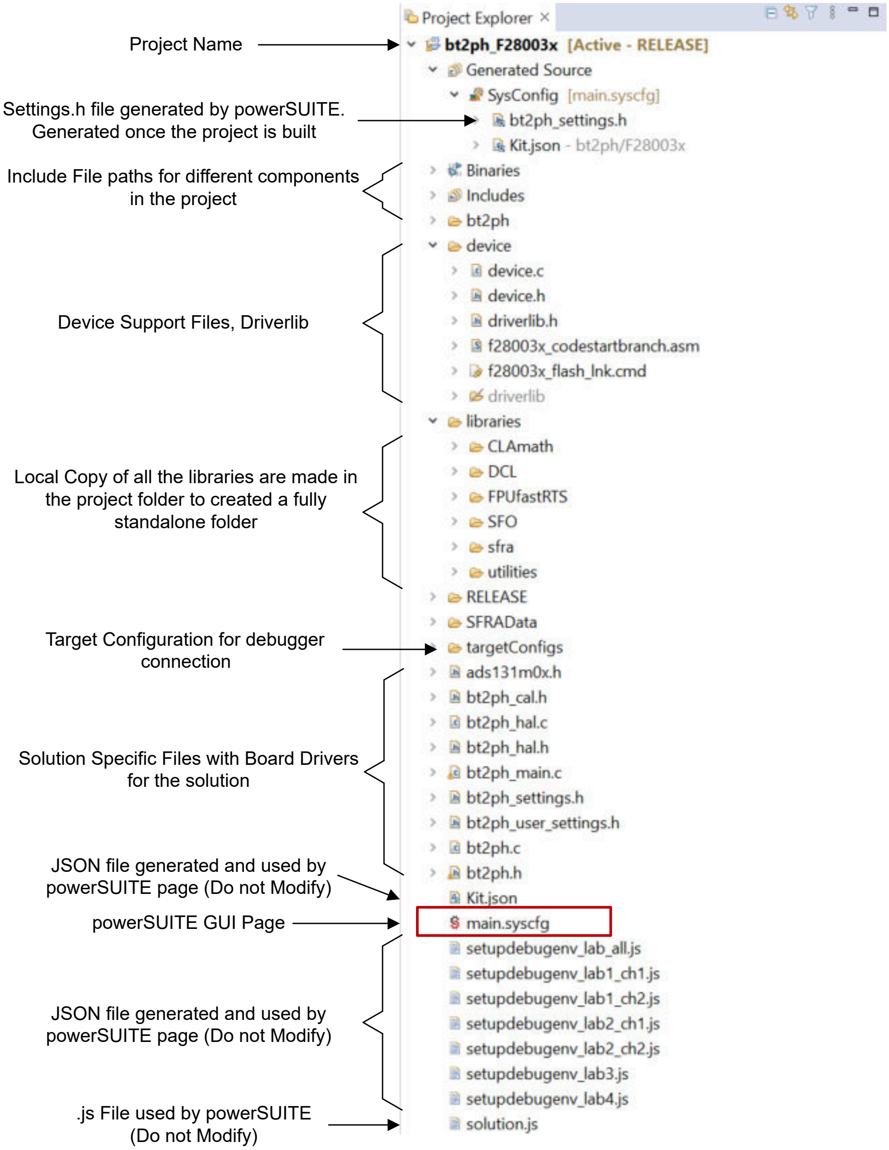
For this design, the solution is *bt2ph*.

The powerSUITE page can be opened by clicking on the *main.syscfg* file, listed under the Project Explorer. The powerSUITE page generates the *_settings.h* file. This file is the only C language based file used in the compile of the project that is generated by the powerSUITE page. The user must not modify this file manually, because the changes are overwritten by powerSUITE each time the project is saved. *_user_settings.h* is included by the *_settings.h* and can be used to keep any settings that are outside the scope of powerSUITE tools such as `#defines` for ADC mapping, GPIOs, and so forth.

The *_cal.h* file consists of gain and offset values for current and voltage measurements.

The *Kit.json* and *solution.js* files are used internally by powerSUITE, and must not be modified by the user. Any changes to these files results in the project not functioning properly.

The solution name is also used as the module name for all the variables and defines used in the solution. Hence, all variables and function calls are prepended by the BT2PH name (for example, BT2PH_userParam_V_I_ch1). This naming convention lets the user combine different solutions while avoiding naming conflicts.



Project Name

Settings.h file generated by powerSUITE. Generated once the project is built

Include File paths for different components in the project

Device Support Files, Driverlib

Local Copy of all the libraries are made in the project folder to created a fully standalone folder

Target Configuration for debugger connection

Solution Specific Files with Board Drivers for the solution

JSON file generated and used by powerSUITE page (Do not Modify)

powerSUITE GUI Page

JSON file generated and used by powerSUITE page (Do not Modify)

.js File used by powerSUITE (Do not Modify)

```
Project Explorer ×
v bt2ph_F28003x [Active - RELEASE]
  v Generated Source
    v SysConfig [main.syscfg]
        bt2ph_settings.h
      > Kit.json - bt2ph/F28003x
  > Binaries
  > Includes
  > bt2ph
  v device
    > device.c
    > device.h
    > driverlib.h
    > f28003x_codestartbranch.asm
    > f28003x_flash_lnk.cmd
    > driverlib
  v libraries
    > CLAmath
    > DCL
    > FPUfastRTS
    > SFO
    > sfra
    > utilities
  > RELEASE
  > SFRAData
  > targetConfigs
  > ads131m0x.h
  > bt2ph_cal.h
  > bt2ph_hal.c
  > bt2ph_hal.h
  > bt2ph_main.c
  > bt2ph_settings.h
  > bt2ph_user_settings.h
  > bt2ph.c
  > bt2ph.h
    Kit.json
    main.syscfg
    setupdebugenv_lab_all.js
    setupdebugenv_lab1_ch1.js
    setupdebugenv_lab1_ch2.js
    setupdebugenv_lab2_ch1.js
    setupdebugenv_lab2_ch2.js
    setupdebugenv_lab3.js
    setupdebugenv_lab4.js
    solution.js
```

**Figure 3-4. Project Explorer View of the BT2PH Project**

The BT2PH project consists of three ISRs (ISR3, ISR4, and ISR7). ISR1, ISR2, ISR5, and ISR6 are reserved for future use.

**ISR3** is used to sense the input voltage and capacitor voltage of the buck converters. ISR3 is triggered by ADCC conversion complete. ADCC is used to sense input voltage and output voltage of the converter to implement soft-start of the DC/DC.

**ISR4** is triggered by the DRDY (Data Ready) Signal of the ADS131M08. The external ADC is programmed for a 15.625-kHz sample rate, which sets the ISR frequency. The ISR runs the current and voltage control loop functions.

**ISR7** is triggered by SPI receive FIFO interrupt. The ISR is used to read the external ADC data from FIFO registers.

Figure 3-5 and Figure 3-6 show the time taken by ISR3, ISR4, and ISR7. The total time taken three ISRs is less than 8 µs. The ISRs take 12.5% of CPU resources for 15.625-kHz ISR frequency.

