*TI Designs*

# *Vienna Rectifier-Based, Three-Phase Power Factor Correction (PFC) Reference Design Using C2000™ MCU*

## Description

The Vienna rectifier power topology is used in high-power, three-phase power factor correction applications such as offboard electric vehicle (EV) chargers and telecom rectifiers. Control design of the rectifier can be complex. This TI Design illustrates a method to control the power stage using C2000™ microcontroller (MCU). The hardware and software available with this design helps accelerate the time to market.

## Resources

| | |
|---|---|
| TIDM-1000 | Design Folder |
| TMS320F28377D | Product Folder |
| UCC21520DW | Product Folder |
| AMC1301 | Product Folder |

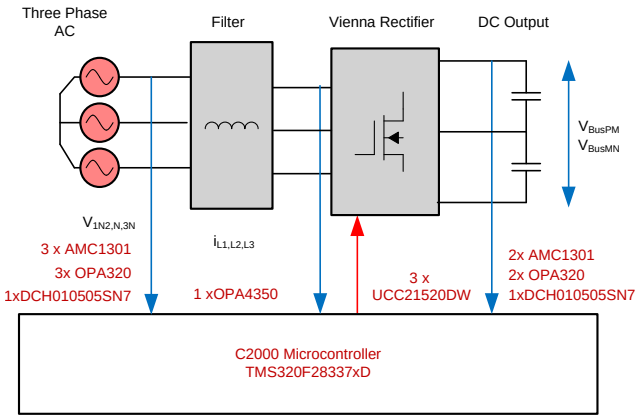
TI E2E™ Community — ASK Our E2E Experts

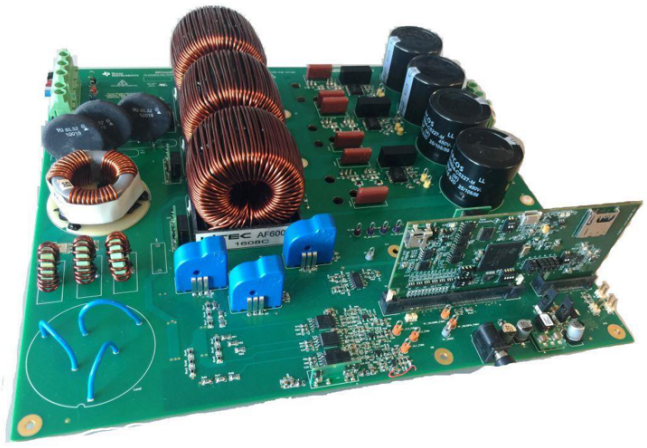
## Features

- Three-Phase Input 208 VL-L 60 Hz, Output 600-V DC Nominal, 1.2 KW
- Three-Phase Input 400 VL-L 50 Hz, Output 700-V DC Nominal, 2.4 KW
- 50-kHz Pulse Width Modulation (PWM) Switching
- Greater Than 98% Peak Efficiency
- Less Than 2% Total Harmonic Distortion (THD) at Full Load and Low Line
- powerSUITE Support for Easy Adaptation of the Design for User Requirement
- Software Frequency Response Analyzer (SFRA) and Compensation Designer for Ease of Tuning of Control Loops

## Applications

- Offboard Chargers for EV
- Telecom Rectifier
- Drives, Welding, and Other Industrial



An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

# 1 System Specifications

## 1.1 System Description

Three-phase power is used by equipment operating at high power in industrial applications. To improve grid power quality and reduce the harmonic currents drawn, power factor correction is needed as many of the forward loads are DC. For example, in an offboard, fast EV charger, operating at 20 KW, the input is a three-phase AC connection from the grid and the output is DC to the battery.

Though many topologies exist for active three-phase power factor conversion, a Vienna rectifier is popular due to its operation in continuous conduction mode (CCM), inherent multilevel switching (three level), and reduced voltage stress on the power devices. Traditionally, hysteresis-based controllers have been used for Vienna rectifiers. Only recently have sine triangle-based PWM been shown to work for Vienna Rectifier control. This control can be quite challenging to design. Several variants of Vienna rectifiers exist, Figure 1 shows the variant of the Vienna rectifier chosen in this design along with the key voltages and currents being sensed.

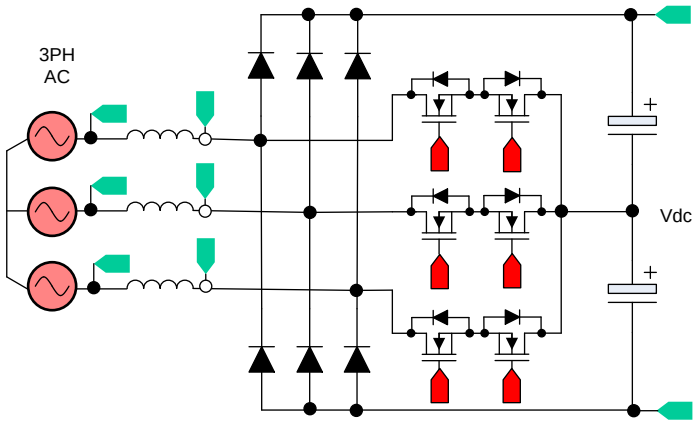



**Figure 1. Vienna Rectifier Variant Implemented**

A Y-connection Vienna rectifier is implemented in this TI Design. With this design the aim is to provide an example of how to control a Vienna rectifier and how to tune the different loops using the C2000 MCU.

## 1.2 Key System Level Specifications

The three-phase vienna rectifier key power specification are given in Table 1.

**Table 1. Key System Specifications**

| PARAMETER | SPECIFICATION |
|---|---|
| Input voltage (Vin) | • AC 208 Vrms VL-L or 120 Vrms L-N , 60 Hz<br>or<br>• AC 400 Vrms VL-L or 230 Vrms L-N , 50 Hz |
| Input current (Iin) | 4 Amps RMS Max |
| Output voltage (Vout) | • 600-V DC bus nominal at 208 Vrms<br>or<br>• 700-V DC bus nominal at 400 Vrms |
| Output current (Iout) | Absolute RMS maximum 5 Amps, pulse maximum 10 Amps |
| Power rating | • 1.2 KW at three-phase 208 Vrms<br>or<br>• 2.4 KW at three-phase 400 Vrms |
| Current THD | • <1% at rated load with 208 Vrms<br>• <4% at rated load with 400 Vrms |
| Efficiency | Peak 98%, average ~97% |
| Primary filter inductor | 3 mH |
| Output capacitance | 180 µF |
| PWM switching frequency | 50 kHz |

**WARNING**

**TI intends this EVM to be operated in a *lab environment only and does not consider it to be a finished product* for general consumer use.**

**TI Intends this EVM to be used only by *qualified engineers and technicians* familiar with risks associated with handling high-voltage electrical and mechanical components, systems, and subsystems.**

**There are *accessible high voltages present on the board*. The board operates at voltages and currents that may cause shock, fire, or injury if not properly handled or applied. Use the equipment with necessary caution and appropriate safeguards to avoid injuring yourself or damaging property.**

**CAUTION**

Do not leave EVM powered when unattended.

TI considers it the user's responsibility to confirm that the voltages and isolation requirements are identified and understood before energizing the board or simulation. ***When energized, do not touch the EVM or components connected to the EVM***.

Some components may reach high temperatures > 55°C when the board is powered on. The user must not touch the board at any point during operation or immediately after operating, as high temperatures may be present.

## 1.3 Block Diagram

Figure 2 shows the block diagram of the Vienna rectifier chosen in this design along with the key voltages and currents being sensed.

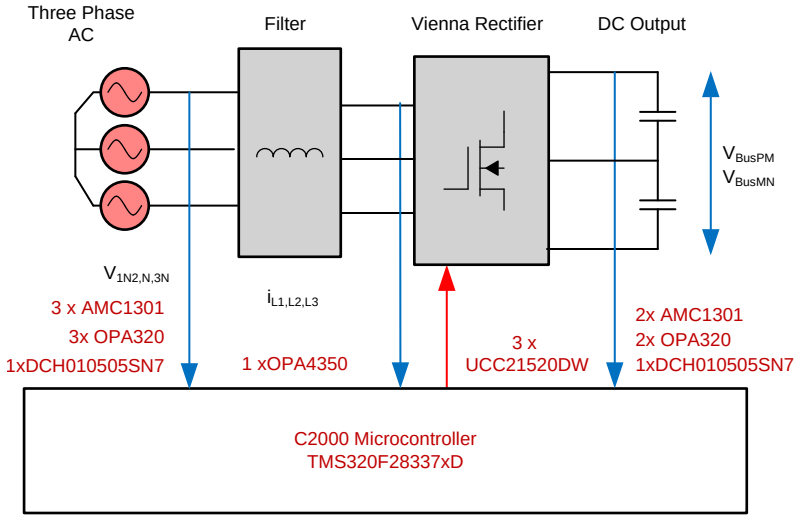



**Figure 2. Block Diagram**

## 1.4 Highlighted Products

### 1.4.1 C2000™ MCU F2837x

The C2000 MCUs are an optimized MCU family for real-time control applications. The fast and high-quality analog-to-digital controller enables accurate measurement of the current and voltage signals, and the integrated comparator subsystem (CMPSS) integrates protection for overcurrent and overvoltage without use of any external devices. The optimized CPU core enables fast execution of control loop. Trigonometric operations are accelerated using the on-chip trigonometric math unit (TMU), which imparts additional speedup in control loop execution.

### 1.4.2 UCC21520

The UCC21520 is an isolated, dual-channel gate driver with a 4-A source and a 6-A sink peak current. The driver is designed to drive power MOSFETs, IGBTs, and SiC MOSFETs up to 5 MHz with a best-in-class propagation delay and pulse-width distortion. The input side is isolated from the two output drivers by a 5.7-kVRMS reinforced isolation barrier, with a minimum of 100-V/ns common-mode transient immunity (CMTI). Internal functional isolation between the two secondary-side drivers allows a working voltage of up to 1500-V DC. A disable pin shuts down both outputs simultaneously when set high and allows normal operation when left floating or grounded. The device accepts VDD supply voltages up to 25 V. A wide input VCCI range from 3 to 18 V makes the driver suitable for interfacing with both analog and digital controllers.

### 1.4.3 AMC1301

The AMC1301 is a precision isolation amplifier with an output separated from the input circuitry by an isolation barrier that is highly resistant to magnetic interference. The input of the AMC1301 is optimized for direct connection to shunt resistors or other low voltage level signal sources.

### 1.4.4 OPA320

The OPA320 is a precision, low-power, single-supply op amp optimized for very-low noise. Operated from a voltage range from 1.8 to 5.5 V, the device is well-suited for driving analog-to-digital converters (ADCs). With a typical offset voltage of 40 µV and very-low drift over temperature (1.5 µV/°C typical), it is very well suited for applications like control loop and current sensing in motor control.

# 2 Control System Design Theory

This section discusses the control system design theory

## 2.1 PWM Modulation

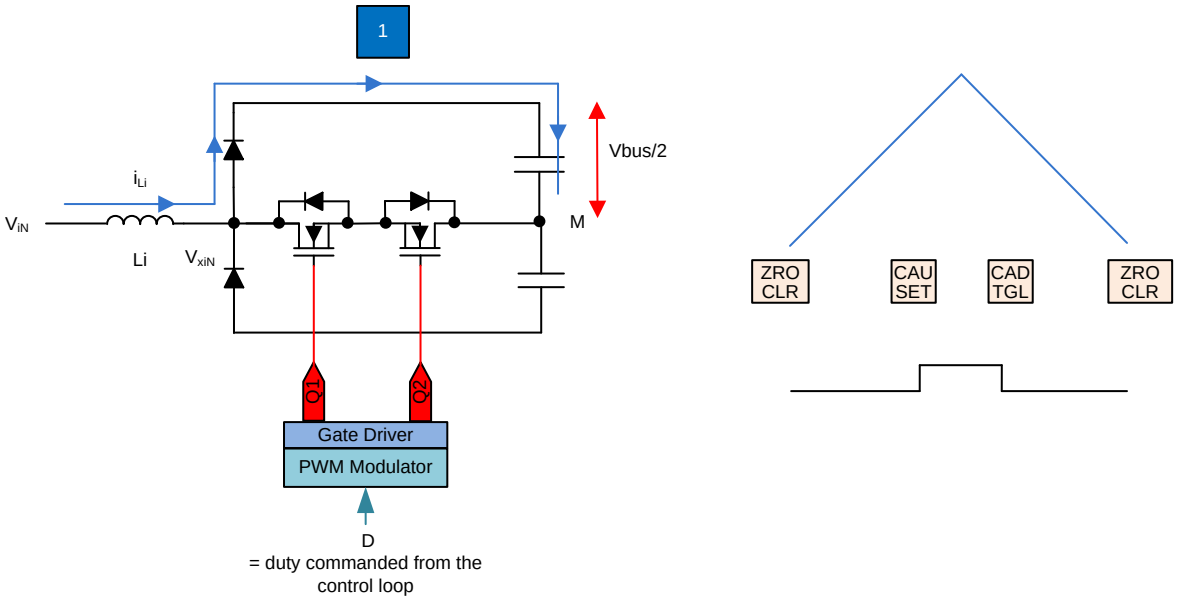



**Figure 3. Single Phase Diagram of Vienna Rectifier**

Figure 3 shows a simplified, single-phase diagram of the Vienna rectifier. To control this rectifier, the duty cycle is controlled such that it regulates the voltage $v_{xiN}$directly. That is, if the software variable *Duty* is set to 1, it makes $v_{xiN}$ the largest voltage possible by never turning on Q1 and Q2 switches and letting the inductor connect to the DC bus though the bridge diode. Similarly when *Duty* is set to 0, PWM is modulated such that Q1 and Q2 always conduct making $v_{xiN}$ connect to the midpoint of the DC bus (which is zero), which makes it the lowest possible voltage for the switching cycle.

Detailed PWM configuration is shown in Figure 4.

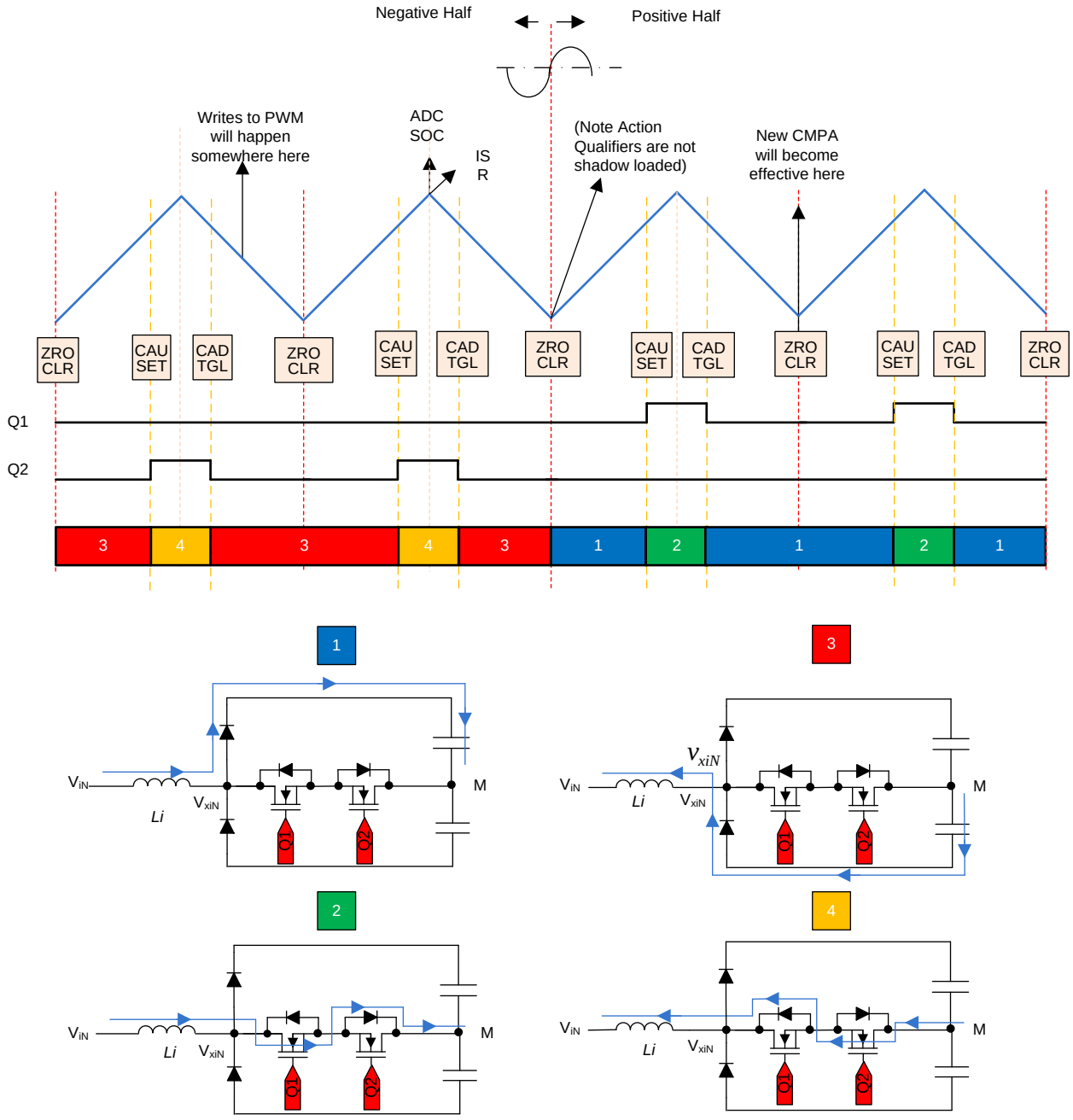



**Figure 4. Vienna Rectifier Detailed PWM Modulation Scheme**

## 2.2 Current Loop Model

To understand the current loop model, first look at the inductor current closely. In Figure 3 the duty cycle D is provided to the PWM modulator, which is connected to the switches Q1 and Q2. With this in mind, see Equation 1:

$$V_{xiN} = D * \frac{V_{bus}}{2} \tag{1}$$

**NOTE:** When D is set to 1, all the switches are *off*, and when D is 0, all switches are *on*, which connect the inductor to the point to M.

To modulate the current through the inductor, the voltage $v_{xiN}$ is regulated using the duty cycle control of Q1 and Q2 switches. Assuming the direction of current is positive in the direction from the AC line into the rectifier and using the DC bus feedforward, the input AC voltage feedforward along with the assumption that the grid is fairly stiff. The current loop can be simplified as shown in Figure 5, and the current loop plant model can be written as in Equation 2.

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

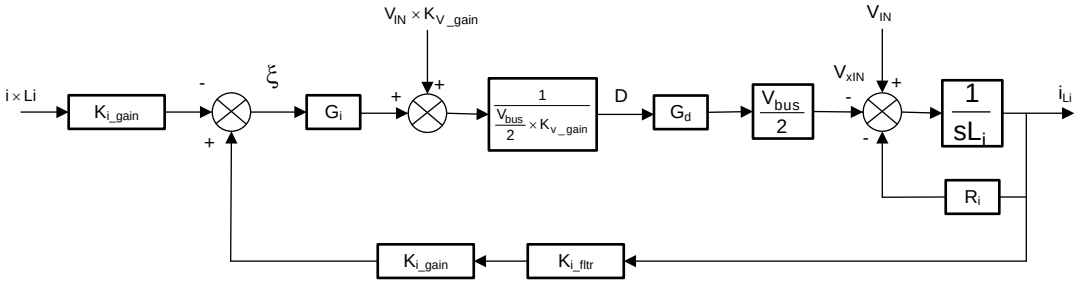



**Figure 5. Current Loop Control Model**

**NOTE:** The negative sign on the reference is in place because the current loop is thought to be regulating the voltage $v_{xiN}$. To increase the current, $v_{xiN}$ must be reduced and, thus, the opposite sign for reference and feedback in Figure 5. This current loop model is used to tune the current compensator. A simple proportional controller is used for the current loop. The gain of the proportional gain is adjusted to ensure the system is stable.

## 2.3 DC Bus Regulation Loop

The DC bus regulation loop is assumed to be providing the power reference. This loop is divided by the square of the line voltages RMS to provide the conductance, which is further multiplied by the line voltage to give the instantaneous current command.

Small signal model of the DC bus regulation loop is developed by linearizing Equation 3 around the operating point.

$$i_{DC} v_{bus} = 3 n v_{Nrms} i_{Nrms} \Rightarrow \hat{i}_{DC} = 3n \frac{\bar{V}_{Nrms}}{\bar{V}_{bus}} i_{Li} \tag{3}$$

For resistive load the bus voltage and current relate, as shown in Equation 4.

$$\hat{V}_{bus} = \frac{R_L}{1 + sR_L C_o} \hat{i}_{DC} \tag{4}$$

The DC voltage regulation loop control model can be drawn, as shown in Figure 6. An additional Vbus feedforward is applied to make the control loop independent of the bus voltage, and, thus, the plant model for the bus control can be written as shown in Equation 5.

$$H_{p\_bus} = H_{load} * N * K_{i\_gain} * K_{v\_gain} * K_{v\_flt} \quad (5)$$

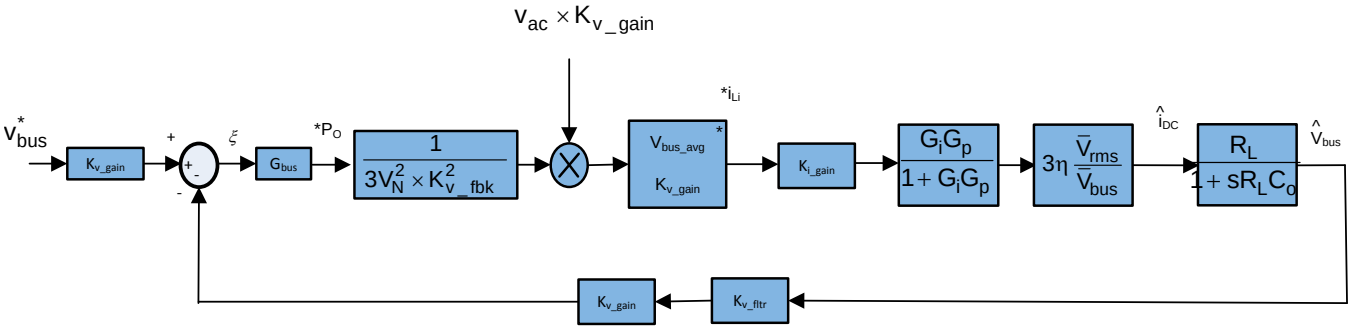



**Figure 6. DC Voltage Loop Control Model**

Using Figure 6, a proportional integrator (PI) compensator is designed for the voltage loop. The bandwidth of this loop is kept low as it is in conflict with the THD under steady state.

Additionally, a non-linear PI loop is used to reduce the transients in case of step load changes. Figure 7 shows the structure of the non-linear PI loop implemented on this design

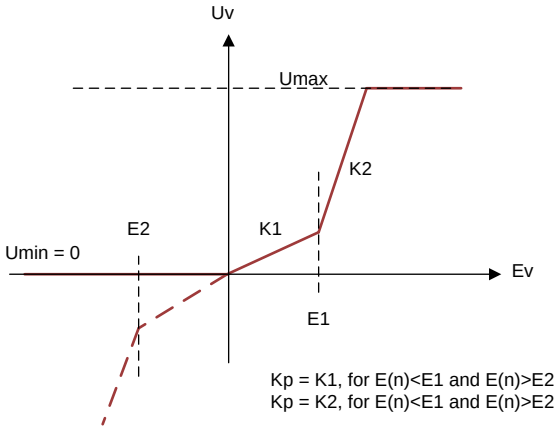



**Figure 7. Non-Linear PI Loop for Voltage Controller**

## 2.4 DC Voltage Balance Controller

A split capacitor is used for the output voltage bus in the Vienna rectifier. The voltages across these capacitors may not naturally stay balanced hence a DC balance controller loop is added. This loop modulates an offset, which is added to the duty cycle, thus, modulating the current through the midpoint to balance the voltages on the split capacitor.

A simple proportional gain is used for the DC bus balance controller, with the output of the balance loop given as in Equation 6.

$$G_{s\_out} = (V_{bus\_PM} - V_{bus\_MN}) * G_{s\_gain\_Kp} \quad (6)$$

# 3 Hardware Design Theory

Below are details on selection of the inductor, output capacitor, and the sensing schemes used in the design. Refer to the excel sheet in the SW install package for more details on each one.

controlSUITE\development_kits\tidm_1000\<version>\hardware\<revision>\

calculations.xlsx

## 3.1 Inductor Design

Input inductor (Li) filters out the switching frequency harmonics. Inductor design, amongst other factors, depends on calculation of the current ripple and choosing a material for the core that can tolerate the calculated current ripple. Figure 8 shows one switching cycle waveform of the inverter output voltage vi with respect to the inductor current.

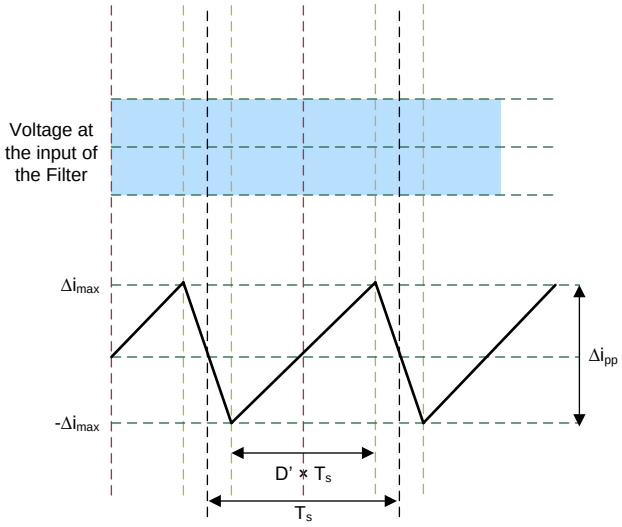



**Figure 8. Current Ripple Calculation**

The voltage across the inductor is given by $V=L_i(di/dt)$. For the Vienna rectifier this equation can be written as Equation 7.

$$\left(\frac{V_{bus}}{2} - V_{in}\right) = L_i * \frac{\Delta i_{pp}}{D' * T_s} \tag{7}$$

where $T_s=1/F_{sw}$is the switching period and D' is the duty cycle for which the switches are ON. For control design, D is assumed to be the voltage at the other terminal of the inductor and is related to D' by D'=1-D. Rearranging the current ripple at any instant in the AC waveform is given as Equation 8.

$$\Delta i_{pp} = \frac{D' * T_s * \left(\frac{V_{bus}}{2} - V_{iN}\right)}{L_i} \tag{8}$$

Now assuming modulation index to be $m_a$the duty cycle can be given as $D'=m_a*\sin(wt)$ and assuming that $v_{iN}=D'*(V_{bus}/2)$, Equation 9 can be derived.

$$\Delta i_{pp} = \frac{\frac{V_{bus}}{2} * T_S * m_a * \sin(\omega t) * (1 - m_a \sin(\omega t))}{L_i} \tag{9}$$

From Equation 9, it is clear that the peak ripple is a factor where the input AC is in the sinusoidal waveform.

To get the maximum value differentiating the equation with respect to time, use Equation 10.

$$\frac{d(\Delta i_{pp})}{dt} = K\left\{\cos(\omega t)(1 - m_a \sin(\omega t)) - m_a \sin(\omega t) * \cos(\omega t)\right\} = 0 \tag{10}$$

Which gives the maximum ripple exists at sin(wt)=1/(2*$m_a$), Substituting this value, Equation 11 is derived.

$$\Delta i_{pp_{max}} = \frac{\frac{V_{bus}}{2} * T_s}{4 * L_i} \Rightarrow L_i = \frac{\frac{V_{bus}}{2}}{4 * F_{sw} * \Delta i_{pp_{max}}} \tag{11}$$

With these values in mind, an appropriate core can be selected along with an inductor designed to meet this inductance value.

### 3.2 Bus Capacitor Selection

The bus capacitor is responsible for removing the ripple on the DC voltage that can be caused by the draw of sinusoidal currents. The capacitor value and the DC bus ripple are related by Equation 12.

$$C = \left(\frac{1}{3}\right) \frac{P_{ac}}{4 * f * \left(V^2 - (V - \Delta V)^2\right)} \tag{12}$$

This equation is used to select the minimum DC bus capacitance value.

### 3.3 Input AC Voltage Sensing

First a virtual neutral is constructed by using a resistor network connected in Y along with some capacitance for stability. In this design the controller is kept on the cold side, thus, an isolated amplifier AMC1301 is used to process the VL-N' voltages as shown in Figure 9. As the AMC1301 is designed considering low-impedance sources for current sensing application, the input differential resistance plays a non-linear role in the total gain calculation. Therefore, a final calibration during build level one must be done, and the maximum AC voltage range must be adjusted according to the calibration. Alternatively, one can use TINA-TI™ software for simulation.

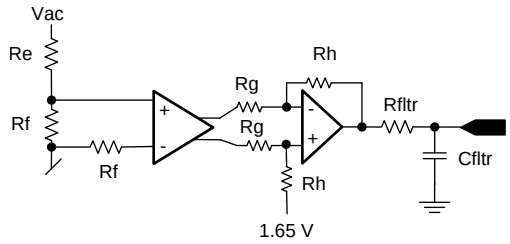



**Figure 9. Input AC Voltage Sensing**

### 3.4 Bus Voltage Sensing

Similarly, the bus voltage, which is split between two capacitors, is sensed using AMC1301 and OPA320 as shown in Figure 10.

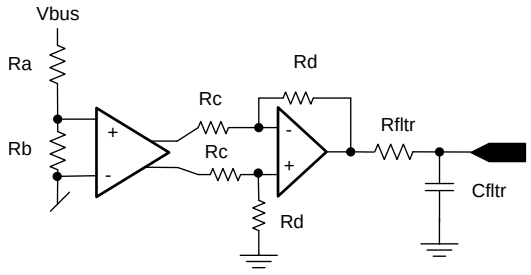



**Figure 10. Bus Voltage Sensing**

As accuracy is of importance in the bus voltage sensing, a further step of calibration with offset and gain adjustment is carried out by doing a regression analysis. A sample of how this is done is shown in the excel sheet in the install package called:

controlSUITE\development_kits\tidm_1000\<version>\hardware\<revision>\BusMeasCalibrationRoutine.xlsx

- sheet-> calData (this is the raw data that is used to run regression)
- sheet->PM_Reg
- sheet->MN_Reg (these are the results of running the regression that provide the intercept and gain values for the adjustment of the measurement)

### 3.5 Inductor Current Sensing

A hall effect sensor is used to sense the current through the inductor. The hall effect sensor has an inbuilt offset, and the range is different than what ADC can measure. Therefore, the voltage is scaled to match the ADC range using the circuit shown in Figure 11.

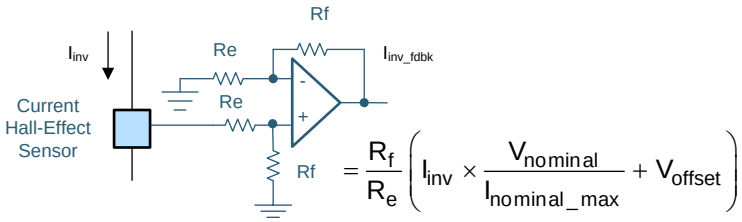



**Figure 11. Current Sense Using Hall Effect Sensor**

### 3.6 Sense Filter

An RC filter is used to filter the signals before connecting to the inverter. A common RC filter is used for all the sensing signals on this design, as shown in Figure 12.

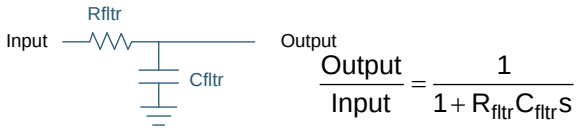



**Figure 12. RC Filter**

## 3.7 Protection (CMPSS)

Most power electronics converters need protection from an overcurrent event. For this design multiple comparators are required, and references for the trip must be generated, as shown in Figure 13.

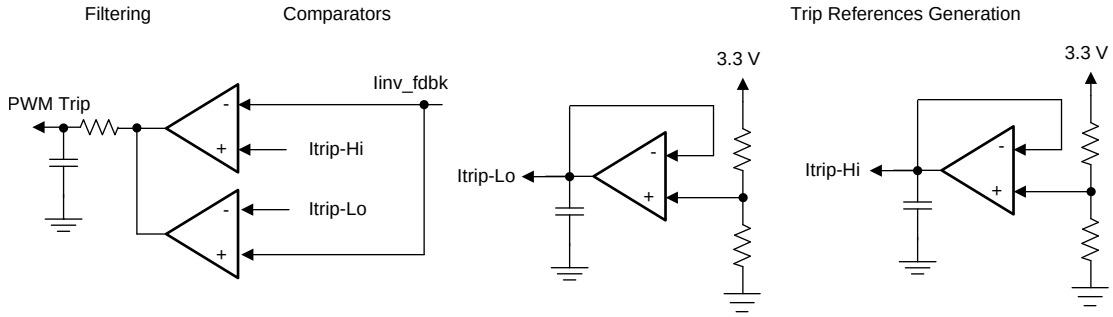



**Figure 13. Trip Generation for PWM Using Comparators and Reference Generators**

All this circuitry is avoided when using the C2000 MCUs such as TMS320F28377Ds, which have on-chip windowed comparator as part of the CMPSS that are internally connected to the PWM module and can enable fast tripping of the PWM. This device saves board space and is cost efficient in the end application as extra components can be avoided using on-chip resources, as shown in Figure 14.

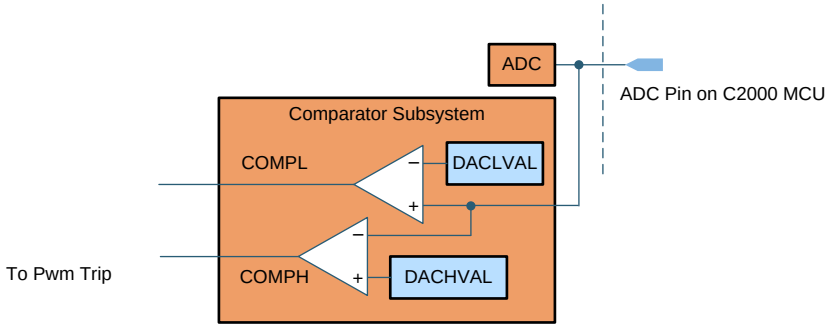



**Figure 14. Comparator Subsystem (CMPSS) Used for Overcurrent Protection**

## 3.8 Efficiency Estimates

An estimate of the expected efficiency is shown in

controlSUITE\development_kits\tidm_1000\<version>\hardware\<revision>\calculations.xlsx

- sheet→ Efficiency Estimate

# 4 Getting Started Hardware

This section details the hardware and explains the different sections on the board. If using the firmware of the design alone through powerSUITE, this section may not be valid.

## 4.1 Base Board Settings

The design follows a HSEC control card concept. Any device for which a HSEC control card is available from the C2000 MCU product family can be potentially used on this design. The key resources used for controlling the power stage on the MCU are listed in Table 2. Figure 15 shows the key power stage and connectors on the TI Design, and Table 3 lists the key connectors and their functions. To get started:

1. Make sure no power source is connected to the design.
2. Insert the control card in the J8-J10 slot.
3. Insert a jumper at J11 and J12 to connect the bias supply for 5 V and 3.3 V .
4. Connect a 12-V DC, 1-Amps power supply at J4. A few LEDs on the base board will light up indicating power. The LED on the control card will also light up. This light indicates the device is powered up as well.
   - Note: The bias for the MCU is separated from the power stage, which enables safe bring up of the system.
5. To connect JTAG, use a USB cable from the control card and connect it into a host computer.
6. A three-phase power supply is connected to the input J2. The board works as a three-wire system, therefore, the connection of the neutral is not required. A virtual neutral is generated on the board for sensing purpose of VL-N voltages.
7. A resistive load of ~700-800Ω should also be connected to the output at J1. The load is connected across the positive and negative terminal (Vbus). The midpoint (M) is not connected to the load.
8. Current and voltage probes can be connected to observe the input current, input voltage, and output voltages, as shown in Figure 16.

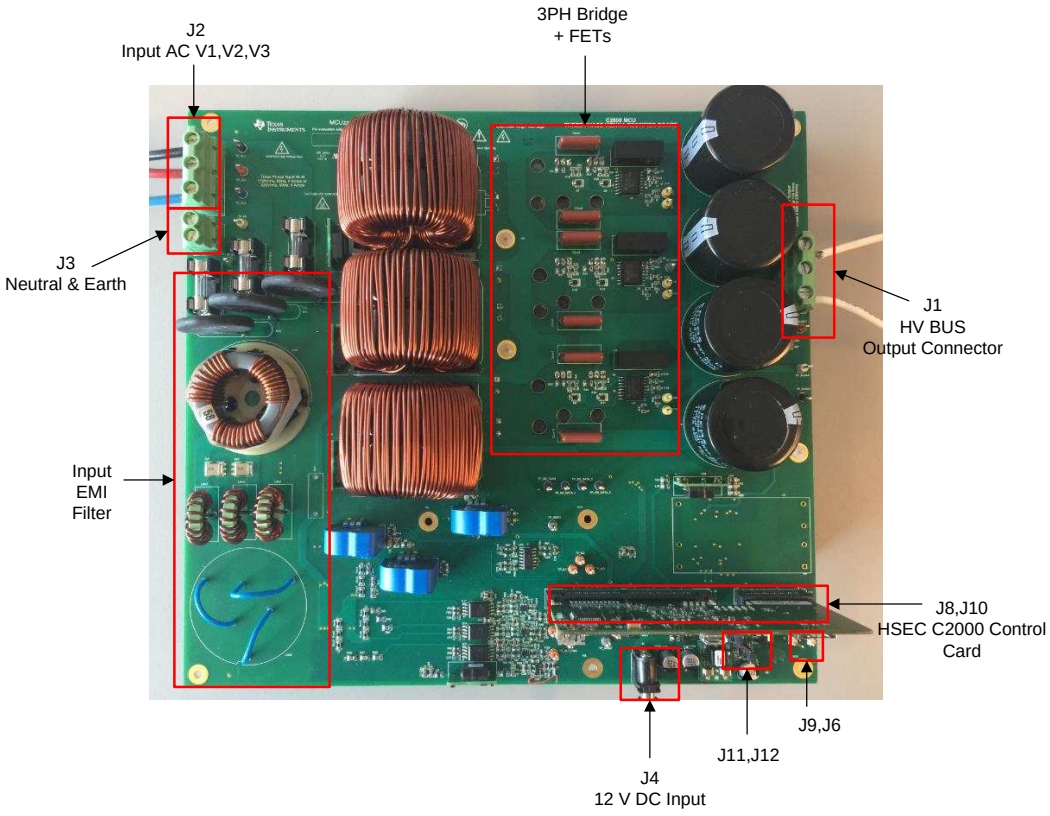



**Figure 15. Board Overview**

**Table 2. Key Controller Peripherals Used for Control of Power Stage on Board**

| SIGNAL NAME | HSEC PIN NUMBER | FUNCTION |
|---|---|---|
| PWM-1A | 49 | PWM : Vienna rectifier PWM Ph1A |
| PWM-1B | 51 | PWM : Vienna rectifier PWM Ph1B |
| PWM-2A | 53 | PWM : Vienna rectifier PWM Ph2A |
| PWM-2B | 55 | PWM : Vienna rectifier PWM Ph2B |
| PWM-3A | 50 | PWM : Vienna rectifier PWM Ph3A |
| PWM-3B | 52 | PWM : Vienna rectifier PWM Ph3B |
| IL1 | 15,20 | ADC w CMPSS : Inductor current measurement Ph1 |
| IL2 | 21.27 | ADC w CMPSS : Inductor current measurement Ph2 |
| IL3 | 25 | ADC w CMPSS : Inductor current measurement Ph3 |
| V1 | 18 | ADC : AC voltage sensing Ph1 |
| V2 | 28 | ADC : AC voltage sensing Ph2 |
| V3 | 34 | ADC : AC voltage sensing Ph3 |
| Vbus_PM | 31.24 | ADC : Bus voltage positive terminal to midpoint |
| Vbus_MN | 37,26 | ADC : Bus voltage midpoint to negative terminal |
| Profiling GPIO1 | 58 | GPIO : Used to profile code (optional) |
| Profiling GPIO2 | 60 | GPIO : Used to profile code (optional) |