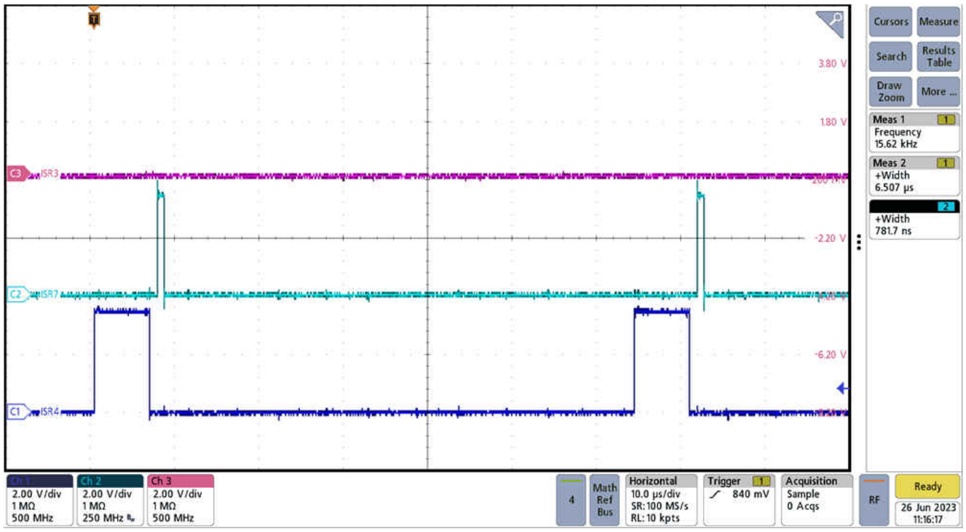


**Figure 3-5. ISR4 and ISR7 Execution Time Measurement**

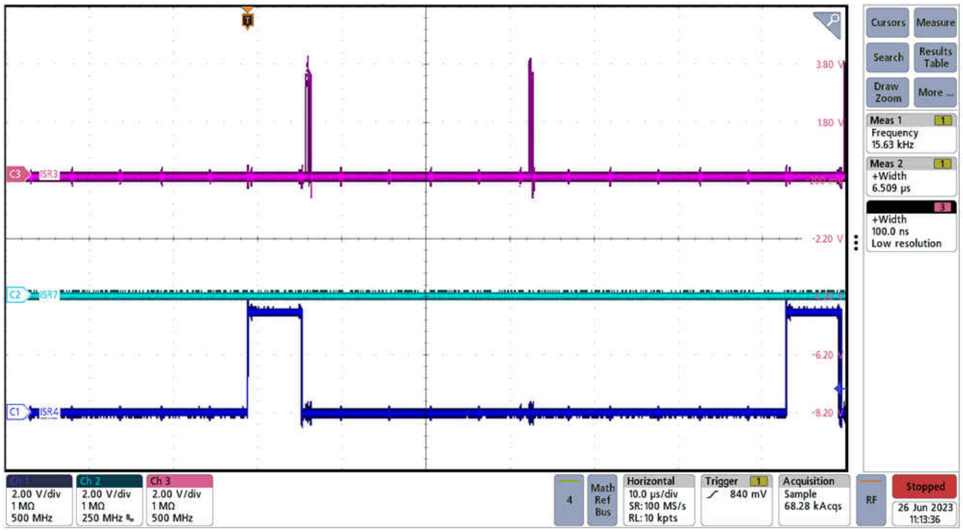


**Figure 3-6. ISR4 and ISR3 Execution Time Measurement**

### 3.2.3 Software Flow Diagram

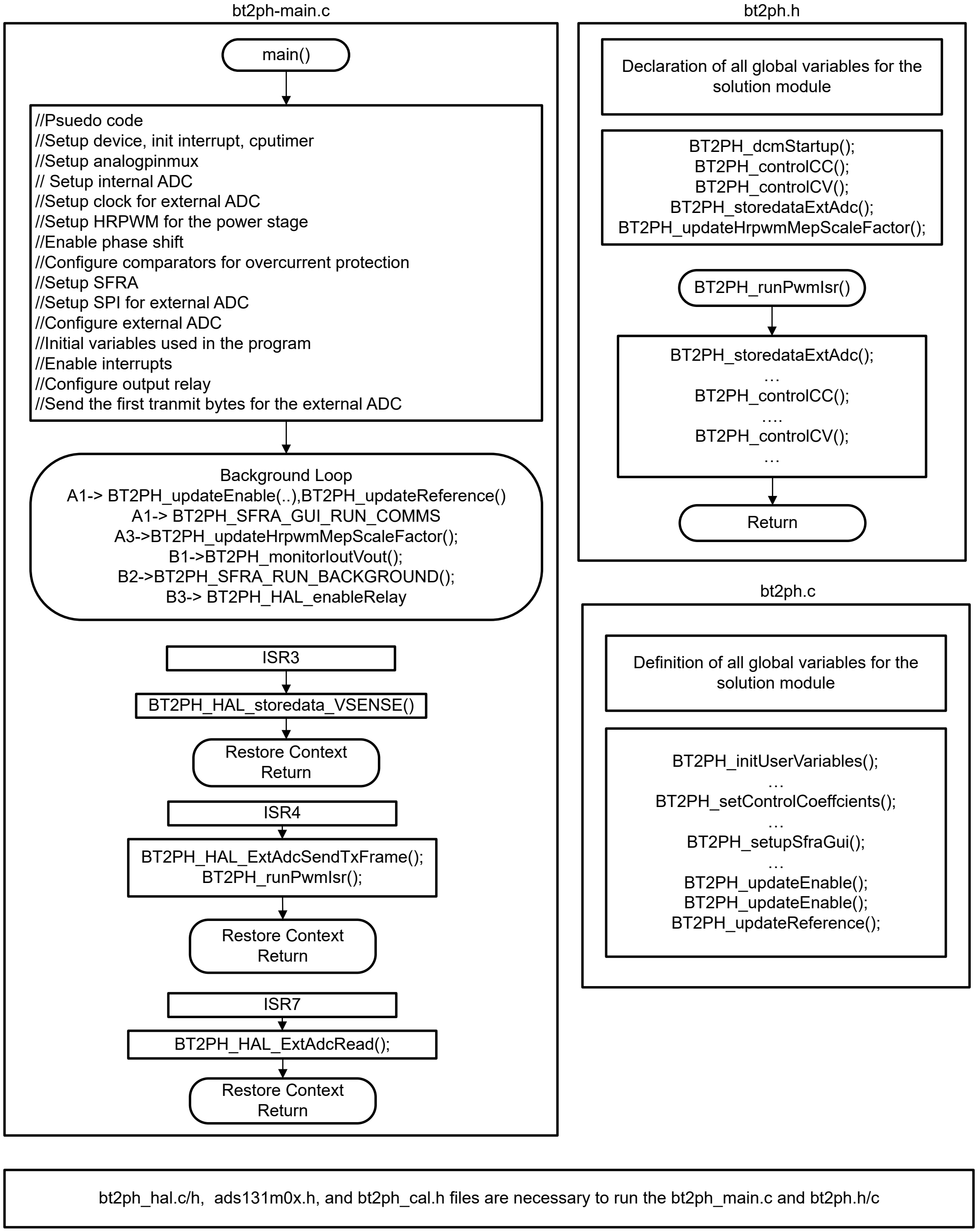

bt2ph-main.c
main()
//Psuedo code
//Setup device, init interrupt, cputimer
//Setup analogpinmux
// Setup internal ADC
//Setup clock for external ADC
//Setup HRPWM for the power stage
//Enable phase shift
//Configure comparators for overcurrent protection
//Setup SFRA
//Setup SPI for external ADC
//Configure external ADC
//Initial variables used in the program
//Enable interrupts
//Configure output relay
//Send the first tranmit bytes for the external ADC
Background Loop
A1-> BT2PH_updateEnable(..),BT2PH_updateReference()
A1-> BT2PH_SFRA_GUI_RUN_COMMS
A3->BT2PH_updateHrpwmMepScaleFactor();
B1->BT2PH_monitorIoutVout();
B2->BT2PH_SFRA_RUN_BACKGROUND();
B3-> BT2PH_HAL_enableRelay
ISR3
BT2PH_HAL_storedata_VSENSE()
Restore Context
Return
ISR4
BT2PH_HAL_ExtAdcSendTxFrame();
BT2PH_runPwmIsr();
Restore Context
Return
ISR7
BT2PH_HAL_ExtAdcRead();
Restore Context
Return
bt2ph.h
Declaration of all global variables for the solution module
BT2PH_dcmStartup();
BT2PH_controlCC();
BT2PH_controlCV();
BT2PH_storedataExtAdc();
BT2PH_updateHrpwmMepScaleFactor();
BT2PH_runPwmIsr()
BT2PH_storedataExtAdc();
…
BT2PH_controlCC();
….
BT2PH_controlCV();
…
Return
bt2ph.c
Definition of all global variables for the solution module
BT2PH_initUserVariables();
…
BT2PH_setControlCoeffcients();
…
BT2PH_setupSfraGui();
…
BT2PH_updateEnable();
BT2PH_updateEnable();
BT2PH_updateReference();
bt2ph_hal.c/h, ads131m0x.h, and bt2ph_cal.h files are necessary to run the bt2ph_main.c and bt2ph.h/c

**Figure 3-7. Software Flow Diagram**

## 3.3 Test Setup

### 3.3.1 Hardware Setup to Tune the Current and Voltage Loops

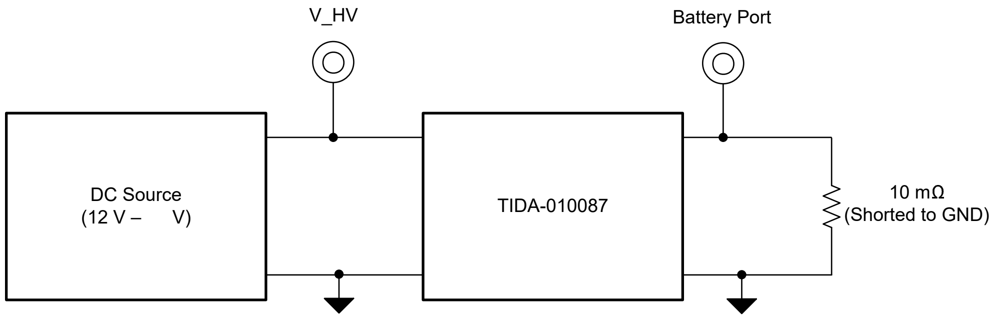


**Figure 3-8. Hardware Setup to Tune the Current and Voltage Loops**

### 3.3.2 Hardware Setup to Test Bidirectional Power Flow

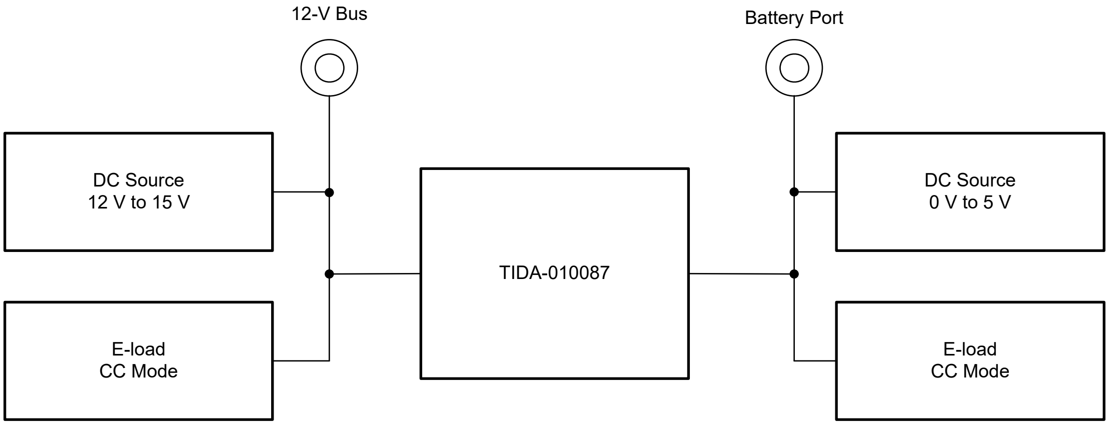


**Figure 3-9. Hardware Setup to Test Bidirectional Power Flow**

### 3.3.3 Hardware Setup for Current and Voltage Calibration

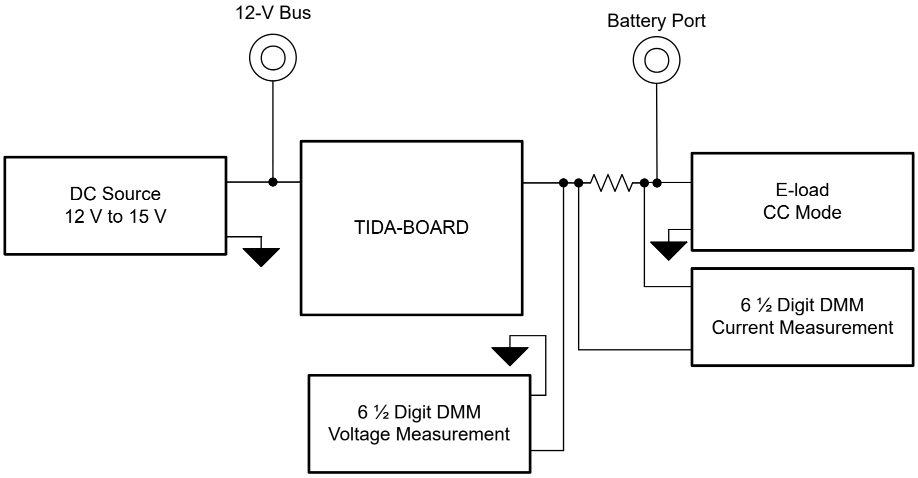


**Figure 3-10. Hardware Setup for Current and Voltage Calibration**

## 3.4 Test Procedure

### 3.4.1 Lab Variables Definitions

BT2PH_userParam_V_I_chx parameters are used to control the power stage in different Labs. BT2PH_userParam_V_I_ch1 and BT2PH_userParam_V_I_ch2 are used in Lab 1 and Lab 2 to control Phase 1 and Phase 2 of the DC/DC converter. Lab 3 and Lab 4 use BT2PH_userParam_V_I_chm variable. See Table 3-1 for the parameter definitions.

**Table 3-1. BT2PH_userParam Definition**

| BT2PH_userParam | DATA TYPE | COMMENTS |
|---|---|---|
| iref_A | float | Set current for both charging and discharging mode [0, 100] |
| vrefCharge_V | float | Set voltage in charge mode [0, 5] |
| vrefDischarge_V | float | Set voltage in discharge mode [0, 5] |
| dir_bool | unsigned int | Set this parameter to 1 for charging mode |
| | | Set this parameter to 0 for discharging mode |
| en_bool | unsigned int | Set this parameter to 1 to enable the channel |
| dutyRef_pu | float | Reference duty cycle for open loop mode. Range = 0 to 1.0 |
| ibatCal_pu | float | Use this parameter to set output current in calibration mode.<br>Range = 0 to 1.0 |
| vbatCal_pu | float | Use this parameter to set output voltage in calibration mode.<br>Range = 0 to 1.0 |
| IoutGain_pu | float | The variable stores current gain calibration data |
| ioutOffset_pu | float | The variable stores current offset calibration data |
| IoutGain_A | float | The variable stores current gain calibration data |
| IoutOffset_A | float | The variable stores current offset calibration data |
| vbatGain_pu | float | The variable stores voltage gain calibration data |
| vbatOffset_pu | float | The variable stores voltage offset calibration data |
| vbatGain_V | float | The variable stores voltage gain calibration data |
| vbatOffset_V | float | The variable stores voltage offset calibration data |

### 3.4.2 Lab 1. Open-Loop Current Control Single Phase

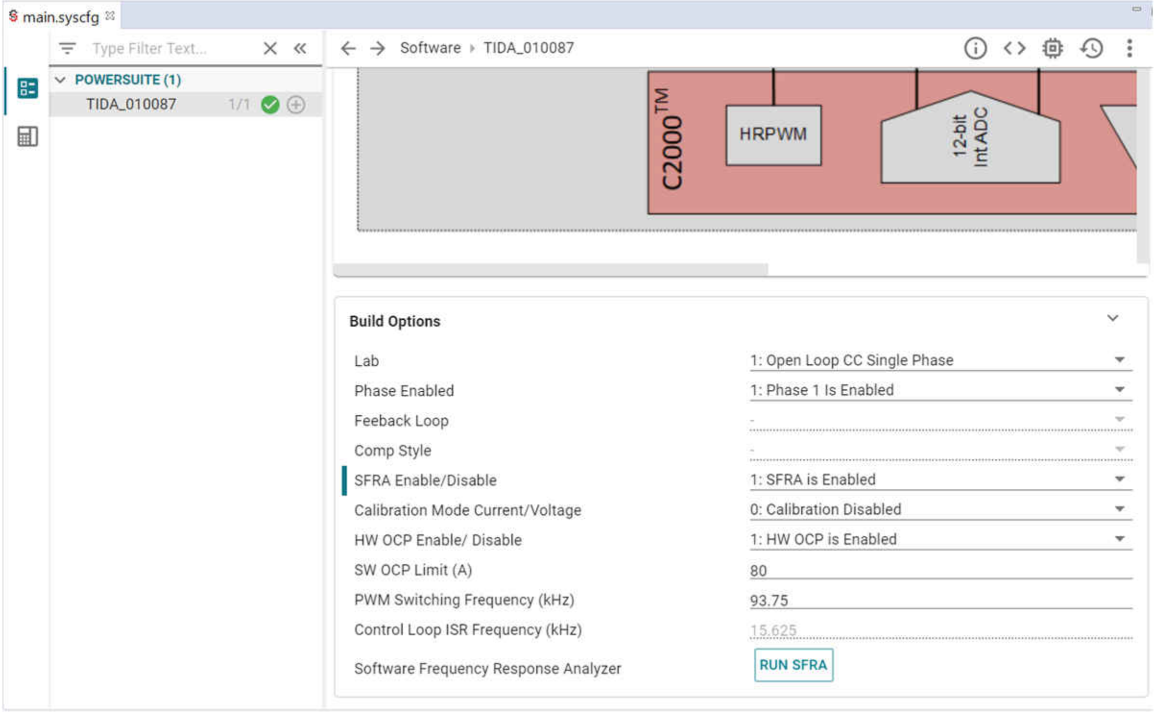
#### *3.4.2.1 Setting Software Options for Lab 1*

1. Open the CCS project as outlined in Section 3.2.1. If using the powerSUITE, go to Step 2, otherwise jump to Step 3.
2. Open the SYSCONFIG page and select under the *Build Options* section:
   - Select *Lab 1: Open Loop CC Single Phase* for the Lab.
   - Change the *Phase Enabled* to Phase 1 or Phase 2.
   - Set the SFRA Enable/Disable to 1.
   - Save the page.
3. When using the non-powerSuite version of the project, the above settings are directly modified in the *solution_settings.h* file.

```
#define LAB_NUMBER (1)

#define PHASE_NUMBER (1)

#define SFRA_ENABLED (true)
```



**Build Options**

| | |
|---|---|
| Lab | 1: Open Loop CC Single Phase |
| Phase Enabled | 1: Phase 1 Is Enabled |
| Feeback Loop | - |
| Comp Style | - |
| SFRA Enable/Disable | 1: SFRA is Enabled |
| Calibration Mode Current/Voltage | 0: Calibration Disabled |
| HW OCP Enable/ Disable | 1: HW OCP is Enabled |
| SW OCP Limit (A) | 80 |
| PWM Switching Frequency (kHz) | 93.75 |
| Control Loop ISR Frequency (kHz) | 15.625 |
| Software Frequency Response Analyzer | RUN SFRA |

**Figure 3-11. Build Options for Lab 1**

### 3.4.2.2 Building and Loading the Project and Setting up Debug Environment

Use the following steps to build and load the project and to set up the debug environment.