**Table 3. Key Connectors and Function**

| CONNECTOR NAME | FUNCTION |
|---|---|
| J2 | Output high voltage bus |
| J2 | Input three phase voltage |
| J3 | Neutral and Earth connection |
| J4 | Input bias supply, 12-V DC 1 Amps |
| J5 | DAC output for debugging |
| J6 | Ground |
| J7 | Profiling GPIO connector |
| J8,J10 | HSEC control card connector slot |
| J9 | Remote enable connector |
| J11 | 5-V jumper, used to disconnect the 5-V supply |
| J12 | 3-V jumper, used to disconnect the 12-V jumper |

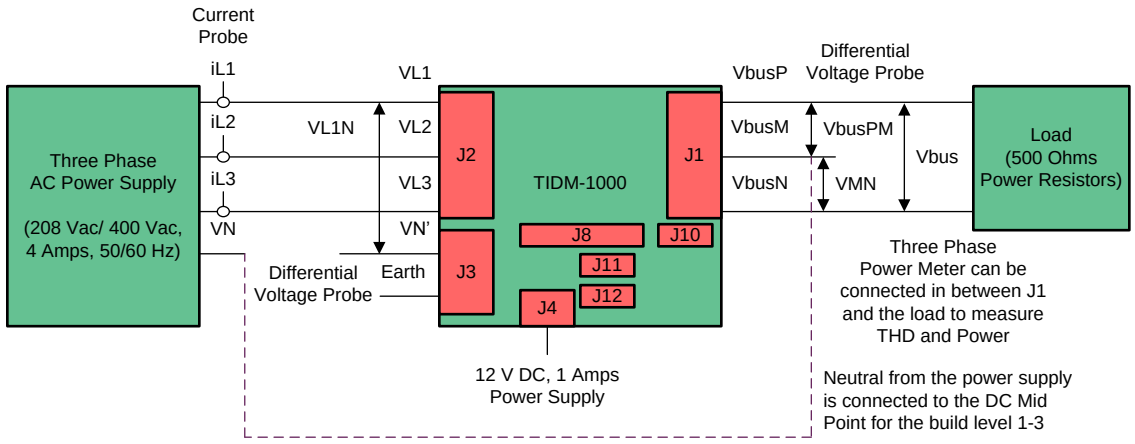



**Figure 16. Hardware Setup to Run Software**

## 4.2 Control Card Settings

Certain setting on the device control card are needed to communicate over JTAG and use the isolated UART port. The settings also provide a correct ADC reference voltage. The following are the settings required on revision 1.1 of the F28377D control card. Refer to the information sheet located inside controlSUITE at

C:\ti\controlSUITE\development_kits\~controlCARDs\TMDSCNCD28377D_v1_3:

1. A:SW1 on the control card must be set on both ends to *ON (up)* position to enable JTAG connection to the device and the UART connection for SFRA GUI. If this switch is *OFF (down)*, one cannot use the isolated JTAG built in on the control card nor can SFRA GUI communicate to the device.
2. A:J1 is the connector for the USB cable that is used to communicate to the device from a host PC on which Code Composer Studio™ (CCS) runs.
3. A 3.3-V reference is desired for the control loop tuning on this design; therefore, set the appropriate jumpers to provide a 3.3-V reference externally to the on-chip ADC. For version 1.3 of the F283779D control card, this means SW3 and SW2 are moved to the end with "." that is, to the left, which puts 3.3-V VDDA as the reference for the ADC. Refer to the information sheet for more information

# 5 Getting Started Firmware

The software of this design is available inside controlSUITE and is supported inside the powerSUITE framework.

## 5.1 Opening the Project Inside Code Composer Studio™

To start:

1. Install CCS from the *Code Composer Studio (CCS) Integrated Development Environment (IDE)* tools folder.
2. Open CCS. Go to *View* → *CCS App Center*. Under *Code Composer Studio Add-ons*, make sure GUI Composer in installed. If the GUI is not installed, install GUI Composer.
3. Install controlSUITE™ at the *controlSUITE Software Suite: Essential Software and Development Tools for C2000 Microcontrollers* tools folder.
   - Note: powerSUITE is installed with controlSUITE in the default install.
4. Close CCS, and open a new workspace. CCS will automatically detect powerSUITE. A restart of CCS may be required for the change to be effective.
5. Go to *View* → *Resource Explorer*. Under the TI Resource Explorer, go to *controlSUITE* → *powerSUITE*.

Open the TI Design software as it is (opens firmware as it was run on this design and hardware, requires the board to be exactly the same as this TI Design, and does not allow modification through the powerSUITE GUI inside the project).

1. Under *powerSUITE*, select *Development Kits* → *Three Phase PFC Vienna Rectifier TIDM-1000*, and click on *Run <device> Project*.
2. This will import the project and the development kit or designs page will show up. This page can be used to browse all the information on the design including this user guide, test reports, hardware design files, and so forth.
3. Click *Run <device_name> Project*.
4. This will import the project into the workspace environment, and a cfg page with a GUI similar to Figure 17 will show up.
   - Note: As this project is imported from the development kit and TI Design page, modifications to the power stage parameters through the GUI will not be allowed.
5. If this GUI page does not appear, refer to the FAQ section under *powerSUITE* in the *controlSUITE* resource explorer.

Open TI Design software for adaptation. The user can modify power stage parameters, which are then used to create the model of the power stage in Compensation Designer and can also modify scaling values for voltages and currents.

1. Click on the *Solution Adapter Tool* ( ).
2. Select *Three Phase PFC* from the list of solutions presented.
3. Select the device this solution needs to run on the next page.
4. Once the icon is clicked, a pop-up window will show up asking for a location to create the project. One can also save the project inside the workspace itself. Once the location is specified, a project is created, and a GUI page will appear with modifiable options for the solution (Figure 17).
5. This GUI can be used to change the parameters for an adapted solution, like power rating, inductance, capacitance, sensing circuit parameters, and so forth.
6. If this GUI page does not appear, refer to the FAQ section under *powerSUITE* in the *controlSUITE* resource explorer.

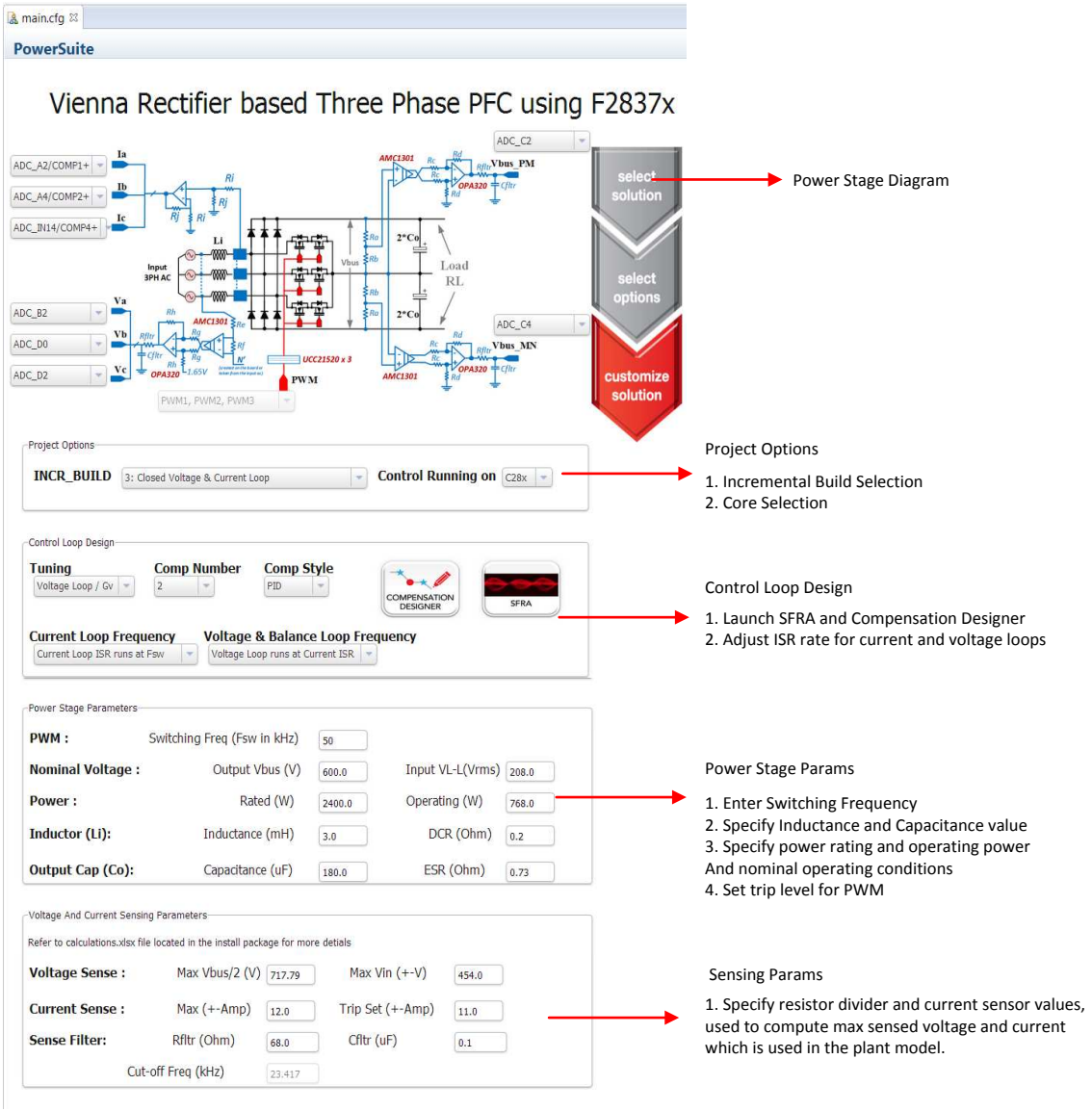



**Figure 17. powerSUITE Page for Vienna Rectifier Solution**

## 5.2 Project Structure

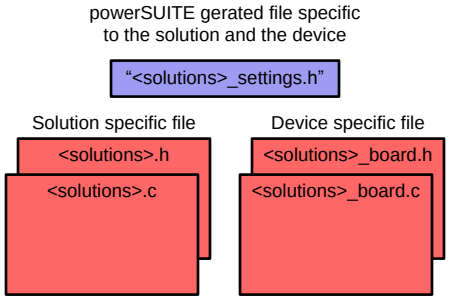



**Figure 18. Project Structure Overview**

The general structure of the project is shown in Figure 18.

Note: The figure shows the project for F2837x; however, if a different device is chosen from the powerSUITE page, the structure will be similar.

Solution specific and device independent files are <*solution*>*.c/h*. This file consist of the main.c file of the project and is responsible for the control structure of the solution.

For this design <solution> is *pfc3phvienna*.

Board-specific and device-specific files are <*solution*>*_board.c/h*. This file consists of device specific drivers to run the solution.

The powerSUITE page can be opened by clicking on the *main.cfg* file, listed under the project explorer. The powerSUITE page generates the <*solution*>*_settings.h* file. This file is the only file used in the compile of the project that is generated by the powerSUITE page. The user must not modify this file manually, as the changes will be overwritten by powerSUITE every time the project is saved.

The *Kit.xml* and *solution.js* files are used internally by the powerSUITE and must also not be modified by the user. Any changes to these files will result in project not functioning properly.

Once the project is imported, the project explorer will appear inside CCS as shown in Figure 19.

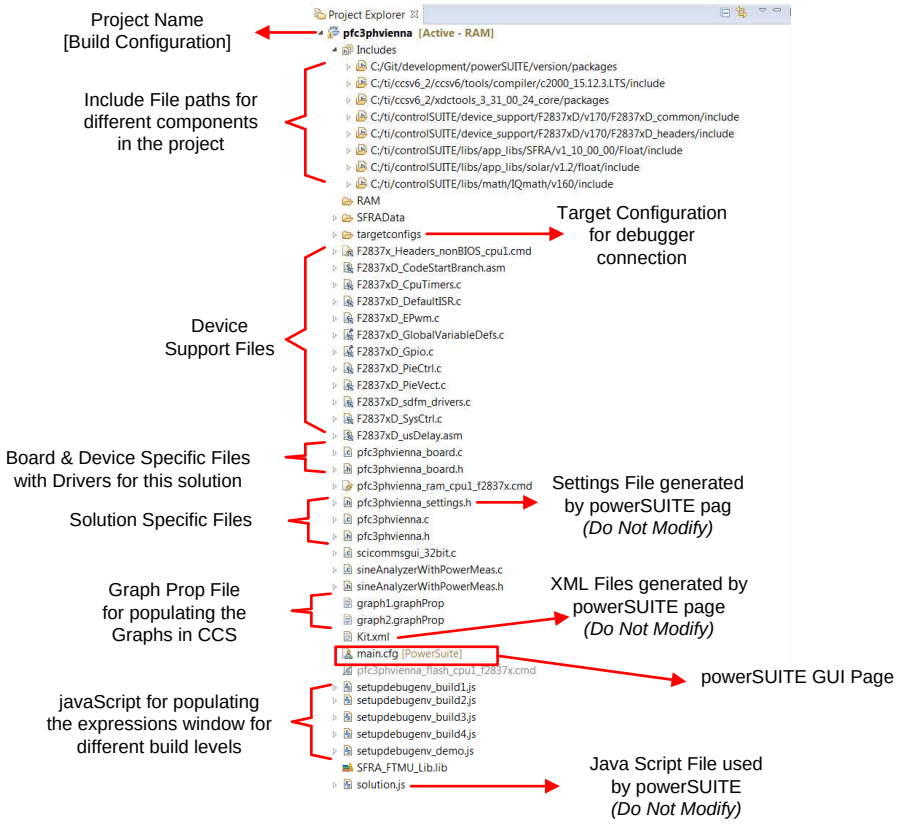



**Figure 19. Project Explorer View of Solution Project**

The project consists of an interrupt service routine, which is called every PWM cycle, called controlISR(), where the control algorithm is executed. In addition to this, there are background tasks A0-A4 and B0-B4, which are called in a polling fashion and can be used to run slow tasks for which absolute timing accuracy is not required. A slower 10-kHz routine *tenkHzISR()* is called for instrumentation and to run some slower tasks that require timing accuracy.

The software of this TI Design is organized in four incremental builds (INCR_BUILD). The incremental build process simplifies the system bring up and design.

- INCR_BUILD 1: Open Loop Check
- INCR_BUILD 2: Closed Current Loop
- INCR_BUILD 3: Closed Voltage and Current Loop
- INCR_BUILD 4: Closed Voltage, Current, and Bus Cap Balance Loop

These build levels are detailed in the section below. If using the TI Design hardware, make sure the hardware setup is completed as outlined in Section 4.

## 5.3 Running the Project

### 5.3.1 INCR_BUILD 1: Open Loop

In this build the board is run in an open-loop fashion with a fixed-duty cycle. The duty cycle is controlled with dutyPU_DC variable. This build verifies the sensing of feedback values from the power stage and also the operation of the PWM gate driver, which ensures there are no hardware issues. Additionally, calibration of input and output voltage sensing can be performed in this build. The software structure for this build is shown in Figure 20. Blocks that run in the slower ISR are marked. Other blocks are run in the fast controlISR.

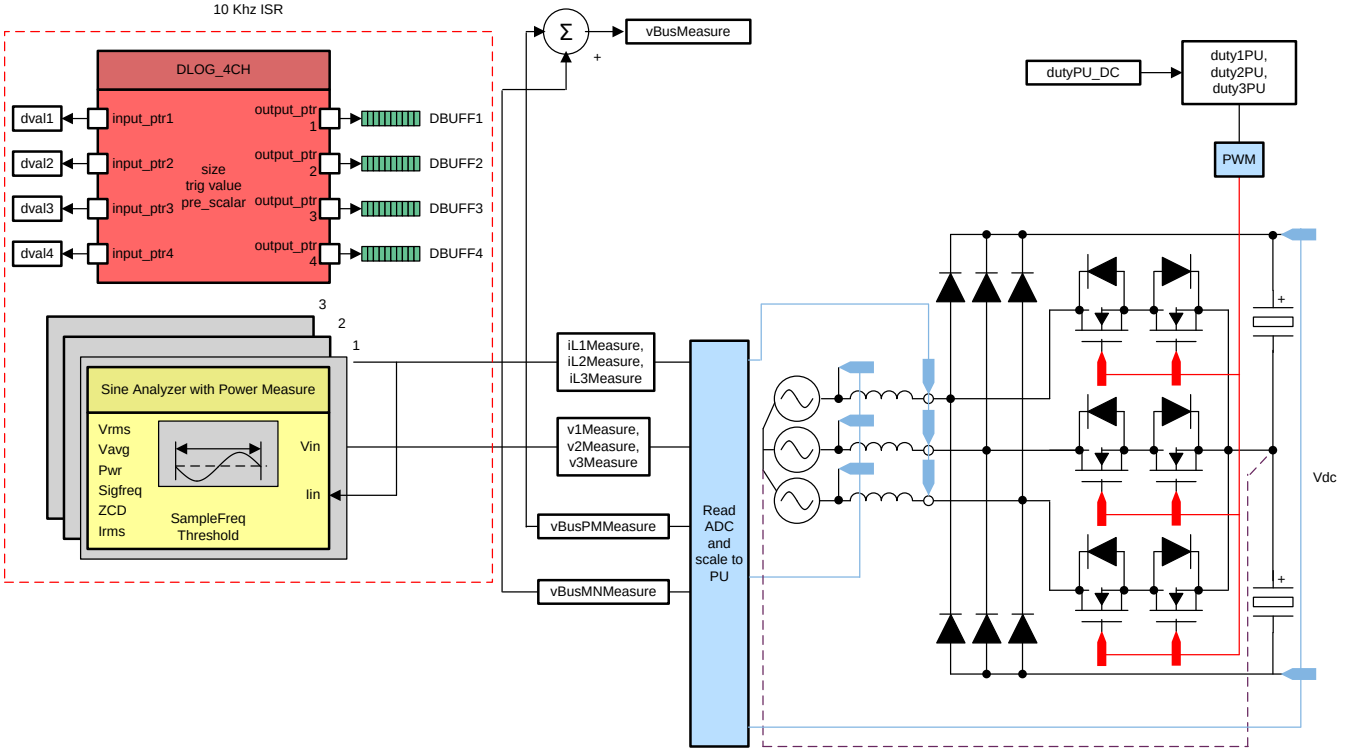





**Figure 20. Build Level 1 Control Software Diagram: Open Loop Project**

#### 5.3.1.1 Setting Software Options for BUILD 1

1. Make sure the hardware is setup as outlined in Section 4.1. Do not supply any high voltage (HV) power yet to the board. For this build, connect the neutral from the three-phase power supply to the board (Figure 16).
2. powerSUITE Settings : On the powerSUITE page select under *Project Option*s section:
   - *Open Loop* for the build level.

   If this is an adapted solution, edit the setting under *ADC Sensing Parameters* with the sensing resistors used in each case. Specify the switching frequency, the dead band, and the power rating. Save the page.

#### 5.3.1.2 Building and Loading the Project

1. Right click on the project name, and click *Rebuild Project*.
2. The project will build successfully.
3. In the *Project Explorer* make sure the correct target configuration file is set as Active under trageteonfigs (Figure 19).
4. Then click *Run* → *Debug*. This will launch a debugging session. In the case of dual CPU devices, a window may appear to select the CPU the debug needs to be performed. In this case, select CPU1.
5. The project will then load on the device, and CCS debug view will become active. The code will halt at the start of the main routine.