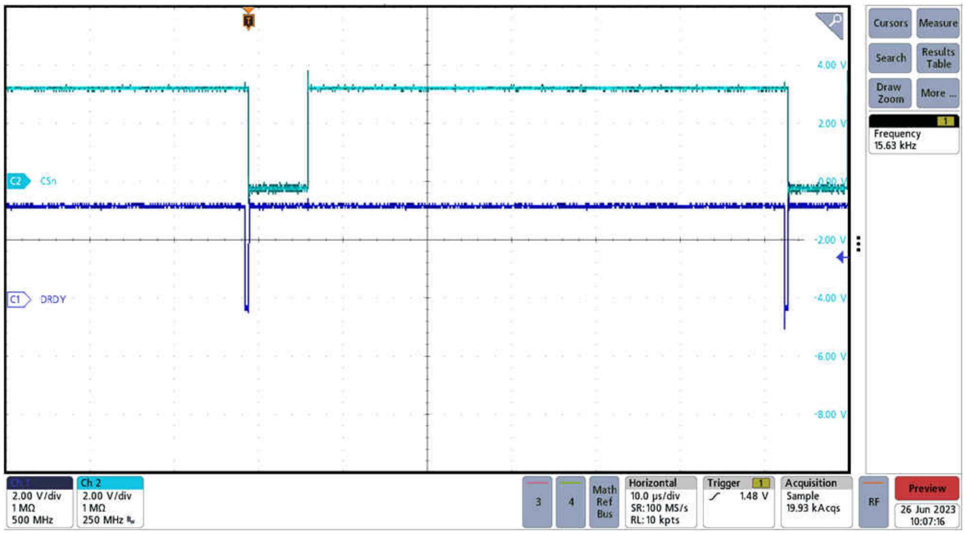
1. Right-click on the project name and click *Rebuild Project*.
2. The project builds successfully.
3. In the Project Explorer, make sure the correct target configuration file is set as Active under targetConfigs.
4. Then, click *Run* → *Debug* to launch a debugging session.
5. The project then loads on the device and the CCS debug view becomes active. The code halts at the start of the main routine.
6. To add the variables in the watch/expressions window, click *View* → *Scripting Console* to open the scripting console dialog box. On the upper right corner of this console, click on *open* and then browse to the *setupdebugenv_lab1_ch1.js* script file located inside the project folder. This populates the watch window with the appropriate variables needed to debug the system. For phase 2, select *setupdebugenv_lab2_ch2.js* script file.
7. Click on the *Continuous Refresh* button on the watch window to enable continuous update of values from the controller.

### 3.4.2.3 Running the Code

Use the following steps to run the code for Lab 1.

1. Use the test setup shown in Section 3.3.1.
2. Run the project by clicking from the menu bar.
3. In the watch view, check if the BT2PH_InputVoltageSense_V is between 12V -15V in the *Expression Window*.
4. Check *DRDY* signal of the external ADC if the frequency is 15.625 kHz using an oscilloscope. Figure 3-12 shows the DRDY and CS signal of ADS131M08 when the MCU is running.
5. Set the following parameters from the *Expression Window*:
   - BT2PH_userParam_V_I_ch1->dutyRef_pu = 0.03
   - Set the BT2PH_userParam_V_I_ch1->en_bool = 1
   - Set the "BT2PH_enableRelay_bool" to 1 to enable the output relay
   - See Figure 3-13 for the Expression Window settings

6. BT2PH_measureMultiphase_V_I variable shows output current and voltage of the DC/DC converter. Adjust the BT2PH_userParam_V_I_ch1->dutyRef_pu to make sure the current is approximately 15 A.
7. SFRA Setup for Open-Loop Current Control shows the SFRA setup to extract the plant model for Open-Loop Current Control. Click on the *Run SFRA* icon from the SYSCONFIG page. The SFRA GUI pops up.
8. Select the options for the device on the SFRA GUI; for example, for F280039, select floating point. Click on *Setup Connection*. In the pop-up window, uncheck the boot-on-connect option and select an appropriate COM port. Click the Ok button. Return to the SFRA GUI and click the *Connect* button.
9. The SFRA GUI connects to the device. An SFRA sweep can now be started by clicking *Start Sweep*. The complete SFRA sweep takes a few minutes to finish. Once complete, a graph with the measurement appears, as shown in Figure 3-15.
10. The Frequency Response Data is saved in the project folder, under an SFRA Data Folder, and is time-stamped with the time of the SFRA run.



Frequency 1
15.63 kHz

Ch 1
2.00 V/div
1 MΩ
500 MHz

Ch 2
2.00 V/div
1 MΩ
250 MHz

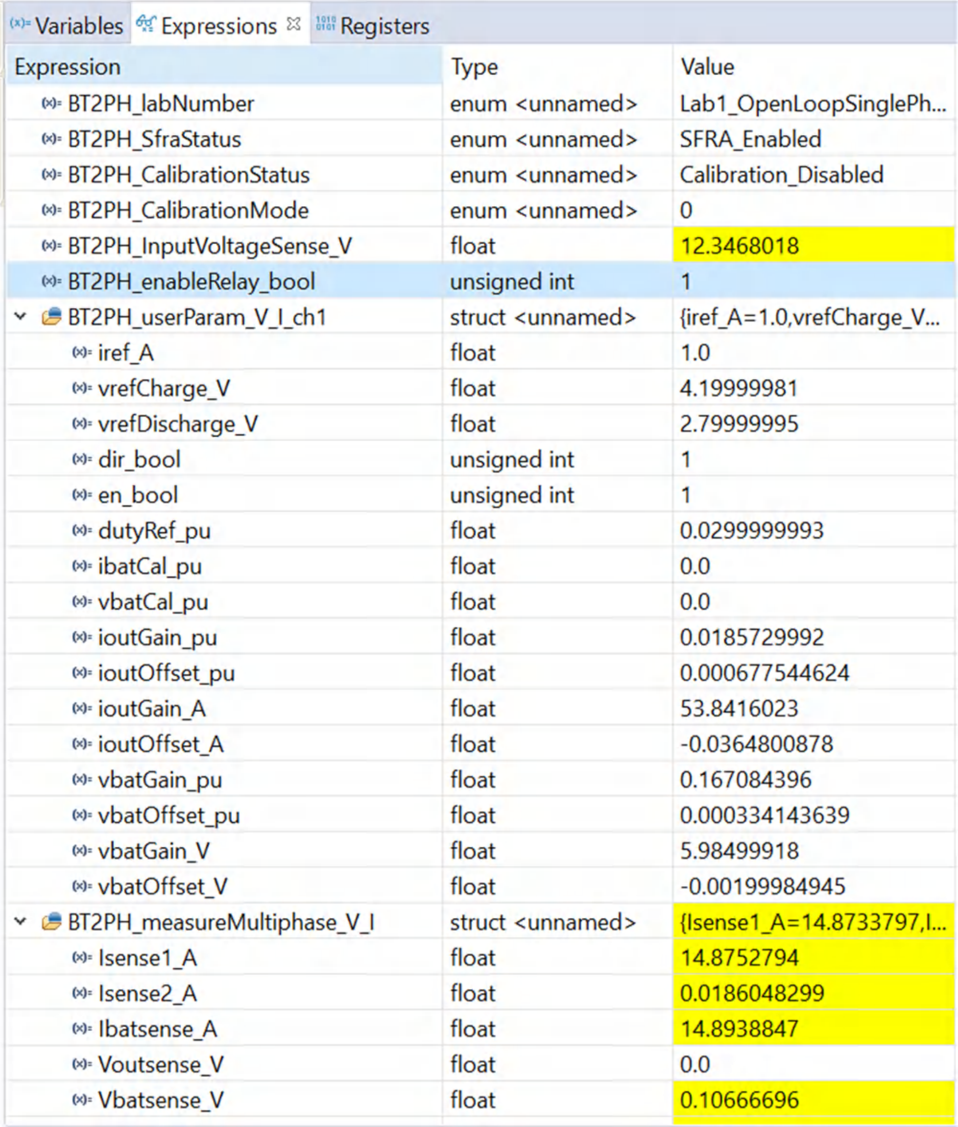
Horizontal
10.0 µs/div
SR: 100 MS/s
RL: 10 kpts

Trigger 1
1.48 V

Acquisition
Sample
19.93 kAcqs

26 Jun 2023
10:07:16

**Figure 3-12. CSn and DRDY Signals of the External ADC**



| 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,I... |
| 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 |

**Figure 3-13. Lab 1 Expression Window, Open Loop**

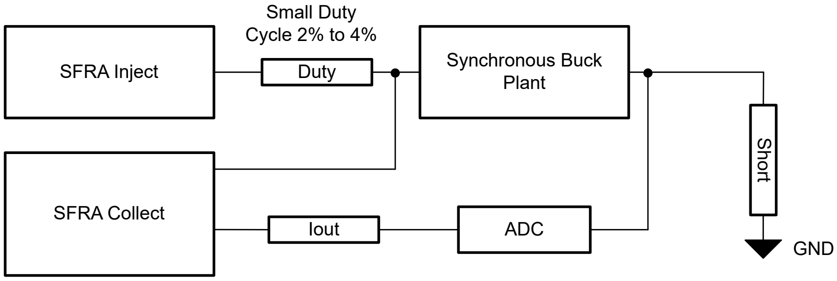
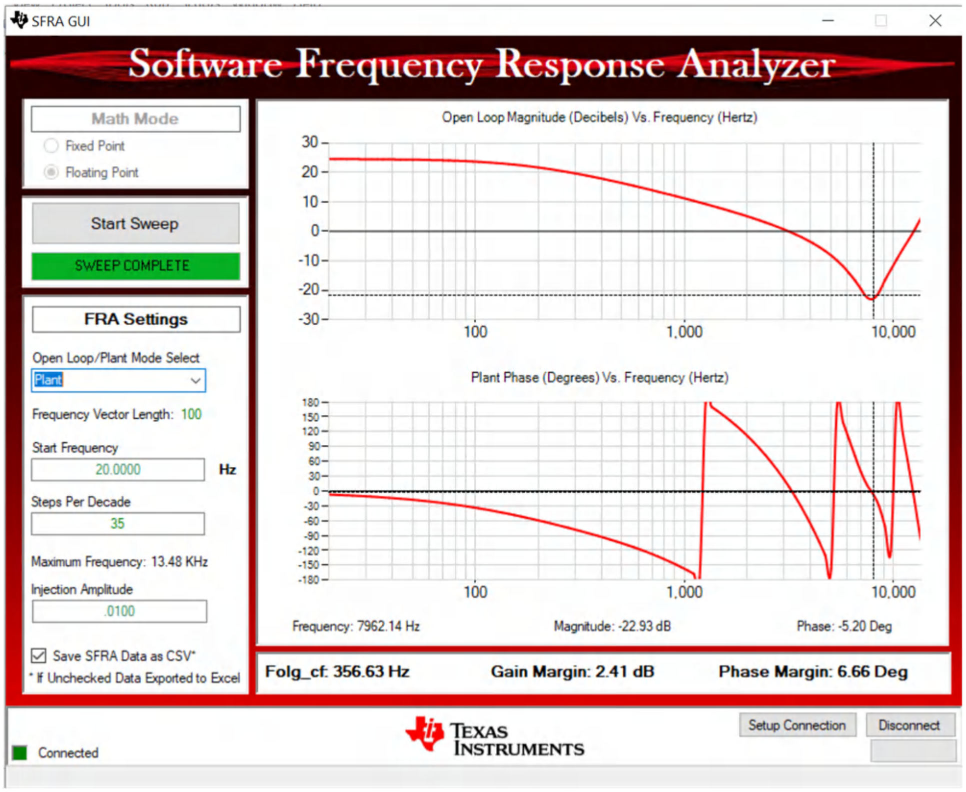