#### 5.3.1.3 Setup Debug Environment Windows

1. To add the variables in the watch and expressions window, click *View* → *Scripting Console* to open the scripting console dialog box. On the upper right corner of this console, click on open then browse to the *setupdebugenv_build1.js* script file located inside the project folder. This script file will populate the watch window with appropriate variables needed to debug the system. Click on the Continuous Refresh button on the watch window to enable continuous update of values from the controller. The watch window will appear as shown in Figure 21.

Variables | Expressions | Registers | Breakpoints

| Expression | Type | Value | Address |
|---|---|---|---|
| BuildInfo | enum enum_BuildLevel | BuildLevel1_OpenLoop | 0x0000B010@Data |
| boardStatus | enum enum_boardStatus | boardStatus_Idle | 0x0000B00E@Data |
| clearTrip | int | 0 | 0x0000B001@Data |
| dutyPU_DC | float | 0.0 | 0x0000B062@Data |
| EPwm1Regs.TZFLG.all | unsigned int | 13 | 0x00004093@Data |
| EPwm2Regs.TZFLG.all | unsigned int | 4 | 0x00004193@Data |
| EPwm3Regs.TZFLG.all | unsigned int | 4 | 0x00004293@Data |
| guiVbus | float | 1.79690909 | 0x0000B0BE@Data |
| guiVbusPM | float | 1.94279742 | 0x0000B0B8@Data |
| guiVbusMN | float | -0.146310508 | 0x0000B0BA@Data |
| guiVrms1 | float | 0.0 | 0x0000B01A@Data |
| guiVrms2 | float | 0.0 | 0x0000B026@Data |
| guiVrms3 | float | 0.0 | 0x0000B03E@Data |
| guiIrms1 | float | 0.0 | 0x0000B0DA@Data |
| guiIrms2 | float | 0.0 | 0x0000B018@Data |
| guiIrms3 | float | 0.0 | 0x0000B01C@Data |
| guiPF1 | float | 0.0 | 0x0000B042@Data |
| guiPF2 | float | 0.0 | 0x0000B036@Data |
| guiPF3 | float | 0.0 | 0x0000B032@Data |
| *Add new expression* | | | |

**Figure 21. Build Level 1 Expressions View**

2. The current and voltage measurements can be verified by viewing the data in the graph window. These values are logged in the slower 10-kHz routine. Go to *Tools* → *Graph* → *DualTime*, and click on *Import* and point to the *graph1.GraphProp* file inside the project folder. This file will populate the graph properties window. Alternatively, the user can enter the values as shown in the Figure 22. Once the entries are verified, click *OK*. Two graphs will appear in CCS. Click on *Continuous Refresh* on these graphs. A second set of graphs can also be added by importing the *graph2.GraphProp* file.

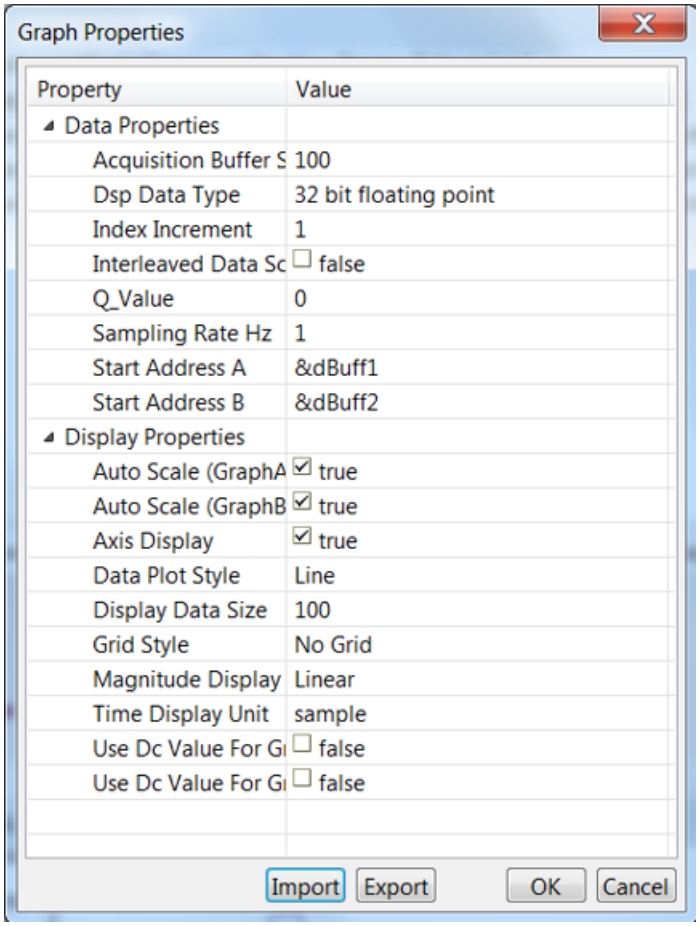
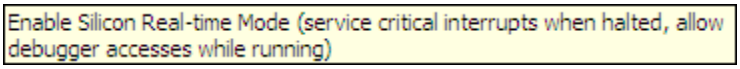



**Figure 22. Graph Settings**

### 5.3.1.4 Using Real-Time Emulation

Real-time emulation is a special emulation feature that allows windows within CCS to be updated while the MCU is running. This feature allows graphs and watch views to update but also allows the user to change values in watch or memory windows and see the effect of these changes in the system without halting the processor.

1. Enable real-time mode by hovering the mouse on the buttons on the horizontal toolbar and clicking the  button.




2. A message box may appear. If so, select *YES* to enable debug events. This will set bit 1 (DGBM bit) of status register 1 (ST1) to a *0*. The DGBM is the debug enable mask bit. When the DGBM bit is set to *0*, memory and register values can be passed to the host processor for updating the debugger windows.

#### 5.3.1.5 Running the Code

1. Run the project by clicking on  .
2. In the watch view, periodically check if the guiVbus variable is updating. If there is no change in the value then make sure the real-time mode is enabled, and the HW is setup correctly. Do not proceed further unless the update is verified.
   - Note: As no power is applied right now, this value will be close to zero.
3. Now slowly increase the input AC voltage from 0 to 120-Vrms L-N.
4. Verifying the voltage sensing: Make sure *guiVbus*, *guiVbusPM*, and *guiVbusMN* display the correct values. For the 120-Vrms L-N guiVbus will be close to 320 V, and the guiVbusPM/MN will be close to 160 V each. The code runs a sine analyzer module, which computes the RMS value of the voltage and current. Notice the value of *guiVrms1/2/3* will be close to the input value, that is, 120Vrms. This verifies the voltage sensing of the board.
5. Verifying the current sensing: Notice the *guiIrms1/2/3*, for the given test condition this value will be close to 0.92 Amps. Additionally, the graphs must be seen to verify the current measurement. Under the test condition they will look as shown in Figure 23.


**Figure 23. Build Level 1: Graph1.GraphProp and Graph2.GrpahProp Files Showing Measured Voltage and Currents**

6. The scope capture of the input voltage and currents is shown in Figure 24.

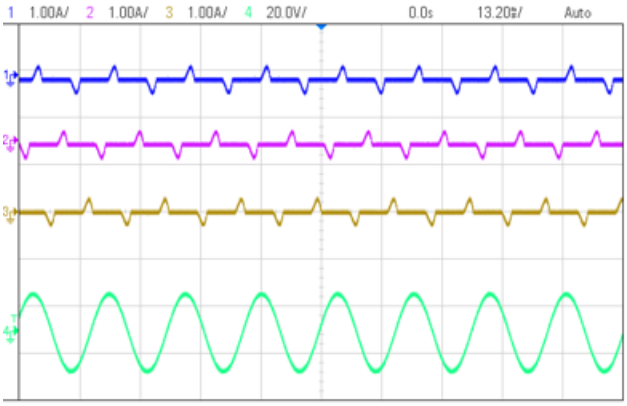



**Figure 24. Build Level 1: Scope Capture IL1, IL2, IL3 and V1 (120Vrms L-N) With PWM Tripped**

7. Next to verify the PWM action, first reduce the input voltage to zero and wait for all the voltages to go down to zero.
8. Now set the *dutyPU_DC* to 0.5 in the expressions view.
9. Next clear the PWM trip by writing a *1* to *clearTrip*.
10. Slowly increase the input voltage, and keep watch on the input current. The duty cycle will impart a boost action. For example, when Vac is 30 Vrms without switching enabled, the guiVbus will be ~84 V; with switching, the guiVbus will rise up to 140 V. Thus, *guiVbusPM/MN* will also be higher than the input voltage maximum.
11. Under the test conditions described in this build, the *guiVbus* will rise to about 600 V and *guiVbusPM/MN* will be close to 300 V each, and the current will be close to 2.8 Vrms when the input voltage reached 120 Vrms L-N. Expressions view will appear as shown in Figure 25. Make sure all the variables are accurate, that is, guiVrms1/2/3, guiIrms1/2/3, guiPF1/2/3. If any variable is not in line with as shown in Figure 25, it points to a hardware issue with the sensing circuit. The scope capture is shown in Figure 26.

Variables | Expressions | Registers | Breakpoints

| Expression | Type | Value | Address |
|---|---|---|---|
| BuildInfo | enum enum_BuildLevel | BuildLevel1_OpenLoop | 0x0000B010@Data |
| boardStatus | enum enum_boardStatus | boardStatus_NoFault | 0x0000B00E@Data |
| clearTrip | int | 0 | 0x0000B001@Data |
| dutyPU_DC | float | 0.5 | 0x0000B062@Data |
| EPwm1Regs.TZFLG.all | unsigned int | 9 | 0x00004093@Data |
| EPwm2Regs.TZFLG.all | unsigned int | 9 | 0x00004193@Data |
| EPwm3Regs.TZFLG.all | unsigned int | 9 | 0x00004293@Data |
| guiVbus | float | 618.878784 | 0x0000B0BE@Data |
| guiVbusPM | float | 309.595947 | 0x0000B0B8@Data |
| guiVbusMN | float | 309.273407 | 0x0000B0BA@Data |
| guiVrms1 | float | 118.676414 | 0x0000B01A@Data |
| guiVrms2 | float | 120.261597 | 0x0000B026@Data |
| guiVrms3 | float | 121.476097 | 0x0000B03E@Data |
| guiIrms1 | float | 2.81716728 | 0x0000B0DA@Data |
| guiIrms2 | float | 2.80645514 | 0x0000B018@Data |
| guiIrms3 | float | 2.82467127 | 0x0000B01C@Data |
| guiPF1 | float | 0.747663856 | 0x0000B042@Data |
| guiPF2 | float | 0.75015074 | 0x0000B036@Data |
| guiPF3 | float | 0.746766269 | 0x0000B032@Data |
| Add new expression | | | |

**Figure 25. Build Level 1: Expressions View With Power Measurement**

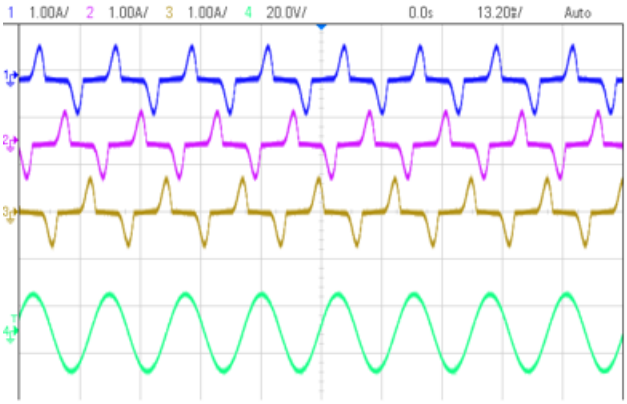



**Figure 26. Build Level 1: Scope Capture IL1, IL2, IL3 and V1 (120-Vrms L-N) With Duty Cycle 0.5**

12. This check verifies at a basic level the PWM driver and connection on hardware.
13. Reduce the input voltage to zero, and watch for the bus voltages to reduce down to zero.
14. This completes the check for this build, the following items are verified on successful completion of this build:
    (a) Sensing of voltages and currents and scaling to be correct
    (b) Interrupt generation and execution of the build 1 code in the controlISR and tenkHzISR()
    (c) PWM driver and switching
    If any issue is observed a careful inspection of the hardware may be required to eliminate any build issues and so forth.
15. The controller can now be halted, and the debug connection terminated.
16. Fully halting the MCU when in real-time mode is a two-step process. First halt the processor by using the *Halt* button on the toolbar ( ) or by using *Target* → *Halt*. Then take the MCU out of real-time mode by clicking on . Finally, reset the MCU by clicking on .
17. Close CCS debug session by clicking on *Terminate Debug Session* (*Target* → *Terminate all*).

### 5.3.2 INCR_BUILD 2: Closed Current Loop

In this build, BUILD 2, the inner current loop is closed, that is, the inductor current is controlled using a current compensator Gi. Both DC bus and output voltage feedforward are applied to the output of this current compensator to generate the duty cycle of the inverter, as shown in Equation 13. This action makes the plant for the current compensator simple, and a proportional (P) controller can be used to tune the loop of the inner current. The model for the current loop was derived in Section 2.2.

$$duty1PU = \frac{(iL1Meas - iL1Ref) * Gi\_GainKp + v1Meas}{vBusHalfMeas} \quad (13)$$

Complete software diagram for this build as illustrated in Figure 27.

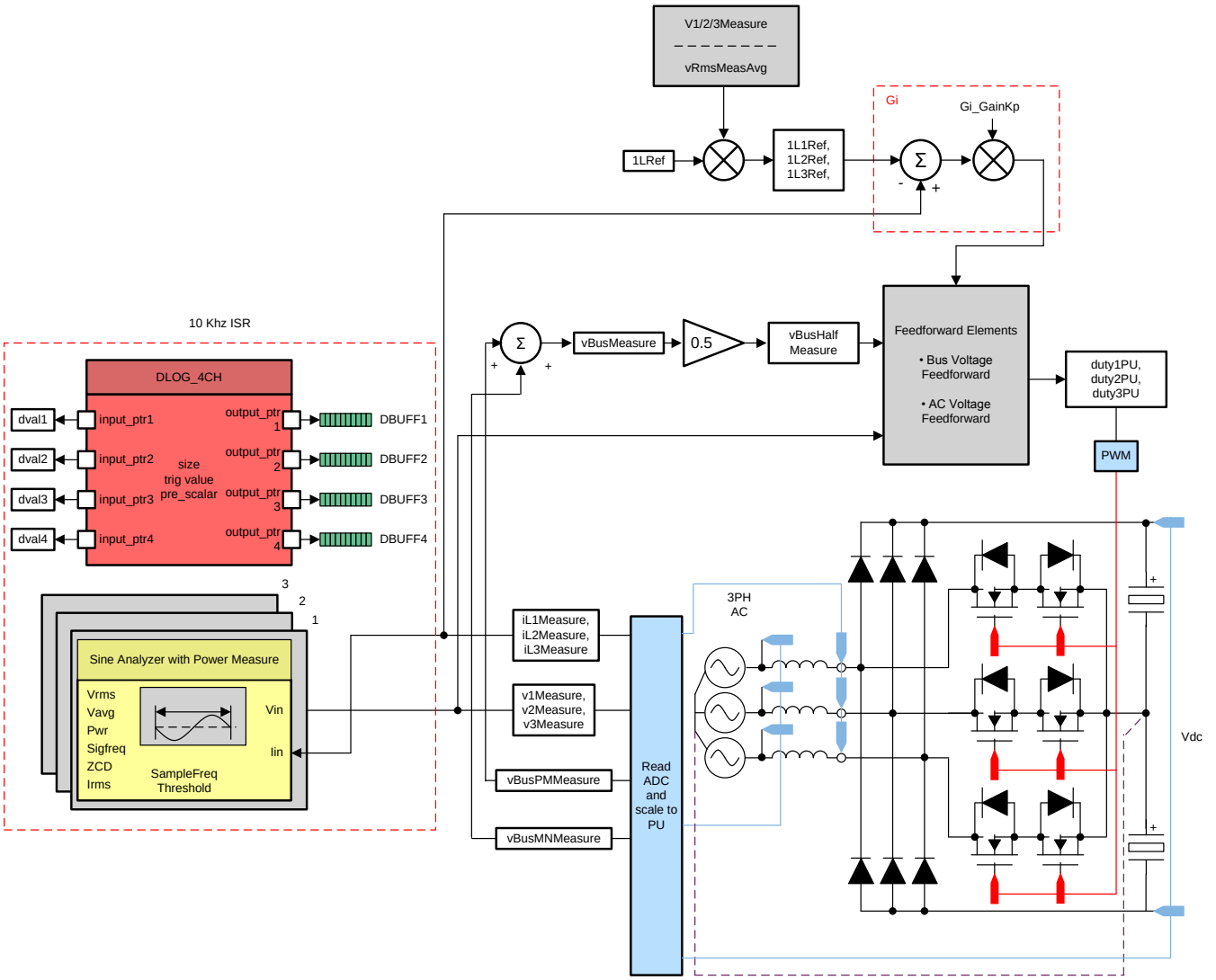





**Figure 27. Build Level 2 Control Software Diagram: Closed Current Loop**

### 5.3.2.1 Setting Software Options for BUILD 2

1. Make sure the hardware is setup as outlined in Section 4.1. Do not supply any HV power yet to the board. For this build, connect the neutral from the three-phase power supply to the board, as shown in Figure 16.
2. powerSUITE Settings : On the *powerSUITE* page under *Project Options* section, select *Closed Current Loop* for the build level. Save the page.
3. Under *Control Loop Design*, options for the current loop tuning will automatically be selected (*Tuning → Current Loop → COMP1 → P*). Now click on the *Compensation Designer* icon ( ).

### 5.3.2.2 Designing Current Loop Compensator

1. Compensation Designer will then launch, with the model of the current loop plant with parameters specified on the powerSUITE page. Proportional gain value can then be changed to ensure stable closed loop operation. Stability of the system when using the designed compensator can be verified by observing the gain and phase margins on the open loop transfer function plot in the Compensation Designer, as shown in Figure 28.