**Figure 3-14. SFRA Setup for Open-Loop Current Control**



Open Loop/Plant Mode Select: Plant

Frequency Vector Length: 100

Start Frequency 20.0000 Hz

Steps Per Decade 35

Maximum Frequency: 13.48 KHz

Injection Amplitude .0100

☑ Save SFRA Data as CSV*

* If Unchecked Data Exported to Excel

Frequency: 7962.14 Hz Magnitude: -22.93 dB Phase: -5.20 Deg

Folg_cf: 356.63 Hz Gain Margin: 2.41 dB Phase Margin: 6.66 Deg

🟩 Connected

Figure 3-15. Current Control Open-Loop Frequency Response

### 3.4.3 Lab 2. Closed Loop Current Control Single Phase

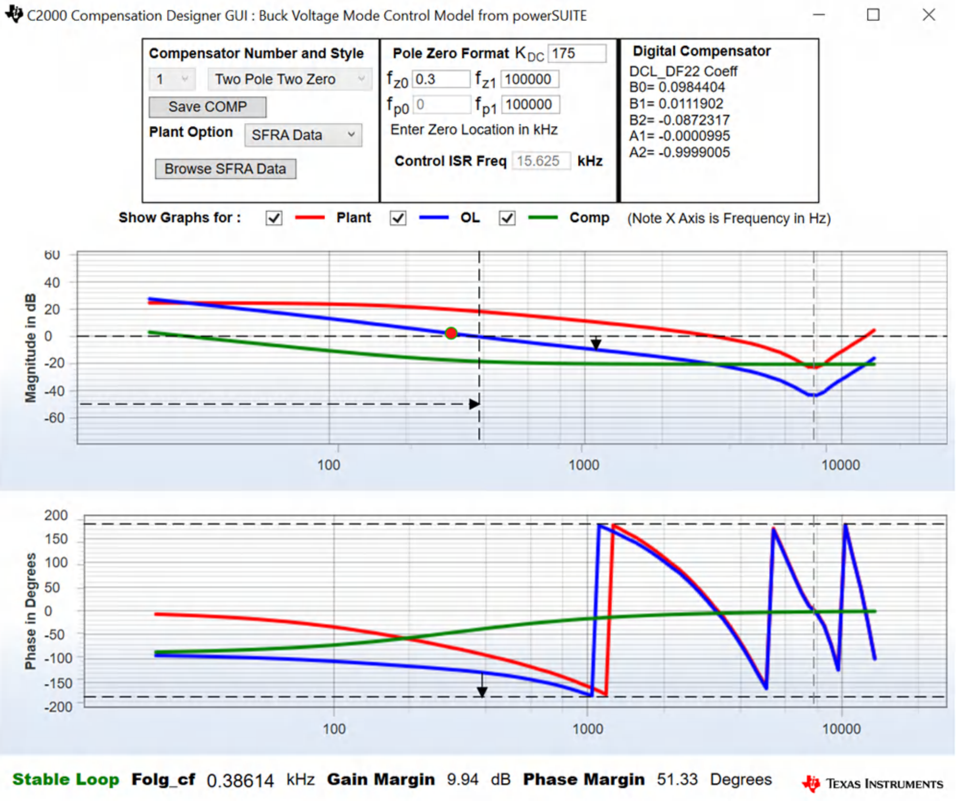
#### *3.4.3.1 Setting Software Options for Lab 2*

1. To run this lab, make sure the hardware is set up as outlined in the previous section,Section 3.4.2.
2. Open the CCS project as outlined in Section 3.2.1. If using the powerSUITE, go to Step 3, otherwise jump to Step 4.
3. Open the SYSCONFIG page and select under the *Build Options* section:
   - Select *Lab 2: Closed Loop CC Single Phase* for the Lab.
   - Change the *Phase Enabled* to Phase 1 or Phase 2>
   - Set the SFRA Enable/Disable to 1.
   - Open the Compensation Designer by clicking the *Run Compensation Design* button.
   - The compensation designer then launches and prompts the user to select a valid SFRA data file. Import the SFRA data from the run in Lab 1 into the compensation designer to design a two-pole, two-zero compensator. Keep more margins during this iteration of the design to make sure that when the loop is closed, the system is stable.
   - Figure 3-16 shows compensation parameters for the Current Loop.
   - Click on the *Save Comp* button to save the compensation. Close the Compensation Designer tool.
   - Save the SYSCONFIG page.
4. When using non-powerSuite version of the project, *Build Settings* are directly modified in *solution_settings.h* file. Compensation Designer is found at C2000Ware_DigitalPower_Install_Location\powerSUITE\source\utils.

```
#define LAB_NUMBER (2)

#define PHASE_NUMBER (1)

#define SFRA_ENABLED (true)
```



C2000 Compensation Designer GUI : Buck Voltage Mode Control Model from powerSUITE

**Compensator Number and Style**
1 Two Pole Two Zero
Save COMP
**Plant Option** SFRA Data
Browse SFRA Data

**Pole Zero Format** $K_{DC}$ 175
$f_{z0}$ 0.3 $f_{z1}$ 100000
$f_{p0}$ 0 $f_{p1}$ 100000
Enter Zero Location in kHz
**Control ISR Freq** 15.625 **kHz**

**Digital Compensator**
DCL_DF22 Coeff
B0= 0.0984404
B1= 0.0111902
B2= -0.0872317
A1= -0.0000995
A2= -0.9999005

**Show Graphs for :** ☑ **Plant** ☑ **OL** ☑ **Comp** (Note X Axis is Frequency in Hz)

**Stable Loop** **Folg_cf** 0.38614 kHz **Gain Margin** 9.94 dB **Phase Margin** 51.33 Degrees

**Figure 3-16. Tuning Current Loop Using Compensation Designer**

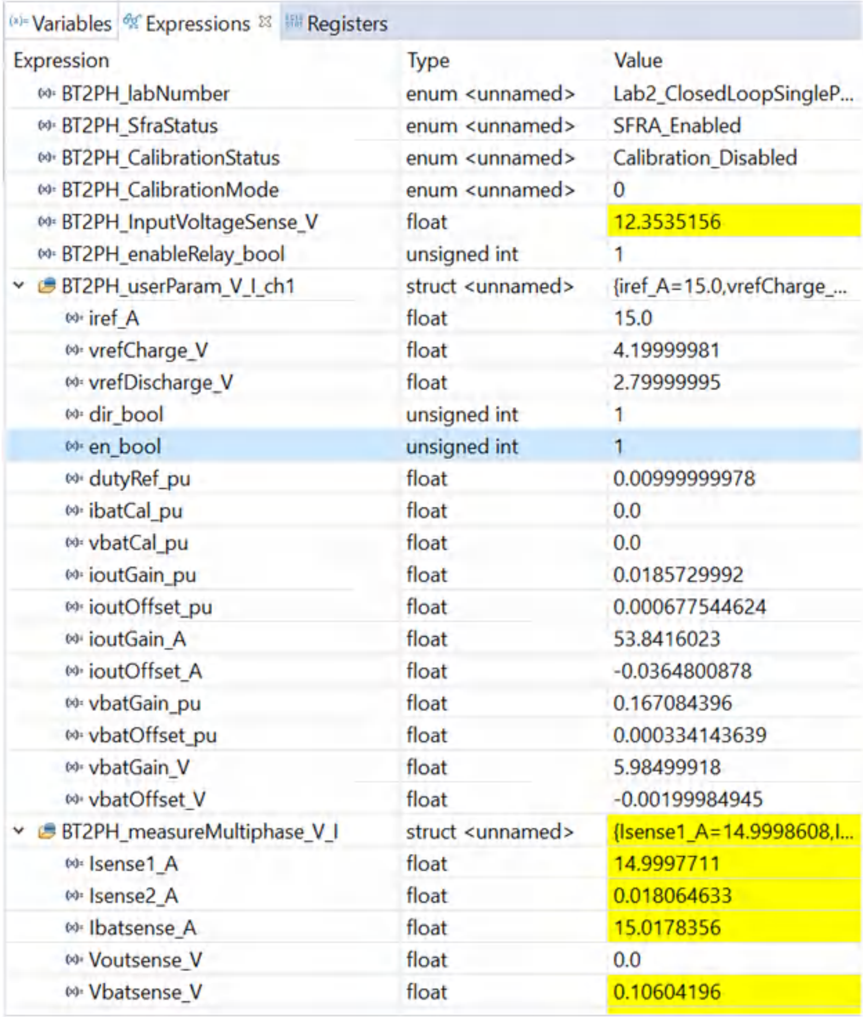
### 3.4.3.2 Building and Loading the Project and Setting up Debug Environment

1. Right-click on the project name and click *Rebuild Project*.
2. The project builds successfully.
3. In the Project Explorer, make sure the correct target configuration file is set as Active under targetConfigs.
4. Then, click *Run* → *Debug* to launch a debugging session.
5. The project then loads on the device and the CCS debug view becomes active. The code halts at the start of the main routine.
6. To add the variables in the watch/expressions window, click *View* → *Scripting Console* to open the scripting console dialog box. On the upper right corner of this console, click on *open* and then browse to the *setupdebugenv_lab2_ch1.js* script file located inside the project folder. This populates the watch window with the appropriate variables needed to debug the system. For phase 2, select *setupdebugenv_lab2_ch2.js* script file.
7. Click on the *Continuous Refresh* button on the watch window to enable continuous update of values from the controller.

### 3.4.3.3 Running the Code

1. To run this lab, make sure the hardware is set up as outlined in the previous section,Section 3.4.2.
2. Run the project by clicking from the menu bar.
3. In the watch view, check if the BT2PH_InputVoltageSense_V is between 12 V –15 V in the *Expression Window*.
4. Set the following parameters from the *Expression Window*:
   - Set the *BT2PH_enableRelay_bool* to 1 to enable the output relay.
   - BT2PH_userParam_V_I_ch1->iref_A = 15.0.
   - Set the BT2PH_userParam_V_I_ch1->en_bool = 1.
   - See Figure 3-17 for the Expression Window settings.
5. BT2PH_measureMultiphase_V_I variable shows output current and voltage of the DC/DC converter. Isense1_A display value is close to iref_A setting with ±1 mA error.

6. Figure 3-18 shows the SFRA setup to test the loop stability. Click on *Run SFRA* icon from the SYSCONFIG page. The SFRA GUI pops up.
7. Select the options for the device on the SFRA GUI; for example, for F280039, select floating point. Click on setup connection. In the pop-up window, uncheck the boot-on-connect option and select an appropriate COM port. Click Ok. Return to the SFRA GUI and click the *Connect* button.
8. The SFRA GUI connects to the device. A SFRA sweep can now be started by clicking *Start Sweep*. The complete SFRA sweep takes a few minutes to finish. Once complete, a graph with the measurement appears, as shown in Figure 3-19.
9. The Frequency Response Data is saved in the project folder, under an SFRA Data Folder, and is time-stamped with the time of the SFRA run.



| 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 |

**Figure 3-17. Lab 2 Expression Window, Closed Loop**

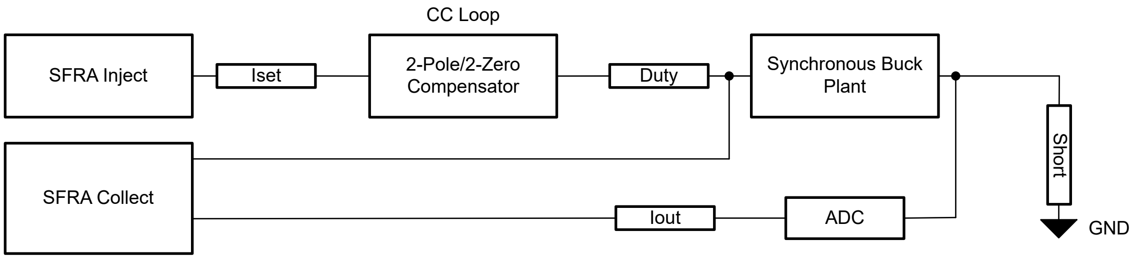
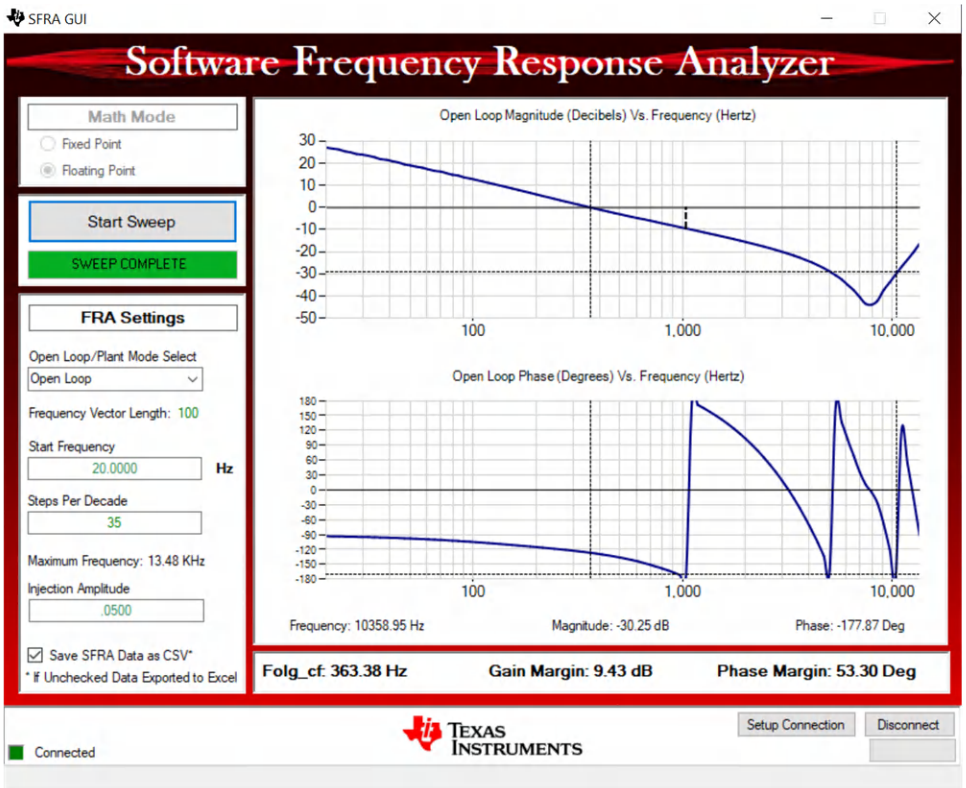