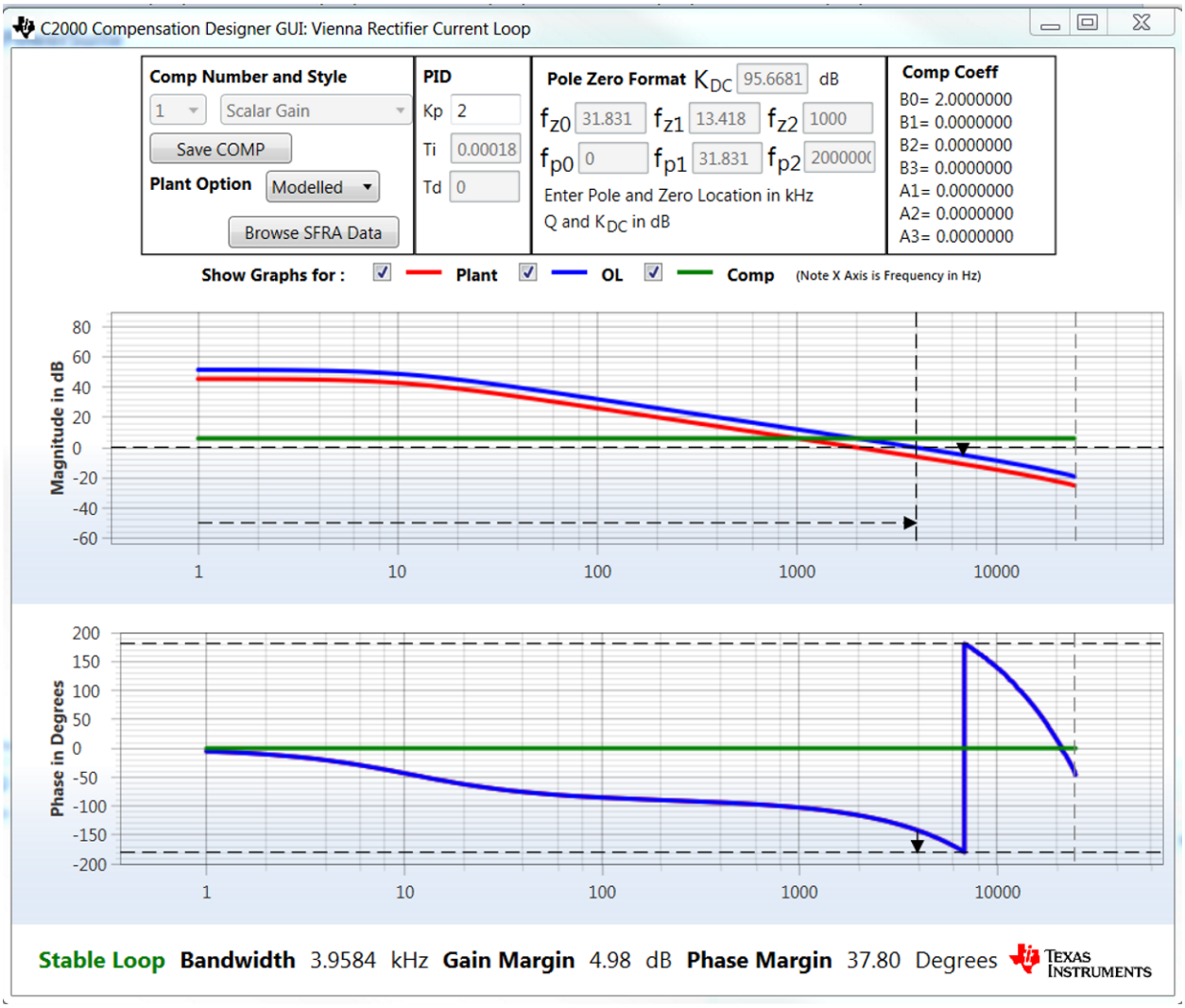



**Figure 28. Current Loop Design Using Compensation Designer**

2. Once satisfied with the proportional gain, click on *Save COMP*. This will save the compensator values into the project.
   - Note: If the project was not selected from the solution adapter, changes to the compensator will not be allowed to design one's own solution. Select the solution through the solution adapter.
3. Close the Compensation Designer, and return to the powerSUITE page.

   Note: The P gain in the code is set to 2.0, this gain would give a OL plot with margins that may look too small; however, this gain has been adjusted keeping in mind the actual measurements from the design where the measured gain is slightly lower. Therefore, a higher gain to the compensator can be applied without sacrificing the actual margins.

#### 5.3.2.3 Building and Loading the Project and Setting up Debug

1. Right click on the project name, and click *Rebuild Project*. The project will build successfully. Click *Run → Debug*, which will launch a debugging session. In the case of dual CPU devices, a window may appear to select the CPU the debug needs to be performed. In this case, select CPU1. The project will then load on the device, and CCS debug view will become active. The code will halt at the start of the main routine.
2. To add the variables in the watch and expressions window click *View → Scripting Console* to open the scripting console dialog box. On the upper right corner of this console, click on *Open* to browse to the *setupdebugenv_build2.js* script file, which is located inside the project folder. This file will populate the watch window with appropriate variables needed to debug the system. Click on *Continuous Refresh* button (  ) on the watch window to enable continuous update of values from the controller. The watch window will appear as Figure 29.

Variables | Expressions | Registers | Breakpoints

| Expression | Type | Value | Address |
|---|---|---|---|
| BuildInfo | enum enum_BuildLevel | BuildLevel2_CurrentLoop | 0x0000B010@Data |
| boardStatus | enum enum_boardStatus | boardStatus_Idle | 0x0000B00E@Data |
| clearTrip | int | 0 | 0x0000B001@Data |
| iLRef | float | 0.0500000007 | 0x0000B024@Data |
| closeGiLoop | int | 0 | 0x0000B00B@Data |
| EPwm1Regs.TZFLG.all | unsigned int | 13 | 0x00004093@Data |
| EPwm2Regs.TZFLG.all | unsigned int | 4 | 0x00004193@Data |
| EPwm3Regs.TZFLG.all | unsigned int | 4 | 0x00004293@Data |
| guiVbus | float | 1.10873163 | 0x0000B0BE@Data |
| guiVbusPM | float | 1.83267128 | 0x0000B0B8@Data |
| guiVbusMN | float | -0.722792923 | 0x0000B0BA@Data |
| guiVrms1 | float | 0.0 | 0x0000B01A@Data |
| guiVrms2 | float | 0.0 | 0x0000B026@Data |
| guiVrms3 | float | 0.0 | 0x0000B03E@Data |
| guiIrms1 | float | 0.0 | 0x0000B0DA@Data |
| guiIrms2 | float | 0.0 | 0x0000B018@Data |
| guiIrms3 | float | 0.0 | 0x0000B01C@Data |
| guiPF1 | float | 0.0 | 0x0000B042@Data |
| guiPF2 | float | 0.0 | 0x0000B036@Data |
| guiPF3 | float | 0.0 | 0x0000B032@Data |
| *Add new expression* | | | |

**Figure 29. Build Level 2: Closed Current Loop Expressions View**

3. Enable real-time mode by hovering the mouse on the buttons on the horizontal toolbar, and clicking the  button.

#### 5.3.2.4 Running the Code

1. Run the project by clicking  .
2. It is advisable to test first at a low voltage. Therefore, the input AC voltage is raised to only 40 Vrms, 60 Hz.
3. The input current and voltage will look like Figure 30.

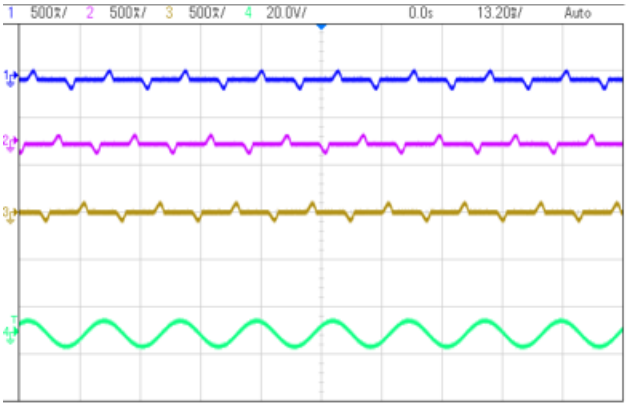



**Figure 30. Build Level 2: Scope Capture IL1, IL2, IL3 and V1 (40Vrms L-N) With PWM Tripped**

4. A current reference is set by changing the iLRef variable in the expressions view. This variable is set to 0.05.
5. Clear the trip by setting the *clearTrip* variable to *1*.
6. As soon as the trip is cleared, a sinusoidal current will be seen to be drawn from the input, which verifies correct operation of the current loop. The waveform will look like Figure 31.

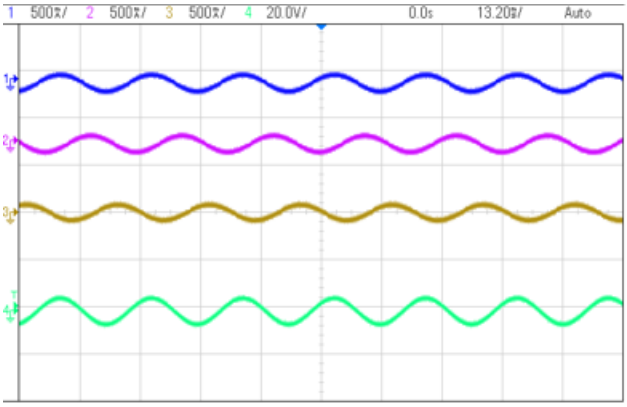



**Figure 31. Build Level 2: Scope Capture IL1, IL2, IL3 and V1 (40Vrms L-N) With Current Loop Closed**

7. The guiVbus will be close to 190 V, and the input AC current per phase will be close to 0.6 Amps
8. Raise the current reference iLRef to 0.1. Observe the bus voltage to go to 266 V and the input current to around 1.2 Amps.
9. Now raise the input AC voltage slowly to 120 Vrms. The board will maintain the input current to be constant as the input voltage rises.

10. Once the input has stabilized to 120 Vrms, raise the iLRef so the guiVbus is 600 V; for the test conditions specified, this will correspond to iLRef = 1.65. The current waveforms will look like Figure 32.

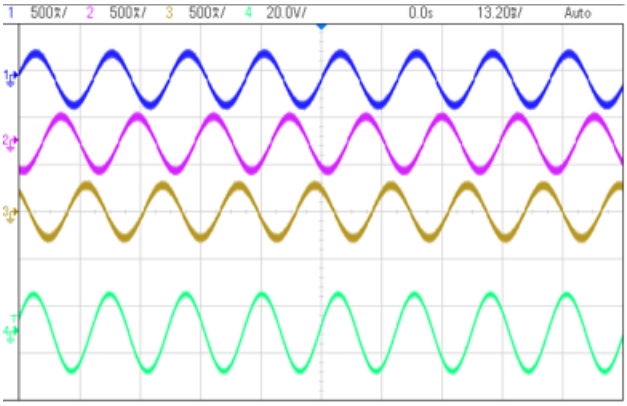



**Figure 32. Build Level 2: Scope Capture IL1, IL2, IL3 and V1 (120Vrms L-N) With Current Loop Closed**

11. As only a proportional gain is used in the compensator, the current reference minus the feedback error is never zero. Notice the current drawn to deviate slightly from the reference.
12. SFRA is integrated in the software of this build to verify the designed compensator provides enough gain and phase margin by measuring on hardware. To run the SFRA keep the project running, and from the cfg page, click on the *SFRA* icon. SFRA GUI will pop-up.
13. Select the options for the device on the SFRA GUI. For example, for F28377D select floating point. Click on *Setup Connection*. On 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 *Connect*.

14. The SFRA GUI will connect to the device. A SFRA sweep can now be started by clicking *Start Sweep*. The complete SFRA sweep will take a few minutes to finish. Activity can be monitored by seeing the progress bar on the SFRA GUI and also checking the flashing of blue LED on the back on the control card that indicates UART activity. Once complete, a graph with the open loop plot will appear, as in Figure 33. This verifies that the designed compensator is indeed stable.

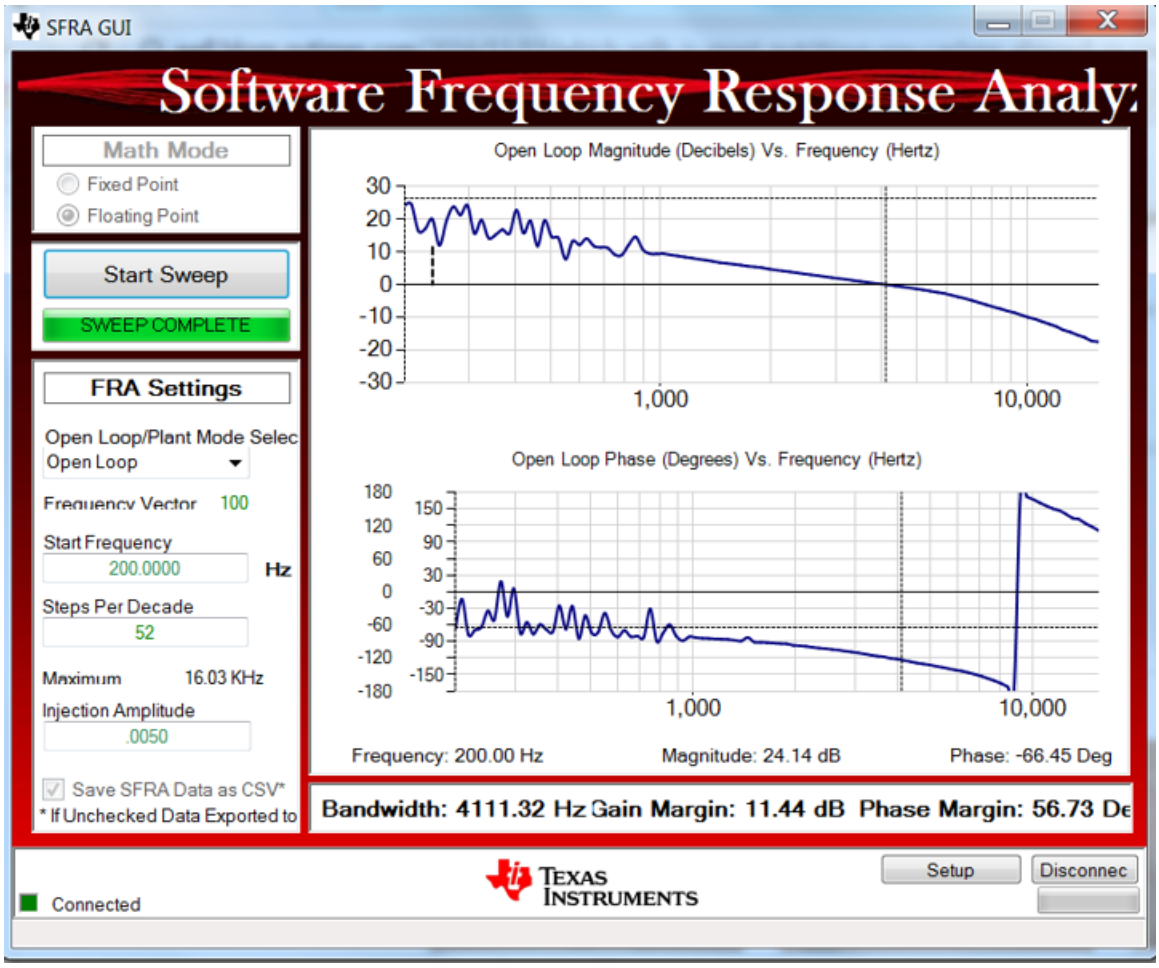



**Figure 33. SFRA Run on Closed Current Loop**

The frequency response data is also saved in the project folder under an SFRA data folder and is time stamped with the time of the SFRA run. Also, note the measured gain and phase margin are close to the modelled values (see Figure 34).

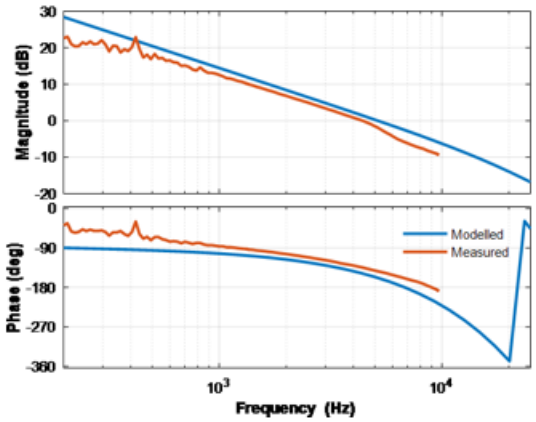



**Figure 34. Modelled Versus Measured OL Response for Current Loop**

15. Click on the Compensation Designer again from the CFG page, and choose *SFRA Data* for plant option on the GUI. This option will use the measured plant information to design the compensator and can be used to fine tune the compensation. By default the Compensation Designer will point to the latest SFRA run and show up as . If a previous SFRA run plant information needs to be used, the user can select the *SFRAData.csv* file by browsing to it by clicking on *Browse SFRA Data*. Close Compensation Designer to return to the cfg page once this is done.

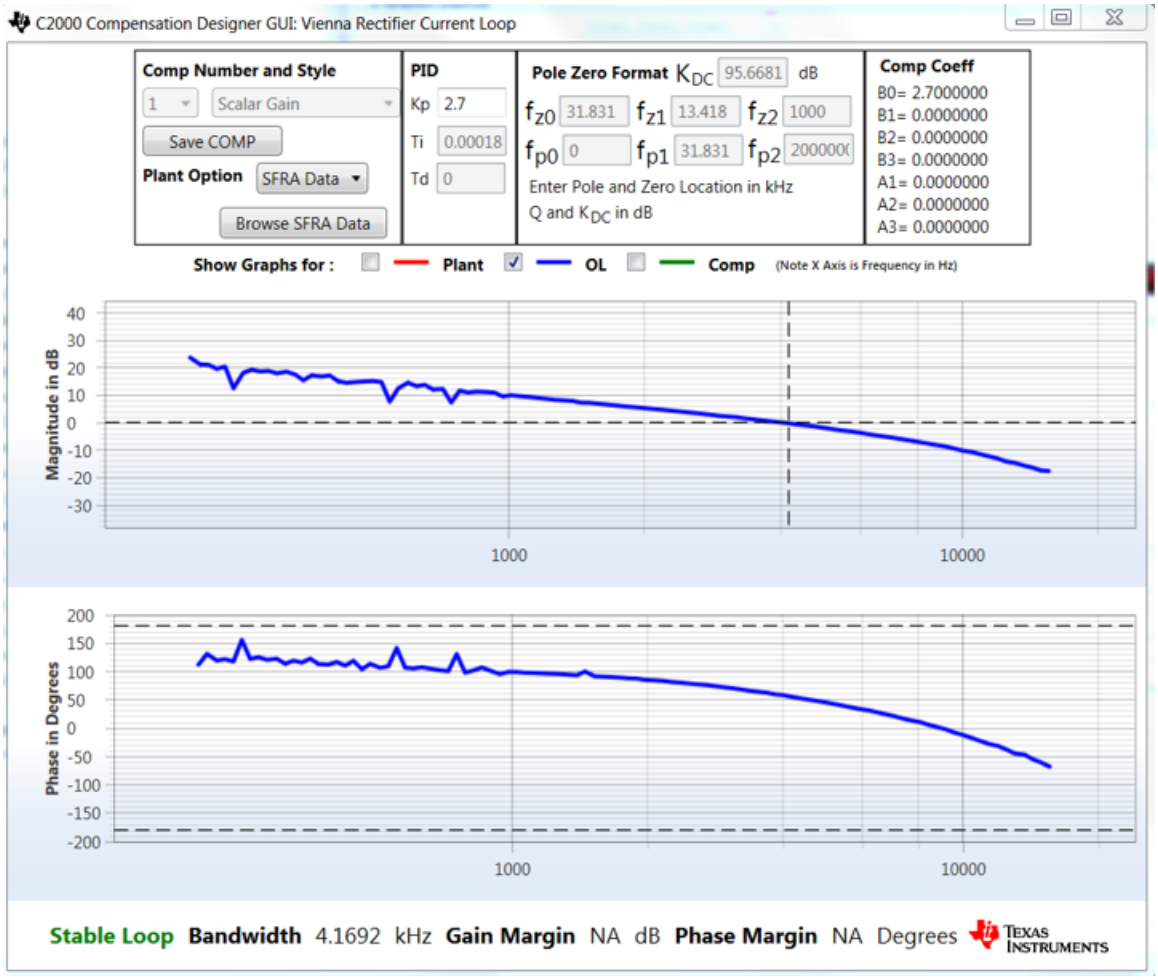



**Figure 35. Compensation Designer With Measured Plant Frequency Response Data**

16. This action verifies the current compensator design. Note to phase if off because there is a negative sign in the control loop.
17. To bring the system to a safe stop, bring the input AC voltage down to zero, and observe the guiVBus will come down to zero as well.
18. Fully halting the MCU when in real-time mode is a two-step process. First halt the processor by using the *Halt* button on the toolbar ( ) or by using *Target* → *Halt*. Then take the MCU out of real-time mode by clicking on . Finally, reset the MCU ( ) .
19. Close the CCS debug session by clicking on *Terminate Debug Session* (*Target* → *Terminate all*).

### 5.3.3 INCR_BUILD 3: Closed Voltage and Current Loop

In this build the outer voltage loop is closed with the inner current loop closed (designed in BUILD 2). The model of the outer voltage loop was derived in Figure 36. A PI-based compensator is used and tuned through the Compensation Designer. Figure 36 shows the software diagram for this build.

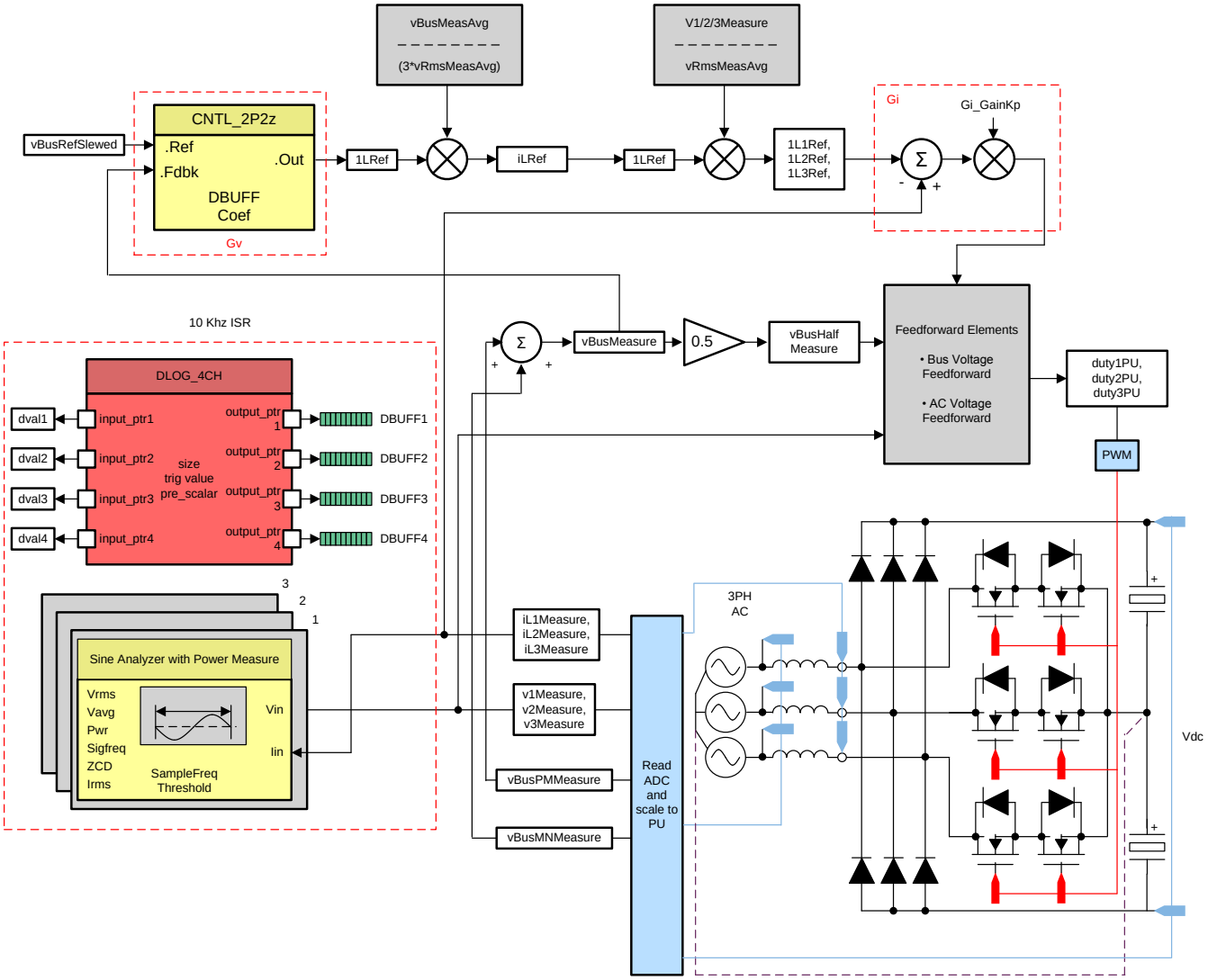





**Figure 36. Build Level 3 Control Diagram: Output Voltage Control With Inner Current Loop**

#### *5.3.3.1 Setting Software Options for BUILD 3*

1. Make sure the hardware is setup as outlined in Section 4.1. Do not supply any HV power yet to the board. For this build connect the neutral from the three-phase power supply to the board Figure 16.
2. Under *Project Options*, select *Closed Voltage & Current Loop*.
3. Under *Control Loop Design*, select *Tuning* as *Voltage Loop*. Style will be preset to *PI*. Save the page by *Ctrl* + *S*, and click on the Compensation Designer button ( ).
4. Make sure the load connected at the output of the board is correctly entered on the powerSUITE cfg page because this load value is used in the design of the voltage compensator.

### 5.3.3.2 Designing Voltage Loop Compensator

1. Compensation designer will then launch with the model of the voltage loop plant, as shown in Figure 37. The PI compensator can be edited to get the desired gain and phase margin, keeping in mind the bandwidth of the voltage loop has an inverse relationship with the THD achieved. Typically in a PFC application, this bandwidth is kept at ~10 Hz.

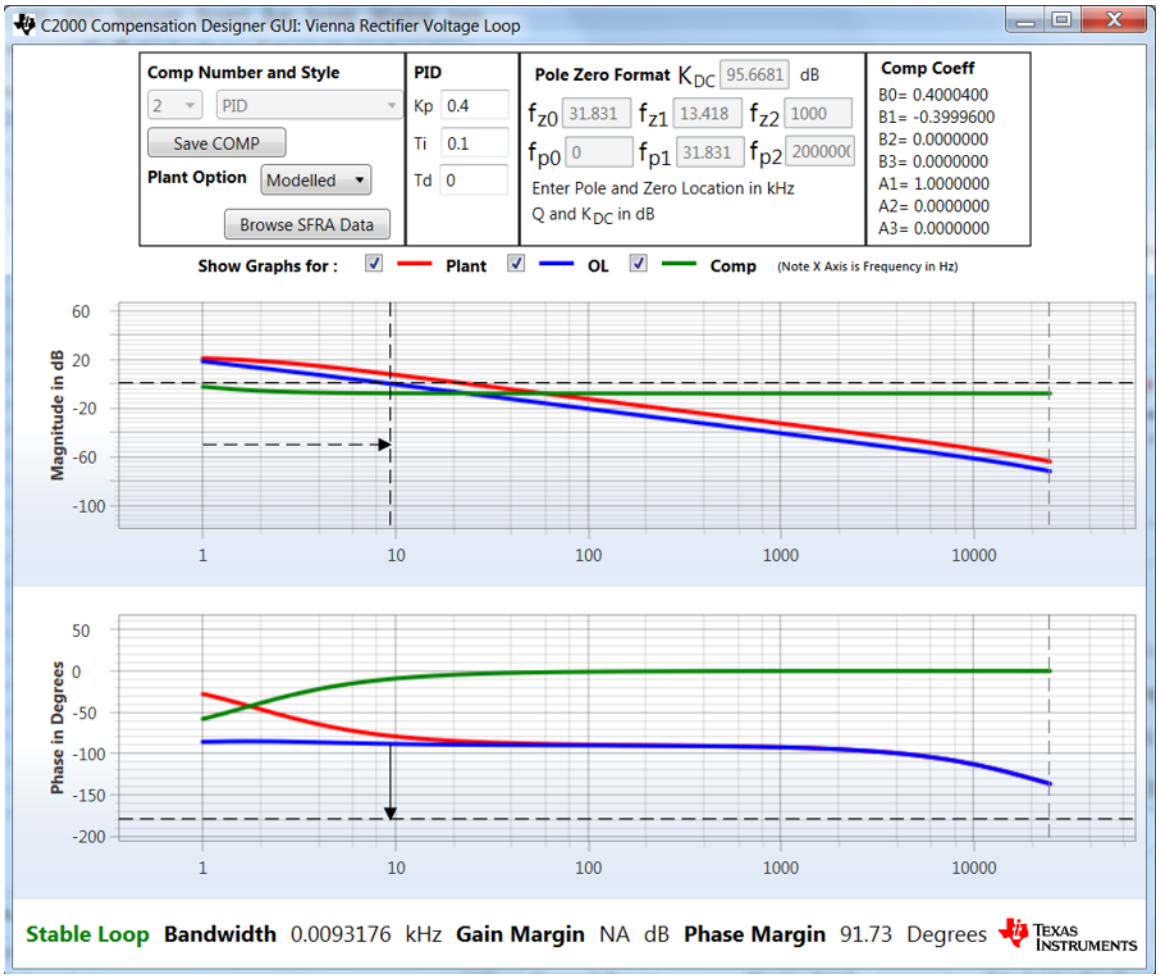



**Figure 37. Voltage Loop PI Compensation Tuning Using Compensation Designer**

2. Once satisfied with the compensator design, click on *Save COMP*. This will save the compensator values into the project.
   - Note: If the project was not selected from the solution adapter, changes to the compensator will not be allowed. To design one's own, select the solution through the solution adapter.
3. Close the Compensation Designer, and return to the powerSUITE page. Save using *Ctrl + S*.

#### 5.3.3.3 Building and Loading the Project and Setting up Debug

1. Right click on the project name, and click *Rebuild Project*. The project will build successfully. Click *Run → Debug*, which will launch a debugging session. In the case of dual CPU devices, a window may appear to select the CPU the debug needs to be performed. In this case, select CPU1. The project will then load on the device, and CCS debug view will become active. The code will halt at the start of the main routine.
2. To add the variables in the watch and expressions window, click *View → Scripting Console* to open the scripting console dialog box. On the upper right corner of this console, click on *Open* to browse to the *setupdebugenv_build3.js* script file located inside the project folder. This file will populate the watch window with appropriate variables needed to debug the system. Click on the *Continuous Refresh* button (  ) on the watch window to enable continuous update of values from the controller. The watch window will appear as shown in Figure 38.

Variables | Expressions | Registers | Breakpoints

| Expression | Type | Value | Address |
|---|---|---|---|
| BuildInfo | enum enum_BuildLevel | BuildLevel3_VoltageAndCurrentLoop | 0x0000B010@Data |
| boardStatus | enum enum_boardStatus | boardStatus_Idle | 0x0000B00E@Data |
| clearTrip | int | 0 | 0x0000B001@Data |
| vBusRef | float | 1.32000005 | 0x0000B028@Data |
| vBusMeas | float | 0.0031801695 | 0x0000B082@Data |
| vBusRefSlewed | float | 0.0 | 0x0000B022@Data |
| closeGvLoop | int | 0 | 0x0000B008@Data |
| closeGiLoop | int | 0 | 0x0000B00B@Data |
| EPwm1Regs.TZFLG.all | unsigned int | 13 | 0x00004093@Data |
| EPwm2Regs.TZFLG.all | unsigned int | 4 | 0x00004193@Data |
| EPwm3Regs.TZFLG.all | unsigned int | 4 | 0x00004293@Data |
| guiVbus | float | 1.63113213 | 0x0000B0BE@Data |
| guiVbusPM | float | 1.89541566 | 0x0000B0B8@Data |
| guiVbusMN | float | -0.264472604 | 0x0000B0BA@Data |
| guiVrms1 | float | 0.0 | 0x0000B01A@Data |
| guiVrms2 | float | 0.0 | 0x0000B026@Data |
| guiVrms3 | float | 0.0 | 0x0000B03E@Data |
| guiIrms1 | float | 0.0 | 0x0000B0DA@Data |
| guiIrms2 | float | 0.0 | 0x0000B018@Data |
| guiIrms3 | float | 0.0 | 0x0000B01C@Data |
| guiPF1 | float | 0.0 | 0x0000B042@Data |
| guiPF2 | float | 0.0 | 0x0000B036@Data |
| guiPF3 | float | 0.0 | 0x0000B032@Data |
| *Add new expression* | | | |

**Figure 38. Build Level 3: Expressions View**

3. Enable real-time mode by hovering the mouse on the buttons on the horizontal toolbar and clicking the  button.

#### 5.3.3.4 Running the Code

1. Run the project by clicking  .
2. Raise the DC bus voltage to 120-Vrms VL-N or 208-Vrms VL-L, 60 Hz.
3. The DC voltage reference is set by the variable vBusRef. This value is set to as 1.32, which corresponds to 600 V for this design. Refer to *calculations.xls* sheet for details.
4. Clear the trip by setting the *clearTrip* variable to *1*. The bus voltage will then rise to be 600 V.
5. Closed loop operation can be verified by comparing the *vBusRef* and *vBusMeas* in the expressions window as shown in Figure 39.

Variables | Expressions | Registers | Breakpoints

| Expression | Type | Value | Address |
|---|---|---|---|
| BuildInfo | enum enum_BuildLevel | BuildLevel3_VoltageAndCurrentLoop | 0x0000B010@Data |
| boardStatus | enum enum_boardStatus | boardStatus_NoFault | 0x0000B00E@Data |
| clearTrip | int | 0 | 0x0000B001@Data |
| vBusRef | float | 1.32000005 | 0x0000B028@Data |
| vBusMeas | float | 1.31717896 | 0x0000B082@Data |
| vBusRefSlewed | float | 1.32000005 | 0x0000B022@Data |
| closeGvLoop | int | 1 | 0x0000B008@Data |
| closeGiLoop | int | 1 | 0x0000B00B@Data |
| EPwm1Regs.TZFLG.all | unsigned int | 9 | 0x00004093@Data |
| EPwm2Regs.TZFLG.all | unsigned int | 9 | 0x00004193@Data |
| EPwm3Regs.TZFLG.all | unsigned int | 9 | 0x00004293@Data |
| guiVbus | float | 599.282104 | 0x0000B0BE@Data |
| guiVbusPM | float | 300.124268 | 0x0000B0B8@Data |
| guiVbusMN | float | 299.153046 | 0x0000B0BA@Data |
| guiVrms1 | float | 118.64994 | 0x0000B01A@Data |
| guiVrms2 | float | 121.02787 | 0x0000B026@Data |
| guiVrms3 | float | 121.515961 | 0x0000B03E@Data |
| guiIrms1 | float | 1.90073025 | 0x0000B0DA@Data |
| guiIrms2 | float | 1.97687936 | 0x0000B018@Data |
| guiIrms3 | float | 1.95430648 | 0x0000B01C@Data |
| guiPF1 | float | 1.00191689 | 0x0000B042@Data |
| guiPF2 | float | 1.00196981 | 0x0000B036@Data |
| guiPF3 | float | 1.00181603 | 0x0000B032@Data |
| Gs_Out | float | 0.0 | 0x0000B046@Data |
| Add new expression | | | |

**Figure 39. Build Level 3: Expressions Window**

6. SFRA is integrated in the software of this build to verify the designed compensator provides enough gain and phase margin by measuring on hardware. To run the SFRA, keep the project running, and from the cfg page, click on the *SFRA* icon. SFRA GUI will pop up.
7. Select the options for the device on the SFRA GUI. For example, for F28377D, select floating point. Click on *Setup Connection*, and on 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 Connect.

8. The SFRA GUI will connect to the device. A SFRA sweep can now be started by clicking *Start Sweep*. The complete SFRA sweep will take a few minutes to finish. Activity can be monitored by seeing the progress bar on the SFRA GUI and checking the flashing of blue LED on the back on the control card that indicates UART activity. Once complete, a graph with the open loop plot will appear, as seen in Figure 40. This action verifies that the designed compensator is indeed stable.

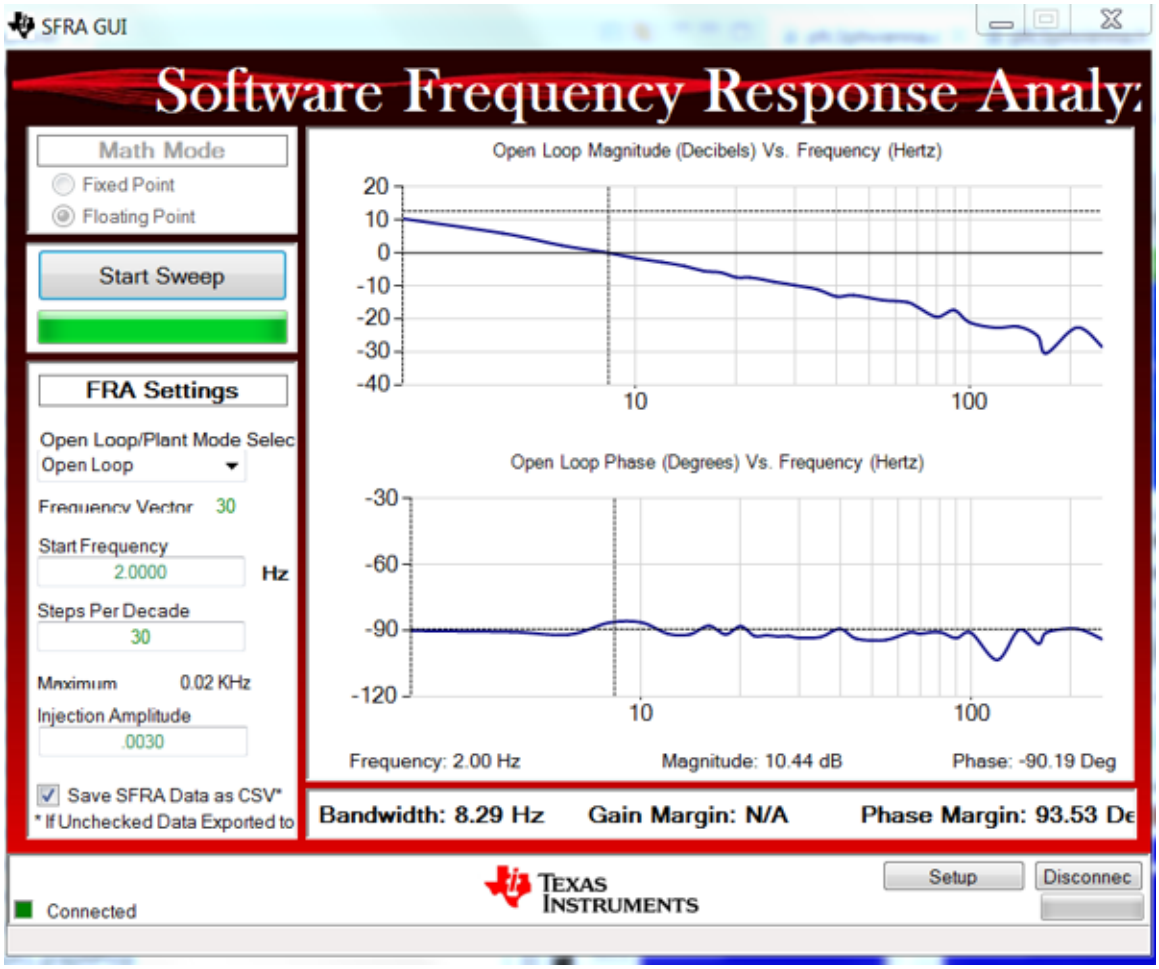



**Figure 40. SFRA Run on Closed Voltage Loop**

The frequency response data is also saved in the project folder under an SFRA data folder and is time stamped with the time of the SFRA run.

Note the measured gain and phase margin are close to the modelled values, as shown in Figure 41.

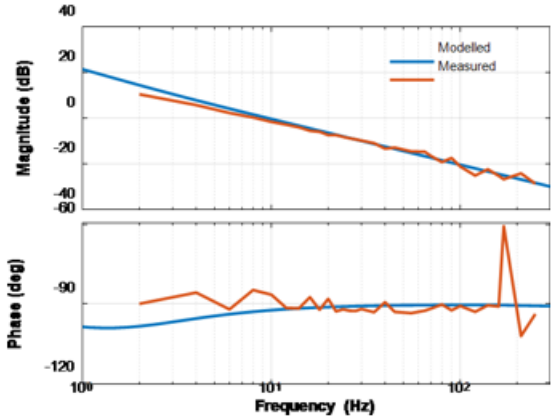



**Figure 41. Voltage Loop Modelled Versus Measured Comparison**

9. Click on the Compensation Designer again from the CFG page, and choose *SFRA Data* for plant option on the GUI. This option will use the measured plant information to design the compensator, as shown in Figure 42. This option can be used to fine tune the compensation. By default the Compensation Designer will point to the latest SFRA run. If a previous SFRA run plant information needs to be used the user can select the *SFRAData.csv* file by browsing to it by clicking on *Browse SFRA Data*. Close Compensation Designer to return to the cfg page once done.