**Figure 3-18. SFRA Setup for Closed-Loop Current Control**



○ Fixed Point
● Floating Point

Open Loop/Plant Mode Select: Open Loop

Frequency Vector Length: 100

Start Frequency: 20.0000 Hz

Steps Per Decade: 35

Maximum Frequency: 13.48 KHz

Injection Amplitude: .0500

☑ Save SFRA Data as CSV*

* If Unchecked Data Exported to Excel

Frequency: 10358.95 Hz Magnitude: -30.25 dB Phase: -177.87 Deg

Folg_cf: 363.38 Hz Gain Margin: 9.43 dB Phase Margin: 53.30 Deg

🟩 Connected

**Figure 3-19. Current Control Closed-Loop Frequency Response**

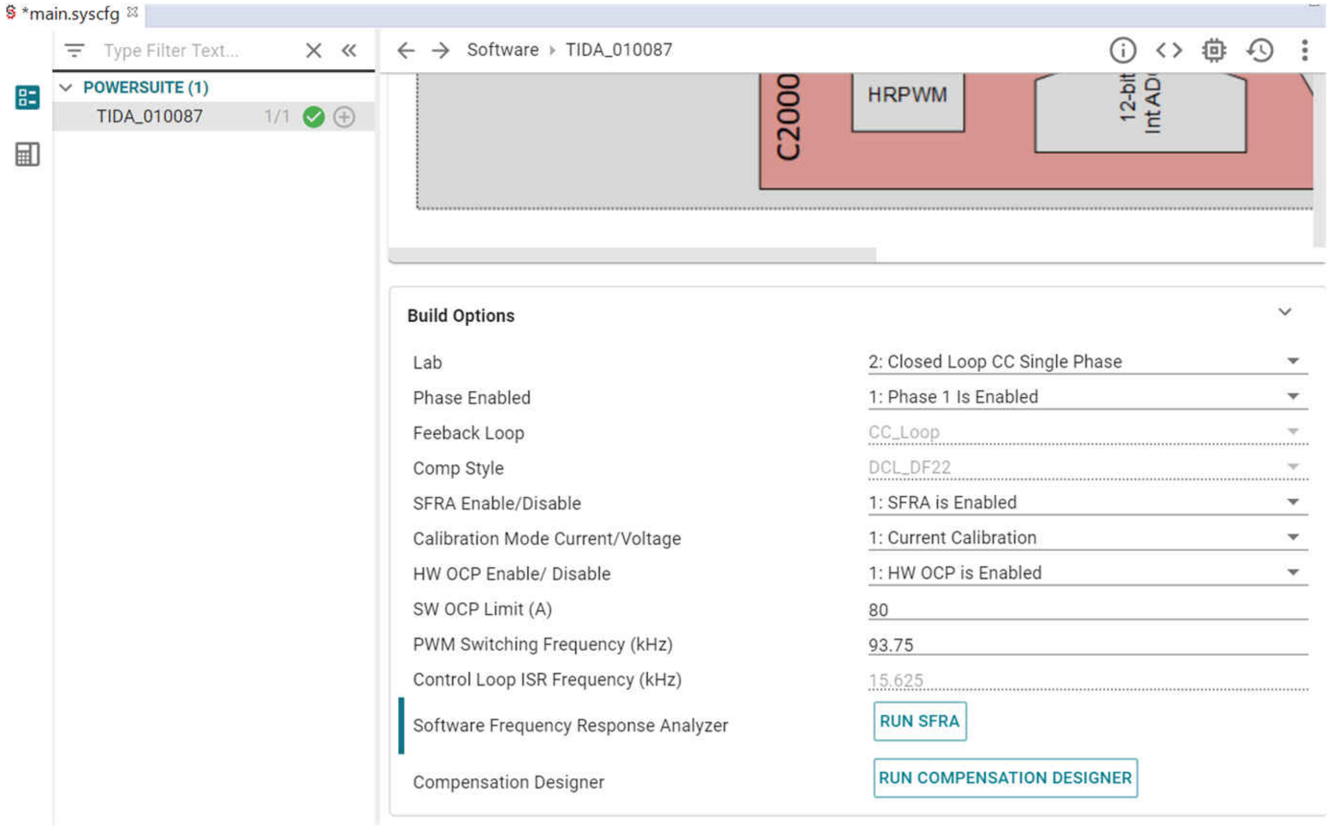
### *3.4.3.4 Current Calibration*

1. To run this lab, make sure the hardware is set up as shown in Section 3.3.3. The 2-points calibration method is used to calibrate gain and offset errors.
2. To measure current, use an external precision resistor and a DMM, or you can use E-Load current readings. Alternatively, voltage across sense resistors on the TIDA-010087 boards can be used to measure the output current.
3. Open the SYSCONFIG page and select under the *Build Options* section:
   - Select *Lab 2: Closed Loop CC Single Phase* for the Lab.
   - Change the *Phase Enabled* to Phase 1.
   - Set the *Calibration Mode* to 1 for current calibration.
   - Save the SYSCONFIG page, and run the code.
   - Open the *Expression Window*.
   - The output current is updated using BT2PH_userParam_V_I_ch1->ibatCal_pu parameter.
   - Set the *BT2PH_enableRelay_bool* to 1 to enable the output relay.
   - Set the BT2PH_userParam_V_I_ch1->en_bool = 1.
   - Set the BT2PH_userParam_V_I_ch1->ibatCal_pu to "0.05" and "0.3", and note the output current readings.
   - Update the actual output current readings in bt2ph_cal.h file.

   ```
   #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)
   ```

   - Repeat the steps for phase 2 of the converter.
   - Set the *Calibration Mode* to 0 disable calibration.
4. When using non-powerSuite version of the project, *Build Settings* are directly modified in *solution_settings.h* file.

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



**Build Options**

| | |
|---|---|
| Lab | 2: Closed Loop CC Single Phase |
| Phase Enabled | 1: Phase 1 Is Enabled |
| Feeback Loop | CC_Loop |
| Comp Style | DCL_DF22 |
| SFRA Enable/Disable | 1: SFRA is Enabled |
| Calibration Mode Current/Voltage | 1: Current Calibration |
| HW OCP Enable/ Disable | 1: HW OCP is Enabled |
| SW OCP Limit (A) | 80 |
| PWM Switching Frequency (kHz) | 93.75 |
| Control Loop ISR Frequency (kHz) | 15.625 |
| Software Frequency Response Analyzer | RUN SFRA |
| Compensation Designer | RUN COMPENSATION DESIGNER |

**Figure 3-20. Build Options for Current Calibration**

### 3.4.4 Lab 3. Closed Loop Current Control Dual Phase

#### *3.4.4.1 Setting Software Options for Lab 3*

1. Use the test setup shown in Section 3.3.1.
2. Open the CCS project as outlined in Section 3.2.1. If using the powerSUITE, go to Step 3, otherwise jump to Step 4.
3. Open the SYSCONFIG page and select under the *Build Options* section:
   - Select *Lab 3: Closed-Loop CC Dual Phase* for the Lab.
   - Change the *Phase Enabled* to 0 for dual-phase mode.
   - Set the SFRA Enable/Disable to 1.
   - Save the page.
4. When using non-powerSuite version of the project, above settings are directly modified in *solution_settings.h* file.

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

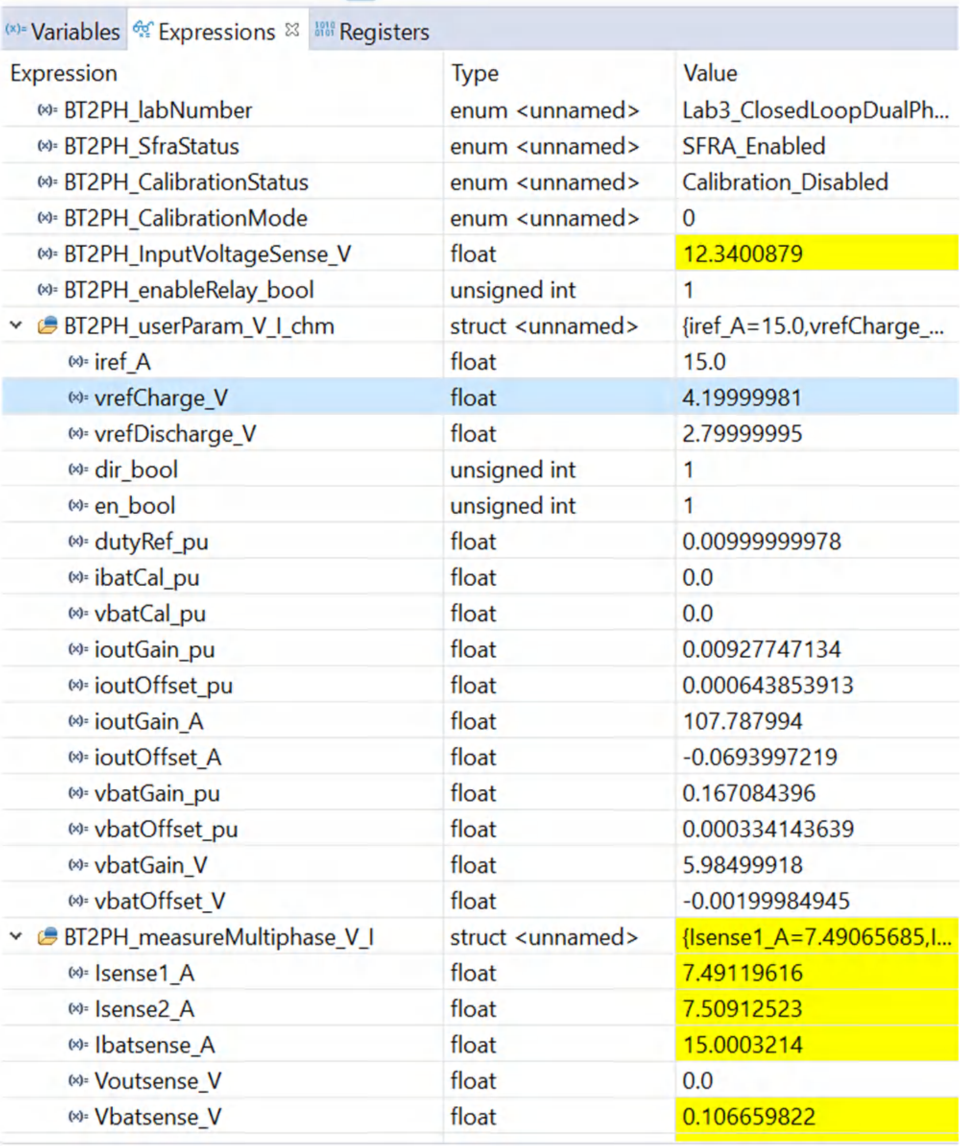
#### *3.4.4.2 Building and Loading the Project and Setting up Debug Environment*

1. Right-click on the project name and click *Rebuild Project*.
2. The project builds successfully.

3. In the Project Explorer, make sure the correct target configuration file is set as Active under targetConfigs.
4. Then, click *Run* → *Debug* to launch a debugging session.
5. The project then loads on the device and the CCS debug view becomes active. The code halts at the start of the main routine.
6. To add the variables in the watch/expressions window, click *View* → *Scripting* Console to open the scripting console dialog box. On the upper right corner of this console, click on the *Open* button and then browse to the *setupdebugenv_lab3.js* script file located inside the project folder. This populates the watch window with the appropriate variables needed to debug the system.
7. Click on the *Continuous Refresh* button ⟳ on the watch window to enable continuous update of values from the controller.

#### 3.4.4.3 Running the Code

1. Use the test setup shown in Section 3.3.1.
2. Run the project by clicking ▶ from the menu bar.
3. In the watch view, check if the BT2PH_InputVoltageSense_V is between 12 V – 15 V in the *Expression Window*.
4. Set the following parameters from the *Expression Window*:
   - Set the *BT2PH_enableRelay_bool* to 1 to enable the output relay.
   - BT2PH_userParam_V_I_chm->iref_A = 15.0.
   - Set the BT2PH_userParam_V_I_ch1->en_bool = 1.
   - See Figure 3-21 for the *Expression Window* settings.
5. BT2PH_measureMultiphase_V_I variable shows output current and voltage of the DC/DC converter. Ibatsense_A display value is close to iref_A with ±1 mA error.
6. Figure 3-22 shows the SFRA setup to measure Open-Loop Voltage Control Frequency Response.
7. Click on the *Run SFRA* icon from the SYSCONFIG page. The SFRA GUI pops up
8. Select the options for the device on the SFRA GUI; for example, for F280039, select floating point. Click on the *Setup Connection* button. In the pop-up window, uncheck the boot-on-connect option and select an appropriate COM port. Click Ok. Return to the SFRA GUI and click the *Connect* button.
9. The SFRA GUI connects to the device. A SFRA sweep can now be started by clicking the *Start Sweep* button. The complete SFRA sweep takes a few minutes to finish. Once complete, a graph with the measurement appears, as shown in Figure 3-23.
10. The Frequency Response Data is saved in the project folder, under an SFRA Data Folder, and is time-stamped with the time of the SFRA run.



| 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 |

**Figure 3-21. Lab 3 Expression Window, Closed Loop**

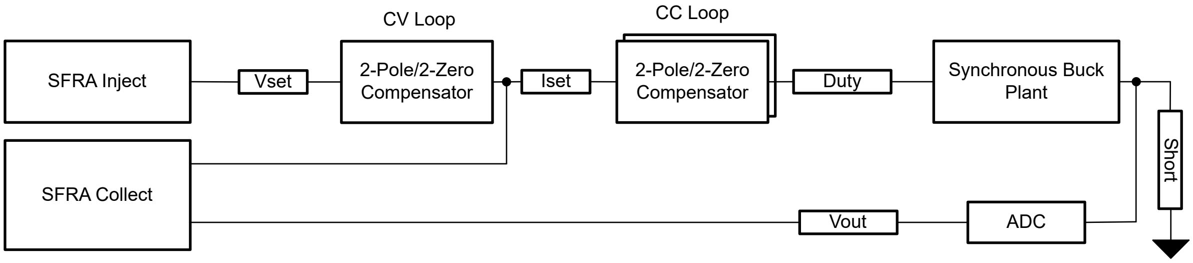
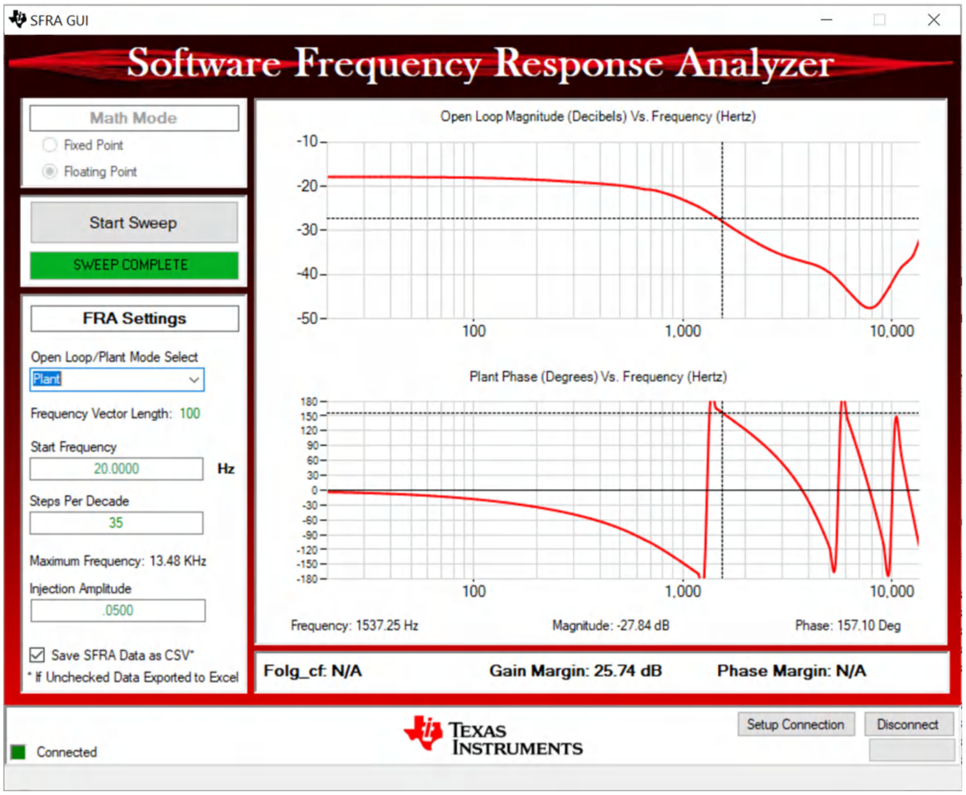


**Figure 3-22. SFRA Setup for Open-Loop Voltage Control**



○ Fixed Point

● Floating Point

Open Loop/Plant Mode Select: Plant

Frequency Vector Length: 100

Start Frequency: 20.0000 Hz

Steps Per Decade: 35

Maximum Frequency: 13.48 KHz

Injection Amplitude: .0500

☑ Save SFRA Data as CSV*

* If Unchecked Data Exported to Excel

Frequency: 1537.25 Hz  Magnitude: -27.84 dB  Phase: 157.10 Deg

Folg_cf: N/A  Gain Margin: 25.74 dB  Phase Margin: N/A

🟩 Connected

**Figure 3-23. Voltage Control Open-Loop Frequency Response**

### 3.4.5 Lab 4. Closed Loop Current and Voltage Control

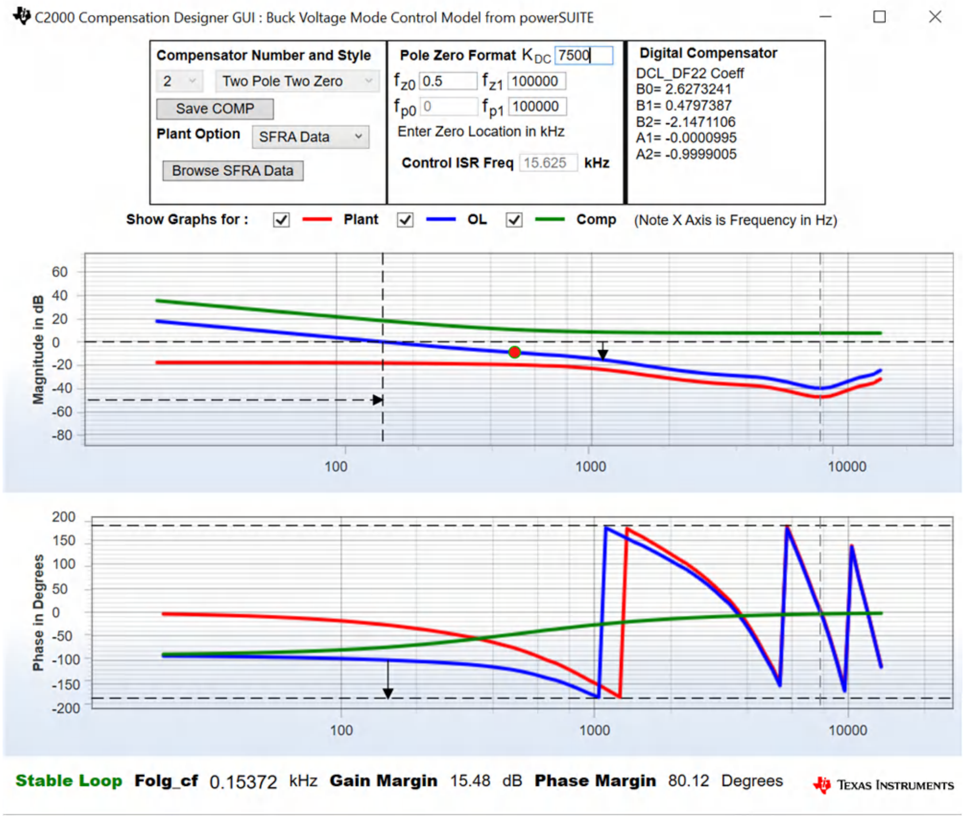
#### *3.4.5.1 Setting Software Options for Lab 4*

1. Use the test setup shown in Section 3.3.1.
2. Open the CCS project as outlined in Section 3.2.1. If using the powerSUITE, go to Step 3, otherwise jump to Step 4.
3. Open the SYSCONFIG page and select under the *Build Options* section:
   - Select *Lab 4: Closed-Loop CCCV Dual Phase* for the Lab.
   - Change the *Phase Enabled* to 0 for dual-phase operation.
   - Set the SFRA Enable/Disable to 1.
   - Open the Compensation Designer by clicking the *Run Compensation Design* button.
   - The compensation designer then launches and prompts the user to select a valid SFRA data file. Import the SFRA data from the run in Lab 1 into the compensation designer to design a two-pole, two-zero compensator. Keep more margins during this iteration of the design to make sure that when the loop is closed, the system is stable.
   - Figure 3-24 shows compensation parameters for the Voltage Loop.
   - Click the *Save Comp* button to save the compensation. Close the *Compensation Designer* tool.
   - Save the SYSCONFIG page.
4. When using the non-powerSuite version of the project, *Build Settings* are directly modified in the *solution_settings.h* file. *Compensation Designer* is found at C2000Ware_DigitalPower_Install_Location\powerSUITE\source\utils

```
#define LAB_NUMBER (4)

#define PHASE_NUMBER (0)

#define SFRA_ENABLED (true)
```



Compensator Number and Style
2 | Two Pole Two Zero
Save COMP
Plant Option: SFRA Data
Browse SFRA Data

Pole Zero Format $K_{DC}$ 7500
$f_{z0}$ 0.5 $f_{z1}$ 100000
$f_{p0}$ 0 $f_{p1}$ 100000
Enter Zero Location in kHz
Control ISR Freq 15.625 kHz

Digital Compensator
DCL_DF22 Coeff
B0= 2.6273241
B1= 0.4797387
B2= -2.1471106
A1= -0.0000995
A2= -0.9999005

Show Graphs for : ☑ Plant ☑ OL ☑ Comp

Stable Loop Folg_cf 0.15372 kHz Gain Margin 15.48 dB Phase Margin 80.12 Degrees

**Figure 3-24. Tuning Voltage Loop Using Compensation Designer**

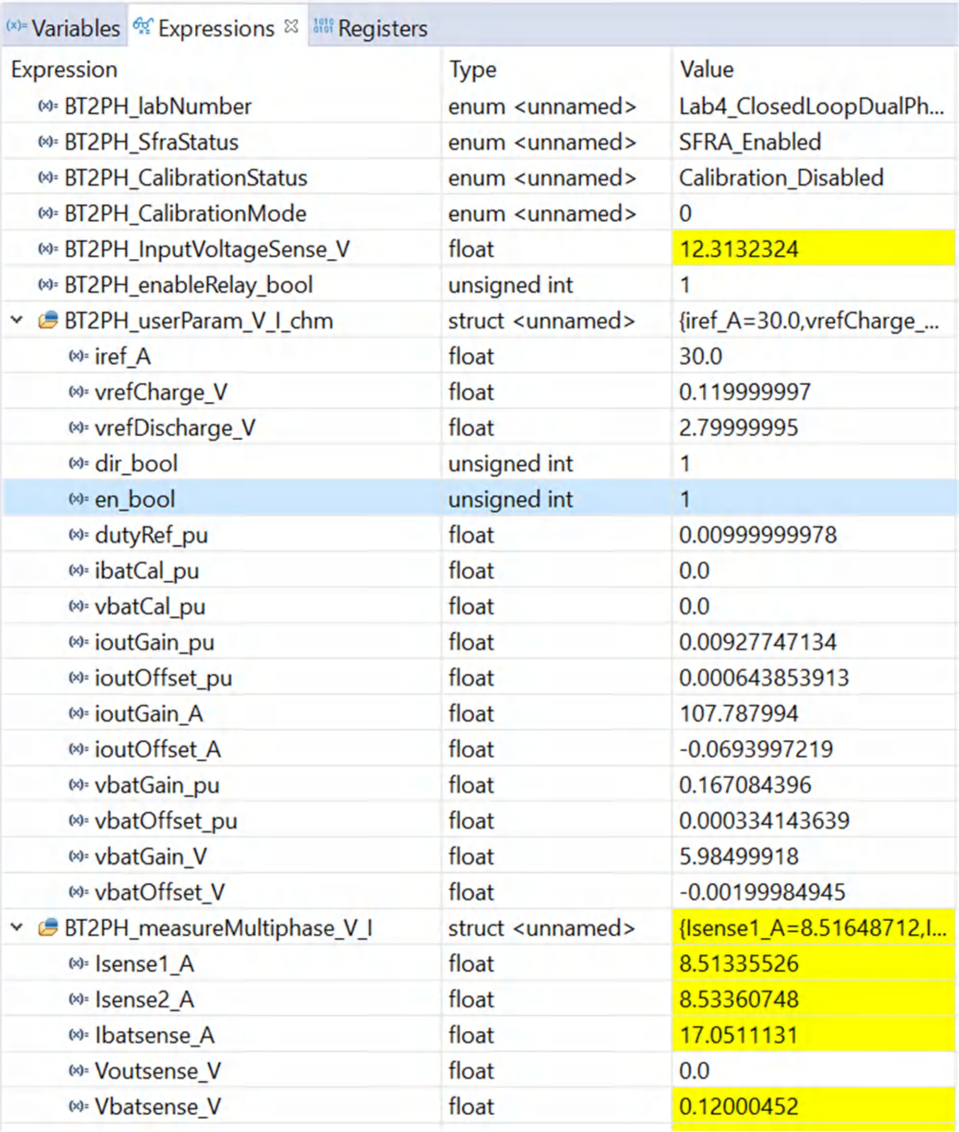
### *3.4.5.2 Building and Loading the Project and Setting up Debug Environment*