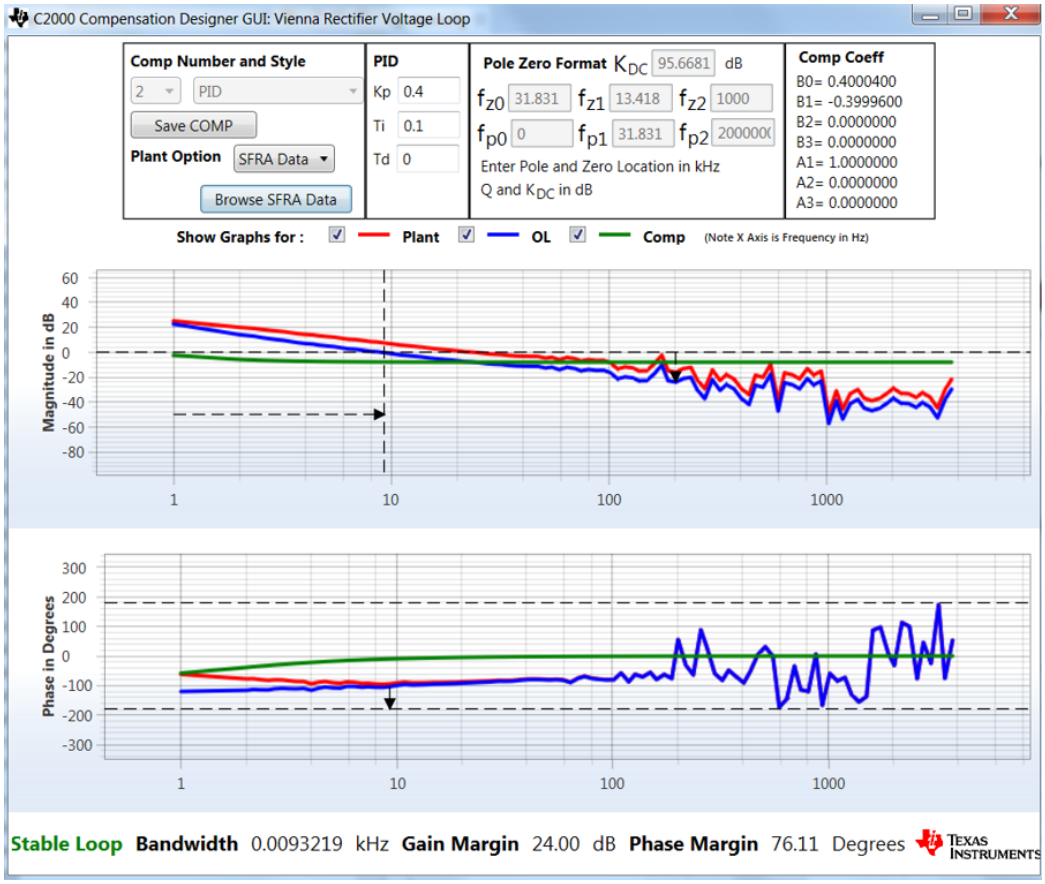



**Figure 42. Voltage Loop Compensation Designed With Measured Plant Information**

10. This verifies the voltage compensator design.
11. To bring the system to a safe stop bring the input AC voltage down to zero, observe the guiVBus will

come down to zero as well.

12. Fully halting the MCU when in real-time mode is a two-step process. First halt the processor by using the *Halt* button on the toolbar ( ) or by using *Target* → *Halt*. Then take the MCU out of real-time mode by clicking on . Finally, reset the MCU ( ).
13. Close CCS debug session by clicking on *Terminate Debug Session* (*Target* → *Terminate all*).

### 5.3.4 INCR_BUILD 4: Closed Balance, Voltage, and Current Loop

In this build the board is operated as a three-wire system, that is, the neutral of the power supply is not connected to the DC midpoint of the output. To maintain the DC bus balance, a balance loop with a simple proportional gain is added in the control structure as shown in Figure 43. In this build a third harmonic injection is also carried out, which helps in stabilizing the DC bus balance point.

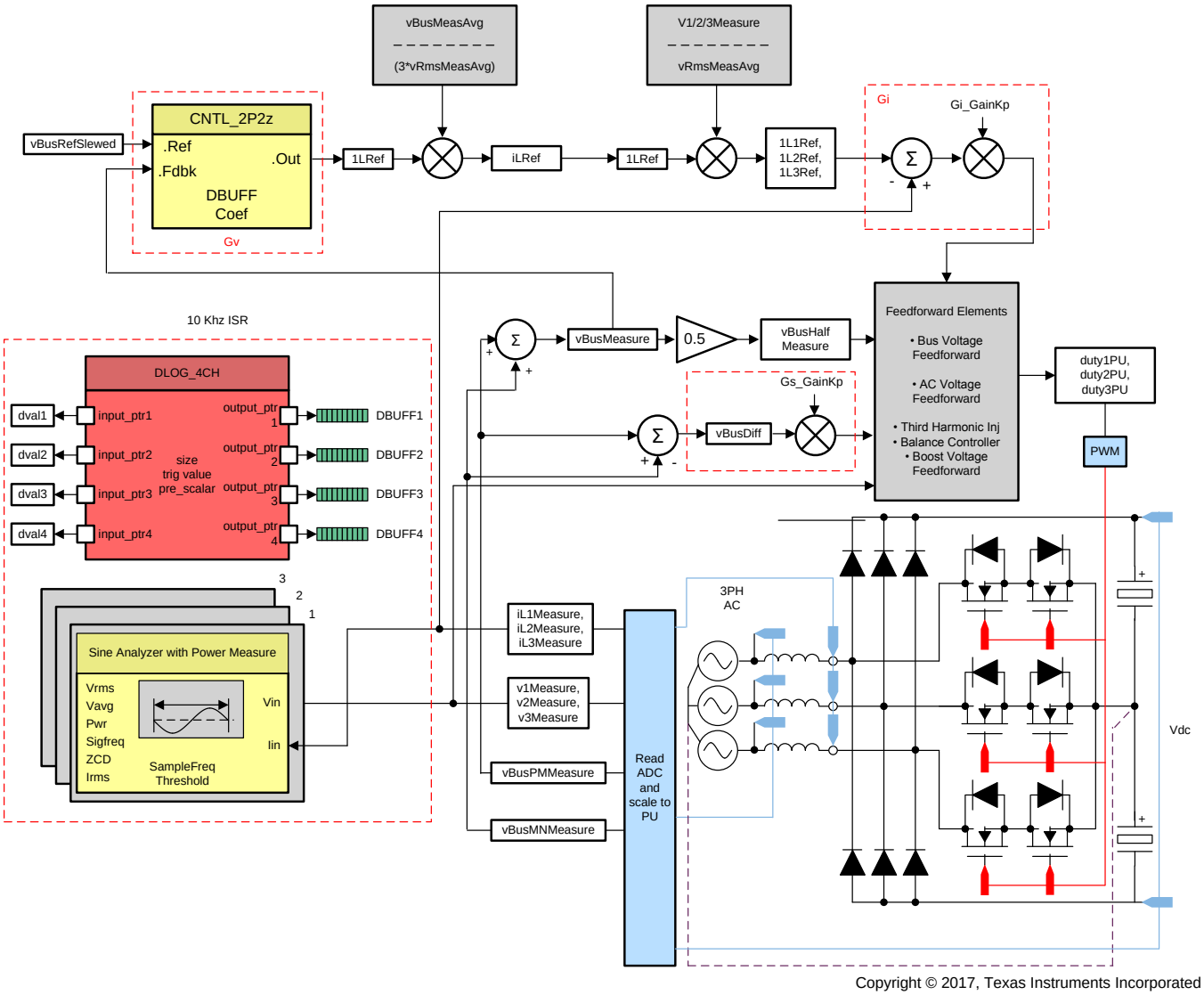





**Figure 43. Build Level 4 Control Diagram: Output Voltage, Inductor Current, and Bus Cap Balance Loop**

#### 5.3.4.1 Setting Software Options for BUILD 4

1. Make sure the hardware is setup, as shown in Figure 44. One major difference from the previous builds is the Neutral is no longer connected to the midpoint of the DC bus capacitor.

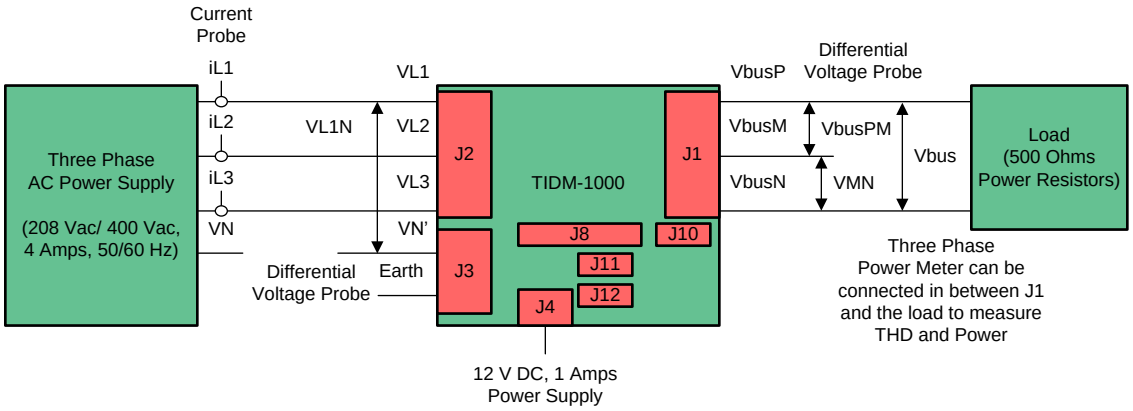



**Figure 44. Build Level 4 Hardware Setup Diagram**

2. On the powerSUITE page under *Project Options* select *Closed Voltage, Current and Bus Cap Balance Loop*.
3. Save the page.

#### 5.3.4.2 Building and Loading the Project and Setting up Debug

1. Now right click on the project name and click *Rebuild Project*. The project will build successfully. Click *Run → Debug*; this will launch a debugging session. In the case of dual CPU devices, a window may appear to select the CPU the debug needs to be performed.. In this case select CPU1. The project will then load on the device and CCS debug view will become active. The code will halt at the start of the main routine.
2. To add the variables in the watch or expressions window click *View → Scripting Console* to open the scripting console dialog box. On the upper-right corner of this console, click on open to browse to the *setupdebugenv_build4.js* script file located inside the project folder. This file will populate the watch window with appropriate variables needed to debug the system. Click on *Continuous Refresh* button (  ) on the watch window to enable continuous update of values from the controller. The watch window will appear as in Figure 45.

Variables | Expressions | Registers

| Expression | Type | Value | Address |
|---|---|---|---|
| BuildInfo | enum enum_BuildLevel | BuildLevel4_BalanceVoltageAndCurr... | 0x0000B010@Data |
| boardStatus | enum enum_boardStatus | boardStatus_Idle | 0x0000B00E@Data |
| clearTrip | int | 0 | 0x0000B000@Data |
| vBusRef | float | 1.32000005 | 0x0000B028@Data |
| vBusMeas | float | 0.00413890835 | 0x0000B082@Data |
| vBusRefSlewed | float | 0.0 | 0x0000B022@Data |
| closeGsLoop | int | 0 | 0x0000B001@Data |
| closeGvLoop | int | 0 | 0x0000B009@Data |
| closeGiLoop | int | 0 | 0x0000B008@Data |
| EPwm1Regs.TZFLG.all | unsigned int | 13 | 0x00004093@Data |
| EPwm2Regs.TZFLG.all | unsigned int | 4 | 0x00004193@Data |
| EPwm3Regs.TZFLG.all | unsigned int | 4 | 0x00004293@Data |
| guiVbus | float | 2.05974889 | 0x0000B0BE@Data |
| guiVbusPM | float | 1.85727727 | 0x0000B0B8@Data |
| guiVbusMN | float | 0.202745244 | 0x0000B0BA@Data |
| guiVrms1 | float | 0.0 | 0x0000B01A@Data |
| guiVrms2 | float | 0.0 | 0x0000B026@Data |
| guiVrms3 | float | 0.0 | 0x0000B03E@Data |
| guiIrms1 | float | 0.0 | 0x0000B0DA@Data |
| guiIrms2 | float | 0.0 | 0x0000B018@Data |
| guiIrms3 | float | 0.0 | 0x0000B01C@Data |
| guiPF1 | float | 0.0 | 0x0000B042@Data |
| guiPF2 | float | 0.0 | 0x0000B036@Data |
| guiPF3 | float | 0.0 | 0x0000B032@Data |
| *Add new expression* | | | |

**Figure 45. Build Level 4: Expressions View**

3. Enable real-time mode by hovering the mouse on the buttons on the horizontal toolbar and clicking the  button.

#### 5.3.4.3 Running the Code

1. Run the project by clicking.
2. Raise the AC input to 120-Vrms VL-L and 208-Vrms VL-L, 60 Hz. A rectified current is going to be drawn from the input with PF close to 0.7 as shown in Figure 46.

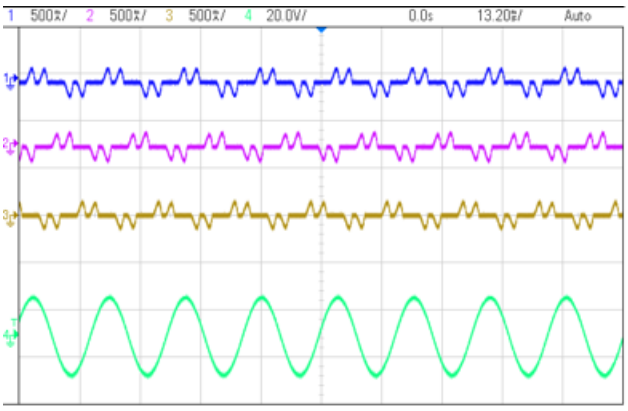



**Figure 46. Build Level 4: Scope Capture IL1, IL2, IL3 and V1 (120Vrms L-N) with PWM Tripped**

3. Bus voltage is set by the variable vBusRef and is set at 1.34 which corresponds to 600 V for this design .
4. Start the PFC action by writing a *1* to clearTrip variable
5. The board will now draw sinusoidal current and the PF will be close to 0.99 and THD will be around 2.5%. The scope capture will look as shown in Figure 47.

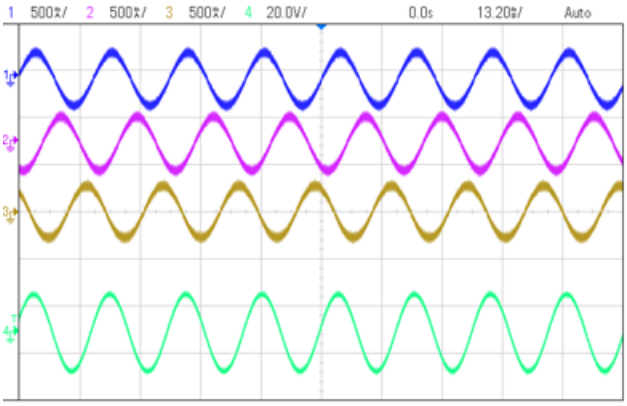



**Figure 47. Build Level 4: Scope Capture IL1, IL2, IL3 and V1 (120-Vrms L-N) With Full PFC Build**

6. The DC bus voltages will also be balanced, that is, guiVbusPM and guiVbusMN will be almost equal, which shows that the closed loop balance controller is working.
7. SFRA is integrated in the software of this build to verify the designed compensator provides enough gain and phase margin by measuring on hardware. To run the SFRA keep the project running and from the cfg page, click on the *SFRA* icon. SFRA GUI will pop-up.
8. Select the options for the device on the SFRA GUI. For example for F28377D, select floating point. Click on setup connection, and on the pop-up window ,uncheck the boot on connect option and select an appropriate COM port and Click *OK*. Return to the SFRA GUI and Click *Connect*.

9. The SFRA GUI will connect to the device. A SFRA sweep can now be started by clicking *Start Sweep*. The complete SFRA sweep will take a few minutes to finish. Activity can be monitored by seeing the progress bar on the SFRA GUI and also by checking the flashing of blue LED on the back on the control card that indicate UART activity. Once complete a graph with the open loop plot will appear, as shown in Figure 48. This graph verifies that the designed compensator is indeed stable.

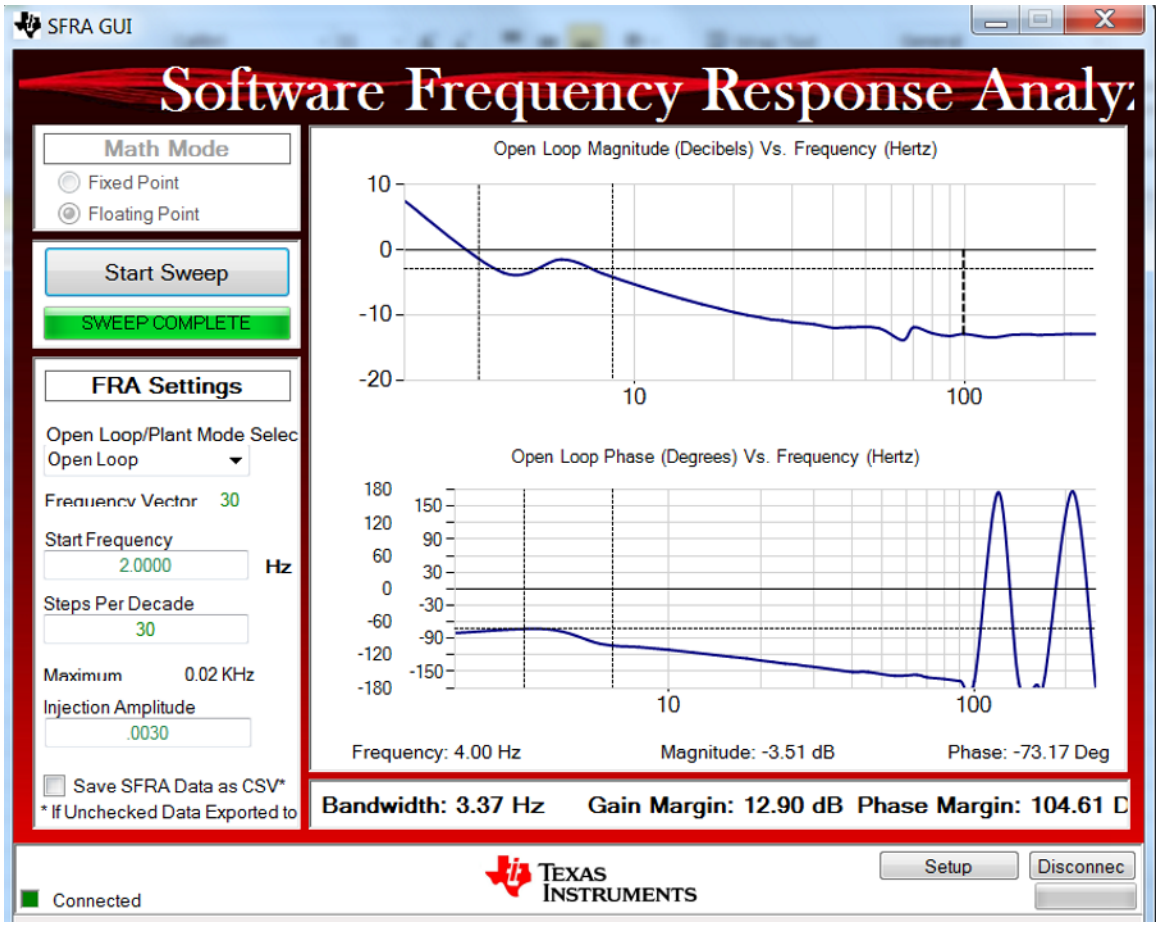



**Figure 48. SFRA Run on Balance Voltage Loop**

10. The balance loop open loop gain is controller by the variable Gs_GainKp and can be adjusted in case the BW is not enough. Though for the balance loop the bandwidth needs to be lower than the outer voltage loop and only 1 to 2 Hz of bandwidth is sufficient.
11. Fully halting the MCU when in real-time mode is a two-step process. First halt the processor by using the *Halt* button on the toolbar ( ) or by using *Target* → *Halt*. Then take the MCU out of real-time mode by clicking on . Finally reset the MCU .
12. Close CCS debug session by clicking on *Terminate Debug Session* (*Target* → *Terminate all*).
.

# 6 Testing and Results

## 6.1 Test Results at Input 208-Vac L-L, 60Hz, Output 600-V DC

### 6.1.1 Startup

The startup sequence of the power stage is shown in Figure 49 with input three phase 208-Vrms VL-L and output bus regulated at 600 V and a 612 W load.

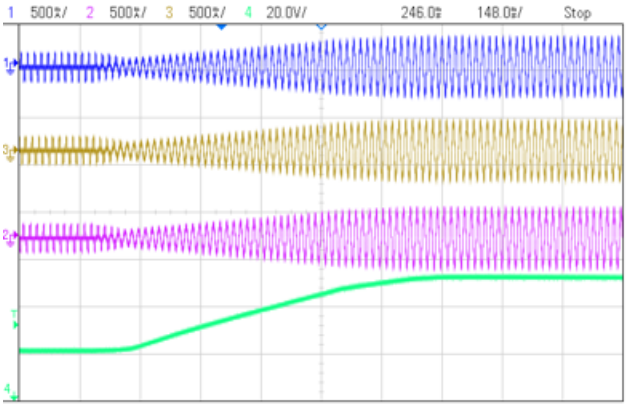



**Figure 49. Startup of PFC Operation at 208-Vac IN, 600-V DC OUT and 612-WΩ Load**

### 6.1.2 Steady State Condition

Steady state current waveform are shown in Figure 50 and Figure 51 for different load conditions 612 W and 1364 W, respectively.

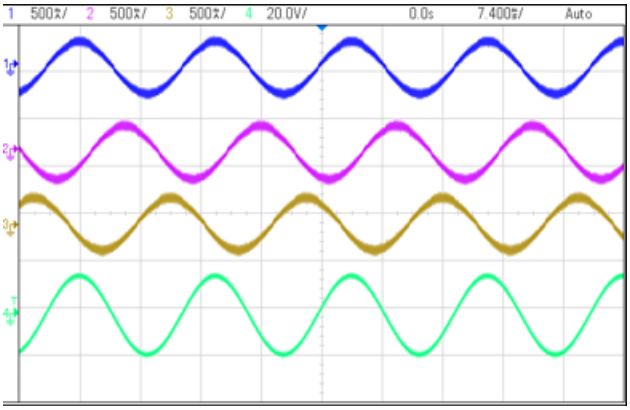



**Figure 50. Steady State 208-Vac IN, 600-V DC OUT 612W, iTHD 2.5%**

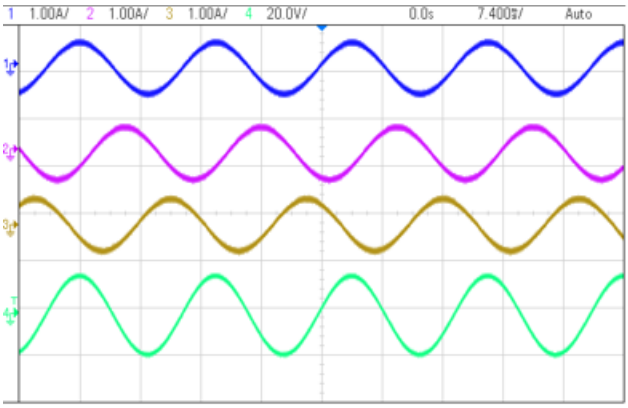



**Figure 51. Steady State 208-Vac, IN 600-V DC OUT 1364W, iTHD 0.96%**

Table 4 lists the detailed test results of this design under varying load conditions with 208-Vac input and 600-V DC output.

**Table 4. Detailed Test Results with 208Vac IN, 600V DC OUT, and Varying Power Levels**

| Vbus_PM | Vbus_MN | VoutTotal | Pin | Iout | Pout | Eff | THD% | PF |
|---|---|---|---|---|---|---|---|---|
| 297.85 | 300.5 | 598.35 | 74 | 0.117 | 70.00695 | 0.946039865 | 23.20% | 0.8263 |
| 298.58 | 300 | 598.58 | 124.4 | 0.2 | 119.716 | 0.962347267 | 17% | 0.9206 |
| 299.12 | 299.8 | 598.92 | 245.1 | 0.4 | 239.568 | 0.977429621 | 8.09% | 0.9777 |
| 299.78 | 299 | 598.78 | 481.3 | 0.788 | 471.83864 | 0.980342074 | 3% | 0.994 |
| 299.52 | 299.6 | 599.12 | 623.7 | 1.022 | 612.30064 | 0.981723008 | 2.54% | 0.9964 |
| 299.66 | 299.4 | 599.06 | 858.8 | 1.405 | 841.6793 | 0.980064392 | 1.71% | 0.998 |
| 299.84 | 299.1 | 598.94 | 1001.6 | 1.637 | 980.46478 | 0.978898542 | 1.52% | 0.9984 |
| 299.3 | 299.82 | 599.12 | 1150.6 | 1.878 | 1125.14736 | 0.977878811 | 1.22% | 0.9988 |
| 300.13 | 298.8 | 598.93 | 1399.7 | 2.278 | 1364.36254 | 0.974753547 | 0.96% | 0.999 |

Figure 52 shows the efficiency data plotted under these test conditions.

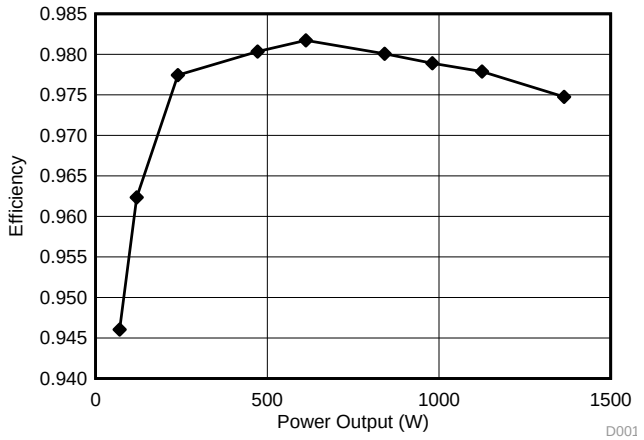



**Figure 52. Efficiency at 208-Vac IN and 600-V DC OUT**

Figure 53 shows the THD data plotted under these test conditions.

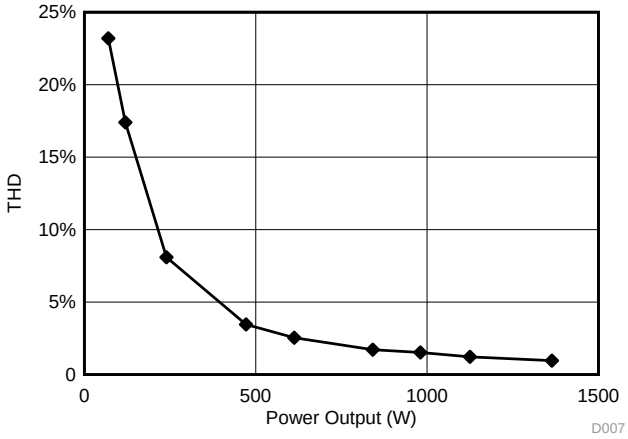



**Figure 53. THD at 208-Vac IN and 600-V DC OUT**

Figure 54 shows the PF data plotted under these test conditions.

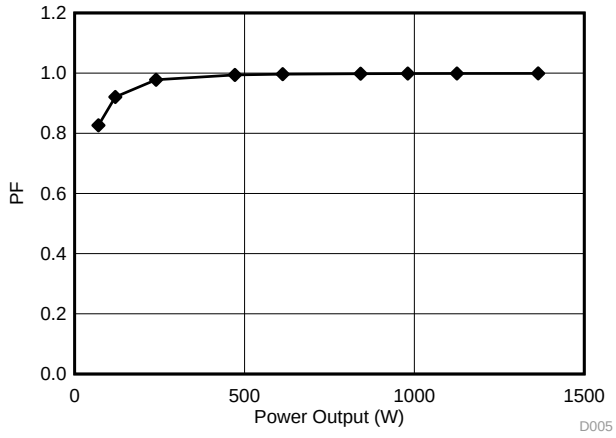



**Figure 54. PF at 208-Vac IN and 600-V DC OUT**

### 6.1.3 Transient Test With Step Load Change

The DC bus regulation loop is kept intentionally low; therefore, to avoid excessive overshoot under load transients, a non-linear element is added to the voltage regulation loop. Figure 55 shows the voltage overshoot with linear PI under load transient, and Figure 56 shows the non-linear voltage loop response to the same load transient. It is clear that with the non-linear loop the overshoot can be reduced significantly, though it can give rise to some extra disturbance. Further fine tuning can be done to make this transition smooth and avoid the extra disturbance.

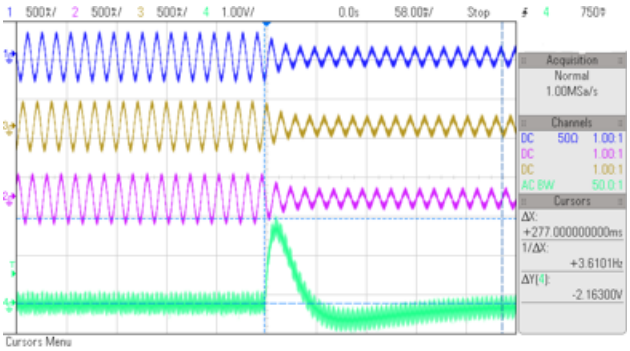



**Figure 55. Linear Voltage Loop Step Load Performance From 238W → 697 W, 208-Vac IN, 600-V DC OUT**

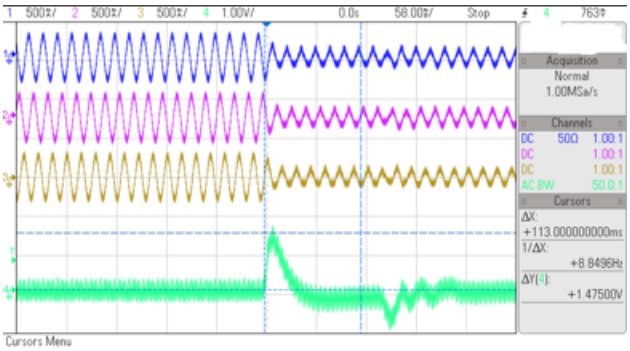



**Figure 56. Linear Voltage Loop Step Load Performance From 238 W → 697 W, 400-Vac IN, 700-V DC OUT**

## 6.2 Test Results at Input 400Vac 50Hz, Output 700V DC

### 6.2.1 Steady State Condition

Steady state current waveform for input Vac 400-Vrms/50H and output 700-V DC are shown in Figure 57 and Figure 58 for different load conditions 960 W and 1865 W, respectively.

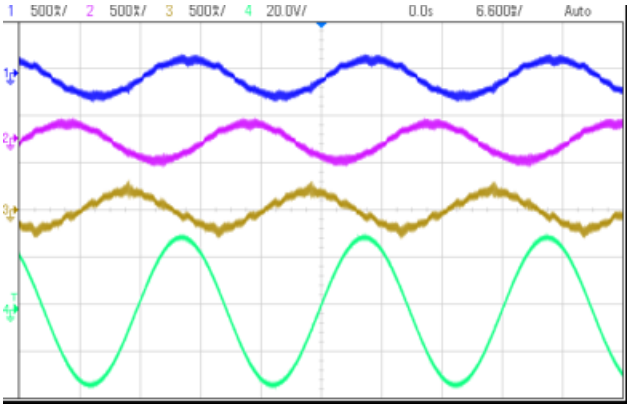



**Figure 57. Steady State 400-Vac IN 50 Hz, 700-V DC OUT 960 W, iTHD 7%**

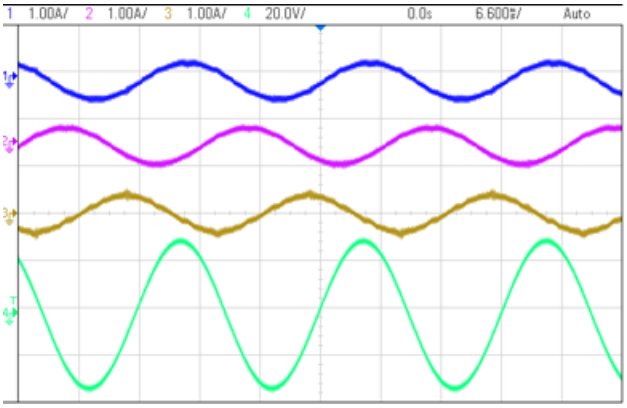



**Figure 58. Steady State 400-Vac IN, 700-V DC OUT 1865 W, iTHD 3.5%**

Table 5 lists the detailed test results of this design under varying load conditions.

**Table 5. Detailed Test Results With 400-Vac IN, 700-V DC OUT, and Varying Power Levels**

| Vbus_PM | Vbus_MN | VoutTotal | Pin | Iout | Pout | Eff | THD% | PF |
|---|---|---|---|---|---|---|---|---|
| 349.41 | 351.2 | 700.61 | 101.9 | 0.137 | 95.98357 | 0.941938862 | 29.10% | 0.7526 |
| 349.41 | 351.9 | 701.31 | 265.3 | 0.37 | 259.4847 | 0.978080286 | 22% | 0.9309 |
| 349.43 | 352 | 701.43 | 581.9 | 0.819 | 574.47117 | 0.987233494 | 11.30% | 0.9823 |
| 349.49 | 351.8 | 701.29 | 900.7 | 1.27 | 890.6383 | 0.988829022 | 7% | 0.9922 |
| 349.46 | 351.9 | 701.36 | 1094.6 | 1.543 | 1082.19848 | 0.988670272 | 6.20% | 0.9982 |
| 349.61 | 351.7 | 701.31 | 1354.1 | 1.913 | 1341.60603 | 0.990773229 | 5.20% | 0.9964 |
| 349.64 | 351.4 | 701.04 | 1553.4 | 2.191 | 1535.97864 | 0.988785014 | 4.70% | 0.9972 |
| 349.75 | 351.2 | 700.95 | 1884.4 | 2.66 | 1864.527 | 0.989453938 | 3.50% | 0.998 |

Figure 52 shows the efficiency data plotted under these test conditions.

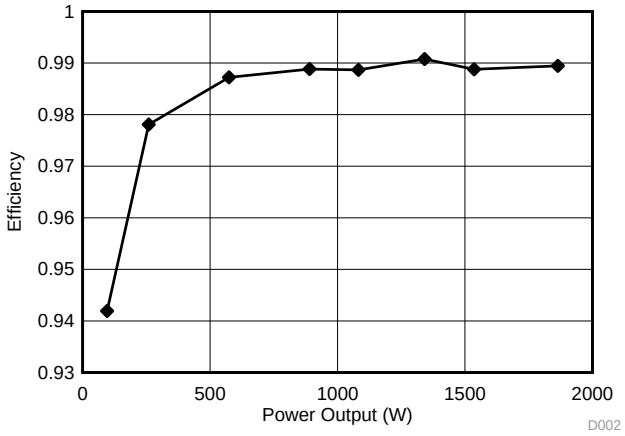



**Figure 59. Efficiency at 400-Vac IN and 700-V DC OUT**

Figure 53 shows the THD data plotted under these test conditions.

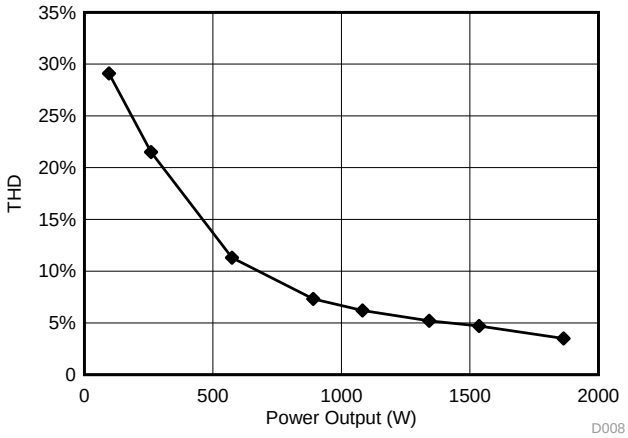



**Figure 60. THD at 400-Vac IN and 700-V DC OUT**

Figure 54 shows the PF data plotted under these test conditions.

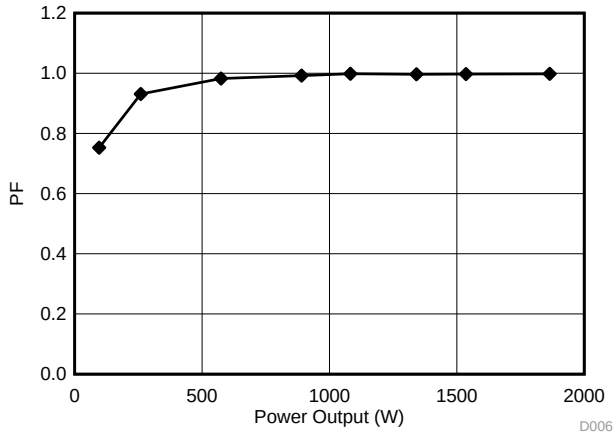



**Figure 61. PF at 400-Vac IN and 700 V DC OUT**

## 7 Design Files

For Schematics, Bill of Material (BOM) , Altium Project, and Gerber files, see the design files at TIDM-1000 or under controlSUITE package at controlSUITE\development_kits\tidm_1000\<version>\hardware\<revision>.

## 8 Software Files

To download the software files for this reference design, please download controlSUITE. Once installed this TI Design can be found at

- controlSUITE\development_kits\tidm_1000\<version>\
  - \docs → Documentation
  - \hardware → PCB Altium Project, Gerbers, BOM, sense_calculation.xlsx
  - \<device>
    - <pfc3phvienna> → CCS Project

.

## 9 Related Documentation

1. Hartmann, M., S.d. Round, H. Ertl, and J.w. Kolar. "Digital Current Controller for a 1 MHz, 10 KW Three-Phase VIENNA Rectifier." *IEEE Transactions on Power Electronics* 24, no. 11 (2009): 2496-508. doi:10.1109/tpel.2009.2031437.

### *9.1 Trademarks*



## 10 About the Author

**MANISH BHARDWAJ** is a Systems Application Engineer with C2000 Microcontrollers System Solutions Group at Texas Instruments, where he is responsible for developing reference design solutions for digital power, motor control, and solar power applications. Before joining TI in 2009, Manish received his Masters of Science in Electrical and Computer Engineering from Georgia Institute of Technology, Atlanta and his Bachelor of Engineering from Netaji Subhash Institute of Technology, University of Delhi, India.

## Revision A History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

| Changes from Original (November 2016) to A Revision | Page |
|---|---|
| • Added For version 1.3 of the F283779D control card, this means SW3 and SW2 are moved to the end with "." that is, to the left, which puts 3.3-V VDDA as the reference for the ADC. | 15 |
| • Changed Figure 20 | 20 |
| • Added If there is no change in the value then make sure the real-time mode is enabled, and the HW is setup correctly. Do not proceed further unless the update is verified. | 23 |
| • Added For example, when Vac is 30 Vrms without switching enabled, the guiVbus will be ~84 V; with switching, the guiVbus will rise up to 140 V. | 24 |
| • Added Make sure all the variables are accurate, that is, guiVrms1/2/3, guiIrms1/2/3, guiPF1/2/3. If any variable is not in line with as shown in Figure 25, it points to a hardware issue with the sensing circuit. | 24 |
| • Changed Figure 27 | 26 |
| • Changed Figure 28 | 27 |
| • Changed 2.7 to 2.0. | 27 |
| • Changed Figure 36 | 33 |
| • Changed Figure 43 | 39 |
| • Added and THD will be around 2.5%. | 42 |

## IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

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