1. Right-click on the project name and click *Rebuild Project*.
2. The project builds successfully.
3. In the Project Explorer, make sure the correct target configuration file is set as *Active* under targetConfigs.
4. Then, click *Run* → *Debug* to launch a debugging session.
5. The project then loads on the device and the CCS debug view becomes active. The code halts at the start of the main routine.
6. To add the variables in the watch/expressions window, click *View* → *Scripting Console* to open the scripting console dialog box. On the upper right corner of this console, click on *open* and then browse to the *setupdebugenv_lab4.js* script file located inside the project folder. This populates the watch window with the appropriate variables needed to debug the system.
7. Click on the *Continuous Refresh* button on the watch window to enable continuous update of values from the controller.

### *3.4.5.3 Running the Code*

1. Use the test setup shown in Section 3.3.1.
2. Run the project by clicking from the menu bar.
3. In the watch view, check if the BT2PH_InputVoltageSense_V is between 12 V – 15 V in the *Expression Window*.
4. Set the following parameters from the *Expression Window*:
   - Set the *BT2PH_enableRelay_bool* to 1 to enable the output relay.
   - BT2PH_userParam_V_I_chm->iref_A = 30.0.
   - BT2PH_userParam_V_I_chm->vrefCharge_V = 0.12.
   - Set the BT2PH_userParam_V_I_ch1->en_bool = 1.
   - See the Figure 3-25 for the *Expression Window* settings.
5. BT2PH_measureMultiphase_V_I variable shows output current and voltage of the DC/DC converter. Vbatsense_V display value is close to vrefCharge_V with ±0.5-mV error.

6. Figure 3-26 shows the SFRA setup to measure Closed-Loop Voltage Control Frequency Response.
7. Click on the *Run SFRA* icon from the SYSCONFIG page. The SFRA GUI pops up
8. Select the options for the device on the SFRA GUI; for example, for F280039, select floating point. Click the *Setup Connection* button. In the pop-up window, uncheck the boot-on-connect option and select an appropriate COM port. Click Ok. Return to the SFRA GUI and select the *Connect* button.
9. The SFRA GUI connects to the device. An SFRA sweep can now be started by clicking the *Start Sweep* button. The complete SFRA sweep takes a few minutes to finish. Once complete, a graph with the measurement appears, as shown in Figure 3-27.
10. The Frequency Response Data is saved in the project folder, under an SFRA Data Folder, and is time-stamped with the time of the SFRA run.



| 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 |

**Figure 3-25. Lab 4 Expression Window, Closed Loop**

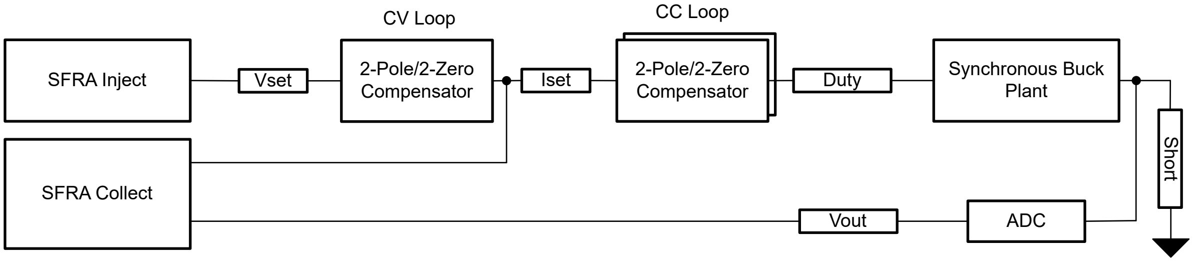
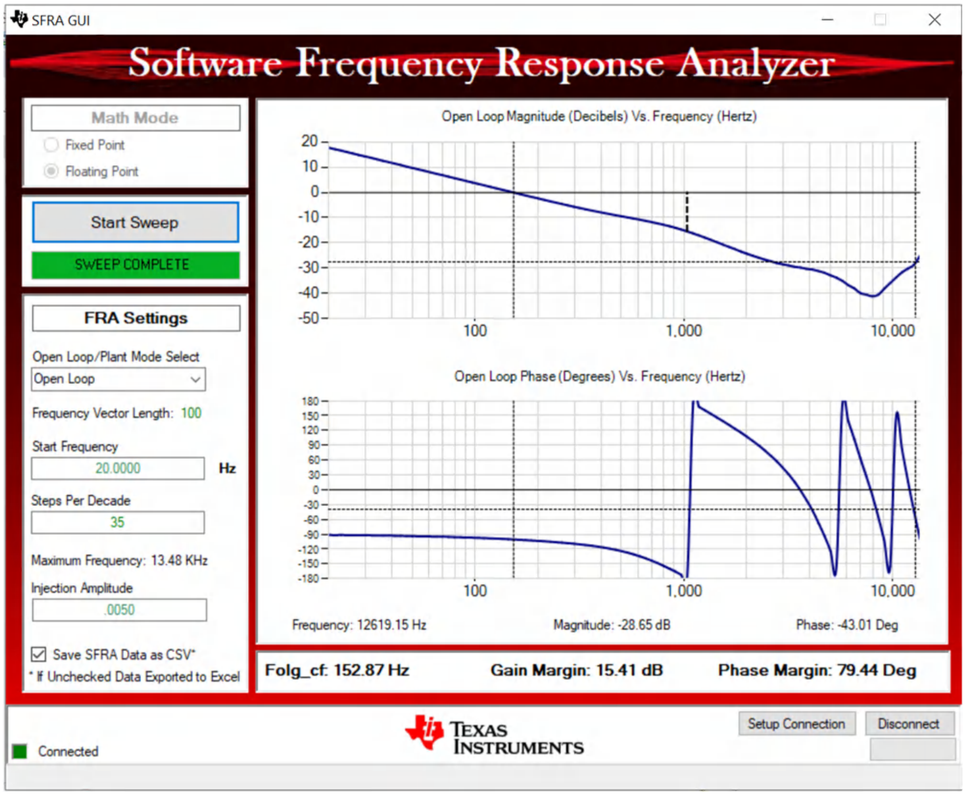


**Figure 3-26. SFRA Setup for Closed-Loop Voltage Control**



○ Fixed Point
● Floating Point

Open Loop/Plant Mode Select: Open Loop

Frequency Vector Length: 100

Start Frequency: 20.0000 Hz

Steps Per Decade: 35

Maximum Frequency: 13.48 KHz

Injection Amplitude: .0050

☑ Save SFRA Data as CSV*

* If Unchecked Data Exported to Excel

Frequency: 12619.15 Hz Magnitude: -28.65 dB Phase: -43.01 Deg

Folg_cf: 152.87 Hz Gain Margin: 15.41 dB Phase Margin: 79.44 Deg

**Figure 3-27. Voltage Control Closed-Loop Frequency Response**

### *3.4.5.4 Voltage Calibration*

1. To run this lab, make sure the hardware is set up as shown in, Section 3.3.3. E-Load can be turned off during calibration or set the E-Load Current Settings to smaller than iref_A to make sure the circuit is in constant-voltage mode. 2-points calibration method is used to calibrate gain and offset errors.
2. To measure voltage, use a DMM.
3. Open the SYSCONFIG page and select under the *Build Options* section:
   - Select *Lab 4: Closed-Loop CCCV Dual Phase* for the Lab.
   - Change the *Phase Enabled* to 0 for Dual-Phase Mode.
   - Set the *Calibration Mode* to 2 for voltage calibration.
   - Save the SYSCONFIG page, and run the code.
   - Open the *Expression Window*.
   - The output current is updated using BT2PH_userParam_V_I_chm->ibatCal_pu parameter.
   - Set the *BT2PH_enableRelay_bool* to 1 to enable the output relay.
   - Set the BT2PH_userParam_V_I_chm->en_bool = 1.
   - Set the BT2PH_userParam_V_I_chm->vbatCal_pu to "0.2" and "0.6", and note the output voltage readings.
   - Update the actual output voltage readings in bt2ph_cal.h file.

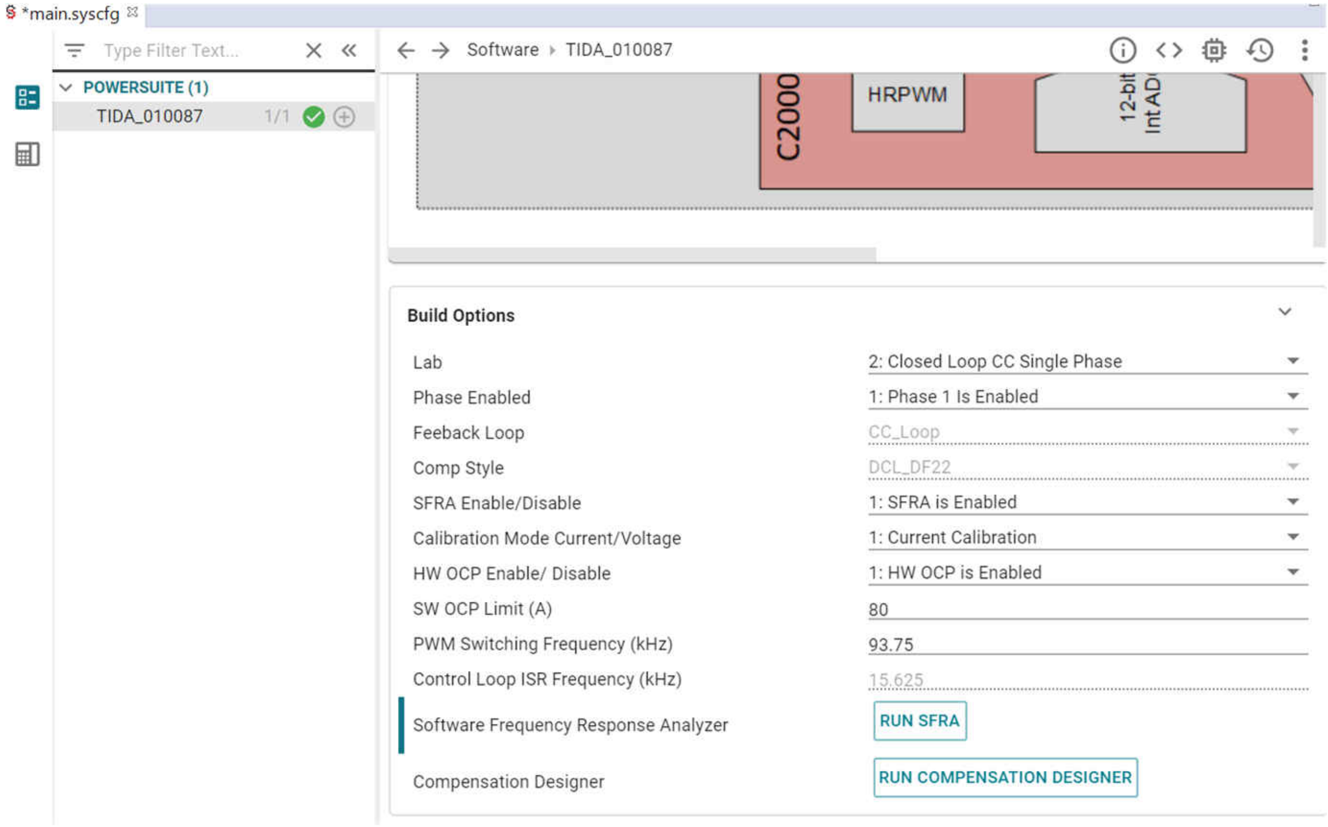
```
#define BT2PH_VBAT_ACTUAL_CH1_P1_V ((float32_t)1.1995)
```

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

- Set the *Calibration Mode* to 0 disable calibration.

4. When using non-powerSuite version of the project, *Build Settings* are directly modified in the *solution_settings.h* file:

```
#define LAB_NUMBER (4)
#define PHASE_NUMBER (0)
#define CALIBRATION_ENABLED (true)
#define CALIBRATION_MODE (2)
```



Build Options

| | |
|---|---|
| Lab | 2: Closed Loop CC Single Phase |
| Phase Enabled | 1: Phase 1 Is Enabled |
| Feeback Loop | CC_Loop |
| Comp Style | DCL_DF22 |
| SFRA Enable/Disable | 1: SFRA is Enabled |
| Calibration Mode Current/Voltage | 1: Current Calibration |
| HW OCP Enable/ Disable | 1: HW OCP is Enabled |
| SW OCP Limit (A) | 80 |
| PWM Switching Frequency (kHz) | 93.75 |
| Control Loop ISR Frequency (kHz) | 15.625 |
| Software Frequency Response Analyzer | RUN SFRA |
| Compensation Designer | RUN COMPENSATION DESIGNER |

**Figure 3-28. Build Options for Current Calibration**

## 3.5 Test Results

### 3.5.1 Current Loop Load Regulation Error

**Table 3-2. Current Loop Load Regulation Error in Charge Mode**

| **FSR (A)** | **100** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **TIDA_010087 ISET (A)** | 0.1 | 0.5 | 1 | 5 | 10 | 20 | 25 | 30 | 39 |
| **ELOAD CV Mode** | **E-LOAD Reading** | | | | | | | | |
| **VSET (V)** | **Iactual (A)** | **Iactual (A)** | **Iactual (A)** | **Iactual (A)** | **Iactual (A)** | **Iactual (A)** | **Iactual (A)** | **Iactual (A)** | **Iactual (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 |
| 4.2 | 0.112 | 0.51 | 1.01 | 5.008 | 10.005 | 20.001 | 25 | 30.001 | 39.001 |
| **Error (mA)** | 0.018 | 0.016 | 0.015 | 0.012 | 0.009 | 0.004 | 0.004 | 0.006 | 0.009 |
| **Error (% FSR)** | 0.018 | 0.016 | 0.015 | 0.012 | 0.009 | 0.004 | 0.004 | 0.006 | 0.009 |

### 3.5.2 Voltage Loop Load Regulation Error

**Table 3-3. Voltage Loop Load Regulation Error in Charge Mode**

| **Full Scale (V)** | 6.25 | | | |
|---|---|---|---|---|
| **TIDA_010087 VSET (V)** | 1 | 2 | 3 | 4.2 |
| **ELOAD CC Mode** | | | | |
| **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 |
| **Error (mV)** | 0.00018 | 0.00039 | 0.00022 | 2E-05 |
| **Error (% FSR)** | 0.00288 | 0.00624 | 0.00352 | 0.00032 |

### 3.5.3 Voltage Transition at No Load

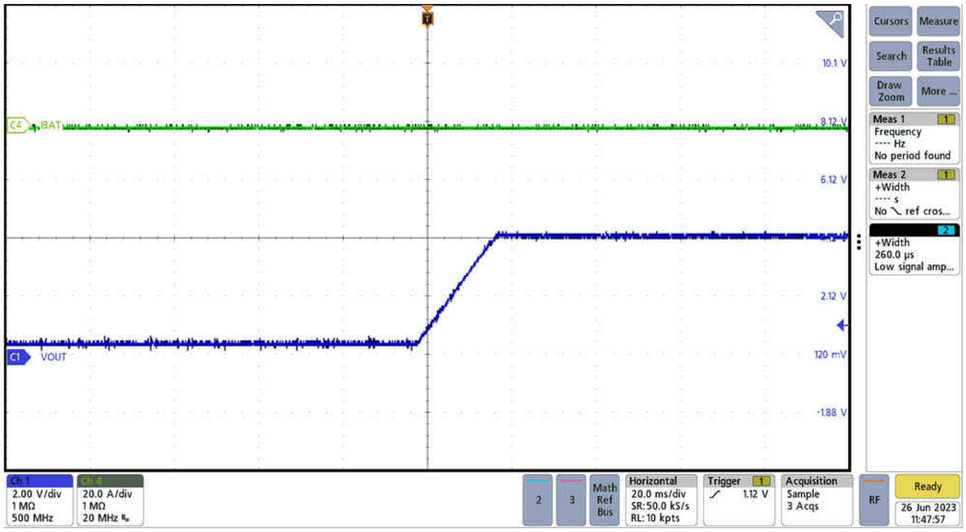
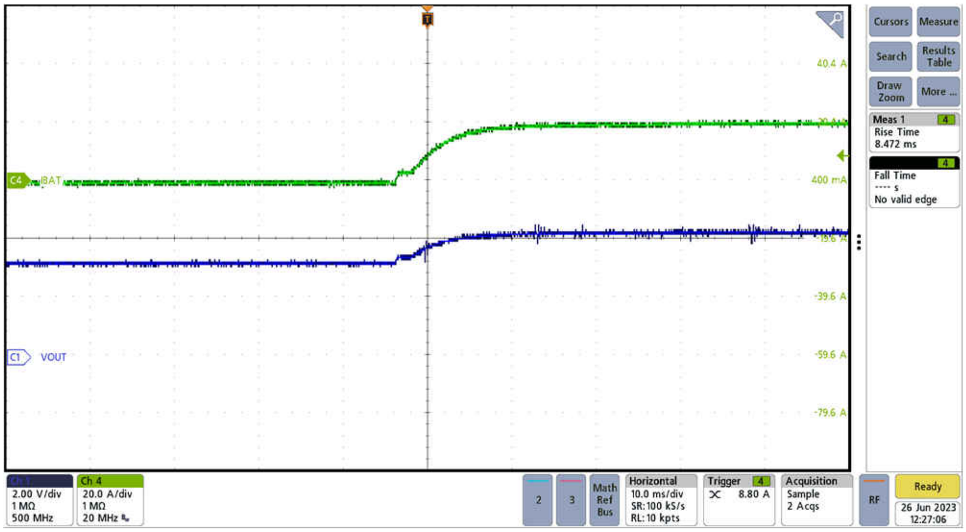


**Figure 3-29. Voltage Transition at No Load**

### 3.5.4 Transient Response at Start-Up



Meas 1 4
Rise Time
8.472 ms

4
Fall Time
---- s
No valid edge

Ch 1
2.00 V/div
1 MΩ
500 MHz

Ch 4
20.0 A/div
1 MΩ
20 MHz

Horizontal
10.0 ms/div
SR:100 kS/s
RL:10 kpts

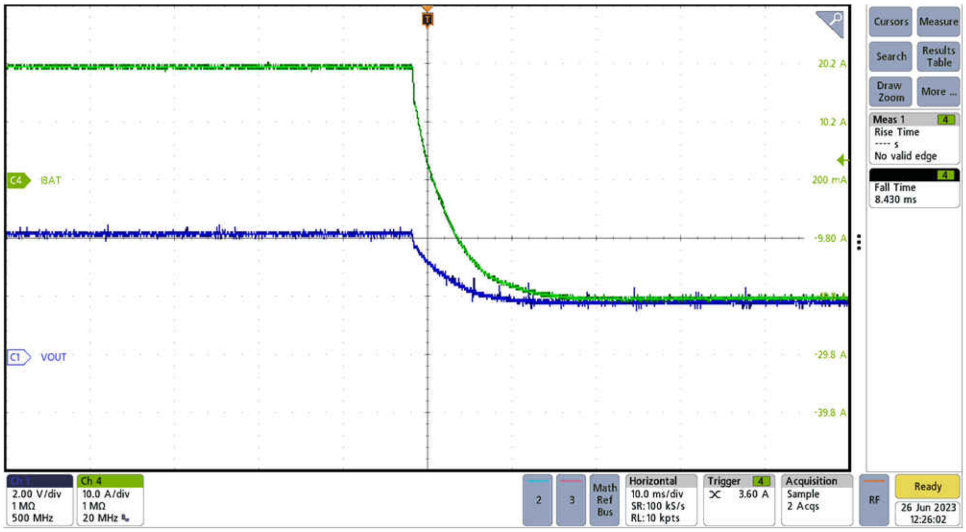
Trigger 4
8.80 A

Acquisition
Sample
2 Acqs

26 Jun 2023
12:27:06

**Figure 3-30. Transient Response at Start-Up**

### 3.5.5 Bidirectional Current Switching Time



Meas 1 4
Rise Time
---- s
No valid edge

4
Fall Time
8.430 ms

Ch 1
2.00 V/div
1 MΩ
500 MHz

Ch 4
10.0 A/div
1 MΩ
20 MHz

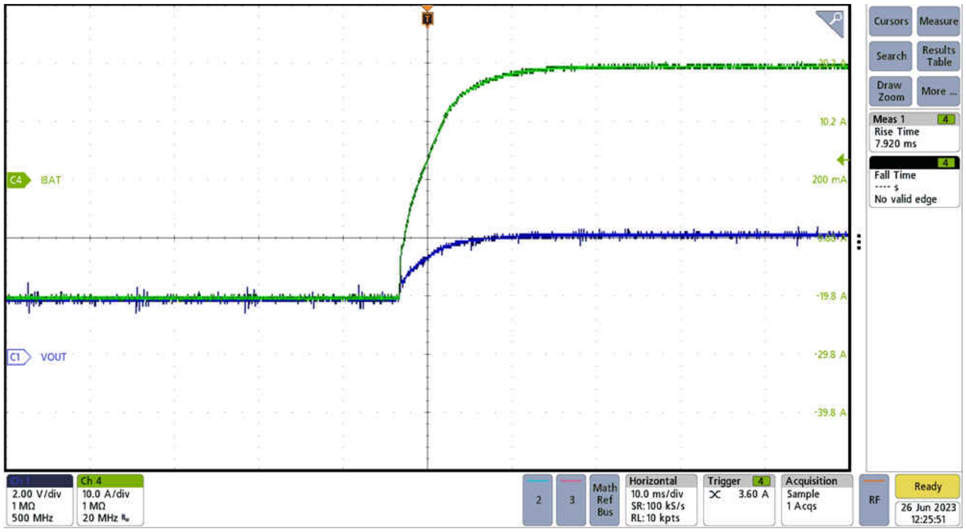
Horizontal
10.0 ms/div
SR:100 kS/s
RL:10 kpts

Trigger 4
3.60 A

Acquisition
Sample
2 Acqs

26 Jun 2023
12:26:02

**Figure 3-31. Current Transition, Charge to Discharge Mode**



Meas 1 4
Rise Time
7.920 ms

4
Fall Time
---- s
No valid edge

Ch 1
2.00 V/div
1 MΩ
500 MHz

Ch 4
10.0 A/div
1 MΩ
20 MHz

Horizontal
10.0 ms/div
SR: 100 kS/s
RL: 10 kpts

Trigger 4
3.60 A

Acquisition
Sample
1 Acqs

26 Jun 2023
12:25:51

Figure 3-32. Current Transition, Discharge to Charge Mode

# 4 Design and Documentation Support

## 4.1 Design Files

### 4.1.1 Schematics

To download the schematics, see the design files at TIDA-010087.

### 4.1.2 BOM

To download the bill of materials (BOM), see the design files at TIDA-010087.

## 4.2 Tools and Software

**Tools**

| | |
|---|---|
| TMDSCNCD280039C | F280039C controlCARD Evaluation Module |

**Software**

| | |
|---|---|
| CCSTUDIO | Code Composer Studio (CCS) Integrated Development Environment (IDE) |
| C2000WARE-DIGITALPOWER-SDK | DigitalPower software development kit (SDK) for C2000™ MCUs |

## 4.3 Documentation Support

1. Texas Instruments, *TMS320F28003x Real-Time Microcontrollers* Data Sheet
2. Texas Instruments, *ADS131M08 8-Channel, Simultaneously-Sampling, 24-Bit, Delta-Sigma ADC* Data Sheet

## 4.4 Support Resources

TI E2E™ support forums are an engineer's go-to source for fast, verified answers and design help — straight from the experts. Search existing answers or ask your own question to get the quick design help you need.

Linked content is provided "AS IS" by the respective contributors. They do not constitute TI specifications and do not necessarily reflect TI's views; see TI's Terms of Use.

## 4.5 Trademarks

TI E2E™ and C2000™, and are trademarks of Texas Instruments.
All trademarks are the property of their respective owners.

# 5 About the Author

**SHAURY ANAND** is a systems engineer at Texas Instruments, where he is responsible for developing reference designs for Test & Measurement applications. Shaury earned his bachelor’s degree (B.Tech) in electrical engineering from the Indian Institute of Technology, Roorkee.

The author thanks VICTOR SALOMON and OZINO ODHARO for the support given with this reference design.

# IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265